Interactive 3D Model Completion

A. van den Hengel, A. Dick, T. Thormählen, B. Ward
School of Computer Science
University of Adelaide
Adelaide 5005, Australia
anton.vandenhengel@adelaide.edu.au

P. H. S. Torr
Department of Computing
Oxford Brookes University
Oxford OX33 1HX, UK
philiptorr@brookes.ac.uk

Abstract

A common problem when using automated structure from motion techniques is that the object to be modelled can only be partially reconstructed from the video. This can occur because not all of the object is visible in the video, or because of featureless or ambiguous regions on the object's surface. In this paper we present an interactive method for rapidly and intuitively generating a complete 3D model from the output of a structure and motion algorithm. The method combines information obtained from the video data with the partial 3D model and user interaction. It is demonstrated on video containing partially seen objects, including planar and curved surfaces, and indoor and outdoor settings.

1. Introduction

Photo-realistic 3D models are now routinely used in applications such as movie post-production, architectural visualisation and video games. However, there is no simple and reliable way to convert images and measurements of an object into a 3D model. Typically this is done in one of three ways:

(a) Build the model in a 3D modelling package such as 'Blender'.
(b) Use a photogrammetry package such as 'Photomodeler' to build the model with the aid of calibrated images.
(c) Use a structure and motion [2] package such as 'PFTrack' to automatically acquire a 3D mesh from the images.

Using option (a) one is able to create a model of almost any real world object, provided one is trained to use a 3D modelling package. However for all but the simplest models it is also extremely labour intensive, as it is difficult through manual editing to generate a model which accurately reflects the shape and appearance of the objects depicted in the input image set.

Option (b) still requires manual delineation of structural features in each image, in order to recover 3D structure. These systems also require camera calibration because they do not make use of all the information that can be derived from the images using vision algorithms. Examples of these systems include Photomodeler, Canoma (derived from Facade [5]), paralleliped based modelling [8] and image based architectural modelling packages such as [4, 3].

Option (c) requires less input from the user, as camera calibration and some scene structure is recovered automatically, but automatically generated mesh-based reconstructions rarely achieve level of fidelity required for real applications.

Unless an image sequence has been carefully captured with the purpose of modelling in mind there are likely to be parts of the scene about which there is not enough information from which to generate a model automatically.

The approach described in this paper is a flexible alternative to those outlined above, and is also able to extend the information recoverable from the image set to produce models that would otherwise be impossible to achieve. In order to achieve this goal we introduce a number of operations for modelling missing or hidden structure, based on parts of the model that are visible. These are combined with other sketching interactions to form a powerful set of modelling operations that allow the rapid creation of complete models from partial data.

1.1. System overview

The modelling system uses structure and motion estimation to obtain camera parameters for each frame of video, and a (possibly empty) set of reconstructed scene points. A model is then created incrementally by sketching freehand strokes on frames of the video. After each sketching interaction, the model estimate is updated (see Figure 1). This overall workflow is similar to [6]. The authors in [6],

however, aim to facilitate the modelling of structures where sufficient information exists in the image set. The work presented here extends this approach to use the combination of image-based information and human input to generate models that would otherwise be impossible to recover.

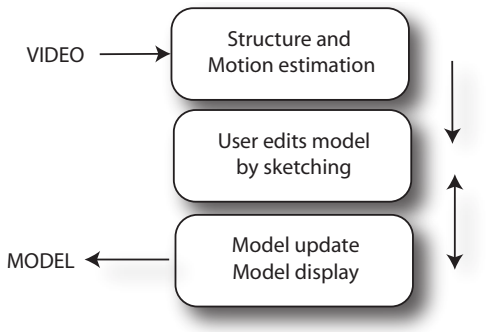

Figure 1. Modelling workflow. A model is incrementally built from the output of structure and motion estimation by sketching on one or more frames of video.

Many interactive 3D modelling techniques are already provided by dedicated modelling packages. Rather than duplicate these in an ad-hoc way we provide a structured mechanism by which they may be incorporated into our modelling process. This is done by the specification of a set of interacting production rules describing the requirements and consequences of each technique. Each rule describes the means by which a certain technique changes the state of the system, and therefore how it may be used in relation to other similarly described techniques.

The use of production rules allows the integration of a wide variety of existing modelling techniques with a novel means of specifying a 3D curve through scene space by drawing on a single image which we describe below. This technique uses both image data and the camera information recovered through structure and motion analysis to determine the most likely 3D location of a point on a 2D curve drawn over a frame of video. This analysis is founded in 3D and thus would not be possible without knowing the camera parameters. This may be contrasted against sequential 2D techniques such as the image stack approach sometimes used in video segmentation (see [7] for example).

Section 2 of the paper describes the general format of a modelling rule, and gives some examples. In Section 3 the fundamental rule for transforming a sketched 2D curve into 3D is described in detail, along with a selection of other rules. These rules can be combined to model complex objects, as shown in Section 4.

2. Production rules

By defining a production rule corresponding to each modelling operation we provide a common framework within which to formulate all modelling operations. One of the most important benefits of the production rule formalism is that it requires each method to use the same input and output formats, allowing the output of one method to form the input of another. This facility is critical for the modelling of non-visible aspects of the scene as it allows a curve modelled on part of the image set to be extruded past the image boundaries and then replicated, for example.

The rules operate on two lists: a list of interactions (we call the I-list) and a list of modelled primitives (called the M-list). Since interactions are all in 2D, the I-list contains 2D primitives (line segments or curves). These are grouped according to whether they are connected in the image. Each image has its own I-list.

2.1. Some basic rules

The user can manually select a number of modelling operations. These modes of operation determine how each stroke is interpreted by the modelling engine, and are indicated in the precondition of a rule by the name of the mode. Where two rules might fire on the same input, we apply the rule which comes last in the list. The ordering of the list thus affects the behaviour of the system. If the application of a rule fails the input is not removed from the input queue. Following is a subset of the implemented rule set in the following format:

Name {Input}[Precondition] $\Rightarrow$ Effect.

- **Add line** {Single drawn line}[Drawing mode] $\Rightarrow$ Add line to a new group in the I-list
- **Group lines** {Single drawn line}[Drawing mode, Drawn line endpoint equals an existing endpoint] $\Rightarrow$ Add new line to existing line's group in the I-list
- **Close curve** {Single drawn line}[Drawing mode,Both line endpoints equal endpoints within existing group] $\Rightarrow$ Add closed curve group to I-list

Further rules are presented in Section 3, along with a description of the processing which they involve.

3. Modelling by sketching

A 3D model is composed of a number of 3D primitives, including planar facets, NURBS surfaces, and curves. This section details how these 3D primitives are obtained from 2D sketching interactions with one or more images, and how the model is re-estimated after each sketching operation.

3.1. A 3D curve from single drawn curve

The production rule for this process is

3D curve from line {Single drawn line}[Drawing mode] $\Rightarrow$ 3D curve added to added to M-list

In order to specify 3D models by sketching, a fundamental operation is to specify a 3D curve by tracing over it in a single image. This allows the user to specify either a curve in space, or more commonly the boundary of a NURBS surface, through a single interaction. A curved line drawn by the user on an image does not, of itself, contain sufficient information to recover the required 3D scene curve. By exploiting the camera parameters recovered through the structure-from-motion process, however, the intended 3D shape can be recovered using image data.

The 2D drawn curve has fixed (known) length and thus can be modelled in parametric form as $\mathbf{l}(u) = (x(u), y(u))^\top$ with $0.0 \leq u \leq 1.0$ such that each allowable value of u specifies a location along the curve. A point $\mathbf{l}(u)$ thus specifies a ray projected into scene space which intersects the 3D scene curve at least once. The points in scene space at which this intersection occurs may be identified by the distance along the projected ray at which they occur. We call this distance d.

In order to simplify the problem of identifying the required scene curve a set $\mathcal{U}$ of 100 equally spaced points u are distributed along the length of the drawn curve. Similarly, a set $\mathcal{D}$ of 500 potential values for d is selected on the basis of the range of distances from the camera centre at which reconstructed points occur. A line $\mathbf{l}$ and a $\{u, d\}$ pair specifies the location of a point in scene space which we label $\mathbf{L}(u, d)$. A 3D scene curve $\mathbf{L}$ is thus fully specified by identifying the set $\mathcal{D}_u = \{d_1, \ldots, d_N\}$ of intersection distances for every allowable u along a drawn curve $\mathbf{l}$.

If a hypothesised 3D point $\mathbf{L}(u, d)$ is part of $\widehat{\mathbf{L}}$ then it must lie on the surface of an object in the scene. If we label the image patch surrounding the projection of the 3D point $\mathbf{P}$ into image $\mathbf{I}$ as $\mathbf{I}(\mathbf{P})$, then the normalised cross correlation between the drawing image patch $\mathbf{I}_d(\mathbf{P})$ and the corresponding target image patch $\mathbf{I}_t(\mathbf{P})$ is $C(\mathbf{I}_d(\mathbf{P}), \mathbf{I}_t(\mathbf{P}))$. If the image set over which this correlation measure is performed is labelled $\mathcal{I} = \{\mathbf{I}_k\}, k = 1 \ldots K$ then the total correlation measure for $\mathbf{P}$ in $\mathcal{I}$ is

$$\mathcal{J}_C(\mathcal{I}, \mathbf{P}) = K^{-1} \sum_{k=1}^{K} C(\mathbf{I}_d(\mathbf{P}), \mathbf{I}_k(\mathbf{P})). \quad (1)$$

Let $\Delta\mathcal{P}$ represent the image intensity gradient of an image patch $\mathcal{P}$. On the basis of the assumption that edges in the scene will correspond to edges in the images we construct image gradient cost function for the point $\mathbf{P}$ by

$$\mathcal{J}_\Delta(\mathcal{I}, \mathbf{P}) = K^{-1} \sum_{k=1}^{K} \Delta\mathbf{I}_k(\mathbf{P})^{-1}. \quad (2)$$

The total cost associated with each curve is constructed from the weighted sum of these measures calculated for each point along the length of the curve:

$$\mathcal{J}(\mathcal{I}, \mathbf{L}) = \sum_{u \in \mathcal{U}} \sum_{d \in \mathcal{D}_u} (\alpha \mathcal{J}_C(\mathcal{I}, \mathbf{L}(u, d)) + \beta \mathcal{J}_\Delta(\mathcal{I}, \mathbf{L}(u, d))) \quad (3)$$

where $\mathcal{D}_u$ is the set of $d \in \mathcal{D}$ such that $\mathbf{L}(u, d)$ is an element of the curve $\mathbf{L}$ and the scalars α and β reflect the relative scales of $\mathcal{J}_C$ and $\mathcal{J}_\Delta$, and have been set at 0.7 and 0.3 respectively on the basis of testing with real image sequences.

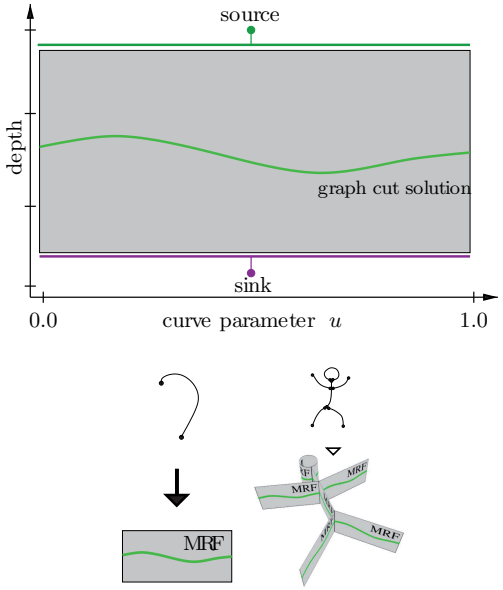

Figure 2. a) A single MRF corresponding to a single drawn curve such as that shown in b), c) shows multiple intersecting MRFs corresponding to a more complex problem involving fitting to multiple drawn lines.

We seek the continuous curve from $\mathcal{L}$ for which $\mathcal{J}(\mathcal{I}, \mathbf{L})$ is minimal. In order to identify the continuous curve with the minimal total cost efficiently, we use the max-flow[1] algorithm to find the lowest cost path through a graph. The graph is effectively a Markov Random Field (MRF) with pairwise costs determined by equation 3. One of the advantages of graph-based max-flow approach is that it allows 'S'-shaped cuts, which provides the flexibility required to represent the many-to-many mapping between u and d. The max-flow process typically requires approximately one second to reach a solution for a single curve.

Each node in the graph represents a $\{u, d\}$ pair selected from $\mathcal{U}$ and $\mathcal{D}$ respectively (see Figure 2). The nodes are arranged in a lattice structure with increasing values of u along the positive x-axis and increasing values of d along

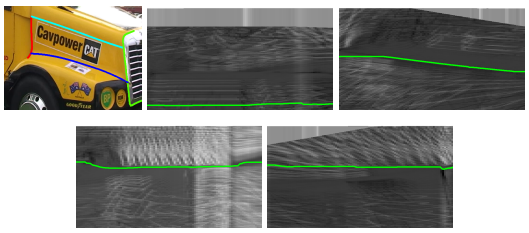

Figure 3. Four curves drawn in one image of the sequence, and the resulting graphs showing the lowest cost paths.

the positive y-axis. Each node is connected to 4 neighbouring nodes, with the weights on the links in the positive y directions set to $\mathcal{J}(\mathcal{I}, \mathbf{L}(u,d))$ and in x direction set to $\gamma\mathcal{J}(\mathcal{I}, \mathbf{L}(u,d))$. Source and sink nodes are connected by zero-weight links to the top and bottom rows of nodes respectively.

The max-flow algorithm finds the continuous cut through the graph which minimises the sum of the weights of the severed links. This sum is approximately equivalent to the value of $\mathcal{J}(\mathcal{I}, \mathbf{L})$. In fact the sum of weights of cut links equals $\mathcal{J}(\mathcal{I}, \mathbf{L})$ if the cut is parallel to the x-axis. The solution is biased against cuts parallel to the y-axis, which means that it is biased against solutions which ascribe multiple d values to the same u and against solutions which have large changes in d. The value of γ can be used to control the influence of this bias. A higher value of γ gives smoother solutions. The final 3D NURBS curve is generated on the basis of the graph cut solution.

3.2. Multiple intersecting 3D curves

The production rule for this process is

> **3D curves from lines** {Line group in I-list}[Drawing mode] $\Rightarrow$ 3D curves added to added to M-list, replacing previous estimates

Curves exist as modelling primitives in their own right, but are most useful in the role they play in specifying the boundaries of surfaces. In order to effectively determine the boundary of a surface, it must be possible to specify that two 3D curves intersect at a particular point. For the purposes of the interface we assume that curves which intersect in the image in which they are drawn are intended to intersect in scene space. We do not assume that these intersections occur at the beginning or end points of curves.

Two drawn curves $\mathbf{l}_1(u_1)$ and $\mathbf{l}_2(u_2)$ which intersect have an image point in common. The fact that the corresponding scene curves $\mathbf{L}_1$ and $\mathbf{L}_2$ also intersect means that there are points on each curve such that $\mathbf{L}_1(u_1, d_1) = \mathbf{L}_2(u_2, d_2)$ and therefore that at the intersection point $d_1 = d_2$. If we are to estimate the 3D scene curves $\mathbf{L}_1$ and $\mathbf{L}_2$ using the graph-based method described above this implies a constraint on the allowable set of solutions. The constraint is that if $\{u_1, d\}$ forms part of cut partitioning the first graph then $\{u_2, d\}$ must form part of cut partitioning the second, and vice versa.

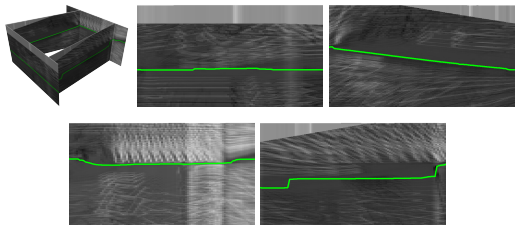

Figure 4. The combined graph generated by considering the four intersecting curves from Figure 3 collectively and the resulting graphs showing the lowest cost paths. Note the differences in minimal cost paths from those in Figure 3

In order to estimate both curves simultaneously while enforcing the constraint we join the two graphs. A set of zero-weight links is inserted between the nodes corresponding to u_1 in the first graph and u_2 in the second which share the same value of d. Note that this does not assume that the sets of joined nodes represent the first or last columns of either graph. This process of joining nodes corresponding to intersecting curves is repeated for every intersection, and the combined problem solved using the max-flow algorithm. Generating a solution typically takes approximately as many seconds as there are curves in the combined graph. A symbolic example of the merging process is shown in Figure 2, and Figures 3 and 4 show the graphs associated with a real example.

3.3. Matching pairs of drawn curves

The production rule for this process is

> **3D curve from 2 lines** {Pair of drawn lines in different images}[Drawing mode] $\Rightarrow$ 3D curve added to added to M-list

In some circumstances the process described above for generating a 3D curve from a single drawn 2D curve fails. This sometimes occurs in the presence of significant occlusion, or when modelling a shape which is not visible in the image set. For example, the cross section of the padded section of the chair shown in Figure 6 is difficult to determine automatically, but can be specified manually with ease. In the event that the automatically detected curve does not match the user's intention, another view of the same scene curve may be drawn in another image to specify its shape.

Generating a 3D curve on the basis of a pair of drawn curves requires that each point along each drawn curve is matched to its corresponding point on the other. The same many-to-many problem occurs as as in the single drawn case above, but in addition to this, it would also be unreasonable to expect the user to draw the two curves so as to have exactly the same beginning and end. The full length of the shorter curve must be used however, solutions which match only parts of the curve will not suffice. The problem is again suitable for formulation as an graph to which the max-flow algorithm might be applied. The details of the graph formulation and the max-flow solution are presented in [6].

3.4. Curves and surfaces

The process by which a set of lines is used to generate a NURBS surface is that described in [6] for generating a Coons surface. The method outlined in Section 3.2, however, provides an improved means by which such surfaces may be specified, requiring only that the user trace the outline of the surface in one image. The NURBS fitting process utilises points from the reconstructed point cloud where possible, but is particularly useful in the case where no such points are available. This is likely to occur in the case where the object being modelled does not exhibit significant surface texture. Any automated means of reconstructing surface shape will fail in this case. Allowing the user to specify the shape of an object simply by modelling its boundaries often overcomes this problem, as most shapes can be divided into sets of abutting NURBS surfaces suitable for specification by this process. The production rule for this process is

> **3D surface from line group** {Closed line group}[Surface mode] $\Rightarrow$ 3D surface added to added to M-list

The chair shown in Figure 6 is a particular example of this process. By tracing the edge of the curved arm rests and extruding it has been possible to model these complex but feature-point free surfaces. Similarly, by tracing the two long edges and top of the padded section of the chair in one image each it was possible to record the majority of the shape information. Generating a profile for the bottom end of the padded section required drawing the same curve in two images, as the images contain insufficient information to achieve this task from a single drawn curve. The result of these 5 interactions is the NURBS surface which was rendered to produce Figure 6.

3.5. Planar Surfaces

If the user sketches a face boundary using only straight lines, it is assumed that that face is planar. Estimation of a planar face can be considered a special case of curved surface estimation, but for the sake of efficiency it is carried out using a simpler method [6]. This fitting and projection process is effectively instant, which allows the user to select another image in which to edit the shape at will. Editing in another image allows manipulation of the 3D placement of the planar surface, but also modification of its boundary. In particular it is possible to turn a straight boundary into a curve by editing. The drawn curve replaces the boundary of the planar surface which was previously specified by the line. If the line to be replaced forms part of the boundary of more than one plane, then the plane with normal closest to the viewing direction has its boundary replaced by the planar curve. The remaining adjacent planar surfaces have their boundaries adjusted on the basis of this planar curve. This adjustment is typically out of the plane of the surface, which is then converted to a set of independent triangles.

3.6. Extrusion

> **Extrude** {Drawn extrusion line}[Extrude mode] $\Rightarrow$ Faces added to M-list to extrude selected surfaces

The extrusion operation creates new model structure based on part of the model that already exists. It is particularly useful for modelling solid objects like cars, that are essentially a planar profile with depth. Extrusion takes a planar set of faces and creates a duplicate of those faces, separated from the original faces by a distance specified by the user. If image information is available, the new faces are fitted to the images. However this operation can also be used to complete an extruded model even if the other side of it is not visible in any image. Figure 5 shows the use of extrusion to model the bumper on the front of a car. Initially, the lower edge of the bumper is traced out. The front face of the bumper is then modelled by extruding this lower edge upwards with a single mouse stroke. Finally, the top of the bumper is modelled by extruding the top edge inwards, again with another mouse stroke.

3.7. Mirroring

Mirroring is a powerful tool for modelling parts of a scene not visible in the image set, allowing visible geometry and texture to be duplicated in order to model non-visible scene aspects. The details of the mirroring process are provided in [6], where a similar idea was applied in order to simplify the process of modelling identical parts of an object. This technique has been extended here to facilitate the modelling of non-visible aspects of an object by allowing more complex geometry to be mirrored, and critically, for mirrored geometry to retain its original texture. This allows texture captured from the image set to be easily replicated

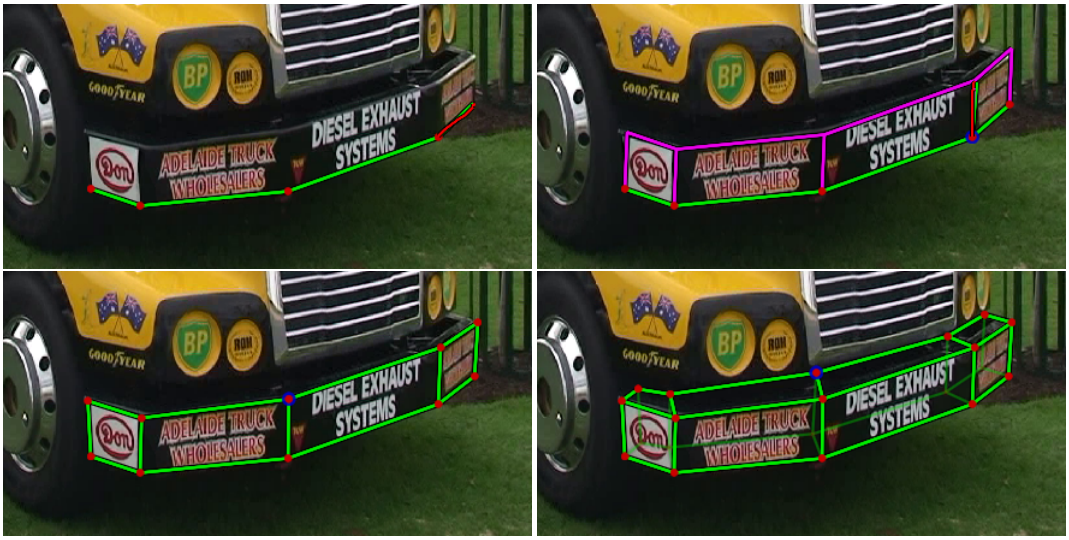

Figure 5. Using extrusion to model a bumper bar. Initially the lower edge of the bumper is traced, and snapped to image edges. Two extrusion operations are then used: first the lower edge of the bumper is extruded upwards, and then the upper edge is extruded inwards. This involves just 2 mouse strokes.

over non-visible aspects of the model. The production rule for this process is

> **Mirror** {Drawn mirror lines}[Mirror mode] $\Rightarrow$ All geometry in M-list (and associated texture information) duplicated

4. Results

The system has been used to model a variety of scenes containing objects that are only partly seen, and cannot be completely reconstructed using automated methods. The chair, shown in Figure 6, presents two main problems for automatic reconstruction: only its front and sides are seen, and its surface contains few identifiable features. However by tracing out one side of the chair, and then extruding it, we are able to obtain a complete model that can be rendered convincingly from any viewpoint.

The truck, also shown in Figure 6, has a more complex shape, and requires several sketching operations to create, extrude and mirror faces in order to model it. The complete set of operations is shown in the video submitted with this paper. Again only one side of the truck, and its front, are visible, and yet the completed model is accurate when viewed its far side.

To test the accuracy of this method, we modelled a cube with an edge length of 150mm on a tabletop. The camera was placed approximately 100 cm from the object, had a vertical FOV of 23 degrees, and image resolution 720x576 pixel. We traced the egdes of the cube in one view and evaluated the estimated 3D reconstruction by projecting the corner points of the reconstructed cube into another view. This second view had a baseline distance of 37 cm to the first view. After sketching the cube and projecting it into the other view 15 times, the standard deviation of the projected reconstructed corner points about the ground truth corner points was 1.8552 pixel, which corresponds to approximately 1.4 mm, or less than 1% of the cube's size.

5. Conclusion

Image based modelling has progressed remarkably in the last two decades, but has yet to realise its goal of fast, reliable creation of photo-realistic 3D models from video. This paper has presented a different approach towards that goal, where user operations are informed by an automatically obtained structure and motion estimate. Operations within the system are defined by rules, that specify their input, preconditions and output. A fundamental operation is the conversion of a 2D sketched stroke into a 3D primitive, allowing the creation of 3D structure by sketching on an image. By combining rules this system is able to rapidly create complex and convincing models in a variety of situations where purely automated systems cannot.

Figure 6. Completed models of a chair and truck, both showing detail not recoverable from the original sequences alone.

References

[1] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[2] R. I. Hartley and A. Zisserman. *Multiple View Geometry*. Cambridge University Press, 2000.

[3] D. Robertson and R. Cipolla. An interactive system for constraint-based modelling. In *Proc. 11th British Machine Vision Conference*, pages 536–545, 2000.

[4] P. Sturm and S. Maybank. A method for interactive 3d reconstruction of piecewise planar objects from single images. In *Proc. 10th British Machine Vision Conference*, pages 265–274, 1999.

[5] C. Taylor, P. Debevec, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. *ACM SIGGraph, Computer Graphics*, pages 11–20, 1996.

[6] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. Torr. Videotrace: Rapid interactive scene modelling from video. In *SIGGraph 2007, to appear*, 2007.

[7] J. Wang, P. Bhat, A. R. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. In *Proceedings of SIGGRAPH*, 2005.

[8] M. Wilczkowiak, P. Sturm, and E. Boyer. Using geometric constraints through parallelepipeds for calibration and3d modeling. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(2):194–207, February 2005.