On the Optimal Placement of Multiple Visual Sensors

E. Hörster, R. Lienhart
Multimedia Computing Lab
University of Augsburg
Augsburg, Germany
{hoerster,lienhart}@informatik.uni-augsburg.de

ABSTRACT

Many novel multimedia systems and applications use visual sensor arrays. An important issue in designing sensor arrays is the appropriate placement of the visual sensors such that they achieve a predefined goal. In this paper we focus on the placement with respect to maximizing coverage or achieving coverage at a certain resolution. We identify and consider four different problems: maximizing coverage subject to a given number of cameras (a) or a maximum total price of the sensor array (b), optimizing camera poses given fixed locations (c), and minimizing the cost of a sensor array given a minimally required percentage of coverage (d). To solve these problems, we propose different algorithms. Our approaches can be subdivided into algorithms which give a global optimum solution and heuristics which solve the problem within reasonable time and memory consumption at the cost of not necessarily determining the global optimum. We also present a user-interface to enter and edit the spaces under analysis, the optimization problems as well as the other setup parameters. The different algorithms are experimentally evaluated and results are presented. The results show that the algorithms work well and are suited for different practical applications. For the final paper it is planned to have the user interface running as a web service.

Categories and Subject Descriptors: I.6.3 [Simulation and Modeling]: Applications; I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture

General Terms: Algorithms, Design

Keywords: Visual sensor placement, Camera arrays

1. INTRODUCTION

Visual sensor arrays are used in many novel multimedia applications such as video surveillance, sensing rooms, assisted living or immersive conference rooms. Most of these applications require the layout of video sensors to assure a minimum level of image quality or image resolution. In visual surveillance a predefined space needs to be covered either partially (i.e. specific areas only) or completely by the visual sensor array. Thus, an important issue in designing visual sensor arrays is the appropriate placement of the cameras such that they achieve one or mulitple predefined goals.

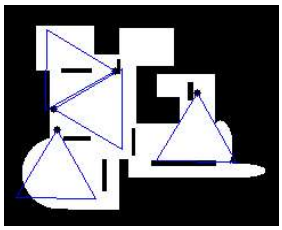


Figure 1: Placement of four cameras to maximize coverage of the given space. Cameras are marked as black points with a blue triangle indicating their field-of-view

Currently most designers of multi-camera systems place cameras by hand because there exists only little theoretical research on visual sensor placement. As video sensor arrays are getting larger, efficient camera placement strategies need to be developed.

Here we focus on the topic of maximizing or achieving coverage with respect to a predefined 'sampling rate' guaranteeing that an object in the space will be imaged at a minimum resolution (see Section 2 for a precise definition). A typical scenario is shown in Figure 1. The space consists of all white regions, black regions mark background and/or obstacles. Camera positions are marked with black circles and their fields-of-view with blue lines.

We consider the visual sensor placement problem in its various facets: Maximizing coverage subject to a given number of cameras is one problem we aim to solve. Often several different types of cameras are available. They differ in their ranges of view, intrinsic parameters, image sensor resolutions, optics, and costs. Therefore in this paper we also consider minimizing the cost of a visual sensor array while maintaining a minimally required percentage of coverage of the given space as well as maximizing coverage given a maximum price for the camera array.

In some situations cameras have already been installed (e.g. at an airport). Positions of the multiple cameras can be determined automatically by camera calibration. Given the fixed initial positions and camera types, we also address the problem of determining the optimal poses with respect to coverage while maintaining the required resolution.

VSSN'06, October 27, 2006, Santa Barbara, California, USA.
https://doi.org/10.1145/1178782.1178800

Different algorithms are proposed to solve these problems. They may be subdivided into algorithms which determine the global optimum solution and heuristics which solve the problem within reaonable time and memory consumption, but do not necessarily determine the global optimum. A careful evaluation and a comparative study of the different approaches show their respective advantages.
A user interface is developed in order to comfortably enter and edit the layout of spaces and setup parameters of the respective optimization problem. This will be running as a web service [1].

1.1 Related Work

Although a significant amount of research exists in designing and calibrating visual sensor arrays, automated visual sensor placement and alignment in general has not been addressed frequently.
There exists some work in the area of (grid) coverage problems and sensor deployment with sensors sensing events that occur within a distance r (the sensing range of the sensor) [14], [3], [17], [15]. Our linear programming work is partly based on the approach presented in [3], but differs in the sensor and space model (e.g. cameras do not posses circular sensing ranges) as well as the cost function and some constraints.
We have presented our visual sensor model in previous work for the two-dimensional [8] and the three-dimensional case [9]. In those works we proposed an linear programming (LP) approach to determine the minimum cost of a sensor array that covers a given space completely, and also an LP approach that determines for fixed sensor arrays their optimal pan and tilt with respect to coverage. In both approaches space is presented as a regular grid.
In [5] a camera placement algorithm based on a binary optimization technique is proposed. Only polygonial spaces are considered, which are presented as occupancy grids in constrast to our approach where we allow spaces to have arbitrary shape and the points representing space are distributed according to a user defined importance distribution. Additionally, in [5] only one of the four problems considered in this paper is covered and the authors do not comment on handling large spaces.
In [4] camera placement for robust motion capture is considered. A quality metric is constructed taking in account resolution and occlusion. This metric is combined with an evolutionary-algorithm based optimizer and used to automate the camera placement process. Despite the fact that the paper only reports two very simple placement results for small problems, the outcome of the algorithm is not necessarily the global optimum. In [12] the problem of finding the optimal camera placement for accurate reconstruction is approached using a multicellular genetic algorithm. Results are only reported on very restricted experiments.
In [11] the authors present a method for sensor planning in dynamic scenes. Therefore they analyze the visibility from static sensors at sampled points statistically and solve the problem by simulated annealing.
The sensor placement problem is also closely related to the guard placement problem (AGP) – the problem of determining the minimum number of guards required to cover the interior of an art gallery. It is addressed by the art gallery theorem [13]. In AGP all guards are assumed to have similar capabilities whereas we also consider cameras with different fields-of-views at different levels of costs. Additionally the field-of-view is restricted in our sensor model due to resolution and sensor properties.

1.2 Contributions

The main contributions of this paper are:

- Space is sampled according to an underlying importance distribution instead of using a regular grid of points.
- A linear programming model for each of our problems is presented which gives a optimal solution to the respective problem. We show how to reduce the number of variables and constraints significantly, thus enabling an optimal solution for larger problems.
- Several heuristics are proposed to approximate the optimal solution of the different camera placement problems.
- An interface that enables the user to comfortably enter and edit the space, the optimization problems as well as the other setup parameters is presented.
- An experimental and competitive evaluation of the different approaches is given showing the different algorithms' specific advantages.

1.3 Paper Organization

The paper is organized as follows. In Section 2, the problem is stated and basic definitions are given. Section 3 presents our different approaches to solve the problem. Our visual user-interface is presented in Section 4. In Section 5 we evaluate our approaches experimentally. Possible extensions are discussed in Section 6, and Section 7 summarizes and concludes the paper.

2. PROBLEM FORMULATION

2.1 Definitions

In the following the term *space* denotes a physical two- or three-dimensional room.
A point in that space is *covered* if that point is captured with a required minimal resolution. The minimal resolution is satisfied if the point in space is imaged by at least one pixel of a camera that does not aggregate more than x cm^2 of a surface parallel to the imaging plane through that point. x is expressed in terms of the sampling frequency f_s and converted into the field-of-view of a camera.
The $field-of-view$ of a camera is defined as the volume in which a pixel aggregates no more than $\frac{1}{f_s^2}cm^2$ of a surface parallel to the imaging plane. Thus an object that appears in the camera's field-of-view is imaged with at least this resolution assuming the object has a planar surface orthogonal to the optical axis. Clearly the resolution is smaller if the surface is not orthogonal. How to account for this case is discussed in Section 6.
We also consider static occlusions. To simplify the derivation and evaluation only the 2D problem is discussed in this paper; however, the presented approaches can be extended easily to the third dimension (see Section 6).

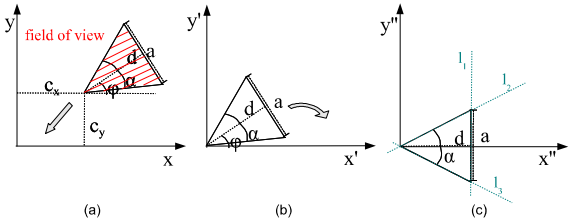


Figure 2: Deriving the model of a camera's field-of-view

2.2 Problem Statement

There are many problems that could be considered in the placement of multiple visual sensors. Here we focus on four problems. Variations can be tackled in a similar fashion.
Given a space to be covered and a sampling frequency f_s, we are interested in the following problems:

- *Problem 1*: Given the number of cameras of one type and their specific parameters, determine their positions and poses in space such that coverage is maximized.
- *Problem 2*: Given several types of cameras, their parameters and specific costs as well as the maximum total price of the visual sensor array, determine the camera types and positions/poses that maximize coverage in the given space.
- *Problem 3*: Given the fixed positions and respective types of a number of cameras determine their optimal poses with respect to maximizing coverage.
- *Problem 4*: Given a minimally required percentage of coverage determine the camera array with minimum cost that satisfies this coverage constraint. In the case that only one type of camera is available the minimum number of cameras of that type that meets the coverage constraint is determined.

2.3 Modeling a Camera's Field-Of-View

We use a simple model for our cameras: the field-of-view of a camera is described by a triangle as shown in Fig. 2 (a). The parameters of this triangle are calculated given the (intrinsic) camera parameters and the sampling frequency f_s using well known geometric relations.
Defining the field-of-view by a triangle enables us to describe the area covered by a camera at position (c_x, c_y) and with pose φ by three linear constraints. Therefore a camera's field-of-view is first translated to the origin of the coordinate system (Fig. 2 (b)):

$$x' = x - c_x, \quad y' = y - c_y \tag{1}$$

Then we rotate the field-of-view such that its optical axis becomes parallel to the x-axis (Fig. 2 (c)):

$$x'' = \cos(\varphi) \cdot x' + \sin(\varphi) \cdot y' \tag{2}$$
$$y'' = -\sin(\varphi) \cdot x' + \cos(\varphi) \cdot y' \tag{3}$$

The resulting area covered by the triangle (Fig. 2 (c)) can now be described by three line equations l_1, l_2, l_3:

$$l1: \quad x'' \leq d \tag{4}$$

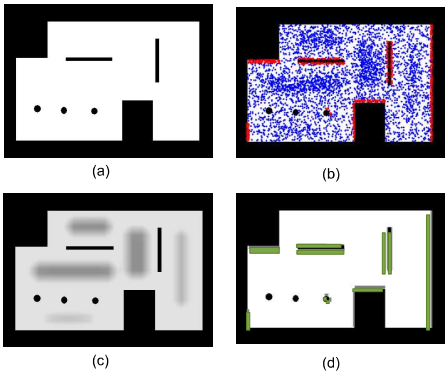


Figure 3: Modeling space by sampling: (a) white area defines space, black area defines walls; (b) sampled space: red dots mark possible camera positions, blue dots control points; (c) importance weighting of space: darker gray regions are more important than lighter regions; (d) allowed positions for camera placement are marked in green

$$l2: \quad y'' \leq \frac{a}{2d} \cdot x'' \tag{5}$$

$$l3: \quad y'' \geq -\frac{a}{2d} \cdot x'' \tag{6}$$

By substitution the following three linear constraints define the area covered by the field-of-view of a camera:

$$\cos(\varphi) \cdot (x - c_x) + \sin(\varphi) \cdot (y - c_y) \leq d \tag{7}$$

$$-\sin(\varphi) \cdot (x - c_x) + \cos(\varphi) \cdot (y - c_y) \leq \frac{a}{2d} \cdot (\cos(\varphi) \cdot (x - c_x) + \sin(\varphi) \cdot (y - c_y)) \tag{8}$$

$$-\sin(\varphi) \cdot (x - c_x) + \cos(\varphi) \cdot (y - c_y) \geq -\frac{a}{2d} \cdot (\cos(\varphi) \cdot (x - c_x) + \sin(\varphi) \cdot (y - c_y)) \tag{9}$$

2.4 Modeling Space

In the ideal case camera positions and poses are continuous in space, i.e. c_x, c_y and φ are continuous variables. As we are not able to solve our problem for this ideal case, we approximate the continuous case by sampling the positions and poses. Cameras can only adopt discrete positions and poses. A second aspect is that cameras usually cannot be installed everywhere in a given space. Thus the user may define regions where cameras could be set up (Figure 3(d))[1]. Those regions are then sampled randomly to identify a number A of camera locations. Poses are also discretized and cameras can only adopt a number of s_φ discrete poses.
In order to define coverage, control points are sampled from the entire space with respect to their weighting. The user can assign importance values to regions in space, which is then sampled with a probability dependend on the importance of this region. If some parts of the room are known to be more important, e.g. at doors, a higher weighting can be given to those parts resulting in a higher density of samples,

[1]Section 4 gives detailed informations about our user-interface.

whereas e.g. parts that are less interesting might be sampled with a lower frequency. This approach is illustrated in Figure 3(c). Darker regions are more important than lighter regions except black pixels, which mark the border of the space or static obstacles constricting the field-of-view of the cameras. Altogether a number of P control points is sampled.
As the number of control points, camera positions and camera poses increases, so does the accuracy of our approximation. For $P \to \infty$, $A \to \infty$ and $s_\varphi \to \infty$ our approximated solution converges to the continuous-case solution.
A resulting configuration is shown in Figure 3(b). Control points are maked by blue dots, possibe camera positions by red dots.
With that our problems turn into set coverage problems. A control point is then assumed to be covered by a certain camera if and only if Eq. 7 to 9 are satisfied and no obstacles constrict the field-of-view of the visual sensor in the direction of the control point.

3. APPROACHES

We propose different approaches to solve the visual sensor placement problems. The approaches can be subdivided into algorithms which give a global optimal solution but are complex and time/memory consuming, and heuristics which solve the problem in reasonable time and with reasonable complexity. Experimental results will evaluate the different proposed heuristics by comparing those to the optimal solution as a bottom line, for comparsion purposes random placement is also described.

3.1 Exact Algorithms

3.1.1 Linear Programming

In this section we describe our linear programming approaches to solve optimally the problems listed in Section 2. In the following we first derive an Binary Integer Programming (BIP) model to solve Problem 4. Subsequently it is shown how to modify this model in order to solve the other problems.
We aim to solve the following optimization problem: Several types of cameras with different sensor resolutions and optics (i.e. focal lengths) are available. For each type of camera k the field-of-view parameters d_k and a_k (see Fig. 2) and a cost K_k are given. Our objective is to find the configuration of cameras that minimizes the total cost of the visual sensors by optimally assigning cameras to possible camera placements and angles while ensuring coverage for at least a minimum percentage p of control points.
We assume that our space consists of P control points. Visual sensors locations are restricted to A positions. Similarily we discretize the angle φ defining a camera's pose to s_φ different poses.
If only one type of camera is available, our binary programming model remains the same, but as the total price is minimized so is the total number of cameras in the array. To obtain the minimal number of cameras that satisfy the constraints the objective value has to be divided by the price of one camera.
Our approach was inspired by the algorithms presented in [3] and [9]. First we define some binary variables. Let a binary variable c_i be defined by:

$$c_i = \begin{cases} 1 & \text{if control point } i \text{ is covered} \\ & \text{by a minimum of } M \text{ cameras} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The total number $nbCovered$ of covered sample points is then given by

$$nbCovered = \sum_i c_i \tag{11}$$

Further we define the two binary variables $x_{kj\varphi}$ and $g(k, j, \varphi, i)$:

$$x_{kj\varphi} = \begin{cases} 1 & \text{if camera of type k is placed at} \\ & \text{position } j \text{ with orientation } \varphi \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

$$g(k, j, \varphi, i) = \begin{cases} 1 & \text{if a camera of type } k \text{ at position } j \text{ with} \\ & \text{orientation } \varphi \text{ covers control point } i \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

$g(k, j, \varphi, i)$ can be calculated in advance for each camera and stored in a table. The total cost of the sensor array is then given by:

$$\min \sum_k K_k \sum_j \sum_\varphi x_{kj\varphi} \tag{14}$$

Next we need to express the variables that define coverage in terms of the other above defined variables. This is done as shown below. Since $c_i = 1$, if and only if at least M cameras cover the control point i we introduce the following two inequalities for each grid point:

$$c_i \cdot (\sum_{k,j,\varphi} x_{kj\varphi} \cdot g(k, j, \varphi, i) - M) \geq 0 \tag{15}$$

$$(1 - c_i) \cdot (M - \sum_{k,j,\varphi} x_{kj\varphi} \cdot g(k, j, \varphi, i)) \geq 0 \tag{16}$$

The constraints 15 and 16 involve products of binary variables, thus they are nonlinear. In order to linearize the inequalities, we introduce a new binary variable for each appearance of a nonlinear term, as well as two additional constraints [16]. Therefore we replace each $c_i \cdot x_{kj\varphi}$ term by a binary variable $v_{kj\varphi i}$ and introduce the following constraints:

$$c_i + x_{kj\varphi} \geq 2 \cdot v_{kj\varphi i} \tag{17}$$

$$c_i + x_{kj\varphi} - 1 \leq v_{kj\varphi i} \tag{18}$$

To ensure that only one camera is assigned to a specific camera position sample, we need to add the constraint

$$\sum_{k,\varphi} x_{kj\varphi} \leq 1 \tag{19}$$

for each possible camera position. To ensure, that the minimal predefined percentage of points is covered, the following constraint is needed, too:

$$\sum_i c_i \geq p \cdot P \tag{20}$$

Our sensor deployment problem can now be formulated as an BIP model. The resulting BIP model is shown in Figure 4. The variable T denotes here the number of available camera types.

$$\min \sum_{k} K_k \sum_{j} \sum_{\varphi} x_{kj\varphi}$$

subject to:

$$c_i + x_{kj\varphi} \geq 2 \cdot v_{kj\varphi i}$$

$$c_i + x_{kj\varphi} - 1 \leq v_{kj\varphi i}$$

$$0 \leq k \leq (T-1), \quad 0 \leq j \leq (A-1), \quad 0 \leq \varphi \leq (s_\varphi - 1), \quad 0 \leq i \leq (P-1)$$

$$\sum_{k,j,\varphi} v_{kj\varphi i} \cdot g(k,j,\varphi,i) - c_i \cdot M \geq 0$$

$$M - \sum_{k,j,\varphi} x_{kj\varphi} \cdot g(k,j,\varphi,i) - c_i \cdot M + \sum_{k,j,\varphi} v_{kj\varphi i} \cdot g(k,j,\varphi,i) \geq 0$$

$$0 \leq i \leq (P-1)$$

$$\sum_{k,\varphi} x_{kj\varphi} \leq 1$$

$$0 \leq j \leq (A-1)$$

$$\sum_{i} c_i \geq p \cdot P$$

$$c_i, v_{kj\varphi i}, x_{kj\varphi} \in \{0,1\}$$

$$0 \leq k \leq (T-1), \quad 0 \leq j \leq (A-1), \quad 0 \leq \varphi \leq (s_\varphi - 1), \quad 0 \leq i \leq (P-1)$$

Figure 4: BIP model to solve Problem 4

P	A	s_φ	max reduc.	min reduc.
500	100	4	25.44%	22.44%
500	100	8	33.21%	23.97%
500	200	4	25.72%	20.60%

Table 1: Minimum and maximum reduction in the number of variables achieved for different parameter settings (assuming the BIP model for problem 1,2 and 4)

The proposed model needs only a few modifications to solve Problem 1 and 2. As the objective in Problem 1 and 2 is to maximize coverage, we need to replace the objective function of the above BIP model to

$$\max \sum_{i} c_i \tag{21}$$

As the maximization procedure favors the variable c_i to be 1, constraint 16 can be dropped now, as it is soley used to force c_i to value 1 if the coverage constraints are satisfied.
To derive the BIP model for Problem 1 we need as well to substitue contraint 20 by:

$$\sum_{k,j,\varphi} x_{kj\varphi} = N \tag{22}$$

where N denotes the number of cameras we are allowed to place.
If we aim to solve Problem 2, we need to substitue constraint 20 by

$$\sum_{k,j,\varphi} K_k \cdot x_{kj\varphi} \leq F \tag{23}$$

where F denotes the maximally allowed total price of the sensor array.

The BIP model that solves Problem 3 is shown in Figure 5.

$$\max \sum_{i} c_i$$

subject to:

$$c_i + x_{n\varphi} \geq 2 \cdot v_{n\varphi i}$$

$$c_i + x_{n\varphi} - 1 \leq v_{n\varphi i}$$

$$0 \leq i \leq (P-1), \quad 1 \leq n \leq N, \quad 0 \leq \varphi \leq (s_\varphi - 1)$$

$$\sum_{n,\varphi} v_{n\varphi i} \cdot g(n,\varphi,i) - c_i \cdot M \geq 0$$

$$0 \leq i \leq (P-1)$$

$$\sum_{\varphi} x_{n\varphi} = 1$$

$$1 \leq n \leq N$$

$$c_i, v_{n\varphi i}, x_{n\varphi} \in \{0,1\}$$

$$0 \leq i \leq (P-1), \quad 1 \leq n \leq N, \quad 0 \leq \varphi \leq (s_\varphi - 1)$$

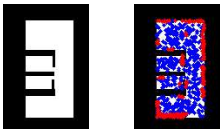
Figure 5: BIP model to solve Problem 3

We have proposed a similar model for solving this problem in the 3D case using regular grid points in [9]. The binary variable $x_{n\varphi}$ is defined by:

$$x_{n\varphi} = \begin{cases} 1 & \text{if camera } n \text{ at position } (c_x, c_y) \text{ with type } k \\ & \text{has the orientation } \varphi \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

We assume in this problem that the positions and types of all N cameras are given and fixed. The last constraint in the shown BIP model (Figure 5) ensures, that exactly one pose is assigned to each camera.

The number of variables and constraints depends on the number of control points and possible cameras' positions and poses. Thus, if we increase those to achieve a better approximation of the continuous case, the number of variables and constraints in our BIP model increases accordingly. As we are not able to solve the BIP problem in a reasonable amount of time and memory with an arbitrarily large number of variables and constraints, it is essential to keep the number of variables and constraints as small as possible. Based on the observation, that possible camera positions are often at obstacles or borders of the space (as it is easy to mount cameras on walls etc.) and those walls/obstacles restrict the cameras view in some of the sampled orientations, we propose to discard combinations of type, pose and position that do not cover any control point. Thus the number of variables $x_{kj\varphi}$ or $x_{n\varphi}$ respectively decreases and hence also the number of variables $v_{kj\varphi i}$ or $v_{n\varphi i}$ as well as the number of constraints. Of course, the factor of reduction depends on the possible camera positions and orientations, the space geometry such as the number of obstacles, and other parameters such as field-of-view parameters or the P/A ratio. The proposed reduction does not change the optimality of the result.
Table 1 shows the percentage of reduction achieved in the number of variables (and thus also in the number of constraints) for different parameter settings and for the example



(a) (b)

Figure 6: Example space used for the variable reduction experiment (a) and sampled space (b), red dots mark possible camera positions and blue dots control points

space shown in Figure 6. For each parameter setting, the experiment has beed run five times, i.e the possible camera positions and control points have been recomputed each time. The number of camera types has been set to one. The table shows the best and worst achieved reduction in the number of variables. The reduction is at least around 20%.

3.2 Heuristics

3.2.1 *Greedy Search*

Our greedy search algorithm corresponds to the one proposed in [6] but introduces different stop criterions and modifies the definition of 'rank'.
A greedy search algorithm is a constructive heuristic which determines the best placement and orientation of one sensor at a time, i.e. the iterative procedure places one camera during each iteration. Therefore we first compute for each discrete camera position, orientation and type a list of control points that are adequately covered by that camera. In Problem 3, for each fixed position and type only the pose is varied. A control point is adequately covered, if it lies in the field-of-view of a camera and the view of the camera is not occluded in the direction of that control point. By doing a greedy selection, i.e. choosing at each step the camera with the highest rank, we find a near optimal solution to our problems.
The rank of a camera is defined for the case of Problem 1 and 3, as the number of adequately covered control points that it adds to those already covered by previously placed cameras. It is denoted as $NbCov_R$. In Problem 2 and 4 the rank of a camera is defined as the inverse of the ratio of r where

$$r = \frac{K_k}{NbCov_R} \tag{25}$$

and K_k denotes the price of the respective camera type considered. The ration r measures the price of adding one control point with this camera type at the current position and with the current orientation. Thus the camera with the cheapest price per added point is chosen. The control points which are covered by the last placed camera are deleted from the current pool of control points and the iterative procedure is continued. If there are multiple highest ranking cameras, we choose of those the camera with the highest rank with respect to the original control point set. This is reasonable because it causes some points to be covered multiple times and hence those might be covered from more than one direction (see also Section 6 for further explanations).
Cameras are added to the solution set until a stop criterion is reached. In Problem 1 the stop criterion is the allowed number of visual sensors in the solution set, in Problem 3 the criterion is reached if an orientation is assigned to all cameras. In Problem 4 the criterion is a sufficient percentage of coverage of all control points and in Problem 2 the upper limit of the total price of the array, i.e. that no camera of any type can be added without exceeding the limit. The pseudo code of our greedy search algorithm is shown in Figure 7.

```
Input:
- a set of possible camera positions, poses and types
- a set of control points

Algorithm:
  begin
    Compute for every camera  position, pose and type combination a list of control
    points that are adequately covered;
    iter=0, stop=0;
    do
      Calculate the rank of each camera position, orientation and type combination;
      Search for the position-orientation-type combination with the highest rank;
      totalPrice(iter)=totalPrice(iter-1)+cost of highest ranking camera;
      if( iter<N or totalPrice(iter)<=F or nbCoveredPoints(iter-1)<p*P)
        Place sensor of type t at the position and pose that has the highest rank;
        Update the set of uncovered control points;
        nbCoveredPoints(iter)=nbCoveredPoints(iter-1)+newly covered points;
        Update the set of possible camera positions;
        Update the set of possible camera types (due to cost constraints);
        iter=iter+1;
      else
        stop=1;
    until (stop=1)
  end
```

Figure 7: Greedy search algorithm

3.2.2 *Dual Sampling*

The dual sampling algorithm is an incremental planner, i.e. also a constructive heuristic. The algorithm has been originally proposed in [7] for the acquisation of range-images using a mobile robot. We modify this algorithm for our multiple visual sensor placement tasks and propose two different algorithms. It should be noted, that both algorithms are not suited to solve Problem 3.
The input to the algorithms is a set of previously computed control points. In each iteration we select randomly one point from the remaining set of uncovered points. We then determine the camera's type, position and pose that covers the selected control point and has the highest rank. We define 'rank' in the same way as in the previous section.
The first version of our dual sampling algorithm determines the highest ranking cameras type, position and pose from a given set of possible types, positions and orientations. This set has been computed previously to calling the placement algorithm, i.e. is an input argument. In our second algorithm a set of possible camera types and orientations is also input to the placement algorithms, but possible positions are computed in every iteration following to the random selection of one control point. Possible positions are obtained by sampling only in a region around this control point (see Figure 8), thus increasing the possibility that the control point can be covered from this location. This procedure enables us to sample possible positions with a locally higher density.
The set of uncoverd control points is reduced in each iteration. The algorithm stops if the stop criterion associated to the current placement problem is reached. The stop criterions are defined as in Section 3.2.1. Figure 9 summarizes the pseudo code of our dual sampling algorithms.

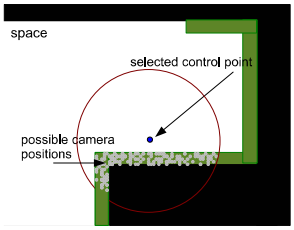


Figure 8: Position sampling in version 2 of the dual sampling algorithm

Input:
- a set of possible camera poses and types
- a set of control points
- if (*version=1*): a set of camera positions

Algorithm:
begin
if(version=1)
Compute for every camera position, pose and type combination a list of control points that are adequately covered;
iter=0, stop=0;
do
Select randomly one control point p from the set of uncovered control points;
if (*version* =2)
Compute set of possible camera positions by sampling the region around control point p;
Compute for every camera position, pose and type combination a list of control points that are adequately covered;
Calculate the rank of each position-orientation-type combination;
Search for the position-orientation-type combination with the highest rank;
totalPrice(iter)=totalPrice(iter-1)+cost of highest ranking camera;
if(iter<N or totalPrice(iter)<=F or nbCoveredPoints(iter-1)<p*P)
Place sensor of type t at the position and pose that has the highest rank and coveres control point p;
Update the set of uncovered control points;
nbCoveredPoints(iter)=nbCoveredPoints(iter-1)+newly covered points;
Update the set of possible camera types (due to cost constraints);
if(*version=1*)
Update the set of possible camera positions;
iter=iter+1;
else
stop=1;
until (stop=1)
end

Figure 9: Version 1 and 2 of the dual sampling algorithm

3.2.3 *Others*

Another method of handling large spaces is to use a divide and conquer scheme to approximate the optimum LP solution as suggested in [3]. The authors propose to divide the space, which in their case is a regular grid, into a number of small sub-grids and combine the optimal solution to each small grid to an approximated solution to the original grid. While one can apply this scheme easily to rectangular grid sensing fields, dividing the space properly to yield a good approximation solution to the original set could be difficult for non-regular control point sets and non-rectangular space boundaries especially in the presence of static obstacles. For the future one approach could be to include the division of the space into the user-interface to enable the user to control and supervise the proper division.

3.3 Random Placement

We have also implemented random camera placement. By comparing the other algorithms with the result given from random placement, we can evaluate in Section 5 to what extend our placement algorithms improve the solution.
To solve one of the four problems, the random placement algorithm selects randomly cameras from a set of possible camera positions, poses and types until the appropriate stop criterium, as defined in Section 3.2.1, is reached. Before selection, position-pose-type combinations which do not cover at least one control point are discarded from the set of possible combinations.

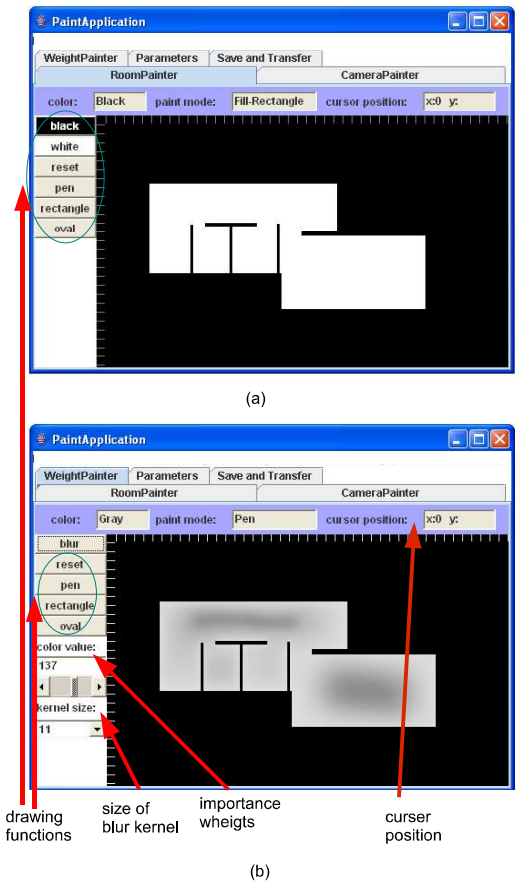


Figure 10: User-interface: (a) entering the space geometry; (b) assigning importance weights to space points

4. USER-INTERFACE

Figure 10 shows parts of our user interface. A Java application has been designed and implemented to enter and edit the spaces, the optimization problems as well as the other setup parameters.
Basically the interface constists of five tabbed panes. The user enters and edits the spaces in the first tab pane (as shown in Figure 10(a)). This is done by using drawing function for rectangles, ellipses and alike. The space is drawn in white color, black color marks borders of the spaces, background and walls as well as other obstacles. A cursor can be used in order to edit the space accurately. It must be noted that the units in which the space is entered need to correspond to the units in which the camera parameters are given.
In the second pane the previous space geometry is loaded and the user marks the possible camera positions (one example is shown in Figure 3(d)).

Next the importance of each region in the space can be entered in the third tab pane (see Figure 10(b)). Therefore color values are assigned to regions corresponding to their importance by again using drawing functions for rectangles and alike. Darker color (except black regions) marks more important regions which are given a higher weight. Borders between regions of different weight may be smoothend by applying a blur filter of adjustable kernel size. Thus each pixel in the space is assigned a weight corresponding to its gray value.
The type of optimization problem, its parameters and other setup parameters are entered in a fourth tab pane. Finally, the last pane enables the user to store and/or submit the entered problem and space geometries to a web service.
This user interface will be running as a web service [1].

5. EXPERIMENTAL EVALUATION

All presented approaches to solve the different placement problems have been implemented in C++. BIP models are solved using the LINGO package (64 bit version) [2]. We chose a professional optimization software over the freely available *lpsolve* package [10], since *lpsolve* has problems solving BIP's that exceed a certain size in terms of variables and constraints.

5.1 Comparision of the Different Approaches

The BIP approach, which determines the global optimum, is not applicable to large spaces, as there exists an upper limit on the number of variables due to memory and computational constraints. In our first set of experiments we aim to determine the 'quality' of the presented heuristics, i.e. we address the question of how well their solutions approximate the global optimum BIP solution.
We used four different small rooms (see Figure 11) in our experiments. These rooms are sampled to obtain sets of discrete space points and camera positions as well as discrete poses. We sampled 50 possible camera positions and between 100 and 150 control points. Cameras could adopt four to eight different discrete orientations and either one or two types of cameras were available depending on the problem. The cheaper camera type was assumed to have a smaller field-of-view.
Those sets were input to our five different algorithms, the BIP approach which determines the optimal solution, the greedy approach, both versions of the dual sampling algorithms and the algorithm of randomly choosing camera positions, poses and types. The performance of the algorithms has been tested separately for all four problem instances, except for the two different dual sampling approaches which do not apply to problem 3. The results are illustrated in Figure 12 and 13 . Due to the stochastic aspect in the outcome of the dual sampling algorithms and the random placement algorithm we have plotted the average results for these algorithms over ten runs.
The results show clearly the improvement for all different algorithms over random placement of the sensors in all cases, i.e. for all spaces and problem instances. All proposed heuristics approximate the BIP solution well. On average, the greedy algorithms is the best performing heuristic. In most cases the difference between the result of the greedy algorithm and the result of the BIP approach is very small or even zero, i.e the algorithms determines the same or an identical performing camera configuration. Both dual sampling

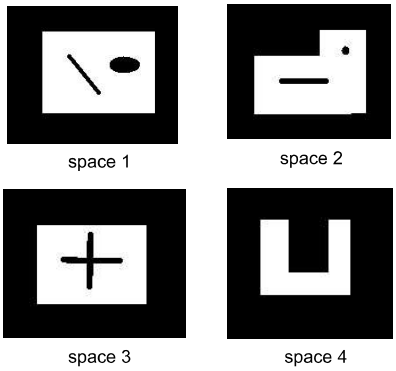
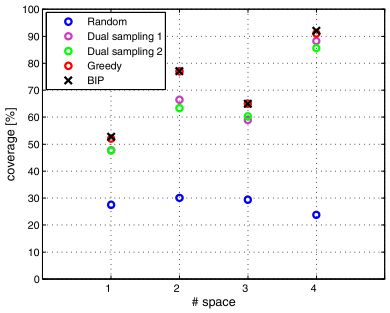


Figure 11: Spaces used for the evaluation

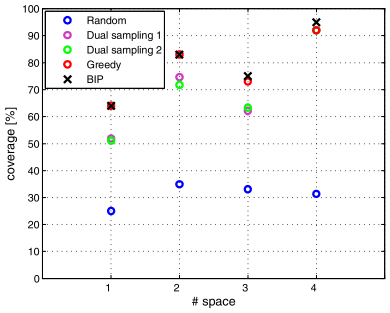
algorithms perform almost equally well but sightly worse than the greedy algorithm in most experiments.
It should be noted, that the BIP algorithm obtains only the optimum solution based on the discrete input to the problems, i.e. based on the given point sets. The average performance of the second version of the dual sampling algorithms is identical or worse than the BIP solution in all our experiments. However in some rarely occuring cases, the resulting configuration obtained by this version of the dual sampling algorithm had a better performance than the BIP solution. This can be explained as follows: This algorithm takes as an input only the control points, it samples the camera positions in each iteration depending on the position of the currently chosen control point (see Section 3.2.2 and Figure 8). If only a limited number of camera positions can be chosen due to runtime and memory constraints this approach has the advantage of being able to sample camera positions with a locally higher density. In turn this implies that if a better solution as the one obtained by solving the BIP model may be found, that most probable the density of discrete camera positions in the space is not sufficient to approximate the continuous case. Thus, the second version of the dual sampling approach is especially suited in situations where we have very large spaces and even the other heuristics, i.e. the greedy approach, are not able to sample with a sufficient density due to a lack of the needed computational resources and/or memory resources.
In summary, the results have shown, that the proposed heuristics approximate the optimum solution well and are thus suited in situations, where the BIP model cannot be solved or only for an insufficient number of control points or camera positions and poses, as these algorithms allow for a higher number of samples. In cases of very large rooms the second version of the dual sampling algorithm should be used.
The experiments have only been performed on small rooms with relatively simple geometries, as we are not able to obtain the BIP results on larger problems due to the lack of computational and memory resources, hence we are not able to compare the results with the optimum solution. The evaluation might be slightly different if the experiments are run on larger or more complicated rooms.

5.2 Complex Space Examples

In this section we present some further results obtained by our different approaches on large rooms. It has to be

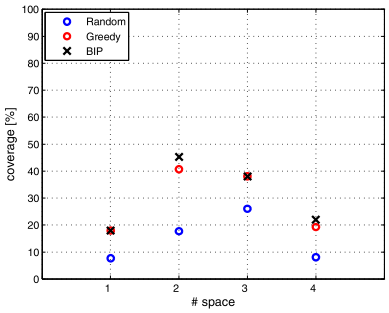


problem 1

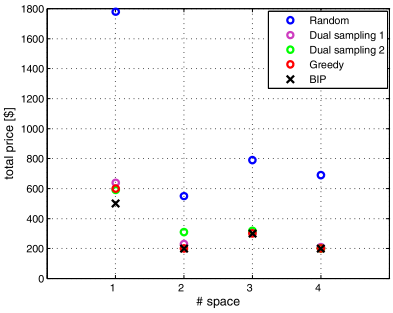


problem 2

Figure 12: Results obtained for problem 1 and 2 using the different proposed algorithms



problem 3



problem 4

Figure 13: Results obtained for problem 3 and 4 using the different proposed algorithms

noted, that although cameras' field-of-views may span over obstacles, those camera can only capture a certain control point, if no obstacle constricts the field-of-view of the visual sensor in the direction of this point.

Figure 14 shows two resulting configurations from the greedy approach. Figure 14 (a) illustrates the placement of six identical cameras such that optimal coverage is achieved; the configuration in Figure 14 (b) shows the result of maximizing coverage for an array of two different camera types with a maximum total cost of $700, where the camera type with the smaller field of view costs $60 and the other $100.

Figure 15 shows results obtained using the two respective versions of the dual sampling algorithm to solve problem 4, i.e. a required minimum percentage of coverage is given and the array with the minimum total price is determined that satisfies this constraint. Either one or more types of cameras were available.

Given fixed positions and camera types for eight cameras in the example space, Figure 16 shows the configuration that maximizes the percentage of covered space obtained by the BIP approach.

The resulting configurations differ if the underlying importance distributions change. This is shown in Figure 17, the associated importance distribution is shown in the top image, the resulting placement that maximizes coverage placing five cameras in the bottom images. It can be seen that more important regions are covered first in the left configuration, whereas in the right placement, due to the equal weighting of all regions, the camera configuration aims to maximize the total area covered.

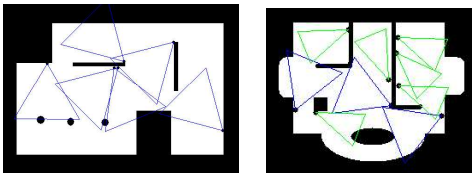
6. POSSIBLE EXTENSIONS

The problems and models considered in this paper can be extended easily.

A more complex field-of-view model that e.g. includes the focus, could be used. Another aspect is that right now we define coverage of a point without accounting for the differents direction a surface through that point could adopt with respect to the covering camera. This could be done by introducing for each control point a number of different directions and checking for each of these directions if coverage is achieved. This requires a slightly different definition of the field-of-view of a camera. The total coverage of the space may then either be calculated summing up over all of those directions and points, or by summing up the average coverage over the directions per control point.

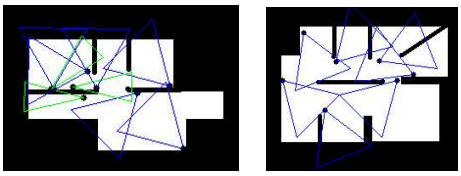
The extension of the BIP as well as the other algorithms to the third dimension is straightforward, but has been excluded in the discussion due to space limitations. We have already proposed a 3D camera model in [9].

In our space definiton we only distinguish between space and obstacles/background. There may occur practical situations, where regions do not have to be covered but they do also not restric the field-of-view of the cameras, i.e. they are no obstacles. An example for such a region is a table in office evironments and a face recognition or person identification application: tables are not restricting the field-of-view of the cameras mounted on walls, but the regions they cover are not interesting for the applicationa. The effect of those regions may be easily modeled by setting the importance weights to zero in these regions, thus no control points will be computed at these locations.



(a) (b)

Figure 14: Configurations resulting from the greedy approach for maximizing coverage for six identical cameras (a) and maximizing coverage assuming two different camera types and a maximum total cost of the sensor array (b)



(a) (b)

Figure 15: Configurations resulting from the dual sampling algorithm version 1 (a) and 2 (b): minimizing the total cost of the visual sensor array while covering 70% of the space assuming two different types of cameras (a) or covering 50% of the space assuming identical cameras (b)

For some applications it may also be desirable to cover regions of the space at different resolution level [5]. Therefore different sampling frequencies f_s need to be defined by the user and assigned to regions of the space. The camera model and the algorithms need to be modified accordingly.

7. CONCLUSIONS

In this paper we addressed the issue of appropriate visual sensor placement with respect to coverage. We formulated and considered four different problems. Different approaches to solve these problems have been presented: BIP models that determine a global optimal solution and various heuristics that approximate this optimum. Experimental evaluation showed the suitability of the algorithms and practicallity of the approaches. A user interface has been developed to support and ease entering and editing spaces, the optimization problem, and the setup parameters. Future work will be modifying and applying the approaches to the 3D case and introducing more complex field-of-view and coverage constraints.

8. ACKNOWLEDGEMENT

The authors would like to thank Benedikt Gleich for his help in implementing the user interface.

9. REFERENCES

[1] *http://mmc36.informatik.uni-augsburg.de/mediawiki/index.php/Research.*

[2] Lingo modeling language and solver. *LINDO Systems Inc*, http://lindo.com/products/lingo/lingom.html.

[3] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transaction on Computers*, 51(12):1448–1453, 2002.

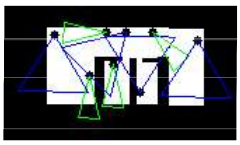
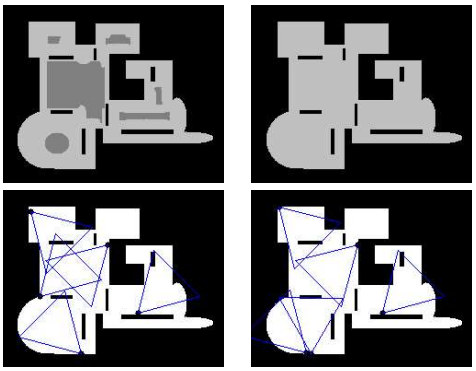


Figure 16: Poses resulting by solving the BIP model for maximizing coverage given fixed sensor positions and types



(a) (b)

Figure 17: Configurations resulting from the greedy approach for the same space (bottom row) but different underlying importance distributions (top row) for maximizing coverage given five identical cameras

[4] X. Chen. *Design of many-camera tracking systems for scalability and efficient resource allocation.* PhD thesis, 2002.

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

[6] S. Fleishman, D. Cohen-Or, and D. Lischinski. Automatic camera placement for image-based modeling. *Computer Graphics Forum*, 19(2):101–110, 2000.

[7] H. Gonzalez-Banos and J. Latombe. Planning robot motions for range-image acquisition and automatic 3d model construction. In *Proc. AAAI Fall Symp.,AAAI Press*, 1998.

[8] E. Hörster and R. Lienhart. Approximating optimal visual sensor placement. In *IEEE ICME2006*, 2006.

[9] E. Hörster and R. Lienhart. Calibrating and optimizing poses of visual sensors in distributed platforms. *ACM Multimedia Systems Journal, Special Issue on Multimedia Surveillance System*, to appear.

[10] P. N. M. Berkelaar and K. Eikland. lpsolve: Open souce (mixed-integer) linear programming system. *Eindhoven U. of Technology*, http://groups.yahoo.com/group/lp_solve/files/Version5.5/.

[11] A. Mittal and L. Davis. Visibility analysis and sensor planning in dynamic environments. In *ECCV04*, pages Vol I: 175–189, 2004.

[12] G. Olague and P. Mohr. Optimal camera placement for accurate reconstruction. *PR*, 35(4):927–944, 2002.

[13] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, 1987.

[14] S. Sahni and X. Xu. Algorithms for wireless sensor networks. *Intl. Journal on Distributed Sensor Networks*, 1(1):35–56, 2005.

[15] J. Wang and N. Zhong. Efficient point coverage in wireless sensor networks. *Journal of Combinatorial Optimization*, to appear.

[16] H. Williams. *Model building in mathematical programming.* J. Wiley, New York, 1985. second edition.

[17] Y. Zou and K. Chakrabarty. Sensor deployment and target localization in distributed sensor networks. *Trans. on Embedded Computing Sys.*, 3(1):61–91, 2004.