High-Performance Multi-View Reconstruction

Christopher Zach Mario Sormann Konrad Karner
VRVis Research Center
Graz, Austria
{zach, sormann, karner}@vrvis.at

Abstract

We present a high performance reconstruction approach, which generates true 3D models from multiple views with known camera parameters. The complete pipeline from depth map generation over depth image integration to the final 3D model visualization is performed on programmable graphics processing units (GPUs). The proposed pipeline is suitable for long image sequences and uses a plane-sweep depth estimation procedure optionally employing robust image similarity functions to generate a set of depth images. The subsequent volumetric fusion step combines these depth maps into an impicit surface representation of the final model, which can be directly displayed using GPU-based raycasting methods. Depending on the number of input views and the desired resolution of the final model the computing times range from several seconds to a few minutes. The quality of the obtained models is illustrated with real-world datasets.

1. Introduction

Creating virtual 3D models solely from a set of digital input images is still an active research topic. The availability of cheap, although high quality digital cameras in combination with the newest generation of powerful programmable graphics hardware give an additional boost to the development of algorithms for 3D model generation. One advantage of using images as the only input source is the independency from the size of the object to be modeled.

This work presents high-performance dense mesh generation methods within the complete 3D reconstruction pipeline based on the computational power of modern programmable graphics hardware. There are at least two motivating reasons for a high-performance approach to 3D reconstruction: at first, some visual feedback is available to a human operator at interactive rates, which allows instant evaluation of the obtained results and immediate adjustment of parameters as necessary. Further, faster processing is always desirable, especially if the number of images is large in order to provide enough redundancy for a complete automated solution.

The proposed methods presented in this paper include a local depth estimation method based on the plane-sweep principle suitable for real-world application. This method is entirely performed by graphics hardware, thus utilizing the huge computational power of modern GPUs. Additionally, a GPU-based volumetric depth integration method is applied to obtain true 3D models from a set of depth maps, which is robust to mismatches and outliers present in depth images.

2. Previous Work

The first established multi-view depth estimation approach executed on programmable graphics hardware was presented by Yang et al. [14], who developed a fast stereo reconstruction method performed on graphics hardware by utilizing a plane sweep approach to find correct depth values. The proposed method uses projective texturing capabilities of 3D graphics hardware to project the given image onto the reference plane. Further, single pixel error accumulation for all given views is performed on the GPU as well. The number of iterations is linear in the requested resolution of depth values, therefore this method is limited to rather coarse depth estimation in order to fulfill the realtime requirements of their video conferencing application. Further, their approach requires a true multi-camera setup to be robust, since the error function is only evaluated for single pixels. As the application behind this method is a multi-camera teleconferencing system, accuracy is less important than realtime behaviour. In later work the method was made more robust using trilinear texture access to accumulate error differences within a window [12]. Their developed ideas were reused and improved to obtain a GPU-based dense matching procedure for a rectified stereo setup [13].

The basic GPU-based plane-sweep technique for depth estimation can be enhanced with implicit occlusion handling and smoothness constraints to obtain depth maps with higher quality. Woetzel and Koch [11] addressed occlusion occuring in the source images by a *best n out of m* and by a *best half-sequence* multi-view selection policy to limit the impact of occlusions on the resulting depth map. In order to obtain sharper depth discontinuities a shiftable correlation window approach was utilitized. The employed image similarity measure is a truncated sum of squared differences.

Cornelis and Van Gool [1] proposed several refinement steps performed after a plane-sweep procedure, which is used to obtain an initial depth map using a single pixel truncated SSD correlation score. Outliers in the initially obtained depth map are removed by a modified median filtering procedure, which may destroy fine 3D structures. These fine details are recovered by a subsequent depth refinement pass. Since this approach is based on single pixel similarity instead of a window based one, slanted surfaces and depth discontinuities are reconstructed more accurately compared with window-based approaches. The disadvantage of a pixel based similarity is that intensities are directly compared, which works well only under a constant brightness assumption. If illumination changes are expected, a similarity measure between the gray value patterns within a window is preferable.

Typically, the correlation windows used in realtime dense matching have a fixed size, which causes inaccuracies close to depth discontinuities. Since large depth changes are often accompanied by color or intensity changes in the corresponding image, adapting the correlation window to extracted edges is a reasonable approach. Gong and Yang [3] investigated in a GPU-based computational stereo procedure with an additional color segmentation step to increase the quality of the depth map near object borders.

We conclude this section by remarking that most work on GPU-accelerated depth estimation methods display only the resulting depth images, but typically no 3D geometry is shown, which allows easier qualitative evaluation of the result. Of course, most work in this field does not address the creation of 3D models, hence other criteria like speed for realtime settings are more important.

3. Small-Baseline Dense Depth Estimation using Plane-Sweep

Plane sweep techniques in computer vision are simple and elegant approaches to image based reconstruction with multiple views, since a rectification procedure as required in many traditional computational stereo methods is not required. The 3D space is iteratively traversed by parallel planes, which are usually aligned with a particular key view (Figure 1). The plane at a certain depth from the key view induces homographies for all other views, thus the reference images can be mapped onto this plane easily.

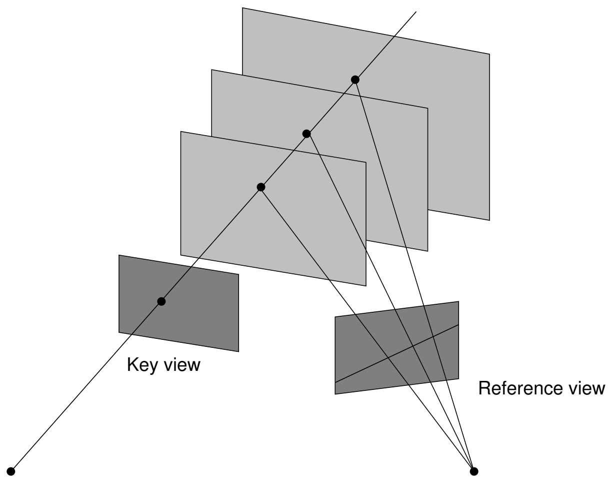


Figure 1. Plane sweeping principle. For different depths the homography between the reference plane and the reference view is varying. Consequently, the projected image of the reference view changes with the depth according to the epipolar geometry.

If the plane at a certain depth passes exactly through the surface of the object to be reconstructed, the color values from the key image and from the mapped references images should coincide at appropriate positions (assuming constant brightness conditions). Hence, it is reasonable to assign the best matching depth value (according to some image correlation measure) to the pixels of the key view. By sweeping the plane through the 3D space (by varying the planes depth wrt. the key view) a 3D volume can be filled with image correlation values similar to the disparity space image (DSI) in traditional stereo. Therefore the dense depth map can be extracted using global optimization methods, if depth continuity or any other constraint on the depth map is required.

Note, that a plane sweep technique in a two frame rectified stereo setup coincides with traditional stereo methods for disparity estimation. In these cases the homography between the plane and the reference view is solely a translation along the X-axis.

There are several techniques to make dense reconstruction approaches more robust in case of occlusions in a multiview setup. Typically, occlusions are only modeled implicitly in contrast to e.g. space carving methods, where the generated model so far directly influences visibility information. Here we discuss briefly two approaches to implicit occlusion handling:

- *Truncated scores:* The image correlation measure is calculated between the key view and the reference view and the final score for the current depth hypothesis is the accumulated sum of the truncated individual similarities. The reasoning behind this approach is that the effect of occlusions between a pair of views on the total score should be limited to favor good depth hypotheses supported by other image pairs.

- *Best half-sequence selection:* In many cases the set of images comprise a logical sequence of views, which can be totally ordered (e.g. if the camera positions are approximately on a line). Hence the set of images used to determine the score wrt. the key view can be splitted into two half-sequences, and the final score is the better score of these subsets. The motivation behind this approach is, that occlusion wrt. the key view appear either in the left or in the right half-sequence.

Dense depth estimation using plane sweeping as described in this section is restricted to small baseline setups, since for larger baselines occlusions should be handled explicitly. Additionally, the inherent fronto-parallel surface assumption of correlation windows yields inferior results in wide baseline cases.

3.1. Image Warping

In the first step, the reference images are warped onto the current 3D key plane $\pi = (n^\top, d)$ using the projective texturing capability of graphics hardware. If we assume the canonical coordinate frame for the key view, the reference images are transformed by the appropriate homography $H = K\left(R - t\,n^\top/d\right)K^{-1}$. K denotes the intrinsic matrix of the camera and $(R|t)$ is the relative pose of the reference view.

In order to utilize the vector processing capabilities of the fragment pipeline in an optimal manner, the (grayscale) reference images are warped wrt. four plane offset values d simultaneously. All further processing works on a packed representation, where the four values in the color and alpha channels correspond to four depth hypotheses.

3.2. Image Correlation Function

After a reference image is projected onto the current plane hypothesis, a correlation score for the current reference view is calculated and accumulated to the total correlation score of the plane hypothesis. The accumulation of the single image correlation scores depend on the selected occlusion handling policy: simple additive blending operations are sufficient if no implicit occlusion handling is desired. If the best half-sequence policy is employed, additive blending is performed for each individual subsequence and a final minimum-selection blending operation is applied.

To our knowledge, all published GPU-based dense depth estimation methods use the simple sum of absolute differences (SAD) for image dissimilarity computation (usually for performance reasons). By contrast, we have a set of GPU-based image correlation functions available, including the SAD, the normalized cross correlation (NCC) and the zero-mean NCC (ZNCC) similarity functions. The NCC and ZNCC implementations use sum tables for an efficient implementation [9]. Small row and column sums can be generated directly by sampling multiple texture elements within the fragment shader. Summation over larger regions can be performed using a recursive doubling approach similar to the GPU-based generation of integral images [4]. Full integral image generation is also possible, but precision loss is oberserved for the NCC and ZNCC similarity functions in this case.

For longer image sequences one cannot presume constant brightness conditions across all images, hence an optional prenormalization step is performed, which substracts the box-filtered image from the original one to compensate changes in illumation. If this prenormalization is applied, the depth maps obtained using the different correlation functions have similar quality.

3.3. Depth Extraction

In order to achieve high performance for depth estimation, we employ a simple winner-takes-all strategy to assign the final depth values. This approach can be easily and efficiently implemented on the GPU using the depth test for a conditional update of the current depth image hypothesis [14].

Unreliable depth values can be masked by a subsequent thresholding pass removing pixels in the obtained depth map, which have a low image correlation.

If the resulting depth map is converted to 3D geometry, staircasing artefacts are typically visible in the obtained model. In order to reduce these artefacts an optional selective, diffusion-based depth image smoothing step is performed, which preserves true depth discontinuities larger than the steps induced by the discrete set of depth hypotheses.

4. Depth Integration

In order to create true 3D models the set of generated depth maps must be combined into a single representation. Our method to create proper 3D models is based on an implicit volumetric representation, from which the final surface can be extracted by any implicit surface polygonization technique. The principles of robust fusion of several depth maps in the context of laser-scanned data was developed by

Curless and Levoy [2] and Hilton et al. [5]. We apply essentially the same technique on depth maps obtained by dense depth estimation procedures. The basic idea of volumetric depth image integration is the conversion of depth maps to corresponding 3D distance fields[1] and the subsequent robust averaging of these distance fields. The resolution and the accuracy of the final model are determined by the quality of the source depth images and the resolution of the target volume.

Instead of using an implicit representation of the surfaces induced by the depth images, one can merge a set of polygonal models directly [10]. Such an approach is sensitive to outliers and mismatches occuring in the depth images. A volumetric approach can combine several surface hypotheses and perform a robust voting in order to extract a more reliable surface. On the other hand, a volumetric range image fusion approach limits the size of 3D features found in the final model dependent on the voxel size.

Our implementation of the purely software based approach, which is based on [2], uses compressed volumetric representations of the 3D distance fields and can handle high resolution voxel spaces. Merging (averaging) of many distance fields induced by the corresponding depth maps is possible, since it is sufficient to traverse the compressed distance fields on a single voxel basis. Nevertheless our original implementation has substantial space requirements on external memory and consumes significant time to generate the final surface (usually in the order of several minutes). Hence this approach is not suitable for immediate visual feedback to the user. At least for fast and direct inspection of the 3D model it is reasonable to develop a very efficient volumetric range image integration approach again accelerated by the computing power of modern graphics hardware. Many steps in the range image integration pipeline are very suitable for processing on graphics hardware and significant speedup can be expected.

The overall procedure traverses the voxel space defined by the user slice by slice and generates a section of the final implicit 3D mesh representation in every iteration. Consequently, the memory requirements are very low, but immediate postprocessing (e.g. filtering) of the generated slices is limited. Although the general idea is very close to [2], several modifications are required to allow an efficient GPU implementations in the first instance. More importantly, the sensitivity to gross outliers frequenlty occuring in input depth maps is reduced by a robust voting approach. The details of our implementation are given in the next sections.

[1]Neither our approach nor [2] calculate true 3D distance fields, but only (optionally weighted) signed distances along predefined directions are employed.

4.1. Selecting the Volume of Interest

In general it is not possible to determine the volume of interest for reconstruction automatically. In case of small objects entirely visible on each of the source images, the intersection of the viewing frustra can serve as indicator for the volume to be reconstructed. Larger objects only partially visible in the source images (e.g. large buildings) require human interaction to select the reconstruction volume. Consequently, there exists a user interface for manual selection of the reconstructed volume. This application displays a set of e.g. 3D feature points generated by the image orientation procedure or 3D point clouds generated from dense depth maps. The user can select and adjust the 3-dimensional bounding box of the region of interest. Additionally, the user specifies the intented resolution of the voxel space, which is set to 256^3 voxels in our experiments.

4.2. Depth Map Conversion

With the knowledge of the volume of interest and its orientation, the voxel space is traversed slice by slice and the values of the depth images are sampled according to the projective transformation induced by the camera parameters and the position of the slice. Since the sampled depth values denote the perpendicular distance of the surface to the camera plane, the distance of a voxel to the surface can be estimated easily as the difference between the depth value and the distance of the voxel to the image plane (see also Figure 2). This difference is an estimated signed distance to the surface; positive values indicate voxels in front of the surface and negative values correspond to voxels hidden by the surface.

The source depth maps contain two additional special values: one value (in our implementation chosen as -1) indicates absent depth values, which may occur due to some depth postprocessing procedure eliminating unreliable matches from the depth map. Another value (0 in our implementation) corresponds to pixels outside some foreground region of interest, which is based on an optional silhouette mask in our workflow [7].

Consequently, the processed voxels fall into one of the following categories:

1. Voxels that are outside the camera frustum are labeled as *culled*.

2. Voxels with an estimated distance D to the surface smaller than a user-specified threshold T_{surf} are labeled as *near-surface* voxels ($|D| \leq T_{surf}$).

3. Voxels with a signed distance greather than this threshold are considered as definitely *empty* ($D > T_{surf}$).

4. The fourth category includes *occluded* voxels, which have a negative distance with a magnitude larger than the threshold ($D < -T_{surf}$).

5. If the depth value of the back-projected voxel indicates an absent value, the voxel is labeled as *unfilled*.

6. Voxels back-projecting into pixels outside the foreground regions are considered as empty.

These categories are illustrated in Figure 2. The threshold T_{surf} specifies essentially the amount of noise that is expected in the depth images.

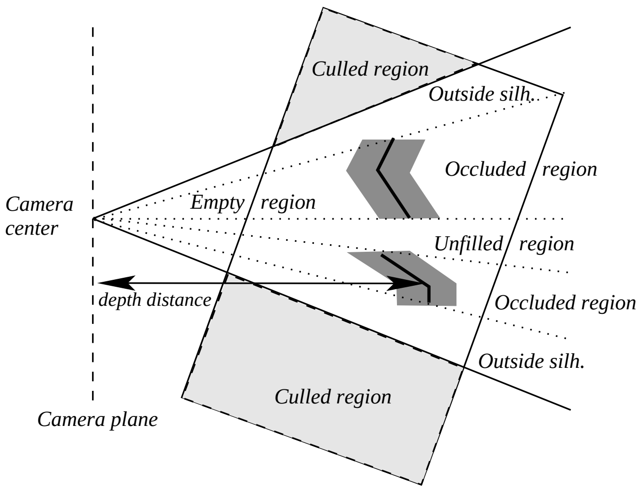


Figure 2. Classification of the voxel according to the depth map and camera parameters. Voxels outside the camera frustum are initially labeled as culled. Voxels close to the surface induced by the depth map are near-surface voxels (on both sides of the surface, indicated by shaded regions). Voxels with a distance larger than a threshold are either empty or occluded, depending on the sign of the distance.

In many reconstruction setups it is possible to classify culled voxel depth values immediately. If the object of interest is visible in all images, culled voxels are outside the region to be reconstructed and can be classified as empty instantly. Declaring culled voxels as unfilled may generate unwanted clutter due to outliers in the depth maps. If the object to be reconstructed is only partially visible in the images, voxels outside the viewing frustum of a particular depth map do not contribute information and are therefore labeled as unfilled. The choice between these two policies for handling culled data is specified by the user. Consequently, the 6 branches described above correspond to four voxel categories.

A fragment program determines the status of voxels and updates an accumulated slice buffer for every given depth image. This buffer consists of four channels in accordance to the categories described above:

1. The first channel accumulates the signed distances, if the voxel is a near-surface voxel.

2. The second channel counts the number of depth images, for which the voxel is empty.

3. The third channel tracks the number of depth images, for which the voxel is occluded.

4. The fourth channel counts the number of depth images, for which the status of the voxel is unfilled.

Thus, a simple but sufficient statistic for every voxel is accumulated, which is the basis for the final isosurface determination. Algorithm 1 outlines the incremental accumulation of the statistic for a voxel, which is executed for every provided depth image. The accumulated statistic for a voxel is a quadruple comprising the components as described above. In addition to the user-specified parameter T_{surf}, another threshold T_{occ} can be specified, which determines the border between occluded voxels and again unfilled voxels located behind the surface. This threshold is set to $10 \cdot T_{surf}$ in our experiments.

4.3. Isosurface Determination and Extraction

After all available depth images are processed, the target buffer holds the coarse statistic for all voxels of the current slice. The classification pass to determine the final status of every voxel is essentially a voting procedure. This step assigns the depth distance to the final surface to every voxel, such that the isosurface at level 0 corresponds with the merged 3D model. For efficiency the voting procedure uses only the statistics acquired for the current voxel, but does not inspect neighboring voxels. Algorithm 2 presents the utilized averaging procedure to assign the signed distance to the final surface.

Up to now the discussed steps in the volumetric range image integration pipeline, depth map conversion and fusion, run entirely on graphics hardware. After the GPU-based computation for one slice of the voxel space is finished, the isovalues of the current slice are transformed into a triangular mesh on the CPU [6] and added to the final surface representation. This mesh can be directly visualized and is ready for additional processing like texture map generation. Instead of generating a surface representation from the individual slices a 3D texture can be accumulated alternatively, which is suitable for volume rendering techniques. The main portion of this approach is performed

Algorithm 1 Procedure to accumulate the statistic for a voxel

```
Procedure $stat$ = AccumulateVoxelStatistic
Input: Camera image plane $imagePlane$, near-surface threshold $T_{surf}$, $T_{occ} > T_{surf}$, $\#Images$
Input: depth image $D$, projective texture coordinate $stq$, 3D voxel position $pos$
Input: Voxel statistics: $stat = (\sum D_i, \#Empty, \#Occluded, \#Unfilled)$ (a quadruple)

  $st \leftarrow stq.xy/stq.z$ {Perspective division}
  if $st$ is inside $[0,1] \times [0,1]$ then
    $depth \leftarrow tex2D(D, st)$ {Gather depth from range image}
    if $depth > 0$ then
      $dist \leftarrow depth - imagePlane \cdot pos$ {Calculate signed distance to the surface}
      if $dist > T_{surf}$ then
        increment $\#Empty$ {Too far in front of surface}
      else if $dist < -T_{occ}$ then
        increment $\#Unfilled$ {Very far behind the surface}
      else if $dist < -T_{surf}$ then
        increment $\#Occluded$ {Too far behind the surface}
      else
        $\sum D_i \leftarrow \sum D_i + dist$ {Near-surface voxel}
      end if
    else
      if $depth = 0$ then
        $stat \leftarrow (0, \#Images + 1, 0, 0)$ {Declare voxel definitely as empty}
      else
        increment $\#Unfilled$
      end if
    end if
  else
    {Execute one of the following lines, depending on handling of culled voxels:}
    increment $\#Empty$, or {Handle culled voxel as empty}
    increment $\#Unfilled$ {Alternatively, handle culled voxel as unfilled}
  end if
  Return $stat$
```

again entirely on the GPU and does not involve substantial CPU computations. In constrast to a slice-based incremental isosurface extraction method, this direct approach requires the space for a complete 3D texture in graphics memory. Since modern 3D graphics hardware is equipped with large amounts of video memory, the 16MB required by a 256^3 voxel space are affordable. Rendering an isosurface directly from the volumetric data requires additional calculation of surface normals, which are directly derived from the gradients at every voxel. By using a deferred rendering approach, computation of the gradient can be limited to the actual surface voxels and the additional memory consumption is minimal.

Algorithm 2 Procedure to calculate the final surface distance for a voxel

```
Procedure $result$ = AverageDistance
Input: User specified constants: $\#RequiredDefinite$, $\#RequiredOccluded$, $UnknownLabel$
Input: Voxel statistics: $\sum D_i$, $\#Empty$, $\#Occluded$, $\#Unfilled$

  $\#Definite \leftarrow \#Images - \#Occluded - \#Unfilled$
  if $\#Definite < \#RequiredDefinite$ then
    if $\#Occluded \geq \#RequiredOccluded$ then
      $result \leftarrow -\infty$
    else
      $result \leftarrow UnknownLabel$
    end if
  else
    $\#NearSurface \leftarrow \#Images - \#Empty - \#Unfilled$
    if $\#NearSurface \geq \#Empty$ then
      $result \leftarrow \sum D_i / \#NearSurface$
    else
      $result \leftarrow +\infty$
    end if
  end if
  Return $result$
```

4.4. Implementation Remarks

Tracking the statistic for each voxel in the current slice requires a four channel buffer with floating point precision to accumulate the distance values for near-surface voxels. By normalizing the distance of these voxels from $[-T, T]$ to $[-1, 1]$ a half precision buffer (16 bit floating point format) is usually sufficient. Furthermore, the final voxel values can be transformed to the range $[0, 1]$ and a traditional 8 bit fix-point buffer offers adequate precision. Using low-precision buffers decreases the volume integration time by about 30%.

5. Results

This section provides visual and timing results for some real datasets. The timings are given for a PC hardware consisting of a Pentium4 3GHz processor and an NVidia Geforce 6800 graphics card. All source views are resized to 512×512 pixels beforehand, and the obtained depth images have the same resolutions (unless noted otherwise). Partially available foreground segmentation data is not used in these experiments.

The first dataset depicted in Figure 3(a) shows one source image (out of 47) displaying a small statue. The images are taken in a roughly circular sequence around the statue. The camera is precalibrated and the relative poses of the images are determined from point correspondences found in adjacent views. From the correspondences and the camera parameters a sparse reconstruction can be triangulated, which is used by a human operator to determine a 3D box enclosing the voxel space of interest. The extension of this box is used to determine depth range used in the subsequent plane-sweep step, which took 53s to generate 45 depth images in total (Figure 3(b)). In the depth estimation procedure 200 evenly distributed depth hypotheses are tested using the SAD for a 5×5 window. In order to compensate illumination changes in several view triplets, the source images were normalized by subtracting its local mean image. Black pixels indicate unreliable matches, which are labeled as unfilled before the depth integration procedure. These depth maps are integrated in just over 4 seconds to obtain a 256^3 volume dataset as illustrated in Figure 3(c). The isosurface displayed in Figure 3(d) can be directly extracted using a ray-casting approach on the GPU [8]. Almost all of the clutter and artefacts outside the proper statue are eliminated by requiring at least 7 definite values for the statistic of a voxel.

The result for another dataset consisting of 43 images is shown in Figure 4(b), for which one source image is depicted in Figure 4(a). The same procedure as for the previous dataset is applied, from which a set of 41 depth images is obtained in the first instance. Plane-sweep depth estimation using the ZNCC correlatio with 200 depth hypotheses requires 97.7s in all to generate the depth maps. The subsequent depth image fusion step requires 4s to yield the volumetric data illustrated in Figure 4(b).

Note that these timing reflect the creation time for rather high-resolution models. If all resolutions are halved ($256 \times 256 \times 100$ depth images and 128^3 volume resolution), the total depth estimation time is 13s and the volumetric integration time is less than 1s for this dataset. We believe that these timing results allow our method to qualify as an interactive modeling approach.

The visual result for another dataset consisting of 16 source views is shown in Figure 4(c) and (d). Depth estimation for 14 views took 34.2s using a 5x5 ZNCC with a best-half-sequence occlusion strategy (200 tentative depth values). Without an implicit occlusion handling approach parts of the sword are missing. Volumetric integration requires another 1.8s to generate the isosurface shown in Figure 4(d).

6. Summary and Conclusions

In this work we demonstrated, that generating proper 3D models from a set of depth images can be achived at interactive rates using the processing power of modern GPUs. The quality of the obtained 3D models depends on the grade of the source depth maps and on the redundancy of data, but the voting scheme is robust in case of outliers usually generated by pure local depth estimation procedures.

Future work needs to address the depth map generation in particular. A variational approach to depth estimation accelerated by the GPU is already available, which provides very good depth maps for datasets suitable for a variational method. However, many real-world datasets consisting of many images contain some variations in illumination and lighting conditions, therefore variational approaches based on the constant intensity assumption fail for many real datasets.

On the other hand, a purely local winner-takes-all approach to depth estimation is very unreliable in low-textured regions. We have started preliminary work on a GPU based scanline optimization procedure in order to enhance the quality of the depth maps. The major obstacle so far is the huge size of the disparity space image utilized in the basic dynamic programming approaches to depth extraction.

Further work needs to be done to obtain cleaner isosurfaces suitable for subsequent processing. In particular, the remaining clutter caused by severe mismatches could be removed by detecting isolated near-surface voxels.

Acknowledgments

This work has been done in the VRVis research center, Graz and Vienna/Austria (http://www.vrvis.at), which is partly funded by the Austrian government research program Kplus.

References

[1] N. Cornelis and L. Van Gool. Real-time connectivity constrained depth map computation using programmable graphics hardware. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1099–1104, 2005.

[2] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH '96*, pages 303–312, 1996.

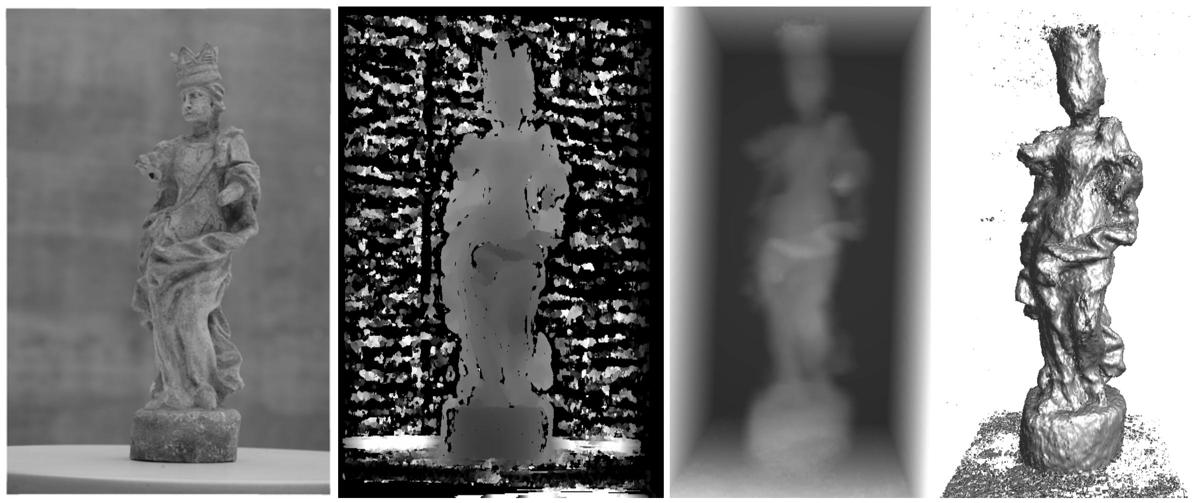
[3] M. Gong and R. Yang. Image-gradient-guided real-time stereo on graphics hardware. In *Fifth International Conference on 3-D Digital Imaging and Modeling*, pages 548–555, 2005.

[4] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra. Fast summed-area table generation and its applications. In *Proceedings of Eurographics 2005*, pages 547–555, 2005.

[5] A. Hilton, A. J. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *European Conference on Computer Vision (ECCV)*, pages 117–126, 1996.

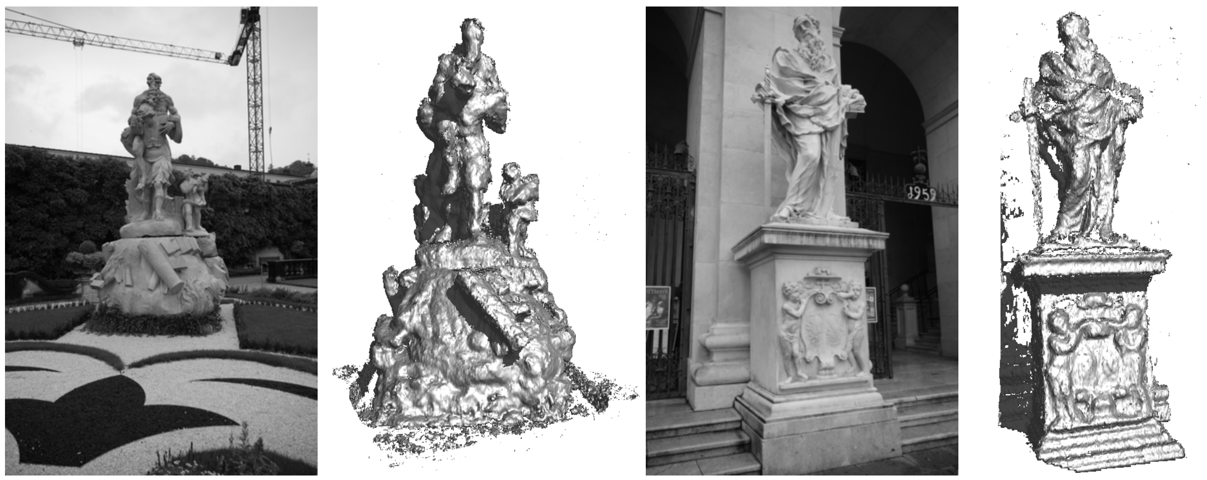
[6] W. Lorenson and H. Cline. Marching Cubes: A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH '87*, pages 163–170, 1987.

[7] M. Sormann, C. Zach, J. Bauer, K. Karner, and H. Bischof. Automatic foreground propagation in image sequences for 3d reconstruction. In *Proc. 27th DAGM Symposium*, pages 93–100, 2005.



(a) One source image (b) One depth image (c) Direct volume rendering (d) Shaded isosurface

Figure 3. Visual results for a small statue dataset generated from a sequence of 47 images. The total time to generate the depth maps and the final volumetric representation is less than 1 min.



(a) One source image (of 43) (b) Shaded isosurface (102s) (c) One source image (of 16) (d) Shaded isosurface (36s)

Figure 4. Source views and isosurfaces for two real-world datasets.

[8] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. A simple and flexible volume rendering framework for graphics-hardware-based raycasting. In *Proceedings of Volume Graphics*, pages 187–195, 2005.

[9] D.-M. Tsai and C.-T. Lin. Fast normalized cross correlation for defect detection. *Pattern Recognition Letters*, 24(15):2625–2631, 2003.

[10] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of SIGGRAPH '94*, pages 311–318, 1994.

[11] J. Woetzel and R. Koch. Real-time multi-stereo depth estimation on GPU with approximative discontinuity handling. In *1st European Conference on Visual Media Production (CVMP 2004)*, pages 245–254, 2004.

[12] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware. In *Conference on Computer Vision and Pattern Recognition*, 2003.

[13] R. Yang, M. Pollefeys, and S. Li. Improved real-time stereo on commodity graphics hardware. In *CVPR 2004 Workshop on Real-Time 3D Sensors and Their Use*, 2004.

[14] R. Yang, G. Welch, and G. Bishop. Real-time consensus based scene reconstruction using commodity graphics hardware. In *Proceedings of Pacific Graphics*, pages 225–234, 2002.