Optimal Camera Placement to measure Distances Conservativly Regarding Static and Dynamic Obstacles

M. Hänel and S. Kuhn and D. Henrich
Angewandte Informatik III
Universität Bayreuth
95440 Bayreuth, Germany
{maria.haenel,stefan.kuhn,
dominik.henrich}@uni-bayreuth.de

J. Pannek and L. Grüne
Lehrstuhl für Angewandte Mathematik
Universität Bayreuth
95440 Bayreuth, Germany
{juergen.pannek,lars.gruene}
@uni-bayreuth.de

***Abstract*— In modern production facilities industrial robots and humans are supposed to interact sharing a common working area. In order to avoid collisions, the distances between objects need to be measured conservatively which can be done by a camera network. To estimate the acquired distance, unmodelled objects, e.g., an interacting human, need to be modelled and distinguished from premodelled objects like workbenches or robots by image processing such as the background subtraction method.**

The quality of such an approach massively depends on the settings of the camera network, that is the positions and orientations of the individual cameras. Of particular interest in this context is the minimization of the error of the distance using the objects modelled by the background subtraction method instead of the real objects. Here, we show how this minimization can be formulated as an abstract optimization problem. Moreover, we state various aspects on the implementation as well as reasons for the selection of a suitable optimization method, analyze the complexity of the proposed method and present a basic version used for extensive experiments.

***Index Terms*— Closed range photogrammetry, optimization, camera network, camera placement, error minimization**

I. Introduction

Nowadays, human/machine interaction is no longer restricted to humans programming machines and operating them from outside their working range. Instead, one tries to increase the efficiency of such a cooperation by allowing both actors to share the same working area. In such a context, safety precautions need to be imposed to avoid collisions, i.e., the distance between human and machine interacting in a common area needs to be reconstructed continuously in order to detect critical situations. To this end, usually a network of cameras is installed to, e.g., ensure that every corner of the room can be watched, every trail can be followed or every object can be reconstructed correctly. Within this work, we focus on computing an optimal configuration of the camera network in order to measure the distances as correct as possible but still conservatively.

After a brief review on previous results concerning the predescribed distance measurement, we show how an unmodelled (human) object can be contoured by a 3D background subtraction method. We extend this scheme to cover both static and dynamic obstacles, some of which are modelled in advance but still occlude the vision of the sensor. In Section III, we rigorously formulate the problem of minimizing the error made by using the associated model instead of the original collective of unmodelled objects. Considering the implementation of a solution method, we discuss various difficulties such as, e.g., the evaluation of the intersection of the cones corresponding to each camera in Section IV and also give an outline of concepts to work these issues. In the final Sections V and VI, we analyze the complexity of our basic implementation by a series of numerical experiments and conclude the article by given an outlook on methods to further improve the proposed method.

This work is part of the project SIMERO 2 and is supported by Deutsche Forschungsgemeinschaft.
Jürgen Pannek was partially supported by DFG Grant Gr1569/12-1 within the priority research program 1305.

II. State of the art

Many camera placement methods have to deal with a trade-off between the quality of observations and the quantity of pieces of information which are captured by the cameras. The latter aspect is important for camera networks which have to decide *whether* an item or an action has been observed. There have been investigations about how to position and orientate cameras subject to observing a maximal number of surfaces [8] and different courses of action [1,5,6] as well as maximizing the volume of the surveillance aread [14] or the number of objects [11]–[13]. Another common goal in this context is to be able to observe all items of a given set but minimize the amount of cameras in addition to obtain their positions and orientations [4,11]–[13]. This issue is called “Art Gallery Problem” especially when speaking of two–dimensional space.

Apart from deciding whether an object has been detected by a camera network, another task is to obtain *detailed* geometrical data of the observed item like its position and measurements of its corners, curves, surfaces, objects etc. As described in [10] determining this information for distances smaller than a few hundred meters by cameras belongs to the field ‘close range photogrammetry’. In order to configure a camera network to cope with such tasks, one usually minimizes the error of observed and reconstructed items. Often the phrase ‘Photogrammetric Network Design’ is used to express minimizing the reconstruction error for several (three-dimensional) points. The default assumption in this context, however, is that no occlusions occur, cf. [7,16]–[19] for details. Optimally localizing an entire object which is not occluded is an assignment treated in [3]. Furthermore, many approaches compensate for the increasing complexity of the problem by oversimplifying matters: One common approach is to restrain the amount of cameras (in [7] two cameras are used) or their position and orientation. Considering the latter, known approaches are the viewing sphere model given in [18,19] or the idea of situating all cameras on a plane and orientating them horizontally, cf. [3]).

In contrast to these approaches, we discuss optimizing positions and orientations of cameras in a network in the context of the background subtraction method which is used to determine a visual hull of a solid object. By means of this visual hull, distances can be computed easily which renders this approach to be a different simplification. Occlusions of solids to be reconstructed obscure the view and enlarge the visual hull. In order to get the minimal error of the construction of the hull, [23] assumes that minimizing the occuring occlusions of solids also reduces and thus specifies their possibile locations. However, neither obstacles nor opening angles other than π are discussed in [23] and additionally the orientation of the camera is neglected as a variable since it is simply orientated towards the object. In [2], static obstacles are considered but the amount of cameras is chosen out of a preinstalled set.

Since we are not interested in optimizing the quantity of observed objects but the quality of data, our approach is different to most of the discussed results. Note that the quality of information can be obtained by various types of image processing. Here, we consider the background subtraction method to obtain a visual hull of a given object. Within our approach, we optimize the positions and orientations of a fixed amount of cameras as to minimize the error that is made by evaluating distances to the visual hull. In contrast to existing results, our goal is to incorporate the aspects of occuring static or dynamic obstacles into our calculations but also to exploit all degrees of freedom available in an unconstrained camera network. Nevertheless, distances are to be evaluated conservativly.

III. Visibility Analysis

Within this section, it will be shown how to condition the objective function on the cameras position and orientation, thus, we successively build up a mathematical representation of the optimization problem. We start off by defining the critical area as well as the to be reconstructed unmodelled objects and corresponding abstract models in Section III-A. This will allow us to formally state the objective function which we aim to minimize. In the following Section III-B, we define the camera network and its degrees of freedom, i.e. the position and orientation of each camera. These degrees of freedom allow us to parametrize the model of the to be reconstructed object. Additionally, this tuple of degrees of freedom will serve as an optimization variable in the minimization problem stated in Section III-C. To cover all possible scenarios, this problem is extended by incorporating both static and dynamic obstacles as well as an evolving time component.

A. Formalizing the Problem

Let $\mathbb{U} \subset \mathbb{R}^3$ be a spacial area based on which information about humans, perils, obstacles and also cameras can be given. Consider $\mathbb{S} \subset \mathbb{U}$ to be the surveillance area, where critical points of the set $C \subset \mathbb{S}$ as well as objects, such a human or a robot, are monitored.

For the moment, we neglect obstacles completely, we just distinguish two types of objects, to explain the basic idea of reconstructing an object by the means of a camera network: If a detailed model of an object exists describing its appearance like location, shape, color or else, the object is called *modelled*. If this is not the case the object is called *unmodelled*. This is motivated by the following scenario: If humans move unpredictably within the surveillance area, i.e. without a given route, their appearance is unmodelled and needs to be reconstructed to be used for further calculations. The model of an unmodelled object can be reconstructed by the means of a camera network. Therefore, let $O_u(a) \subset \mathbb{S}$ be a complete set of points included in one or more unmodelled objects, depending on the appearance of unmodelled objects specified by the parameter $a \in \mathbb{R}^k$. We refer to these objects as *unmodelled collective*. Since automaticly placing the cameras for such a scenario is incomputable without information on the unmodelled collective, we impose the assumption that the distribution $\mathcal{P} : 2^{\mathbb{R}^k} \to [0,1]$ of the appearance $a \in \mathbb{R}^k$ is known.

As the safety of a human being must be guaranteed in any case, the distance

$$d(C, O_u(a)) := \min\{d(x,y) \mid x \in C, y \in O_u(a)\}$$

has to be computed conservatively and security measures need to be taken if the unmodelled collective $O_u(a)$ appoaches the critical points C. Here $d(\cdot,\cdot)$ denotes a standard distance function. If the exact set $O_u(a)$ was known, this distance could be evaluated easily. As we do not directly know the value of $a \in \mathbb{R}^k$ and therefore can only guess the points that are included in $O_u(a)$, we need to approximate a (as a consequence also conservative) model $M(a) \subset \mathbb{S}$, see Fig. 1.

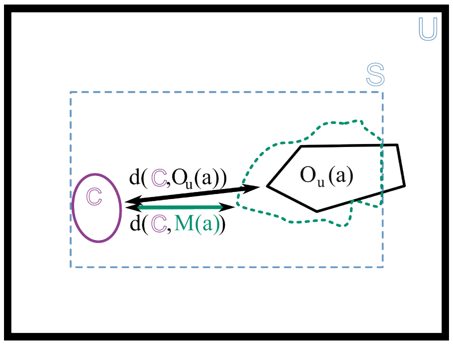

Fig. 1. Surveillance area $\mathbb{S}$: Distance between critical points C and unmodelled collective (black) and distance to the approximated model (green)

Note that for now $M(a)$ is an abstract approximation of $O_u(a)$ with respect to the parameter a only. In order to actually compute $M(a)$, a sensor network and its degrees of freedom come into play, see Section III-B for details. Still, the abstract approximation allows us to formalize our overall task, i.e. to minimize the difference between the approximation based distance $d(C, M(a))$ and the real distance $d(C, O_u(a)))$. Taking the assumed distribution of the parameter a into account, we aim to minimize the functional

$$\int\limits_{a\in\mathbb{R}^k} \Big[d(C, O_u(a)) - d(C, M(a))\Big]^2 \, d\mathcal{P}(a). \tag{1}$$

Note that for the optimization we need to be aware of possible appearances of the object in order to let the integral pass through their space $a \in \mathbb{R}^k$. Thus all appearances $a \in \mathbb{R}^k$ of the unmodelled collective should be known.

B. Building a Model with the Camera Network

In the previous section we saw that in order to evaluate the functional (1), a model $M(a)$ of the unmodelled collective $O_u(a) \subset \mathbb{U}$ is required. To obtain such a model, we impose a camera network $\mathcal{N}$ consisting of $n \in \mathbb{N}$ Cameras. Each camera can be placed and orientated with a setting $\mathbb{E} = (\mathbb{U} \times [-\pi, \pi] \times [-\frac{\pi}{2}, \frac{\pi}{2}])$. Here, the first term corresponds to the position of the camera whereas the second and third denote the angles 'yaw' and 'pitch' respectively. For simplicity of exposition, we exclusively considered circular cones in our implementation which allowed us to neglect the angle 'roll' as a degree of freedom in the setting of a single camera. Hence, each camera exhibits five degrees of freedom, three for the *position* and two for its *orientation*.

Thus, each camera can be regarded as a tupel $(e, p) \in \mathbb{E} \times \mathbb{U}$ whereas its produced output regarding the parameter $a \in \mathbb{R}^k$ of unmodelled collective is a function

$$\begin{aligned} \kappa : (\mathbb{E} \times \mathbb{U} \times \mathbb{R}^k) &\to \mathbb{V} \\ (e, \ p, \ a) &\mapsto \kappa_{e,a}(p) \end{aligned}$$

that is - given the setting $e \in E$ and the appearance of the unmodelled collective $a \in \mathbb{R}^k$ - each point $p \in \mathbb{U}$ is mapped onto a sensor value $v \in \mathbb{V}$ where

$$\mathbb{V} := \{\text{free}, \text{occupied}, \text{undetectable}\}.$$

This set is adjusted to the evaluation of the network's images by the change detection method (e.g. background subtraction). The sensor value $\kappa_{e,a}(p)$ of a point $p \in \mathbb{U}$ is *free* if this point is perceived as not part of the unmodelled collective. The value *occupied* resembles the possibility that the point could be part of the unmodelled collective (i.e. the point might be occupied by the collective). If the sensor cannot make the decision, e.g. this is the case for cameras that cannot 'see' behind walls, the value is *undetectable*. Obstacles like walls will be discussed in Section III-C. To obtain the values of set $\mathbb{V}$ one could apply the method of background subtraction, which is discussed in [9] elaborately. Although our method is not restricted to a pixel model which is considered in [9], the idea of this work remains the same. Thus, we will only provide the prior formulization of the values, as to explain their role in building the model of an unmodelled collective.

According to the definition of the set $\mathbb{V}$, all cameras split the set $\mathbb{U}$ into three different subsets:

$$\begin{aligned} \mathbb{P}_{\text{f}}(e, a) &= \{u \in \mathbb{U} \mid \kappa_{e,a}(u) \mathrel{\widehat{=}} \text{'free'}\} \\ \mathbb{P}_{\text{oc}}(e, a) &= \{u \in \mathbb{U} \mid \kappa_{e,a}(u) \mathrel{\widehat{=}} \text{'occupied'}\} \\ \mathbb{P}_{\text{nd}}(e, a) &= \{u \in \mathbb{U} \mid \kappa_{e,a}(u) \mathrel{\widehat{=}} \text{'undetectable'}\} \end{aligned}$$

We state here without proof that we have constructed these parts to be a pairwise disjoint conjunction of $\mathbb{U}$, i.e.

$$\mathbb{U} = \mathbb{P}_{\text{f}}(e, a) \cup \mathbb{P}_{\text{oc}}(e, a) \cup \mathbb{P}_{\text{nd}}(e, a)$$

with $\mathbb{P}_{\text{f}}(e, a) \cap \mathbb{P}_{\text{oc}}(e, a) = \mathbb{P}_{\text{f}}(e, a) \cap \mathbb{P}_{\text{nd}}(e, a) = \mathbb{P}_{\text{oc}}(e, a) \cap \mathbb{P}_{\text{nd}}(e, a) = \emptyset$ hold.

The unmodelled collective $O_u(a)$ cannot be situated inside $\mathbb{P}_{\text{f}}(e, a)$, all we know is

$$O_u(a) \ \subset \ \mathbb{P}_{\text{oc}}(e, a) \cup \mathbb{P}_{\text{nd}}(e, a) \ = \ \mathbb{U} \backslash \mathbb{P}_{\text{f}}(e, a).$$

Since this inclusion holds for the parameter $a \in \mathbb{R}^k$ and one camera with settings $e \in \mathbb{E}$, obviously the following is true if we consider a camera network $\mathcal{N}$ consisting of n cameras with settings $e_i, \ i = 1, \ldots, n$:

$$\begin{aligned} O_u(a) \ &\subset \ (\mathbb{U} \backslash \mathbb{P}_{\text{f}}(e_1, a) \cap \ldots \cap \mathbb{U} \backslash \mathbb{P}_{\text{f}}(e_n, a)) \\ &= \ \mathbb{U} \backslash \big(\mathbb{P}_{\text{f}}(e_1, a) \cup \ldots \cup \mathbb{P}_{\text{f}}(e_n, a)\big) \end{aligned}$$

Note that this set is already a good approximation of the unmodelled collective if we considered the entire set $\mathbb{U}$. However, as we only monitor the surveillance area $\mathbb{S}$, we define the desired model $M(a)$ of the unmodelled collective $O_u(a)$ as the intersection with the set $\mathbb{S}$, i.e.,

$$\begin{aligned} M(a) &\equiv M(a, e_1, \ldots, e_n) \\ &:= \mathbb{S} \cap \Big(\mathbb{U} \backslash \big(\mathbb{P}_{\text{f}}(e_1, a) \cup \ldots \cup \mathbb{P}_{\text{f}}(e_n, a)\big)\Big) \end{aligned} \tag{2}$$

This is the basic model that can be used to calculate Formula (1). In the following, we will extend our setting to incorporate a time dependency and to cover for different types of obstacles.

C. Adding Time and Obstacles

So far, we have only considered a static scene to be analyzed. Motivated by moving objects, we extend our setting by introducing a time dependency to the process under surveillance. Therefore, we declare the time interval of interest $I = [t_0, t_*]$, in which t_0 denotes the moment the reference image is taken and t_* corresponds to the last instant the surveillance area ought to be observed. Thus, the unmodelled collective $O_u(a(t))$, its probability distribution $\mathcal{P}(a(t))$ and its approximation $M(a(t)) \subset \mathbb{S}$ as well as the set of critical points $C(t)$ change in time $t \in I$. As a simple extension of (1) we obtain the time dependend error functional

$$\int\limits_{t_0}^{t_*} \int\limits_{a(t)\in\mathbb{R}^k} \Big[d(C(t), O_u(a(t))) - d(C(t), M(a(t), e_1, \ldots, e_n))\Big]^2 \, d\mathcal{P}(a(t))dt \tag{3}$$

In a second step, we add some more details to the scene under surveillance. To this end, we specify several categories and properties of objects $O \subset \mathbb{S}$, which we are particularly interested in and which affect the reconstruction of the current scene. Right from the beginning we have considered unmodelled objects. In contrast to modelled objects, these objects need to be reconstructed in order to track them. In the following, we additionally distinguish objects based on the characteristical behavior "static/dynamic", "target/obstacle" and "rigid/nonrigid", neglecting those objects that cannot be noticed by the sensors (like a closed glass door for cameras without distance sensor).

We define a *target* $T \subset O$ of a sensor network as an object which ought to be monitored and in our case reconstructed. An object $B \subset O$ which is not a target is called *obstacle*. Furthermore, we distinguish obstacles based on their physical character: An obstacle B features a *rigid* nature (like furniture), if the inpenetrability condition $T \cap B^r = \emptyset$ holds, and is denoted by the index r in B^r.

The method proposed in [9] constructs a visual hull of an object by background subtraction, i.e. via change detection. In context of change detection methods another characteristical behavior of objects is relevant: A *static object* is an object $O_s \subset O$ which is known to affect the given sensors in the same way at any time. If this is not the case, it is called *dynamic*, which we indicate by adding a subscript and a time dependency $O_d(t)$. More specifically, within the proposed background subtraction method the value of each pixel of a current image is subtracted from its counterpart within the reference image which has been taken beforehand. Thus, any change (like size/color/location) occuring after the reference image has been taken leaves a mark on the subtracted image, i.e., if the scene consists of static objects only, then the subtracted image is blank. For this reason, static objects must be placed in the scene before the reference picture is taken, and dynamic objects must not.

Within the rest of this work, we consider all unmodelled objects to be reconstructed, i.e. in (3) we have

$$O_u(a(t)) := T(a(t)) \tag{4}$$

Consequently, the unmodelled collective and its distance to the critical points are dynamic targets. Thus, we always consider an obstacle to be a *modelled* obstacle since all our unmodelled objects are targets. Furthermore, all obstacles are considered rigid. To formalize the human-robot-scene let $B_s^r \subset O$ and $B_d^r(t) \subset O$ be the *collective of static* and *of dynamic obstacles* with time $t \in [t_0, t_*]$, respectively. We incorporate these new aspects into the model of the unmodelled collective in (3) by intuitively extending our notation to

$$M(a(t), e_1, \ldots, e_n) := M(a(t), e_1, \ldots, e_n, B_s^r, B_d^r(t)). \tag{5}$$

Last – as a robot is a dynamic obstacle in addition to a security thread (f.e. when moving too fast) – we define the critical points in (3) as the collective of dynamic obstacles

$$C(t) := B_d^r(t). \tag{6}$$

Note that there are dynamic obstacles next to dynamic targets i.e. the unmodelled collective. Thus a dynamical obstacle could easily be regarded as an object of the unmodelled collective since both evoke akin reactions of the change detection method. In our approach the obstacles are fully modelled and thus define a target free zone since they are physical obstacles. Still, inaccuracies of the accidental change detection leave fragments outside the dynamic obstacle, in our case outside the critical points. As a consequence the required distance between critical points and target is reduced to zero. Publication [9] solves this issue by introducing plausibility checks, in which predicates that characterize the target (like volume, height, etc.) are used to sort out the fragments.

In conclusion, our aim is to solve the problem

$$\begin{gathered}\text{Minimize (3)}\\ \text{using definitions (4), (5) and (6)}\\ \text{subject to } e_1, \ldots, e_n \in E\end{gathered}$$

i.e., to compute the optimal positions of n cameras with settings $e_1, \ldots, e_n$ such that the measurement error is minimized.

IV. Aspects of Optimization

There are various ways to compute Equation (3) referring to: Representing the model, solving the integral and solving the optimization, as can be seen further on.

A. Discretization of time and distribution

We would at first like to state that the distance $d(C, M(a, \ldots 3))$ between the model and another set does not need to be continuous at every appearance a even if the distance $d(C, O_u(a))$ to the unmodelled collective is continuous at a. This point can also be made for Equation (3) but we stick to Equation (1) for reasons of simplicity. Such a case is illustrated in Fig. 2. As the original unmodelled collective $O_u(a)$ of the appearance $a \in \mathbb{R}^k$ does not necessarily need to be convex or even connected, given the settings $e_i, i = 1, \ldots, n \in E$, the unfree parts of the sensors $\mathbb{U} \backslash \mathbb{P}_f(e_i, a)$ do not need to be connected, either. The model is constructed of an intersection of these parts (see Equation (2)). But, as intersections of disconnected parts do not need to be continuous on $a \in \mathbb{R}^k$ (e.g. referring to Hausdorff–metrics), the distance $d(C, M(a, \ldots))$ between the model and another set does not need to be continuous at every appearance a.

Since only integrals with continuous integrants can generally be calculated as a whole or else need to be splitted, such a discontinuous function becomes a problem when being an integrant as of Formula (1). In our case a point of discontinuity of the distance as a function of a cannot be derived easily, as it would have to be extracted from an individual nonrelated analysis depending not only on a or t but also on the sensor settings $e_i, i = 1, \ldots, n$. While

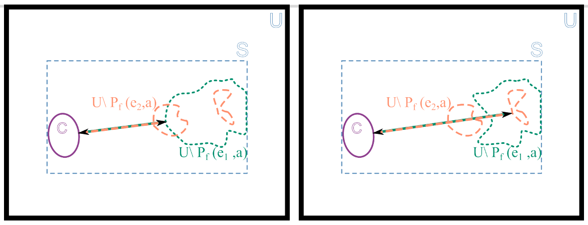

Fig. 2. Discontinuity of the distance between perilous points C and the approximated model, consisting of and intersection the nonfree part of camera 1, $\mathbb{U}\backslash\mathbb{P}_{\mathrm{f}}(e_1, a)$(green), and the nonfree part of camera 2, $\mathbb{P}_{\mathrm{f}}(e_2, a)$(orange).

in simple cases this is possible, we spare such an altering analysis by discretizing appearance and time. Here, just the $l = 1, \ldots, L$ most important appearances of the unmodelled collective $a_l \in \mathbb{R}^k$ and $h = 1, \ldots, H$ most important time steps $t_h \in [t_0, t_*]$ with $t_0 = t_1$ and $t_* = t_H$ and with their weights

$$\omega_{l,h} = \mathcal{P}(a_l(t_h)) \in [0, 1]$$

are modelled. Accordingly, the following weighted sum approximates the integral of Formula (3):

$$\begin{aligned} Err_{L,H}(e_1, \ldots, e_n) \quad &= \qquad (7) \\ \sum_{h=1}^{H}\sum_{l=1}^{L} \quad \omega_{l,m}\cdot \quad &\Big[d\big(C(t_h), O_u(a_l(t_h))\big) \\ - \quad &d\big(C(t_h), M(a_l(t_h), e_1, \ldots, e_n, t_h)\big)\Big]^2 \end{aligned}$$

B. Discretizing space by voxels

The next challenge – building an intersection of (free-form) solids – has claimed to be subject of discussion for more than a quater of a century and still is an issue of recent investigations. The publication [22] describes three main areas of solving this issue depending on their representation, each going with pros and cons.

Solids represented by polygonial meshs can be intersected by exact arithmetic and intervall computation, checking surface membership afterwards. The major concerns of this approach are robustness and efficiency (e.g., while intersecting two tangetially connected polyhedra/polygons inside-out facettes are computed).

Approximate methods (e.g. applying exact methods to a rough mesh of solids and refine the result) exist for meshs, too. Robustness problems (constructing breaks in the boundary) are in this case compensated by time consuming perturbation methods or interdependent operations which prevent parallel computing.

There are also techniques for solids transfered to image space (ray representation). While many of these mainly help rendering rather than evaluating the boundary, there are some that can be applied to intersection purposes (Layered Depth Image). Unfortunately, when computing these representations back into meshs many geometric details are destroyed.

Loosing geometric details is also the case for volumetric approaches. Converting surfaces with sharp corners and edges into volumetric data (like voxels) and not loosing data for reconstruction purpose is a challenging task even with oversampling. This also holds true for a voxel representation, but voxels on the other hand are easily obtained and robustly being checked by boolean operations. In addition to that we need a data structure, distances and volumes which are calculated easily, properties which are ensured for voxels. For these reasons our approach uses a voxel based model which is obtained by boolean operations on the free parts of the sensor.

C. Optimization method

After having evaluated existing solutions by plugging them in the objective function of a problem, the solver of an optimization problem is a strategy to improve solutions until an optimum of the objective function is reached. To choose a suitable solver for the specified problem, there are different characteristics of the objective function $Err(e_1, \ldots, e_n)$ that need to be considered.

At first, we associate the cone of a camera subtracted from the surveillance area as the 'undetectable'-part of this camera, depending on the setting e of the camera. Remember that the undetectable area could be part of the model of the unmodelled collective. Now imagine the cone rotating in 'yaw'-direction continuously. One can easily see that the distance between any given point of the surveillance area and this cone is not convex in e (as an exception, the chosen point can be included in the 'undetectable'-part and the distance is therefore 0 for all e).

The second characteristic to be discussed is the discontinuity of $Err(e_1, \ldots, e_n)$ with respect to e. Due to the voxel based model distances are only evaluated to a finite set of points. When calculating the distances we need to jump from one point to the next even if settings are just altered gradually. Thus, the objective function is discontinuous and constant in between these discontinuities. Even if we used a non–voxelbased model, discontinuities would appear due to the intersections of disconnected parts mentioned in Section IV-A.

The objective function's properties complicate the search for a suitable solver. As elucidated in standard references on nonlinear optimization like [15], most algorithms take advantage of a characteristical behavior like convexity, differentiability or at least continuity which cannot be guaranteed in our case. This applies to all determinisic solvers for nonlinear programs such as the Sequential Quadratic Programming, all kinds of local search algorithms (Downhill-Simplex, Bisection, Newton, Levenberg-Marquard etc.) and many others. Moreover, the problem cannot be transformed to a standard form of solvers like branch-and-bound, decompositions, cutting planes or outer approximation. This leaves us with non-deterministic, e.g., stochastic solvers. We have chosen the method MIDACO which is based on the ant-colony algorithm and samples solutions randomly where they appear to be most promising, see [20,21] for details.

D. Complexity

The solver is an iteration which generates a tuple of settings e_i, $i = 1, \ldots, n$ (one setting for each camera) within each iteration step stochastically, based on knowledge of previous generations. Given these settings the model, the distances and the objective function consisting of the weighted sum given in Equation (8) are evaluated. This continues until a stopping criteria is fulfilled. In order to compute the complexity of the method, assume that upon termination the I-th iteration step has been reached. The process of obtaining the objective value of Formula (8) is only implemented in a basic version, whereas for the given tuple of settings e_i all of the H time steps and L appearances are to be evaluated to test all of the r voxels whether they are included in the intersection in question. The intersection test uses all of the f_u^{max} facets of the unmodelled collective as well as most of f_s static facets and f_d^{max} dynamic facets. Summing up these components give us the complexity

$$\mathcal{O}\Big(\ I \cdot r \cdot \{\ n\ f_s\ +\ H\ (n + L)\ f_d^{max}\ +\ H\ L\ n\ f_u^{max}\ \}\ \Big)$$

of the method.

V. Experiments

Since we use a stochastical solver on the non-convex problem of camera configuration, the obtained solutions (i.e. tuple of settings) most likely differ from one another although the same objective value (i.e. deviation of distances) might have been found. Therefore, we ran groups of 20 solver calls with the same parameters to perceive the average outcome. One examination consists of a few groups of test runs which only differ in one parameter. We made examinations about changing resolutions, facets, objects, amount of events, amount of cameras and starting point. As long as there are no other assumptions the basic setup stated in Tab. I is used.

modelled part of the scene	dimension/amount
surveillance area S:	cuboid $4m \times 3m \times 3m$
voxel resolution:	$(16 \times 12 \times 12)$
critical points:	all point inside the dynamic collective
static collective:	8 facets at 2 objects
dynamic collective:	24 facets at 6 objects in 2 timesteps
unmodelled collective:	24 facets at 6 objects in 3 events (of distrib.)
camera placement:	6 cameras all over the surveillance area
starting solution:	cameras are placed and orientated randomly all over S
stop criteria:	maximal time limit 3h optimization tolerance $(diagon.o.voxel)^2$

TABLE I
BASIC SETUP, WHICH IS USED IF NO OTHER ASSUPTIONS ARE MADE

We additionally assumed that the dynamic collective is also considered to be the set of critical points. Thus, we were able to model a robot (dynamical object and critical points) spinning too fast in direction of a human (unmodelled object).

Furthermore, Tab. II contains all test parameters and their ranges. The aim of this section is to summarize all the examinations defined in Tab. II and, in particular, to answer the following central questions: Can the desired optimization tolerance be satisfied in time, ie. will the target be approximated as accuratly as needed? How many iteration cycles are needed? What is the operating time of one cycle, of each iteration step and the components of one step? What is the highest memory consumption?

modelled part of the scene	alterations
voxel res.:	$(16 + 4i \times 12 + 3i \times 12 + 3i)$ for $i = 0, 1, 2, 3, 4, 5$
static coll.:	$8 + 60i$ facets for $i = 0, \ldots, 5$ at 2 obj.
	$6 + 4i$ facets at $2 + i$ objects $i = 0, \ldots, 3$
dynamic coll.:	$24 + 60i$ facets $i = 0, \ldots, 5$ at 3 obj./2 timest.
	$2 + i$ obj. $i = 0, \ldots, 3$ w. $24 + 4i$ fac./1 timest. objects placed randomly
	$i = 1, \ldots, 5$ timest. w. $2i$ obj $8i$ fac. 3 events
unmodelled coll.:	$24 + 60i$ facets $i = 0, \ldots, 5$ at 2 obj./3 events
	$2 + i$ obj. $i = 0, \ldots, 3$ w. $24 + 4i$ fac./1 event objects placed randomly
	$i = 1, \ldots, 5$ events w. $2i$ obj $8i$ fac. 3 timest.
cameras:	$i = 3, \ldots, 9$
restrictions of the settings' domain:	cameras placed only at 'ceilling'
	cameras placed only in the 'upper fourth'

TABLE II
THIS IS AN OVERVIEW OF ALL EXAMINATIONS. AN EXAMINATION CONSISTS OF A FEW GROUPS OF TEST RUNS, EACH GROUP DIFFERS ONLY IN ONE PARAMETER.

A. Hardware and Software

We implemented the optimization problem in C++ and compiled it with 'gcc' version 4.0.20050901 (prerelease) optimized with the setting '-O3' on SuSE Linux version 10.0. We have used only one of the two cores of an AMD Opteron(tm) Processor 254 with 2.8 GHz Power(dynamical from 1GHz - 2.8GHz).Further information can be taken from Tab. III and IV.

B. Optimization tolerance

As a second stopping criteria next to the three hour time limit we introduced the optimization tolerance, which is the maximal objective value a tuple of settings must be mapped at, for the optimization to terminate. This is desinged to depend on the length of a voxel's diagonal. In many cases the solver was able to satisfy the desired optimization tolerance in the predefined maximal time. Following exceptions have exceeded the time limit: We recorded an increasing time consumption of one iteration step (beyond linear) when gradually raising the resolution of the voxel discretization. Due to the time criterion, approaches with a resolutions of more than $24 \times 18 \times 18$ were terminated before satisfying

model name:	AMD Opteron(tm) Processor 254
cpu MHz:	1004.631
cache size:	1024kB
clflush size:	64
cache_alignment:	64

TABLE III

PART OF THE OUTPUT OF $: CAT /PROC/CPUINFO

MemTotal:	4038428kB
MemFree:	886856kB
Buffers:	431016
Cached:	2079360
SwapCached:	0kB
Active:	1431004kB
Inactive:	1138428kB
SwapTotal:	12586916kB
SwapFree:	12586916kB

TABLE IV

PART OF THE OUTPUT OF $: CAT /PROC/MEMINFO

the optimization tolerance (see Fig. 3 and 5). Increasing the number of dynamic obstacles resulted in too many iteration steps (over 160000 at most compared to less than 45000 when increasing the amount of static obstacles, cf. Fig. 4) and thus decreasing the amount of tests the optimization tolerance was satisfied for, in time, as illustrated in Fig. 6. In rare cases, a similar outcome was observed if theamount of randomly placed unmodelled objects is increased.

A combination of both occurrances – the time loss in each iteration step and the requirement of too many iteration steps – has been observed for test runs utilizing a small number of cameras (considering three cameras it was literally impossible to compute a satisfactory result, see Fig. 7). In case of the tests on dynamic obstacles and too few cameras, the model of the unmodelled collective could not be produced optimally before the maximal computing time was up. We experienced similar results for all tests concerning restrictive domains: None of the tests reached the optimization tolerance (0.046 m^2) but all of them stayed below the value 0.25 m^2. This could be a sign, e.g. that in our test setting six cameras on the ceilling cannot assimilate the unmodelled collective close enough by the model.

C. Time consumption

When raising the amount of events, time steps, facets or objects of any of the collectives we have also recorded a linearly increasing time consumption for one iteration step. Out of these, the resolution of voxel discretization and the amount of dynamic objects appear to be the most critical ones. Using more cameras, however, resulted in a lower time loss in one iteration step in our range of camera amounts (for three cameras we required about 315 ms on average whereas for nine cameras ca. 170 ms were needed). Of course, this effect can only last until optimization tolerance is satisfied (i.e. the model assimilates the unmodelled collective as accurat as needed), and hence time consumption will slope up when using a greater amount of cameras.

Without giving a detailed explanation about the way one iteration step is calculated with our test setting's camera network, we would like to state that extending the amount of facets, cameras and refining the voxel resolution enlarges time consumption of the intersection test. However, no intersection test except for those with refined voxel resolution has exceded $15ms$ on average. The test runs with $36\times27\times27$ voxel, six cameras and 24 facets have reached an average of $50ms$. After intersecting areas the related voxels need to be combined to clusters, as to be able to check a free part's height or volume (and to compare whether it could be human). This task took about twice up to four times as long as the intersection test, a fact which is mainly due to its direct dependence on the resolution, but also due to the misshaping

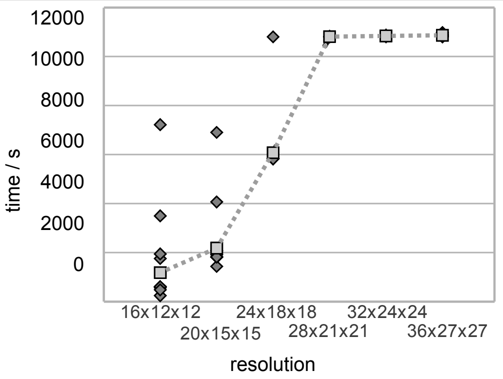

Fig. 3. Scatter plot: With refined resolution the mean (light grey squares) of the amount of iteration steps (each represented in a dark grey sqare) in one group was higher. Columns: groups of 20 iterations with different resolutions;

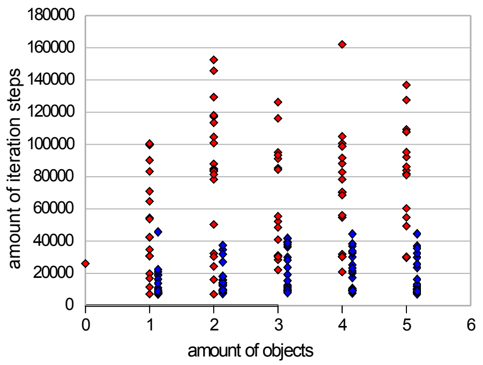

Fig. 4. Scatter plot: Dynamic obstacles (and perilious points) are complicating the iteration. Columns: groups of 20 iterations of additional dynamic objects (red) and static objects (blue)

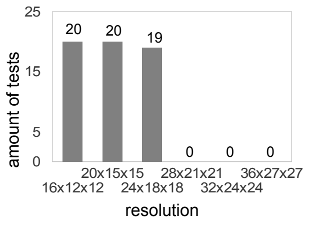

Fig. 5. Bar Chart: More refined resolution than $(24 \times 18 \times 18)$ made it impossible to satisfy the predefined optimization tolerance in time

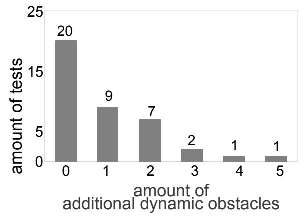

Fig. 6. Bar Chart: The more dynamic objects were spread across the surveillance area, the less test runs satisfied the desired optimization tolerance

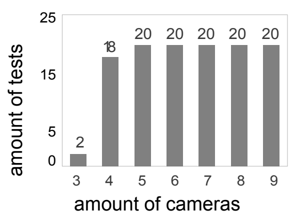

Fig. 7. Bar Chart: Five till nine cameras could contour the unmodelled collective best.

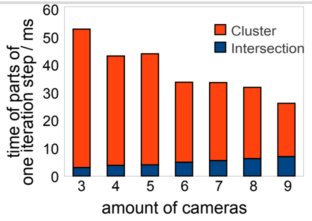

Fig. 8. Bar Chart: The total time consumption of an increased amount of cameras sloped down because the clustering (orange) weighted more than the actual intersection test (blue).

of the model (as the clustering seems to depend indirectly on the amount of cameras). As the period of an iteration step is mostly filled with intersecting and clustering, Fig. 8 also shows the decreasing time consumption while using more cameras.

D. Memory

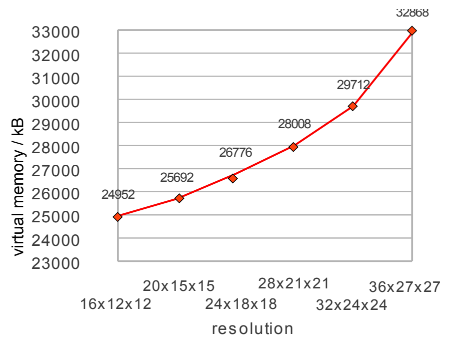

Fig. 9. Plot of increasing maximal virtual memory that was used when refining the resolution of voxels

Measurements of the maximal virtual memory while altering the resulution resulted an ascending graph (beyond linear), cf. Fig. 9. The highest demand for virtual memory was measured while testing with the resolution $36 \times 27 \times 27$ (a total of 32868 kB). The graphs concerning the maximum demand for virtal memory versus facets and amount of cameras are only ascending slowly. Both show a linear slope of about 350 kB to 450 kB in our range of parameters.

VI. Conclusion and Future Prospects

We managed to build up a camera placement optimization algorithm that computes location and orientation of a given amount of cameras inside of a specified surveillance area. Only randomly placed dynamic obstacles, too few cameras or too restricted placements and a too refined voxel resolution are a critical for this method. Apart from that we have succeeded to minimize the error made by evaluating distances to the visual hull of a given object up to the optimization tolerance. In contrast to existing results, we are able to model a surrounding area with static and moving obstacles without limiting camera positions or orientations and still evaluate distances conservatively.

Still, as to assimilate the model and the unknonwn collective even better, higher resolutions are desired. This leads to the fact that some improvements of the algorithm still need to be implemented. Following alterations of the algorithm may lead to an improved time consumption: First of all, it is possible to parallelize the iterations of the solver as well as some intersection tests. But as the amount of iteration steps of the solver ranged in between about 500 and 160000, the first goal should be to decrease both the expected number of iteration steps as well as their variance. Placing the initial position of the cameras roughly around the surveillance area and leaving the fine tuning to the algorithm could do the trick.

Some consideration should also be paid to save many clustering and intersecting processes by leaving out unnecessary caculations. One of these calculations is the summing up $L \cdot H$ addends (the number of appearances times number of time steps), which all have to be simulated. Time loss will be minimized if cancelling the evaluation of the sum when it trespasses the current optimal value. Also, appropriate data structures like Oct-Trees and BSP-Trees for intersection and inclusion tests have not been implemented, yet, which improve the time loss during the intersection test.

References

[1] Bodor, R. *et al.*: *Optimal camera placement for automated surveillance tasks*. Journal of Intelligent and Robotic Systems, 50(3):257–295, 2007.

[2] Ercan, A., Gamal, A., and Guibas, L.: *Camera network node selection for target localization in the presence of occlusions*. In *In SenSys Workshop on Distributed Cameras*, 2006.

[3] Ercan, A. *et al.*: *Optimal placement and selection of camera network nodes for target localization*. Distributed Computing in Sensor Systems, pp. 389–404, 2006.

[4] Erdem, U. and Sclaroff, S.: *Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints*. In *OMNIVIS Workshop*. Citeseer, 2004.

[5] Fiore, L. *et al.*: *Multi-camera human activity monitoring*. J Intell Robot Syst, 52(1):5–43, 2008.

[6] Fiore, L. *et al.*: *Optimal camera placement with adaptation to dynamic scenes*. In *IEEE International Conference on Robotics and Automation*, pp. 956–961, 2008.

[7] Hartley, R.: *Estimation of relative camera positions for uncalibrated cameras*. In *Computer Vision-ECCV'92*, pp. 579–587, 1992.

[8] Holt, R. *et al.*: *Summary of results on optimal camera placement for boundary monitoring*. In *Proceedings of SPIE*, vol. 6570, p. 657005, 2007.

[9] Kuhn, S. and Henrich, D.: *Multi-view reconstruction of unknown objects in the presence of known occlusions*. Techn. rep., Universität Bayreuth, Angewandte Informatik III (Robotik und Eingebettete Systeme), 2009.

[10] Luhmann, T. *et al.*: *Close Range Photogrammetry, Principles, techniques and applications*. Whittles Publishing, 2006.

[11] Mittal, A.: *Generalized multi-sensor planning*. European Conference on Computer Vision 2006, pp. 522–535, 2006.

[12] Mittal, A. and Davis, L.: *Visibility analysis and sensor planning in dynamic environments*. European Conference on Computer Vision 2004, pp. 175–189, 2004.

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

[14] Murray, A. *et al.*: *Coverage optimization to support security monitoring*. Computers, Environment and Urban Systems, 31(2):133–147, 2007.

[15] Nocedal, J. and Wright, S.J.: *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer-Verlag, New York, second ed., 2006.

[16] OLAGUE, G.: *Design and simulation of photogrammetric networks using genetic algorithm*. In *Proceedings of the 2000 Meeting of the American Society for Photogrammetry and Remote Sensing (ASPRS 2000)*, 2000.

[17] OLAGUE, G. and DUNN, E.: *Developement of a practical photogrammetric network design using evolutionary computing*. The Photogrammetric Record, 22:22–38, 2007.

[18] OLAGUE, G. and MOHR, R.: *Optimal 3d sensors placement to obtain accurate 3d points positions*. In *Proceedings of the Fourteenth International Conference on Pattern Recognition*, vol. 1, pp. 16–20, 1998.

[19] OLAGUE, G. and MOHR, R.: *Optimal camera placement for accurate reconstruction*. Pattern Recognition, 35:927–944, Januar 1998. Publisher: Elsevier.

[20] SCHLÜTER, M., EGEA, J., and BANGA, J.: *Extended ant colony optimization for non-convex mixed integer nonlinear programming*. Comput. Oper. Res, 36(7):2217–2229, 2009.

[21] SCHLÜTER, M. and GERDTS, M.: *The oracle penalty method*. Springer Science+Budiness Media, LLC, 30, 2010.

[22] WANG, C.C.: *Approximate boolean operations on large polyhedral solids with partial mesh reconstruction*. Transactions on Visualization and Computer Graphics, NA, 2010.

[23] YANG, D. *et al.*: *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.