A Virtual Reality Multi-Sensor 3d Reconstruction System

Markus Friedrich, Kyrill Schmid
Ludwig-Maximilians-University Munich
Munich, Germany
{markus.friedrich / kyrill.schmid}@ifi.lmu.de

***Abstract*—We propose the concept for a distributed multi-sensor 3d reconstruction system that includes a virtual reality (VR) scan visualization component. It fuses depth data from two different sources: A mobile time-of-flight (ToF) depth sensor and an image-based 3d reconstruction module that estimates depth based on camera images. The continuously evolving 3d data set is visualized on-the-fly in a specialized VR application which allows for interactive scanning scenarios. Currently, two sensors are considered but we plan to extend the system to support a multitude of depth sensors that iteratively contribute to the global data set.**

***Index Terms*—depth sensor fusion, distributed 3d reconstruction, virtual reality, data visualization**

I. Introduction

The advent of cheap depth sensors shipped in consumer devices like the Microsoft© Kinect™ or the Asus© Zenfone AR™ has led to a boost in 3d reconstruction research (see e.g. [GPB11], [SKSC13], [FG11]) and 3d scanning application development. At the same time, virtual reality (VR) hardware has left the labs and is now mature enough for widespread adoption.
In this work, we introduce a concept that combines the best of both worlds through a distributed multi-sensor 3d scanning system that allows a user to scan real-world objects while providing instantaneous feedback on the quality and shape of the already measured object parts using a VR-based visualization.
The system uses a Zenfone AR™ device which is equipped with a time-of-flight (ToF) sensor as main source for depth values. Since the used VR head-mounted display (HMD) comes with a front-facing camera, we would like to use it to compute additional depth values in complement to those provided by the Zenfone AR™ using an image-based 3d reconstruction approach [SCD$^+$06]. We exploit the fact that for each image, the pose of the camera is known since it is fixed on the HMD and the HMD computes its exact pose which is needed for VR applications. In addition, the internal camera parameters (e.g. focal length and pixel resolution) are well-known and do not change over time.
Our system consists of the following core components:

- A 3d reconstruction component that estimates depth values based on images that come from the front-facing camera within seconds ($\leq$ 2s) for each incoming image.
- A depth value fusion strategy that merges depth values from different sources into a single consistent data set in near real-time (maximum delay $\leq$ 2s).
- A VR visualization module that is able to blend depth measurements into the camera image of the real-world object without artefacts and noticable delays in order to avoid VR sickness symptoms.

Currently, the system uses two depth sensors. Next steps involve the support of a multitude of depth sensors that all contribute to the global data set.

II. Background

A. Rigid Body Poses

The 3d pose (position and orientation) of a rigid body (e.g. a camera) can be described by 4×4 matrices of the form

$$T = \left(\begin{array}{c|c} R & t \\ \hline 0_{1\times 3} & 1 \end{array} \right),$$

where $t \in \mathbb{R}^3$ is the translation vector and R is a 3×3 rotation matrix. R is orthogonal and has a determinant of $+1$. T represents a rigid body pose and has thus six degrees of freedom (three for translation and three for rotation). The matrix T_{AB} represents the coordinate frame transformation from frame B to frame A.

B. Voxels & Truncated Signed Distance Functions

Voxels can be seen as 3d pixels that discretize a certain subset of the 3d real Euclidean point space. Each voxel stores a domain-specific n-tuple (e.g. signed surface distance and color). When stored in a regular grid, space complexity is $\mathcal{O}(n^3)$. More space efficient data structures exist, e.g. hierarchical space partitioning schemes or spatial hash maps [NZIS13].
A signed distance function $d : \mathbb{R}^3 \mapsto \mathbb{R}$ maps a point $\mathbf{X}$ to a signed value that represents the distance between the point and the surface of an object. Thus, the surface is implicitly defined by the zero-set of d: $\{\mathbf{X} \in \mathbb{R}^3 : d(\mathbf{X}) = 0\}$.
Truncated signed distance functions (TSDFs) are only defined for a certain distance interval and are - in their discretized form - widely used for depth sensor fusion (e.g. in [GPB11]). Our system uses a voxel-based surface representation in combination with discretized TSDFs, where each voxel stores a TSDF value, a color and a fusion weight.

C. 3d Reconstruction

The goal of 3d reconstruction is to estimate the surface of real-world objects based on sensor data. Our system currently foresees two different depth value sources: A ToF sensor that measures the phase shift between an emitted and the corresponding reflected infrared light pulse in order to estimate the distance to an object. And a method that estimates depth using techniques from stereophotogrammetry that work on calibrated camera images [SCD+06]. The term calibrated means that for each image, the corresponding pose T_{WC} of the camera is known and can be used to transform a point from the camera coordinate frame to the world coordinate frame.

III. Related Work

Real-time 3d reconstruction, depth value fusion and voxel visualization in real-time is a well-established field of research [GPB11], [SKSC13], [FG11]. One example is the work of Graber et al. [GPB11] that uses similar methods for 3d reconstruction and fusion than we do. The main difference to our work is the missing VR approach for visualization. A combination of 3d scanning and VR is proposed in [DSLU15]. The authors describe a VR-based, user-guided 3d scanning simulator that uses the VR controller as a virtual depth sensor. Created point clouds are visualized within the VR application. While related to our system, it does not deal with real data.

IV. Concept

The current system design foresees four hardware components: A workstation for sensor fusion and data set visualization (1), a workstation for 3d reconstruction based on camera images (2), a VR controller with mounted depth sensor (3, Asus© Zenfone AR™) and the VR HMD (4, HTC© Vive™) as display and camera.
The depth sensor is attached to the wireless VR controller in order to get its pose in the VR system's coordinate frame (up to a fixed offset). This is necessary in order to transform incoming depth values into the world coordinate frame for sensor fusion. The user holds this controller during the scan process and moves it around the object that should be scanned. The HMD displays the scene which contains both, the current camera image and the rendered global data set containing the already scanned object parts. Without showing the latest camera image, the user would lose orientation during the scan process. See Figure 1 for an overview of the system.

A. 3d Reconstruction

The 3d reconstruction component (Figure 1, component 2) uses a variant of the plane-sweep method to retrieve depth values from a series of calibrated images recorded by the front-facing camera [Col96]. It is optimized for modern GPUs and achieves near real-time performance on images with HD resolution.
Its principles are derived from basic triangulation. Corresponding image points are searched with help of pixel color-based similarity measures within calibrated views. With known correspondences in at least two views, the scene point can be

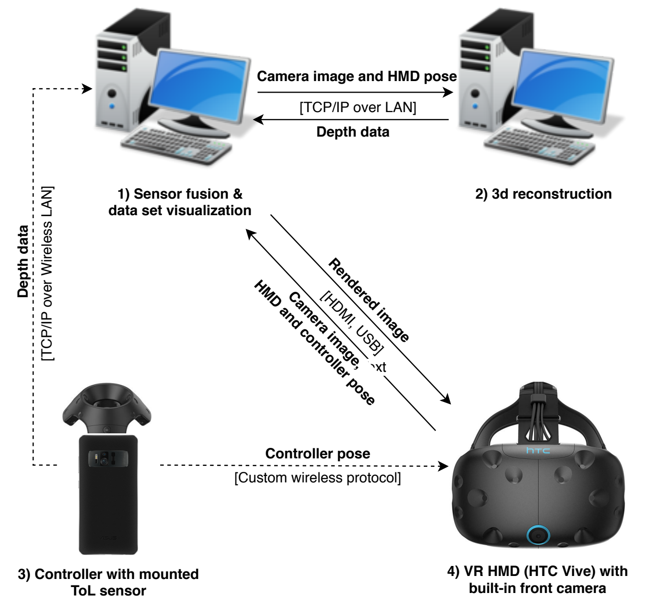
Camera image and HMD pose
[TCP/IP over LAN]
Depth data
1) Sensor fusion & data set visualization
2) 3d reconstruction
Depth data
[TCP/IP over Wireless LAN]
Rendered image
[HDMI, USB]
Camera image, HMD and controller pose
Controller pose
[Custom wireless protocol]
3) Controller with mounted ToL sensor
4) VR HMD (HTC Vive) with built-in front camera

Fig. 1. Overall system architecture. Communication protocols are denoted in square brackets. Note that the selected VR system also uses two additional sensor boxes for HMD pose estimation. Since their role is transparent, they are ommitted in this diagram.

estimated.
The detailed explanation of the method is based on Figure 2. A reference camera view and two other camera views with

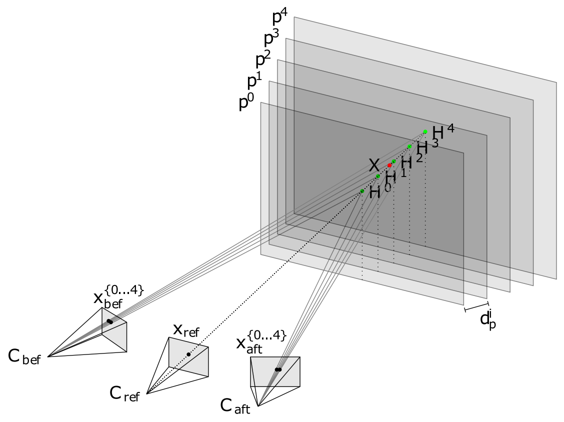

Fig. 2. The plane-sweep setting for a reference view, two sensor views and constant plane distances d_p^i.

centers $\mathbf{C}_{\text{ref}}$,$\mathbf{C}_{\text{bef}}$ and $\mathbf{C}_{\text{aft}}$ with partially overlapping frustums are considered.
The scene point $\mathbf{X}$ (red point) corresponding to an image point $\mathbf{x}_{\text{ref}}$ (black point) in the reference view is to be estimated. $\mathbf{X}$ has to be located on the view ray v_{ref} through the camera center $\mathbf{C}_{\text{ref}}$ and the image point $\mathbf{x}_{\text{ref}}$.
In order to estimate $\mathbf{X}$, a set of n_p so-called sweeping planes $\{p^0, \ldots, p^{n_p-1}\}$ parallel to the view plane of the reference camera are placed along v_{ref}. They divide the reconstruction volume. The distances between the sweeping planes, $d_p^i, i \in \{0, \ldots, n_p - 1\}$, can be user-defined and constant or

determined by more sophisticated methods.
The intersection points between sweeping planes and view ray v_{ref}, $\mathbf{H}^i$ (green points), are estimates of the location of $\mathbf{X}$. In order to determine the best candidate $\mathbf{H}^b$, all $\mathbf{H}^i$ are consecutively projected on the view planes of the views V_{bef} and V_{aft}, resulting in image points $\mathbf{x}^i_{\text{bef}}$ and $\mathbf{x}^i_{\text{aft}}$ (black points). Correlation measures $c^i_{\text{bef}}, c^i_{\text{aft}} \in [0,1]$ are then computed based on the color values of the view images I_{ref} and I_{bef} at $\mathbf{x}^i_{\text{ref}}$ and $\mathbf{x}^i_{\text{bef}}$ (resp. I_{ref} and I_{aft} at $\mathbf{x}_{\text{ref}}$ and $\mathbf{x}_{\text{aft}}$). Suitable correlation measures are e.g. the Euclidean distance or normalized cross-correlation [Lew95]. The best candidate index b is then determined by

$$b = \underset{i \in \{0,\dots,n_p-1\}}{\operatorname{argmax}} (\max(c^i_{\text{bef}}, c^i_{\text{aft}})).$$

Thus, the sweeping plane with the best overall correlation value is chosen. The plane-sweep method is parallelizable (one thread per image point in the reference view) and can therefore be executed efficiently on GPUs In addition, the quality is adjustable by changing the sweeping plane distances d^i_p which alters the number of sweeping planes n_p.

B. Depth Sensor Fusion

All scene points estimated during a 3d reconstruction iteration and all new scene points from the ToF sensor need to be integrated into the global data set (Note that the ToF sensor automatically converts a scalar depth value to a full 3d scene point and all scene points are relative to the coordinate frame of their corresponding camera or sensor). This is done by the fusion component (Figure 1, component 1).
For fusion, a simple and well-known voxel fusion approach was selected [CL96]. For each voxel, the estimated signed distance D to the real surface is stored together with a fusion weight W which serves as a confidence measure.
Each scene point is transformed from camera-space to world-space using T_{WC}, resulting in the world-space scene point set $\{s_i\}$. The estimated distance to the surface, is defined for a certain interval t_{max} along the scene point's view ray v_i (with a distance of 0 at the exact position of the scene point). This represents the TSDF $d_i(\cdot)$ for that particular scene point defined on its view ray, see Figure 4 for an example.
The view ray interval in world-space is then discretized based on the edge length of a voxel e_v. For each sample point $\mathbf{X}$, $d_i(\cdot)$ is evaluated. The set of resulting signed distances is then fused with the voxels in the grid at $\mathbf{X}$. The new distance value D^* at grid position $\mathbf{X}$ is computed by

$$D^*(\mathbf{X}) = \frac{W(\mathbf{X})D(\mathbf{X}) + w_i d_i(\mathbf{X})}{W(\mathbf{X}) + w_i}, \tag{1}$$

where w_i equals the correlation value of the corresponding scene point. The equation for calculating the new fusion weight W^* is

$$W^*(\mathbf{X}) = W(\mathbf{X}) + w_i. \tag{2}$$

Figure 3 shows an example of this fusion scheme.
In [CL96], a grid in combination with run-length encoding (RLE) for data compression is employed as voxel storage

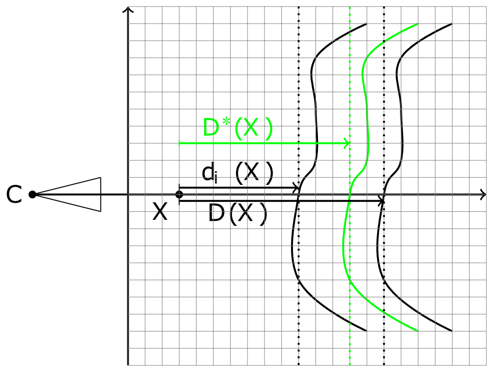

Fig. 3. The signed distance to the surface, $D(\mathbf{X})$, at position $\mathbf{X}$ is updated by the signed distance, $d_i(\mathbf{X})$ at that position. The influence of both values is determined by the weights $W(\mathbf{X})$ and w_i. The new signed distance $D^*(\mathbf{X})$ is illustrated in green.

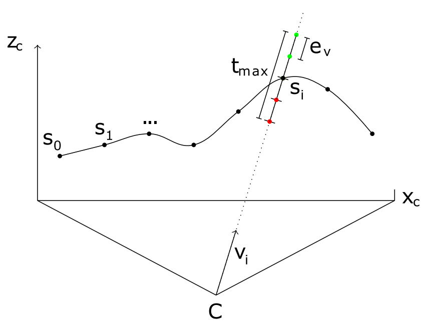

Fig. 4. Illustration of the TSDF value generation process reduced to the xz-plane in camera coordinates. Green dots indicate positive signed distances, red dots negative ones.

data structure. A more elaborate data structure based on hash mapping principles was tested for the proposed system concept [NZIS13]. It stores small cubes of $8 \times 8 \times 8$ voxels in a spatial hash map with a hash function proposed by Teschner et al. [THM$^+$03]. It reads

$$h(x,y,z) = ((xa) \oplus (yb) \oplus (zc)) \bmod n_b,$$

where a = 86,028,157, b = 73,856,093, c = 19,349,663 are large prime numbers, n_b is the number of hash map buckets and $\oplus$ is the bitwise XOR operator. It is parametrized by the integer world-space position of the voxel cube $(x,y,z)^T$.

C. Visualization

The visualization of implicit surfaces stored in voxels can be done using sophisticated ray-tracers like the one employed in [Gra11]. Another option would be to generate a polygon mesh with help of the marching cubes algorithm [LC87] which is then rendered using standard 3d engines. Sophisticated 3d point cloud renderers can be employed as well but then only voxels with a signed distance of 0 can be considered in order to avoid cluttering.
We have implemented a basic renderer that retrieves voxel cubes from the hash map and renders small camera aligned rectangles for each voxel. See Figure 5 for details. Figure 6 shows a low-res voxel visualization with ca. 20,000 voxels.

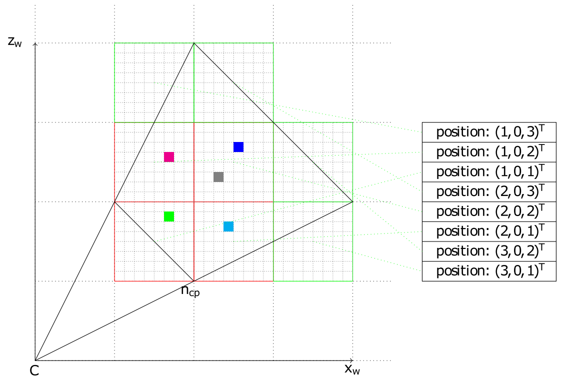

Fig. 5. The render pipeline. The scene camera frustum (black lines) is defined by the camera origin $\mathbf{C}$, the near clipping plane n_{cp} and the far clipping plane f_{cp} (its orientation is indicated by the orientation of the black triangle). Eight potential voxel cubes (green and red lines) lie within or intersect with the frustum. Their positions are saved in the query voxel cube position list which is used to query for voxel cubes within the frustum. Note that only those voxel cubes surrounded by red lines are existent in the fusion data structure.

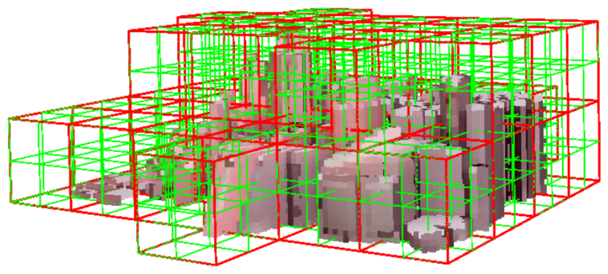

Fig. 6. Voxel set test rendering with around $20,000$ voxels. The green boxes indicate voxel cubes.

V. Status Quo, Questions and Future Work

In context of this project, we prototyped several crucial parts of the system. We implemented the plane-sweep reconstruction method described in Chapter IV-A, the fusion technique and the hash map data structure discussed in Chapter IV-B for GPUs using OpenCL [Mun11]. Conducted benchmarks revealed that both meet our requirements in terms of processing speed and robustness. We also experimented with different visualization techniques for implicit surfaces represented through discrete TSDFs. The method sketched in Chapter IV-C turned out to be a good trade-off between visual quality and speed. In addition, we developed a JSON-based message format to transport depth data from the mobile ToF sensor to the sensor fusion system.

The following questions need to be addressed in future work:

- **Connectivity:** We currently use a Wireless LAN connection between the mobile depth sensor and the fusion component. In order to extend battery life, other options need to be investigated.
- **Data distribution:** In future iterations of the system, multiple depth sensors should contribute to the scan result. The amount of sensor data might easily exceed the memory capacity of the GPU. Thus, intelligent data streaming and distribution schemes have to be developed.
- **3d reconstruction:** The plane-sweep method is vulnerable to complex real-world lighting conditions, especially if object surfaces have non-diffuse reflection properties. We need to evaluate more robust 3d reconstruction techniques.
- **Sensor fusion:** Each depth sensor type has its own maximum measurement precision. The fusion mechanism should account for that and prefer high-precision over low-precision measurements.
- **Visualization:** We need to investigate rendering techniques that are able to seamlessly visualize scan data together with real-world camera images.

Next steps include the integration of all system components into a coherent 3d reconstruction and visualization system.

References

[CL96] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[Col96] R.T. Collins. A space-sweep approach to true multi-image matching. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pages 358–363, 1996.

[DSLU15] Malvin Danhof, Tarek Schneider, Pascal Laube, and Georg Umlauf. A virtual-reality 3d-laser-scan simulation. In *Proceedings of the 2015 BW-CAR Symposium on Information and Communication Systems (SInCom)*, volume 2, 01 2015.

[FG11] Simon Fuhrmann and Michael Goesele. Fusion of depth maps with multiple scales. In *ACM Transactions on Graphics (TOG)*, volume 30, page 148. ACM, 2011.

[GPB11] Gottfried Graber, Thomas Pock, and Horst Bischof. Online 3D reconstruction using convex optimization. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 708–711, November 2011.

[Gra11] Gottfried Graber. Realtime 3D Reconstruction. Master's thesis, Graz University of Technology, 2011.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, August 1987.

[Lew95] JP Lewis. Fast normalized cross-correlation. In *Vision interface*, volume 10, pages 120–123, 1995.

[Mun11] Aaftab Munshi. The OpenCL Specification Version: 1.1. https://www.khronos.org/registry/cl/specs/opencl-1.1.pdf, 2011. [Online; accessed 28/06/2018].

[NZIS13] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D Reconstruction at Scale using Voxel Hashing. *ACM Transactions on Graphics (TOG)*, 2013.

[SCD+06] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 519–528, Washington, DC, USA, 2006. IEEE Computer Society.

[SKSC13] F. Steinbruecker, C. Kerl, J. Sturm, and D. Cremers. Large-Scale Multi-Resolution Surface Reconstruction from RGB-D Sequences. Sydney, Australia, 2013.

[THM+03] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomeranets, and Markus Gross. Optimized spatial hashing for collision detection of deformable objects. *Proceedings of the 18th International Workshop on Vision, Modeling and Visualization*, 2003.