

VideoTrace: Rapid interactive scene modelling from video

Anton van den Hengel^{1*} Anthony Dick¹ Thorsten Thormählen¹ Ben Ward¹ Philip H. S. Torr^{2*}
¹University of Adelaide, AUSTRALIA <http://www.acvt.com.au> ²Oxford Brookes University, UK
<http://cms.brookes.ac.uk/staff/PhilipTorr/>



Figure 1: Left to right: a frame from the input video sequence, a partial tracing of the model, the final model overlaid on the video, and the result of rendering the final model back into the original sequence.

Abstract

VideoTrace is a system for interactively generating realistic 3D models of objects from video—models that might be inserted into a video game, a simulation environment, or another video sequence. The user interacts with VideoTrace by tracing the shape of the object to be modelled over one or more frames of the video. By interpreting the sketch drawn by the user in light of 3D information obtained from computer vision techniques, a small number of simple 2D interactions can be used to generate a realistic 3D model. Each of the sketching operations in VideoTrace provides an intuitive and powerful means of modelling shape from video, and executes quickly enough to be used interactively. Immediate feedback allows the user to model rapidly those parts of the scene which are of interest and to the level of detail required. The combination of automated and manual reconstruction allows VideoTrace to model parts of the scene not visible, and to succeed in cases where purely automated approaches would fail.

CR Categories: I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Surface Fitting; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages

Keywords: Image-Based Modelling, Model-Based Reconstruction, Structure-from-motion

1 Introduction

*This work was funded by ARC Discovery Grant DP0558318, and EP-SRC EP/C006631/1(P).

The recovery of 3D models from video has for a number of years been a goal of both the computer vision and graphics communities. In computer vision, several systems have been developed to automatically recover a cloud of 3D scene points from a video sequence (e.g. [Pollefeys et al. 2004]). However these are vulnerable to ambiguities in the image data, degeneracies in camera motion, and a lack of discernible features on the model surface.

These difficulties can be overcome by manual intervention in the modelling process. In the extreme case, a modelling package such as Blender3D can be used to build a model manually, but it is difficult and time consuming to create a photorealistic result by this process. A more appealing option is to use all of the information that can be derived from the video using computer vision techniques to inform and accelerate an interactive modelling process.

The question then arises: how should these interactions be implemented so they are (a) intuitive to a non-expert user and (b) powerful and informative to the underlying modelling engine, so that only a small number of interactions are required? The VideoTrace system supports a novel sketch based interface that allows a user to create a 3D model by simply tracing out its structure as it appears in one or more frames of video (see Figure 1). Polygonal and curved surfaces, curved lines and extrusions can be modelled through these 2D interactions. This is made possible by the use of automatic reconstruction techniques from computer vision to assist in interpreting these 2D interactions as steps in constructing a 3D model.

VideoTrace is novel in that it provides a new interactive method for generating a surface-based reconstruction of general objects on the basis of a video sequence. Tracing over video is a flexible, intuitive, and efficient means of specifying 3D shape. The sophisticated exploitation of 3D information automatically obtained through image-analysis techniques enables VideoTrace to perform the otherwise impossible translation from traced outlines to realistic 3D models.

1.1 Previous work

Several existing systems combine image data and user interaction to create 3D models. However, because they do not make use of structure and motion analysis, they typically require a significant amount of user input. Photomodeler [Eos Systems 2005], for ex-

ample, allows 3D models to be created by marking up structure in one or more images, and requires that measurements from the scene and the cameras be input manually. The Facade system [Taylor et al. 1996] reconstructs architectural scenes as a collection of polyhedra. However Facade requires the user to outline each block in each image, and manually label corresponding features—a time consuming process. Wilczkowiak [2005] presents a more general approach to interactive modelling based on parallelepipeds as scene primitives. However this still requires the corners of the primitives to be marked manually in each image.

Quan et al. [2006] have developed an interactive method of modelling plants from image sets which uses strong priors to develop models of this particular class of objects. The prior in this case is specific to the particular variety of plant being modelled. Other work, including the use of architectural grammars [Dick et al. 2004; Müller et al. 2006] and the hair modelling of Wei et al. [2005], makes extensive use of prior information, but is too slow to be used interactively.

SmoothSketch [Karpenko and Hughes 2006] provides an intuitive sketching-based interface for creating models of smooth 3D shapes. Models are generated purely on the basis of user input, as no image-based information is used.

VideoTrace can be seen as an interactive means of upgrading the point-based reconstruction generated by systems such as boujou (by 2d3), the Voodoo Camera Tracker [Thormählen 2006] and PF-Track (the successor to ICARUS [Gibson et al. 2003]) to a surface-based reconstruction, and of correcting the results of such systems when they fail. It is not intended to replace 3D modelling packages, but rather to assist in the case of modelling an object as it appears in a video sequence. In the fusion of automated vision methods with manual interaction it follows on from previous work of the authors [van den Hengel et al. 2006] and is analogous to [Agarwala et al. 2004] which builds on visual tracking to drive the creation of 2D animations.

2 Video pre-processing

Before any interactive modelling takes place, structure and motion analysis [Pollefeys et al. 2004] is carried out on the video sequence. Structure and motion analysis is a computer vision technique that automatically reconstructs from a video sequence a sparse set of 3D scene points and the camera parameters which describe the relationship between the camera and the scene. We use the Voodoo Camera Tracker [Thormählen 2006]. The resulting 3D point cloud can be seen overlaid on a frame of the input video in Figure 3. This pre-processing enables VideoTrace to give 3D meaning to 2D interactions performed on the video.

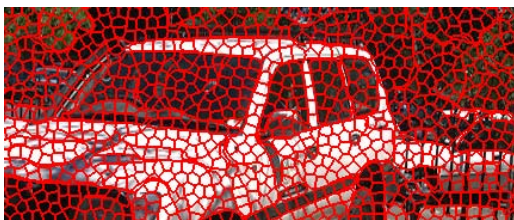


Figure 2: Superpixel segmentation of a section of the input sequence.

Additionally, each frame of video is segmented using a deliberately low threshold, so that the output is a large number of small clusters of adjoining pixels, where all pixels belonging to each cluster have a similar colour (see Figure 2). All 2D image operations are based

on these “superpixels” [Ren and Malik 2003] rather than the raw image pixel grid. This accelerates region clustering and edge finding operations, since only super-pixels as a whole and super-pixel boundaries need be considered.

3 Interactions

In this section we describe the modelling primitives provided by the system, and the types of interaction that are made possible by sketching. We illustrate their application by building a model of a car. The video sequence was captured using a hand-held consumer-grade video camera. At any time during modelling, the user can choose to trace on any frame of video in which the object is visible. By default, traced lines and curves are automatically refined in 2D by fitting to local strong superpixel boundaries, so a user requires no artistic ability, nor is “pixel perfect” tracing required.

3.1 Tracing polygons

To model a polygonal face, a user traces out its boundary in a video frame as a set of line segments. Nearby lines are automatically joined using the endpoint clustering method of [Shpitalni and Lipson 1997]. Each closed set of line segments that does not contain any internal lines is then used to generate an object face. This fit can be overridden if necessary by manually moving line endpoints, re-drawing lines, and drawing new lines on existing faces. An example of this interaction is shown in Figure 3.

The user can then navigate to another frame of the video to refine this model. The outline of the current 3D shape of the model is projected into this new frame to assist the user in making adjustments. In order to perform this projection a 3D model is calculated on the basis of all interaction thus far, as described below. The user can then drag line endpoints or lines in this new view so that they trace out the boundaries of the same polygons as in the initial view. These interactions are used to refine the 3D model so that it fits the projections in both frames.

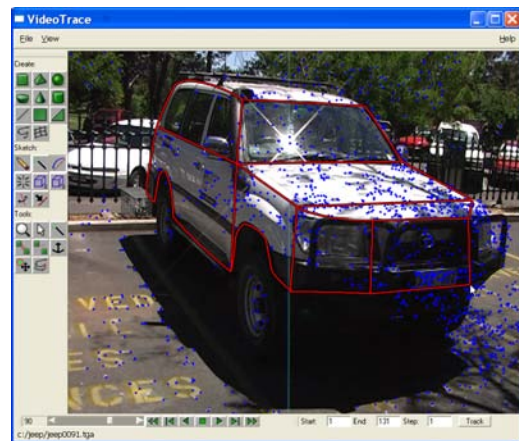


Figure 3: Traced lines (red) and reconstructed feature points (blue) drawn over a frame of the input sequence.

In the background, VideoTrace repeatedly re-estimates the 3D shape and position of the model in order to support these interactions. An initial estimate of the 3D position and orientation of each face is obtained by fitting planes to the set of reconstructed 3D points whose images lie within the area bounded by the user drawn line segments. To ensure robustness to 3D points that do not belong to the plane, this is done by fitting multiple planes to subsets of the

points. This process is carried out for all faces of the model, to generate a set of hypothesised planes for each. These sets of planes are then combined to form a set of hypothesised object models, each of which is tested against the image data as follows.

Support for each hypothesised model is initially calculated by measuring the distance from each face to the corresponding set of reconstructed 3D points. The support for the model is taken as the sum of the support values measured for each face. Models with sufficient support are subsequently evaluated on the basis of 2D image information. Each face in the hypothesised model is projected into the frames of video with which the user has interacted. The RGB values of the pixels in each frame that lie within each projected face in the model are accumulated to form a colour histogram for each face in each image. The difference between the histograms for corresponding faces in different images, summed over all faces, is used as an error measure for the 3D model hypothesis. As it is more computationally intensive, this measure is only calculated for hypothesised models which score well on the 3D distance measure.

The fitting process occurs in real time. This allows the user to switch between images from the original sequence naturally, selecting the most appropriate view of the scene at each stage. Through this process the user rapidly develops an accurate 3D model of the object through a set of intuitive 2D interactions. The model can be rendered using texture maps obtained from frames of the video [Niem and Broszio 1995].

3.2 Tracing curves

VideoTrace includes two modelling modalities which are based on Nonuniform Rational B-Splines (NURBS). NURBS curves allow the user to define a curved line, in 3D, through the scene space, whereas NURBS surfaces specify a 2D curved surface (also through 3D scene space).

NURBS curves can be specified either as a free-standing 3D object, or as one of the edges of a polygon. In order to define a free-standing curve the user draws a line over one of the images of the original video. A second image is selected and another line drawn. Both line drawing operations typically correspond to tracing the projected shape of the object through an image.

In order to estimate the curve, the problem is posed as a graph-based optimisation process, and the max-flow [Boykov and Kolmogorov 2004] algorithm used to find the best solution. Let the lines drawn in each of the images be L_1 and L_2 and define each in parametric form such that

$$\begin{aligned} L_1(u_1) &= (x_1(u_1), y_1(u_1))^T & \text{with } 0.0 \leq u_1 \leq 1.0 & \text{ and} \\ L_2(u_2) &= (x_2(u_2), y_2(u_2))^T & \text{with } 0.0 \leq u_2 \leq 1.0 & . \end{aligned} \quad (1)$$

Each line-of-sight through a point $(x_1(u_1), y_1(u_1))^T$ on $L_1(u_1)$ in the first image, defines a ray when projected into the second image. By calculating the intersection of this projected ray with $L_2(u_2)$ we can identify a correspondence between u_1 and u_2 . The set of correspondences defines the reconstructed curve. The projected ray may, however, intersect $L_2(u_2)$ multiple times, and may also be tangential to $L_2(u_2)$ in places. The correspondences between u_1 and u_2 may thus be many-to-many.

In order to represent the problem as a graph, both L_1 and L_2 are sampled regularly along their lengths. Each (u_1, u_2) pair is allocated to a row and column of the graph respectively. Each node in the graph thus represents a correspondence between a particular value of u_1 and u_2 , and thus identifies a potential point on the reconstructed intersection curve. Figure 4 shows an image pair with traced lines, and the corresponding graph. Each node in the graph

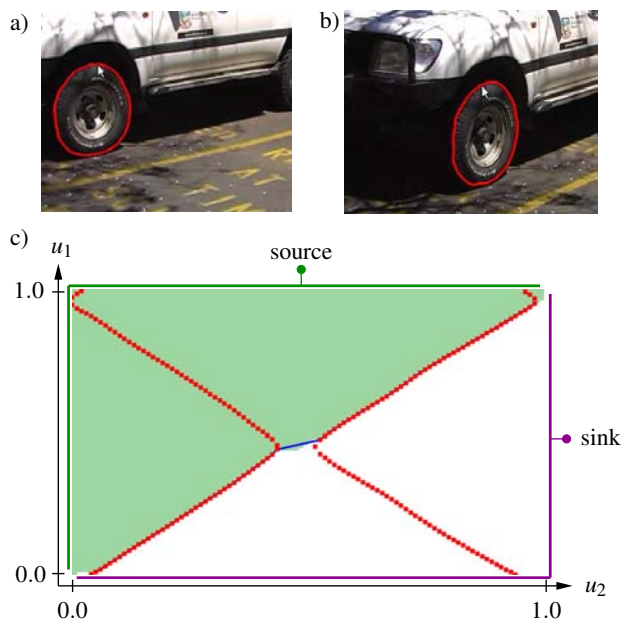


Figure 4: User interaction tracing out a curve over an image pair (a and b), and the resulting graph (c).

is connected to its 4 closest neighbours, and all edge weights set to 1 initially. Each node which represents the intersection of a ray and a curve has all of its edge weights set to 0 (these nodes are red in the graph in Figure 4). The left and upper boundary nodes of the graph are connected to the source by an edge with weight 0, and the right and bottom boundaries to the sink similarly. The optimal cut through the graph partitions the nodes along the minimal cost path, which corresponds to the minimal cost intersection curve. This method is fast enough to be used interactively, and flexible enough to deal with missing data and matching ambiguities. The missing data problem arises particularly when the start and end points of the curve are not the same in the two images, and when the 3D curve is partially occluded in one image.

NURBS curves can also be used to refine the edges of polygonal surfaces. If the line is drawn over an existing edge of the polygonal model, a 3D curve is generated by projecting the 2D image line onto the corresponding 3D plane. If the existing edge represents the boundary between two neighbouring polygonal surfaces the curve is projected onto the surface with normal that aligns best with the viewing direction. The opposing neighbouring surface is altered so as to have the same boundary, and is thus no longer necessarily planar.

3.3 Extrusions

Once a NURBS curve has been estimated using the above procedure, it is then possible to constrain it to lie on a plane if required. This is achieved by least median of squares fitting of a plane to a set of points sampled evenly along the length of the curve in 3D. The 2D NURBS curve is then generated by projecting onto the resulting plane. An extrusion is defined by dragging a 2D NURBS curve across an image (see Figure 5). The vector (in scene-space) along which the shape is extruded is defined by the normal to the plane in which the curve lies. The extent to which the shape is extruded is determined by projecting the vector defined by the drag interaction onto the normal vector. The extrusion is updated as the user moves the cursor, allowing accurate alignment with respect to the current

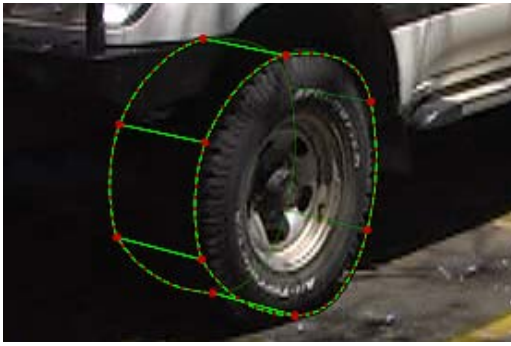


Figure 5: Creating an extrusion from a NURBS curve.

frame.

NURBS curves, whether 2D or 3D, can also be fattened, to model pipes and similar. The degree to which a particular curve is widened is controlled by dragging across the image perpendicular to the objects axis of symmetry.

3.4 NURBS Surfaces

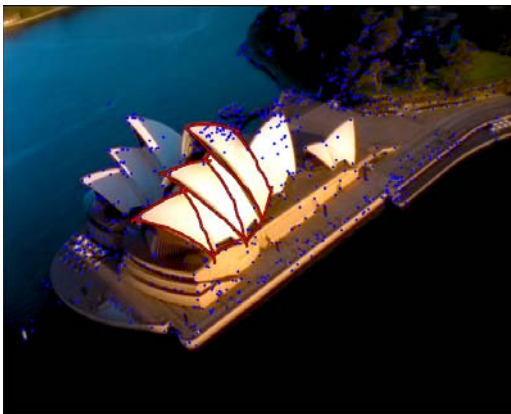


Figure 6: Generating NURBS surfaces from boundary curves in the absence of reconstructed object points.

Once multiple NURBS curves are reconstructed they can be used to generate NURBS surfaces. For example, a 1-rail sweep surface can be constructed from two curves by sweeping a section curve along a rail curve, a 2-rail sweep surface be generated from three curves by sweeping a section curve along two rail curves, or a Coons surface can be interpolated from four compatible boundary curves [Piegl and Tiller 1997]. In many cases the quality of the generated surfaces is very good (see Figs. 6 and 7). However, if reconstructed object points are available, the surface can be refined by fitting the surface to the object points, as shown in Fig. 8.

A Levenberg-Marquardt optimiser uses the existing surface as a starting point and minimizes the distance between the object points and the surface by altering the control points of the NURBS surfaces. The number of control points in the NURBS surface must be selected so as to reflect the number of data points available. This is achieved using the method in [Piegl and Tiller 1997] which generates a control point between every neighbouring pair of data points. Note that control points on the boundary curves remain unchanged during the optimization process.

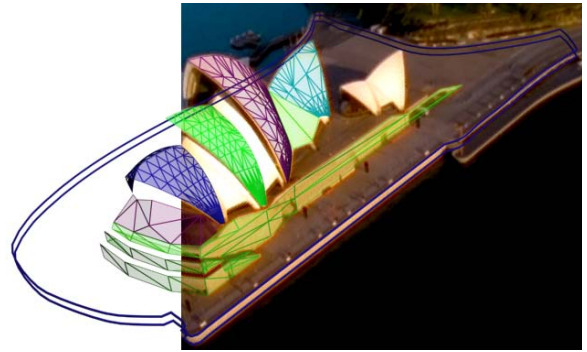


Figure 7: NURBS surface fitted in the absence of reconstructed object points.



Figure 8: NURBS surface fitted to reconstructed object points.

4 Mirror planes

Although tracing out surfaces is fast and intuitive, it has limitations. Because it involves tracing over image edges, it can only be applied to parts of an object that are visible in the video. Also, because it estimates each surface separately, it may fail to capture global properties of the structure such as regularity and symmetry.



Figure 9: User interaction identifying location of mirror plane (left, sketched mirror line in blue). Model after mirroring showing match with original video (right).

To address these issues we incorporate extra interactions that apply to sets of surfaces. For example, drawing a mirror plane is an intuitive way to build a complete model of an object which is symmetric about a plane. The interaction involves selecting a mesh of surfaces that have already been traced and are to be mirrored. Lines are then traced along surfaces to indicate their intersection with the mirror plane (see Figure 9). Mirrored versions of the selected surfaces are then added to the model where necessary to create a symmetric object.

VideoTrace uses the mirror lines to estimate a plane about which to mirror the selected surfaces. The midpoint of each surface edge that intersects a mirror line is calculated. It is assumed that at least

a subset of these midpoints belong to the mirror plane. Potential mirror planes are generated by choosing sets of 3 such midpoints, forming a plane from them, and mirroring the selected geometry about that plane. Each potential plane is evaluated by measuring the total distance from the reflected copy of the vertices to the previously modelled vertices on the far side of the mirror plane. The plane that minimises this error is chosen, and then optimised using the same measure. Once it has been optimised, the existing part of the model on the far side of the mirror plane is replaced by a reflected version of the model on the near side of the plane. This includes the vertices on the far side of the surfaces on which the mirror lines were drawn.

5 Discussion and conclusion

VideoTrace has been successfully evaluated on a variety of video sequences. Because it makes use of structure and motion analysis, it inherits some well known limitations in situations such as degenerate camera motion and completely planar scenes. However, despite these limitations, structure and motion based systems are still widely used in major post production companies. We have also mitigated some common errors by using robust surface fitting methods, so that for instance reflective surfaces can still be modelled despite containing some misplaced feature points.

VideoTrace makes it easy to create 3D models from video, which can then be used in a variety of tasks (see, for example, Figure 10). It supports a range of simple 2D interactions with frames of video, including tracing out lines and curves, sweeping and extruding lines and curves to form surfaces, and mirroring shapes about planes. By combining these interactions with camera and sparse 3D scene information, it is able to construct a 3D model of a scene with little user intervention. By their interactions the user can control which parts of the scene are modelled, and to what detail, and can even model parts of the scene that are not visible in the video. By making use of 3D information where it is available, but allowing the scene to be modelled by tracing even where it is not, VideoTrace combines the benefits of automatic and manual modelling systems.



Figure 10: Inserting a synthetic object into video, using the known motion of the camera (see video attached to this paper). As the 3D shape of the car is known, it can interact convincingly with the object.

References

- AGARWALA, A., HERTZMANN, A., SALESIN, D. H., AND SEITZ, S. M. 2004. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graphics* 23, 3, 584–591.
- BOYKOV, Y., AND KOLMOGOROV, V. 2004. 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.
- DICK, A., TORR, P., AND CIPOLLA, R. 2004. Modelling and interpretation of architecture from several images. *International Journal of Computer Vision* 60, 2 (November), 111–134.
- EOS SYSTEMS, 2005. Photomodeler: A commercial photogrammetry product <http://www.photomodeler.com>.
- GIBSON, S., HUBBOLD, R., COOK, J., AND HOWARD, T. 2003. Interactive reconstruction of virtual environments from video sequences. *Computers & Graphics* 27, 2 (April), 293–301.
- KARPENKO, O. A., AND HUGHES, J. F. 2006. Smoothsketch: 3D free-form shapes from complex sketches. In *ACM SIGGraph, Computer Graphics*.
- MÜLLER, P., WONKA, P., HAEGLER, S., ULMER, A., AND GOOL, L. V. 2006. Procedural modeling of buildings. *ACM Transactions on Graphics* 25, 3, 614–623.
- NIEM, W., AND BROSZIO, H. 1995. Mapping texture from multiple camera views onto 3d-object models for computer animation. In *Proc. International Workshop on Stereoscopic and Three Dimensional Imaging*, 99–105.
- PIEGL, L., AND TILLER, W. 1997. *The NURBS book (2nd ed.)*. Springer-Verlag New York, Inc., New York, NY, USA.
- POLLEFEYS, M., GOOL, L. V., VERGAUWEN, M., VERBIEST, F., CORNELIS, K., TOPS, J., AND KOCH, R. 2004. Visual modeling with a hand-held camera. *International Journal of Computer Vision* 59, 3, 207–232.
- QUAN, L., TAN, P., ZENG, G., YUAN, L., WANG, J., AND KANG, S. B. 2006. Image-based plant modeling. *ACM Transactions on Graphics* 25, 3, 599–604.
- REN, X., AND MALIK, J. 2003. Learning a classification model for segmentation. In *Proc. 9th Int'l. Conf. Computer Vision*, vol. 1, 10–17.
- SHPIHALNI, M., AND LIPSON, H. 1997. Classification of sketch strokes and corner detection using conic sections and adaptive clustering. *Trans. of the ASME, Journal of Mechanical Design* 119, 1, 131–135.
- TAYLOR, C., DEBEVEC, P., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *ACM SIGGraph, Computer Graphics*, 11–20.
- THORMÄHLEN, T. 2006. *Zuverlässige Schätzung der Kamerabewegung aus einer Bildfolge*. PhD thesis, University of Hannover, related software 'Voodoo Camera Tracker' can be downloaded from <http://www.digilab.uni-hannover.de>.
- VAN DEN HENGEL, A., DICK, A., THORMÄHLEN, T., TORR, P., AND WARD, B. 2006. Building models of regular scenes from structure and motion. In *Proc. 17th British Machine Vision Conference*, 1:197–206.
- WEI, Y., OFEK, E., QUAN, L., AND SHUM, H.-Y. 2005. Modeling hair from multiple views. In *ACM SIGGraph, Computer Graphics*, ACM Press, New York, NY, USA, 816–820.
- WILCZKOWIAK, M., STURM, P., AND BOYER, E. 2005. Using geometric constraints through parallelepipeds for calibration and 3d modeling. *IEEE Trans. Pattern Analysis and Machine Intelligence* 27, 2 (February), 194–207.