In: 5th International Symposium on Visual Computing, Las Vegas, Nevada, USA, Nov 30 – Dec 2, 2009
G. Bebis et al. (Eds.): ISVC 2009, Part I, LNCS 5875, pp. 784-795, Springer-Verlag
The original publication is available at www.springerlink.com.

Multi-View Reconstruction of Unknown Objects within a Known Environment

Stefan KUHN and Dominik HENRICH

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

Abstract. We present a general vision-based method for reconstructing multiple unknown objects (e.g. humans) within a known environment (e.g. tables, racks, robots) which usually has occlusions. These occlusions have to be explicitly considered since parts of the unknown objects might be hidden in some or even all camera views. In order to avoid cluttered reconstructions, plausibility checks are used to eliminate reconstruction artifacts which actually do not contain any unknown object. One application is a supervision/surveillance system for safe human/robot-coexistence and –cooperation. Experiments for a voxel-based implementation are given.

1 Introduction

Geometrical information about objects is required in many applications. In several cases this information is *known*. For example, most industrial robots act in a geometrically completely known environment. It is indispensable, to guarantee the correctness of this information anytime, in order to avoid collisions. Therefore, fences and safety light barriers are set up to guarantee this correctness and to stop the robot in an unexpected situation. In other cases, the geometrical information is *not known* in advance. Thus, vision sensors can be used, to reconstruct this information. In the example, it would become possible, that a human can walk through the robots workspace, since its geometrical information is reconstructed and included in the robot's environment model, such that even in this case no collision occurs.

The reconstruction of objects based on its silhouettes in multiple cameras is known as *surface from silhouette* or *inferred visual hull* ([12], [15]). Many volume-based ([3], [11], [16]) and surface-based ([5], [13]) approaches have been investigated in the past. Most of them have the assumption that the object(s) to reconstruct resides within the common volume which is seen by all cameras. Furthermore, almost all approaches have the assumptions that the object(s) to reconstruct is not occluded by static obstacles – like tables, racks or the robot itself in the example. It may result in an incomplete reconstruction of a human, since conventional background subtraction methods ([4], [6]) are not able to generate the silhouette of the occluded parts of the human. Thus, it is necessary to explicitly consider these occlusions.

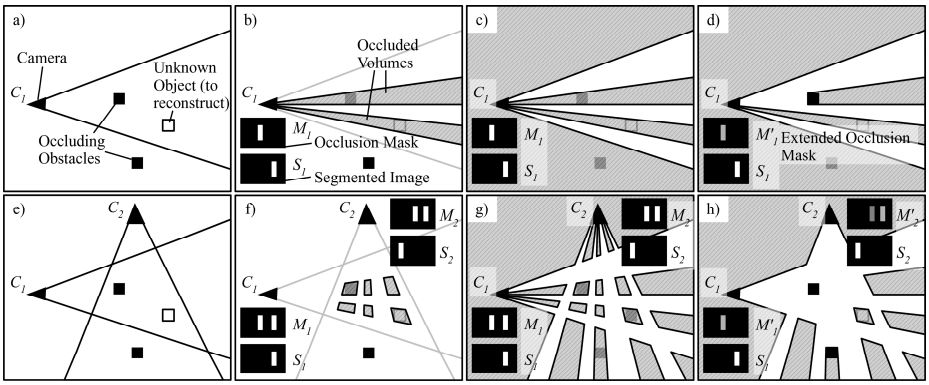

Fig. 1. Illustration of the visibility for a setup with one camera C_1 (first row, a-d) and a setup with multiple cameras C_i (second row, e-h) using different approaches for treating occlusions (b-d and f-h) resulting from the known environment (a, c).

Recently, two concepts to overcome these limitations have been investigated. First, occlusions can be modeled by *occlusion masks* M_i which are binary masks in image space marking regions where occlusions can occur. These marked regions are simply added to the segmented regions S_i detected by the background subtraction method when calculating the inferred visual hull (Fig. 1 b, f). In [2] and [10] the concept of occlusion masks has been used for *masking* dynamic occluding objects (i.e. robots) in order to avoid future collisions with objects, while [7], [8] and [9] automatically *generate* these occlusion masks for static occluding objects by observing active objects in a scene. Unfortunately, using occlusion masks causes more occlusions in the reconstruction than necessary since the volume between the camera and the occluding object is interpreted as occlusion as well even though a background subtraction method could detect objects in front of this occluding object. Second, objects which reside outside the common observed volume can be correctly integrated in the inferred visual hull, if the complement of the fused back-projected free space is calculated [5]. In Fig. 1 c, g this concept is applied in conjunction with the previously described occlusion masks. Note, the object residing outside the common observed volume is now included. But still the reconstruction contains unnecessary occlusions.

Thus, we propose a new approach (Fig.1 d, h), for treating occlusions in a general and more accurate way by additionally utilizing the geometrical information of the known environment (Section 2). Both concepts described above are contained by our approach. Furthermore, the information about how far a pixel can see up to the first occluding object is included (M'_i), resulting in more accurate reconstructions. In most cases, it results in cluttered reconstructions due to the occlusions. Therefore, we propose using plausibility checks, which revise reconstruction artifacts that do not contain an object. Furthermore, a voxel-based algorithm of our approach is provided in Section 3. The memory consumption of look-up tables, which are used for optimization purpose, is analyzed. Experimental results are discussed in Section 4. The paper concludes with Section 5.

2 Occlusions and Visibility

This section comprises a theoretical look at the problem of occlusions and visibility in a multi-camera setup using plausibility checks to eliminate pseudo objects, i.e. reconstruction artifacts which actually do not contain an object.

In the first subsection, the types of objects that are contained by surveyed scenes are described. In the following subsection, visibility and occlusions for a single camera are considered, taking the types of objects into account. Thereafter, the simultaneous use of several cameras with different perspectives onto the surveyed scene is described. The last subsection discusses how reconstruction artifacts can be revised, using plausibility checks.

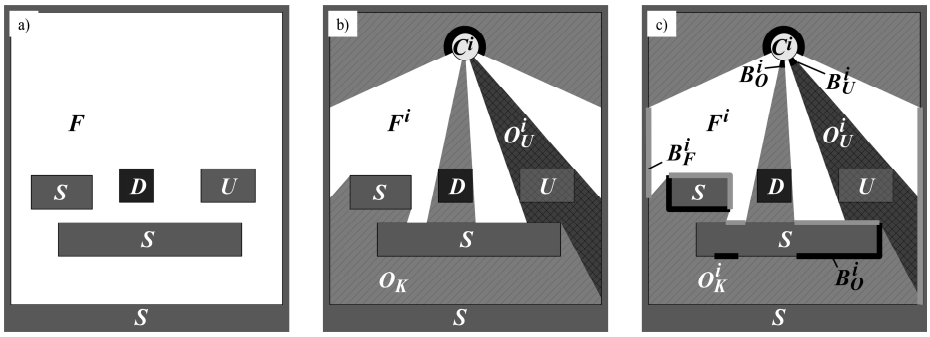

Fig. 2. Illustration of the visibility and the occlusions using color or grayscale cameras with common background subtraction methods. Nomenclature: C^i: Camera i; F^i: Free in camera i; S: Static known object; D: Dynamic known object; U: Unknown object; O^i_K: Known occlusion in i; O^i_U: Unknown occlusion in i; B^i_F: Free boundary in i; B^i_O: Occlusion boundary in i; B^i_U: Unknown boundary in i.

2.1 Object Types

The surveyed scene contains static and dynamic objects. *Static* objects S are racks, tables etc. The geometry, position and the appearance of those objects are *known* and do not change over the time (apart from possibly occurring shadows and from illumination changes caused by the dynamic objects). *Dynamic* objects are robots, conveyor belts, humans etc. This group must be divided into two subgroups. The first subgroup contains *known* dynamic objects D, with changing geometry, position and the appearance but in a known manner. The robots and conveyor belts pertain to this subgroup. The second subgroup contains dynamic objects U with *unknown* changing geometry, position and appearance, e.g. humans. In the majority of cases approximate information about size, volume or similar can be provided. Note that static unknown objects do not exist. The *free space* F of a surveyed scene does not contain any known object but may contain unknown objects. Fig. 2 illustrates the introduced object types. In summary, the following three equations hold true:

$$S \cup D \cup F = \mathbf{E}^n, \text{ with } \mathbf{E}^n\text{: Euclidian Space} \quad (1)$$

$$S \cap D = S \cap F = D \cap F = \varnothing \text{ and } U \subseteq F \quad (2)$$

2.2 Single Camera

A number of N calibrated cameras with focal points $C^i \in \mathbf{E}^n$, $i \in \{1, \ldots, N\}$ and a frustum $L^i = \{x \in \mathbf{E}^n | \, x \text{ is projected via } C^i \text{ onto the image plane of camera } i\}$, are used to detect and finally reconstruct the unknown objects which reside in-between the known objects as accurate as possible. Here, we only consider color and grayscale cameras, but the approach can easily be extended to depth cameras.

One fundamental characteristic of these vision sensors is that they can only see up to the surface of the nearest opaque object per viewing direction (e.g. pixel center direction). Thus, occlusions always occur at the rear side of an opaque object. Moreover, the visibility of these sensors is limited by the frustum L^i. Outside this frustum, the sensor is not able to see anything. Thus, these parts can be interpreted as occlusions as well.

Now, the terms *visibility* and *occlusions* have to be introduced and detailed (Fig. 2). The *visibility* V^i of a camera i is the region of the free space F where a camera is able to detect unknown objects. The *known occlusion* $O_K{}^i$ of a camera i is the region of the free space F where a camera can not detect unknown objects due to occlusions caused by known objects. Thus, it can be stated that $O_K{}^i \cap V^i = \varnothing$ and $O_K{}^i \cup V^i = F$ for each camera i. The *unknown occlusion* $O_U{}^i$ of a camera i is the region of the visible space V^i where an unknown object has to be assumed, due to the evaluation of a camera image by a background subtraction method. The free space seen by camera i F^i is defined by $F^i = V^i \backslash O_U{}^i$. It can be stated, that $O_K{}^i \cap O_U{}^i = O_K{}^i \cap F^i = O_U{}^i \cap F^i = \varnothing$ and $O_K{}^i \cup O_U{}^i \cup F^i = F$ for each camera i. The concrete structures of the sets V^i, $O_K{}^i$, $O_U{}^i$ depend directly on the used camera type with its detection capabilities.

In order to describe our formalism for a specific camera type (here color-/grayscale cameras), we use the concepts of rays and segments in the Euclidean Space. A ray(S, E) and a segm(S, E) are point sets and are defined by

$$\text{ray}(S,E) := \{S + t \cdot (E - S) \mid t \in \mathbf{R}_0^+, S, E \in \mathbf{E}^n\} \quad (3)$$

$$\text{segm}(S,E) := \{S + t \cdot (E - S) \mid t \in \mathbf{R} \wedge 0 \le t \le 1, \;\; S, E \in \mathbf{E}^n\} \quad (4)$$

Furthermore, a distance function dist(P, Q), with P, $Q \in \mathbf{E}^n$ exists, since the Euclidean Space is a metric space. Having the visibility and the occlusions, sets containing the most distant visible points and the nearest occluded points (per viewing direction) can be specified by

$$B_F^i = \{x \in V^i \mid \text{dist}(C^i, x) = \max_{y \in V^i \cap \text{ray}(C^i, x)} \{\text{dist}(C^i, y)\}\} \quad (5)$$

$$B_O^i = \{x \in O_K^i \mid \text{dist}(C^i, x) = \min_{y \in O_K^i \cap \text{ray}(C^i, x)} \{\text{dist}(C^i, y)\} \quad (6)$$

$$B_U^i = \{x \in O_U^i \mid \text{dist}(C^i, x) = \min_{y \in O_U^i \cap \text{ray}(C^i, x)} \{\text{dist}(C^i, y)\} \quad (7)$$

Using color or grayscale cameras, conventional background subtraction methods can be utilized. These methods segment an image into foreground and background based on the known appearance and a current image of the surveyed scene. If an unknown object resides in the scene and is not occluded by the known environment, it is marked as foreground in the segmented image. But usually the dynamic known objects are also – if not occluded – identified as foreground in the segmented image. Thus, the detection of unknown objects *in front* of a dynamic known object is not possible. Since the change detection method is not able to decide whether the cone between the camera and the dynamic known object is free or contains unknown objects, it must be interpreted as known occlusion. Thus, the visibility is described by

$$\begin{aligned} V^i = \{x \in L^i \mid & \text{segm}(C^i, x) \subseteq F \wedge \\ & (\text{ray}(C^i, x) \cap D = \emptyset \vee \\ & (\text{ray}(C^i, x) \cap D \neq \emptyset \wedge \text{ray}(C^i, x) \cap S \neq \emptyset \wedge \\ & \exists_{y \in S \cap \text{ray}(C^i, x)} \; \forall_{z \in D \cap \text{ray}(C^i, x)} \text{dist}(C^i, y) < \text{dist}(C^i, z)))\} \end{aligned} \quad (8)$$

As detailed above, the known occlusions are formulated by $O_K^i = F \setminus V^i$. Since depth values are not available for unknown objects with this sensor type, unknown occlusions start at the camera:

$$O_U^i = \{x \in V^i \mid \text{ray}(C^i, x) \cap U \cap V^i \neq \emptyset\} \quad (9)$$

2.3 Simultaneous Use of Several Cameras

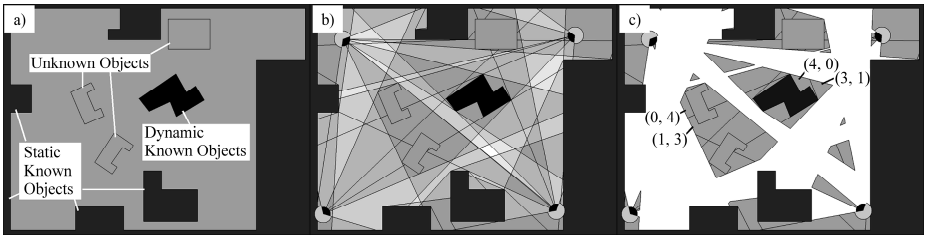

Fig. 3. Illustration of combining several camera views. (a) setup; (b) partitioning of the space; (c) Occluded parts with occlusion-tuples.

Using several cameras with different perspectives onto the surveyed scene, each camera that is used provides a different occlusion and visibility situation, as discovered in the previous section that now has to be merged.

For every camera *i*, a collection of sets can be provided describing the known and unknown occlusions as well as the surveyed free space by $Q^i = \{O_K^i, O_U^i, F^i\}$

The partitioning of the free space via the reconstruction step can be described by

$$R = \{A = q^1 \cap ... \cap q^N \mid q^i \in Q^i \wedge \\ \forall x, y \in A : \exists f : [0,1] \to A, f(0) = x \wedge f(1) = y\} \quad (10)$$

In words, all different labeled regions of all cameras are intersected among each other. Furthermore, the resulting intersections are grouped into connected components. All these connected components are contained by *R* (Fig. 3 b). Again, it can be stated that $\cup_{A \in R} A = F$.

Volumes in the free space of the surveyed scene which are actually seen as free (cf. F^i) by at least one camera are not further considered, since no unknown object can reside there. This results in (Fig. 3 c):

$$R' = \{x \in R \mid \forall i = \{0,...,N\} : x \notin F^i \wedge x \neq \emptyset\} \quad (11)$$

To each set of *R'* a tuple (*o*, *u*) can be assigned, containing the number of seen known occlusions and unknown occlusions (Fig. 3 c).

2.4 Plausibility Checks

In the majority of applications some information like size, volume, etc. about the unknown objects is available. Several sets of *R'* actually cannot contain an unknown object. Thus, plausibility checks are used to eliminate those occlusions which do not contain unknown objects. The mentioned plausibility checks aim for a *quasi-static* consideration. Another kind of plausibility checks can utilize *temporal* considerations, like "an unknown object can not suddenly appear in and surrounded by free space".

Note plausibility checks can apply to the whole scene or only to a part of the scene. In the following, a couple of quasi-static plausibility checks are discussed.

Minimum Volume: If only unknown objects like humans with a typical volume of 0.075 m³ should be detected, one might set a maximum volume for occlusions to be eliminated to 0.05 m³. Thus, all connected sets of *R'* obtained as described in the previous section with a volume smaller than 0.05 m³ can be safely removed. Only unknown objects with a specified minimum volume remain.

Maximum Distance to Ground: Typically, objects do not hover but have contact with the ground. If this can be guaranteed, all connected sets of *R'* with no contact to the ground *S* or *D* can be eliminated. More general, all objects with a distance larger than a specified maximum distance to the ground can be eliminated. Thus, setting the maximum distance to 1 m also a jumping human can be detected and is not removed by this plausibility check.

Surveillance Zones: In most cases, only certain parts of the whole surveyed scene are actually interesting so that unknown objects outside this part can be eliminated.

Occlusion parameter θ: If it can be guaranteed, that an unknown object can be completely occluded by the known environment in a maximum number of θ cameras, all regions where more than θ cameras see a known occlusion can be eliminated if no

other region is connected to it with equal or less than θ cameras which see a known occlusion, since no unknown object can reside within this region. For more details about θ, see [10].

3 Reconstruction Algorithm

In this section we provide a voxel-based reconstruction algorithm, which works on the surfaces of the objects and is capable to deal with occlusions. Regarding the camera model, we only assume a two-dimensional field of connected pixels, with their back projected volumes also connected. Furthermore, the position and geometry of the pixels and the back projected volumes have to be known. Thus, we assume neither a pinhole camera model nor undistorted images.

3.1 Surface Voxel Determination

Given several calibrated cameras and images segmented into *free*, *known* and *unknown* and a voxel space, surface voxels can be determined by the following algorithm. Then the result is a voxel space with voxels marked according to the occlusions of all perspectives and a list containing all these voxels. At first, the needed functions are explained.

The classification value (*free*, *known* or *unknown*) of a pixel P is provided by the function *classification(P)*. Assuming that two adjacent pixels are separated by a pixel edge E, a list of voxels that are intersected by the back projection of this pixel edge E down to its visibility depth is provided by the function *voxelList(E)*. The function *neighborClassification(E)* for a pixel edge E provides the value *unknown*, if one of the two pixels is classified as *unknown*. It provides *known*, if one pixel is classified as *known* and the other as *free*. In all other cases, it provides *free*. For each voxel, the pixels it projects to in all cameras are needed. For simplification, here we only use the center of the voxels with the consequence, that objects that are smaller than the half of the voxel diagonal may be reconstructed incorrectly. Thus, it is necessary to choose an appropriate small voxel size. (Another voxel-like but camera centric-representation called conexels [1] could be applied, which avoids this drawback). The pixel of the projection of the voxel center V into a camera image C is provided by the function *projectVoxelCenter(V, C)*. The distance for a voxel center V to a camera C is provided by the function *distance(V, C)*. Per pixel P, the visibility depth (distance to B^i_F) and the occlusion depth (distance to B^i_O) as described in Section 2.2 is provided by the functions *visibleDepth(P)* and *occlusionDepth(P)*. The function *markVoxelAndAddToList(V, O_K, O_U)* marks the voxel V in voxel space by the two counter variables O_K, O_U representing the number of known and unknown occlusions respectively, and adds it to a list containing all surface voxels.

```
foreach camera C do
  foreach silhouette pixel edge E do
    foreach voxel V in voxelList(E) do
      counter O_U = 0, F = 0, O_K = 0
      if neighborClassification(E) == unknown
```

```
        O_U = 1
      else if neighborClassification(E) == known
        O_K = 1
      endif
      foreach camera C' != C do
        pixel P = projectVoxelCenter(V, C')
        if classification(P) == known
           or distance(V, C') ≥ occlusionDepth(P)
          O_K++
        else if classification(P) == unknown
           and distance(V, C') ≤ visibleDepth(P)
          O_U++
        else
          F++
        endif
      done
      if F == 0
        markVoxelAndAddToList(V, O_K, O_U)
      endif
    done
  done
done
```

In summary, each camera provides lists of potential surface voxels due to the segmentation. These voxels are sequentially tested in all other cameras. If no camera marks a voxel as free, it actually is a surface voxel. All actual surface voxels are stored in a list and marked in voxel space by the tuple (O_K, O_U).

Since the surface is not necessarily closed at the known objects, one may use a constrained flood fill algorithm to close it. Furthermore, completely occluded regions exclusively caused by the static environment are not revealed by this algorithm but can be determined in an initialization step by testing each voxel for visibility against the static environment in all cameras.

Having the surface voxels, partitions of related voxels, i.e. voxels with the same known and unknown occlusion counter can be built. Then, the sorted plausibility checks can be applied according to the costs and success probability. Dependent on the plausibility check additional information like volume of a partition, has to be calculated.

Besides pixel discretization, the accuracy of the voxel based algorithm depends on the voxel size and can be described by $e = \pm v/2 \cdot \sqrt{3}$, with v length of a voxel edge, while the quality of the reconstruction depends on the scene and camera positions, i.e. the visibility and the occlusions.

3.2 Memory Consumption

Some of the used functions can be implemented as look-up tables to enable fast calculations. In order to give a memory consumption estimation M of these look-up tables, the following variables are introduced: A voxel space with dimensions X, Y and Z is used and the resolution of N cameras is provided by W and H.

The visibility depth and occlusion depth per pixel has a memory consumption for all images of:

$$M_1 = 2 \cdot N \cdot W \cdot H \tag{12}$$

The memory consumption for the voxel lists per pixel edges and for all cameras can be estimated by:

$$M_2 \leq N \cdot [((H + W + 2)) \cdot E \ + ((W + 1) \cdot (H - 1) + (W - 1) \cdot (H + 1)) \ \cdot G \ ], \tag{13}$$
$$\text{with } G = X + Y + Z \text{ and } E = Z \cdot Y + Z \cdot X, \ X \leq Y \leq Z$$

Furthermore, the distances for each voxel to all cameras results in a memory consumption of:

$$M_3 = 2 \cdot N \cdot X \cdot Y \cdot Z \tag{14}$$

Thus, the overall memory consumption is bounded by $M \leq M_1 + M_2 + M_3$. As an example the parameters are set to $N = 4$, $W = 320$, $H = 240$ and $X = Y = Z = 100$, with a typical camera placement and voxel-, pixel-addresses and floating point variables of 4 bytes, results in an upper bound of $M \leq 907$ MB and actually of 411 MB.

4 Experiments

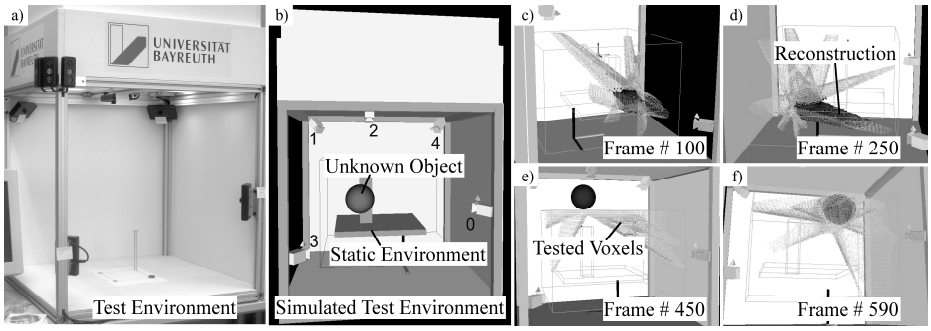

Fig. 4. Our test environment (a), the simulated one (b) and four frames of the experiment (c-f).

In order to evaluate our methods and algorithms, we set up a test environment (Fig. 4 a), with five color cameras mounted around the scene to survey. It is available in a virtual simulation environment, too (Fig 4. b).

4.1 Hardware and Software Configuration

The computer contains an Intel Core™2 Quad CPU, with 2.6 GHz, 6 MB Cache and 4 GB RAM, but currently only one core of the CPU is utilized by our implementation. The graphics card is an NVIDIA GeForce 9600 GT with 512 MB and it is CUDA enabled. The operating system is a SUSE 11.0, with the gcc/g++ compiler suite version 4.3.1. The cubical volume of the test environment is 76 cm × 76 cm × 76 cm. Five Unibrain FireWire Fire-i™ Digital Board Cameras with 15 and 30 fps and a

resolution of 640x480 Bayer Pattern are used. The calibration results, obtained by [14] for the images with a resolution of 640x480 have a low 3D position projection error (mean deviation < 1.6 pixels and standard deviation < 1 pixel).

The following performance tests for the reconstruction uses the virtual test environment, based on the real test environment providing a virtual object (here: sphere with a radius of 6 cm) and its segmented camera images. Additionally, a surveillance zone and a static object are included (Fig 4. b).

4.2 Performance tests

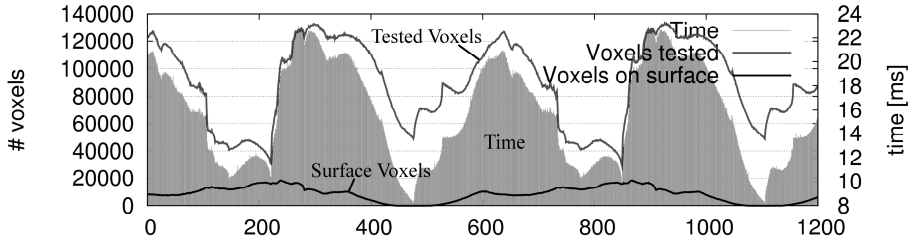

Fig. 5. Diagram of the computation times (gray area) for reconstructing the sphere using five cameras with a resolution of 320x240 and a voxel space of 152x152x138 voxels. Additionally, the number of tested voxels is described by the upper curve and the voxels that actually lie on the surface are described by the lower curve.

The unknown object in the virtual test environment is moved on a circular path around and through the static known object in the middle of the scene. The virtual object is projected into all camera images simulating a conventional background subtraction. The segmented images are used to reconstruct the unknown object within the predefined surveillance zone and in consideration of the occlusions. Two cycles of this movement with a total of 1200 frames have been recorded. Fig. 4 c-f illustrates four interesting frames of the recorded sequence. The white dots represent voxels which have been tested for being surface voxels. The resulting surface voxels are shown containing the visibility tuples.

Dependent on the position of the unknown object, different numbers of pixels and voxels are marked and thus, different computation times are needed. The diagram in Fig. 5 shows the computation time for reconstructing the unknown object. Further it shows the number of tested voxels and the number of voxels that actually lie on the surface. Obviously, the calculation time corresponds to the number of potential surface voxels which have to be tested for each camera. Furthermore, the number of actual surface voxels must always be smaller or equal to the number of potential surface voxels. The calculation time is high, if the unknown object is seen by all cameras, such that many potential voxels have to be tested (frame# 250). Although the unknown object may be outside the surveillance zone, potential surface voxels have to be tested because of the absent depth information of this unknown object. Only the number of actual surface voxels is zero (frame# 450). In Fig 4. d the lower part of the sphere is only seen by the rightmost camera. Thus, the complete cone of potential surface voxels within the surveillance zone caused by that camera actually

results in surface voxels. In this case, the ratio between actual surface voxels and potential surface voxels is relatively high.

Table 2. summarizes the measured computation times by comparing the average values of different configuration pairs. A and B use a camera resolution of 320x240, a voxel space resolution of 152x152x138 and two different number of cameras – 3 and 5. C and D use four cameras, a voxel space of 152x152x138 and two different camera resolutions – 160x120 and 320x240. E and F use four cameras, a camera resolution of 320x240 and two different voxel space resolutions of 76x76x69 and 152x152x138.

The quintessence of this table is that although multiplying the number of pixels or voxels by a factor, the average time increases slower. This behavior is due to the consideration of surfaces and silhouettes instead of volumes and areas, respectively.

Table 2. Comparison of different configuration pairs for reconstructing the sphere.

	# cameras	**Avg. number of potential surface voxel tested**	**Avg. number of actual surface voxel tested**	**Avg. time [ms]**
B	5	89021	8281	15.74
A	3	52488	9168	9.88
B/A	1.667	1.696	0.903	1.593
	# pixel			
D	172800	99095	11536	19.06
C	19200	51821	6166	9.01
D/C	9	1.91	1.87	2.11
	# voxel			
F	10760688	134968	15717	24.43
E	398544	31175	3674	5.9
F/E	27	4.33	4.28	4.14

5 Conclusions

For the first time, a general and consistent formalism for describing the visibility and occlusions within a camera surveyed scene with a known environment is provided. To do so, objects are classified as known/unknown and static/dynamic. A voxel-based algorithm constructing the visual hull, which works on surfaces using grayscale/color cameras in combination with a conventional background subtraction method, has been presented. The experimental results show that the computation time for the reconstruction step depends mainly on the number of tested surface voxels. Additionally, the measurements show that the computation time increases slower than the camera resolution and voxel space resolution, due to the surface and silhouette consideration.

In the future, the plausibility checks especially the temporal ones will be considered more intensively, since these promises a valuable enhancement in the reconstruction of unknown objects. The plausibility checks will be integrated into the voxel-based algorithm. Furthermore, the presented algorithm can be parallelized, such

that potential surface voxels are tested simultaneously. For this, NVIDIAs CUDA seems to be suited. In addition, non-voxel-based approaches will be investigated.

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. D. Ebert and D. Henrich, "Safe Human-Robot-Cooperation: Image-based Collision Detection for Industrial Robots", In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Lausanne, 2002.
3. 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.
4. A. Elgammal, D. Harwood and L. Davis, "Non-parametric Model for Background Subtraction", Computer Vision – ECCV, Vol. 1843, 2000.
5. J.-S. Franco and E. Boyer, "Efficient polyhedral modeling from silhouettes", IEEE Transact. on Pattern Analysis and Machine Intelligence, Vol. 31, Is. 3, pp. 414-427, 2009.
6. 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.
7. L. Guan et al. "Visual Hull Construction in the Presence of Partial Occlusions", In Proc, of the 3rd International Symposium on 3D Data, 2006.
8. L. Guan, J.-S. Franco and M. Pollefeys, "3D Occlusion Inference from Silhouette Cues", In Proc. Comp. Vis. And Pattern Rec. (CVPR), 2007.
9. M. Keck and J.W. Davis, "3D Occlusion Recovery using Few Cameras", In: Conf. on Computer Vision and Pattern Recognition (CVPR), 2008.
10. S. Kuhn, T. Gecks and D. Henrich, "Velocity control for safe robot guidance based on fused vision and force/torque data", In: IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems", Heidelberg, Germany, 2006.
11. 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.
12. A. Laurentini, "The Visual Hull Concept for Silhouette-Based Image Understanding" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 16, pp. 150-162, 1994.
13. S. Lazebnik, Y. Furukawa and J. Ponce, "Projective Visual Hulls", International Journal of Computer Vision, Vol. 74, Nr. 2, August 2007.
14. T. Svoboda, D. Martinec and Tomas Pajdla, "A Convenient Multi-Camera Self-Calibration for Virtual Environments", In: Presence: Teleoperators and Virtual Environments, Vol 14, Issue. 4, 2005.
15. G. Slabaugh, B. Culbertson, T. Malzbender, and R. Shafer. "A survey of methods for volumetric scene reconstruction from photographs." In Intl. WS on Volume Graphics, 2001.
16. R. Szeliski, "Rapid Octree Construction from Image Sequences", CVGIP: Image Understanding, 58(1):23-32, 1993.