Optimal Placement of 3D Sensors Considering Range and Field of View

Joacim Dybedal* and Geir Hovland*

*University of Agder
Faculty of Engineering and Science
Jon Lilletunsvei 9, 4879 Grimstad, Norway

Abstract **This paper describes a novel approach to the problem of optimal placement of 3D sensors in a specified volume of interest. The coverage area of the sensors is modelled as a cone having limited field of view and range. The volume of interest is divided into many, smaller cubes each having a set of associated Boolean and continuous variables. The proposed method could be easily extended to handle the case where certain sub-volumes must be covered by several sensors (redundancy), for example ex-zones, regions where humans are not allowed to enter or regions where machine movement may obstruct the view of a single sensor. The optimisation problem is formulated as a Mixed-Integer Linear Program (MILP) utilising logical constraints and piecewise linearisation of nonlinear functions. The final MILP problem is solved using the Cplex solver interfaced with Matlab.**

A.1 Introduction

The use of 3D sensors and algorithms is a key enabler in autonomous systems. Currently the automotive and self-driving vehicles industry is leading the development, see for example [A1], [A2] and [A3]. Typical sensors used are radar, lidar, GNSS, vehicle odometry and computer vision. In this paper the optimal placement of 3D sensors to cover a volume of interest is considered. Fig. A.1 shows an offshore drilling rig, which is one of the intended application domains for the work presented in this paper.

The optimal placement of 3D sensors is a challenging problem. Most of the publications found in the open literature are limited to solving the 2D problem, for example solving a surveillance problem of a 2D floor plan. Mixed-integer programming is one of the approches successfully used in solving the 2D problem. Some previous work has attempted solving the 3D problem by using heuristic approaches such as Genetic Algorithms. However, such approaches usually end up in a local minimum.

Figure A.1: Example of candidate volume for 3D instrumentation in an offshore drilling rig. Photo courtesy National Oilwell Varco.

In [A4] it was stated that "although the discovery of an algorithm that can solve the most general case of the camera layout problem for a given volume of interest is highly desirable, it may prove quite challenging. The authors therefore focused on a more manageable subclass of this general problem that can be formulated in terms of planar regions that are typical of a building floor plan. The region was approximated by a polygon. This was a valid assumption since most buildings and floor plans consist of polygonal shapes or can be approximated by a collection of polygons. The problem then became to reliably compute a camera layout given a floor plan to be observed, approximated by a polygon. A solution to this problem was obtained via binary optimisation over a discrete problem space". The work is limited to 2D, and the authors write: "In future work, we hope to pursue solutions

to the optimisation in the continuous space as opposed to the discrete one".

In the paper [A5], the problem of optimally placing a minimum number of distributed sensors to fulfil and optimise task requirements in a 2D environment was addressed. As stated in [A5]: "Overall, the research focus of indoor localisation lies on the development of signal extraction and localisation methods. Sensor placement is often done by hand, using the system developers best guess. Thus the resulting localisation error is neither calculated nor considered". In [A5] the problem was formulated in a discrete and continuous search space. The discrete formulations were evaluated using Binary Integer Programming (BIP) and Mixed Integer Programming (MIP). The continuous formulation was evaluated using nonlinear programming (NLP) methods. All evaluations were done using the properties of a visual sensor system that exploited the thermal infrared radiation of humans for indoor localisation. All the presented methods were implemented using a 2D environment description and the resulting problems were solved respectively using MIP, BIP and NLP solvers. The results were compared against each other, taking their exactness and solving speed into account. An extension of the proposed methods to a 3D environment would be possible using the appropriate visibility calculations but was not part of the work.

In [A6] the following was stated: "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". The approaches taken in [A6] were subdivided into algorithms which gave a global optimal solution but were complex and time/memory consuming, and heuristics which solved the problem in reasonable time and with reasonable complexity. The main contributions of [A6] were: 1) Space was sampled according to an underlying importance distribution instead of using a regular grid of points. 2) A linear programming model for each problem was presented which gave an optimal solution to the respective problem. It was shown how to reduce the number of variables and constraints significantly, thus enabling an optimal solution for larger problems. 3) Several heuristics were proposed to approximate the optimal solution of the different camera placement problems. 4) An interface that enables the user to comfortably enter and edit the space, the optimisation problems as well as the other setup parameters was presented. 5) An experimental and competitive evaluation of the different approaches was given showing the different algorithms' specific advantages. In [A6] it was stated: Future work will be modifying and applying the approaches to the 3D case and introducing more complex field-of-view and coverage constraints.

In [A7] it was stated: "Although there are many studies about coverage for wireless multimedia sensor networks, most of them are based on two-dimensional terrain

assumptions. However, particularly for outdoor applications, three-dimensional (3-D) terrain structure affects the performance of the WMSN remarkably". The optimisation method used in the paper is a hybrid heuristic Genetic-Algorithm. The optimisation formulation was found to be NP-complete and for such problems there is no known efficient way to locate a solution. Hence, the authors implemented a GA-based heuristic approach, but such approaches usually end up in a local minimum.

In [A8] the problem of 3D camera placement for optical motion capture systems was studied. Successful performance depended on points being visible from at least two cameras (redundancy) and on the accuracy of the triangulation. The paper introduced and compared two methods for camera placement. The first method was based on a metric that computed target point visibility in the presence of dynamic occlusion. The second method was based on the distribution of views of target points. Algorithms, based on simulated annealing, were introduced for estimating the optimal configuration of cameras for the two metrics and a given distribution of target points. The running time for one example took about 5 hours to accomplish over 62,000 iterations. The goal in [A8] was to find a configuration that was very good, but not necessarily optimal.

The previous work referenced demonstrates that the 3D sensor placement problem is a challenging problem, and to the authors knowledge, no optimal and efficient solution to this problem has previously been published.

The paper is organised as follows: Section A.2 presents the required constraints required to formulate the problem as a Mixed-Integer Linear Program. Section A.3 presents the problem formulation. Section A.4 presents the results from two case studies, while Section A.5 presents discussion and conclusions.

A.2 Optimisation Method

This section contains some of the 'building blocks' required to formulate the problem as a Mixed-Integer Linear Program, see for example [A9]. Section A.2.1 shows how a non-linear function can be approximated by several piecewise linear line segments. The accuracy of the approximation can be adjusted by varying the number of line segments $i \in \{1, \cdots, N_L\}$. Section A.2.2 to A.2.4 show how to implement three different logical constraints (IF-THEN) rules relating continuous and integer variables.

A.2.1 Linearisation of Nonlinear Function

Line segment coefficients:

$$a_i = \frac{d}{dx_i} f(x_i) \tag{A.1}$$

$$b_i = f(x_i) - a_i x_i \tag{A.2}$$

where $f(x_i)$ is the nonlinear function to be approximated, a_i is the slope of the line and b_i the offset.

Constraints:

$$y_i = a_i x + b_i \tag{A.3}$$

$$\left(x_i - \frac{\delta}{2} \leq x \leq x_i + \frac{\delta}{2} - \epsilon \right) \implies d_i \tag{A.4}$$

$$z_i = d_i y_i \tag{A.5}$$

$$y = \sum z_i \tag{A.6}$$

$$\sum d_i = 1 \tag{A.7}$$

where $i \in \{1, \cdots, N_n\}$ is an index to the ith line segment of the linearisation. A total of N_n new help variables y_i (continuous), d_i (Boolean) and z_i (continuous) are introduced.

A.2.2 Implication 1

Constraints:

$$(f_1(x) \leq x \leq f_2(x)) \implies d \tag{A.8}$$

is equivalent to:

$$(m - \epsilon) b_1 \leq (f_1(x) - x) - \epsilon \tag{A.9}$$

$$(m - \epsilon) b_2 \leq (x - f_2(x)) - \epsilon \tag{A.10}$$

$$b_1 + b_2 - d \leq 1 \tag{A.11}$$

where m is a constant smaller than both $f_1(x)$ and $f_2(x)$ while ϵ is a small positive constant. In this paper $\epsilon = 1e - 4$ is used. d is a Boolean variable, while x, $f_1(x)$ and $f_2(x)$ are continuous.

A.2.3 Implication 2

Constraints:

$$(f(x) \geq k) \implies d = 0 \tag{A.12}$$

is equivalent to:

$$-(\epsilon + M)(1 - d) \leq -f(x) - k - \epsilon \tag{A.13}$$

where M is a constant larger than $f(x)$. d is a Boolean variable, while x and $f(x)$ are continuous.

A.2.4 Implication 3

Constraints:

$$\sum_{i=1}^{n-1} d_i = 0 \implies d_n = 0 \tag{A.14}$$

is equivalent to:

$$-d_1 - d_2 - \cdots + d_n \leq 0 \tag{A.15}$$

where d_i and d are Boolean variables.

A.3 Problem Formulation

The volume covered by a 3D sensor is modelled as a cone having a limited field of view S_f and range S_r as illustrated in Fig. A.2. The variables A and Q in the figure describe the position (x,y,z) and orientation (quaternion Q) of the sensor. A point P lies inside the cone covered by the sensor if the angle q is smaller than the field of view S_f and if the distance between the two points A and P is less than S_r.

The entire volume is divided into smaller cubes. For each cube i, the variables in Table A.1 are defined, where $i \in \{1, \cdots, N_c\}$, $s \in \{1, \cdots, N_s\}$, N_s is a constant equal to the number of sensors and N_c equals to the number of smaller cubes covering the volume.

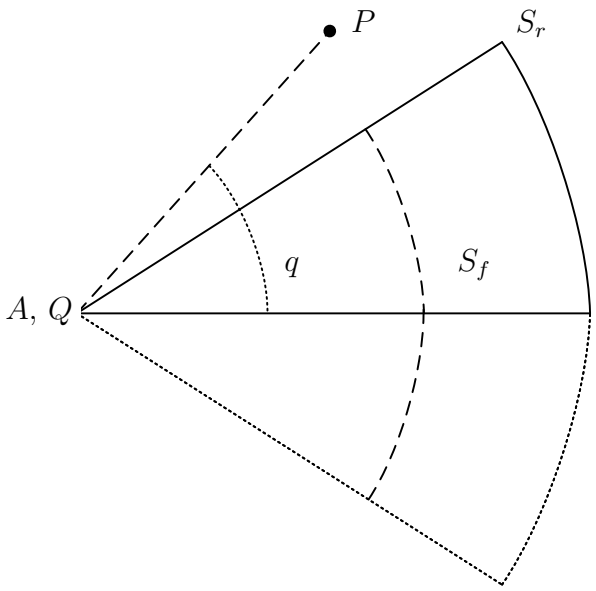

Figure A.2: Illustration of sensor range (S_r) and field of view (S_f). A and Q describe the sensor's location and orientation (quaternion) , respectively. P is a point outside the sensor's field of view and q is the angle between the sensor's center line and the line PA.

The following constraints are used in the optimisation problem.

$$0 \leq S_{x,s} \leq L \tag{A.16}$$

$$PA_{x_i,s} = x_i - S_{x,s} \tag{A.17}$$

$$PA_{y_i,s} = y_i - S_{y,s} \tag{A.18}$$

$$PA_{z_i,s} = z_i - S_{z,s} \tag{A.19}$$

$$q_{i,s} = f_1(PA_{x_i,s}, PA_{y_i,s}, PA_{z_i,s}) \tag{A.20}$$

$$L_{i,s}^2 = PA_{x_i,s}^2 + PA_{y_i,s}^2 + PA_{z_i,s}^2 \tag{A.21}$$

$$L_{i,s}^2 >= S_{r,s}^2 \implies b_{i,s} = 0 \tag{A.22}$$

$$q_{i,s} >= S_{f,s} \implies b_{i,s} = 0 \tag{A.23}$$

$$\sum_{s=1}^{N_s} b_{i,s} = 0 \implies b_i = 0 \tag{A.24}$$

Note that in this set of constraints, the free variables to optimise are the $S_{x,s}$, $S_{y,s}$ or $S_{z,s}$ locations of the sensors, but only one of them for each sensor. Eq. (A.21) (the linearisation of the square function) is defined only for a one-dimensional function.

In eq. (A.20) the nonlinear function $f_1(x_1, x_2, x_3)$ equals

$$L = \sqrt{x_1^2 + x_2^2 + x_3^2} \tag{A.25}$$

$$f_1(\cdot) = \arccos\left(D_1 x_1/L + D_2 x_2/L + D_3 x_3/L\right) \tag{A.26}$$

Table A.1: MILP Variable definitions.

b_i	Boolean variable for cube i
$b_{i,s}$	Boolean variable for each cube i and each sensor s
x_i	x-location of centre of cube i
y_i	y-location of centre of cube i
z_i	z-location of centre of cube i
$S_{x,s}$	x-location of sensor s
$S_{y,s}$	y-location of sensor s
$S_{z,s}$	z-location of sensor s
$PA_{x_i,s}$	The x-distance between sensor s and cube i
$PA_{y_i,s}$	The y-distance between sensor s and cube i
$PA_{z_i,s}$	The z-distance between sensor s and cube i
$q_{i,s}$	Angle between sensor s and cube i
$L_{i,s}$	Distance between sensor s and cube i

where D_1, D_2 and D_3 are the vector components corresponding to the sensor orientation (quaternion Q) illustrated by the cone's centre line shown in Fig. A.2.

The two nonlinear functions f_1 and $(\cdot)^2$ in eqs. (A.20)-(A.21), respectively, are linearised using the approach specified in eqs. (A.1)-(A.7). Fig A.3 shows the linearisation of the acos function in eq. (A.26). Fig A.4 shows the corresponding errors caused by the linearisation for $N_n = 10$ and $N_n = 20$. Fig A.4 shows that the error caused by linearisation can be made small, but the price is the introduction of additional help variables and a larger optimisation problem to solve when N_n increases.

The logical IF-THEN type constraints in eq. (A.22) and (A.23) are implemented using Implication 2 defined in Section A.2.3 while the IF-THEN type constraint in eq. (A.24) is implemented using Implication 3 defined in Section A.2.4. Eq. (A.24) means that if no sensor covers the cube i, then the Boolean variable for that cube is set to zero.

The total number of constraints for the optimisation problem becomes equal to $((32N_n + 14)N_s + 2)N_c$, where N_n is the number of line segments in the linearisation functions, N_c number of cubes and N_s number of sensors. Eq. (A.24) gives two constraints per cube and for each sensor there are $32N_n + 14$ constraints per cube. Although the number of constraints grow linearly with N_c, N_c increases with $\left(\frac{1}{L}\right)^3$, which makes scaling of the proposed method a challenge.

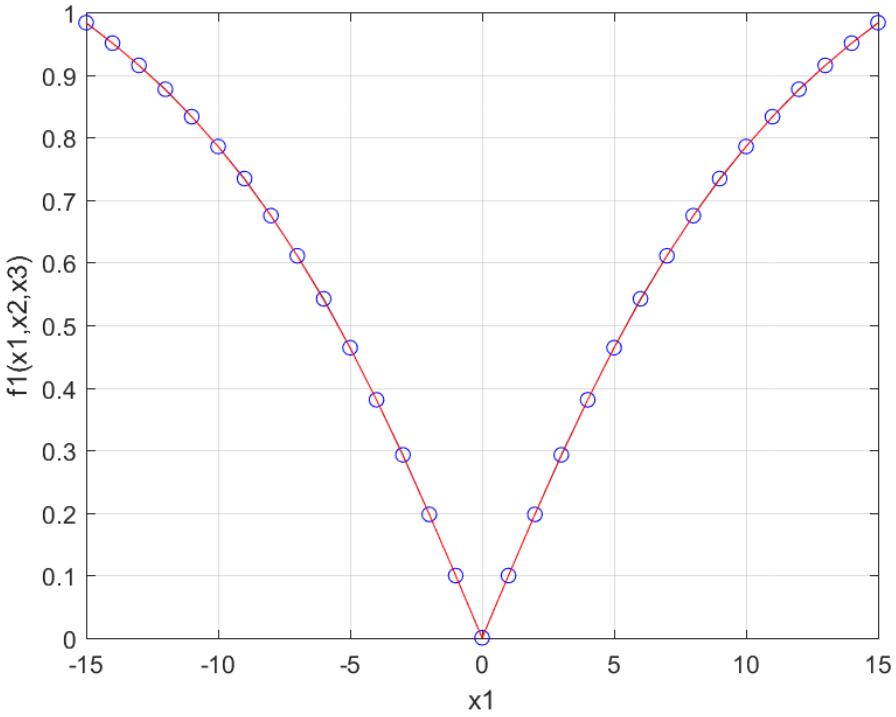

Figure A.3: Plot of eq. (A.26) when $N_n = 20$ between $-15 \leq x_1 \leq 15$. Blue circles: exact sample points. Red: Piecewise linear line segments.

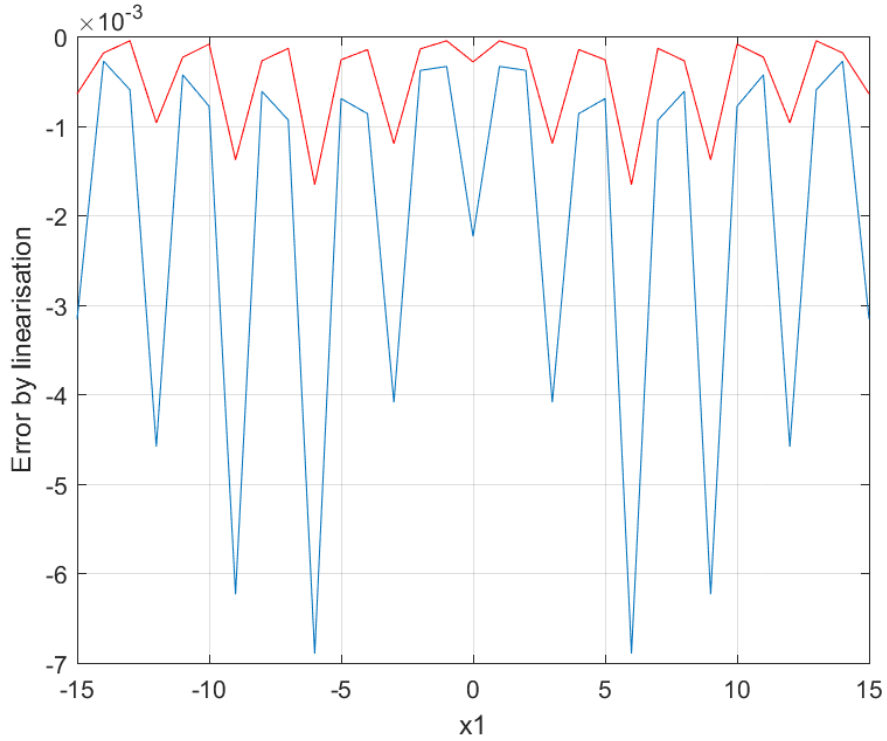

Figure A.4: Error plot corresponding to Fig. A.3. Blue: $N_n = 10$, Red: $N_n = 20$.

A.4 Case Studies

This section contains two case studies. First, a case study containing two opposing sensors is presented, with $N_n = 20$ piecewise line-segments in the nonlinear function approximations. Second, a case study with four sensors and $N_n = 10$ piecewise line-segments is presented.

A.4.1 Case Study I

In this first case study a volume with dimensions $10 \times 10 \times 10\ m^3$ is considered. For illustration purposes, only $N_s = 2$ sensors are specified. The locations, range and field of view of the two sensors are given as follows.

$$S_{y,1} = 10 \tag{A.27}$$

$$S_{y,2} = 0 \tag{A.28}$$

$$S_{z,1} = S_{z,2} = 10 \tag{A.29}$$

$$S_{r,1} = S_{r,2} = 8 \tag{A.30}$$

$$S_{f,1} = S_{f,2} = \frac{\pi}{10} \tag{A.31}$$

$S_{x,1}$ and $S_{x,2}$ are the free variables to optimize. The orientations of the two sensors are specified as follows:

Sensor 1: $\text{RotZ}\left(-\frac{\pi}{2}\right)$, $\text{RotY}\left(\frac{\pi}{4}\right)$
Sensor 2: $\text{RotZ}\left(\frac{\pi}{2}\right)$, $\text{RotY}\left(\frac{\pi}{4}\right)$

where RotY and RotZ are the standard 3×3 rotation matrices about the Y and Z axes, respectively. The corresponding quaternions with these rotational angles are:

$Q_1 = [0.6533, 0.2706, 0.2706, -0.6533]$
$Q_2 = [0.6533, -0.2706, 0.2706, 0.6533]$

The corresponding direction vectors for the two sensors are then given as:

Sensor 1: $D_1 = 0$, $D_2 = -0.7071$, $D_3 = -0.7071$
Sensor 2: $D_1 = 0$, $D_2 = 0.7071$, $D_3 = -0.7071$

Hence, both sensors are located at the top of the volume, angled 45^o downwards towards the floor, and with opposite Y-directions.

Since two sensors of the specified type are not able to cover the entire volume, the objective function in this case study is defined to maximise the number of cubes i covered by the sensors. Hence,

$$\max \left(\sum_{i=1}^{N_c} b_i \right) \tag{A.32}$$

For the presented case study with $N_s = 2$, $N_n = 20$, a volume of $10 \times 10 \times 10 m^3$

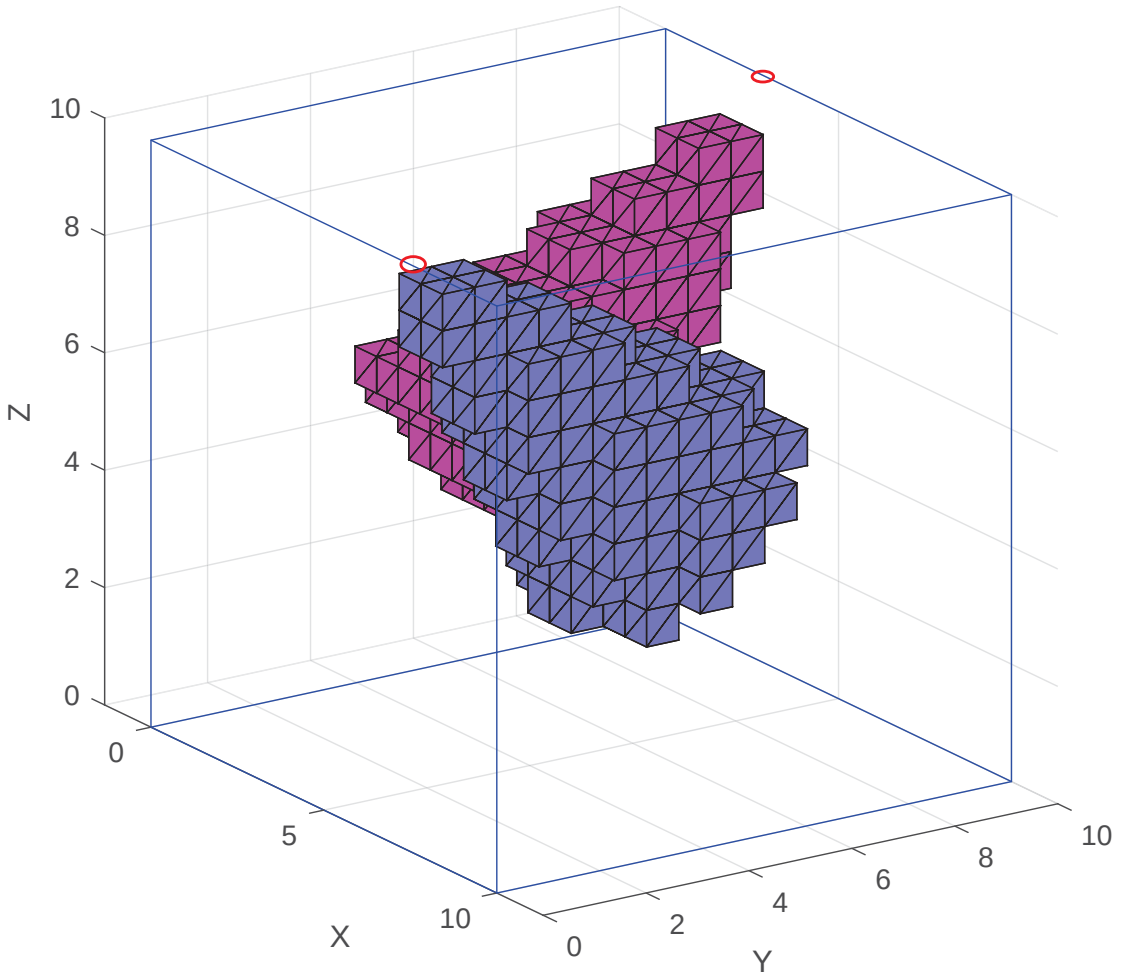

Figure A.5: Result of optimsation with two 3D sensors. Sensor positions are marked with red circles.

and a cube size of $1 \times 1 \times 1m^3$ resulted in a total of $820,002$ variables and a total of $1,310,000$ constraints. Since most of the elements in the constraints are zeros, sparse matrices were used in the Matlab interface to Cplex. The Cplex presolver was able to reduce the problem size down to $216,034$ variables and $417,658$ constraints. Still, Cplex v12.7 required several hours to solve the problem on an Intel i7-6700K CPU @ 4.00GHz with 32GB of RAM.

Fig. A.5 illustrates the result of the optimisation. The cubes inside the volume covered by the two 3D sensors are plotted in the figure with different colours (blue and red). The optimised positions for the two sensors are

$$S_{x,1} = 3.1 \tag{A.33}$$

$$S_{x,2} = 6.9 \tag{A.34}$$

The sensors are positioned in such a way that there is zero overlap between the cubes $b_{i,s}$ and the boundaries of the defined volume is avoided. In this way the maximum

value of the objective function in eq. (A.32) is obtained.

A.4.2 Case Study II

In this second case study the same volume with dimensions $10 \times 10 \times 10\ m^3$ is considered. This time $N_s = 4$ sensors are specified. The locations, range and field of view of the two sensors are given as follows.

$$S_{y,1} = 10, \quad S_{y,2} = 0 \tag{A.35}$$

$$S_{x,3} = 10, \quad S_{x,4} = 0 \tag{A.36}$$

$$S_{z,1} = S_{z,2} = 10 \tag{A.37}$$

$$S_{z,3} = S_{z,4} = 0 \tag{A.38}$$

$$S_{r,1} = S_{r,2} = S_{r,3} = S_{r,4} = 8 \tag{A.39}$$

$$S_{f,1} = S_{f,2} = S_{f,3} = S_{f,4} = \frac{\pi}{10} \tag{A.40}$$

$S_{x,1}$, $S_{x,2}$, $S_{y,3}$ and $S_{y,4}$ are the free variables to optimize. The orientations of the four sensors are specified as follows by the direction vectors:

Sensor 1:	$D_1 = 0,$	$D_2 = -0.7071,$	$D_3 = -0.7071$
Sensor 2:	$D_1 = 0,$	$D_2 = 0.7071,$	$D_3 = -0.7071$
Sensor 3:	$D_1 = -0.7071,$	$D_2 = 0,$	$D_3 = 0.7071$
Sensor 4:	$D_1 = 0.7071,$	$D_2 = 0,$	$D_3 = 0.7071$

The first two sensors are located at the top of the volume, angled 45^o downwards towards the floor, and with opposite Y-directions. The last two sensors are located at the floor level ($S_{z,3} = S_{z,4} = 0$), angled 45^o upwards and with opposite X-directions. In the constraints, the only difference when optimising the sensor y-location instead of the x-location occurs in eq. (A.20) where $PA_{y,s}$ becomes the free variable instead of $PA_{x,s}$, where $PA_{y,s}$ depends on the sensor's y-location $S_{y,s}$ in eq. (A.18). It would also be straightforward to allow Z as the free variable of one or more sensor locations.

The objective function is kept the same as in Case Study I, ie.

$$\max \left(\sum_{i=1}^{N_c} b_i \right) \tag{A.41}$$

For the presented case study with $N_s = 4$, $N_n = 10$, a volume of $10 \times 10 \times 10m^3$ and a cube size of $1.25 \times 1.25 \times 1.25m^3$ resulted in a total of $428,548$ variables and a total of $685,056$ constraints. The Cplex presolver was able to reduce the problem size down to $65,877$ variables and $135,650$ constraints. Several hours were required to solve the problem on an Intel i7-6700K CPU @ 4.00GHz with 32GB of RAM.

Fig. A.6 shows the result of the optimisation. The cubes inside the volume covered by the four 3D sensors are plotted in the figure with different colours. The

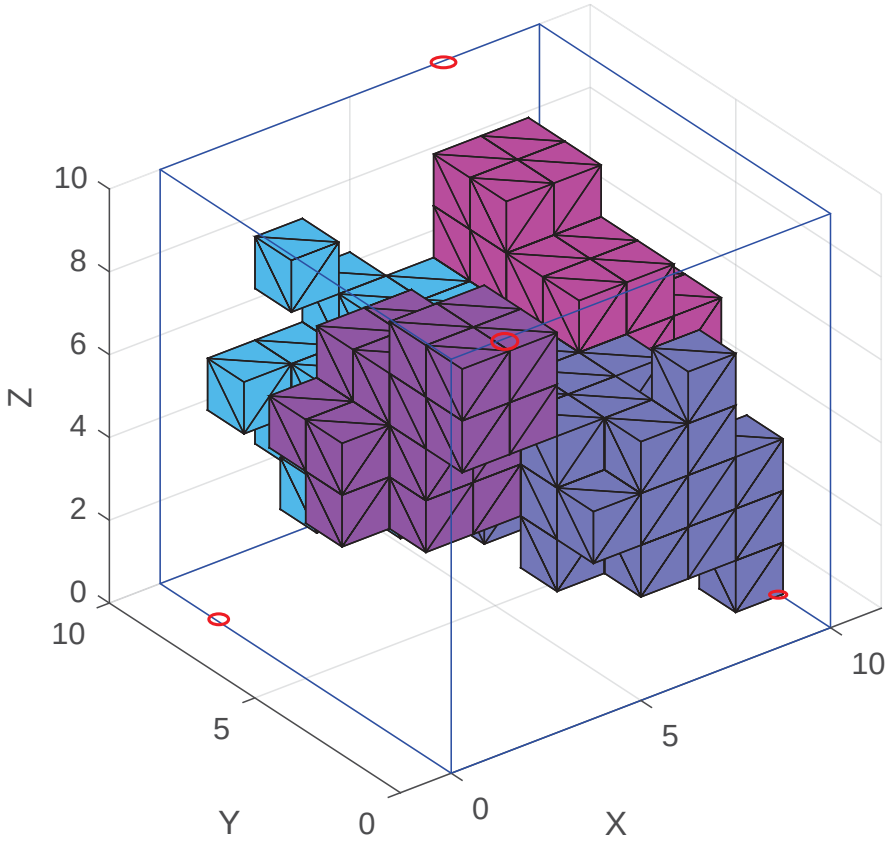

Figure A.6: Result of optimzation with four 3D sensors. Sensor positions are marked with red circles.

optimised positions for the sensors are

$$S_{x,1} = 7.51 \quad S_{x,2} = 2.51 \tag{A.42}$$

$$S_{y,3} = 2.46 \quad S_{y,4} = 7.54 \tag{A.43}$$

The sensors are positioned in such a way that there is zero overlap between the cubes $b_{i,s}$ and the boundaries of the defined volume are avoided. In this way the maximum value of the objective function in eq. (A.41) is obtained.

A.5 Discussion and Conclusion

In this paper the problem of optimal placement of 3D sensors has been addressed based on a Mixed-Integer Linear Programming framework. Logical constraints of IF-THEN type have been converted to the matrix inequality format $\mathbf{Ax} \leq \mathbf{b}$. Two types of nonlinear functions (acos and $(\cdot)^2$) were linearised by introducing a set of help variables (y_i, d_i, z_i). In order to keep the number of help variables to a manageable level, the nonlinear functions had to be chosen such that they were only dependent on one single variable. For this reason, the free variable to optimise for

the sensors in the presented case study where the mounting locations of the sensors in the x-, y- or z-directions. The authors do not think this limitation is a significant restriction of the proposed method in practice. For example, in Fig. A.1 the sensors would typically be mounted on the vertical or horizontal support beams of the rig. The orientation of a 3D sensor is also fixed in the proposed method. Often a gimbal head is used to mount sensors and then the orientation of the sensor can be chosen quite freely. To include the possibility of several different mounting orientations in the optimisation problem, a discrete set of identical sensors can be added to the problem formulation, with the same localisation direction to optimise ($S_{x,s}$, $S_{y,s}$ or $S_{z,s}$), but with different orientation vectors.

In this paper the objective function was chosen to maximise the number of cubes covered by the sensors. This choice was selected mainly because it is easy to verify graphically (Figs. A.5 and A.6) that the optimal solution has no overlap in cubes covered by the different sensors and that the sensors are not mounted near the boundaries of the volume. However, a more realistic objective function in practice would be to minimise the cost of the sensors needed to cover the entire area. In this case several different sensors with different cost, range and field of view would be made available along each possible mounting position and additional constraints would be introduced requiring every cube in the volume to be covered.

The proposed method can easily handle the case where a certain sub-volume must be covered by several sensors (redundancy). For example assume that the sub-volume of cubes $i \in I$ must be covered by at least N sensors. The constraints to specify this requirement would be

$$\forall i \in I \quad \sum_{s=1}^{N_s} b_{i,s} \geq N. \tag{A.44}$$

Examples of sub-volumes requiring redundancy could be explosive zones on a rig, regions where humans are not allowed to enter or regions where machine movement would cause shadow volumes not visible when using only a single sensor.

Even though the proposed method currently takes several hours to solve, the method is not prone to ending up in local minima, which can be the case with more heuristic approaches to solving the 3D sensor placement optimisation problem. The general problem of optimal placement of 3D sensors is challenging. Most literature in this area has limited the study to more manageable subclasses of the problem, such as for example 2D floor plan cases. The reference [A8] also demonstrates that other solution frameworks, in their case simulated annealing, can also take several hours to solve.

One limitation of the presented method is the potentially large number of variables

and constraints required due to the discretisation of the volume of interest into smaller cubes, and the need to linearise the nonlinear functions acos and $(\cdot)^2$ for each cube element. Large sets of constraints and variables in particular, slow down the MILP solver. Future work by the authors will focus on techniques to reduce the number of variables and constraints required and as a result faster solution times. One way in which this could be achieved could be to follow the approach suggested by [A6]: Sample space according to an underlying importance distribution instead of using a regular grid of points.

A.6 Acknowledgment

The research presented in this paper has received funding from the Norwegian Research Council, SFI Offshore Mechatronics, project number 237896, https://sfi.mechatronics.no

References – Paper A

[A1] D. Göhring, M. Wang, M. Schnürmacher, and T. Ganjineh. Radar/lidar sensor fusion for car-following on highways. In *Proc. Intl. Conf. Automation, Robotics and Applications (ICARA)*, 2011.

[A2] Y. Alkhorshid, K. Aryafar, S. Bauer, and G. Wanielik. Road detection through supervised classification. In *Proc. Intl. Conf. Machine Learning and Applications (ICMLA)*, 2016.

[A3] E. Ward and J. Folkesson. Vehicle localization with low cost radar sensors. In *Proc. IEEE Intelligent Vehicles Symposium*, 2016.

[A4] U.M. Erdem and S. Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Computer Vision and Image Understanding*, 103:156–169, 2006.

[A5] Nicolaj Kirchhof. Optimal placement of multiple sensors for localization applications. In *Proc. IEEE Intl. Conf. on Indoor Positioning and Indoor Navigation*, 2013.

[A6] E. Hörster and R. Lienhart. Optimal placement of multiple sensors for localization applications. In *Proc. 4th ACM Intl. Workshop on Video Surveillance and Sensor Networks*, 2006.

[A7] H. Topcuoglu, M. Ermis, I. Bekmezci, and M. Sifyan. A new three-dimensional wireless multimedia sensor network simulation environment for connected coverage problems. *Simulation: Transactions of the Society for Modeling and Simulation International*, 88(1):110–122, 2006.

[A8] P. Rahimian and J.K. Kearney. Optimal Camera Placement for Motion Capture Systems. *IEEE Trans. on Visualization and Computer Graphics*, 23(3):1209–1221, March 2017.

[A9] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.