Rapid Part-Based 3D Modeling

Ismail Oner Sebe
PhD. Student
Integrated Media Systems Center
Los Angeles, CA 90089-0781
(213) 740-4250
iosebe@graphics.usc.edu

Suya You
Research Assistant Professor
Integrated Media Systems Center
Los Angeles, CA 90089-0781
(213) 740-4250
suyay@graphics.usc.edu

Ulrich Neumann
Associate Professor
Integrated Media Systems Center
Los Angeles, CA 90089-0781
(213) 740-4250
uneumann@graphics.usc.edu

ABSTRACT

An intuitive and easy-to-use 3D modeling system has become more crucial with the rapid growth of computer graphics in our daily lives. Image-based modeling (IBM) has been a popular alternative to pure 3D modelers (e.g. 3D Studio Max, Maya) since its introduction in the late 1990s. However, IBM techniques are inherently very slow and rarely user friendly. Most IBM techniques require either very extensive manual input and/or multiple images. In this paper, we present an IBM technique that gives high level of detail with 1-2 minutes of manipulation from a novice user using only single, un-calibrated image. Our system modifies a generic part-based model of the object under investigation. User inputs are entered via a simple interface and converted into modifications to the whole 3D model. We demonstrate the effectiveness of our modeler by modeling several vehicles, such as SUVs, sedan/hatchback/coupe cars, minivans, trucks and more.

Categories and Subject Descriptors

I.4.1 [**Digitization and Image Capture**], I.3.5 [**Computational Geometry and Object Modeling**], I.3.7 [**Three-Dimensional Graphics and Realism**]

General Terms

Algorithms

Keywords

Image Based Modeling and Rendering, Rapid 3D Modeling, Part-Based Modeling

1. INTRODUCTION

3D models in conjunction with textures are the most important part of computer graphics. Despite their importance, no simple and fast method exists for creating 3D models of real-world objects. This task is often left to digital artists and generally requires hours of manual work. Creating 3D models via combining other 3D models is previously proposed to mitigate this problem [3, 5]. These pure 3D-model-based methods require a big database of objects and are limited to the variety of models or their parts in the database. Image-based modeling (IBM), on the other end of the spectrum, promises an easy path for 3D modeling using only images.

Most IBM techniques require two or more images of the object from different viewpoints. Reconstruction from silhouettes requires segmented images from calibrated cameras and often creates crude models [10]. Volumetric reconstruction technique involves a similar idea but use the color of the interior region to estimate the 3D model [8]. Reconstruction from one image is possible under certain assumptions. Liebowitz *et al* presented a system that uses vanishing points to reconstruct the scene, where the scene is composed of planar regions [9]. An automatic, simpler version of this technique is presented in [6]. Rather than modeling everything, Façade system focuses on building modeling [4]. The system uses various object specific information to make the modeling system more automatic. In spite of its popularity in building modeling, a generalization of this idea to other object types is not obvious (at least to the authors of this paper).

In this paper, we present a part-based object modeling technique that does not require multiple images or tedious manual work. This system can be used by novice users and is able generate acceptable quality models in a few minutes. Proposed system lies somewhere between pure image-based techniques and pure 3D-model-based techniques. Our modeler uses a part-based generic model of the object, where each part affects the others via connectivity of the parts. The user is able to appropriately change the 3D model via a simple interface using the image as a guide. In particular, we choose to model vehicles. Our present work is able to model sedan/hatchback/coupe cars, SUVs, minivans and many others with the same pipeline. The next section gives an overview of the whole system. Section 3 describes the details of the algorithm: camera calibration, texturing, model inputs and model update. Results are presented in section 4 along with a discussion.

2. SYSTEM OVERVIEW

In order to model a class of objects (e.g. vehicles) with our framework, a generic model of that class needs to be created. For example, for vehicles a generic sedan is used (see Figure 2). A single un-calibrated image of a vehicle is enough to use the system. As an initial step, generic 3D model and the image is loaded to the system. First, camera calibration is done by user interaction and the features already available in the image.

Second, visibility of each triangle in the object model is automatically estimated and projected onto the image (see Figure 4). Third, the 3D model is textured using this visibility information. After this step, the user can select a particular part and move it in a way that looks appropriate. The inputs are given in 2D and converted to 3D by back-projection. The rest of the model is updated using this input and connectivity of the 3D model. Current framework is limited to moving parts and cannot add or delete parts; this is being investigated for future work. The user can also switch between the 2D image (with the 3D model projected on it) and the 3D rendering of the 3D model (with the 2D image as texture). This dual visualization allows for better understanding of the model since both have their strengths and weaknesses. Final output is a modified 3D model with texture information. Once the modeling is finished, the modified 3D model is saved for future use. The whole procedure can be repeated using this newly created 3D model and a new image of the object from a different viewpoint. This is often required if the vehicle has some important features that were not visible from the previous view. The second image can be taken from another camera and does not have to have an overlap with the first image.

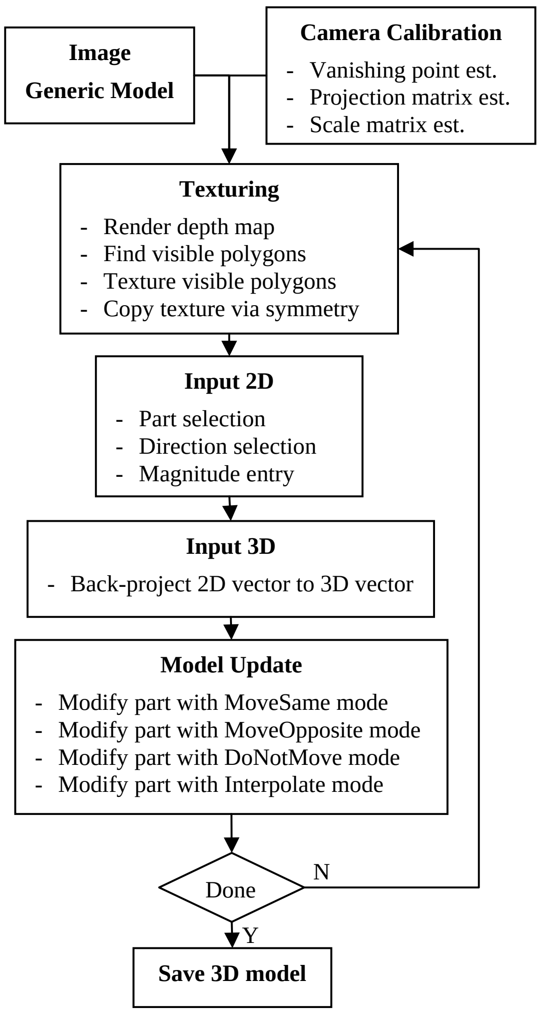

Figure 1. Flowchart of the algorithm

3. PROPOSED METHOD

3.1 Generic 3D Model

Most man-made objects are composed of parts. For example, a table is composed of legs and a top; a car is composed of a bumper, tires, roof, windows, etc. In this paper, we will concentrate on modeling of vehicles only. Most vehicles have the same parts, though they may vary in shape, size and their relative location. For example, a sedan has a long trunk, whereas a hatchback's trunk is small and as SUV does not have one.

Classification by parts is partially analyzed in [7]. Huber *et al* divide vehicles into three parts: front, middle, and back. Although that is a good starting point, this representation is not detailed enough to be used for modeling different vehicles. We further divide vehicles into four vertical groups: bottom, lower middle, upper middle, and top. In particular, we have 28 parts in our car model. These parts are front and back bumper, front and back windshield, four windows, four doors, four tires, four fenders, front/middle/back of the underside, left and right separators between the back windows and back windshield, roof, trunk, and hood. A sample view of our model is shown in Figure 2. These parts not necessarily correspond to the actual parts of the car.

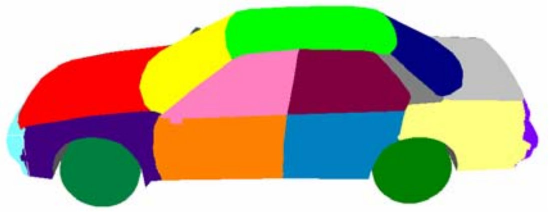

Figure 2. Generic Car Model

3.2 Camera Calibration

We use calibration via the vanishing points approach, which is widely used in architectural modeling [2]. This approach, under no radial distortion, fixed aspect ratio and zero skew assumptions, gives the internal camera parameters and the rotation matrix. The camera translation, i.e. location of the generic model relative to the camera, and the scale matrix, i.e. the size of the generic model, are not known. The user is required to locate the center of the tires (two image clicks), where location of the back tire is set to be the center of the model. Translation of the camera is estimated by intersecting the backprojection lines from the center of the tires. A scaling matrix (3x3 diagonal matrix) is estimated by least squares minimization on the projection error of the chosen points. Calibration plus scaling is done via a total of 13 image clicks. All matrices are combined in a 3x4 projection matrix. Objects which do not have straight lines can be calibrated via inserting a calibration object that does have these lines, e.g. a box.

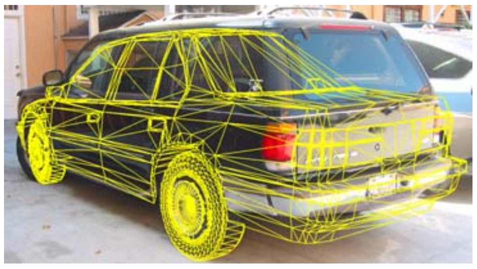

Figure 3. Projected 3D model after calibration

3.3 Visibility and Texture Mapping

Once the projection matrix is estimated, the 3D model can be projected on the image. However, projecting the whole 3D model results in a cluttered view. Instead, a depth map is rendered and each polygon is tested for visibility. Only the visible polygons are displayed (see Fig 3). This visibility information is further used to texture the 3D model, where the portions of the car that are not visible are textured using the symmetry of the car. Although this texturing recovers most of the vehicle, there are still portions where neither the actual location nor the symmetric counterpart is visible, e.g. the hood in Fig 4. These parts are rendered with the median color of the car that is estimated from the visible polygons. The texture information is used both as an overlay of the 3D model on the image and texture on the 3D model.

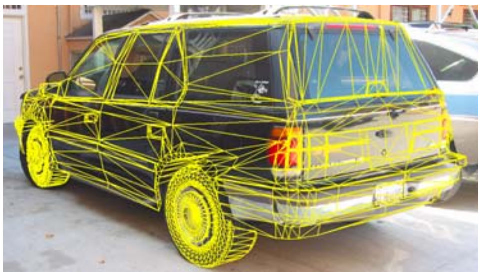

Figure 4. Projected 3D model after modeling

3.4 Modeling via Part-Based Modifications

3.4.1 Input

Inputs to the modeler directly affect the performance of the overall system. Although a 3D movement is required for modeling, most novice computer users are more comfortable with 2D images. Furthermore, texture on 3D models often covers the model inaccuracies, which makes it harder to model objects. We propose an easy and intuitive interface in 2D for part modifications. For example, the required changes to the model in Figure 3 are mainly on the back windshield and the hood, where they are easy to spot. The user clicks on one particular part to select it. Later, a 2D movement vector is inputted by a mouse drag. This 2D vector needs to be converted to a 3D vector, since the model is in 3D. This conversion is not trivial when the parts can move freely in 3D. However, this can be solved when the movement is limited to a small finite number of directions, such as three major axes and the normal of the surface. These vectors are projected on the image and one of these vectors is chosen by user interaction. The magnitude of the movement is estimated by back-projecting the 2D mouse drag onto the 3D direction vector selected by the user.

3.4.2 Model Update

Once the part, direction and magnitude of the movement are known, modification is applied to the selected part. Unfortunately, repeating this procedure for every part causes two major problems. First, a lot more input/iteration needs to be done for the modeler to give an output, since the same type of movement needs to be iterated for every part. Second, the neighboring parts may get disconnected due to the differences in inputs. In order to prevent this, our system creates a connectivity map of all the vertices in the 3D model. Connectivity enables the system to distribute the selected movement to other parts accordingly. Furthermore, enforcing connectivity prevents holes in the 3D model (although there may be certain types of objects where this is desired).

Parts are modified by one the following four modes: MoveSame, MoveOpposite, DoNotMove, and Interpolate. MoveSame mode directly translates the part with the given amount and direction. MoveOpposite translates the part with the given amount but in the mirror symmetric direction (almost all vehicles are horizontally symmetric). DoNotMove sets the movement of the part to zero. Interpolate first finds the vertices in the part that are already assigned a movement (some boundary points shared with other parts may already have movements assigned), then using a scattered data interpolation method the movement is interpolated to the other points that have not yet been assigned a movement. Interpolation is done on the movements rather than the actual 3D location of the points to preserve the individual shape of each part. We chose the triangulation-based scattered data interpolation technique due its piece-wise linearity property [1]. Although parts can be modified in any order, order affects the quality of the overall model. We propose that parts should be modified in order of their modification mode. The order used in our framework is MoveSame, MoveOpposite, DoNotMove and Interpolate, respectively. This order creates good models since scattered data interpolation performs better when the number of points with known values is large and well distributed, so it should be performed last. The modes are assigned to the parts according to which part is selected as the input part. MoveSame is assigned to the selected part, and the symmetric counterpart of the selected part gets MoveOpposite mode. The neighbors of these two parts are in mode Interpolate and the rest of the parts are in DoNotMove mode.

4. RESULTS

Using a regular off-the-shelf digital camera, we took single images of various vehicles. Due to space limitations, only three examples are demonstrated here: an SUV (Ford Explorer), a coupe (Honda Civic), and a hatchback (Toyota Matrix). The proposed algorithm requires no change in the processing of these images. The average number of modifications is around 10. The total execution time is about 1-2 minutes for each model (the program is able to apply the changes at interactive rates but it takes several seconds for the user to figure out the best next move). The results of our part-based modeler can be seen in Figures 5, 6, and 7. (a) and (b) images are renderings of the reconstructed models without texture and (c) images are these 3D models textured using the technique in Sec 3.3. There are two major limitations of our system. First, it is not able to delete/add parts, where this can be incorporated using a similar technique to [3, 5]. For example, the Civic is a coupe but the model created is still a four door (although the back door is modified to be smaller to match the image). However, replacing doors with a flat surface will not drastically change the appearance of the vehicle. Second, objects can not be conceptual, since at least one image is required.

5. CONCLUSION

In this paper, we presented a rapid 3D modeling scheme that uses a part-based representation for the objects that are created. 3D models of vehicles are created from a single un-calibrated image with moderate user interaction in about a minute of manipulation time. Different vehicles like SUVs, minivans, and sedan/hatchback/coupe cars are created without any change in the program. For the future, an automatic modeler using this framework is being investigated. Also, addition and deletion of parts from the model is being studied.

6. ACKNOWLEDGMENTS

This work was supported by the National Geospatial Intelligence Agency (NGA) under a NGA University Research Initiative (NRUI) program. We thank the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, for their support and facilities.

7. REFERENCES

[1] I. Amidror. "Scattered data interpolation methods for electronic imaging systems: a survey". Journal of electronic imaging 11(2), 157-176. 2002

[2] R. Cipolla. T. Drummond, and D. Robertson. "Camera calibration from vanishing points in images of architectural scenes". British Machine Vision Conference Proceedings 99

[3] M. Cohen. "Everything by example". Keynote talk at Chinagraphics 00, 2000.

[4] P.E. Debevec. "Modeling and rendering Architecture from photographs". PhD Thesis, UC Berkeley, 1996

[5] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, D. Dobkin. "Modeling by example". SIGGRAPH 04, pp. 652-663, 2004

[6] D. Hoiem, A.A. Efros, M. Hebert. "Automatic Photo Pop-up". To appear in SIGGRAPH 05. 2005

[7] D. Huber, A. Kapuria, R. Donamukkala, M. Hebert. "Part-based 3D object classification". CVPR 04, 2004

[8] S. Kutulakos, K. Seitz. "A theory of shape by space carving". In International conference on computer vision, pages 307-314, 1999

[9] D. Liebowitz, A. Criminisi, and A. Zisserman. "Creating architectural models from images". In Eurographics 99, pages 39-50, 1999.

[10] W. Matusik, C. Buehler, R. Raskar, L. McMillan, and S.J. Gortler," Image-Based Visual Hulls", SIGGRAPH 00, 2000

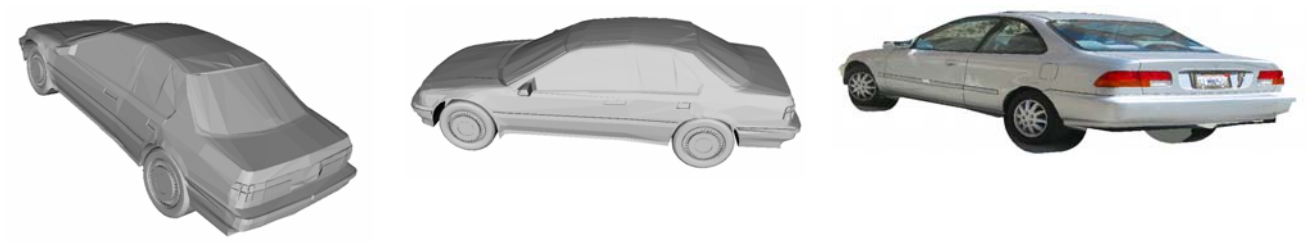
(a) 3D model without texture view 1 **(b) 3D model without texture view 2** **(c) 3D model with texture**

Figure 5. Reconstructed 3D model of Honda Civic

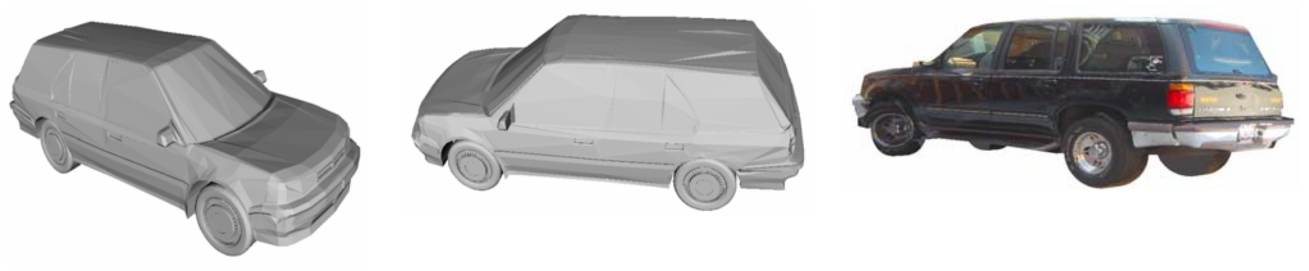
(a) 3D model without texture view 1 **(b) 3D model without texture view 2** **(c) 3D model with texture**

Figure 6. Reconstructed 3D model of Ford Explorer

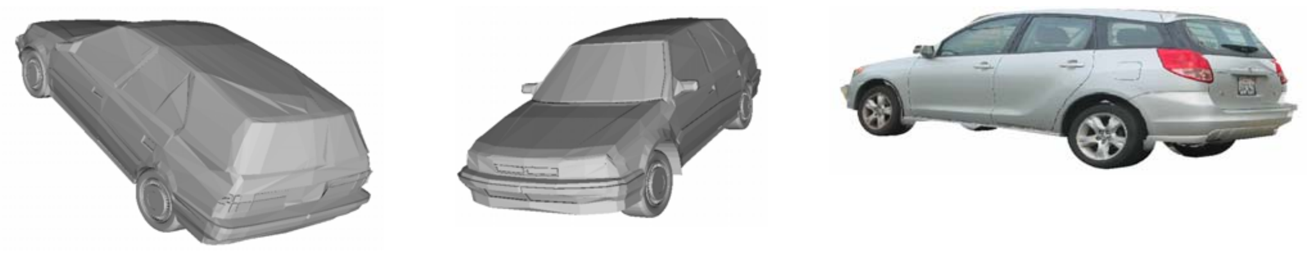
(a) 3D model without texture view 1 **(b) 3D model without texture view 2** **(c) 3D model with texture**

Figure 7. Reconstructed 3D model of Toyota Matrix