A Shadow Based Method for Image to Model Registration

Alejandro J. Troccoli Peter K. Allen
Department of Computer Science
Columbia University
New York, NY 10027
{atroccol, allen}@cs.columbia.edu

Abstract

This paper presents a novel method for 2D to 3D texture mapping using shadows as cues. This work is part of a larger set of methods that address the entire 3D modeling pipeline to create geometrically and photometrically accurate models using a variety of data sources. The focus is on building models of large outdoor, urban, historic and archaeological sites. We pose registration of 2D images with the 3D model as an optimization problem that uses knowledge of the Sun's position to estimate shadows in a scene, and use the shadows produced as a cue to refine the registration parameters. Results are presented for registration where ground truth is known and also for a large scale model consisting of 14 3D scans and 10 images on a large archaeological site in Sicily.

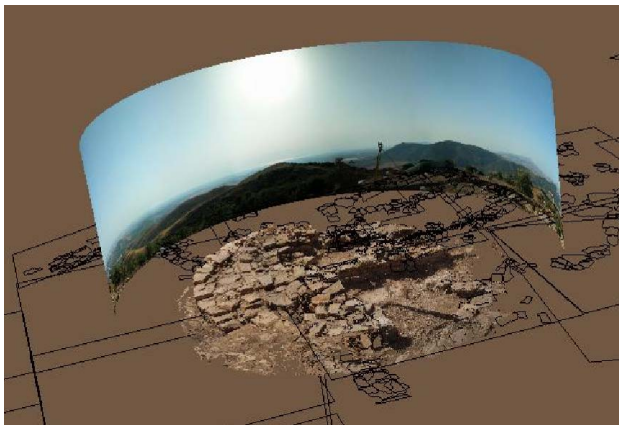

Figure 1. Different data sources can be integrated to provide archaeologist with tools to visualize and document sites. This figure shows a textured 3D model combined with a panoramic image and GIS data.

1. Introduction

The field of 3D reconstruction has been rapidly growing during the last decade as range finders became more accurate, affordable, available, and deployed in a wide variety of applications. In particular, an area that has welcomed the new advances is cultural heritage preservation, where 3D modeling provides a means of recording historic sites and allow wider audiences to virtually see or tour these sites. Unfortunately, the volumes of data are usually of considerable size and the entire modeling pipeline requires significant user interaction. Our work centers on developing new tools and methods to recover complete geometric and photometric models of large sites that minimize human intervention in 3D to 3D registration and 2D to 3D texture mapping of the models with imagery [1].

This paper presents a novel method for 2D to 3D registration using shadows as cues. Shadows have been used in photogrammetry to perform height determination in aerial imagery [8], and are also a valuable source of information for numerous computer vision applications. We use knowledge of the sun's position to estimate the location of shadows in the 3D scene and match these with the shadows in the image to solve the for the camera position and orientation.

We pose the image registration problem in the context of a broader project in which our goal is to accurately document, using range, image, and GIS data, the steps of the archaeological excavation in progress. An excavation is a destructive process, once an object of interest is found, it needs to be removed from the site. 3D modeling using range data is an excellent tool for recording each step of the excavation. Not only does dense range data provide a more complete record than traditional GIS and image data, but it also allows for new ways of visualization. Figure 1 shows a textured 3D model combined with a panoramic image and GIS data. Integrated models such as these allow dynamic user interaction with the model, and provide accurate, in-depth visualizations both on-site and remotely. We present results from the Acropolis at Mt. Polizzo, which is located

0-7695-2158-4/04 $20.00 (C) 2004 IEEE

in the Mediterranean island of Sicily, and hosts the remains of ancient Greek civilizations.

The remainder of our paper is structured as follows. Section 2 gives an overview of the different techniques used for image to model registration in projects similar to ours. Section 3 reviews our scanning methodology and the problems we encountered that lead to development of a new method for 2D to 3D registration based on the shadows cast by the sun; section 4 gives a detail description of our shadow based registration and section 5 presents a detailed performance analysis of our algorithm. We close with a discussion and a description of ongoing and future work.

2. Previous Work

In a typical 3D acquisition pipeline [2], a specialized 3D scanner is used to acquire precise geometry and a digital camera captures appearance information. Unless the 3D scanner provides a calibrated built in camera, the 3D model and images must be registered together in order to connect geometry and texture information. Even if the scanner provides a pre-calibrated built in camera, this might not be suitable for the application at hand, in which case it will be necessary to acquire photographs using a different camera and register the images to the model.

This problem of image to model registration is closely related to the problem of camera calibration, which finds a mapping between the 3D world (object space) and a 2D image. This mapping is characterized by a rigid transformation and a camera model, also referred to as the camera's extrinsic and intrinsic parameters. The rigid body transformation takes 3D points from object space to 3D points in the camera's reference frame, and the camera model describes how these are projected onto the image plane.

The camera calibration problem is solved by matching features in the 3D model with features in the image. These features are usually points, lines or special designed objects that are placed in the scene. The matching process can be automatic or user driven, and the number of feature pairs required will depend on whether we are solving for the intrinsic, extrinsic or both parameters sets. Several methods have been developed to solve the camera calibration problem using grid patterns (e.g. [16], [4]).

In the area of image registration for 3D modeling using dense 3D data, several approaches can be taken, depending on factors such as the type of range scanner being used (laser-stripe vs. time of flight), the scale of the model to be built and the environment conditions. For example, some scanning configurations allow texture and geometry to be acquired by the same camera as in the works of Pulli et al. [11], and Bernardini et al. [3], and hence there is no need for calibration. Such a case is usually limited to laser stripe range finders, which can only cover short ranges and have a restricted set of working environment conditions.

If the images are to be acquired by a camera that is not built in to the range scanner, then it is possible to pre-calibrate before deployment. Such a system was developed for the Digital Michelangelo project of Levoy et al. [10]. Pre-calibration, when possible, has the advantage of avoiding any user driven post-processing. However, one must be certain that the calibration remains constant during the whole scanning process to avoid unexpected errors, or provide a re-calibration mechanism. Pre-calibration works well for objects that are scanned at short distances because a good pixel to area ratio is then guaranteed.

When pre-calibration is not possible, then each image will have to be registered. A typical scenario of small scale object modeling where each image is individually registered is described by Rocchini et al. [13]. Here, image to model registration is done manually by a user who selects corresponding pairs of points. In a similar context of small object modeling, Lensch et al. [9] present an automatic method for image registration based on silhouette matching, where the contour of a rendered version of the object is matched against the silhouette of the object in the image. No user intervention is required, but their method is limited to cases where a single image completely captures the object.

Departing from the above are those methods specifically developed for large scale modeling. Our methods fall in this area. Not only is the scale different in our case, but we are also dealing with outdoor scenes, where the environment conditions are not easily controlled. This usually rules out the possibility of using triangulation based range finders. Instead, time of flight laser scanners are typically used. To capture texture information, a camera can be fixed to the scanner and calibrated. Still in some cases, when the scanner is reaching objects that are more than 50m away, it might be necessary to capture additional images closer to the scene to obtain a better pixel to area resolution. In [15] we present an automatic method for image to model registration of urban scenes, where 3D lines are extracted from the point clouds of buildings and matched against edges extracted from the images. Ikeuchi et al. [6] in their Great Buddha work, use reflectance edges obtained from the 3D points and match them against edges in the image to obtain the camera position.

3. Modeling the Acropolis at Mt. Polizzo

Geometry, in the form of point clouds, is acquired by a laser range finder, and texture, in the form of photographs, is obtained with a digital camera. Details for each stage of our modeling pipeline are given next.

Scan acquisition. To model the acropolis at Monte Polizzo we used a time-of-flight laser scanner (Cyrax 2500) to measure the distance to points on the site. Data from the

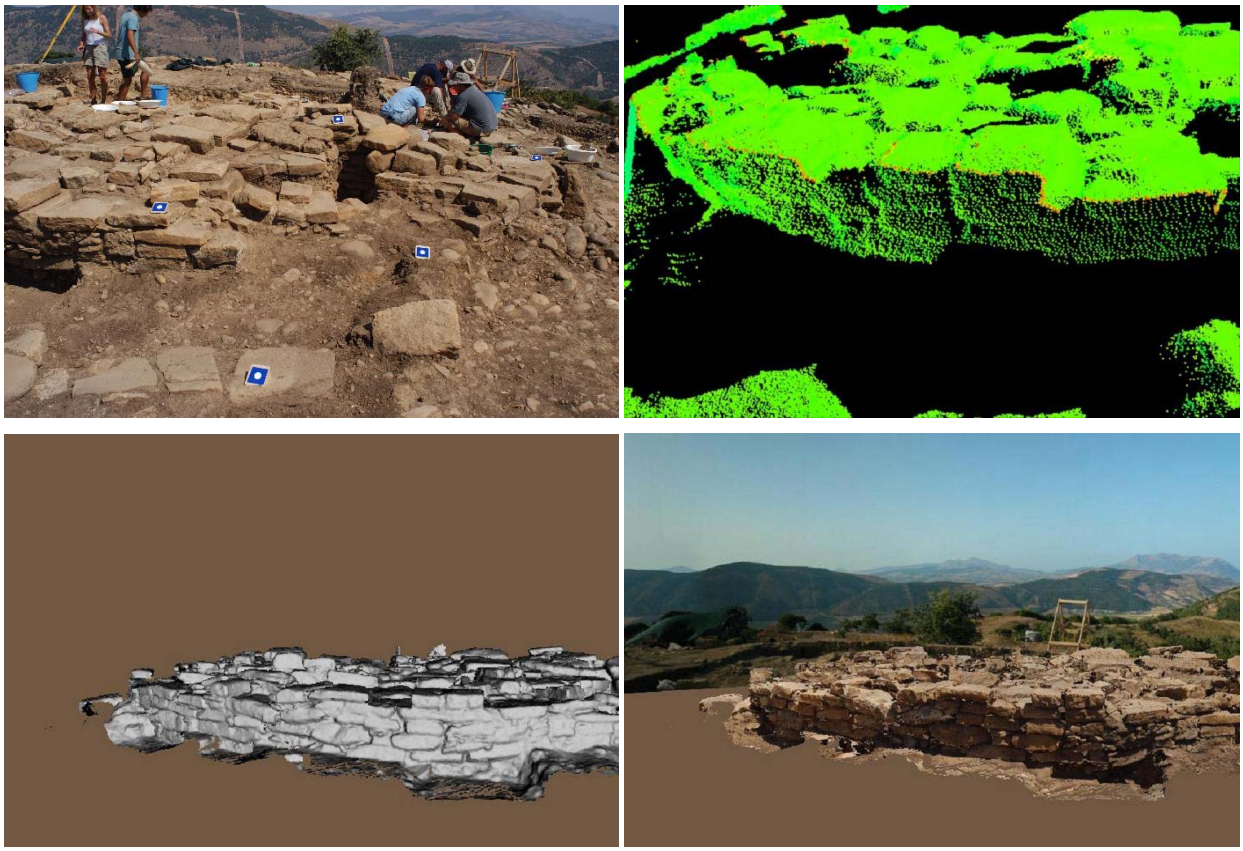

Figure 2. Modeling the Acropolis at Mt. Polizzo. Top left: Targets have been placed in the area to be scanned. Top right: Point cloud of scanned area. Bottom left: Mesh resulting from merging all point clouds. Bottom right: Textured mesh with panorama as background.

scanner comprises point clouds, with each point consisting of three coordinates (x,y,z) and a value representing the amplitude of the laser light reflected back to the scanner.

Scan registration. Multiple scans are required to completely acquire a site such as the Monte Polizzo acropolis. The resulting point clouds need to be registered together. Typically, the coordinate system of one of the point clouds is chosen as the coordinate system for the model. In archaeology, however, a global site coordinate system is set up from GPS data. A set of control points are accurately measured using (preferably differential) GPS, and are then used to initialize a theodolite in the world coordinate system. The theodolite is used to measure points of interests, such as the location of findings, rocks, or the terrain contour. We need to determine the registration of each point cloud with respect to the site's coordinate system. We solve this problem by using a set of targets that the scanner can automatically recognize, shown in Figure 2. Before taking a scan, we place the targets on the area we plan to cover, and use the theodolite to measure their positions in the site's coordinate system. Afterwards, we scan the scene at a low resolution to identify and acquire the targets' positions in the scanner's coordinate system, and so solve the 3D-to-3D registration problem. The targets are then removed and a full-resolution scan is acquired.

Surface generation. From sets of registered point clouds that represent the state of the site at the same point in time, we generated a triangular-mesh surface, using the VripPack software developed by Curless and Levoy [5]. VripPack outputs the best mesh that fits the point cloud data, smoothed to account for registration errors.

Texture acquisition. In addition to the scanner, we used a Nikon D100 digital camera, mounted on the scanner's case, to acquire texture information. For each scan we acquired, we took a photograph.

Local texture registration. We performed a simple calibration prior to our trip to estimate the camera's external and internal parameters. We determined the camera calibration by scanning a flat wall with the room lights off and the camera's shutter open for the eight-second duration of the scan. This provided us with an image of the grid pattern described by the laser as it sampled the wall and the 3D coordinates of each sample. We repeated the scanning from a different distance and angle to acquire more sam-

ples. We then segmented the images to obtain the centroid of each grid point, and solved the calibration problem using the 2D-to-3D correspondences just obtained.

Global texture registration. While the local texture calibration procedure provided us with a good estimate of the camera's parameters, we found that our images were slightly misaligned with respect to the complete model. One reason for this is that our calibration was local to the scanner's coordinate system. To texture-map the final model, this local registration had to be transformed to the site's coordinates. Hence, any errors in scan-to-scan registration will also affect the texture registration. In addition, our initial calibration was accurate at the depths at which calibration points had been measured, but not as accurate at other ranges. To solve these misalignments, we developed a new method based on the shadows cast by the sun. Our method performs a global texture registration; it registers the texture with respect to the model's coordinate system, as opposed to the scanner's coordinate system. Since we have the latitude and longitude of the site and the time at which each photograph was taken, we can compute the location of the sun and find portions of the 3D model that should be in shadow. By matching these with the shadows in the image we solve the 2D to 3D registration problem.

Texture-map generation. The final texture mapped model was created using a software package we developed, which, given a mesh and a set of calibrated images, assigns each mesh triangle a suitable texture. Our software can also integrate GIS data and panoramic images, as shown in Figure 1.

4. Using shadows for 2D to 3D registration

The main contribution of this paper is a novel technique for image to model registration based on the shadows cast by the sun. Assuming the internal parameters of a camera are known, we find the camera position with respect to the 3D model:

$$\mathbf{c} = (\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)$$

This is a 6 parameter rigid body transform that maps a point X_w in world coordinates into its corresponding point X_c in the camera reference frame. The first three parameters (Euler angles) represent the angles of rotation about each of the coordinate axes and form a rotation matrix $\boldsymbol{R}(\phi_x, \phi_y, \phi_z) = \boldsymbol{R_x}(\phi_x)\boldsymbol{R_y}(\phi_y)\boldsymbol{R_z}(\phi_z)$. The remaining three are the components of a translation vector $\mathbf{t}$. Together, they satisfy the following relationship:

$$X_c = \boldsymbol{R}(\phi_x, \phi_y, \phi_z)X_w + \mathbf{t}$$

If we knew the correct set of external camera parameters $(\phi_{x_f}, \phi_{y_f}, \phi_{z_f}, t_{x_f}, t_{y_f}, t_{z_f})$ then an orthographic rendering of a textured version of the model with the eye set at some point E_s above the ground and looking in the direction of the sun rays should show no texture representing shadows (see Figure 3). However, if the texture is misaligned such a rendering will show a number of shadow pixels. Our method exploits this idea by searching the parameter space of camera positions for a point that minimizes the number of pixels representing shadows in the rendered image of the model.

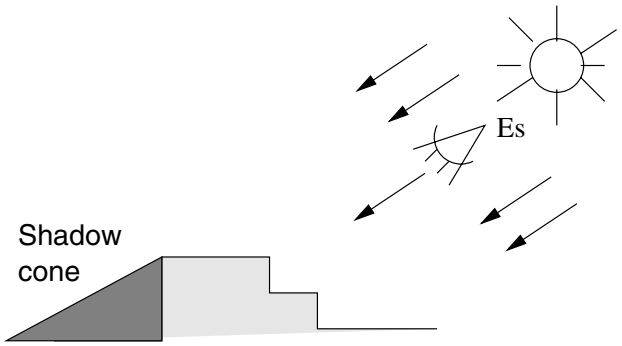

Figure 3. An orthographic view from the point E_s looking at the model in the direction of the sun rays should contain no shadows.

The problem is properly stated as follows. Let I denote the image to be registered and M the model, then f, our cost function, is defined as

$$f(I_r) = \sum_{x,y \in I_r} shadow(I_r, x, y)$$

where I_r stands for a *rendered image of M as seen from E_s textured with I using a texture camera with external parameters set to* $\mathbf{c}$ and

$$shadow(I_r, x, y) = \begin{cases} 1 & \text{if pixel}(x,y) \text{ of } I_r \\ & \text{is a shadow pixel} \\ 0 & \text{otherwise.} \end{cases}$$

Given an initial estimate of the camera position $\mathbf{c}_0$, then the problem is to find a point $\mathbf{c}_f$ that minimizes f. In our case, we already have the initial estimate. But if we wanted to generalize this method to cases where images are acquired by a camera whose position is unrestricted, a user could provide an initial estimate before starting the minimization.

The complete registration pipeline consists of 2 stages: a pre-processing stage and a minimization stage. In the pre-processing stage the shadows in the image are found and masked out with a given color. In the minimization stage, an optimization algorithm searches for a global minimum of f starting from the initial estimate.

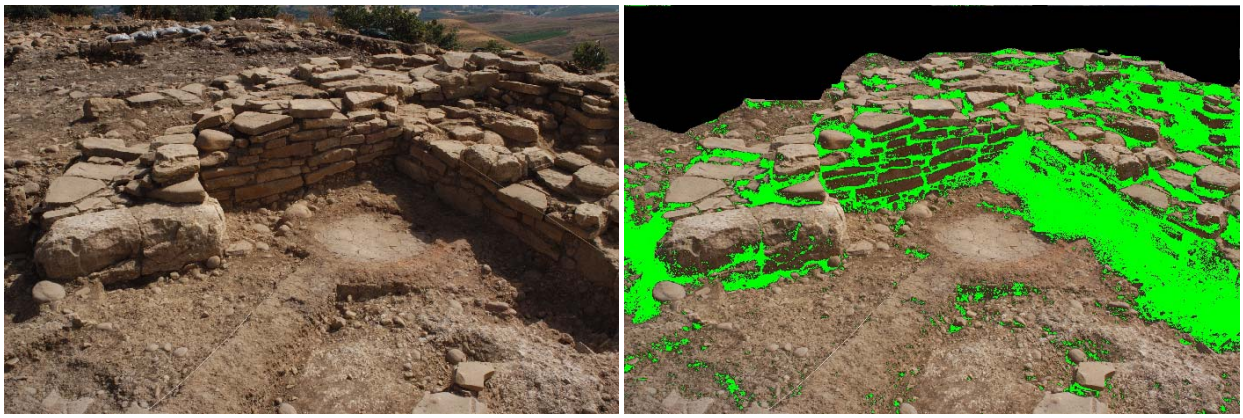

Figure 4. Left: Image of the Acropolis at Monte Polizzo. Right: The same image after shadow pixels have been masked in green

4.1. Preprocessing

The computational cost of evaluating f depends on the ease of counting shadow pixels in the rendered scene. A simple evaluation algorithm for f is possible if all the pixels in I that represent regions in shadow are first masked out with a known color (see Figure 4). Then, the evaluation of f reduces to counting these masked pixels in I_r. We find the shadows in I by selecting those pixels whose gray value lies below a threshold. Our system suggests a threshold based on the histogram of the image, which the user can adjust if necessary.

In addition, before starting the minimization, the position of the viewpoint point E_s is computed. From the timestamp of the image and the latitude and longitude coordinates of the surveyed area we compute the elevation and azimuth (θ, ϕ) coordinates of the sun on the sky [12]. Then the eye point is defined as,

$$E_s = P_c + V(\theta, \phi) * d$$

where P_c is the point where the optical axis of the camera's initial estimate as defined by $\mathbf{c_0}$ intersects the model, $V(\theta, \phi)$ is a 3D vector on the unit sphere that corresponds to the elevation θ and azimuth ϕ, and d is the distance from the eye to the model, which is interactively selected by the user in such a way that the model is seen at a reasonable size.

4.2. Minimization

For the minimization stage we use an iterative nonlinear minimizer. The cost function f is not only non-linear, but also highly dependent on the geometry of the model M. In addition, it has no analytical derivatives and it usually contains several local minima, which makes the search process more difficult.

We use simulated annealing [7] to drive the minimization process. Simulated annealing has the advantage of avoiding local minima by randomly sampling the parameter space and occasionally accepting parameters that drive the search uphill to a point of higher cost, as opposed to gradient descent methods that only allow downhill steps.

There are several other minimization algorithms for non-linear functions, some of which require partial derivatives. For our cost function, these derivatives could be computed using finite differences at the expense of extra function evaluations per iteration. But these still can converge to a local minimum if the initial estimate is not good enough.

4.3. Cost function evaluation

On each iteration k of the non-linear optimization a set of camera parameters $\mathbf{c}_k$ are provided by the minimization algorithm and the cost function is evaluated.

One might be tempted to simply render the model with texture mapping enabled and count the number of shadow pixels in the rendered image. However, not every scene point is visible from the texture camera, a fact which must be accounted for in order to avoid incorrect results. It is then necessary to compute on each iteration a visibility map that will define which scene points are visible from the texture camera. Fortunately, today's graphics hardware allows us to compute this visibility map in real-time by following an approach similar to that of shadow mapping [14].

Function evaluation is a two pass rendering process, consisting of the following steps

1. Visibility computation (rendering pass 1)
   (a) Set eye to camera position and orientation $\mathbf{c}_k$
   (b) Set the projection model as specified by the camera's internal parameters
   (c) Render the model using depth buffering

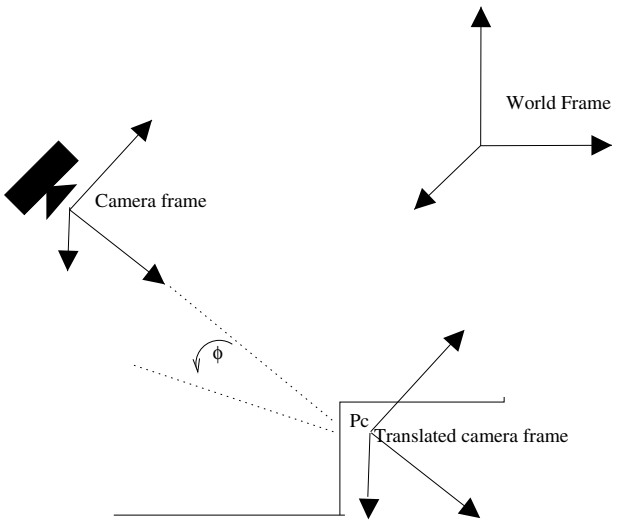

Figure 5. Cost function reparametrization. The camera frame is translated to the point where the initial camera's optical axis intersects the scene. Rotations and translations are described wrt to this translated camera frame.

(d) Store the depth values into a texture V. This will be our visibility map.

2. Model rendering with I as texture (rendering pass 2)

   (a) Set eye to E_s, the position of the sun.

   (b) Set an orthographic projection.

   (c) Set V and I as textures. Use V to decide which areas are visible from the camera.

   (d) Render the scene.

   (e) Capture the frame buffer and store it in I_r

3. Compute cost. Count the number of shadow pixels in I_r

4.4. Cost function reparametrization

The optimization search has been described over a vector of 6 parameters $(\phi_x, \phi_y, \phi_z, t_x, t_y, t_z)$. This is not a suitable parametrization for a minimization search for several reasons. First, the parameters are interdependent. Rotations parametrized using Euler angles can suffer from singularities (e.g. Gimbal lock). Also, the use of a translation vector adds a dependency between the rotation parameters and the position of the camera. And as cited by Lensch et al. [9], this parametrization has the disadvantage that a small rotation around the camera results in a large displacement in camera coordinates of distant points.

Instead of using Euler angles to represent a rotation, we use a rotation axis $\mathbf{q}$ and rotation angle ω, just as quaternions do. We represent the rotation axis by its spherical coordinates (θ, ϕ) on the unit sphere, and then a complete rotation is described by the three parameters (θ, ϕ, ω). We also change the center of rotation. Instead of rotating about the origin of the world coordinate frame, we will rotate around the point P_c (see Figure 5).

To establish the position of the camera, we use a displacement vector d which describes the position of the camera in the translated coordinate system. The reparametrized camera is then described by,

$$\mathbf{c} = (\theta, \phi, \omega, d_x, d_y, d_z)$$

4.5. Cost function optimization

Even after reparametrization, the search process might place the camera in a configuration where it does not see any point in the model. In such a case, the value of f will be 0 and the minimization will converge to an incorrect result.

To avoid these situations, one can to take into account the number of textured pixels in I_r. This number is proportional to the number of scene points that are visible from the texture camera and it is to be expected that the correct camera configuration will cover a large area of the scene. We can compute the number of textured pixels at the same time we count the shadow pixels if we set the screen background to a given color (e.g. black) and use this color to render scene points that are not visible from the texture camera. This adds no computational cost to the evaluation of f.

There are several ways in which this information can be used. We define a new cost function f_1 as

$$f_1(I_r) = \begin{cases} \frac{f(I_r)}{\text{number of pixels in } I_r} & \text{if } v(I_r) \geq k_v.visible_0 \\ 1.0 & \text{otherwise.} \end{cases}$$

where

$$v(I_r) = \sum_{x,y \in I_r} visible(I_r, x, y)$$

and

$$visible(I_r, x, y) = \begin{cases} 1 & \text{if pixel}(x,y) \text{ of } I_r \\ & \text{is a non-black pixel} \\ 0 & \text{otherwise.} \end{cases}$$

Here $f(I_r)$ is the count of shadow pixels, $v(I_r)$ is the count of textured, k_v is a threshold value and $visible_0$ is the number of visible pixels at the initial configuration. This form has the advantage of favoring camera configurations with a larger coverage of the scene.

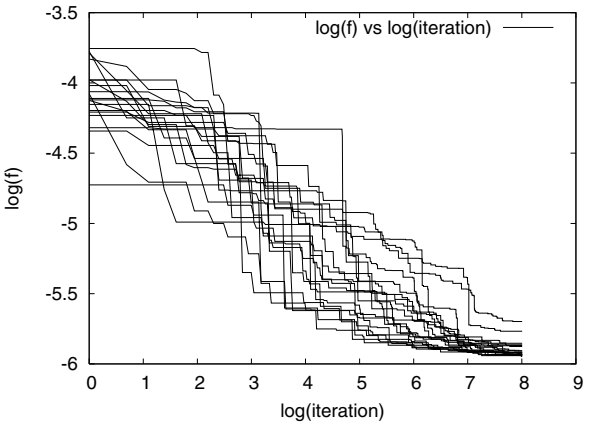

Figure 6. This figure shows a plot of the $log(f_1)$ against the log of the number of minimization iterations. The 20 series correspond to 20 simulation iterations.

5. Results

To test the performance of our algorithm we ran a set of simulation experiments for which we knew the camera calibration. The simulations were performed on a mesh obtained from a single scan, thus avoiding mesh averaging errors caused by merging multiple scans. This mesh consisted of 20367 vertices and 34807 triangles. The size of the texture image was of 3008 by 2000 pixels. An accurate calibration was achieved by placing targets on the scene and acquiring their position with the 3D scanner and a geo-referenced total station. Using this camera calibration as ground truth, we created a sequence of initial positions by randomly perturbing the orientation of the camera by as much as -5 to 5 degrees in each rotation angle and the translation by -0.25 to 0.25m in each axis. From each of these initial positions, we ran our algorithm using cost function f_1 with k_v set to 0.6 and then compared the resulting camera with the original one.

Table 1 shows the results of 20 simulation iterations. To accurately compare the camera obtained from our algorithm with the control camera, we project the mesh vertices to the image plane of each camera and define the reprojection error as the distance between these two points. Columns two and three on Table 1 list the average reprojection error and columns four and five the number of shadow pixels at the start and end of the minimization. Notice that the average reprojection error over the 20 iterations is 7 pixels, which is acceptable. In the worst case, the average reprojection was 15 pixels, which is still good. The running time for each iteration was approximately 12 minutes on a Pentium IV machine. This time corresponds to 3000 iterations of our minimization. Simulated annealing is not fast, but in our case provides accurate results in the presence of a large

Table 1. Simulation results

	Avg. reprojection error		Shadow pixels	
Iteration	Initial	End	Initial	End
0	661.74	5.63	3405	531
1	823.03	15.05	2945	619
2	711.32	10.43	3699	663
3	639.82	6.15	4486	530
4	707.36	6.14	2876	522
5	611.09	6.93	3699	523
6	711.28	6.87	3200	524
7	817.58	6.81	3109	568
8	606.78	7.57	3556	522
9	737.70	4.28	4297	538
10	718.43	6.31	3211	533
11	626.94	8.50	2573	562
12	742.79	6.29	2635	517
13	688.23	10.65	3244	559
14	638.68	6.05	3354	522
15	572.92	6.24	3708	555
16	699.08	5.74	1754	531
17	743.26	5.83	2979	527
18	827.53	9.06	4532	538
19	795.37	6.18	4628	520
avg	704.05	7.34	3394	545

Table 2. Realignment results

	Shadow pixels		
Image	Initial	End	Improvement %
1	3971	1535	61.34%
2	4641	2454	47.12%
3	4140	1926	53.48%
4	811	705	13.07%
5	3760	770	79.52%
6	2171	308	85.81%
7	2573	1382	46.29%
8	2838	1329	53.17%
9	1081	356	67.07%
10	4528	2975	34.30%
avg	3051	1374	54.12%

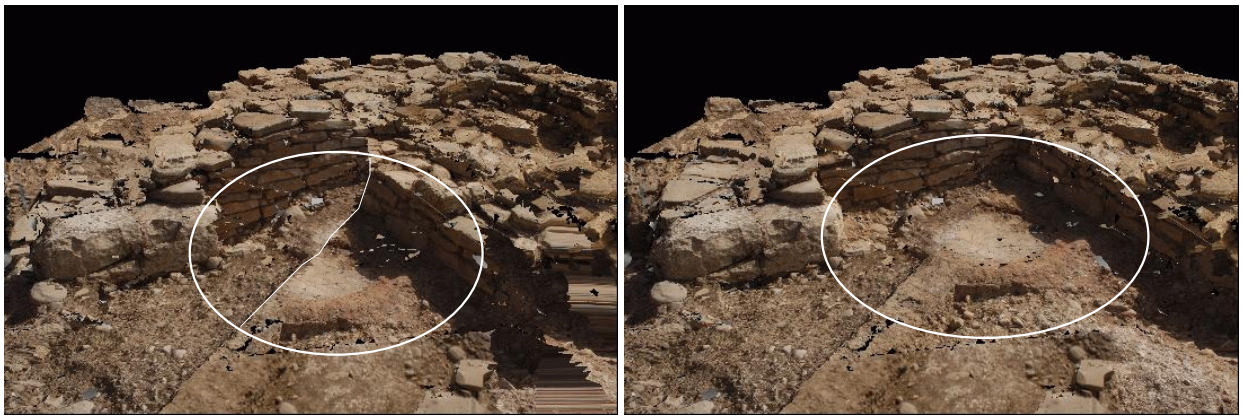

Figure 7. Acropolis 1. Left: An initial configuration as generated during our simulation experiments. Note that the image does not align well with neighboring textures. In particular, the area enclosed within the ellipse is misaligned. A white line has been drawn over the border between separating the textures. Right: Result obtained after running our algorithm.

Figure 8. Realignment of the textures of our Acropolis model. Left: The textures are misaligned and artifacts can be observed (enclosed within white ellipse). Right: View after the textures have been re-aligned using our algorithm.

number of local minima. The running time depends on two main factors: the number of iterations and the size of the rendering window. A maximum of 3000 iterations might be excessive as Figure 6 suggests. This figure shows the log of the cost of the the best configuration found so far against the log of the number of minimization iterations for each of the 20 simulations. The reason for choosing a logarithmic scale is because simulated annealing has an exponential temperature (cooling) schedule. This type of plot is useful to estimate the number of iterations required for the minimization to converge, which in this case is between e^7 and e^8 or 1096 and 2980 iterations, closer to the 1096 end.

Figure 7 shows two rendered models of the Acropolis at Mt. Polizzo, the first one setting the texture camera to the initial position of one of the simulation iterations and the second one with the same camera set to the results of our algorithm. Notice how the region within the ellipse, that is misaligned and does not agree with its neighbor texture is correctly aligned after running our algorithm.

In addition, we used our registration method to realign the images of our Acropolis model (Figure 8). The model consists of a mesh of 138,000 triangles and is textured using twelve 3008 x 2000 pixel images. We were able to successfully re-align 10 of the 12 images. In one case, due to holes in the acquired scan our algorithm failed to find the intersection of the camera's optical axis and the scene. In the other case, the algorithm failed because the texture image was taken late in the afternoon, close to sun set , and areas which were not in shadow were incorrectly masked as shadow areas during the shadow detection phase. Table 2 shows the count of shadow pixels at the start and end of our algorithm and the percentage improvement.

6. Discussion

In this paper we have presented a complete pipeline for building a a photorealistic 3D model of an archaeological site, with emphasis on a novel method for 2D to 3D registration based on the shadows cast by the sun. We showed how we successfully applied the proposed algorithm to solve for texture misalignments. There are several areas in which our method can be improved. One of them is shadow detection in the images. This is important because misidentified shadow pixels can drive the search to an incorrect result. We are also looking at different ways of improving the running speed. The speed is highly dependent on the size of the rendered image I_r so we are experimenting with different window sizes and evaluating how the speed and accuracy are affected.

Our shadow based method can be used in different contexts. One of them is robot localization: a robot could use a camera to take an image and then use our shadow based algorithm to find its position.

Finally, this is just a piece of the on going research in the area of 3D modeling. We are beginning to look at further ways of improving the final textured model. One problem that we are looking at is simultaneous texture alignment. The algorithm presented in this paper takes a single texture image at a time and aligns it with respect to the 3D model. But it does not take into account how well this alignment corresponds with previously aligned texture images. Another problem we need to address is the color constancy problem: the colors of overlapping images do not match. This is due to the images being taken under different times of the day and different illumination conditions.

Acknowledgments

In addition to the paper authors, the field team that went to Mt. Polizzo was also integrated by Benjamin Smith, Hrvoje Benko, Edward Ishak, Steve Feiner, Lynn Meskel and James Conlon. We would link to thank them for their invaluable effort and suggestions. This research was funded in part by NSF grant IIS-0121239 and from gifts by Alias Systems. Also, additional thanks go to Ian Morris and the Stanford Archaeology Center for images and data from the Monte Polizzo excavation.

References

[1] P. K. Allen, A. Troccoli, B. Smith, S. Murray, I. Stamos, and M. Leordeanu. New methods for digital modeling of historic sites. *IEEE Comput. Graph. Appl.*, 23(6):32–41, 2003.

[2] F. Bernardini and H. Rushmeier. The 3D model acquisition pipeline. *Computer Graphics Forum*, 21(2):149–172, June 2002.

[3] F. Bernardini, H. Rushmeier, I. M. Martin, J. Mittleman, and G. Taubin. Building a digital model of Michelangelo's Florentine Pietà. *IEEE Computer Graphics and Applications*, 22(1):59–67, /2002.

[4] J.-Y. Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguet/calib_doc, 2001.

[5] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM Press, 1996.

[6] K. Ikeuchi, A. Nakazawa, K. Nishino, and T. Oishi. Creating virtual buddha statues through observation. In *IEEE Workshop on Applications of Computer Vision in Architecture*, volume 1 of *Conference on Computer Vision and Pattern Recognition*, 2003.

[7] L. Ingber. Very fast simulated re-annealing. *Mathl. Comput. Modelling*, 12(8):967–973, 1989.

[8] R. B. Irvin and J. David M. McKeown. Methods for exploiting the relationshio between buildings and their shadows in aerial imagery. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1564–1575, December 1989.

[9] H. P. Lensch, W. Heidrich, and H.-P. Seidel. A silhouette-based algorithm for texture registration and stitching. *Graphical Models*, 63(4):245–262, 2001.

[10] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3D scanning of large statues. In *Siggraph 2000, Computer Graphics Proceedings*, pages 131–144, 2000.

[11] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle. View-based rendering: Visualizing real objects from scanned range and color data. In *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 23–34, New York, NY, 1997. Springer Wien.

[12] I. Reda and A. Andreas. Solar position algorithm for solar radiation applications. Technical report, National Renewable Enery Laboratory, Golden, Colorado, June 2003.

[13] C. Rocchini, P. Cignomi, C. Montani, and R. Scopigno. Multiple textures stitching and blending on 3D objects. In *Rendering Techniques '99*, Eurographics, pages 119–130. Springer-Verlag Wien New York, 1999.

[14] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadows and lighting effects using texture mapping. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 249–252. ACM Press, 1992.

[15] I. Stamos and P. K. Allen. Automatic registration of 2-D with 3-D imagery in urban environments. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, pages 731–737, Los Alamitos, CA, July 9–12 2001. IEEE Computer Society.

[16] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, 3:323–344, 1987.