In-Situ 3D Sketching Using a Video Camera as an Interaction and Tracking Device

Gilles Simon

In-Situ 3D Sketching Using a Video Camera as an Interaction and Tracking Device

Gilles SIMON, Université Henri Poincaré / INRIA project-team MAGRIT, Nancy, FRANCE

Abstract

In this paper, we present a novel method for in-situ 3D sketching of polyhedral scenes. A video camera is used as both an interaction and a tracking device, which makes the system particularly suitable for handheld devices such as PDAs and mobile phones. The efficiency and accuracy of the method are demonstrated using a miniature scene and a real outdoor scene.

Categories and Subject Descriptors (according to ACM CCS): Artificial Intelligence [I.2.10]: Vision and Scene Understanding—Modeling and recovery of physical attributes Computer Graphics [I.3.6]: Methodology and Techniques—Interaction techniques

1. Introduction

Acquiring the 3D geometry of arbitrary scenes has been a primary objective of both the computer vision and graphics communities for many decades. Applications are numerous in various domains such as construction, geographic information systems, robotics and augmented reality (AR). Existing modeling methods usually rely on two separate stages: first some data about the scene (photographs, videos, physical measurements, laser scanners, ...) are acquired on-site. Then these data are treated off-line using some specific manipulations and algorithms (e.g. [HDT*07]). Unfortunately, this process has two drawbacks: first, some of the involved tasks (data acquisition and/or geometry extraction from these data) can be very tedious. Secondly, it is not guaranteed that the desired structure is fully extractable from the acquired data and additional acquisitions are sometimes required in order to supplement the missing parts. In this paper, we propose to bridge the gap between data acquisition and their exploitation. We describe a purely image-based system for in-situ 3D sketching of polyhedral scenes. The built parts of the scene are immediately shown superimposed on the environment, which allows the user to verify the geometry against the physical world in real-time.

In-situ modeling has been introduced in [PT01] under the name of "construction at a distance". The principle is to allow mobile AR users to capture planar shaped objects from the physical world using simple 3D primitives and constructive solid geometry. The user interacts with the computer using a set of pinch gloves and hand tracking and the camera is tracked using an inertial sensor and a GPS; this system thus requires several special equipments and devices and is only suitable for outdoor use. By contrast, a purely image-based interactive model building system has recently been proposed in [BMC08]. This work has several common aspects with our work, such as the use of a camera-mouse and model-based tracking. However, the modeling task is less constrained in our system in that it does not require any initial template to be placed in the scene, nor that the current model partly stays in the camera field of view while new structures are added. Moreover, defining a vertex in [BMC08] is based on parallax motion which is not well appropriate for modeling objects far from the camera.

Finally, our system can be seen as an immersive version of the widely used 3D drawing software Google SketchUp™ (http://sketchup.google.com), whose principles are described e.g. in [OSD05]. This tool combines some of the features of pencil-and-paper sketching and some of the features of CAD systems to provide a lightweight, gesture-based interface for 3D polyhedral modeling. In the latest releases of SketchUp, the user is able to align the world axes to match a photo perspective. With this done, he can create models using the photo as a direct reference; mouse strokes

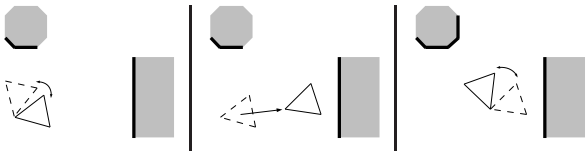


Figure 1: *3-DOF camera rotations (in modeling mode) alternate with 6-DOF camera motions (in tracking mode). Thick lines represent the modeled faces of the scene.*

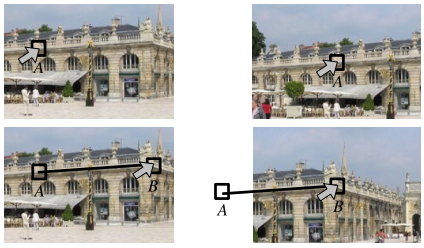


Figure 2: *Equivalence between moving cursor / fixed camera (left) and fixed cursor / moving camera (right).*

are converted into 3D-space using inverse ray intersections with the previously defined geometry (or the ground plane by default). All these principles have been taken up in our implementation, but with the important difference that we consider dynamic video images instead of static ones. Moreover, the video camera is used as the interaction device instead of a mouse, which makes our system particularly suitable for mobile devices such as PDAs, wearable computers and mobile phones.

The system alternates between two modes of operations (Fig. 1):

Modeling mode. Pure rotations are applied to the camera, e.g. when using a head-mounted display, this amounts to look around the scene by only turning the head. Using the camera as an interaction device, the user is able both to calibrate the camera and to define the scene geometry (section 2). It must be noticed that this mode may be used alone, providing already interesting contributions to classical single view metrology, as (i) by turning the head the reachable field of view is much larger than the camera's field of view and (ii) mouse manipulations are replaced by camera manipulations, which can be of great interest in a mobile context.

Tracking mode. When at least one face of the scene has been described and the camera undergoes a general motion, a 6-DOF (degrees of freedom) camera tracking is performed, based on the available geometry (section 3). This allows the user to get closer to some parts of the scene or make some new faces visible before continuing modeling. In this mode, the geometry previously modeled has to stay (at least partly) visible in the camera field of view, which is not required in modeling mode. A pose recovery procedure based on SIFT features can be called upon user request or each time a tracking failure is detected by the system.

The system starts in modeling mode. Once a 3D geometry has been initialized, switches between modeling and tracking modes are done automatically, depending on the motion applied to the camera (Akaike's motion model selection is used [VBS03]).

2. Modeling Interactions

A key idea of this work is that user interactions can be done using the video camera itself. Indeed, let us consider that the user wants to draw a line stroke on a video image between two physical points A and B of a real-world scene; there are two possible ways to do this (Fig. 2) : (i) the camera stays fixed and a mouse is used to move a cursor to the two respective endpoints; (ii) the camera is rotated so that the physical points apparently move to a fixed cursor, for instance at the center of the image. The second solution, which is used in our implementation, does not require mouse support. However, the position of point A has to be updated from frame to frame while point B is aimed at. Fortunately, camera rotations only induce homographic deformations of the image, that can be computed easily e.g. using keypoint matches. In both cases a click has to be done when the cursor (or the camera) is correctly positioned. In the second solution, mouse clicks can be replaced by key presses or any other input controls such as buttons, voice commands, etc. In our system, only three keys are used for all operations: one to "click" or "drag and drop" in the image, a second to cancel the current operation or request a pose recovery and a third to scroll the tools menu.

Before being able to model the scene, the user has to calibrate the camera: this is done by indicating two sets of horizontal parallel lines orthogonal to each other (Fig. 3, frame a). In addition, the user can drag and drop the origin of the world frame, providing a ray of possible 3D positions of that point. The depth of the origin is chosen so that the unit up vector has an arbitrary size in the image plane. The user can also change the scale of the scene by moving the extremity of the unit up vector. Once the camera parameters are known, the user can start modeling. New faces are instantiated in contact with existing faces or the ground plane by default. Contacts are guaranteed by snapping the clicked points to existing 3D points which appear in a specific color when close to the cursor: yellow for a face vertex, blue for the middle of an edge and red for the closest point on an edge. Faces under the cursor are also specifically outlined and new 3D vertices can be generated by using inverse ray intersec-

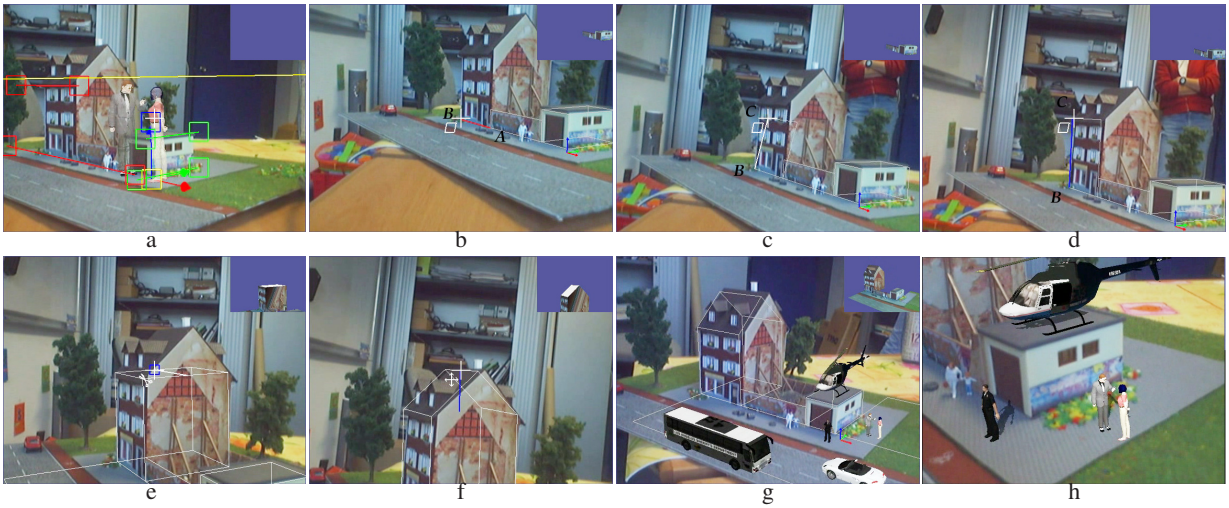


Figure 3: *Snapshots of the system in use.*

tion. In addition, most actions are guided using a line stroke, which itself can be snapped to the projected world axes.

Six different tools have been implemented which are summarized in Tab. 1. Fig. 3 illustrates how a simple house can be modeled using the Add, Extrude, Cut and Move tools: a rectangular face is first created using two line strokes AB and BC: according to the priority rules described at bottom of Tab. 1, AB is instantiated in the ground plane and parallel to the red axis (frame b); the vertex C is dragged in the plane orthogonal to AB (frame c) and finally such that BC is aligned with the blue axis (frame d); the new face is extruded forming a box and the top face of this box is subdivided so that the cut line joins the middles of two opposite edges (frame e); finally, the cut line is moved along the blue axis in order to form the roof (frame f).

	Add a rectangular face by clicking three points $A, B, C^{(*)}$. The rectangle is generated in the plane ABC using A and C as diagonally opposite vertices.
	Extract texture and SIFT features from the selected face.
	Extrude the selected face.
	Move the selected vertex, edge or face.
	Subdivide the selected face by joining two of its vertices.
	Delete the selected face.

$^{(*)}$ The inverse ray intersection priorities are: A: selected vertex > selected face > ground; B: selected vertex > projected world axis $\parallel AB$ > plane(A); C: selected vertex > projected world axis $\parallel BC$ > plane $\perp [AB]$

Table 1: *Modeling tools used in our prototype.*

3. Camera Tracking

Camera poses are computed using image content only and the modeled planar polygons [VBS03]: Harris corners are matched from frame to frame using normalized cross-correlation of patches in a search window; these corners are detected in the whole image for a 3-DOF tracking or inside the projected faces of the model for a 6-DOF tracking; outlier matches are rejected using a RANSAC computation of the planar homographies that best fit the sets of corners between the subsequent views; finally, the camera pose is computed by iteratively minimizing the sum of squared transfer errors (SSTE) of the inlier matches, using the homographies induced by the pose parameters and the equations of the involved planes.

This procedure is fast and robust but has two drawbacks: first, due to the recursive nature of the algorithm, errors induced by noise (e.g. in corner detection) accumulate over time. Secondly, as the size of the search window has to be bounded both to reduce matching ambiguities and to achieve real-time, camera tracking may fail in case of fast motions. To tackle the first problem, we use the texture information extracted by the user during the modeling mode. Like in [RD06], the current textured model is rendered from the current pose using the OpenGL graphic engine. Harris corners are detected in the rendered view and matched with the Harris corners of the current frame. These matches, which are merged with the set of matches obtained in the previous video frame, tend to reduce the drift. For performance issues, this correction procedure can be called only every k frames ($k = 2$ in our implementation).

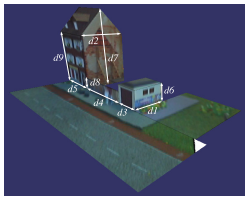
Tracking failures are detected automatically by testing whether the number of inlier matches falls under a threshold. The user is also able to manually require for pose recovery when he visually estimates that it is necessary. Pose recovery is basically based on SIFT feature matching [Low04]. SIFT features have a higher computational cost than Harris corners but are invariant to image scale and rotation and robust to some extend to change in viewpoint. When the "Extract texture" tool is used, SIFT features are extracted inside the

projected faces of the model and stored (in a kd-tree) associated with the current pose. When the recovery procedure is called, the list of textured faces is traversed until one face gets enough matches with the SIFT features of the current frame. If one face is found, it is put at the top of the list (so that it will be evaluated first at the next tracking failure) and a rough estimate of the current pose is obtained by minimizing the SSTE of the inlier matches. This rough pose is then used to render an image of the textured model and the correction procedure is called.

The recovery procedure slightly differs in modeling mode: when the system enters this mode, an initial set of SIFT features is detected in the whole image. These features are transferred from frame to frame using the RANSAC-estimated homographies. The image is subdivided into a 3x3 grid; when one square of the grid is empty after the SIFT features have been transferred, new SIFT features are detected inside this square and added to the current set of SIFT features (see the outdoor video). When a tracking failure occurs, SIFT features are detected in the current image and matched with the current set of SIFT features.

4. Experiments

All experiments were performed using a Dell Precision M6300 laptop coupled with a simple Logitech webcam. The system runs at video rate in standard mode and 2 to 6Hz when the recovery procedure is called. Two videos are associated with this paper, showing the system in action. A miniature scene is used to assess the accuracy of the method. Several modeling and tracking sequences alternate during the working session (Fig. 3). It can be distinguished in the video between when the system is running in modeling mode and when it is running in tracking mode, as the cross cursor disappears in the second case. Fig. 4 shows the errors obtained on the recovered 3D geometry (scaled so that the recovered distance d_1 is equal to the expected distance). These errors are acceptable for many applications: for instance, to prove that the recovered model is suitable to perform AR, we added a seventh tool to our platform, enabling the user to add virtual objects under the camera cursor. The added objects appear rigidly anchored in the scene, even for a large range of distances from the camera to the scene (Fig. 3, frames g,h).



	A(mm)	Δ(mm)	Δ/A(%)
d_1	61	Ref	Ref
d_2	88	-0.3	-0.3
d_3	38	1.5	3.9
d_4	92	4.2	4.6
d_5	75	-5.2	-6.9
d_6	34	-1.3	-3.8
d_7	139	-8.4	-6.0
d_8	21	0.8	3.8
d_9	95	-4.0	-4.2

Figure 4: *Recovered geometry with actual distances A (mm), absolute errors Δ (mm) and relative errors Δ/A (%).*

The modeling mode is also illustrated using a real outdoor scene (an athletism stadium with a changing house). The first part of the video shows how the set of SIFT features is expanded. Several abrupt motions are done, requiring the recovery procedure to be called. The camera is then calibrated and some modeling operations are performed before a billboard is added along the track (Fig. 5).

Figure 5: *One snapshot of the outdoor sequence.*

5. Conclusion

In this paper, we have proved that in-situ modeling is possible using a standard webcam as only device. The system can be improved in several ways. For instance, including edge features to the tracking algorithm [RD06] would help to handle such scenes like the athletism stadium where little texture information appears. Localization in larger areas would also require to make the recovery procedure more scalable and efficient. Such techniques like the Potentially Visible Sets [AWK*09] may be useful for that purpose.

References

[AWK*09] ARTH C., WAGNER D., KLOPSCHITZ M., IRSCHARA A., SCHMALSTIEG D.: Wide Area Localization on Mobile Phones. In *ISMAR 2009*.

[BMC08] BUNNUN P., MAYOL-CUEVAS W. W.: OutlinAR: an assisted interactive model building system with reduced computational effort. In *ISMAR 2008*.

[HDT*07] VAN DEN HENGEL A., DICK A., THORMÄHLEN T., WARD B., TORR P. H. S.: VideoTrace: Rapid Interactive Scene Modelling from Video. In *SIGGRAPH 2007 papers*.

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *Int. Jour. of Computer Vision 60*, 2 (2004).

[OSD05] OH J.-Y., STUERZLINGER W., DANAHY J.: Comparing SESAME and Sketching on Paper for Conceptual 3D Design. In *EUROGRAPHICS Workshop* (2005).

[PT01] PIEKARSKI W., THOMAS B. H.: Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer. In *ISWC 2001*.

[RD06] REITMAYR G., DRUMMOND T. W.: Going out: Robust tracking for outdoor augmented reality. In *ISMAR 2006*.

[VBS03] VIGUERAS F., BERGER M.-O., SIMON G.: Iterative multi-planar camera calibration: Improving stability using model selection. In *Vision, Video and Graphics* (2003).