Machine Vision and Applications (2007) 18:65–72
DOI 10.1007/s00138-006-0048-9

ORIGINAL PAPER

Shadow based texture registration for 3D modeling of outdoor scenes

Alejandro Troccoli · Peter Allen

Received: 6 April 2006 / Accepted: 6 September 2006 / Published online: 21 November 2006

Abstract This paper presents a novel method for the registration of texture images with a 3D model of outdoor scenes. We pose image registration 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 solve for the registration parameters. Results are presented on a controlled experiment and for a large scale model of an archaeological site in Sicily.

Keywords 3D modeling · Texture registration · Shadows

1 Introduction

This paper presents a method for finding the pose of a 2D camera with respect to an acquired 3D model of an outdoor scene where shadows are present. We developed this method as a part of a larger modeling and visualization system for archaeological site documentation [1]. Our goal is to construct a photo-realistic and geometrically accurate 3D model by combining geometric measurements obtained from a range sensor with photographic observations taken with a freely moving 2D camera. In this context, we solve the pose estimation problem (with known camera intrinsics) by matching the position of the shadows in the 2D image with the position of the shadows in the 3D model and solving for the location of the camera in the 3D world.

Automatic pose estimation is a difficult problem which combines 3D and 2D feature extraction with feature matching. Most systems that do automatic or semi-automatic pose estimation use corners, lines or silhouettes [10,14] as features; but these are hard to come by in an archaeological site. By using the shadows we can overcome this inherent lack of traditional geometric features. In fact, shadows have been used in many computer vision applications. For example, shadows can reveal information about scene structure [7,12,13,24] or light sources [21]. In our case, both scene structure and light source position are known; we take advantage of this information and use it to compute the camera position.

Given that our technique is based on shadow detection and matching, the following pre-conditions apply: shadows should be detectable in the image; the 3D model's geolocation (latitude, longitude and orientation with respect to North) must be known; and objects casting shadows should be present in the model. With the exception of the first condition, which does not hold on cloudy days, the other two assumptions are typically met in archaeological excavations and other 3D outdoor modeling applications. On the other hand, the requirement that shadows are present in the images might not always be met. This does by no means invalidate our method; it makes it one more tool available in our 3D modeling system. For images with shadows, we use the tool; for images without shadows, we fall back to a user-based manual registration tool. The difference between the two tools is the amount of user interaction. Where the manual point and click registration tool is very user intensive, the shadow-based method only

A. Troccoli · P. Allen (✉)
Department of Computer Science, Columbia University,
1214 Amsterdam Ave. MC 401,
New York, NY 10027, USA
e-mail: atroccoli@acm.org

P. Allen
e-mail: allen@cs.columbia.edu

requires minimum user interaction. We have applied the method we present in this paper to a Stanford University archeological excavation at the acropolis of Mt. Polizzo, Sicily.

2 Background

In a typical 3D model acquisition pipeline [3], a 3D scanner is used to acquire precise geometry and a digital camera to capture texture information. Unless the 3D scanner has a calibrated built in camera, the 3D model and images are unconnected and must be registered. Texture registration is a camera calibration task which solves for a mapping between a 3D world coordinate system and a 2D image plane. This mapping consists of a rigid transformation that takes the 3D points from the world coordinate system to the camera coordinate frame, plus a projection model that maps these into the image plane. The calibration is obtained by matching features in the 3D model with features in the image.

Previous work on 3D modeling from range and intensity images has approached the camera pose problem in different ways. Some triangulation based scanners can acquire both texture and geometry with the same camera [4,17], hence not requiring a registration step. Some other systems rigidly fix the camera to the 3D scanner and run a camera calibration procedure only once, in a laboratory setting with objects of known geometry, as [15]. However, such systems constrain texture gathering both in time and space, because the images must be acquired at the same time the 3D scans are taken with the position of the camera restricted to the location of the scanner. In certain cases, it is desirable to be able to acquire the images with a free-moving camera; either to get a close shot, for example, or to re-shoot poorly lit scenes. In this scenario, all images must be individually registered with the 3D model. This registration could be done manually, by having a user select corresponding points from the 3D model and the 2D image as in [19], or (semi)-automatically, using a system that finds correspondences. Lensch et al. [14] present an automatic method for texture 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. In urban modeling, Stamos and Allen [23] presented an automatic texture registration algorithm method that uses line features. 3D lines are extracted from the point clouds of buildings and matched using a RANSAC-like approach against edges extracted from the images. More recently, Liu and Stamos [16] extended the use of 3D and 2D line features to develop an algorithm that groups the lines into higher order features (3D parallelepipeds and 2D rectangles) to efficiently search the space of 3D and 2D feature matches. Ikeuchi et al. [10] use reflectance edges obtained from the 3D point cloud and match them against edges in the image to obtain the camera position, which is computed using a robust non-linear minimization once a user selects an initial estimation.

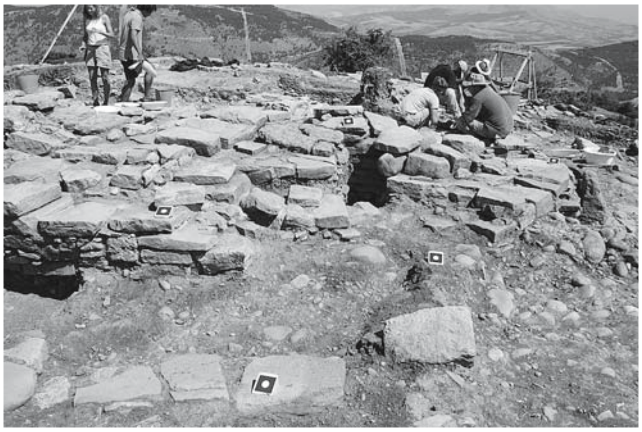
Fig. 1 A view of the archaeological excavation

3 Model acquisition

We build the 3D model by acquiring a set of range and intensity images to cover the entire region of interest within the archaeological site. Our range sensor is a Cyrax 2500 time-of-flight laser scanner, which can gather 1 million points within a field of view of 40 by 40 degrees. Each point measurement consists of its 3D Cartesian coordinates (x, y, z) in the scanner's local coordinate system and a fourth value representing the amplitude of the laser light reflected back to the scanner. At each scanning position we also acquire a photograph by placing the camera in close proximity to the scanner. Thus, the complete acquisition process interleaves a 3D sensing with 2D sensing.

The range scans are registered using fiduciary markers that are placed in the scene before scanning, and that are optically designed to be recognized by the scanner, shown in Fig. 1. The scanner measures the position of each marker in its local coordinate system. In addition, we also measure the coordinates of each marker in the site's coordinate system using a total station device that was initialized from a set of geo-referenced control points. From the these point-clouds we build a triangular mesh using the VripPack package developed by Curless and Levoy [6].

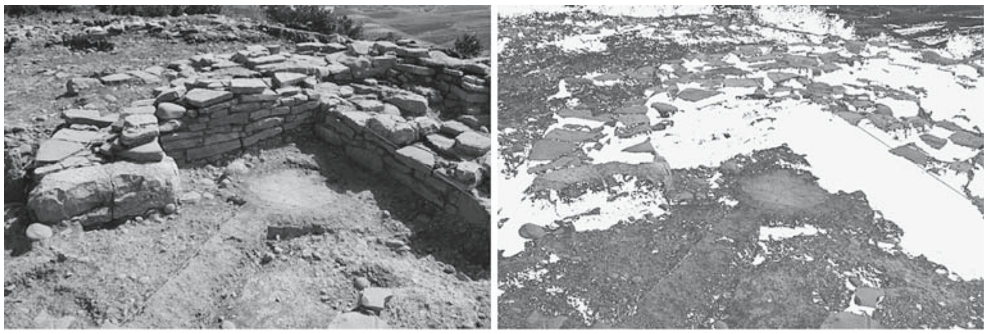

Fig. 2 *Left* Image of the Acropolis at Monte Polizzo. *Right* The same image after shadow pixels have been masked (shown here in *white*)

4 The 3D to 2D registration algorithm

Consider an ideal pinhole camera C with a 3×4 projection matrix $\boldsymbol{P}$, $\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\mathbf{t}]$, where $\boldsymbol{K}$ is a 3×3 camera matrix (see [9] page 143), $\boldsymbol{R}$ is a rotation matrix that describes the orientation of the camera and $\mathbf{t}$ is a translation vector that indicates the camera position in the world. The camera matrix $\boldsymbol{K}$ is pre-computed using a camera calibration package such as [5]; the unknowns are $\boldsymbol{R}$ and $\mathbf{t}$. If we knew the correct parameters $(\boldsymbol{R_f}, \mathbf{t_f})$ for the camera position and orientation, then an orthographic rendering of a textured version of the model viewed from the position of the sun should show no shadow pixels. However, if the texture is misaligned, shadow pixels will be visible. Using this idea, we frame our solution as an optimization problem. Given an initial camera position $(\boldsymbol{R_0}, \mathbf{t_0})$, we search the parameter space of Euclidean transformations in the vicinity of this initial configuration for a point that minimizes a cost function whose value is proportional to the number of visible shadow pixels in the rendered model. We set the initial position of the camera to the location of the scanner in its corresponding range scan.

4.1 Shadow detection in the image

As a first step in our shadow based algorithm, shadows in the input image are masked with a pre-defined color. We detect the shadow regions in the image using global thresholding on the luminance channel. The threshold selection is done by a user. It is known that accurate detection of cast shadows and their boundaries is a difficult task due to the complex effects of penumbrae and inter-reflections. While this is generally true, shadows cast by the sun are significantly dark because the intensity of sun light is much greater than skylight. For this reason, we have found that shadow detection by global thresholding to produce good results in most of our test cases, except for late afternoon images. Other methods for shadow detection could be used [8,20]. It is important to note that for the algorithm to work, perfect shadow detection is not required, as long as the following conditions are met: (i) some cast shadows should be detected, but it is not necessary for all of them to be detected. (ii) it is desirable that shadows due to surface normals pointing away from the sun be also detected. (iii) a pixel that is not in shadow should not mistakenly detected as a shadow pixel. When using global thresholding, it can be seen from these conditions that the ideal threshold is the minimum luminance value of a non-shadow pixel. In our system, the selection of the threshold is an interactive process in which a user picks a threshold value that satisfies the above conditions (Fig. 2).

4.2 View setup

The effect of rendering the model as it would be "seen" from the sun is achieved by setting up an orthographic projection with the view vector parallel to the direction of the sun rays. This direction is calculated using an astronomical formula [18] that takes as inputs the timestamp of the image and the latitude and longitude values of the site. Our system sets the view direction automatically, and lets the user translate the model until the desired section of the model is visible. The view position is chosen in such a way that the area of the model imaged in the photograph is completely seen.

4.3 Search space parametrization

The cost function is optimized with respect to the six dimensional space of camera positions and orientations. For the optimization to converge, a suitable parametrization of this space is required. Note that the trivial parametrization, using a 3-vector for the camera position and a 3-vector with the euler angles of the camera orientation, has the disadvantage that a small change in the camera orientation can produce a big displacement

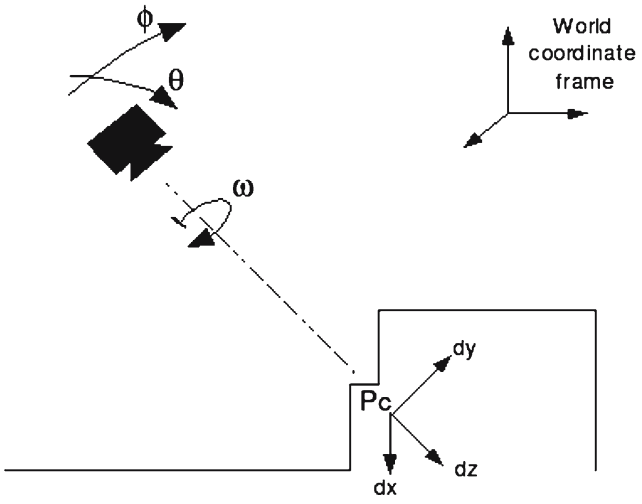

Fig. 3 Cost function parametrization. The camera frame is translated to the point where the camera's optical axis intersects the scene (from the initial camera configuration). Rotations and translations are described wrt to this translated camera frame

of scene elements within the image (specially those that are far from the camera). To overcome this problem we chose a parametrization in which the center of rotation is placed in the scene and not in the camera. Consider a camera restricted to lie on a sphere of radius r around a fixed point P_c in the model; then the search space could be defined by the 2D spherical coordinates of a point in the sphere and a roll angle. To add the three remaining degrees of freedom, we allow the sphere center to translated away from P_c. This parametrization, shown in Figure 3, describes the entire six dimensional space of camera positions and orientations. The camera pose is described by a 6-vector $(\theta, \phi, \omega, d_x, d_y, d_z)$, where θ and ϕ are the spherical coordinates of the camera position in the sphere, ω is the camera's roll angle and (d_x, d_y, d_z) is the displacement of the sphere center from P_c. We set P_c by taking the initial camera pose and tracing a ray along the camera's view direction into the model.

4.4 Cost function

The cost function to minimize is the number of visible shadow pixels in a rendering of the model. In practice, however, we have found that this function will not converge to the correct camera position if the optimizer selects a candidate pose in which the area of the model imaged by the camera is small. In this case, the reduction on the number of visible shadow pixels will not be due to a better camera pose estimate, but to the fact that the camera is looking at a much smaller region of the model. Therefore, a good cost function should penalize camera configurations in which the intersection of the camera's viewing frustum and the scene is small or empty. This can be achieved by keeping track of the number of textured pixels in the rendered model. The more pixels that are textured, the larger the area the camera is viewing.

Let I denote the image to be registered, M the model, and I_r a rendered image of M textured with I and camera parameters $(\theta, \phi, \omega, d_x, d_y, d_z)$. Then, we define the cost function as:

$$f(I_r) = \begin{cases} \frac{\text{shadow\_count}(I_r)}{\text{number\_of\_pixels}(I_r)} & \text{if } v(I_r) \geq t \\ 1.0 & \text{otherwise.} \end{cases} \tag{1}$$

where $v(I_r)$ is the count of textured pixels and t is a threshold value. This threshold forces the camera to a position that looks at the scene. In practice, we have defined this threshold as a fraction of the total number of textured pixels in the rendering produced when the camera pose is set to its initial estimate.

4.5 Minimization

Since the cost function f defined in (1) has no analytical derivatives and typically contains several local minima, most gradient descent methods fail to converge to a good solution. For this reason, we employ a variant of simulated annealing [11], which 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 take downhill steps.

4.6 Model rendering

At each iteration of the optimization, the minimizer provides a set of camera parameters $\mathbf{c}_k$. Using these parameters, the model is rendered and the cost function is evaluated over the generated image. To obtain a correct rendering requires occlusions to be detected and accounted for. Not every scene point is visible from the texture camera. If projective texture mapping is used, these occluded points will still be textured. We can avoid this problem by applying a technique similar to shadow mapping [22]. First, we render the scene from the position of the camera and make a copy of the depth buffer. Then we set the viewpoint to the position of the sun and re-render the model with shadow testing enabled so that scene points that are not visible from the camera are neither textured nor rendered.

5 Results

We have run a variety of simulation and real experiments to test the performance and robustness of the presented algorithm with respect to the different sources of error: (1) selection of shadow threshold, (2) resolution of 3D model (i.e accuracy of scanned geometry), (3) accuracy of the sun position. All of these three parameters can affect the final registration result. To obtain ground truth, we placed registration targets on the scene. The 3D position of these targets was measured by the range scanner with a precision higher than 3 mm. We manually selected these targets in the image and computed the camera pose using the linear pose estimation method of [2] followed by nonlinear optimization.

For our simulation experiments, we created a sequence of camera positions by randomly perturbing the orientation of the camera by as much as ±5° in each rotation angle and the translation by ±0.25 m in each axis. From each of these positions, we ran our algorithm setting the visibility threshold to a 60% of the original visible pixels.

Table 1 shows the results of 10 simulation runs. We use the reprojection error of the mesh vertices as a metric to evaluate the resulting camera: first the mesh vertices are projected to image coordinates using the ground-truth camera, then the image coordinates are computed using the camera position resulting from our algorithm. The reprojection error is the distance between the two points. Each column in Table 1 shows the computed average reprojection error for different shadow thresholds and model resolutions. Columns (A) to (E) were obtained using a high resolution model and varying shadow thresholds (30, 40, 60, 80 and 140). A threshold value of 30 fails to select most of the pixels corresponding to cast shadows. As a result, the algorithm performs poorly. This is explained by the fact that there are many camera configurations for which the number of shadow pixels is minimized. A threshold of 40 does detect most of the cast shadows and some attached shadows, improving the registration results. A value of 60 detects more shadows and labels some non-shadow pixels incorrectly as shadowed, but the results are not affected. Only when the shadow threshold is increased significantly to 140, where a large number of non-shadowed pixels are masked as being in shadow, are the registration results affected. For this test image, our method performs well over thresholds in the range [40,80]. All images are extremely large, 3008 × 2000 pixels, hence an average re-projection error of 4.9 pixels, as shown in column (C), is unnoticeable. This error corresponds to roughly 1.8 pixels for a 1024 × 768 image. The execution time for each simulation run was approximately 12 minutes on a Pentium IV machine. This time corresponds to 3,000 minimization iterations. Figure 4 shows the minimization progress over time. It shows cost of the the best configuration found against the number of iterations for each of the 10 simulations run in column (E).

5.1 Robustness against geometry resolution and sun position

To evaluate the effects that errors in the geometry can introduce in our algorithm, we have decimated the model reducing the number of triangles by 80%, and performed a set of simulation runs using a threshold of 60 for the shadow detection. The results are shown in column (F) of Table 1. The average reprojection error is higher than the error obtained for the same shadow threshold (column (E)) using the full resolution model. However, one advantage of using a decimated model is

Table 1 Shadow registration simulation results—initial and final average reprojection errors

Shadow threshold		30	40	60	80	140	60	60	60
Resolution (F/D)		F	F	F	F	F	D	F	F
Time offset (min)		0	0	0	0	0	0	−10	+10
Run no.	Initial	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)
1	143.5	10.3	5.4	6.0	5.1	6.9	14.1	5.6	6.0
2	101.4	5.8	5.8	4.1	5.0	7.0	9.0	6.0	5.9
3	208.3	9.2	5.8	4.5	4.9	6.8	14.5	5.9	5.8
4	125.3	10.7	5.7	4.9	5.6	6.4	16.8	5.2	5.6
5	171.2	7.8	5.0	4.9	6.0	6.1	11.3	6.0	5.3
6	81.1	6.8	5.0	4.7	5.2	7.0	14.4	5.3	6.7
7	207.4	8.8	4.7	4.9	5.7	6.2	12.4	5.0	8.5
8	77.0	10.1	5.3	5.2	4.9	6.4	11.4	5.5	5.3
9	238.4	7.7	4.8	4.8	5.0	6.3	12.7	5.8	5.1
10	180.6	7.2	5.4	4.8	5.8	6.9	13.0	5.4	4.9
Average	153.4	8.4	5.3	4.9	5.3	6.6	13.0	5.6	5.9

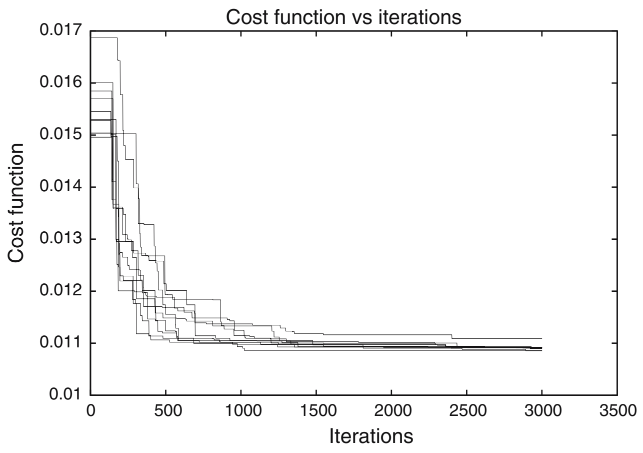

Fig. 4 Cost function optimization. This plot shows the best cost found against the number of iterations for ten simulation runs. It can be seen how the optimization converges

an increase in running speed, since less geometry has to be processed. This suggests an area of future exploration that could allow a speed increase: to run the optimization first with a highly decimated model and then refine the obtained registration with a more detailed one.

We also run the experiments introducing a small error in the time of the day used to calculate the position of the sun. The results for a time offset of ±10 minutes are shown in columns (G) and (H) of Table 1. The reprojection errors are higher in these cases, but not significantly, suggesting that a highly accurate time of the day (and hence position of the sun) is not required.

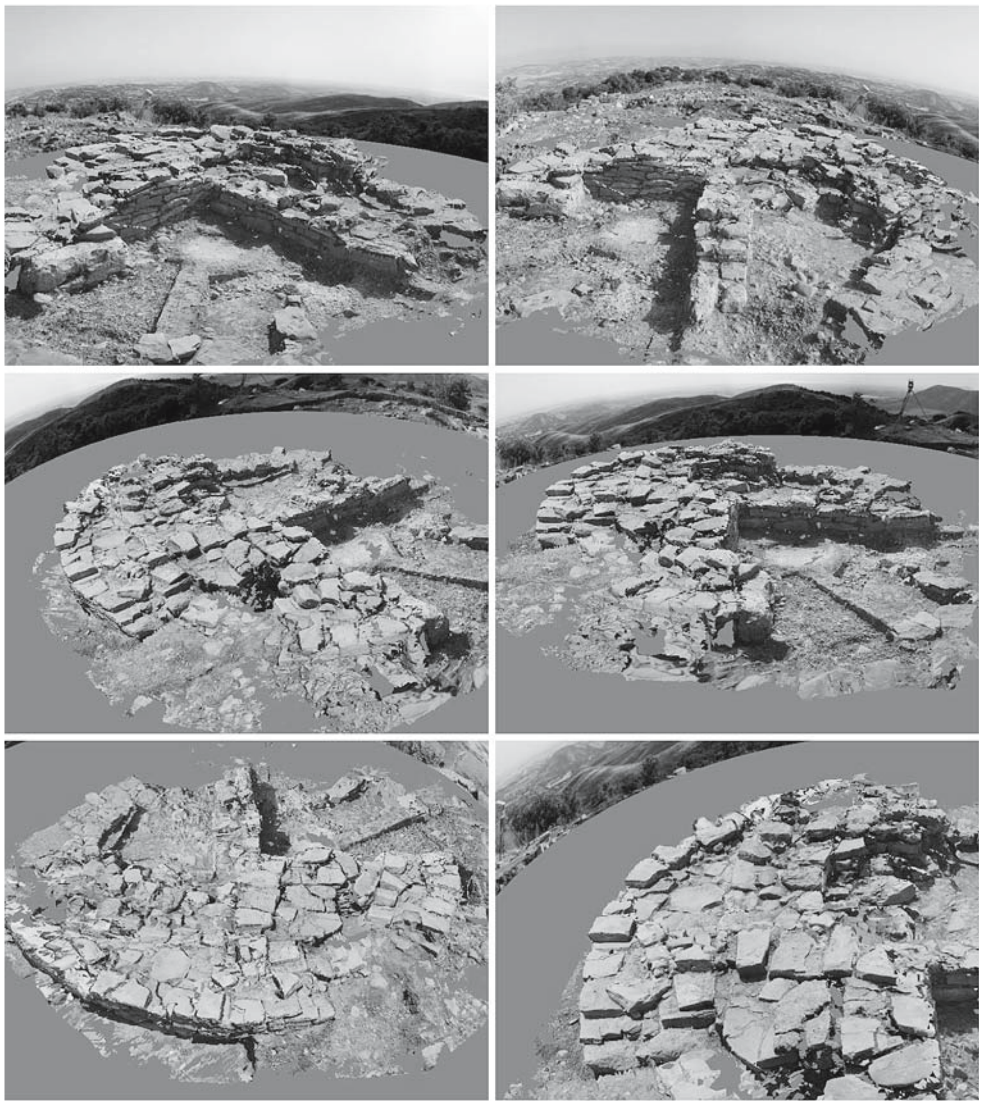

Fig. 5 Six different views of the texture model as obtained with our method. The background is given by a panoramic mosaic

5.2 Results on archaeological data

In addition, we used our registration method to align the images of our Acropolis model. The model consists of a 138K triangle mesh that is textured using 12 3,008 × 2,000 pixel images. We were able to successfully align 10 of the 12 images. For these cases we do not know the ground-truth, so we can not provide a quality measure for the registration, but we noticed two cases in which it failed. In one case the algorithm failed to find the intersection of the camera's optical axis and the scene (i.e. point P_c) due to missing geometry . In the other case, the algorithm failed because the shadows in the image, which had been taken in a late afternoon, did not have enough contrast and some non-shadowed regions were incorrectly masked as shadowed. Figure 5 shows the resulting model. In addition, a short video showing the entire model in 3D from many different viewpoints can be found at www.cs.columbia.edu/∼allen/sicily.avi.

6 Conclusions

We have presented an algorithm for texture registration that uses the shadows cast by the sun and which we successfully applied to the registration of images of an archaeological site. We have identified the different sources that can introduce errors in the procedure and shown quantitative results for the performance of the algorithm.

The use of shadows as features places some limitations on the type of applications in which the algorithm can be used. The presence of strong cast shadows is required for shadow detection, but we have have shown that global thresholding works well, even for a range of threshold values. Secondly, all geometry that is casting shadows in the scene must be present in the 3D model. In addition, it is also desirable, but not strictly required, that there be geometry for every pixel in the image. We have found that this is not always the case, and sometimes there is no geometry for distant objects or the sky. A user can mask out these regions. It is important to make a point here that this last restriction is not intrinsic to our method but also applies to other texture registration techniques (e.g. silhouette based algorithms require the background to be masked out).

Our current method can be improved by reducing the execution time. This time is highly dependent on the size of the rendered image I_r. By adopting a hierarchical scheme in which the model is rendered on different window sizes, similar to the one used by Lensch et al. in [14], the running speed could be improved.

Acknowledgement Ian Morris and the Stanford Archeology Center graciously provided access to and images and data from Monte Polizzo. This research was funded in part by NSF grant IIS-0121239.

References

1. Allen, P., Feiner, S., Troccoli, A., Benko, H., Ishak, E., Smith, B.: Seeing into the past: Creating a 3D modeling pipeline for archaeological visualization. In: Proceedings of 2nd International Symposium on 3D Data Processing, Visualization and Transmission (2004)
2. Ansar, A., Daniilidis, K.: Linear pose estimation from points or lines. IEEE Trans. Pattern Anal. Mach. Intell. **25**(5), 578–589 (2003). DOI http://dx.doi.org/10.1109/TPAMI.2003.1195992
3. Bernardini, F., Rushmeier, H.: The 3D model acquisition pipeline. Computer Graphics Forum **21**(2), 149–172 (2002)
4. Bernardini, F., Rushmeier, H., Martin, I.M., Mittleman, J., Taubin, G.: Building a digital model of Michelangelo's Florentine Pietà. IEEE Comput. Graph. Appl. **22**(1), 59–67 (2002)
5. Bouguet, J.Y.: Camera calibration toolbox for Matlab. (2001). Http://www.vision.caltech.edu/bouguet/calib_doc
6. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 303–312. ACM Press (1996)
7. Daum, M., Dudek, G.: On 3-d surface reconstruction using shape from shadows. In: CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, p. 461. IEEE Computer Society, Washington (1998)
8. Funka-Lea, G., Bajcsy, R.: Combining color and geometry for the active, visual recognition of shadows. In: ICCV '95: Proceedings of the 5th International Conference on Computer Vision, p. 203. IEEE Computer Society, Washington (1995)
9. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2000)
10. Ikeuchi, K., Nakazawa, A., Nishino, K., Oishi, T.: Creating virtual buddha statues through observation. In: IEEE Workshop on Applications of Computer Vision in Architecture, vol. 1 (2003)
11. Ingber, L.: Very fast simulated re-annealing. Math. Comput. Modelling **12**(8), 967–973 (1989)
12. Irvin, R.B., David M. McKeown, J.: Methods for exploiting the relationship between buildings and their shadows in aerial imagery. IEEE Trans. Syst. Man Cybern. **19**(6), 1564–1575 (1989)
13. Kriegman, D.J., Belhumeur, P.N.: What shadows reveal about object structure. In: ECCV '98: Proceedings of the 5th European Conference on Computer Vision, vol. II, pp. 399–414. Springer, Berlin Heidelberg New York (1998)
14. Lensch, H.P., Heidrich, W., Seidel, H.P.: A silhouette-based algorithm for texture registration and stitching. Graph. Models **63**(4), 245–262 (2001)
15. Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., Fulk, D.: The digital Michelangelo project: 3D scanning of large statues. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 131–144 (2000)
16. Liu, L., Stamos, I.: Automatic 3d to 2d registration for the photorealistic rendering of urban scenes. In: CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol.

2, pp. 137–143. IEEE Computer Society, Washington (2005). DOI http://dx.doi.org/10.1109/CVPR.2005.80

17. Pulli, K., Cohen, M., Duchamp, T., Hoppe, H., Shapiro, L., Stuetzle, W.: View-based rendering: Visualizing real objects from scanned range and color data. In: Rendering Techniques '97, pp. 23–34. Springer, Berlin Heidelberg New York (1997)
18. Reda, I., Andreas, A.: Solar position algorithm for solar radiation applications. Tech. Rep., National Renewable Enery Laboratory, Golden, Colorado (2003)
19. Rocchini, C., Cignomi, P., Montani, C., Scopigno, R.: Multiple textures stitching and blending on 3D objects. In: Rendering Techniques '99, Eurographics, pp. 119–130. Springer, Wien (1999)
20. Salvador, E., Cavallaro, A., Ebrahimi, T.: Cast shadow segmentation using invariant color features. Comput. Vis. Image Underst. **95**(2), 238–259 (2004). DOI http://dx.doi.org/10.1016/j.cviu.2004.03.008
21. Sato, I., Sato, Y., Ikeuchi, K.: Illumination from shadows. IEEE Trans. Pattern Anal. Mach. Intell. **25**(3), 290–300 (2003). DOI http://dx.doi.org/10.1109/TPAMI.2003.1182093
22. Segal, M., Korobkin, C., van Widenfelt, R., Foran, J., Haeberli, P.: Fast shadows and lighting effects using texture mapping. In: Proceedings of the 19th annual conference on Computer graphics and interactive techniques, pp. 249–252. ACM Press (1992)
23. Stamos, I., Allen, P.K.: Automatic registration of 2-D with 3-D imagery in urban environments. In: Proceedings of the 8th International Conference On Computer Vision (ICCV-01), pp. 731–737. IEEE Computer Society, Los Alamitos, CA (2001)
24. Yu, Y., Chang, J.T.: Shadow graphs and 3d texture reconstruction. Int. J. Comput. Vis. **62**(1–2), 35–60 (2005). DOIhttp://dx.doi.org/ 10.1007/s11263-005-4634-5