Fast Freehand Acquisition of 3D Objects and their Visualization

Gabriele Peters and Klaus Häming

University of Applied Sciences and Arts Dortmund
Department of Computer Science
Visual Computing
Emil-Figge-Str. 42, D-44227 Dortmund, Germany
gabriele.peters@fh-dortmund.de
http://www.inf.fh-dortmund.de/personen/professoren/peters/

Abstract. In many applications 3d models of real-world objects are required. We introduce a tool which allows an untrained user to take three images of an object freehand with a simple consumer camera. From these images a 3d model of the visible parts of the object is reconstructed. From a research point of view we propose solutions for three weaknesses of the state-of-the-art reconstruction pipeline: an improved SIFT-based feature detection, a two-stage RANSAC process facilitating a faster selection of relevant object points, and a novel texture mapping. From a practice point of view we present a user-friendly tool for fast acquisition and realistic visualization of 3d models of real-world objects. We go into detail of the necessary user interaction and discuss usability aspects of the GUI. As it requires low-level technical equipment only and no technical knowledge of the user this tool may be attractive to a large number of people in the future.

Key words: Object Acquisition, Object Reconstruction, User Interfaces, Interaction Techniques, Usability

1 Introduction

There is a growing demand for realistic 3d models of real-world objects in a large number of fields of applications, such as the entertainment industry, design and construction, or human-robot interaction. One way to obtain such models is the cumbersome manual creation of 3d point clouds, meshes, and textures with a modelling software, which is still the prevalent method in game design. In contrast to such manual methods automatic methods of object acquisition can either be image-based or not. Among non-image-based methodes are the acquisition with CMM scanners (Coordinate Measuring Machine) [1] (which, in addition, requires mechanical contact with the object), laser scanning [2], and the application of structured light [3]. Image-based methods are, e.g., stereo vision or multi-camera techniques [4]. These methods require a complex technical

equipment, a highly skilled expert to operate it, or both. Overall, existing methods for object acquisition are far from being user-friendly or accessible to the broad public.

In this article we describe a prototype of an intuitive, image-based acquisition tool which allows an untrained user to take a few images of an object freehand from slightly different but random viewpoints with a cheap, uncalibrated consumer camera. From these images a 3d model of the visible parts of the object is reconstructed and can be used for further processing. The model is visualized in a realistic way and can be rotated by the user in the interface for further analysis and, if necessary, for improvement by taking further images into account.

We address both, research and practice. On the one hand, we explore the possibility to improve methods of image processing, reconstruction, and visualization. On the one hand, we transfer results of our research to a real-world application which has already attracted attention of the public in several demonstrations and a television broadcast. From a research point of view the contribution of this paper is threefold: First, on the level of image processing, we introduce a SIFT-based method (Scale-Invariant Feature Transform) for the detection of features in the source images. It exhibits a higher robustness in comparison to existing methods. Secondly, on the level of reconstruction, we propose a novel two-stage RANSAC (RANdom SAmple Consensus) process which facilitates a faster selection of those object points relevant for reconstruction. Thirdly, on the level of visualization, we introduce a fast mapping of triangles of image texture on the 3d model, resulting in a realistically rendered 3d surface. The research aspect of the object acquisition system is described in section 2.

From a practice point of view we present a user-friendly tool and interface which allows for the reconstruction and visualization of 3d models of real-world objects from a few snapshots within seconds. The snapshots can be taken freehand with an ordinary consumer camera. As it requires low technical equipment and no technical knowledge of the users this approach may have the potential - besides numerous applications in engineering and technique - to evolve into a device regularly used by the general public. The practical point of view of our architecture is described in section 3. We go into the details of the user interaction necessary for object acquisition and the main interaction elements of the graphical user interface (GUI). We conclude with a discussion of the weaknesses and strengths of our approach in section 4, including considerations of the usability of the user interface.

2 Research

For the purpose of image-based object reconstruction commonly three steps are carried out [5–7] (figure 1). We start with a short overview of this object acquisition pipeline and introduce our contributions to this pipeline in the subsections 2.1, 2.2, and 2.3. Image-based object reconstruction starts by taking a number of images of different viewpoints of one object. Particular features of the object are expected to be visible in each image. Such an n-tuple of corre-

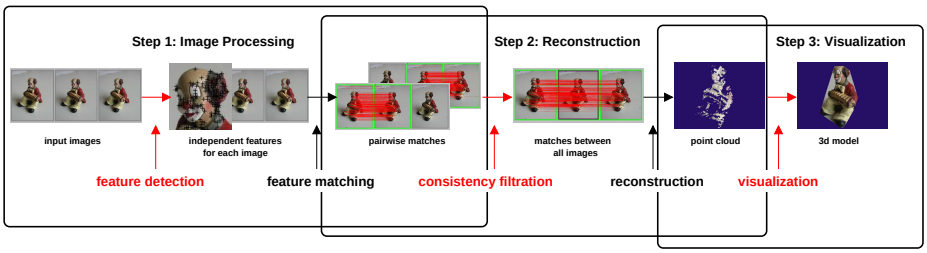

Fig. 1. Object acquisition pipeline. 1. Step: Image processing with feature detection and feature matching between pairs of images. 2. Step: Reconstruction with a filtration of consistent feature matches between all source images and reconstruction of the 3d positions of remaining correspondences. 3. Visualization with mapping of texture on the point cloud. Those parts of this pipeline we contribute to are highlighted in red.

sponding features between n images is called correspondence. Parameters of a camera can be divided into internal ones (such as the focal length) and external ones, i.e., especially the position and orientation of the camera in 3d space. If the parameters of the cameras and their positions are known for each taken image, the 3d position of an observed object feature can be derived directly by an intersection of the rays from the camera centers through the feature positions in the respective images. This technique is called triangulation. In our application images are taken freehand, i.e., camera parameters are not given. But the detected correspondences are sufficient to reconstruct their 3d position, if at least three images are available, because correctly determined correspondences meet certain geometrical constraints. These constraints can be algebraically represented as multi-view tensors, namely the fundamental matrix F in the 2-view case and the trifocal tensor T in the 3-view case. F and T can be recovered once a sufficient number (seven for F and six for T) of correspondences is available. Tensor computation allows for the computation of *generic* cameras, which create the same multi-view relation as the real cameras we used to take the images. The difference between the set of generic cameras and the set of real cameras consists in a projective transformation, which comprises translation, rotation, scaling, and perspective distortions. Thus, a 3d reconstruction of the object from generic cameras would differ from the real object by perspective distortions. But such a projective ambiguity can be upgraded by a subsequent autocalibration step [8] to a metric ambiguity, which consists of translation, rotation, and scaling only. Because of a lack of an absolute coordinate system, a metric reconstruction is the best reconstruction achievable.

Step 1 (Image Processing). To identify correspondences between the source images, first so-called *features* have to be detected by so-called *feature detectors*. Each feature is characterized numerically by a so-called *descriptor* which describes the properties of its surrounding image patch. Descriptors are utilized in the subsequent feature matching, the result of which are matches between image pairs (figure 1). Our contribution to this image processing step consists in a new

combination of feature detection and description providing a higher robustness. It is introduced in subsection 2.1.

Step 2 (Reconstruction). To eliminate possible mismatches a consistency filtration is necessary, i.e., only those correspondences which are consistent throughout the complete image sequence should be filtered and survive for the subsequent processing (figure 1). We have developed a two-stage process based on RANSAC [9], which facilitates a faster selection of those features relevant for reconstruction. This is described in subsection 2.2. RANSAC takes samples from the set of all correspondences and computes a compatible multi-view geometry. The support for that geometry is determined by examining the complete population of correspondences. All correspondences that agree (with a certain tolerance) on the geometry induced by the initial samples are called *inliers*. The total error of this estimated geometry is the sum of the errors of each inlier. After a subsequent metric reconstruction we obtain a 3d point cloud representing points on the object's surface.

Step 3 (Visualization). For the purpose of a realistically rendered representation of the object the point cloud has to be converted into a surface in the last step (figure 1). In subsection 2.3 we introduce our third conribution, a new method to map triangles of image texture on the point cloud, resulting in a arbitrarily rotatable 3d surface.

2.1 Feature Detection

One of the most widely used feature detector is the Harris detector [10]. It uses the auto-correlation matrix to detect corners. As the visibility of a feature depends on the distance between object and camera, the notion of *scale space* has been introduced [11]. It embeds feature detection in a framework of repeated image smoothing. In order to compare the cornerness output of the detectors across scales they have to be adapted. The adaptation necessary for the Harris detector can be found in [12]. Another feature detector uses a corner detector combined with a local Hough transform to determine the position of features [13]. In recent years, the SIFT detector and descriptor [14] was widely used, which utilizes a difference-of-gaussians (DOG) detector combined with a descriptor that is build from local gradient distributions. Another widely accepted detector/descriptor combination is SURF [15], which improves SIFT and can be evaluated fast.[1]

As a sparse point cloud would lead to an insufficient visualization, we depend on as many reliable correspondences between the input images as possible. Thus, the number of reliable correspondences, that survive the consistence filtration, is the criterion for the choice of our detector and descriptor. Although the SIFT provides reliable correspondences, the number of features it detects is not sufficient for our purpose. For this reason we developed a scale adapted Harris feature detector combined with a variation of the gradient based SIFT descriptor.

[1] The application of a tracking algorithm can be rejected, because the differences between images taken freehand usually are too large for the successful application of a tracking algorithm.

Choice of Detector. Our method is based on the approach of [16]. The fourfold smoothing is done by a convolution with the binomial kernel $(1, 4, 1; 4, 8, 4; 1, 4, 1)$, which preserves gradients well and is fast to compute.

Choice of Descriptor. We borrowed our descriptor from SIFT, with slight improvements to fit into our framework. First, we calculate a dominant gradient direction to make the descriptor invariant to rotations. In [14] the descriptor is rotated after its computation to fit that main orientation. Contrary to this, we map an image patch around the feature location into a 16×16 array. This mapping consists in an affine transformation that combines the scale and orientation information. This simplifies the descriptor computation as well as its implemetation. The values in the grid are used to compute the 16 gradient histograms the descriptor consists of. Finally, the matching is accomplished using a kd-tree with best-bin-first optimization [17].

Results. Since we want many features to survive the RANSAC stage, we compared the numer of reliable inliers provided by our approach with the results of other detectors. For the descriptor of all runs we used our variation of the SIFT descriptor. We tested two objects, clown (figure 1) and sokrates (figure 4), and took three images of each object. For each detector method (Hough transform, SIFT, SURF, and our Harris-SIFT) we carried out 15 detection runs, counted the correspondences that remained after the consistency filtration, and calculated the average of this number. In addition, we assessed the remaining correspondences reliable, if the metric reconstruction was possible.[2] The left part of table 1 shows the results of this evaluation. Our Harris-SIFT combination outperformes the other approaches for both objects in terms of numbers of inliers and at the same time provides reliable correspondences as well.

Table 1. Comparison of our methods with others. Left: Comparison of feature detectors. Right: Comparison of consistency filtration. Our algorithm compared to the one scored best in [18]. Each number in the first column is the average value of 200 runs. The last column shows the the relative running time with 100 assigned to our algorithm.

	clown		sokrates	
	reliable	av. num. inliers	reliable	av. num. inliers
Hough	no	139	no	123
SIFT	yes	92	yes	85
SURF	yes	67	yes	46
Harris-SIFT	yes	219	yes	174

	av. num. inliers	time
clown, our algo	226	100
clown, best of [18]	228	157
sokrates, our algo	413	100
sokrates, best of [18]	412	119

[2] A metric reconstruction is not possible, if either the number of the correspondences after the filtration is too small, or if the reprojection error after the projective reconstruction is too large.

2.2 Consistency Filtration

Once feature matches between pairs of source images have been identified, those of them have to be determined that agree on a common geometry. The standard approach to this is the RANSAC method [9]. Among many improvements proposed are Locally Optimzed RANSAC (LO-RANSAC) [18] and PROgressive ranSAC (PROSAC) [19]. LO-RANSAC improves the estimated geometry by applying a model refinement using a larger set of samples consisting of inliers only. A least squares approach is used for refinement. Afterwards the set of inliers that belongs to the refined estimation is computed from the complete population of correspondences. PROSAC reduces the running time by prefering more promising samples. Correspondences are sorted according to their matching costs computed during feature matching. Assuming that lower matching costs imply better matches, a growth function is applied to progressivly take more samples into account. One advantage of PROSAC consists in its speed, whereas LO-RANSAC is characterized by its accuracy. We aim at both, speed and accuracy, thus we combined LO-RANSAC with PROSAC. PROSAC can be implemented straight forward, but LO-RANSAC rises the question of how to implement the local optimization. We propose following method for consistency filtration: First, we take a minimal number of samples in an outer RANSAC loop to estimate the multi-view geometry. The samples are taken according to their matching costs as in PROSAC. Secondly, after each improvement of the error value, we take the inliers of the outer loop and perform a traditional RANSAC on them in an inner loop. This time we take twice the number of features as in the minimum case to rubustify the least squares solution. The inliers are again computed on the complete population of correspondences. This scheme is performed in a two-stage process. First, the fundamental matrix F is estimated using the standard 7-point algorithm for the minimum case [5] and the linear algorithm for the inner loop [5]. Then the trifocal tensor T is estimated using the algorithm proposed in [20] in the minimum case and the algebraic minimization algorithm in the inner loop [5]. From the application of this method we expect an advantageous compromise between speed and accuracy.

Results. We again tested two objects, clown (figure 1) and sokrates (figure 4). We took two images of each object and performed the feature detection on them. For each object we carried out 200 runs on the two images with our filtration method and 200 runs with the filtration method scored best in [18]. We counted the correspondences that remained after the performance of the filtration, and calculated the averages of these numbers. The right part of table 1 shows the results of this evaluation. Our approach outperformes the other approach for both objects in terms of speed and at the same time provides nearly the same number of correspondences.

2.3 Texture Mapping

At this point we constructed a metric reconstruction of single points of the observed object, i.e., a point cloud. In order to obtain a more realistic 3d representation of the object an improvement of the visualization remains to be done.

For this purpose we seek for a fast and simple approach. Commonly, only the 3d point cloud is presented as the result of the reconstruction [7, 21]. Disparity maps have been proposed to model an object's surface [22]. They assign each pixel of one image to a corresponding pixel of another image by generating an offset image. In [23] several iterative schemes for the direct creation of a surface from a sparse point cloud in 3d space are compared.

In general, disparity maps work well for highly textured images, but the background in our application often lacks texture. In addition, disparity maps work best for linear camera motion without tilting, which does not hold for freehand acquisition. Consequentially, tests of a disparity map approach in our application turned out unsuccessfully. The approach of [23] takes at least 5 minutes just to compute the viualization and must be considered useless for an adoption in practice. As we aim at a fast and straight forward method, we decided to apply a delaunay triangulation to the feature points of one of the three images taken. Since a 3d point of the point cloud can be assigned to its correponding 2d feature of a source image, it is possible to build a 3d mesh using the feature points of the image for coordinate calculation. The last step consists in a mapping of the texture of the image triangles to the triangels of the 3d mesh. In this way we gain a very fast method that creates realistically visualized object surfaces, given the feature points cover the object sufficiently dense. Figures 3 and 4 show results for two objects.

3 Practice

From a practice point of view we have already pointed out in section 2.3 that many systems aim at a reconstruction of a 3d object from single images. But most of them do not focus on the simplicity and speed necessary to turn it into a technique attractive to the average user. Recently, much attention has been payed to systems that do aim at a wider audience, such as Make3D [25] and Phototourism [24]. But these approaches lack speed (especially Phototourism, because too many images are needed as input) or do not pursue the goal of creating a 3d surface. A system that creates a surface from images is presented in [23], but from a usability point of view it is too complex as well as too slow.

In this section we describe the steps of user interaction with our system, necessary to carry out the reconstruction of a 3d model from a few snapshots of an object (subsection 3.1). In addition, the graphical user interface of the acquisition system is described (subsection 3.2).

3.1 User Interaction

The necessary equipment for object acquisition consists of the computer with the acquisition software and a camera only. In figure 2 the steps of user interaction are displayed. We distinguish following 5 steps:
1. Take at least three snapshots (figure 2a). Result: Three images are stored in the storage of the camera.

2. Connect the camera to the computer. The connection can be wireless, otherwise the user has to connect the camera by a cable to the computer (figure 2b).
3. Click button to initiate image loading from camera to computer. Result: Images are stored on the computer and deleted from the camera storage. Images are displayed in the left of the lower window of the GUI (figure 2b).
4. Click button to initiate model creation (figure 2b). Result: The algorithms described in section 2 are initiated. A progress bar is displayed in the main window of the GUI giving feedback on the remaining time until the 3d model is created, which usually takes a few seconds (figure 2c). After this period the generated model is displayed in the main window (figure 2d).[3]
5. Move mouse on main window of GUI with left button pressed to rotate the model around its center and thus view it from arbitrary viewpoints. Result: The 3d model rotates intuitively according to the direction of the mouse gesture. The model is displayed continuously during rotation (figures 2e and f and figure 4).

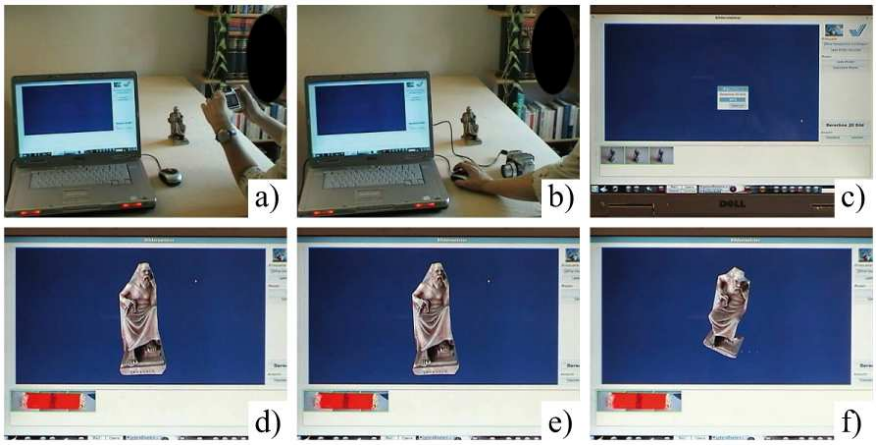

Fig. 2. Steps of user interaction. a) Step 1: Take snapshots. b) Steps 2, 3, and 4: Connect camera to computer, click button to initiate image loading, click button to initiate model creation. c) Progress bar is displayed. Wait a few seconds while model is generated. d) Generated 3d model is displayed. e),f) Step 5: Visualization of model from different viewpoints during rotation via mouse movement on main window.

3.2 Graphical User Interface

The GUI is spatially devided into three regions, a large main window in the upper left, a smaller window at the bottom, and a control region in the upper

[3] Only those object parts visible in the source images are represented in the model. About 10 images covering the complete viewing sphere of an object are necessary to reconstruct a complete model.

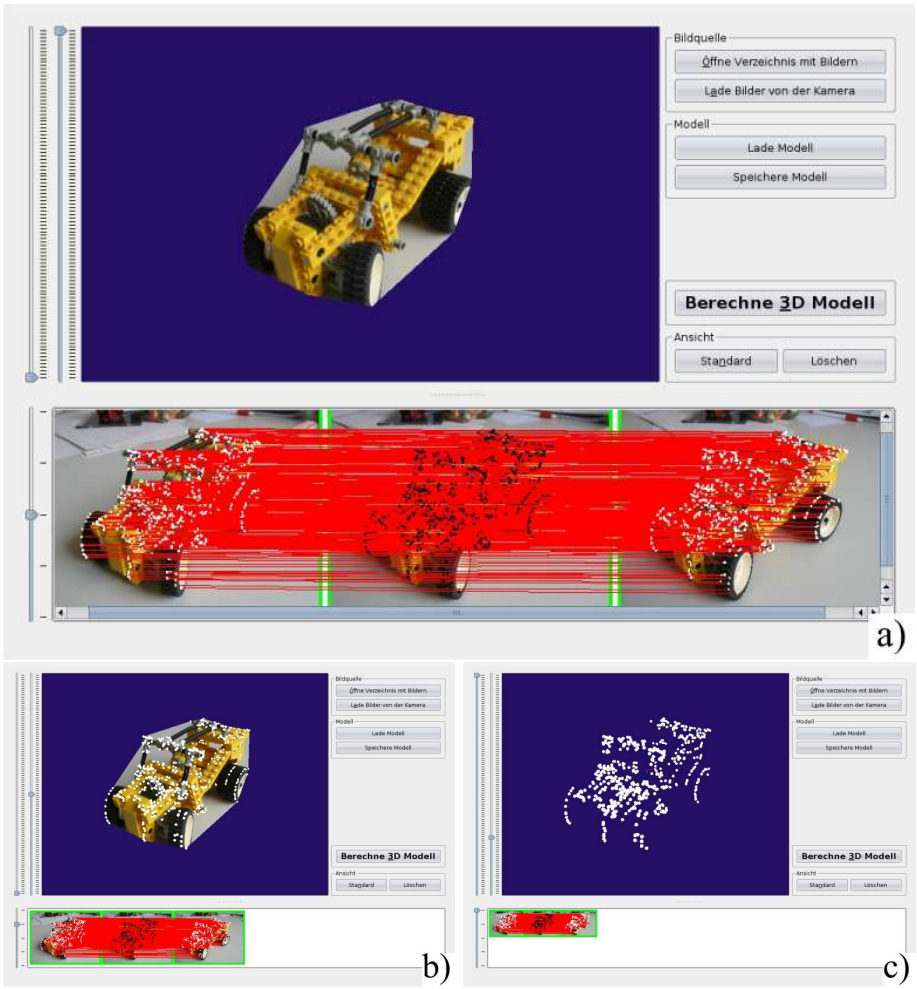

Fig. 3. Graphical User Interface. a) Main regions of the GUI: main window with generated 3d model, bottom window with input images and correspondences, and region with command buttons. b) Superimposition of point cloud with mapped texture, smaller display window for the loaded images. c) Main window with point cloud only, even smaller display of input images.

right of the GUI (figure 3a). In the windows results of the computations are displayed. In the main window the generated 3d model, the calculated point cloud of the object, or a mixture of both are visualized, depending on user interaction. In the bottom window the input images are displayed, as well as the correspondences between the different views of the object in the form of red lines connecting corresponding object points. Interaction is controlled by basic interaction elements [26] such as command buttons for action initiation via mouse-click and sliders to gradually vary between different display modes. In addition, we employ scroll bars and split bars to enable the user to customize some properties of the GUI. As application dependent interaction elements the user can execute mouse gestures in the main window to rotate the object model. In more detail, we employ command buttons for the following actions (figure 3a) upper right region of the GUI from top to bottom): load images already stored on the computer, load images from camera, load a previously stored 3d model, store a caculated 3d model, calculate 3d model. (The remaining two command buttons cause the main window to display the model from its front view or to delete the display, respectively.) The most important of these command buttons for our interaction design are: load images from camera and calculate 3d model. Additonal controls are given by two sliders to the left of the main window and one slider to the left of the bottom window. The first slider to the left of the main window controls the visibility of the textures mapped to the point cloud, i.e., if the slider is at the bottom the calculated 3d model is visualized completely with texture (figure 3a). If it is at the top, only the point cloud is visualized (figure 3c), and if it moves from the bottom to the top mapped texture triangles are one after another removed from the display as shown in figure 5a. The second slider to the left of the main window controls the visibility and size of the points of the calculated object point cloud in the main window. If it is at the top no 3d points are displayed on the generated model. If the user moves the slider from top to bottom the point cloud appears with increasing size of the points (figure 3b) until the points reach their maximum size when the slider is at the bottom. The slider to the left of the bottom window adjusts the size of the input images and their correspondences in the bottom window (figure 3a-c). This, in conjunction with the adjustable split bar between the main and the bottom window, enables the user to inspect the identified correspondences more precisely. In case the zoom factor for the images is larger than the region reserved for the bottom window, scroll bars appear to the bottom and right of the window (figure 3a). An important interaction element of the GUI is given by mouse gestures which can be carried out on the main window when a 3d model and/or a point cloud is displayed. The user can move the mouse with left button pressed to rotate the model around its center according to the direction of the mouse gesture. Thus, the model can be viewed intuitively from arbitrary viewpoints, e.g., to analyse its quality or just for fun.

Fig. 4. Model rotation via mouse gestures in the main window of the GUI. The upper row shows four views of the calculated point cloud of an object. The lower row shows the corresponding views of the final 3d model, created from three source images, after the visualization step has been completed. From left to right the different views are displayed by moving the mouse in the particular direction, inducing the impression that the model is rotated around its center.

4 Discussion and Outlook

In this section weaknesses and strenghts of the proposed architecture from the research point of view, as well as from its practical point of view are discussed. In addition and where applicable, we give an outlook on to further developments.

The stability of the proposed feature matching is quite good, but we have not tested the SURF descritors yet, which are claimed to be more stable than the SIFT descriptors and may also speed up the matching. The consistency filtration we proposed solves the filtering task well and is entirely sufficient for our few-images application. Currently missing is a detection of degenerate cases in which no multi-view geometry can be computed, e.g., because a single dominant plane is present in the images or cameras of the freehand acquisition share the same center. We experience difficulties when objects with a homogeneous texture are to be reconstructed. If we would postpone the decision about ambiguous matches until a geometry estimation is established, we hope to be able to include more features and create an even smoother surface also for such difficult objects.

We tested our system on other objects such as buildings as well. As different problems are faced during the reconstruction of outdoor objects, a different approach than the one described is probably necessary. One problem that occurs in urban environments is the fact that buildings rarely are freestanding so not each necessary viewpoint can be covered without occlusions. Other problems consist in instable lighting conditions and moving objects that occur in the scene such as padestrians or cars. We tested our prototyp on a freestanding building with three sample views (figure 5a). Many correspondences have been detected which are not part of the building. Those points preferrably are located in trees and bushes. Beeing not optimized for this kind of difficulties, our system neverthe-

less yielded an acceptable reconstruction of the building with the exception that parts of a tree in the forground of the images are incorporated in the 3d model. Figures 5b and c show the reconstructed point cloud from different viewpoints with the faulty tree representation in the left and in the right, respectively. Future research will broaden the field of reconstructable objects to outdoor objects such as buildings or cars.

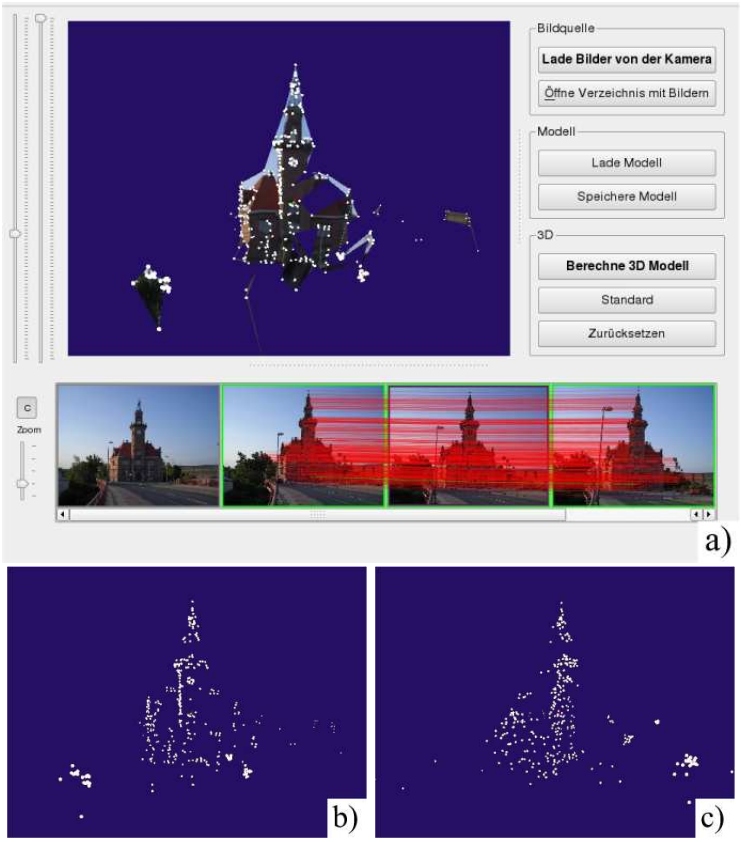

Fig. 5. Reconstruction of a building. Not designed for reconstruction of outdoor objects this application reveals some weaknesses of the proposed system.

We did not carry out systematic, scientific studies on the usability of our object acquisition system yet. But we took the advantage of several demonstrations of our prototyp in the public to carry out approximately 50 heuristic user tests. Apart from beeing representative the participating users have been heterogeneous, from househusbands over school children to senior citizens. In figure 6 three frames of a television broadcast about such a public demonstration are displayed. As there are only 5 steps to carry out (section 3.1) the huge majority of

Fig. 6. Television broadcast about our system.

users was able to accomplish the object acquisition at once after one demonstration only. The operation of the GUI worked intuitively and faultlessly. Moreover, especially children had much fun during the interaction with the prototyp.

With respect to basic interaction design principles and goals of usability [27, 28], such as effectiveness, efficiency, satisfaction, and ease of learning, it seems that the presented acquisition system can be a starting point for a fast, robust, and intuitive tool for the reconstruction of 3d models of objects. Such a tool can be applied in many fields, for example in game development or more general the leisure and entertainment industry, in art and culture, in online shopping, in e-learning and edutainment, in robot navigation, human-robot interaction, and augmented reality, in machine building and manufactoring, or in design and construction, e.g., in the automotive industry. Of course, the usability of our prototype has to be evaluated more systematically in the future. But as its handling is in the truest sense of the word child's play we hope this tool is not only useful for professional applications but also attractive to the broad public.

Acknowledgments. This research was funded by the German Research Association (DFG) under Grant PE 887/3-3.

References

1. Wikipedia, Wikimedia Foundation, Inc., Coordinate-measuring machine, `http://en.wikipedia.org/wiki/Coordinate_measuring_machine`
2. Bernardini, F., Rushmeier, H.: The 3D Model Acquisition Pipeline. Computer Graphics Forum, vol. 21(2), 149–172 (2002)
3. Peng, T., Gupta, S.K.: Model and algorithms for point cloud construction using digital projection patterns. Journal of Computing and Information Science in Engineering, vol. 7(4), 372–381 (2007)
4. Murray, D.W.: Recovering Range Using Virtual Multicamera Stereo. CVIU, vol. 61(2), 285–291 (1995)
5. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Second edn. Cambridge University Press (2004)
6. Beardsley, P.A., Torr, P.H.S., Zisserman, A.: 3d model acquisition from extended image sequences. In: ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II, 683–695, Springer, London (1996)

7. Fitzgibbon, A.W., Zisserman, A.: Automatic camera recovery for closed or open image sequences. In: Proceedings of the European Conference on Computer Vision, 311–326 (1998)
8. Pollefeys, M., Koch, R., van Gool, L.J.: Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In: ICCV. 90–95 (1998)
9. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM, vol. 24(6), 381–395 (1981)
10. Harris, C., Stephens, M.: A Combined Corner and Edge Detector. In: 4th ALVEY Vision Conference, 147–151 (1988)
11. Lindeberg, T.: Feature detection with automatic scale selection. International Journal of Computer Vision, vol. 30(2), 77–116 (1998)
12. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. International Journal of Computer Vision, vol. 60(1), 63–86 (2004)
13. Shen, F., Wang, H.: A local edge detector used for finding corners. Proc. ICICS (2001)
14. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision, vol. 60(2), 91–110 (2004)
15. Bay, H., Tuytelaars, T., Gool, V., L.: Surf: Speeded up robust features. In: 9th European Conference on Computer Vision, Graz (2006)
16. Lindeberg, T., Bretzner, L.: Real-time scale selection in hybrid multi-scale representations. In: Proceedings Scale-Space. LNCS, vol. 4128, pp. 148–163 (2003)
17. Beis, J., Lowe, D.: Shape indexing using approximate nearest-neighbor search in highdimensional spaces. In: Proc. IEEE Conf. Comp. Vision Patt. Recog., 1000–1006 (1997)
18. Chum, O., Matas, J., Kittler, J.: Locally optimized ransac. In: DAGM-Symposium. 236–243 (2003)
19. Chum, O., Matas, J.: Matching with PROSAC - progressive sample consensus. In: Schmid, C., Soatto, S., Tomasi, C. (eds.): Proc. of Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, 220–226, IEEE Computer Society, Los Alamitos (2005)
20. Schaffalitzky, F., Zisserman, A., Hartley, R.I., Torr, P.H.S.: A six point solution for structure and motion. In: ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I, 632–648, Springer, London (2000)
21. Schaffalitzky, F., Zisserman, A.: Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In: Proceedings of the 7th European Conference on Computer Vision. vol. 1, 414–431, Springer (2002)
22. Pollefeys, M., Gool, L.V., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. Int. J. Comput. Vision, vol. 59(3), 207–232 (2004)
23. Lhuillier, M., Quan, L.: A quasi-dense approach to surface reconstruction from uncalibrated images. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27(3), 418–433 (2005)
24. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring photo collections in 3d. ACM Transactions on Graphics (SIGGRAPH Proceedings), vol. 25(3), 835–846 (2006)
25. Saxena, A., Sun, M., Ng, A.Y.: Make3d: Depth perception from a single still image. In: Fox, D., Gomes, C.P. (eds.): AAAI, 1571–1576 (2008)
26. Balzert, Helmut: Lehrbuch der Software-Technik, Bd. 1: Software-Entwicklung. Second edn. Spektrum Akademischer Verlag (2000)

27. ISO 9241-11, Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability (1998)
28. Nielsen, Jakob: Usability Engineering. AP Professional, Boston, Mass. (1993)