Rapid Reconstruction of Small Objects on Mobile Phones

Andreas Hartl, Lukas Gruber, Clemens Arth, Stefan Hauswiesner, Dieter Schmalstieg
Institute for Computer Graphics and Vision
Graz Technical University
Inffeldgasse 16/2, 8010 AUSTRIA
{hartl,lgruber,arth,hauswiesner,schmalstieg}@icg.tugraz.at

Abstract

Augmented Reality (AR) on mobile phones is receiving more and more attention recently, becoming a popular research topic and an important commercial field. In this paper we present a lightweight method to create coarse 3D models of small-scale objects. The goal is to give the users the possibility to create and maintain AR content themselves without the need for expensive tools and complex interaction. Our algorithm is based on shape-from-silhouette using voxel carving and runs on modern smartphone hardware. 3D models of certain object groups can be generated interactively and instantly. The actual result is visualized continuously using image based rendering methods to inform the user about the actual model quality. Given a suitably accurate model it can further be used for any means of AR and can easily be shared with other users. The usability of our approach is evaluated using modern smartphone hardware (see Figure 1). The advantages are demonstrated using a set of models for playing a board game.

1. Introduction

In Augmented Reality (AR) we deal with fusing the real and the virtual in a natural and visually appealing way. The view onto the real world is enhanced with any kind of virtual content, be it textual, graphical or streaming audio or video information. Most currently available types of conventional media such as newspapers or television leave the user as a passive observer, flooded by content coming from some sort of preprocessed source. It is clear that also for current AR applications, the easiest design is to follow this scheme. For the users, however, large parts of the fun and power of AR arise not until they are given the opportunity to actively alter the scene and interact with it themselves. In turn, this requires that such applications are able to collect, build, maintain, store, load and share content in order to allow this level of user experience.

In this paper we are dealing with the problem of small-scale 3D model generation on smartphones. Since 3D models form an essential part of AR suitable content, the goal is to provide the user with the possibility of generating such 3D models on the fly by using his mobile phone, to use it in any chosen AR application or game, and to share it on social networks. 3D model generation is a research topic with a long tradition, being manifold in terms of scale, the employed tools and the amount of computational resources available at the same time. As we are targeting embedded hardware, we have to put special attention to trading algorithm complexity with computational resources available. In this respect, efficient Computer Vision (CV) algorithms are the key to make the task manageable at all.

Figure 1. The *Nokia N900* smartphone used for our evaluation. The screen depicts a scene from the modeling procedure. The current modeling result is rendered in the lower left corner.

The main contribution of this paper is an approach to generate simple, yet visually appealing 3D models of small objects in real-time on modern smartphone hardware. By using segmentation on structured background and so-called *Voxel Carving* we can create the *Visual Hull* of the object, which can be visualized by simple texture mapping and image-based rendering. The resulting model is displayed

continuously, which allows the users to refine it iteratively until they are satisfied with the quality. The system only requires very simple user inputs, like moving the camera and pressing a button, which makes it suitable for novice users or difficult environment conditions. Despite some well-known shortcomings of the involved algorithms, the overall approach gives reasonable results and proves itself plausible for the given scenario. While the explicit voxel representation requires a relatively large amount of memory, it is superior to other visual hull algorithms in terms of processing: it allows for refining the model incrementally without recomputing already carved regions. Consequently, our approach can be used intuitively and creates models that improve in quality with each iteration.

This paper is structured as follows. Section 2 gives an overview of related work in the area of modeling. In Section 3 we are describing our approach in detail and elaborate on the special adaptions and modifications needed to achieve the goal of real-time application on smartphone hardware and the reception of visually appealing results. In Section 4 some results of our approach are shown and we give an overview of resources used for a specific hardware and model acquisition setup. Section 5 concludes with a summary and an outlook on future work.

2. Related Work

The generation of 3D models has had a long history in Computer Vision (CV) and Computer Graphics (CG). On the one hand, the area of CV contributes a lot of automatic methods for reconstructing objects or environments. Special attention has been drawn to algorithms that allow the online generation of 3D models at interactive framerates just recently. On the other hand, in the area of CG the typical access is the manual generation of models using commercial CAD software, such as *3ds MAX* or *Maya*, or by using open-source tools like *Blender*. For the application of 3D models in AR researchers likewise borrow approaches from both areas, dependent on the offline or online creation of models. For the latter, the goal is to generate some sort of model automatically and instantaneously, probably with interaction by the user and, if necessary, on embedded hardware. In the following we focus on related work in two separate areas. The first area is online modeling, where the methods that target AR applications on mobile devices will be mentioned explicitly. The second area is modeling methods using shape-from-silhouette techniques. This topic is of special relevance since our proposed algorithm has a strong relationship to this group of methods. Note that the list of references is not complete.

Online Interactive Modeling Modeling in AR can already be found, for example, in the work of Lee *et al*. [12] dating back to 2001, where a head-mounted display (HMD) is used together with an optical tracking system and a high-end graphics workstation to interactively model objects. Piekarski and Thomas [17] present an approach to model outdoor scenes with geometric primitives using a wearable computer. In the field of Computer Vision, Akbarzadeh *et al*. [1] describe a system to generate models of urban environments while passing with a vehicle. A system called *Videotrace* is presented in the work of van den Hengel *et al*. [20] to model environments and objects from videos. With a special application to AR, Bunnun and Mayol-Cuevas [6] describe a system called *OutlinAR* to interactively model objects using a simple makeshift joystick. The system was also shown to work in a mobile setup recently [5]. Klein and Murray [10] present the *Parallel Tracking and Mapping* (PTAM) system on a mobile phone. The system creates a sparse feature model of a workspace sized area on the fly and uses it as a tracking environment for AR. Pan *et al*. [16] describe a system for reconstructing building facades outdoors from wide-field-of-view images. The algorithm calculates feature matches out of a given set of images, triangulates these matches and creates a textured model of the given scene on smartphone hardware within several seconds. Simon [19] presents a system for interactive 3D Sketching which runs on a mobile computer. Closely related to our method is the work from Bastian *et al*. [3], in which the authors present an interactive approach to model objects using PTAM and segmentation methods. The user marks parts of an object to add to a prior, with which a model is created through an interactive segmentation and tracking process.

Modeling using Shape-from-Silhouette The coarse shape of a 3D object can be computed from a set of silhouette images from calibrated cameras. In CV, this is traditionally called shape-from-silhouette. The principal concept was first introduced by Baumgart [4]. Later, this coarse model was given the name *Visual Hull* by Laurentini [11]. It is defined as the maximal approximation of the object that conforms to the silhouettes. Creating novel views of a visual hull can be performed efficiently using image-based visual hulls (IBVH) as shown by Matusik *et al*. [14]. Modern GPUs allow for efficient implementations (see the work of Hauswiesner *et al*. [9]). Matusik *et al*. [13] have shown that the IBVH representation can also be transformed into a 3D model, which are naturally view dependent, however. Therefore, the entire visual hull needs to be recomputed frequently.

Another popular approach is volume carving (or voxel carving). It works by successively removing empty regions of an explicit volume representation as shown by Chien and Aggarwal [7] and Potmesil [18]. The representation can be a simple voxel grid or a more complex tree structure. Discrete points in this representation are projected onto each of

the silhouette images. If the projected point lies outside the silhouette in any of these images, it gets discarded.

Discussion Interactive modeling for AR on mobile devices is becoming more important steadily. However, its research topic has not been profoundly addressed yet, which can be seen from the low number of approaches running on handheld devices. Real-time modeling of objects is still a computationally demanding task, even on modern desktop hardware. The approach by Bastian *et al*. [3] runs at around 5 fps on 640x480 pixel images using a medium-end desktop PC with heavy usage of GPU shader implementations. The results are convincing, the computational complexity, however, is far beyond what can currently be achieved on off-the-shelf smartphone hardware. Note that the authors also use voxel carving for generating an estimate of the object volume.

Although some sort of user interaction during modeling on mobile devices is indispensable, there are severe restrictions to this. Smartphone screens are relatively small and touch input suffers from significant inaccuracies. Thus, it is almost impossible to accurately mask or mark objects on the screen. In our system, user input is restricted to moving the device and taking pictures. These forms of input do not suffer from accuracy issues.

Voxel carving has several advantages over conventional reconstruction techniques, such as stereo matching or IBVH, which make it especially suitable for use on embedded hardware. Firstly, only simple operations are required to process it, such as projection by vector-matrix products and image lookups. Secondly, because the voxel representation is explicit, refining the model with additional silhouette images does not require to recompute everything from scratch. Instead, only voxels which survived the carving process of all previous silhouette images need to be checked against a newly added image. Furthermore, these voxels only need to be compared to the new image and not to all of the already captured ones. Also, in contrast to techniques that use point correspondences, reconstruction from silhouette images is robust against reflections and view-dependent lighting effects. This allows, for example, to reliably capture metallic or glossy objects, given that a silhouette may be obtained. Lastly, the growing computational power of mobile GPUs creates an interesting potential for further optimization.

3. 3D Model Generation

Our proposed algorithm aims at the automatic creation of simple models from a set of images. These images have to be registered with respect to the target scene (*i.e*. the reconstructed object). To facilitate this, we use a simple frame marker which can be efficiently detected and tracked using popular AR tracking software like *Studierstube* or *ARToolkitPlus*[1]. In Figure 2 an examplary modeling setup is depicted, showing a set of images taken around an object located on a marker target. The object itself is rendered using image information.

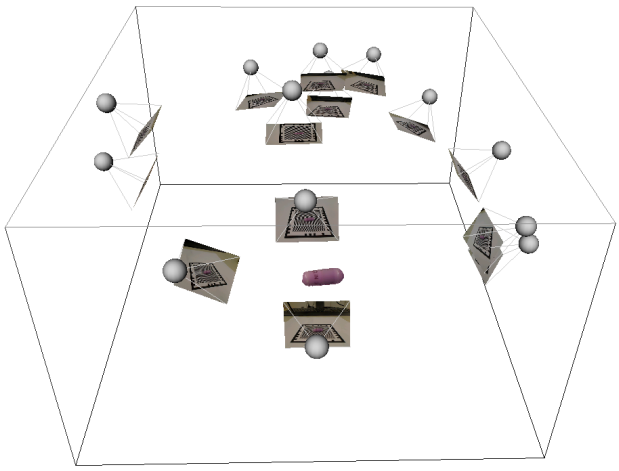

Figure 2. An example for the rendering of a scene, showing camera positions, frustums and images around the rendering of a reconstructed purple pill.

Given the camera pose for a single image, an important problem is the efficient segmentation of the object, while the object color should not be restricted. The use of popular methods like graph-cuts or MSER is prohibitive on smartphone hardware due to the computational expense of these algorithms. We situate the objects on a checkerboard pattern instead, which we place in the middle of the marker. This approach allows us to apply an efficient segmentation method in order to acquire the required silhouettes (see the work of Anonymous [8]). This segmentation algorithm relies on local adaptive thresholding and simple morphological operations. Subsequently, an efficient labeling procedure with integrated contour-tracing is used to obtain a segmentation of convex or non-convex objects. While being especially adapted to small, untextured objects it is also possible to process arbitrary objects (see e.g. Figure 6), provided that they do not contain checkerboard-like structures. One key aspect concerning the applicability is the excessive use of context information (homography, target dimensions, checkerboard pattern size) to limit the amount of processing required, but also to gain robustness. Information obtained from this step can be directly used to reduce the amount of input data for the proposed reconstruction method.

Since model quality is directly dependent on silhouette quality, we use an incremental user-guided approach for segmentation. By visual inspection the user may check at any time whether the acquired silhouette is good enough

[1] http://studierstube.icg.tu-graz.ac.at

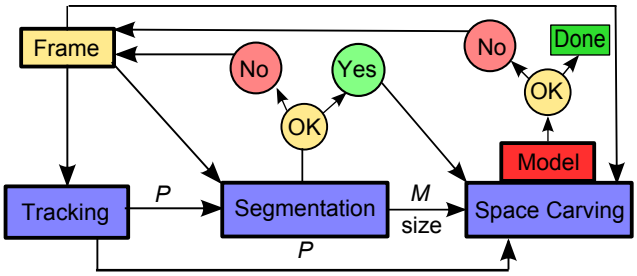

Figure 3. Flowchart of our sample application. *P* denotes the projection matrix, and *M* denotes the silhouette mask respectively.

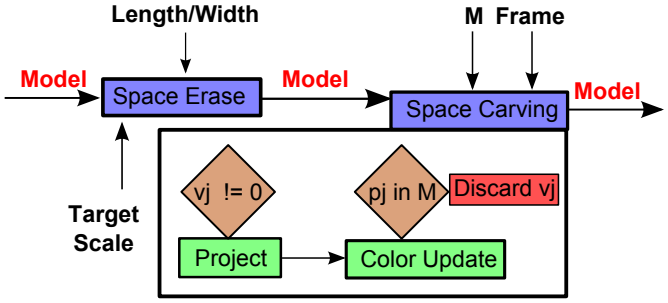

Figure 4. Outline of our voxel carving procedure.

for model refinement or not. Such silhouettes, as well as the obtained projection matrices from the tracker, serve as input for our voxel carving module (see Figure 3). The resulting model is immediately shown to the user. The process of adding images to the carving and model generation engine is repeated until the user is satisfied with the visual quality of the model.

3.1. Voxel Carving Module

Our model generation algorithm relies on voxel carving in its simplest form, where we use an array to represent a volume. We project each remaining voxel v_j using the projection matrix P_i of the current view and get the pixel position p_j. Afterwards, we test whether the computed pixel position lies within the silhouette of view i. If it does not, we discard (carve away) the current voxel.

The size of our array is determined by the side length t_l of our square marker-based target, the maximum desired object height z_{omx} and the required accuracies δx, δy and δz (see Equation 1).

$$s = \frac{(t_l)^2 z_{omx}}{\delta(xy)\delta z} \tag{1}$$

Note that here the same accuracy in width and length ($\delta(xy)$) is assumed. If we therefore have objects with $z_{omx} = 20mm$ and use the segmentation approach in [8], giving $\delta(xy) = \delta z = 0.5mm$ on a marker-based target with checkerboard length $t_l = 60mm$, we get an array size $s = 576000$.

For the majority of objects we are able to severely reduce the amount of required projection steps by exploiting size information obtained from segmentation. The used segmentation approach gives the metric length and width of the object. We may use this information to erase a huge part of the space before the actual carving takes place. This decision is supported by the fact that objects need to be much smaller than the target itself, since it must be possible to segment the object solely on the checkerboard area. In order to account for inaccuracies, we add some small guard border b to the dimensions reported from segmentation. So, in case segmentation reports a square object with side length $x_{omx} = y_{omx} = 20mm$ and we use a border $b = 5mm$, the remaining amount of voxels is just $s = 100000$. In Figure 4 a summary of these steps is shown.

Although a rough voxel space may give enough information for measuring objects, we additionally store texture information obtained from the corresponding input images. This makes it easier for the users to judge the quality of the current session, as well as to obtain a usable model for other purposes.

3.2. Fast 3D Model Rendering

The quality of the model can be improved by more and more reconstruction iterations. Therefore the users are able to see the result immediately after each reconstruction step and they can decide if a refinement step is necessary for themselves. As a consequence we implemented a fast visualization method of the voxel volume which provides visual user feedback.

Since mobile phones with hardware constraints are the targeted platform we aimed for a lightweight visualization solution. Our approach is based on the following assumptions: (i) the object is placed on a plane, (ii) the object has a convex height profile, and (iii) the voxels in the voxel volume are equally distant to each other. These assumptions allow us to directly map the heights of the voxel volume to a regular pre-triangulated grid or heightfield. In doing so we avoid the employment of a computationally more expensive triangulation algorithm such as QHull[2] [2].

For coloring the model we implemented an offline image based rendering method. The common image based rendering approach projects each frame, the texture or image from any view onto the geometry. To avoid expensive texture look-ups at each frame we pre-compute the color and store it as a vertex color. Consequently, the color value of each voxel is the average of the color values measured from the camera input images.

The fast 3D model rendering method can be divided into the following steps:

- Initialization step: Creation of a heightfield matching

[2] http://www.qhull.org

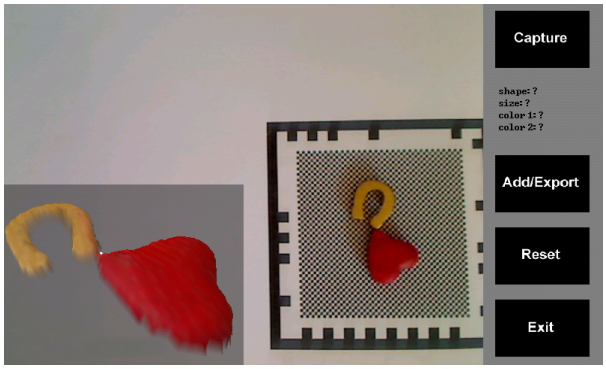

Figure 5. Fast rendering of the voxel volume. With each additional image the new result is visualized automatically.

the x and y dimensions of the voxel volume.

- Voxel carving step: In this step the voxel volume is refined. Additionally we project the 3D voxel into the current camera image to obtain a color value for the voxel. In case the same voxel is seen from more than one view multiple color entries are stored in a vector.
- Heightfield update step: The z value of each vertex v(x,y,z) on the regular grid corresponds to the voxel with the highest z at x and y in the voxel volume. The color of the vertex is determined by averaging the different color values of the corresponding voxel.
- Update step 3: The surface normal vectors of the heightfield are computed.

The result of a fast visualization is depicted in Figure 5. An entire sequence of images, together with the carving results is depicted in Figure 6. The disadvantages of this approach are that the mesh has to be as dense as the voxel volume to avoid sampling artifacts. Additionally, the class of shapes is limited during visualization. The final voxel model, however, does not suffer from this limitation.

3.3. Model Refinement for Height Measurement

Although the proposed method may give visually appealing results for certain objects, some postprocessing of the voxel space is necessary for more accurate measurements. The major problem is that camera poses have a certain minimum height above the marker plane. This is given by target and object dimensions and leads to artifacts. In general, the upper part of the object is formed like a pointed roof, which can severely deteriorate model quality or hamper more accurate measurements of objects. A possibility to tackle this problem would be visual inspection and manual correction of the model by the users, which is possible for all kinds of objects. In order to make our approach more accessible to untrained users, we opted for an automatic method. In order to avoid bigger targets (*i.e.* larger frame markers) or finer resolutions, which would alleviate this problem, we resolved this by estimating the real height directlyfrom the voxel space. This is possible for symmetric objects, where the amount of voxels necessary for representation (starting from the ground plane and moving up) is approximately constant (e.g. simple objects such as medical pills). More specifically, we search for a z-index into the voxel space that allows us to select a suitable partition which resembles one half of a symmetric object lying on the groundplane. The process is as follows:

1. Iterate the voxel space along the z-axis from the bottom upwards and record the amount of voxels (*i.e.* the object cross section) at each index z into a vector $profile_z$. Stop if the # of voxels become zero.
2. Compute derivatives $dprofile_z$ by using forward and backward differences and record each maximum z_{max}
3. Analyze the maxima using mean and standard deviation and come up with a peak threshold th_{peak}
4. Select connected segments having $dprofile_z \geq th_{peak}$
5. Select the longest segment s_{max} and compute its central index zc_{smax}. This gives an estimate of half the height of the object.
6. The final height index z_{max} is given by $2 \cdot zc_{smax}$.

Using the known scaling δz of the voxel space the corrected height may be computed and the voxel volume itself may also be cut. Additional improvements, like the consideration of systematic errors, can be made but these must be applied with care, depending on the nature of objects.

4. Experimental Results

In the following we show experimental results of our approach. Firstly, we give an overview of the computational and memory requirements of our approach, evaluated on ordinary smartphone hardware. Secondly, we give some experimental results to demonstrate the power of the approach for acquiring size measurements of given objects. This can be especially useful for tasks where the dimensions of an object represent a valuable cue for recognition (see e.g. [8]). Lastly, we demonstrate how the algorithm can be used to generate models of small-scale objects used in an augmented board game.

4.1. Smartphone Application

Our algorithm is evaluated on a *Nokia N900* smartphone running *Maemo* Linux, featuring a 600 Mhz ARM Cortex A8 CPU (see Figure 1). We use an input resolution

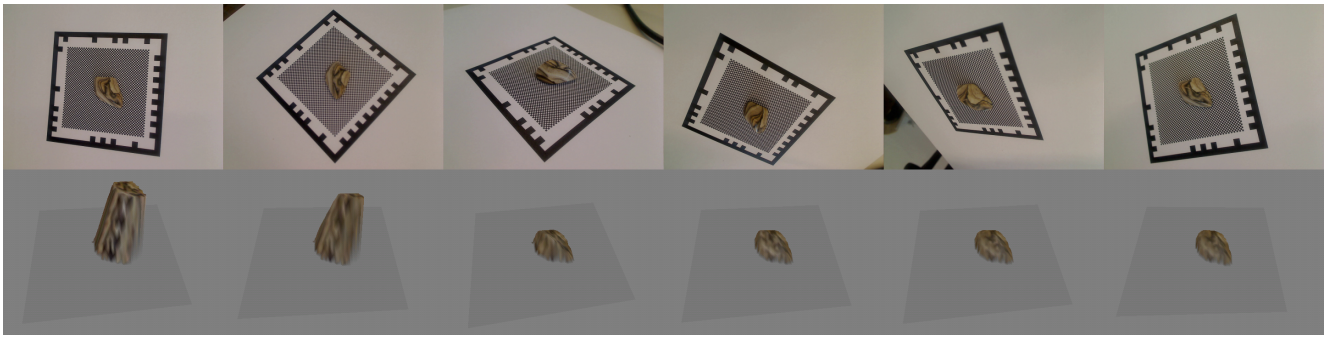

Figure 6. A sequence of images from a given object and the corresponding carving results. While the first few images have a large impact, with an increasing number of images the visible changes become marginal.

Voxel Size (l/w/h) [mm]			# Voxels	Time [ms]	Memory [kB]
0.5	0.5	0.25	2985984	2378.70	2916
0.5	0.5	0.5	1492992	1211.05	1458
1.0	1.0	0.25	746496	146.49	729
1.0	1.0	0.5	373248	74.84	364.5
1.0	1.0	1.0	186624	40.68	182.25

Table 1. Runtime estimation and memory consumption for different voxel sizes on our smartphone platform.

Voxel Size (l/w/h) [mm]			# Voxels	Mean Err.	Std Dev.
0.5	0.5	1.0	746496	0.57	0.34
0.5	0.5	0.5	1492992	0.46	0.43
0.5	0.5	0.25	2985984	0.28	0.28
0.5	0.5	0.125	5971968	0.26	0.26

Table 2. Summary of results for height measurements on medical pills (values are given in [mm]).

of 640x480 pixels, utilizing the *FCam API*[3] for adjusting suitable camera parameters. Note that we do not take any advantage of the separate DSP available, nor do we use the mobile GPU and OpenGL ES 2.0 for performance optimization. The entire carving algorithm runs on the CPU only.

In Table 1 the average runtime for the carving procedure and the memory consumption for the voxel space is listed for different voxel sizes. The runtime estimate is averaged out of the carving runs on 10 different objects and refers to the average time spent in each call to the carving routine. The initial voxel space has a size of 72x72x36 millimeters. The size of the voxels is the critical factor for the memory consumption, as can be seen in the last column of Table 1. A smaller voxel size results in a carving space with a higher resolution. In turn, the computational and memory demands increase and the model quality improves.

In Figure 7 the time spent for the incremental carving steps is drawn from different voxel space configurations. The first step takes the most time, since most pixels are carved away initially. However, for all additional images the time needed for carving decreases significantly. At the same time, the carving result becomes more accurate. Due to the current implementation of the carving space and the corresponding carving routine, the entire approach becomes notably slow for calculating the first carving result when the spatial resolution of the carving space becomes high. Since the user has to take multiple images of the object to be modeled, an intuitive way to circumvent the application stalls is to use the algorithm in a multi-threaded application environment. Calculating the carving result can be done in a background thread, while the application still remains responsive for taking additional snapshots.

[3] http://fcam.garage.maemo.org

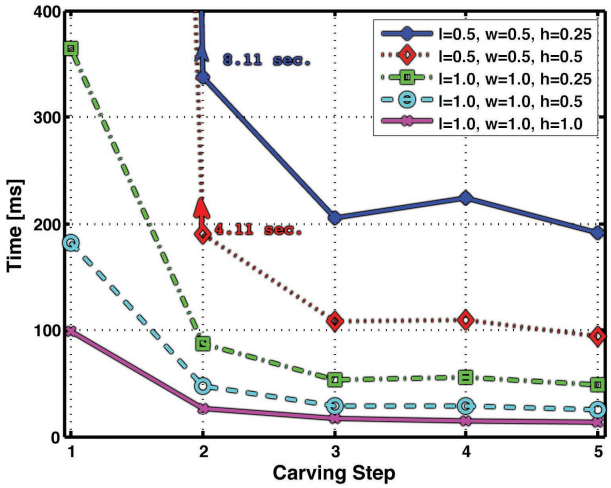

Figure 7. Time spent for carving away voxels. The time needed for carving decreases significantly with an increasing number of images.

4.2. Measurement Accuracy

Given a representative set of medical pills, the proposed modeling procedure is used to acquire 3D measurements based on five images of each object. We limit our experiments to pills which are symmetric along the z-axis for reasons described in Section 3.3.

In Figure 8 some results for the height estimation are

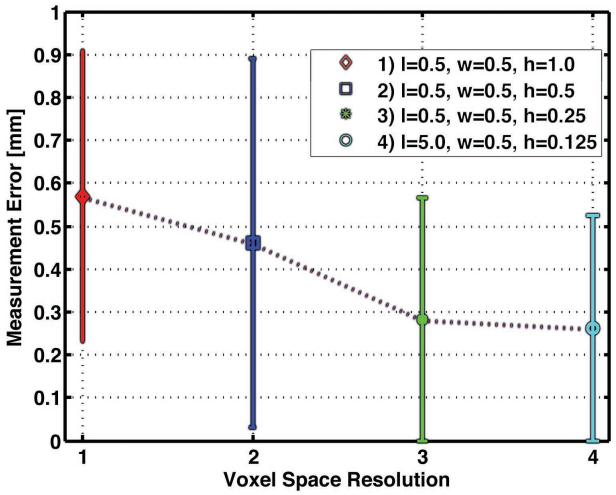

Figure 8. Results for height measurements on medical pills with an increasing amount of voxels.

depicted, given different settings for the voxel space resolution. In this evaluation process we used the mean measurement error and its standard deviation from the absolute values throughout the testset (see Figure 8).

At an equal spacing of $\delta x = \delta y = \delta z = 0.5$ the expected mean error is 0.56mm. This may be seen as a tradeoff, being suitable for a real system (see Table 2 for a summary of results). A main limitation for calculating measurements in the third dimension is the inherent necessity to track the planar marker target. For carving the voxel space in the "height" dimension accurately, the user has to acquire images at the shallowest angle possible. This might become a difficult task due to the limited extent of the target and the limited capabilities of the tracking algorithm. Furthermore, the angular limits also depend on the actual object size, since the object must be segmented in front of the checkerboard background pattern.

4.3. Monopoly in AR Mode

To demonstrate the advantageous features of our approach we modeled several objects used for playing the well-known Monopoly[4] game on the fly on the smartphone. Models of these objects are shown in Figure 9. Molla and Lepetit have recently presented some of their work in the context of AR for board games, also utilizing the Monopoly game as an example [15]. In their work the physical objects used in the game are recognized with CV-based object recognition methods, followed by the overlay of more visually appealing virtual models. By contrast, we create models of these physical objects using the method presented and use these models only for game-play.

[4]Monopoly^{TM}is the Trademark of Hasbro Company, Rhode Island, United States.

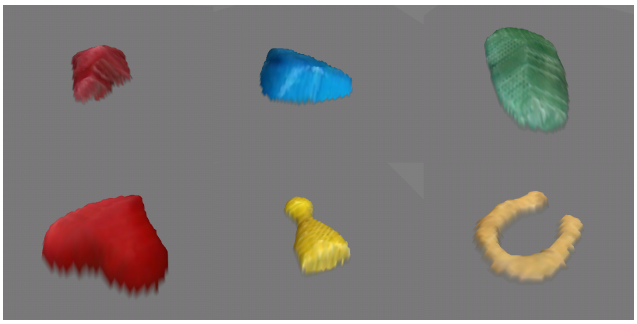

Figure 9. Several objects reconstructed for using the board game (small pyramids, blue stone, green stone, heart, yellow cone, horseshoe).

Figure 10. Snapshot from the augmented Monopoly game. Several of the objects modeled in advance are registered to the board game. Note that we use a grayscale board to make the colored objects more visible.

Figure 10 shows a scene from our Monopoly game with objects registered to the actual game board. The user can model individual objects in advance and use them later for interactive game-play.

5. Conclusion

In this paper we presented an approach to online model and measure small-scale objects interactively on mobile phone hardware. Our approach uses voxel carving and simple image-based rendering to generate visually appealing and accurate models of objects instantly. The models can easily be used for AR applications as demonstrated in an exemplary AR scenario. The approach is easy and intuitive to use. It gives immediate feedback to the users, who decide whether the result is sufficiently accurate for their needs or not.

Nevertheless, the current version of our application has several drawbacks that give room for further improvements. As we did not include any optimization for dedicated hardware, such as a GPU implementation of the voxel carving step, the CPU is currently used slightly inefficiently. Further improvements will resolve this issue, however. An-

other matter are the excluded geometric priors or color consistency to help with the modeling. Since the algorithms are based on the idea of visual hulls they not accurately model objects of highly convex structure. Nonetheless, a more elaborated modeling step may be included to improve results for this sort of models.

An important point for future improvements is to get rid of the marker target with the embedded checkerbord pattern. Although marker targets are advantageous in terms of efficient detectability, their optical appeal is poor. Using more advanced tracking methods could help with both, improving modeling accuracy using more evolved segmentation approaches, and with more general applicability of our approach without additional printouts and hardware.

Acknowledgements

This work was partially sponsored by the Christian Doppler Laboratory for Handheld Augmented Reality and partially supported by FFG grant # 822702.

References

[1] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nister, and M. Pollefeys. Towards Urban 3D Reconstruction from Video. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, 3DPVT '06, pages 1–8, Washington, DC, USA, 2006. IEEE Computer Society.

[2] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull Algorithm for Convex Hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[3] J. Bastian, B. Ward, R. Hill, A. van den Hengel, and A. Dick. Interactive Modelling for AR Applications. In *Proc. International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–8, 2010.

[4] B. G. Baumgart. *Geometric Modeling for Computer Vision.* PhD thesis, Stanford University, Stanford, CA, USA, 1974. AAI7506806.

[5] P. Bunnun, D. Damen, S. Subramanian, and W. W. Mayol-Cuevas. Interactive Image-based Model Building for Handheld Devices. In *Augmented Reality Supermodels Workshop (held in conjunction with ISMAR)*, pages 1–8, 2010.

[6] P. Bunnun and W. Mayol-Cuevas. OutlineAR: an Assisted Interactive Model Building System with Reduced Computational Effort. In *Proc. International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–8, 2008.

[7] C. H. Chien and J. K. Aggarwal. Volume/surface octrees for the representation of three-dimensional objects. *Comput. Vision Graph. Image Process.*, 36:100–113, November 1986.

[8] A. Hartl, C. Arth, and D. Schmalstieg. Instant Segmentation and Feature Extraction for Recognition of Simple Objects on Mobile Phones. In *Int. Workshop on Mobile Vision (in conjunction with CVPR)*, pages 17–24, 2010.

[9] S. Hauswiesner, M. Straka, and G. Reitmayr. Coherent image-based rendering of real-world objects. In *Proceedings of the 2011 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Game*, San Francisco, CA, 2011.

[10] G. Klein and D. Murray. Parallel Tracking and Mapping on a Camera Phone. In *Proc. International Symposium on Mixed and Augmented Reality (ISMAR)*, Orlando, October 2009.

[11] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16:150–162, February 1994.

[12] J. Lee, G. Hirota, and A. State. Modeling Real Objects Using Video See-Through Augmented Reality. In *In Proc. ISMR 01 (Int. Symp. on Mixed Reality*, pages 144–157, 2001.

[13] W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *In Proceedings of Twelfth Eurographics Workshop on Rendering*, pages 115–125, 2001.

[14] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based Visual Hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 369–374, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[15] E. Molla and V. Lepetit. Augmented Reality for Board Games. In *Proc. International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–8, 2010.

[16] Q. Pan, C. Arth, E. Rosten, and G. Reitmayr. Towards Rapid 3d Reconstruction on Mobile Phones from Wide-Field-of-View Images. In *Augmented Reality Supermodels Workshop (held in conjunction with ISMAR)*, pages 1–8, 2010.

[17] W. Piekarski and B. Thomas. Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer. In *Wearable Computers, 2001. Proceedings. Fifth International Symposium on*, pages 31 – 38, 2001.

[18] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *Comput. Vision Graph. Image Process.*, 40:1–29, October 1987.

[19] G. Simon. In-Situ 3D Sketching Using a Video Camera as an Interaction and Tracking Device. In *31st Annual Conference of the European Association for Computer Graphics - Eurographics 2010*, Norrköping Suède, 05 2010.

[20] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. Videotrace: Rapid Interactive Scene Modelling from Video. *ACM Trans. Graph.*, 26, July 2007.