Fast multi-camera reconstruction and surveillance with human tracking and optimized camera configurations

Antje Ober-Gecks[1], Computer Science III, University of Bayreuth, antje.ober-gecks@uni-bayreuth.de, Germany
Maria Hänel, Computer Science III, University of Bayreuth, maria.haenel@uni-bayreuth.de, Germany
Tobias Werner, Computer Science III, University of Bayreuth, tobias.werner@uni-bayreuth.de, Germany
Prof. Dr. D. Henrich, Computer Science III, University of Bayreuth, dominik.henrich@uni-bayreuth.de, Germany

Human-robot cooperation and coexistence requires robust surveillance. Our system creates a visual hull as volume approximation of humans within the work cell from images of multiple static color cameras. We present a fast, exact, and conservative reconstruction from high-resolution input data and apply temporal filtering after the reconstruction to address detection failure of the cameras. Furthermore, we avoid incorrect placement of cameras by automatically optimizing the camera positions. All processing steps handle occluding obstacles.

1 Introduction

In the long term, the SIMERO[2] project shall establish strategies which allow robust human-robot cooperation and coexistence for combining the endurance and precision of autonomous robots with the error tolerance and adaptivity of human beings [9] [12] [16] [17].

Our main component is a monitoring system, consisting of multiple static calibrated color cameras, that serves as protection device for humans in a given environment, e.g. the robot work cell. In the following, this surveillance system will briefly be described. The parts of the environment that are targeted by the system, e.g. humans, are called *objects*, the remaining parts, such as tables and racks, are called *obstacles*. Objects are a priori unknown, obstacles are supposed to be given. The part of the environment that is completely observed by the monitoring system is called *surveillance volume*.

Figure 1: Left: The current image of a camera. Right: The silhouette image after background subtraction.

In particular the system performs in each camera a background subtraction to distinguish between objects and obstacles. As illustrated in Figure 1, this method results in a *silhouette image* that associates each pixel with either an unknown object in the *foreground* (red) or a known obstacle in the *background* (gray). The information of this image can be transferred to the 3D environment: The projection from a background pixel to the next obstacle in the environment will not contain objects, is therefore *object-free space*. The space behind the obstacle is occluded, which means an object could be hidden there.

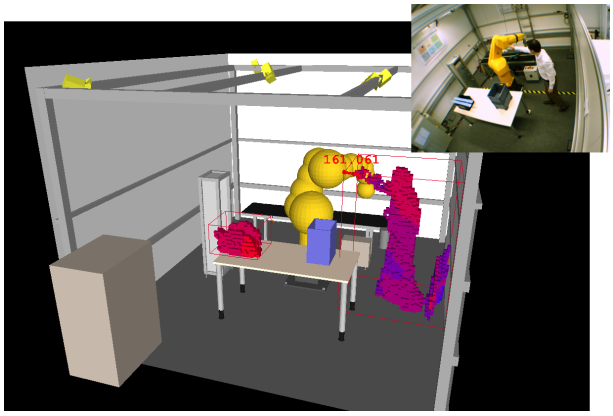

Figure 2: Reconstruction of the visual hull (red).

The reconstructed *visual hull* of an object is the approximated space of the surveillance volume, where an object is located according to the combined silhouette images of multiple cameras. Each additional camera results in more object-free space and thus refines the visual hull. In contrast to the standard visual hull, we integrate context information of obstacles in the scene and design conservative algorithms [17]. Hence, we create a conservative visual hull that avoids false-negative object detection even in the presence of occlusions, as shown in Figure 2.

Our current research aims at bringing the system closer to its applicability in real environments. One aspect we discuss, is the use of modern Full HD color cameras as high-resolution input sources. For this purpose, one must use reconstruction algorithms that can process a huge amount of data to deliver high-detail output over a large spatial area. At the same time, these algorithms must meet real-time and anytime constraints to guarantee fast responses, which are required for human-robot cooperation. Section 2 describes such an algorithm. Another focus of this

[1]Universität Bayreuth, Lehrstuhl für Angewandte Informatik III, 95440 Bayreuth, Fax: 0921 557682

[2]This work was supported by DFG (German Research Foundation) Grant He2696/11 in the project SIMERO-2 “Safety strategies for human robot cooperation and coexistence”

work is the improvement of human localization in situations with pseudo objects and coarse visual hulls due to occlusions. Additionally, we want to deal with noise and short-time detection failures in the silhouette images as so far the objects are considered to be distinguishable from the background obstacles. In order to loosen this limitation we apply temporal filtering after the reconstruction step, which will be described in Section 3. Again, the challenge is the handling of obstacles and occlusions in the scene. Furthermore, a correct camera placement is regarded, here: When the object-free space is maximized the remaining visual hull is minimized, which is the only space where the objects can be located. We will address a characterization of a respective objective function for optimization purposes in Section 4.

2 Efficient, Precise Reconstruction

Current solutions to visual hull reconstruction in a robot workspace favor naive voxel discretization over more intricate algorithms. For example, the reconstruction in [23] exploits per-voxel data parallelism to efficiently perform voxel-to-silhouette tests on a GPU at 10 ms per 256^3 voxels. Another, more optimized approach [3] restricts updates to modified entries within incoming silhouettes for a time of 100 ms per 256^3 voxels.
Yet, even optimized voxel representations are not suited for high-detail and efficient reconstruction: They do not scale well to high input and output resolutions, they do not adapt to inhomogeneous object distributions, and they do not offer reasonable early-out capabilities for supporting real-time constraints.
Hierarchical data structures, such as octrees and quadtrees, promise advantages over brute force voxel reconstruction. A sample hierarchical approach [18] builds an octree reconstruction via image-space downsampling and single-pixel projection tests at 25 ms per 128^3 voxel equivalent detail. Likewise, the algorithm [22] performs parallel, load-balanced octree reconstructions on CPU or GPU at 25 ms per 128^3 voxel equivalent.
However, no existing approach satisfies all requirements for close-quarter collaboration of robots and humans. In particular, a suitable visual hull reconstruction must be efficient, precise, and conservative. An efficient reconstruction delivers results in time for an appropriate robot reaction. While a definite time limit depends on many factors (e.g. robot speed or sensor frame rate), we desire a maximum processing time of 30 ms. We further understand a precise reconstruction as to provide centimeter resolution within a standard-sized 64m^3 robot workspace, an equivalent of at least 1024^3 voxels. In our experiments, multiple Full HD cameras are necessary to generate input for this level of detail. Finally, a conservative reconstruction guarantees that the resulting visual hull will at least contain all objects present within the scene. Given ground-truth silhouette images, this strong bias to false-positive object detection ensures workspace safety. Our approach meets all three named requirements: it is efficient, precise, and conservative.

2.1 Our Approach

In brief, we support hierarchical, incremental, real-time- and anytime-capable multi-camera reconstruction of visual hulls from silhouette images. Our respective algorithm performs two separate steps: The first step calculates conservative quadtrees over all current silhouette images. The second step incrementally computes a conservative octree reconstruction of the observed objects. This involves projecting octree nodes into camera images and efficient testing for object silhouettes within respective quadtrees. Both steps track modifications in-between consecutive input images in order to reuse quadtree test results or unmodified octree branches. We apply various knowledge-based and low-level optimizations to further improve reconstruction efficiency and precision. Finally, client applications can raise an exit flag to abort reconstruction prematurely. This allows for realtime and anytime capabilities within incremental computations.
In the following, we discuss quadtree setup, incremental octree updates, and algorithmic optimizations in detail.

2.2 Silhouette Quadtrees

We build quadtrees over all input images in a bottom-up merge process. The lowest level of each quadtree always holds the respective silhouette image. We then apply a merge function to reduce four element squares from any level to a single element within the next-coarser level. In order to guarantee conservativeness, the merge function must at most increase the perceived volume of any silhouette. We achieve this by an additional mixed-content value on higher quadtree levels.

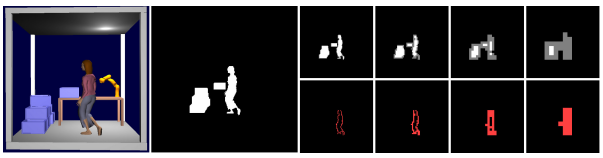

Figure 3: From left to right: A synthetic test scene with foreground objects (human, crates) and background obstacles (table, robot, work cell). A silhouette image of the foreground objects. Four quadtree levels over the next image in the sequence, with half-tones for mixed content, and corresponding modification entries in red.

Later octree updates use the resulting quadtrees to efficiently determine whether a given octree node corresponds to an actual object or object-free space. In particular, a top-down test over all quadtree levels returns an early-out result if the node completely maps to either occupied or free space. Only nodes that project to mixed content need to test against lower quadtree levels. Hence, quadtrees avoid many pixel-level or quadtree element tests for spatially coherent objects, which is a major contribution to efficient reconstruction.
We avoid another substantial number of pixel tests by incremental computations. To this end, we track pixels that changed state in-between the current and the preceding silhouette image. We then mark each quadtree ele-

ment as modified if it covers at least a single modified pixel. Later incremental updates exploit this information to maintain entire octree branches over consecutive reconstruction frames without individual per-node tests.
Figure 3 illustrates the overall quadtree build process for a sample silhouette.

2.3 Incremental Octree

Nodes within our octree carry one of three different states: Nodes can be entirely object-free, can be completely occupied by an object, or can hold mixed content. Further subdividing of object-free or full nodes is futile. Hence, such nodes form the leaves of the octree. In contrast, we split nodes of mixed content to refine reconstruction precision later on.
Apart from its state, each node also stores its projection into all silhouette images. A high-precision reconstruction cannot afford pre-calculating these projections for all nodes as in [18] due to excessive memory requirements. Hence, we rebuild all projections on creation of individual nodes. To avoid expensive per-pixel projections, we approximate node projections by conservative image-space bounding boxes. Conservative boundings at most consider additional silhouette pixels, hence maintain overall conservativeness. Finally, our approximation does not reduce reconstruction precision, as we usually stop subdividing the octree at pixel-sized boundings.
We initialize the actual octree with a single root node of mixed content that spans the entire surveillance volume. We then incrementally update the octree for each consequent series of silhouette images.
In detail, our algorithm recursively traverses the existing octree. The algorithm tests each node bounding against the modified entries of each silhouette image quadtree. If modified entries indicate that some node did not change in any camera image, we maintain the respective octree branch in its entirety without further calculations. Otherwise, node contents have been modified in at least one camera, so we need to rebuild the node state.
In order to determine the current state of a node, we check the actual contents of each silhouette quadtree within the node boundings: A node is object-free if its projection into a single camera is object-free. Likewise, a node is entirely filled by an object if its projection into all cameras maps to object silhouettes. In all other cases, the node has mixed content. For an additional efficiency gain, we cache results of these node-to-silhouette tests. In particular, we only need to rebuild silhouette test results if preceding modification tests indicated a changed silhouette within the node projection.
Once our algorithm has set a new state on a node, it updates the octree accordingly: The algorithm deletes any children of empty or full nodes, and potentially creates new children on mixed nodes.
Incremental updates then recurse on the children of mixed nodes. As mentioned previously, recursion usually continues until a node projects to a single-pixel bounding box in each camera. Reconstruction now has exploited all available input information, and further subdivision is pointless. Once our algorithm has explored all branches to either full, empty, or pixel-sized nodes, reconstruction terminates with an updated, conservative octree.
Finally, we introduce a second exit condition to implement realtime capabilities: Client applications may raise an external abort flag in order to force a return of premature reconstruction results. In this case, reconstruction returns instantly. To satisfy conservativeness, we must now consider any untouched octree branches as mixed content (i.e. potential objects). Realtime capabilities are especially important for fixed-interval distance tests, as required for path planning in robotics.
See Figure 4 for an example incremental update.

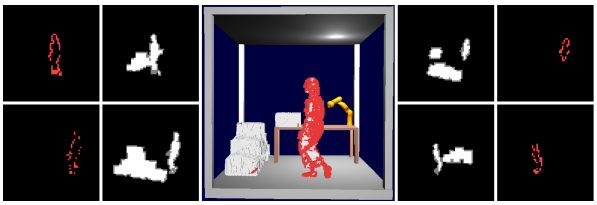

Figure 4: Left, right: Four incoming silhouettes and associated modification tags on a synthetic scene, both in form of quadtrees as in Figure 3. Center: A corresponding reconstruction. Our algorithm only touched octree branches that contain red nodes.

2.4 Optimizations

Once our algorithm has finished, we apply additional knowledge-based refinement criteria [16] to increase reconstruction precision. For instance, if the target application only requires to avoid robot-human collisions, we can ignore any coherent object that does not meet the minimum volume occupied by a human. All knowledge-based refinement efficiently integrates into our octree: We only need to touch few nodes for volume or neighborhood tests, as opposed to an expensive voxel flood fill.
We also parallelize reconstruction over multiple root nodes in order to achieve maximum throughput on modern multi-core CPUs. We do not use advanced per-thread load-balancing as in [22]. Instead, we generate a large number of initial roots (e.g. 512 nodes) and distribute these over any waiting threads. This enables fully data-parallel processing per root node. Consequentially, our experiments still achieve an almost linear speedup.

2.5 Evaluation

We evaluated our reconstruction algorithm in an online test on synthetic data. In particular, we placed multiple virtual cameras into an animated, virtual scene. These cameras generated high-resolution ground-truth silhouettes of objects in the scene at real-time frame rates via OpenGL rendering. Our algorithm then combined resulting silhouettes with a-priori ground-truth camera parameters to update an incremental octree representation of the

virtual scene. We measured performance on commodity hardware, an i5-3320m dual-core notebook CPU and 16GB of dual-channel RAM. Even with these constraints, our algorithm delivered the framerate of 30hz desired for high-resolution input and high-detail output.

3 Human Tracking with Obstacles

We apply a human tracking approach to reconstructed voxel data (occupancy grids) in order to improve the human localization in situations with pseudo objects and coarse visual hulls due to occlusions. In addition, the concept of the conservative visual hull described so far assumes ground-truth silhouette images, which means that objects are considered to be distinguishable from the background obstacles. Temporal filtering supposedly looses this limitation and allows some noise as well as short-time detection failures.

Methods that use voxel data as input for human tracking with skeletal models are applied in the field of markerless motion capturing, e.g. in [15, 5]. Human poses and motions are required for animations in movies and computer games or for motion analysis in sports and medical applications. Motion capturing approaches commonly assume empty scenes, so that they only have to handle the problem of self-occlusion to get sufficiently good human state approximations.

By comparison, tracking in industrial environments is similar to tracking in home or office environments [13, 6], whereas less camera sensors might be involved and the scenes usually contain obstacles like tables and chairs. Due to the coarse human approximation, which results from the reconstruction in such scenarios, skeletal models are not convenient, especially when only parts of the humans are visible in the sensors. Instead, simple shape models like ellipsoids [6] are proposed. Different tracking methods are applied, e.g. a particle filter [6] or the Viterbi algorithm [13].

However, we did not find an approach that thoroughly considers obstacle and occlusion information in the tracking step which is adequate for industrial applications. Thus, we discuss the integration of 3D modeled obstacles as context knowledge in two tracking steps: the likelihood evaluation and the particle propagation.

3.1 Tracking Approach

We search for the best estimation $\mathbf{X}_t$ of the unknown state $\mathbf{x}_t$ of the human at time t by using the particle filter as applied in [6]. The particle filter [1] is chosen to handle non-Gaussian posterior probability densities. Those can easily appear in scenes with obstacles, coarse reconstruction results, and pseudo objects. The amount of tracked persons in the targeted scenario is small. Thus, for every human a new instance of the particle filter is initialized, despite the growing computation time. The Standard Sampling Importance Resampling (SIR) [10] is applied.

A simple ellipsoid shape model approximates the human with the unknown state $\mathbf{x}_t$. We have given the state space $\mathbf{x}_t = \left(x_t, y_t, z_t, k_t, l_t, m_t\right) \in \mathbb{R}^6$ with x_t, y_t, z_t representing the ellipsoid center and k_t, l_t, m_t representing the length of the ellipsoid axes, that are aligned to the axes of the voxel space. The observation for each time step t is a set of n voxels $\mathcal{V} = \{\mathbf{v}_k\}$ with $k = 1, .., n$ whereas $\mathbf{v}_k = \left(a_k, b_k, c_k\right)$ with a_k, b_k, $c_k \in \mathbb{Z}$. We consider four different voxel states for use in the likelihood function of the particle filter: $f_t : \mathcal{V} \rightarrow \{\textit{known}, \textit{occluded}, \textit{unknown}, \textit{free}\}$. The synthesis of the respective, pairwise disjoint voxel subsets $\mathcal{V} = \mathcal{U}_t \cup \mathcal{K}_t \cup \mathcal{O}_t \cup \mathcal{F}_t$ is now explained in detail.

a) In an online reconstruction of the visual hull we create voxels that approximate the volume of the visible parts of humans (unknown objects) and label those as *unknown*. We get the respective voxel set $\mathcal{U}_t$. In comparison to the standard visual hull, context knowledge of the obstacles is integrated in the reconstruction process. More precisely, a depth map is built for each camera, which holds, for each pixel, the distance to the closest modeled obstacle, similar to a range sensor [17]. Thus, a camera contributes only to the classification of a voxel (*occupied* or not) if the voxel is not occluded in that camera. The result is a correct visual hull for all parts of the objects which are visible for at least one camera.

b) A voxelization of the modeled obstacles (triangle meshes) gives us a set $\mathcal{K}_t$ of *known* voxels in the voxel space. Those voxels represent the physically occupied volume in the scene (except the occupation by unknown objects).

c) With help of the depth maps we construct the visual hull of the known obstacles to get a set of voxels $\mathcal{Z}_t$ that are not visible in any camera. This set is composed of voxels that are empty and voxels that contain parts of obstacles. Hence, a valid assumption is that $\mathcal{K}_t$ of *known* voxels is enclosed by $\mathcal{Z}_t$ with $\mathcal{K}_t \subseteq \mathcal{Z}_t$. We produce the set of empty *occluded* voxels $\mathcal{O}_t$ with $\mathcal{O}_t = \mathcal{Z}_t \setminus \mathcal{K}_t$.

d) All remaining voxels are labeled as "free", given the respective voxel set $\mathcal{F}_t$.

In the case of dynamic obstacles, we would need to compute the steps b) and c) at every time step t. Currently we use static obstacles only and perform the computation once in an offline step.

3.1.1 Likelihood Evaluation

The authors in [6] use the associated ellipsoid of each particle j in state $\mathbf{x}_t^j$ and consider the binary information of voxels $\mathcal{M}_t^j$ (*occupied* or not) that are located inside the ellipsoid for particle weighting. We adopt this method but extend the likelihood function with the described voxel states of obstacles (state *known*) and occlusions (state *occluded*). Every voxel $q \in \mathcal{M}_t^j$ is assigned a different weight $s(\mathbf{x}_t^j, q) \in \mathbb{R}$.

$$s\left(\mathbf{x}_t^j, q\right) = \begin{cases} \rho & \text{if} \quad f_t(\mathbf{v}_k) = \textit{unknown} \\ \sigma & \text{if} \quad f_t(\mathbf{v}_k) = \textit{occluded} \\ \tau & \text{if} \quad f_t(\mathbf{v}_k) = \textit{known} \\ \epsilon & \text{if} \quad f_t(\mathbf{v}_k) = \textit{free} \end{cases}$$

In the first line (ρ) we reward (value > 0) every *unknown* voxel within the ellipsoid. In the second line (σ), also every *occluded* voxel is rewarded. Even though it is not visible for any camera, it may nevertheless contain an unknown object. This reward allows a person to completely move into an occluded volume without leading to a termination of the filter. To avoid that the filter moves into an occlusion, though the person does not, it is required that $\sigma < \rho$. In addition, for a person that steps outside an occlusion the filter would stay in the occlusion otherwise (depending on the size of the occlusion). The third line (τ) requires a penalty (value ≤ 0) , because every voxel which is located inside a known modeled obstacle can impossibly contain an unknown object (human) at the same time. Line four (ϵ) calls for a penalty, too, because a free voxel is not part of an unknown object. The weight w_t^j of a particle is updated proportional to the prior density $w_t^j \propto p(\mathbf{z}_t, \mathbf{x}_t^j)$ with the observation $\mathbf{z_t}$ of all voxels (see [6]). We normalize each particle weight to $[0, 1]$ by the sum S of all individual ellipsoid weights, whereas an ellipsoid weight is the sum of all respective weights $s(\mathbf{x}_t^j, q)$.

$$p(\mathbf{z}_t|\mathbf{x}_t^j) = \max\left(\frac{1}{S}\sum_q s\left(\mathbf{x}_t^j, q\right), 0\right), \text{ with } q \in \mathcal{M}_t^j$$

The SIR [10] is conducted subsequently.

3.1.2 Particle Propagation

The new state $\mathbf{x}_{t+1}^r$ of each resampled particle r is propagated with $\mathbf{x}_{t+1}^r = \mathbf{x}_t^r + \mathcal{N}$. A Gaussian diffusion $\mathcal{N}$ is added to account for noise and changes in direction. As constraint, a maximum ellipsoid volume is assumed. Since creating an exact motion model is the key for good tracking results, we are currently working on refining the prediction step, e.g. by applying adaptive noise parameters.

During multi-person tracking, we observe situations in which a filter located in an occlusion (e.g. of a person under the table) completely diverges through an obstacle into the adjacent measurement of another filter (e.g. of a person on the table). This is caused by the gradient in the weights of *occluded* voxels (σ) and *unknown* voxels (ρ). A stringent condition to avoid such effects can be met by replacing all particles whose ellipsoids collide with an obstacle. Unfortunately, we deem this inappropriate for two reasons. First, we observe that the filter terminates in certain situations, e.g. when a person crawls under a table. In that case, all resampled particles that collide, e.g. with the table legs, are replaced. Thus, only a few valid particles are predicted under the table, which finally results in a termination of the tracker. Second, the ellipsoid is just a coarse approximation of a human's shape, so it should not disturb some invalid *known* voxels at the exterior of the ellipsoid. Moreover, this can improve tracking situations in which persons are located very close to an obstacle. One solution to handle the last two issues is already implemented with the penalization in the likelihood (τ). However, it does not avoid divergences through obstacles in rather unfavorable situations that can appear in multi-person tracking. Hence, we need a combination of the penalization in the likelihood and the prevention of a complete divergence through obstacles in the particle propagation step. Therefore, we propose the following approach.

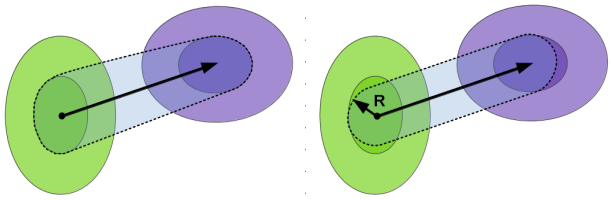

Figure 5: Left: Sweeping volume for collision test (dotted), constructed between the ellipsoid kernels (dark colored) of the predicted particle at state $\mathbf{x}_{t+1}^r$ (violet) and the particle at the previous state $\mathbf{x}_t^r$ (green). Right: Using a capsule with radius R as approximation of the sweeping volume for efficiency reasons.

A sweeping volume (Figure 5, left) is spanned by the ellipsoid kernels of a predicted particle at state $\mathbf{x}_{t+1}^r$ and the particle at the previous state $\mathbf{x}_t^r$. The kernels are smaller versions of their parents with the same proportions and the same center. The sweeping volume must not collide with an obstacle, otherwise the respective particle is rejected and replaced by a collision-free one.

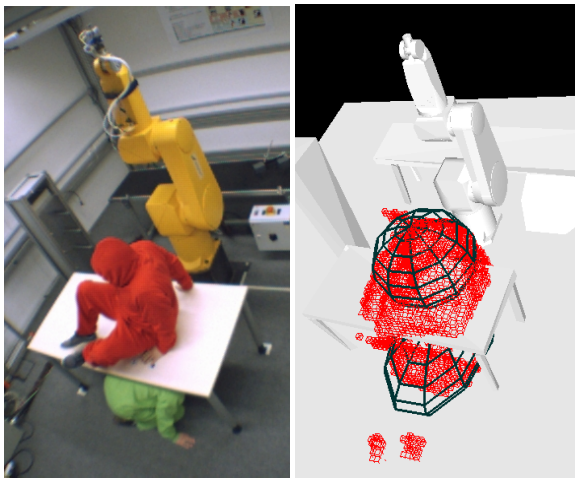

Figure 6: Left: Real-world situation of two persons on and below a table. Right: The *unknown* voxels of the observation are shown (red) as well as the ellipsoids of the state estimations $\mathbf{X}_t$ (blue).

A collision outside the kernel is ignored and will be penalized in the next likelihood evaluation step at time $t+1$. As the execution of the described collision test would be very time consuming, we approximate the ideal sweeping volume through a capsule with radius R aligned to the ellipsoid centers, as shown in Figure 5, right. The radius R should be smaller than the minimum lengths of all six ellipsoid axes of the parent ellipsoids. For efficiency the obstacles are also modeled as capsules and other suited primitives.

3.2 Experiments

The proposed tracking method is evaluated by a visual analysis of a real-world scenario as shown in Figure 6. We used the following parameters: $\rho = 1.0$, $\sigma = 0.1$, $\tau = -5.0$, $\epsilon = -0.1$, $R = 200\,mm$.

4 Camera Placement

One of the failures of such a system is caused by the incorrect placement of the cameras. It is possible to optimize a camera network manually, by heuristics like placing the cameras in the corners of the robot work cell. But heuristics, as well as common sense, can be error-prone, e.g. what if an obstacle occludes the view from the corner of the work cell? Hence, the computational automation of camera placement is formulated as an optimization problem, here.

Depending on the application of the camera network, such an optimization could be the maximization of the field of view of a camera, minimization of the amount of cameras or costs, or the minimization of an error. The latter could be the error that is made when reconstructing a target as in [11] and [25]. In our case, the visual hull is a conservative approximation of an object, meaning the object is completely inside the visual hull. Thus, the objects are better outlined if their visual hull is minimal.

There has been extensive research in the optimal placement of cameras. We will address an exact measure of the volume of the field of view of a camera and a measure of the volume of the visual hull with the prospect of constructing derivatives, here. This could be of help in the maximization of continuous values as in [20, 14], in contrast to the maximization of integer values as the number of objects or paths. Also, it could be helpful for calculating the visual hull used by [11] or [25], or when deriving continuous constraints to the art gallery problem as in [2, 7, 24]. But within their visibility analysis, all of the mentioned publications discretize the surveillance area into cells, also called voxels. After simulating the visibility of every voxel, the visible voxels are then combined to the field of view or visual hull. The major drawback is that the simulated objective cannot be derivated, which is necessary for so many deterministic non-linear programs.

The frequent use of voxels is a hint that an exact visibility analysis as approached in this paper has not been done, yet. In 2006 the authors of [8] have started to analyse the volume of the field of view in 2D and to examine the smoothness of the volume of an omni-directional camera in a polygonial environment. His result: The volume is almost everywhere locally Lipschitz.

Our work is closest to his work. We will transfer some ideas of his into 3D and add the theory about the minimization of the visual hull of an object.

4.1 The visibility analysis

Finding the field of view of one/several cameras is called *visibility analysis* in [20] and is a central issue in camera placement. Whereas the visibility analysis is part of the objective function when maximizing the field of view, the visibility analysis is part of the constraints minimizing costs or errors. E.g. one usually wants the complete work cell to be visible when minimizing the costs of a camera network. But the visibility analysis is a real challenge as shown in the next section.

4.1.1 The challenges

Computational problems of the visibility analysis include: First, the field of view of one camera, let alone a network of cameras, can only be derived geometrically. That takes time, e.g. [19] reported "For very high dimensional spaces (>8), ..., it sometimes took several hours to jump to a better solution". For comparison: The positioning and orientating of one camera in a plane includes three variables (x- and y-position, and one orientation angle). That means, even in a 2D-room, eight variables correspond to not even three cameras. Second, recent research, that could reduce the complexity of the task, utilizes z-buffer methods. But these methods are only suitable for computing the image of the field of view, not the geometrical shape. Third, the field of view of several cameras is an intersection/union of polyhedra, which is known to be a non-robust computation.
As an objective function the visibility analysis is challenging, too, e.g. have a look at the volume of the field of view of one camera: First, it is non-linear and only piecewise differentiable in the placement and orientation of a camera. Second, given the geometrical property of the problem only few approaches exist that provide a formula of the volume of the (unlimited) field of view in 2D, none in 3D with limited field of view. Thus, its convexity or differentiability is hardly analysed in literature. Third, lacking a formula, a gradient of the objective made of the partial derivatives of the volume is not available. The numerical approximation by the difference quotient is an alternative, but in a one dimensional domain it needs two function evaluations, in a two dimensional domain three function evaluations, etc. This adds to the complexity of the visibility analysis.
These and other challenges are why we have chosen to start analysing the limited field of view of a camera in a work cell in 3D including a formula.

4.1.2 The field of view of a camera as a star-shaped polyhedron

We have investigated the limited field of view of one camera. The field of view inside a polyhedron turns out to be a polyhedron with three types of faces: Environmental faces (E) share at least three points with the work cell. Opening faces (O) are the faces along the limits of the field of view. For the third type edges are relevant, which inner angle exceeds π. These are called *reflex edges*. The

third type of faces are projection faces (P), their hyperplane includes the viewpoint x and a reflex edge.
Since we have found that all vertices of the field of view except x are on an environmental face, 3!=6 types of vertices can be deduced: EEE, EEP, EEO, EPP, EOO, EPO; The reflex edges of two different projection faces can be skew or can intersect, denoted by EPPs and EPPi, respectively. We can show that the path of an EEE when moving x is a single point, the paths of EEP, EPPi, EEO, and EOO are line segments or points, and the paths of EPPs and EPO are 2-dimensional quadrics. The seven types of vertices are illustrated in Figure 7.

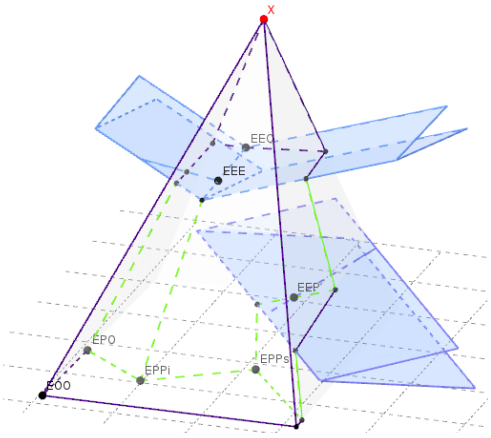

Figure 7: The seven types of vertices of the field of view result from three types of faces, environmental (E, blue face), projection (P, bounded by green and blue lines), and opening faces (O, bounded by purple and green lines).

With these vertices, we can state an analytical formula for the volume of the field of view suitable for differentiation: We know that the field of view $V(x)$ is star-shaped from the viewpoint $x \in \mathbb{R}^3$. That means P can be divided into pyramids with environmental faces as a base. Therefore, let F be a polygon in space with x not in F's hyper plane. We define $[x, F] := \{y \in \mathbb{R}^3 \mid \exists f \in F : y \in [x, f]\}$ as a *pyramid* and F its *base face*. Then, with the set of all environmental faces $\mathcal{F}$ of the polyhedron of the field of view V, the volume can be calculated by the area of the base face $\lambda(F)$ and the distance $\mathrm{d}(x, F)$:

$$\lambda(P) = \sum_{F \in \mathcal{F}} \lambda([x, F]) \quad \text{with } \lambda([x, F]) = \frac{1}{3}\mathrm{d}(x, F) \cdot \lambda(F) \tag{1}$$

Thereby, the area $\lambda(F)$ can be developed as a function of the vertices' coordinates: Let $F \subset \mathbb{R}^2$ be a polygonal area with vertices $v_i \in \mathbb{R}^2$, $i = 1, ..., n$, and let $v_i^{(x/y)}$ denote the x- and y-components in the x-y-Plane. From [4, cf. Section 3.5.2.2] we know:

$$\lambda(F) = \frac{1}{2}\sum_i (v^x_{(i-1)} - v^x_{(i+1)}) \cdot v^y_i + \left((v^x_n - v^x_2) \cdot v^y_1\right) + \left((v^x_{(n-1)} - v^x_1) \cdot v^y_n\right)$$

Thus, we have been able to develop the volume of the field of view suitable for differentiation and convexity analysis.
We can show that this volume is piecewise continuously differentiable. The non-continuous positions of the camera x lie on planes. The volume is non-continuous, when passing x through a wall or a reflex edge. The non-differentiable positions lie on planes and 3D-quadrics. The volume is non-differentiable, when two vertices vanish and it is not two times differentiable, when one vertex vanishes. A *vertex vanishes* when it hits another face: After a vertex hits an opening face it is outside the frustum. Similarly, after it hits a projection face it is occluded behind a reflex edge.

4.1.3 The minimization of the error of the visual hull

The theory about the vertices of the field of view can be enlarged, when considering not only the maximum field of view as an objective, but also the minimum error of the visual hull. We have already stated that the minimization of the visual hull of an object corresponds to the minimization of the error of reconstruction. This corresponds to a maximization of the definitely object free space.
So, in case the camera placer knew where the dynamical objects are, the maximization of the definitely-object-free-space is a suitable alternative to maximizing the field of view.

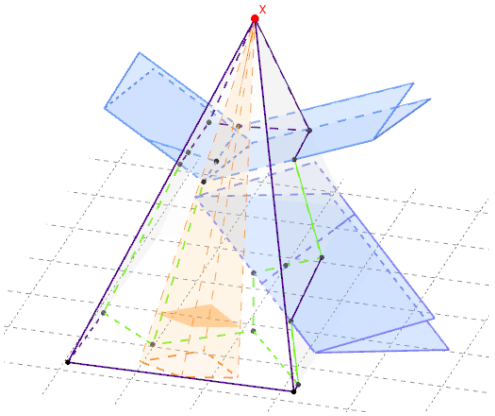

Figure 8: Figure 7 with an additional orange object resembling a dynamic object; The volume of the definitely-object-free-space is the field of view minus the orange pyramid from x to the bottom of the environment.

In Figure 8 the same image as in Figure 7 is depicted, with an additional orange object, resembling a dynamic object. The dynamic object casts a shadow of an orange pyramid from x to the bottom of the environment. As before, the volume of this pyramid can be calculated by the vertices on top of an environmentally induced face (the face below the dynamical object). The vertices of the orange pyramid are of type EEE, EEP, EPPi, EPPs, EEO, or EPO, not EOO. The volume of the definitely-object-free-space is the field of view V minus the orange pyramid. In short, the only thing that has changed in formula (1) is the set $\mathcal{F}$, which is now the set of environmentally induced

facets with the additional property that the faces are not part of a dynamical object. For this calculation the object needs to be part of the polyhedron that resembles the environment.

5 Conclusions and Future Work

In this paper we focused on a surveillance system with multiple static calibrated color cameras for human localization in industrial environments with occluding obstacles. First, we discussed the use of Full HD resolution cameras as input for the reconstruction of the visual hull. A conservative octree-based reconstruction is proposed, which meets real-time and anytime requirements. We evaluated our reconstruction algorithm in simulated scenarios and achieved the desired framerate of 30 hz. Second, we applied a particle filter to a voxel-based visual hull and integrated knowledge of occlusions and modeled obstacles. In result an occupancy grid with four different voxel states is considered in the likelihood function. To avoid the divergence of a filter through obstacles (an issue in multi-person tracking) a collision test with obstacles is conducted in the motion model. Third, we considered a favorable camera placement. Therefore, the camera placement is formulated as an optimization problem. To maximize the object-free space in the scene, not only the maximum field of view is considered in an objective function, but also the minimization of the error of reconstruction.
Further research should aim at combining the individual software components to a joint system. A framework for robust multi-person tracking is being developed. Fault tolerant concepts as provided in [21] might be adapted to the reconstruction.

References

[1] Sanjeev Arulampalam, Simon Maskell, Neil J. Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[2] Enric Becker, Gutemberg Guerra-Filho, and Fillia Makedon. Automatic sensor placement in a 3d volume. In *Proceedings of the 2nd International Conference on Pervasive Technologies Related to Assistive Environments*, page 36. ACM, 2009.

[3] Ali Bigdelou, Alexander Ladikos, and Nassir Navab. Incremental visual hull reconstruction. In *BMVC*. British Machine Vision Association, 2009.

[4] Ilja N. Bronstein, Konstantin A. Semendjajew, Gerhard Musiol, and Heiner Mühling. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 2001.

[5] Fabrice Caillette and Toby Howard. *Real-time markerless 3-D human body tracking*. PhD thesis, 2006.

[6] Christian Canton-Ferrer, Joseph R. Casas, Montse Pardàs, and Enric Monte. Multi-camera multi-object voxel-based monte carlo 3d tracking strategies. *EURASIP J. Adv. Sig. Proc.*, 2011:114, 2011.

[7] Ugur Murat Erdem and Stan Sclaroff. Optimal placement of cameras in floorplans to satisfy task requirement and cost constraints. In *OMNIVIS Workshop*. Citeseer, 2004.

[8] Anurag Ganguli, Jorge Cortés, and Francesco Bullo. Maximizing visibility in nonconvex polyhgons: Nonsmooth analysis and gradient algorithm design. *SIAM Journal on Control and Optimization*, 45(5):1657–1679, 2006.

[9] Thorsten Gecks. *Sensorbasierte, echtzeitfähige Online-Bahnplanung für die Mensch-Roboter-Koexistenz.* PhD thesis, University of Bayreuth, 2011. http://d-nb.info/1015875025.

[10] Neil J. Gordon, David J. Salmond, and Adria F. M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEEE Proceedings F, Radar and Signal Processing*, 140(2):107–113, 1993.

[11] Maria L. Hänel, Stefan Kuhn, Dominik Henrich, Lars Grüne, and Jürgen. Pannek. Optimal camera placement to measure distances regarding static and dynamic obstacles. *International Journal of Sensor Networks*, 12(1):25–36, 2012.

[12] Dominik Henrich and Thorsten Gecks. Multi-camera collision detection between known and unknown objects. In *ICDSC*, pages 1–10. IEEE, 2008.

[13] Martin Hofmann, Moritz Kaiser, Nicolas H. Lehment, and Gerhard Rigoll. Event detection in a smart home environment using viterbi filtering and graph cuts in a 3d voxel occupancy grid. In Leonid Mestetskiy and José Braz, editors, *VISAPP*, pages 242–247. SciTePress, 2011.

[14] Robert J. Holt, Man Hong, Rainer Martini, Iraban Mukherjee, Ravi Netravali, and Jing Wang. Summary of results on optimal camera placement for boundary monitoring. In *Proceedings of SPIE*, volume 6570, page 657005, 2007.

[15] Roland Kehl and Luc J. Van Gool. Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding*, 104(2-3):190–209, 2006.

[16] Stefan Kuhn. *Wissens- und sensorbasierte geometrische Rekonstruktion*. PhD thesis, University of Bayreuth, 2012.

[17] Stefan Kuhn and Dominik Henrich. Multi-view reconstruction of unknown objects within a known environment. In George Bebis and Richard D. Boyle, editors, *ISVC (1)*, volume 5875 of *Lecture Notes in Computer Science*, pages 784–795. Springer, 2009.

[18] Alexander Ladikos, Selim Benhimane, and Nassir Navab. Real-time 3d reconstruction for collision avoidance in interventional environments. In *Proceedings of the 11th International Conference on Medical Image Computing and Computer-Assisted Intervention, Part II*, pages 526–534, Berlin, Heidelberg, 2008. Springer-Verlag.

[19] Anurag Mittal and Larry S. Davis. A general method for sensor planning in multi-sensor systems: Extension to random occlusion. *International Journal of Computer Vision*, 76(1):31–52, 2008.

[20] Alan T. Murray, Kamyoung Kim, James W. Davis, Raghu Machiraju, and Richard E. Parent. Coverage optimization to support security monitoring. *Computers, Environment and Urban Systems*, 31(2):133–147, 2007.

[21] Antje Ober and Dominik Henrich. A safe fault tolerant multi-view approach for vision-based protective devices. In *AVSS*, pages 17–25. IEEE Computer Society, 2010.

[22] Luciano Soares, Clément Ménier, Bruno Raffin, and Jean-Louis Roc. Work stealing for time-constrained octree exploration: Application to real-time 3d modeling. In *Proceedings of the 7th Eurographics Conference on Parallel Graphics and Visualization*, EG PGV'07, pages 61–68, Aire-la-Ville, Switzerland, 2007. Eurographics Association.

[23] Dominik Stengel, Thomas Wiedemann, and Birgit Vogel-Heuser. Efficient 3d voxel reconstruction of human shape within robotic work cells. In *Mechatronics and Automation (ICMA), 2012 International Conference on*, pages 1386–1392, Chengdu, August 2012. IEEE.

[24] Kenichi Yabuta and Hitoshi Kitazawa. Optimum camera placement considering camera specification for security monitoring. In *IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008*, pages 2114–2117, 2008.

[25] Danny B. Yang, Jaewon Shin, Leonidas J. Guibas, and Ali.O. Ercan. Sensor tasking for occupancy reasoning in a network of cameras. In *Proceedings of 2nd IEEE International Conference on Broadband Communications, Networks and Systems (BaseNets' 04)*. Citeseer, 2004.