In: http://opus.ub.uni-bayreuth.de/volltexte/2010/661/ urn:nbn:de:bvb:703-opus-6619 March 10, 2010, Bayreuth/Germany

Multi-View Reconstruction in-between Known Environments

Stefan KUHN and Dominik HENRICH
Lehrstuhl für Angewandte Informatik III (Robotik und Eingebettete Systeme)
Universität Bayreuth, D-95440 Bayreuth, Germany
{Stefan.Kuhn | Dominik.Henrich}@uni-bayreuth.de

***Abstract*—We present a novel multi-view 3D reconstruction algorithm which unifies the advantages of several recent reconstruction approaches. Based on a known environment causing occlusions and on the cameras' pixel grid discretization, an irregular partitioning of the reconstruction space is chosen. Reconstruction artifacts are rejected by using plausibility checks based on additional information about the objects to be reconstructed. The binary occupancy decision is solely performed in reconstruction space instead of fusing back-projected silhouettes in image space. Hierarchical data structures are used to reconstruct the objects progressively focusing on boundary regions. Thus, the algorithm can be stopped at any time with a certain conservative level of detail. Most parts of the algorithm may be processed in parallel using GPU programming techniques. The main application domain is the surveillance of real environments like in human/robot coexistence and cooperation scenarios.**

Keywords – Inferred Visual Hull; Multi-View Reconstruction; Occlusions; Visibility; Static Environment; Dynamic Environment; Computer Vision; Color cameras; 3D Scene Analysis; Irregular Space Partitions

I. INTRODUCTION

The multi-view reconstruction in-between a known environment is useful for surveillance tasks such as in the domain of human/robot coexistence and cooperation. Today's safety fences and light barriers in industrial settings become dispensable since humans can be geometrically reconstructed in 3D and added to the robot's environment model. Thus, dangerous collisions can be avoided for example by dynamically limiting the speed of the robot or by performing path re-planning in the vicinity of the human.

Multi-view reconstruction and visual hull [10] algorithms have been investigated for many years with different focus, e.g. creating exact models including coloring [13], handling occlusions [4, 6, 9, 11] or, automatically reconstructing occluding objects [7, 8]. Different representations have been used, e.g. polyhedra [4, 6, 7], voxels [9, 11], or conexels [1, 12]. Several optimizations like octrees applied to voxel spaces [9], m-trees applied to irregular partitioned spaces [1] and parallel processing aspects [9] have been discussed. Occupancy decisions have been carried out in reconstruction space [12]. The use of plausibility checks to reject reconstruction artifacts based on additional information about the objects to reconstruct has been proposed [11].

The main requirements in surveillance tasks as in human/robot coexistence and cooperation are *robustness*, *correctness*, *speed*, and *any-time* ability. *Correctness* requires considering the known environment with its occlusions. *Robustness* of the reconstruction can be achieved by deciding on occupancy in reconstruction space. *Speed* is achieved by optimized data structures and parallel algorithms. *Any-time* ability provides conservative results at any time. Furthermore, a camera-based space partitioning also improves these aspects.

II. STATE-OF-THE-ART

These key aspects are now discussed for some recent state-of-the-art approaches. The following Table (Figure 1) summarizes the result of the comparison.

In [2] a voxel-based approach with an octree-like but dynamical decomposition is used. Due to efficiency reasons, the area a voxel projects to is sampled by eight points. The likelihood that a pixel belongs to foreground or background is measured. The voxel is classified in *background*, *edge*, *foreground* and *unknown* by considering all samples of all views. On demand, the voxel is decomposed. The calculation of the used Mahalanobis distance is described to be parallel processable by SIMD instructions like SSE (Intel's Streaming SIMD Extension), leading to an effective speedup factor of 2

	Decision on occupancy in reconstruction space	Camera-based space partitioning	Parallelizable algorithms	Hierarchical data structures	Considering Occlusions	Plausibility Checks
[2] (Cailette)	+	-	o	+	-	-
[1, 12] (Casas, Salvador)	+	+	-	+	-	-
[9] (Ladikos)	o	-	+	+	o	-
[4] (Franco)	-	+	-	-	o	-
[7] (Guan)	-	+	-	-	o	-
[11] (Kuhn)	-	-	-	-	+	+
This paper	+	+	+	+	+	+

Figure 1: Table, summarizing properties of state-of-the-art approaches.

to 3.

A camera-based space partitioning is proposed by [1, 12]. The camera images are sub-divided into quadrants. On the basis of epipolar geometry and by using the edges of the quadrants of different views, a volume can be identified. This volume can be analyzed by an arbitrary analysis function using the pixels within the according image quadrants of all views [1]. A consistency check determines whether a volume has to be sub-divided. The sub-division is achieved by dividing the image quadrants into four smaller image quadrants. In [12] a reconstruction method based on this approach using probability maps is presented.

In [9] two similar voxel-based methods which are processed in parallel using Nvidia's CUDA are presented. An octree data structure is used. The first approach considers the pixel areas a voxel projects to, while the second approach considers only the projection of the center of a voxel but into a Gaussian image pyramid. Occlusions are considered by means of so-called occlusion masks. Occlusion masks identify pixels in image space where objects can be occluded. These masks are simply added to the detected foreground and thus interpreted as objects before reconstructing the silhouettes. Since pixel areas of all views a voxel projects to are considered for decision in reconstruction space, an 'o' is inserted into Table (Figure 1) although the images are binary.

A polyhedral and thus implicitly camera-based approach is presented by [4]. Occlusions are only considered in terms of *inside* and *outside* the camera viewing cones. Hence, regions seen by only a fraction of all cameras are also correctly reconstructed.

In [6] occlusion masks are introduced and used in conjunction with a polyhedral approach.

An exact occlusion handling is discussed in [11], which proposes a voxel-based approach and mentions the possibility of the use of the conexels approach [1]. In its discussion, objects can be detected in front of an occluding static environment contrary to the concept of occlusion masks. Furthermore, information about the objects to reconstruct can be specified such that reconstruction artifacts can be rejected.

None of the approaches above can be directly used to satisfy all of the requirements of Section 1. The aim of this paper is to present a novel approach, which fulfills all requirements. The reconstruction space is partitioned taking the cameras' pixel grid and the known environment into account (Section 3). The basic algorithm uses occupancy decisions solely in reconstruction space. The improved version uses hierarchical data structures enabling efficient calculations and any-time ability (Section 4). In Section 5, parallel processing aspects of the algorithm are discussed. Finally, Section 6 describes the integration of plausibility checks.

III. Irregular Space Partitioning

The main drawback of using voxel spaces is that a voxel located near a camera usually projects onto many pixels while many voxels located farther away project onto one single pixel. The camera-based space partitioning approach as described in [1] is adapted in different ways. The partitioning

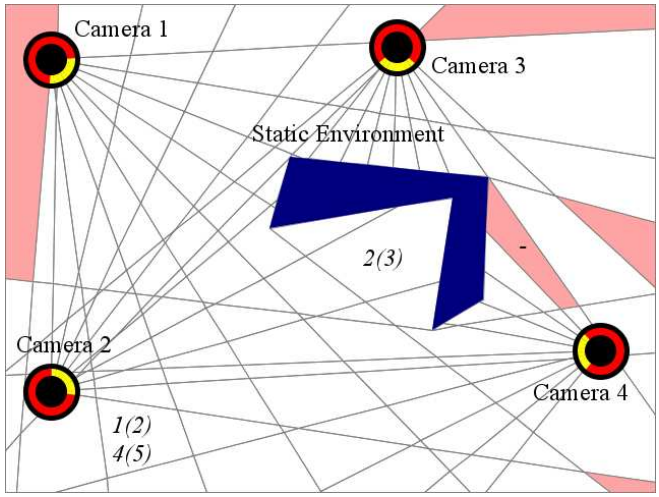

Figure 2: Irregular space partitioning (white and red cells) by four cameras (circles) with eight pixels each and a static known environment (blue). The visibility is exemplified for three partitions: The partition on the left at the bottom is visible to Camera 1 (Pixel 2) and Camera 4 (Pixel 5). The red partitions are not visible to any camera. Yellow in the camera images represents *free*, red *occluded*.

is extended to be suitable to static known environments (e.g. tables, racks, etc.) and also outside the cameras' viewing cones and thus correctly handling occlusions as discussed in [11].

In a static known environment, i.e. static objects $O \subseteq \mathrm{R}^n$, appearances, and static lightning conditions, the values c(i) of all pixels $i \in I$ of all cameras are static apart from noise. A point e of the n-dimensional (typically $n \in \{2,3\}$) reconstruction space $F = \mathrm{R}^n \setminus O$ is visible to a pixel, if a dynamic object located at that point e directly may change the value of the pixel. This excludes indirect changes caused e.g. by shadows or caustics.

The set $P \subseteq F$ of all points e visible to the same set of pixels $V \subseteq I$ of all cameras describes one irregular space partition. The pixel set V is called *visibility* and can be applied to an element $e \in F$ or to a whole partition $P \subseteq F$.

The following equation summarizes the visibility of a point $e \in F$ (Figure 2).

$$V(e) = \{i \in I \mid c(i) \text{ may be changed by a dynamic object at } e\}$$

The irregular space partitioning of the reconstruction space F can be described by the set S of all partitions P.

$$S = \{P \subseteq F \mid \forall a,b \in P, c \notin P : V(a) = V(b) \wedge V(b) \neq V(c)\}$$

An irregular space partition is the union of all points visible to the same set of pixels. Thus, the partitions represent the logical connection between the cameras' pixels.

Note, this irregular space partitioning works using distorted images, thus undistorting images is not necessary neither while creating the partitions nor while performing the reconstruction process online.

IV. Reconstruction

Change detection and background subtraction techniques

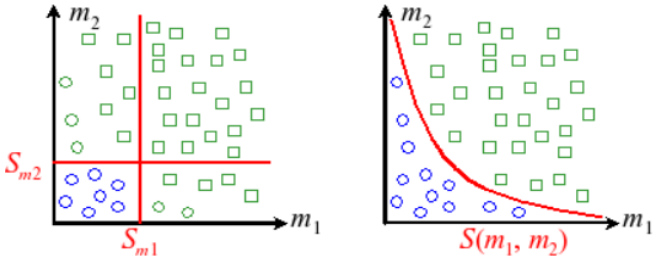

Figure 3: Illustration of combining two binary decisions in image space (left) compared to the decision function combining two probabilities in the reconstruction space (right). The circles should be classified as *blue*, while the rectangles should be classified as *green*. The classification is more flexible and thus more accurate.

are used to determine regions in images which have changed compared to a set of previous image frames or a known background model. Usually, a binary image is created in image space describing foreground and background. Since the reconstruction consists of multiple perspectives onto the same regions in reconstruction space, it is reasonable to decide on foreground and background, i.e. occupancy in reconstruction space based on a kind of probabilities in image space.

The probability that a pixel sees foreground can be back-projected into the reconstruction space. More precisely, all partitions visible to this pixel may contain an object to be reconstructed with the determined probability for that pixel. Taking the probability of several views into account, more accurate decisions about occupancy can be made within the reconstruction space as stated above. The decision function based on the probabilities for each associated pixel can freely be chosen. This enables for example a different weighting of the different views dependent on the according detectability (Figure 3).

Now, a basic reconstruction algorithm can be formulated, working on the pre-calculated irregular space partitions (Figure 4):

```
01 for each partition
02   get associated foreground probabilities
03   decide on occupancy: occupied, free
04   if decided to occupied
05     append partition to the set of occupied partitions
06   end if
07 end for
```

This algorithm reconstructs the objects in-between the known environment on a logical base. All occupied partitions are contained by the *set of occupied partitions* determined by the algorithm. An arbitrary geometrical description can be attached to each partition. Thus, the partition can be represented as polyhedron, spheres or voxels.

Using high camera resolutions result in enormous calculation times, since each partition has to be tested on occupancy. This issue can be solved by using hierarchical data structures that allow stepwise refinement only of the non-homogeneous regions, i.e. object boundary regions in

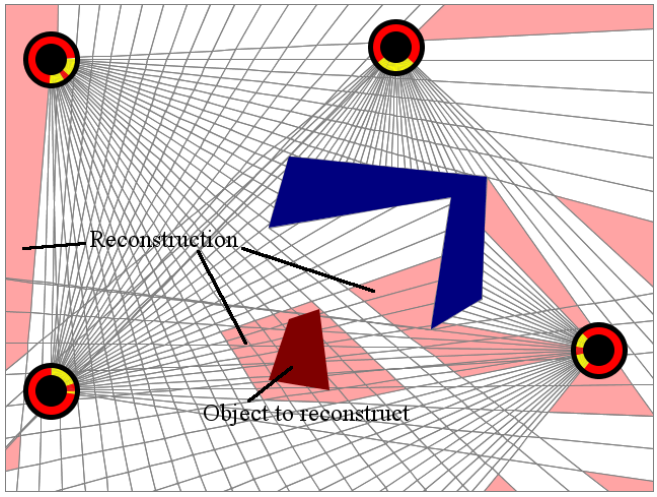

Figure 4: Illustration of the camera-based reconstruction (all light red regions) of an object (dark red region) in-between a known environment (blue) using the simple algorithm which iterates over all partitions. Formally, the reconstruction is the region of the space where an object to reconstruct has to be assumed. Thus, there are several regions contained to the reconstruction.

reconstruction space starting at a coarse partitioning level. Therefore, trees are used for both image and reconstruction space (Figure 5a).

The trees in image space are built by combining regions on different levels (Figure 5a, left). For example quad-trees can be used, combining each four pixel region of a higher resolution into one single pixel in a lower resolution until images with one-pixel-resolutions are obtained. The combination should consider both, a kind of maximum foreground probability and a minimum foreground probability. These two values can be used as homogeneity criterion. A combined region with a high maximum and a low minimum foreground probability signals an inhomogeneity and thus two different types of sub-regions. Of course, further values to describe a region in image space can be attached to each node of the tree enabling a more complex decision function in reconstruction space.

The tree on reconstruction space side (Figure 5a, right) describes the irregular space partitioning as introduced in the previous section, but for each layer in the image tree. Thus, many small partitions in a lower layer of the reconstruction space tree caused by a higher image resolution are combined to a single partition in a higher layer of the reconstruction space tree caused by the lower image resolution. This approach of building up the reconstruction space tree has the advantage that also distorted images can be used since small valid partitions are composed to one single valid partition implicitly including the information about distortion. All nodes caused by the one-pixel-resolution images in the image trees are combined to the root of the reconstruction space tree. Each node contains the visibility of this partition, i.e. the set of pixels in the according layer of the image trees. Furthermore, each node has three states: *occupied*, *free*, *mixed*.

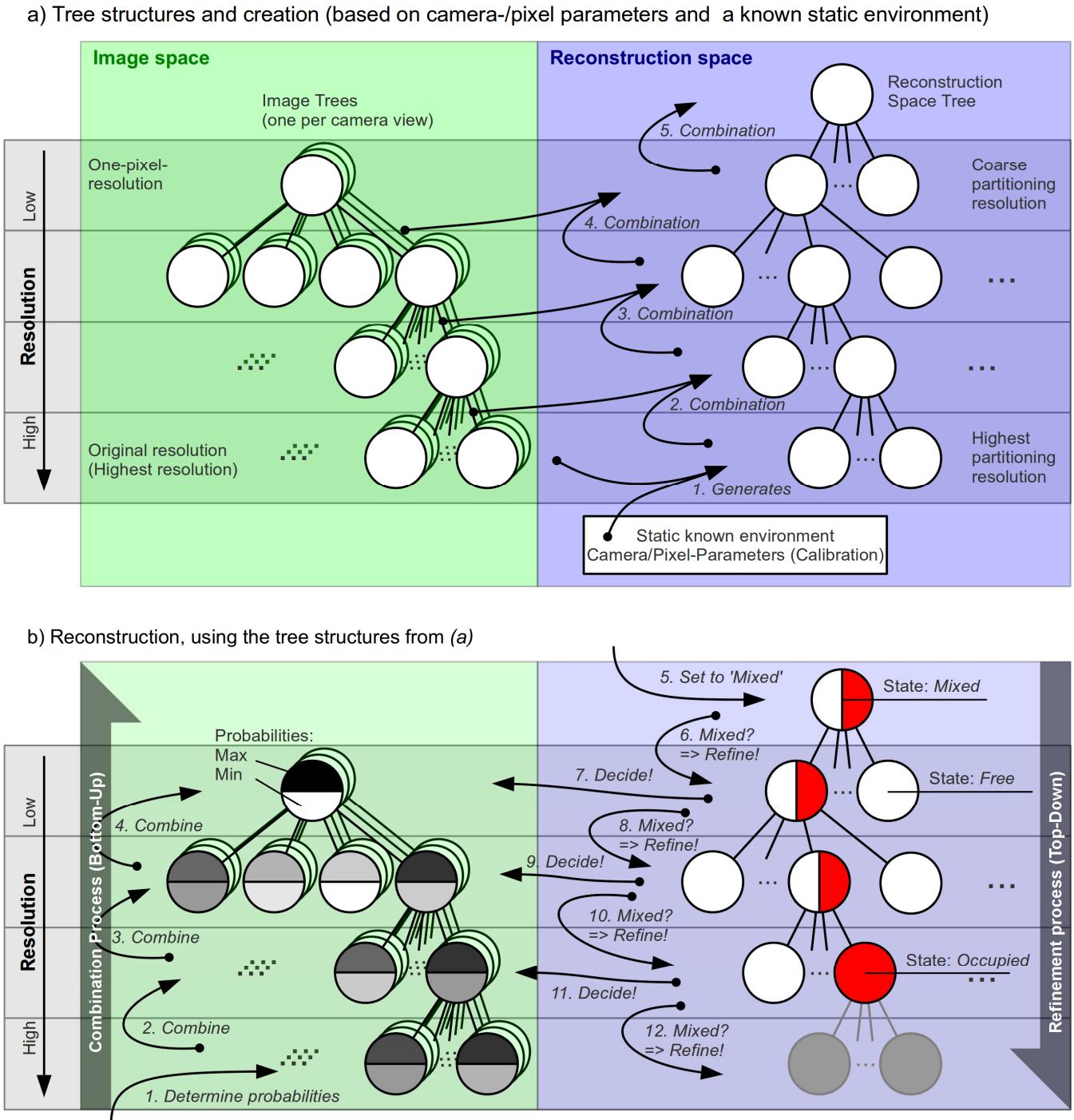

Figure 5: Illustration of the tree structures, their creation, and the interrelation between image and reconstruction space at initialization time (a) and reconstruction time (b). The built trees (a) are used to reconstruct objects in-between a known environment. The tree structures reflect the camera- and environment-based partitioning linked with the associated image pixels. The values and states of the nodes reflect the current state of the observed scene. At image space probabilities are used (horizontal divided circles). In reconstruction space (b) the states *Free*, *Occupied* and *Mixed* are used (red-white circles). The values in image space are calculated bottom-up; the states in reconstruction space are calculated top-down by considering only the mixed nodes.

The structures and the interconnection of these trees can be calculated off-line or are known in advance (cf. quad-trees), since the irregular space partitioning does not change – it directly depends on the static known environment. Only the states of the nodes, describing the current scene must be calculated on-line.

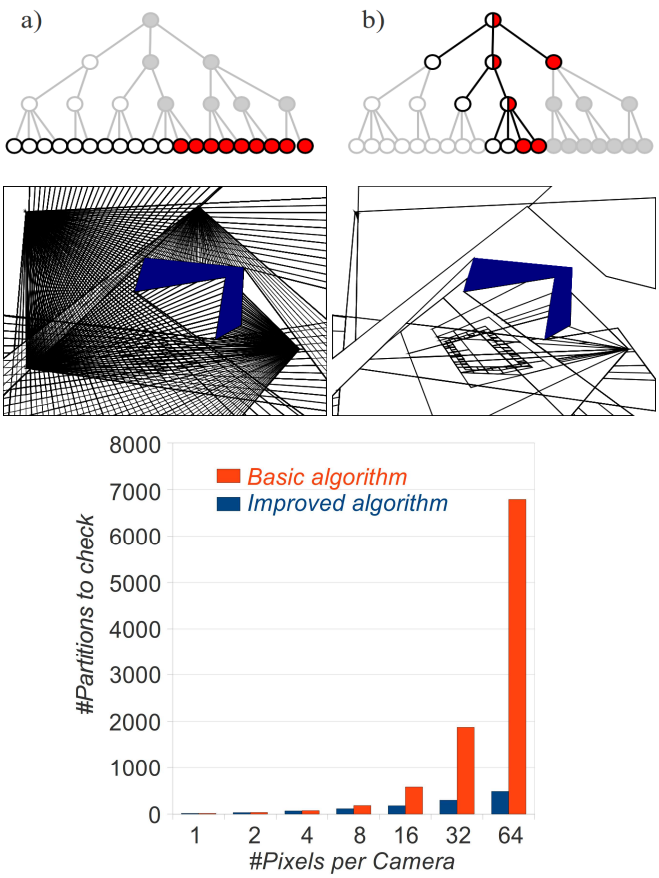

Figure 6: Top and second row: Comparison between the basic algorithm (a) and the improved algorithm (b) using hierarchical data structures. Red circles represent *occupied*, while white circles represent *free*. The basic algorithm has to process all partitions in high resolution, while the efficient algorithm only refines mixed nodes (white-red circles) of the tree. If the efficient algorithm is stopped, a valid layer can be used as reconstruction. Bottom: Number of partitions to check for occupancy dependent on the number of pixels per camera in our example scene (second row).

Thus, the reconstruction process is performed on-line (Figure 5b), and can be roughly described by two steps: Assigning the values attached to the image tree bottom-up (Figure 5b, left) and then assigning the values attached to the reconstruction space tree top-down layer-by-layer, whereby only mixed nodes have to be refined by analyzing it leafs until the highest resolution is reached or the algorithm is stopped (Figure 5b, right):

```
01  calculate image (quad-) trees values
02  use coarsest reconstruction space tree node
03  set reconstruction space tree root to mixed
04  append reconstruction space tree root to the set of
                                          occupied nodes
05  while not at highest reconstruction tree resolution
06    for each node at current layer set to mixed
07      for each child of this node
08        get associated foreground probabilities
09        decide on occupancy: occupied, free, mixed
10        if decided to mixed or occupied
11          append child to the set of occupied nodes
12        end if
13      end for
14      remove current node from the set of occupied nodes
15    end for
16    use next layer
17  end while
```

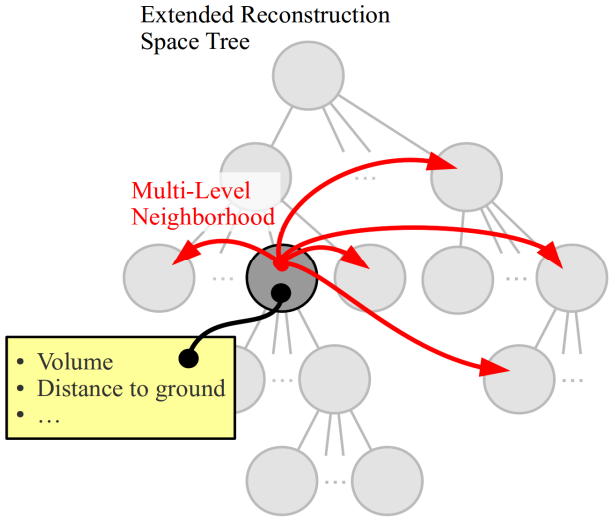

Figure 7: Reconstruction space tree extended by additional information per partition/node to enable plausibility checks: The multi-level neighborhood describes connected partitions over all levels, such that clustering in a partially explored tree is possible. Furthermore, the volume of a partition, its distance to the ground, etc. can be calculated in the initialization step. Thus, reconstruction artifacts can be fast identified and rejected.

All nodes contained by the *set of occupied nodes* represent the logical reconstruction. If the algorithm is interrupted, also mixed nodes are contained by this set. These nodes represent a conservative approximation of the highest reconstruction resolution since all occupied regions and additional free regions are covered by it.

Compared to the basic algorithm, now it is reasonable to attach a geometrical representation to each node, i.e. to each partition in each level-of-detail, since the occupied regions are not necessarily refined. The difference between the basic and the current algorithm is illustrated in Figure 5. While the basic algorithm needs to iterate over all high resolution partitions, the second algorithm only needs to refine mixed nodes. Further, the any-time ability is given, since the algorithm can be interrupted any time.

V. Parallel Processing

Optimization of algorithms does not only concern efficient data structures, but also concerns the ability to process the calculations in parallel. Thus, above algorithms are analyzed with respect to being parallelizable. The main focus lies on the parallelization on the GPU, for example using Nvidia's

CUDA.

The object probability for each pixel in each image can be calculated using Gaussian distributions as background model and comparing it to the current pixel values. Since the calculations are independently among each pixel, it is trivial to parallelize this part of the algorithm. More sophisticated background subtraction and change detection techniques taking regions and illumination changes into account can be modified such that a more accurate probability can be specified for each pixel. Some existing background subtraction approaches are specialized for being performed on the GPU [5].

Building up the quad-tree on image side based on the probabilities can easily be parallelized on the GPU since again the region combination process is completely independent among each region.

The refinement process of the reconstruction is similar parallelizable, but in contrast to the image tree, not the whole tree is explored. Thus at each step, a list of nodes to refine must be extracted, which then can be processed parallel in the succeeding step.

Thus, the whole algorithm is parallelizable up to this step.

VI. Plausibility Checks

Plausibility checks are used to reject artifacts which cannot contain an object, due to the information given about the objects to be reconstructed [11]. The plausibility checks work on whole objects. Thus, a clustering of connected occupied partitions must be performed. This can be done, by storing the neighbourhood of a partition and by using a 3-d flood fill algorithm. Since the tree is not completely refined, it is necessary to store the neighbouring partitions of all levels.

The clustered partitions are checked for plausibility using e.g. minimum volume and/or maximum distance to ground information. For example, if humans should be reconstructed, a certain minimum volume can be assumed and thus all artifacts with a smaller volume can be safely rejected. Furthermore, if the application allows us to assume that a human does not jump higher than one meter within the observed volume, all artifacts with a larger distance than one meter to ground can also be safely removed. In order to perform these plausibility checks fast, volumes and distances for each partition in each level-of-detail can be pre-calculated. The volume A of a cluster C containing a set of partitions $P \subseteq S$ (cf. Section 3), providing its volume via a function $vol(p \in P)$ can be calculated by

$$A = \sum_{p \in P} vol(p)$$

The minimum distance to ground D of this cluster, whereby the distance to ground of a partition can be queried by $dist(p \in P)$ can be calculated by

$$D = \min_{p \in P}(p)$$

Usually, dynamic known objects like robots cause the same 3-d reconstructions as the objects to be reconstructed (e.g. human) [11]. Thus, if the robot speed should by dynamically limited dependent on the distance to the human, the reconstructed hull has to be rejected if possible. But if none of the plausibility checks is able to reject this hull, a zero-distance must be assumed. Another, special plausibility check is very useful for these kinds of objects: The hull thickness check. If a geometrical model is available for both the known dynamic object and the partitions, this thickness can be calculated. If the largest thickness between hull and known dynamic object is smaller than the minimum thickness of a human, the hull can be rejected, too – the human cannot hide within this hull.

Note, the plausibility checks can be applied at each level-of-detail while reconstructing. This enables early rejecting artifacts, i.e. setting partitions to non-occupied and thus speeding up the process in only detailing actual objects.

VII. Conclusions

A multi-view reconstruction algorithm, which unifies the advantages of many different approaches has been presented: It reconstructs 3D objects with pixel accuracy taking a known environment into account; it decides in reconstruction space; it uses efficient data structures which focus on boundary regions; it allows parallel calculations; it rejects reconstructed artifacts; furthermore, undistorted images can be used directly.

Future work may focus on the analysis of suitable change detection methods for describing object probabilities on image side. For example, the method of [3, 5] could be extended. Since [5] uses a mask to estimate the succeeding background model, a projection of the actual reconstruction is possible. Additionally, the *min* and *max* determination for each image region in the image tree can be transformed to a more general *upper*/*lower* bound description, not fixed to the minimum or maximum of probabilities. Finally, appropriate decision functions in reconstruction space may be investigated.

Acknowledgment

This work has been supported by the German Research Foundation (DFG) under the project name “Sicherheitsstrategien für die Mensch/Roboter-Koexistenz und -Kooperation” (SIMERO).

References

[1] J. Casas and J. Salvador, “Image-Based Multi-view Scene Analysis using 'Conexels'”, ACM Proceedings of the HCSNet workshop on use of vision in human-computer interaction, Vol. 56, Canberra, Australia, 2006.

[2] F. Cailette and T. Howard, “Real-Time Markerless Human Body Tracking with Multi-View 3-D Voxel Reconstruction”, In Proc. British Machine Vision Conference (BMVC), pp.. 597-606, Oxford, 2004.

[3] A. Elgammal, D. Harwood and L. Davis, “Non-parametric Model for Background Subtraction”, Computer Vision – ECCV, Vol. 1843, 2000.

[4] J.-S. Franco and E. Boyer, “Efficient polyhedral modeling from silhouettes”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 31, Is. 3, pp. 414-427, 2009.

[5] S. Fukui, Y. Iwahori and R.-J. Woodham, “GPU Based Extraction of Moving Objects without Shadows under Intensity Changes”, IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Hong Kong, 2008.

[6] L. Guan et al. “Visual Hull Construction in the Presence of Partial Occlusions”, In Proc, of the 3rd International Symposium on 3D Data, 2006.

[7] L. Guan, J.-S. Franco and M. Pollefeys, “3D Occlusion Inference from Silhouette Cues”, In Proc. Comp. Vis. And Patttern Rec. (CVPR), 2007

[8] M. Keck and J.W. Davis, “3D Occlusion Recovery using Few Cameras”, In: Conf. On Computer Vision and Pattern Rec. (CVPR), 2007

[9] A. Ladikos, S. Benhimane and Nassir Navab, “Efficient Visual Hull Computation for Real-Time 3D Reconstruction using CUDA”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Anchorage, Alaska (USA), June 2008.

[10] A. Laurentini “The Visual Hull Concept for Silhouette-Based Image Understanding”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, pp. 150-164, 1994.

[11] S. Kuhn, D. Henrich, “Multi-View Reconstruction of Unknown Objects within a Known Environment”, In: Proc International Symposium on Visual Computing (ISVC), Las Vegas, 2009.

[12] J. Salvador, J. Casas, “Shape from Probability Maps with Image-Adapted Voxelization”, In: Workshop on Multi-camera and Multi-modal Sensor Fusion-Algorithms and Applications (M2SFA2), 2008, Marseille, France

[13] S.M. Seitz et al, “A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms”, In: Proc. of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) – Vol 1, pp 519-528, 2006.

[14] T. Svoboda, D. Martinec and Tomas Pajdla, “A Convenient Mullti-Camera Self-Calibration for Virtual Environments”, In: Presence: Teleoperators and Virtual Environments, Vol 14, Issue 4, 2005.