Interactive 3D Scene Reconstruction from Images

Aaron Hertzmann
Media Research Laboratory
Department of Computer Science
New York University
719 Broadway, 12th Floor
New York, NY 10003
hertzman@mrl.nyu.edu
http://www.mrl.nyu.edu/hertzman
NYU Computer Science Technical Report

April 22, 1999

Abstract

We propose an interactive framework for reconstructing an arbitrary 3D scene consistent with a set of images, for use in example-based image synthesis. Previous research has used human input to specify feature matches, which are then processed off-line; however, it is very difficult to correctly match images without feedback. The central idea of this paper is to perform and display 3D reconstruction during user modification. By allowing the user to interactively manipulate the image correspondence and the resulting 3D reconstruction, we can exploit both the user's intuitive image understanding and the computer's processing power.

1 Introduction and Related Work

The traditional goal of computer vision research is to computationally understand 3D scenes from sensor data. This research primarily targets real-time applications (such as robot guidance) and understanding the human visual system; hence, there is ideally no human intervention in the image analysis process. Although many impressive systems have been demonstrated, it is difficult to speculate if and when we will see effective algorithms for reconstructing *arbitrary* scenes from images.

A more recent research goal is the creation of 3D models for visualization and computer animation. The requirements for this problem differ from the requirements of real-time robot vision. Human assistance in the 3D reconstruction is often acceptable, because modeling is done in advance of rendering and viewing. However, artifacts and reconstruction errors are unacceptable, because the resulting models will ultimately be viewed by humans, in situations such as in movies or video games.

Many of the existing example-based image synthesis systems have used human input at some point in the correspondence/reconstruction process [3, 6, 7, 8, 9, 12, 14], particularly those that extrapolate views in addition to interpolating. User interaction, as reported in the literature, has been limited to marking feature correspondences, from which a 3D model is then automatically extracted. (Many of the systems cited represent the model implicitly, with structures such as dense point correspondences and a fundamental matrix. Up to a projective transform, the implicit representations are equivalent to reconstructing a 3D model that is *consistent* with the example images.) Work that has focused on specialized domains, such as architecture [3, 12] and human faces [7], has been particularly successful. However, it is exceptionally difficult to correctly mark image correspondences for arbitrary surfaces, or even to know which features to mark. Image correspondences that appear reasonable often result in grossly incorrect 3D models (Figure 2), requiring the user to repeat the tedious process of matching several times until an acceptable model has been created.

Our approach is to tightly couple user interaction with 3D reconstruction. The idea is to always display the 3D model during interaction. As the user selects and modifies point and line matches in the source images, the resulting 3D model is continually updated in another window. The user can correct a match to subpixel accuracy by dragging a corresponding point in the image plane, or by directly modifying the 3D geometry. This method gives the user immediate feedback as to the quality of the correspondence and the reconstruction.

We believe that including the human in the reconstruction loop will produce a quantum leap in our ability to reconstruct the geometry of arbitrary scenes. This will be achieved by properly combining the user's intuitive image understanding and qualitative reasoning with the computer's processing power.

2 Camera and Scene Model

We first describe the general mathematical structure of our system. The input is a set of images (e.g. photographs) of a scene, along with camera calibration information. Each camera calibration consists of:

- A focal center $\mathbf{c}_i$.
- A projection function $\mathbf{P}_i : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ that maps world points to image points.
- An inverse projection function $\mathbf{Q}_i : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ that maps image points to unit-length world vectors.

These quantities are related by $\mathbf{Q}_i(\mathbf{P}_i(\mathbf{x})) = \frac{\mathbf{x}-\mathbf{c}_i}{||\mathbf{x}-\mathbf{c}_i||}$, where $\mathbf{x}$ is any world space point visible to camera i. In our system, we have assumed a pair of pinhole projection cameras in the parallel configuration[1] (Figure 1).

A scene reconstruction is represented as a scattered set of world points $\mathbf{x}_j$. These points specify a set of sparse correspondences: $\mathbf{P}_0(\mathbf{x}_j)$ in image 0 matches $\mathbf{P}_1(\mathbf{x}_j)$ in image 1. We denote the image space points as $x^i_j = \mathbf{P}_i(\mathbf{x}_j)$. A line correspondence is represented as a pair (i, j) that indicates an edge between $\mathbf{x}_i$ and $\mathbf{x}_j$.

In order to render the surface in our system, we triangulate the world points using the topology of the constrained Delaunay triangulation [10] of the points x^0_i, with line correspondences used as triangulation constraints. The triangles are rendered in 3D, and texture-mapped with a source image, using the image coordinates as texture coordinates [2, 4]. These operations are fast enough to perform in real-time.

[1] This configuration is also known as the stereo configuration.

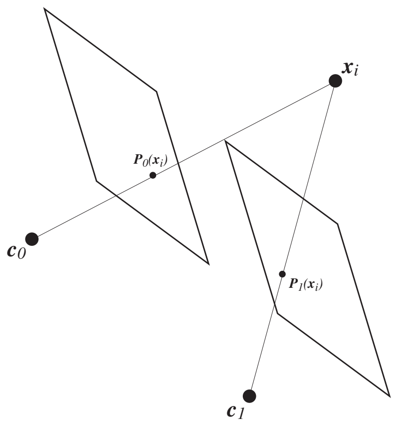


Figure 1: The parallel camera configuration.

For the goal of example-based image synthesis, it is sufficient to create a 3D model that is *consistent* with the source images. We could also use a representation such as point correspondences and a fundamental matrix that would make the calibration ambiguities explicit. We find the 3D formulation to be simpler to implement and understand, and easier to generalize. This use of 3D is inspired by [5].

3 The Interactive System

The 3D reconstruction begins with an initial point matching. We begin with an empty matching (no vertecies or triangles); an automatically-generated disparity map could also be used.

Given the initial model, the user may add point matches and modify point matches. To add a point match, the user simply clicks the point's initial location x_0^i in one of the images. The matching point in the other image is computed using a default disparity d: $x_0^1 = x_0^0 - [d, 0]^T$ or $x_0^0 = x_0^1 + [d, 0]^T$. $\mathbf{x}_0$ is computed as the intersection of the rays from the camera centers to the image points (i.e. $\mathbf{x}_0 = \mathsf{ray}(\mathbf{c}_0, \mathbf{Q}_0(x_0^0)) \cap \mathsf{ray}(\mathbf{c}_1, \mathbf{Q}_1(x_0^1))$, where $\mathsf{ray}(P, V)$ denotes a ray from point P in direction V). The new mesh is then retriangulated and rendered.

The majority of the user time with our method is spent modifying point matches. Point matches are displayed on each image as green crosses; when the mouse is moved over a match, the corresponding point in the other image is highlighted. The user first identifies a point match that does not appear correct, usually by looking for errors in the 3D mesh or by visually comparing the images under the match. Once a match is selected, the user simply drags the match in one of the images to a new location. While the match is being modified, the 3D model is continually updated and rendered, allowing the user to interactively refine the mesh shape. This "human-in-the-loop" reconstruction idea is simple, but very effective.

The new world point location $\mathbf{x}_i$ for the match is computed by intersecting the ray through the new match point $r_0 = \mathsf{ray}(\mathbf{c}_0, \mathbf{Q}_0(x_i^0))$ and the ray through the corresponding point $r_1 = \mathsf{ray}(\mathbf{c}_1, \mathbf{Q}_1(x_i^1))$. If the rays do not intersect, then the point on r_1 is chosen that is nearest to r_0, and new values for the image points are computed by projection ($x_j^i = \mathbf{P}_i(\mathbf{x}_j)$). This is equivalent to constraining the match to lie on the epipolar line of the corresponding point. If the user drags with the right mouse button instead of the left button, then the point on r_0 nearest to r_1 is chosen. In this case, the user may move a point freely, and the corresponding point is constrained to lie on the new epipolar line.

The user creates a line correspondence simply by specifying two existing points as the endpoints. Line constraints are useful for restricting mesh topology.

In addition to the textured rendering mode, we display a difference image (Figure 4). The difference image is the absolute value of difference between the left source image rendered onto the mesh and the right image rendered onto the mesh. Poorly-matched pixels are light and good matches are dark. With this image, the user can try to manipulate matching points in order to "black out" the difference image. This method is very useful in textured areas without many discontinuities. However, the difference image is often an insufficient measure of a match, especially in areas with little texture; difference image editing must be used in conjunction with the standard editing modes.

4 Discussion and Future Work

We have presented an approach to reconstructing arbitrary 3D scenes from models, by exploiting human image understanding together with computational power. We believe that this framework can potentially make 3D reconstruction cheap and effective for a wide variety of applications.

There are several important extensions.

- Support many example images, to allow reconstruction of large environments.
- Provide a standard 3D modeling interface for direct modification of the geometry and topology of the 3D model. For instance, the ability to mark discontinuities in the mesh is critical.
- Provide interactive modification of the camera calibration.
- Include local stereo computations during interactive updates. For example, surfaces could "stiffen" in proportion to the level-set evolution term of [5].

For each of these extensions, the underlying mathematics are straightforward, but the user interface is not.

Acknowledgments

Delaunay triangulations were computed using [11]. The head image pair was obtained from [13] and is copyright © INRIA-Syntim. (We did not make use of the calibration data provided for the source images.) The author is supported by NSF grant DGE-9454173.

References

[1] Shai Avidan, Amnon Shashua. Novel View Synthesis by Cascading Trilinear Tensors. *IEEE Transactions on Visualization and Computer Graphics*. Vol. 4, No. 4, October/December 1998

[2] Lucia Darsa, Bruno Costa, and Amitabh Varshney. Navigating Static Environments Using Image-Space Simplification and Morphing. *ACM Symposium on Interactive 3D Graphics,* Providence, RI, 1997, pp. 25–34

[3] Paul E. Debevec, Camillo J. Taylor, Jitendra Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach. SIGGRAPH 96 Conference Proceedings. August 1996. pp. 11–20.

[4] Kenneth E. Hoff, Understanding Projective Textures: Correcting the Double-Perspective Distortion. http://www.cs.unc.edu/~hoff/techrep/index.html

[5] Olivier Faugeras, Renaud Keriven. Complete Dense Stereovision using Level Set Methods. *Proc. ECCV '98.*

[6] Leonard McMillan, Gary Bishop. Plenoptic modelling: an image-based rendering system. *SIGGRAPH 95*, (August 95) pp. 39–46.

[7] Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski and David H. Salesin. Synthesizing Realistic Facial Expressions from Photographs. SIGGRAPH 98 Conference Proceedings. July, 1998. pp. 75–84.

[8] Luc Robert, Stephane Laveau. Online 3-D Modeling with images. http://www-sop.inria.fr/robotvis/projects/Realise/java

[9] Steve Seitz and Charles Dyer, View Morphing, in Proc. SIGGRAPH-96, pp. 21-30, 1996.

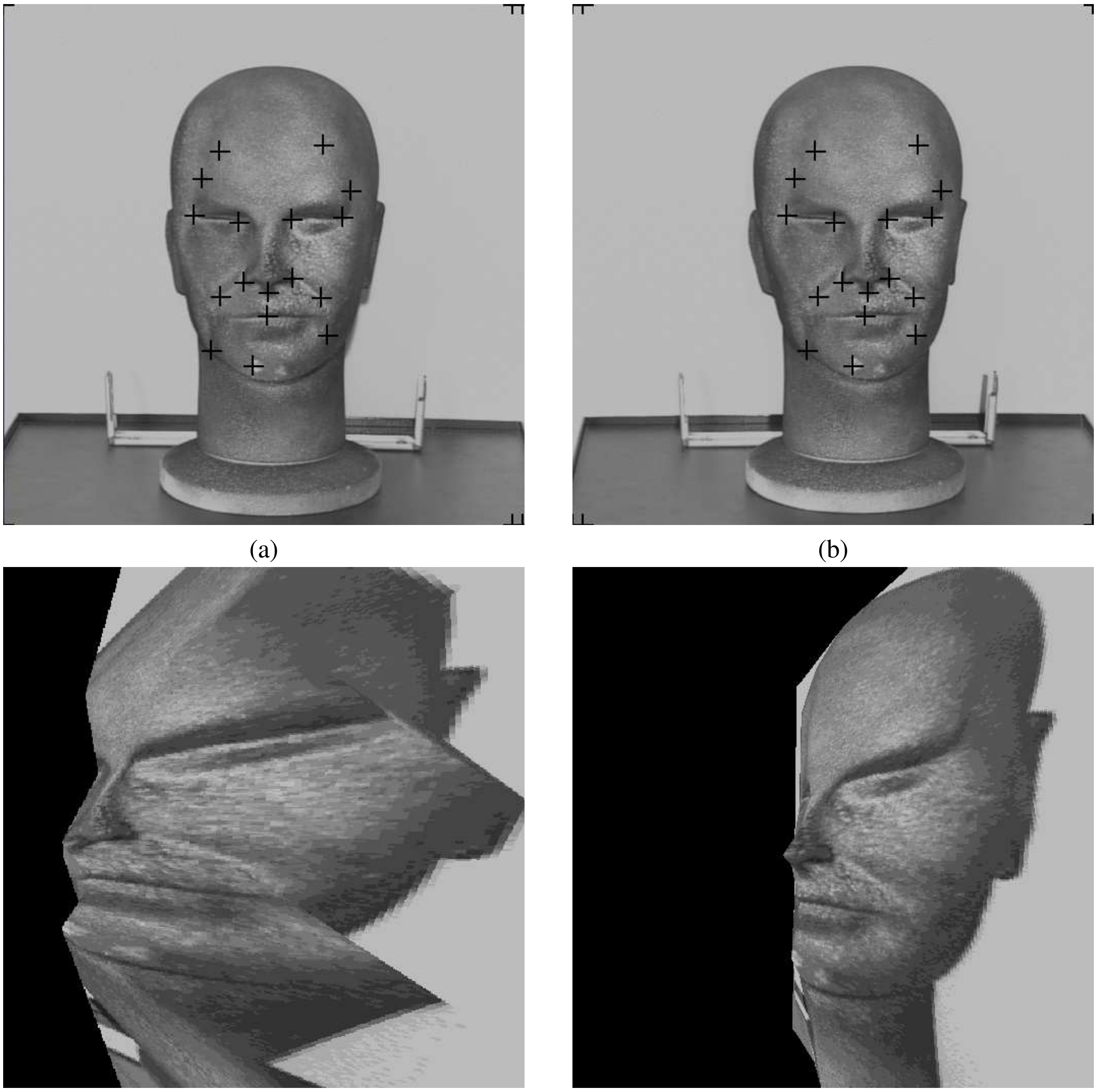
[10] Jonathan Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. *First Workshop on Applied Computational Geometry*. p 124-133, 1996.

[11] Jonathan Shewchuk. Triangle: A Two-Dimensional Quality Mesh Generator. http://www.cs.cmu.edu/~quake/triangle.html

[12] Heung-Yeung Shum, Mei Han, Richard Szeliski. Interactive Construction of 3D Models from Panoramic Mosaics. *IEEE CVPR*, pages 427-433, Santa Barbara, June 1998.

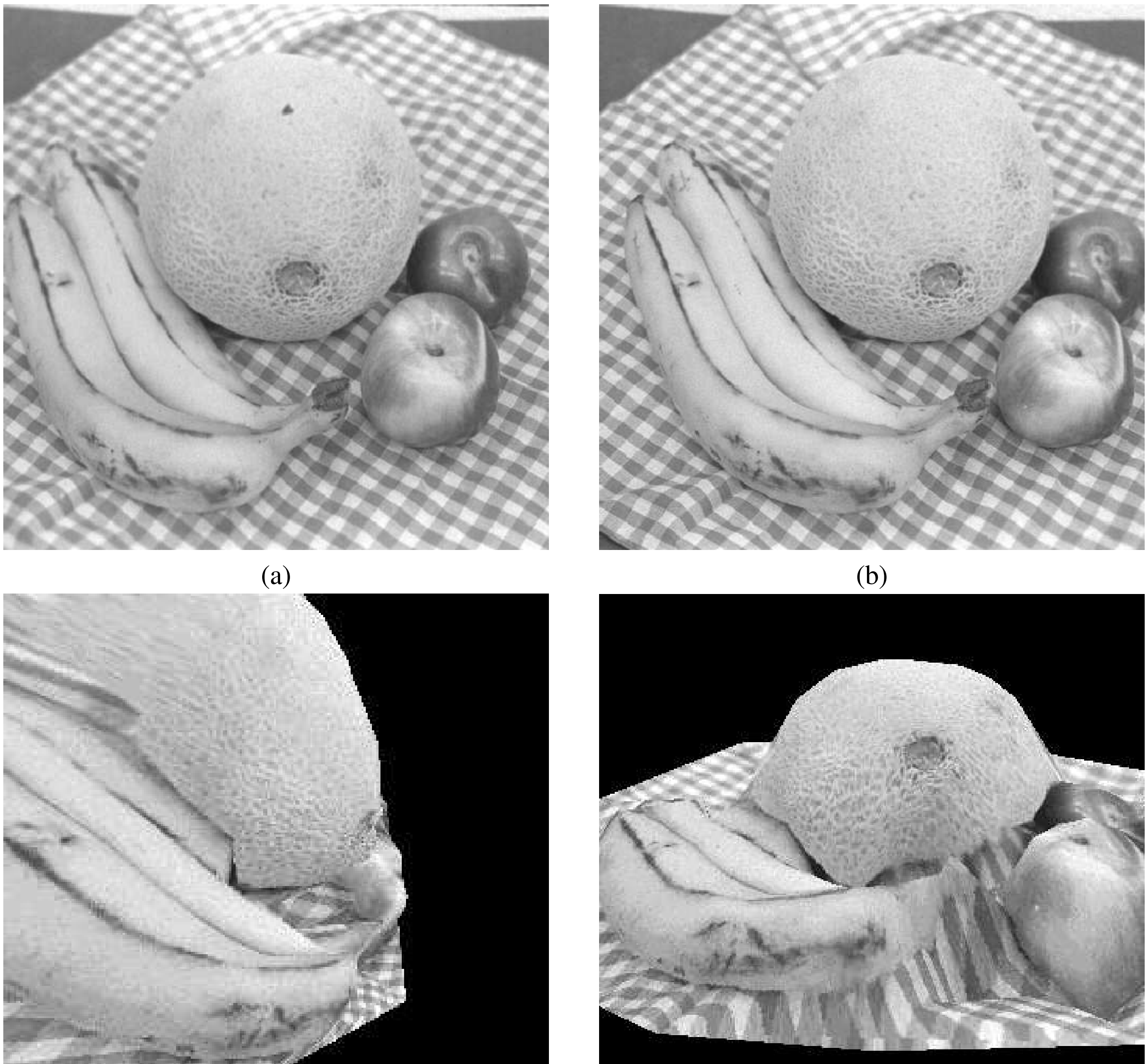
[13] Jean-Phillippe Tarel. Stereograms of the Syntim project. http://www-syntim.inria.fr/syntim/analyse/paires-eng.html

[14] Randall Warniers. Every Picture Tells A Story. *Computer Graphics World.* October 1998.



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

Figure 2: (a),(b) Hand-marked feature correspondences, without interactive reprojection. (c) Reprojection of feature matches. Feature correspondences that look reasonable on the image plane often produce grossly incorrect 3D models. (d) Interactive adjustments to the matching can vastly improve the 3D reconstruction.



(a) (b)

(c) (d)

Figure 3: (a),(b) Fruit stereo pair. (c),(d) Model generated from fruit stereo pair.

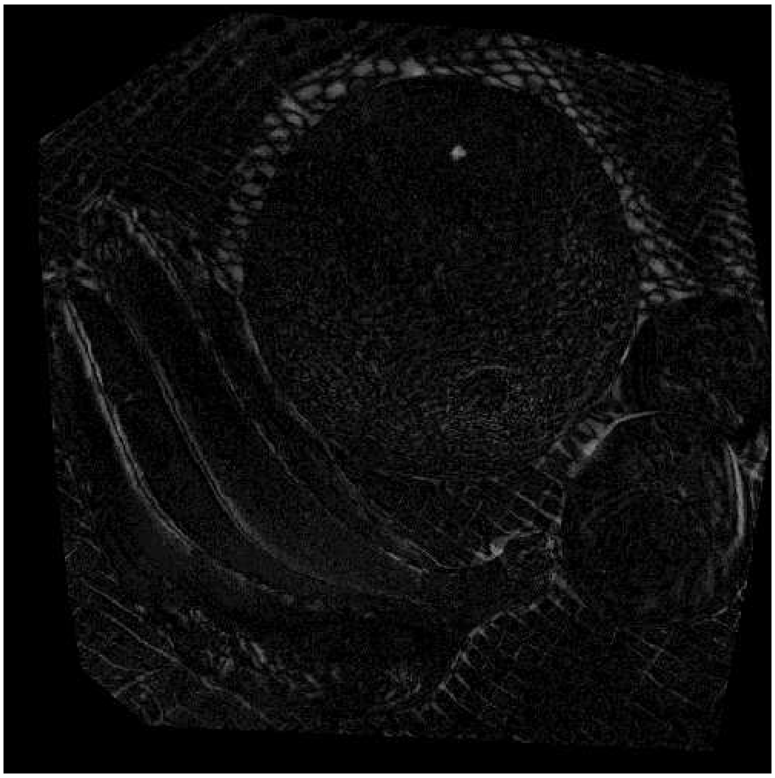


Figure 4: Difference between source images textured on the same mesh. Most errors appear near disconti-nuites, which are not yet handled by our system.