Optimal camera placement to measure distances regarding static and dynamic obstacles

Maria L. Hänel*, Stefan Kuhn and Dominik Henrich

Chair of Applied Computer Science III,
University of Bayreuth,
D-95445 Bayreuth, Germany
Email: maria.haenel@uni-bayreuth.de
Email: stefan.kuhn@uni-bayreuth.de
Email: dominik.henrich@uni-bayreuth.de
*Corresponding author

Lars Grüne

Chair of Applied Mathematics,
University of Bayreuth,
D-95445 Bayreuth, Germany
Email: lars.gruene@uni-bayreuth.de

Jürgen Pannek

Faculty of Aeronautics,
University of the Federal Armed Forces Munich,
85577 Munich/Neubiberg, Germany
Email: juergen.pannek@unibw.de

Abstract: In modern production facilities industrial robots and humans are supposed to share a common working area. To avoid collisions, the distances between objects need to be measured conservatively, which can be done by a camera network. To estimate these distances, unmodelled objects, e.g. an interacting human, need to be modelled and distinguished from pre-modelled objects, like workbenches or robots, by image processing such as the background subtraction method. The quality of such an approach massively depends on the position and orientation of each camera. Of particular interest in this context is the error minimisation of the above mentioned distance determined by image processing. Here, we formulate this minimisation as an abstract optimisation problem. Moreover, we state various aspects on the implementation, e.g. reasons for the selection of a suitable optimisation method, analyse the complexity of the proposed method and present a basic version used for extensive experiments.

Keywords: close range photogrammetry; optimisation; camera network; camera placement; error minimisation; obstacles; distance measurement; optimal placement; static obstacles; dynamic obstacles; optimal image processing; position and orientation; sensor network; occlusions.

Reference to this paper should be made as follows: Hänel, M.L., Kuhn, S., Henrich, D., Grüne, L. and Pannek, J. (XXXX) 'Optimal camera placement to measure distances regarding static and dynamic obstacles', *Int. J. Sensor Networks*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Maria L. Hänel received her Diploma in Technomathematik (Major: Mathematics, Minors: Computer Science, Engineering) in April 2010 from the University of Bayreuth, Germany. There, she has been working as a research associate at the Chair of Applied Computer Science III (Robotics and Embedded Systems), since. Her research interests lie in optimisation strategies and the simulation of sensor networks, imaging sensors in particular, and in image processing and its applications.

Stefan Kuhn achieved his Diploma in Computer Science in 2004 from the University of Kaiserslautern, Germany. Since 2004 he is a research associate at the chair for Applied Computer Science III (Robotics and Embedded Systems) in the Institute for Computer Science at the University of Bayreuth, Germany. His research interests include parallel manipulators, room surveillance, multiple view reconstruction and parallel computing.

Dominik Henrich achieved his Diploma in Computer Science and PhD from the University of Karlsruhe, Germany, in 1991 and 1994, respectively. From 1996 to 1999, he was Lecturer at the Informatics Faculty of the University of Karlsruhe. From 1999 to 2003, he headed as professor

the research group 'Embedded Systems and Robotics (RESY)' at the Informatics Faculty of the University of Kaiserslautern. Since August 2003 he holds the chair for Applied Computer Science III at the University of Bayreuth. His research interests are collision detection, motion planning, room surveillance, self-adapting robots, sensor-based manipulation, intuitive robot programming, human/robot cooperation, and robot-assisted surgery.

Lars Grüne is Professor and Head of the Chair of Applied Mathematics at the University of Bayreuth, Germany. He received his Diploma and PhD in Mathematics in 1994 and 1996, respectively, from the University of Augsburg. From 1997 until 2002 he worked at the J.W. Goethe University in Frankfurt/M. He is Editor in Chief of the *Mathematics of Control, Signals and Systems*, member of the Steering Committee of the International Symposium on Mathematical Theory of Networks and Systems (MTNS) and of the Managing Board of the International Association of Applied Mathematics and Mechanics (GAMM). His research interests lie in the area of mathematical systems and control theory, in particular on numerical and optimisation based methods for nonlinear systems.

Jürgen Pannek is currently a research associate at the Faculty of Aeronautics at the University of the Federal Armed Forces Munich, Germany. He received his PhD in Mathematics from the University of Bayreuth in 2009 and his Diploma in Business Mathematics from the University of Bayreuth in 2005. From 2010 to 2011 he was University Associate at Curtin University of Technology Perth and The University of Western Australia. His research interests include nonlinear model predictive control, networked control systems and decentralised control as well as system's theory and optimisation methods.

1 Introduction

Humans and animals use their senses to work and live in a shared environment, mostly without accidents. They use several senses simultaneously in order to prepare for critical events, and thus to avoid collisions. In this work we consider situations in which humans and machines interact in a shared working area. Unfortunately, a machine lacks senses and thus the ability to avoid collisions on its own. To compensate for this issue, we use a sensor network to compute coarse visual hulls of humans within the area under surveillance. Then the distance between machines and humans is evaluated conservatively. Typically, such a network is considered to be reliable if one can ensure that every corner of the room can be watched, every trail can be followed and or every object can be reconstructed correctly. The proposed work answers an inverse question: Given a particular sensor fusion method to generate a map of the area of interest and compute the corresponding distances, what are the optimal positions and orientations of the sensors? For such an optimal configuration the visual hull is computed more accurately than in the case of non-optimal sensor positions. Thus, the probability of false alarms is reduced.

The paper is organised as follows: A brief review on previous results is given concerning the predescribed distance measurement. Then, we show how an unmodelled object (human) can be contoured by a 3D background subtraction method using multiple cameras. We extend this scheme to cover both static and dynamic obstacles, which partly occlude the vision of the sensors. In Section 3, we rigorously formulate the problem of minimising the error made by using the associated model instead of the original collective of unmodelled objects. In Section 4, we discuss various difficulties that arise for a solution method, such as the evaluation of the intersection of visibility cones, and also give an outline of concepts to work these issues. In the final Sections 5 and 6 we analyse the complexity of our (basic) implementation by a series of numerical experiments and conclude the paper by giving an outlook on methods to further improve the proposed method.

2 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 are investigations about how to position and orientate cameras subject to observing a maximal number of surfaces (e.g. Holt et al., 2007) and different courses of action (Bodor et al., 2007; Fiore et al., 2008a; Fiore et al., 2008b) as well as maximising the volume of the surveillance area (e.g. Murray et al., 2007) or the number of objects (e.g. Mittal and Davis, 2004; Mittal, 2006; Mittal and Davis, 2008). Another common goal in this context is to be able to observe all items of a given set, but minimise the amount of cameras in addition to obtain their positions and orientations (e.g. Erdem and Sclaroff, 2004; Mittal and Davis, 2004; Mittal, 2006; Mittal and Davis, 2008). 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 by Luhmann et al. (2006) 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 minimises the error of observed and reconstructed items. Often the phrase 'Photogrammetric Network Design' is used to express minimising the reconstruction error for several (three-dimensional) points. The default assumption in this

context, however, is that no occlusions occur (for details cf. Hartley, 1992; Olague and Mohr, 1998a; Olague and Mohr, 1998b; Olague, 2000; Olague and Dunn, 2007). Optimally localising an entire object which is not occluded is an assignment treated by Ercan et al. (2006a, 2006b). 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 Hartley (1992) two cameras are used] or their position and orientation. Considering the latter, known approaches are the viewing sphere model given by Olague and Mohr (1998a) and Olague and Mohr (1998b) or the idea of situating all cameras on a plane and orientating them horizontally (cf. Ercan et al., 2006a; Ercan et al., 2006b).

In contrast to these approaches, we discuss an approach which is related to Vecherin et al. (2011) and deals with optimising 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 Yang et al. (2004) assume that minimising the occurring occlusions of solids also reduces and thus specifies their possible locations. However, neither obstacles nor opening angles other than π are discussed by Yang et al. (2004) and additionally the orientation of the camera is neglected as a variable since it is simply orientated towards the object. In the work of Ercan et al. (2006a, 2006b) static obstacles are considered, but a subset of cameras is chosen out of a pre-installed set instead of rearranging the whole set.

Since we are not interested in optimising the quantity of observed objects but in 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 optimise the positions and orientations of a fixed amount of cameras as to minimise 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 occurring 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 conservatively.

3 Visibility analysis

Within this section, we show how to design the objective function on the cameras position and orientation. Thus, we successively build up a mathematical representation of the optimisation problem. We start off by defining the critical area as well as the unmodelled objects which are to be reconstructed and corresponding abstract models in Section 3.1. This will allow us to formally state the objective function, which we aim to minimise. In the following Section 3.2, 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 parametrise the model of the object we wish to reconstruct. Additionally, this tuple of degrees of freedom will serve as an optimisation variable in the minimisation problem stated in Section 3.3. To cover all possible scenarios, the problem is extended by incorporating both static and dynamic obstacles as well as an evolving time component.

3.1 Formalising the problem

Let $\mathbb{U} \subset \mathbb{R}^3$ be a spatial 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 as humans or robots, are monitored.

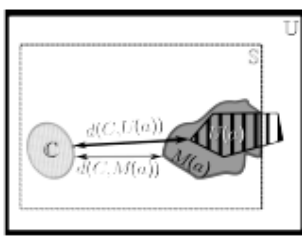
For the moment, we neglect obstacles completely. We just distinguish two types of objects to explain the basic idea of reconstructing an object by the means of a camera network: If a detailed model of an object exists, describing its appearance, like location, shape and colour or else, the object is 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 trajectory, 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 $U(a) \subset \mathbb{S}$ be a complete set of points included in one or more unmodelled objects, depending on the appearance of unmodelled objects motivated above, specified by the parameter $a \in \mathbb{R}^k$. We refer to these objects as unmodelled collective. Since automatically 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

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

has to be computed, conservatively and security measures need to be taken if the unmodelled collective $U(a)$ approaches the critical points C. Here $d(\cdot,\cdot)$ denotes a standard distance function. If the exact set $U(a)$ 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 $U(a)$, we need to approximate the model $M(a) \subset \mathbb{S}$ of the unknown collective, see Figure 1. As a consequence this model also needs to be conservative.

Figure 1 Surveillance area $\mathbb{S}$: Distance between critical points C and unmodelled collective (stripes), and distance to the approximated model (grey)



Note that, for now $M(a)$ is an abstract approximation of $U(a)$ 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 3.2 for details. Still, the abstract approximation allows us to formalise our overall task, i.e. to minimise the difference between the approximation based distance $d(C, M(a))$ and the real distance $d(C, U(a))$. Taking the assumed distribution of the parameter a into account, we aim to minimise the functional.

$$\int_{a\in\mathbb{R}^k} \left[d\left(C,U(a)\right) - d\left(C,M(a)\right) \right]^2 d\mathcal{P}(a) \tag{1}$$

Note that for the optimisation 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.

3.2 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 $U(a) \subset \mathbb{U}$ 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 $\mathbb{E} = \left(\mathbb{U} \times [-\pi, \pi] \times \left[-\frac{\pi}{2}, \frac{\pi}{2} \right] \right)$. 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*.

With a chosen appearance $a \in \mathbb{R}^k$ of unmodelled collective, each camera can be specified by the setting $e \in \mathbb{E}$, whereas its produced output is a function of $p \in \mathbb{U}$

$$\kappa : \left(\mathbb{E} \times \mathbb{U} \times \mathbb{R}^k \right) \to \mathbb{V}$$

$$(e, p, a) \mapsto \kappa_{e,a}(p)$$

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:

$$\mathbb{V} := \{\text{free; occupied; undetectable}\} \tag{2}$$

This set is adjusted to the evaluation of the 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 3.3. Section 4.3 roughly describes how to obtain the values of set $\mathbb{V}$. For the moment, we will only provide the prior formulisation of the values, as to explain their role in building the model of an unmodelled collective.

According to the definition of the set $\mathbb{V}$, a camera with setting $e \in E$ splits the set $\mathbb{U}$ into three different subsets:

$$\mathbb{P}_{\text{f}}(e,a) = \left\{ p \in \mathbb{U} \,\middle|\, \kappa_{e,a}(p) \triangleq \text{'free'} \right\}$$

$$\mathbb{P}_{\text{oc}}(e,a) = \left\{ p \in \mathbb{U} \,\middle|\, \kappa_{e,a}(p) \triangleq \text{'occupied'} \right\}$$

$$\mathbb{P}_{\text{nd}}(e,a) = \left\{ p \in \mathbb{U} \,\middle|\, \kappa_{e,a}(p) \triangleq \text{'undetectable'} \right\}$$

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 $\begin{array}{l} \mathbb{P}_{\text{f}}(e,a) \cap \mathbb{P}_{\text{oc}}(e,a) = \mathbb{P}_{\text{f}}(e,a) \cap \mathbb{P}_{\text{nd}}(e,a) = \mathbb{P}_{\text{oc}}(e,a) \\ \cap \mathbb{P}_{\text{nd}}(e,a) = \varnothing \text{hold} \end{array}$.

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

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

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

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

Note that this set is already a good approximation of the unmodelled collective if we consider 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 $U(a)$ as the intersection with the set $\mathbb{S}$, i.e.

$$\begin{aligned} M(a) &\equiv (a, e_1, \ldots, e_n) \\ &:= \mathbb{S} \cap \left(\mathbb{U} \setminus \left(\mathbb{P}_{\text{f}}(e_1,a) \cup \ldots \cup \mathbb{P}_{\text{f}}(e_n,a) \right) \right) \end{aligned} \tag{3}$$

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

3.3 Adding time and obstacles

So far, we have only considered a static scene to be analysed. 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 to be $I = [t_0, t_*]$, 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 $U(a(t))$, its probability distribution $\mathcal{P}(a(t))$ and its approximation $M\left(a(t)\right) \subset \mathbb{S}$, as well as the set of critical points $C(t)$ change in time $t \in I$. As a simple extension of Formula (1) we obtain the time dependent error functional

$$\int_{t_0}^{t_*} \int_{a(t)\in\mathbb{R}^k} \delta\left(e_1,\ldots,e_n,a(t),t\right) d\mathcal{P}\left(a(t)\right) dt \tag{4}$$

with the difference between the approximation-based distance and the real distance $\delta\left(e_1,\ldots,e_n,a(t),t\right) = \left[d\left(C(t),U\left(a(t)\right)\right) - d\left(C(t),M\left(a(t),e_1,\ldots,e_n\right)\right)\right]^2$.

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 behaviour 'static/dynamic', 'target/obstacle' and 'rigid/non-rigid', 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 impenetrability condition $T \cap B^r = \varnothing$holds , and is denoted by the superscript *r* in B^r.

The method proposed by Kuhn and Henrich (2009) constructs a visual hull of an object by background subtraction, i.e. via change detection. In context of change detection methods another characteristical behaviour of objects is relevant: A *static object* is an object 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*. More specifically, within the proposed background subtraction method, the value of each pixel in the current image is subtracted from its counterpart within the reference image, which has been taken beforehand. Thus, any change (like size/colour/location) occurring 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 will use the following notation. We consider all unmodelled objects to be reconstructed, i.e. in Formula (4) we have

$$U\left(a(t)\right) := T\left(a(t)\right) \tag{5}$$

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 formalise the human-robot-scene, let $B_s^r \subset O$ and $B_s^r(t) \subset O$ be the collective *of static* and *of dynamic obstacles* with time $t \in [t_0, t_*]$, respectively. Table 1 gives an overview of the types of considered objects.

Table 1 The considered objects from the static collective B_s^r, the dynamic collective $U(a(t))$ and the unmodelled collective $B_d^r(t)$

	static		*dynamic*	
	obst.	*target*	*obst.*	*target*
Modelled	B_s^r	–	$B_d^r(t)$	–
Unmodelled	–	–	–	$U(a(t))$

We incorporate these new aspects into the model of the unmodelled collective in Formula (4) by intuitively extending our notation to

$$M\left(a(t),e_1,\ldots,e_n\right) := M\left(a(t),e_1,\ldots,e_n,B_s^r,B_d^r(t)\right) \tag{6}$$

Finally, as a robot is a dynamic obstacle in addition to a security thread, e.g. when moving too fast, we define the critical points in Formula (4) as the collective of dynamic obstacles.

$$C(t) := B_d^r(t) \tag{7}$$

In conclusion, our aim is to solve the problem *Minimise Formula (4) using Formula (5); (6) and (7) subject to* $e_1,\ldots e_n \in E$, i.e. to compute the optimal positions of n cameras with settings $e_1,\ldots,e_n$ such that the measurement error is minimised.

As an afterthought, 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 define a target free zone since they are rigid obstacles. Still, inaccuracies of the change detection method leave fragments of occupied space outside the dynamic obstacle, in our case outside the critical points. These fragments are confused with the target if one uses the model of Formula (6). As a consequence, the required distance between critical points and target is reduced to zero, accidentally. Kuhn and Henrich (2009) solve this issue by introducing plausibility checks, in which predicates that characterise the target (like volume, height, etc.) are used to sort out these fragments.

4 Aspects of optimisation

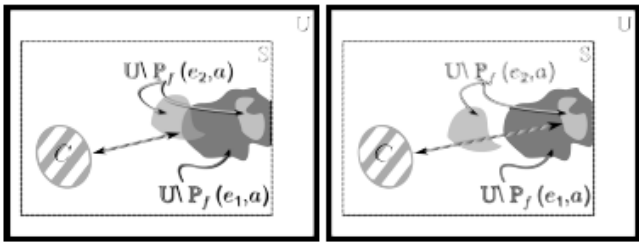
There are various ways to compute equation (4). In any case, an approximated model $M(a(t))$, made of the intersections of equation (3), is essential. Therefore, the representation of the

model needs to be chosen and the scene needs to be simulated, including views of the cameras. Next to these issues, we answer the question of how to solve the integral and discuss several difficulties in the optimisation in this section.

4.1 Integrals – discretisation of time and distribution

We would first like to state that the distance $d(\cdot, M(a,\ldots))$ between the model and another set does not need to be continuous at every appearance a, even if the distance $d(\cdot, U(a))$ between the unmodelled collective and the same set is continuous at a. Such a case is illustrated in Figure 2. This point can also be made for equation (4), but we stick to equation (1) for reasons of simplicity. As the original unmodelled collective $U(a)$ of the appearance $a \in \mathbb{R}^k$ neither needs to be convex nor even connected, given the settings $e_i \in E, i = 1,\ldots,n$, the unfree parts of the sensors $\mathbb{U} \setminus \mathbb{P}_f(e_i, a)$ do not need to be connected, either. The model is constructed of an intersection of these parts (see equation (3)). But, as intersections of disconnected parts do not need to be continuous on $a \in \mathbb{R}^k$ (e.g. referring to Hausdor-metrics), the distance $d(C, M(a,\ldots))$ between the model and another set does not need to be continuous at every appearance a.

Figure 2 Discontinuity of the distance between perilous points C (stripes) and the approximated model, consisting of an intersection between the non-free part of camera 1 $\mathbb{U} \setminus \mathbb{P}_f(e_1, a)$ (dark grey), and the non-free part of camera 2, $\mathbb{U} \setminus \mathbb{P}_f(e_2, a)$ (light grey)



Since only integrals with continuous integrants can generally be calculated accurately by numerical algorithms or else need to be splitted, such a discontinuous function becomes a problem when being an integrant as in Formula (1). In our case, a point of discontinuity of the distance as a function of a cannot be derived automatically, as it would have to be extracted from an individual non-related analysis depending not only on a or t, but also on the sensor settings e_i, $i = 1,\ldots,n$. While in simple cases this is possible, we spare such an altering analysis by discretising 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 $t_* = t_H$ are used together with their weights.

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

Accordingly, the following weighted sum approximates the integral of Formula (4).

$$E_{rr_{L,H}}(e_1,\ldots,e_n) = \sum_{h=1}^{H}\sum_{l=1}^{L} \omega_{l,h} \cdot \Big[d\big(C(t_h), U(a_l(t_h))\big) - d\big(C(t_h), M(a_l(t_h), e_1 \ldots, e_n, t_h)\big)\Big]^2 \quad (8)$$

4.2 The representation of the model – discretising space by voxels

The next challenge – building an intersection of (freeform) solids, has claimed to be subject of discussion for more than a quarter of a century and still is an issue of recent investigations. Depending on the representation of the solids, each going with pros and cons, Wang (2010) describes three main ways of solving this issue:

First, solids represented by polygonial meshes can be intersected by exact arithmetic and interval computation, checking surface membership afterwards. The major concerns of this approach are robustness and efficiency (e.g. while intersecting two tangentially connected polyhedra/polygons inside-out facets could be computed). Approximate methods (e.g. applying exact methods to a rough mesh of solids and refine the result) exist for meshes too. In this case, robustness problems (constructing breaks in the boundary) are compensated by time consuming perturbation methods or interdependent operations which prevent parallel computing.

Secondly, there exist techniques for solids transferred 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 these representations are transformed back into meshes, many geometric details are destroyed.

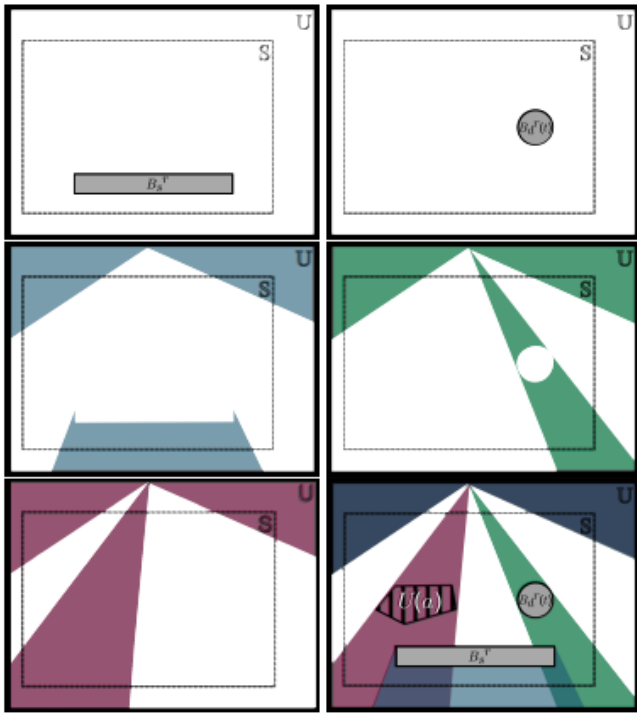
Last, loosing geometric details is also the case for volumetric approaches. Converting surfaces with sharp corners and edges into volumetric data (like voxels) and not loosing data for reconstruction purpose is a challenging task even with oversampling. This also holds true for a voxel representation, 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 of 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.

4.3 Free, occupied, undetectable – simulation of camera network views

So far we have not described how to gain the sensor values in the set $\mathbb{V}$ from Formula (2) for each camera. Kuhn and Henrich (2009) developed a change detection system which is able to cope with static and dynamic obstacles via a background-subtraction method. Although our method is

not restricted to Kuhn and Henrich's (2009) pixel model, the idea remains the same. To illustrate the idea of this method, which we adopted for our purposes, we show different stages of a simulation of a single camera in Figure 3: The pictures of the first row show the inclusions of static (left) and dynamic obstacles (right). From this we can conclude that a voxel is not part of the model of Formula (6) if the voxel is inside a static or dynamic modelled object. In the second row to the left the undetectable part $\mathbb{P}_{nd}$ is illustrated. The coloured area on top illustrates the opening angle of a single camera and the one to the bottom illustrates the voxels which are undetectable due to static occlusion. The pictures in the second row to the right and the third row to the left indicate the occupied part $\mathbb{P}_{oc}$ due to dynamic occlusions. The fusion of all the five pictures is shown in the third row to the right. The white (not occupied or undetectable) and grey areas (known obstacles) resemble the *free part* $\mathbb{P}_f$. Here, we would like to recall that only the free part does not belong to the approximated model.

Figure 3 The surveillance area $\mathbb{S}$ containing a static (B_s^r) and dynamic obstacle (B_d^r) in grey, and an unmodelled dynamic target (U). The first row shows inclusions of the modelled obstacles, static (left), dynamic (right). The undetectable part is displayed in the picture of the second row to the left. The occupied part is shown in the picture to the right and in the next row to the left, caused by dynamic obstacles and dynamic targets, respectively. The last picture shows the free part of this camera in white and grey (see online version for colours)



By the means of a floodfilling algorithm the connected voxels itself are grouped into clusters. These clusters are checked whether their individual volume is smaller than a predefined threshold, here 0.06 m^3. Regarding the volume one can determine if a cluster could contain an unmodelled object, e.g. a human (0.075 m^3). Note that for just a single camera this model reveals a huge visual hull, in our case it is one single cluster. Additional cameras added by the prescription of Formula (3) will reduce the volume.

4.4 Optimisation method

After having evaluated existing solutions by plugging them in the objective function of a problem, the solver of an optimisation 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 $E_{rr_{L,H}}(e_1,\ldots,e_n)$ 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 for all e and the distance therefore remains 0 and convex).

The second characteristic to be discussed is the discontinuity of $E_{rr_{L,H}}(e_1,\ldots,e_n)$ 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-voxel-based model, discontinuities would appear due to the intersections of disconnected parts mentioned in Section 4.1.

The properties of the objective function complicate the search for a suitable solver. As elucidated in standard references on non-linear optimisation (e.g. Nocedal and Wright, 2006), most algorithms take advantage of a characteristical behaviour like convexity, differentiability or at least continuity which cannot be guaranteed in our case. This applies to all deterministic solvers for non-linear programmes 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 (for e.g. Schlüter et al., 2009; Schlüter and Gerdts, 2010).

MIDACO solves the general Mixed Integer Non-linear Programme (MINLP) consisting of continuous and integer variables x and y, equality constraints and inequality constraints.

$$\begin{aligned} &\text{minimise } f(x,y) && (x \text{ continuous}, y \text{ integer}) \\ &\text{subject to } g(j)=0 && (j=1,\ldots,m_{equality}) \\ &\quad g(j)\geq 0 && (j=1,\ldots,m_{equality}+1,\ldots,m) \\ &\quad x_l \leq x \leq x_u \\ &\quad y_l \leq x \leq y_u \end{aligned}$$

The optimisation problem is designed as a black-box model and no derivative information is required. Since only the objective and constraint function values must be provided, the algorithm can be applied to our problem without modifications by setting the number of equality constraints and integer variables to zero and providing the required objective value.

4.5 Complexity

The solver iteratively generates a tuple of settings e_i, $i = 1,\ldots,n$ (one setting for each camera) which are 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 are 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 implemented in the following version: For the given tuple of settings e_i the distance difference of all the H time steps and L appearances are to be evaluated. This is accomplished by testing 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. Without giving the details of a proof, the sum of these components gives us the complexity

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

of the method. Further details can be read in the work of Hänel (2010).

5 Experiments

Since we used 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 consisted of several groups of test runs, the groups only differ in one parameter. We made examinations about changing resolutions, facets, objects, amount of time and appearance steps, amount of cameras and starting point. In order to compare different examinations, the so called 'basic setup', stated in Table 2 was used as the first group of test runs in every examination.

Table 2 'Basic setup', i.e. the setup which is used if no other assumptions are made

Parameter	*Dimension/Amount*
Surveillance area $\mathbb{S}$:	cuboid 4 m × 3 m × 3 m
voxel resolution:	(16 × 12 × 12)
critical points:	all point inside the dynamic collective
static collective:	8 facets at 2 objects
dynamic c.:	24 facets at 6 objects in 2 time steps
unmodelled c.:	24 facets at 6 objects in 3 appearances
cameras:	6 cameras all over the surveillance area
starting point:	cameras are placed and orientated randomly all over $\mathbb{S}$
stop criteria:	maximal time limit 3 h reached or optimisation tolerance (ca.0.0469) reached

The basic setup consists of three appearances of the unmodelled collective and two-time steps for the collective of dynamic obstacles. In Figure 4 the first appearance at the first time instant is illustrated. The surveillance area $\mathbb{S}$ is represented by the white outline of the cuboid. The pair of blue tetraeders resembles the unmodelled targets. As described in Section 3.3, also static and dynamic obstacles can be taken into account. Here, static obstacles are resembled by yellow tetraeders while dynamic ones are resembled by orange tetraeders. We additionally assumed the dynamic obstacles to contain the set of critical points. Thus, a robot could also be modelled as a dynamical object.

Figure 4 First appearance and the first time step of the so called 'basic setup': Surveillance area $\mathbb{S}$ in white, the unmodelled targets (blue tetraeders), static (yellow) and dynamic (orange) obstacles. The orange tetraeders also contain the set of critical points (see online version for colours)

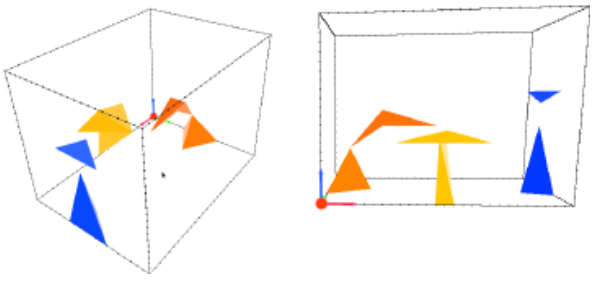


Table 3 contains all test parameters and their ranges. Furthermore, one examination was also dedicated to the restrictions of the settings' domain: We placed the cameras in one group of solver calls only at 'ceiling', in one only in the 'upper fourth' of the room.

Table 3 This is an overview of all examinations. An examination consists of several groups of test runs, each group differs only in one parameter

Parameter	*Parameter ranges*
voxel res.:	(16 + 4i × 12 + 3i × 12 + 3i) for i = 0, 1, 2, 3, 4, 5
static c.:	8 + 60i facets for i = 0,…,5 at 2 obj. 6 + 4i facets at 2 + i objects i = 0,…,3
dynamic c.:	24 + 60i facets i = 0,…,5 at 3 obj./2 timest. 2 + i obj. i = 0,…,3 w. 24 + 4i fac./1 timest. objects placed randomly i = 1,…,5 timest. w. 2i obj 8i fac. 3 appear.
unmodel. c.:	24 + 60i facets i = 0,…,5 at 2 obj./3 appear. 2 + i obj. i = 0,…,3 w. 24 + 4i fac./1 appear. objects placed randomly i = 1,…,5 appear. w. 2i obj 8i fac. 3 timest.
cameras:	i = 3,…,9

The aim of this section is to summarise all the examinations defined in Table 3, and in particular to answer the following central questions: Can the desired optimisation tolerance be satisfied in time, i.e. will the target be approximated as accurately as needed? How many iteration cycles are needed? What is the operating time of one cycle, of each iteration step and of the components of one step? What is the highest memory consumption? The results can also be looked up in the more elaborative work of Hänel (2010).

As an example, Figure 5(left) shows three randomly placed cameras (green) in the environment of Figure 4 (16 × 12 × 12 Voxels, no additional objects, no additional facets, two time steps and three appearances). The model of the unmodelled collective built by this camera network is resembled by the blue voxels in Figure 5(right). For this case, an optimal cameras configuration is shown in Figure 6. The resulting model of the unmodelled collective consists of five voxel clusters (purple, light blue, pink, turquoise and brown) and is more accurate than in the case of randomly place cameras, cf. the blue cluster in Figure 5(right). The improvement can also be measured in terms of the objective value which is the average deviation over time and appearances of the unmodelled collective. In the case of the random placed cameras in Figure 5 we obtain a value of 3.1251 while in case of the optimal placement shown in Figure 6 only the value of 0.0368 is measured.

Figure 5 Left: Three randomly placed cameras (green) in the basic setup. Right: The model of the unmodelled collective built by these three cameras out of blue voxels. The objective value is relatively high: about 3.1251 (see online version for colours)

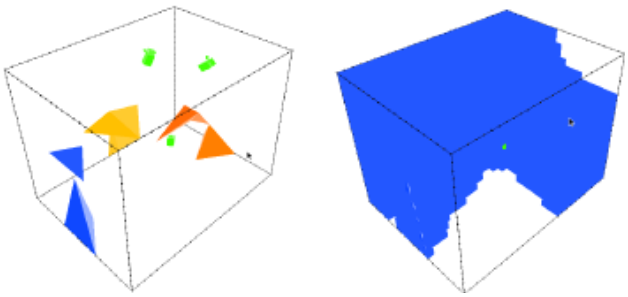
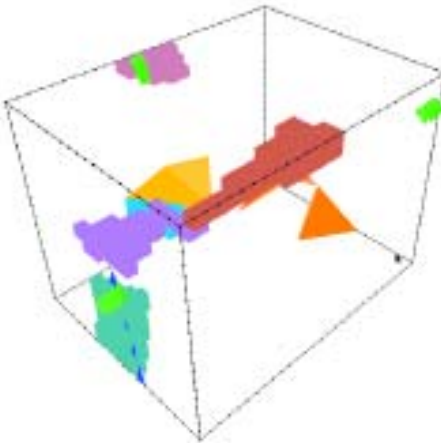


Figure 6 The same amount of cameras as in Figure 5, here optimized with a much more accurate model consisting of five voxel clusters (purple, light blue, pink, turquoise and brown). The corresponding optimized objective value is about 0.0368 (see online version for colours)



5.1 Hardware and software

We implemented the optimisation problem in C++ and compiled it with 'gcc' version 4.0.20050901 (pre-release) optimised 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 to 2.8GHz) with 4 GB RAM.

5.2 Optimisation tolerance

As stopping criteria next to a three hour time limit, we introduced the optimisation tolerance. This is the maximum threshold a tuple of settings must be mapped at for the optimisation to terminate. It is designed to depend on the length of a diagonal of a voxel (see Hänel, 2010). In many cases, the solver was able to satisfy the desired optimisation tolerance in the predefined maximal time. Following exceptions exceeded the time limit: We recorded an increasing time consumption of one iteration step (beyond linear) when gradually raising the resolution of the voxel discretisation. Due to the time criterion, approaches with a resolution of more than 24 × 18 × 18 were terminated before satisfying the optimisation tolerance (see Figures 7 and 9). Increasing the number of randomly placed dynamic obstacles resulted in too many iteration steps (over 160,000 at most compared to less than 45,000 when increasing the amount of static obstacles, cf. Figure 8), and thus resulted in decreasing the amount of tests in which the optimisation tolerance was satisfied in time, as illustrated in Figure 10. In rare cases, a similar outcome was observed when the amount of randomly placed unmodelled objects was increased.

A combination of both occurrences – the time loss in each iteration step and the requirement of too many iteration steps – has been observed for test runs utilising a small number of cameras (considering three cameras it was literally impossible to compute a satisfactory result, see Figure 11). 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 optimisation tolerance (0.046 m^2) but all of them stayed below the value 0.25 m^2. This could be a sign, for e.g. that in our test setting six cameras on the ceiling cannot assimilate the unmodelled collective close enough by the model.

Figure 7 Scatter plot: With refined resolution, the mean (light grey squares) of the amount of iteration steps (each represented in a dark grey square) in one group was higher. Columns: groups of 20 iterations with different resolutions

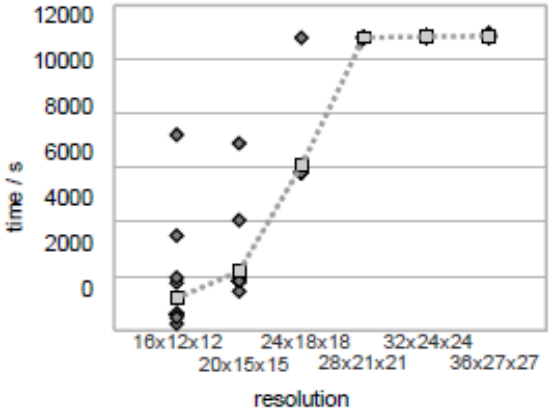


Figure 8 Scatter plot: Dynamic obstacles (and perilous points) are complicating the iteration. Columns: groups of 20 iterations of additional dynamic objects (dark) and static objects (light)

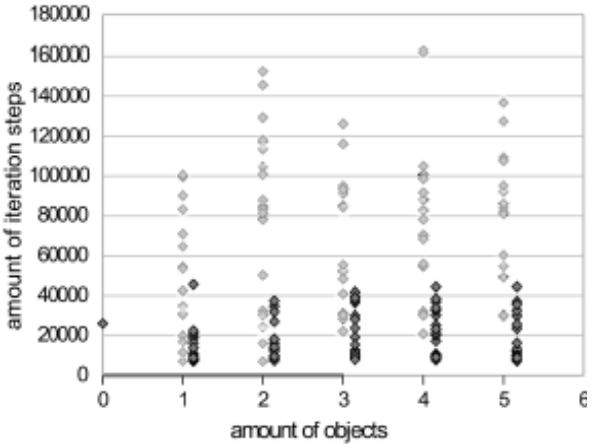


Figure 9 Bar Chart: More refined resolution than (24 × 18 × 18) made it impossible to satisfy the predefined optimization tolerance in time

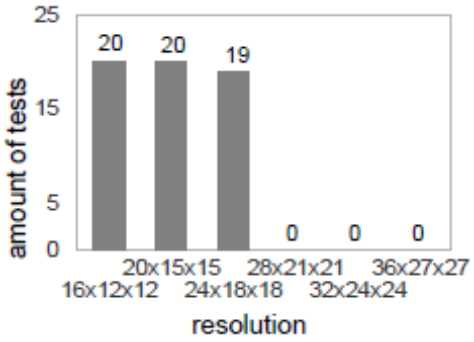


Figure 10 Bar Chart: The more dynamic objects were spread across the surveillance area, the less test runs satisfied the desired optimization tolerance

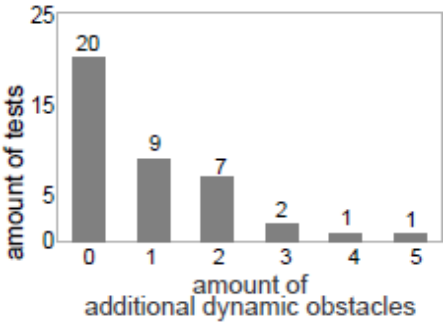
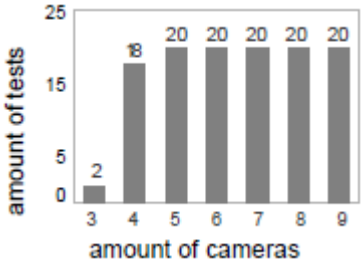


Figure 11 Bar Chart: Five till nine cameras could contour the unmodelled collective best

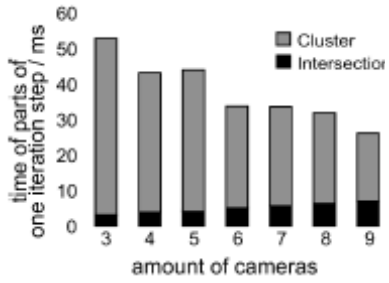


5.3 Time consumption

When raising the amount of appearances, time steps, facets or objects of any of the collectives we also recorded a linearly increasing time consumption for one iteration step. Out of these, the resolution of voxel discretisation 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 optimisation tolerance is satisfied (i.e. the model assimilates the unmodelled collective as accurate as needed), and hence time consumption will slope up when using a larger amount of cameras.

Without giving a detailed explanation about the way one iteration step is calculated with our test setting 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 exceeded 15 ms on average. The test runs with 36 × 27 × 27 voxel, six cameras and 24 facets have reached an average of 50 ms. After intersecting areas, the related voxels need to be combined to clusters, as to be able to check a free parts 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 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, Figure 12 also shows the decreasing time consumption while using more cameras.

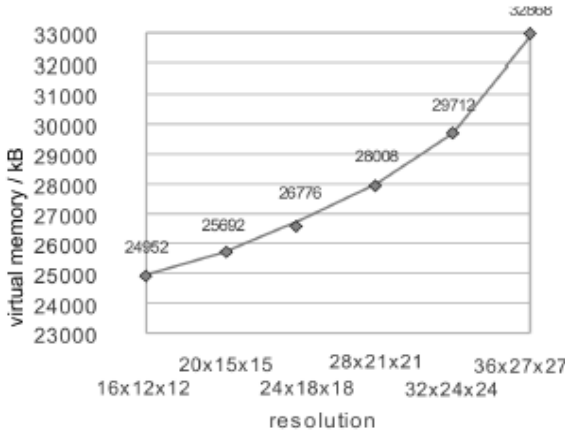
Figure 12 Bar Chart: The total time consumption of an increased amount of cameras sloped down because the clustering (light) weighted more than the actual intersection test (dark)



5.4 Memory

Measurements of the maximal virtual memory while altering the resolution resulted in an ascending graph (beyond linear), cf. Figure 13. The highest demand for virtual memory was measured while testing with the resolution $36 \times 27 \times 27$ (a total of 32,868 kB). The graphs concerning the maximum demand for virtual memory versus facets and amount of cameras are only ascending slowly. Both show a linear slope of about 350–450 kB in our range of parameters.

Figure 13 Plot of increasing maximal virtual memory that was used when refining the resolution of voxels



6 Conclusion and future prospects

We managed to build up a camera placement optimisation algorithm that computes location and orientation of a given amount of cameras inside 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 minimise the error made by evaluating distances to the visual hull of a given object up to the optimisation 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 unknown collective even better and 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 improved time consumption: First of all, it is possible to parallelise the iteration steps of the solver as well as some included intersection tests. But, as the amount of iteration steps of the solver ranged in between about 500 and 160,000, 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 calculations. One of these calculations is the summation of $L \cdot H$ addends (the number of appearances times number of time steps), which all have to be simulated. Time loss will be minimised if cancelling the evaluation of the sum when it trespasses the current optimal value. Also, optimised data structures, like Oct-Trees and BSP-Trees for intersection and inclusion tests may be implemented in future, which improve the time loss during the intersection test.

Acknowledgements

This work was supported by DFG (German Research Foundation) Grant He2696/11 in the project called SIMERO-2, “Sicherheitsstrategien für die Mensch/Roboter-Kooperation und-Koexistenz”. Jürgen Pannek was partially supported by DFG Grant Gr1569/12-1 within the priority research programme 1305 as well as the German National Academy of Sciences Leopoldina Grant LP2009–36.

References

Bodor, R., Drenner, A., Schrater, P. and Papanikolopoulos, N. (2007) ‘Optimal camera placement for automated surveillance tasks’, *Journal of Intelligent and Robotic Systems*, Vol. 50, No. 3, pp.257–295.

Ercan, A., Gamal, A. and Guibas, L. (2006a) ‘Camera network node selection for target localization in the presence of occlusions’, 31 October–3 November, Boulder, Colorado, USA, *SenSys Workshop on Distributed Cameras*,

Ercan, A., Yang, D., Gamal, A. and Guibas, L. (2006b) ‘Optimal placement and selection of camera network nodes for target localization’, *Distributed Computing in Sensor Systems*, pp.389–404.

Erdem, U. and Sclaroff, S. (2004) ‘Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints’, *Proceedings of International Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras, OMNIVIS*, pp.30–41.

Fiore, L., Fehr, D., Bodor, R., Drenner, A., Somasundaram, G. and Papanikolopoulos, N. (2008a) ‘Multi-camera human activity monitoring’, *Journal of Intelligent and Robotic Systems*, Vol. 52, No. 1, pp.5–43.

Fiore, L., Somasundaram, G., Drenner, A. and Papanikolopoulos, N. (2008b) ‘Optimal camera placement with adaptation to dynamic scenes’, *IEEE International Conference on Robotics and Automation*, 19–23 May, Pasadena, CA, pp.956–961.

Hänel, M.L. (2010) *Optimierung von Kamerapositionen zur Überwachung teils unbekannter Umgebungen*, Master's Thesis, University of Bayreuth, Germany.

Hartley, R. (1992) 'Estimation of relative camera positions for uncalibrated cameras', *Proceedings of the 2nd European Conference on Computer Vision, ECCV'92*, 19–22 May, Santa Margherita Ligure, Italy, pp.579–587.

Holt, R., Hong, M., Martini, R., Mukherjee, I., Netravali, R. and Wang, J. (2007) 'Summary of results on optimal camera placement for boundary monitoring', *Proceedings of SPIE*, Vol. 6570, p.657005.

Kuhn, S. and Henrich, D. (2009) *Multi-View Reconstruction of Unknown Objects in the Presence of Known Occlusions*, Technical Report, Angewandte Informatik III (Robotik und Eingebettete Systeme), Universität Bayreuth,.

Luhmann, T., Robson, S., Kyle, S. and Harley, I. (2006) *Close Range Photogrammetry, Principles, Techniques and Applications*, Whittles Publishing.

Mittal, A. (2006) 'Generalized multi-sensor planning', *Proceedings of 9th European Conference on Computer Vision*, 7–13 May, Graz, Austria, pp.522–535.

Mittal, A. and Davis, L. (2004) 'Visibility analysis and sensor planning in dynamic environments', *Proceedings of 8th European Conference on Computer Vision*, 11–14 May, Prague, Czech Republic, pp. 175–189.

Mittal, A. and Davis, L. (2008) 'A general method for sensor planning in multi-sensor systems: extension to random occlusion', *International Journal of Computer Vision*, Vol. 76, No. 1, pp.31–52.

Murray, A., Kim, K., Davis, J., Machiraju, R. and Parent, R. (2007) 'Coverage optimization to support security monitoring', *Computers, Environment and Urban Systems*, Vol. 31, No. 2, pp.133–147.

Nocedal, J. and Wright, S.J. (2006) *Numerical Optimization, Springer Series in Operations Research and Financial Engineering*, 2nd ed., Springer-Verlag, New York.

Olague, G. (2000) 'Design and simulation of photogrammetric networks using genetic algorithm', *Proceedings of the 2000 Meeting of the American Society for Photogrammetry and Remote Sensing (ASPRS 2000)*.

Olague, G. and Dunn, E. (2007) 'Development of a practical photogrammetric network design using evolutionary computing', *The Photogrammetric Record*, Vol. 22, pp.22–38.

Olague, G. and Mohr, R. (1998a) 'Optimal 3D sensors placement to obtain accurate 3D points positions', *Proceedings of the 14th International Conference on Pattern Recognition*, 16–20 August, Brisbane, Australia, Vol. 1, pp.16–20.

Olague, G. and Mohr, R. (1998b) 'Optimal camera placement for accurate reconstruction', *Pattern Recognition*, Vol. 35, No. 4, pp.927–944.

Schlüter, M., Egea, J. and Banga, J. (2009) 'Extended ant colony optimization for non-convex mixed integer non-linear programming', *Computer & Operations Research*, Vol. 36, No. 7, pp.2217–2229.

Schlüter, M. and Gerdts, M. (2010) 'The oracle penalty method', *Journal of Global Optimization*, Vol. 47, No. 2, pp.293–325.

Vecherin, S., Wilson, D. and Pettit, C. (2011) 'Optimal sensor placement with signal propagation effects and inhomogeneous coverage preferences', *International Journal of Sensor Networks*, Vol. 9, No. 2, pp.107–120.

Wang, C.C. (2010) 'Approximate Boolean operations on large polyhedral solids with partial mesh reconstruction', *IEEE Transactions on Visualization and Computer Graphics*, Vol. 17, No. 6, pp.836–849..

Yang, D., Shin, J., Guibas, L. and Ercan, A. (2004) 'Sensor tasking for occupancy reasoning in a network of cameras', *Proceedings of 2nd IEEE International Conference on Broadband Communications, Networks and Systems (BaseNets'04)*.