Automatic Camera Placement for Large Scale Surveillance Networks

Anton van den Hengel Rhys Hill Ben Ward Alex Cichowski
Henry Detmold Chris Madden Anthony Dick John Bastian
{anton,rhys,ben,alex,henry,chris,ard,john}@cs.adelaide.edu.au

Abstract

Automatic placement of surveillance cameras in arbitrary buildings is a challenging task, and also one that is essential for efficient deployment of large scale surveillance networks. Existing approaches for automatic camera placement are either limited to a small number of cameras, or constrained in terms of the building layouts to which they can be applied. This paper describes a new method for determining the best placement for large numbers of cameras within arbitrary building layouts. The method takes as input a 3D model of the building, and uses a genetic algorithm to find a placement that optimises coverage and (if desired) overlap between cameras. Results are reported for an implementation of the method, including its application to a wide variety of complex buildings, both real and synthetic.

1. Introduction

The placement of cameras within a surveillance network critically affects its performance. This paper investigates the problem of finding the optimal arrangement for a set of cameras, and describes software which has been created to achieve this goal. It describes typical industry practice, and explores the state of the art in automated camera placement.

Traditional camera placement methods are heavily reliant upon human expertise, perhaps with the assistance of specialised 3D modelling software [6]. Such a process is workable for networks up to tens of cameras, but rapidly degrades as the scale of the network increases. Recent advancements in intelligent analysis algorithms[12, 4, 13] can use automated or semi-automated techniques to assist security operators, however human experts who focus upon camera placement are often not experienced with such systems and find it difficult to place cameras to utilise them effectively. Thus the paper describes a software system developed to identify the optimal placements for a set of cameras given the shape of the space to be observed, the number and type of cameras available, and the level of automated software assistance desired.

The software uses a 3D model of the space to be observed in order to calculate the fields of view of each of the cameras in the network. By contrast, most existing systems operate on a building floor plan, and calculate visibility only in 2D (see [3] for example). The 3D approach is superior in that it calculates visibility where it is relevant: for example, on the upper torso or head of the target rather than the floor. It also takes into account obstacles to the view of the camera that cannot be represented accurately on a 2D floor plan, because their shape varies with height. Examples of such obstacles include anything that does not stretch uniformly from floor to ceiling, such as signs, partitions, furniture, equipment and even other people.

Using 3D techniques leads to more relevant measures of how "useful" a camera placement is, based upon the potential of cameras to observe actual targets to a required degree of accuracy, rather than just their ability to observe an area. In this paper, we focus on faces as they are currently used in surveillance as a key identifying feature; however it is also possible to use a full human model to allow for analysis based upon the entire individual. Additionally, using 3D models allows for enhanced visualisation of the camera placement results, which gives the network operators better tools for manually deciding on how to place cameras.

Most large scale video surveillance systems currently installed, or being installed, use human experts for camera selection and placement. However, the human camera placement technique is incapable of effectively weighing up the multitude of competing factors which must be considered for even a relatively small network. The optimal placement of cameras for accurate analysis depends upon the level of automation to be used. The required coverage and level of view overlap between neighbouring cameras increases for higher levels of automation. This is due to the limitations of techniques for tracking targets between camera views. Coverage requirements will affect the cost of the system, due to the implied increase in the number of cameras required, and the cabling and computing equipment to needed to accomo-

This work has been partially funded by the Defence Science Technology Organisation and the South Australian Premier's Science and Research Fund.

978-1-4244-5498-3/09/$25.00 ©2009 IEEE

date them.

The rest of this paper is structured as follows. Section 2 describes a range of work from the literature related to camera placement. Section 3 describes the algorithm developed to determine an optimal camera placement for a particular 3D model. In Section 4 we describe the results of testing the placement software upon a range of models from a circular room, to more complex models including a floor plan based model of the Changi international airport in Singapore, and a realistic three storey office building. Finally, Section 5 summarises the contribution of the paper and of the developed software.

2. Related Work

Early literature in the area of camera placement has been modelled upon the *Art Gallery Problem* (AGP) [9] and its variants like the *Floodlight Illumination Problem* (FIP) [1]. The AGP requires the assignment of a minimal set of 'guards' to monitor all points on the boundary of an arbitrary polygon, which for camera placement will be sourced from a 2D floor plan. Efficient algorithms have been developed that yield approximate solutions this problem; however the exact solution has been shown to be NP-hard [9]. The FIP aims to 'illuminate' a polygonal area. Unfortunately, current algorithms designed to solve these problems employ many assumptions that are violated in the camera placement problem. Camera properties, such as field of view or resolution, are not taken into account.

The current literature on the placement of surveillance cameras falls into two categories: those that aim to assist a human operator to manually place cameras, and those that seek to automate the process. The work of Reiffel *et al.* [11] describes a tool called *DOTS*, which provides a graphical interface to manually place cameras within a virtual environment. It provides a floor plan and 3D model over which the camera positions are superimposed, allowing the operator to view the scene as the camera would observe it. Whilst this system allows for the visualisation of the scene and can include furniture and modelled people, it explicitly requires a human expert to optimise the camera positions. Williams *et al.* examine the optimal placement of cameras to reduce occlusions; however mostly in the context of creating detailed object models from dense camera arrays rather than for surveillance [14].

Murray *et al.* examined occlusion as a camera placement driver with large outdoor environments [7]. However object resolution, an important consideration when surveilling large areas, is ignored.

Erdem and Scarloff [3] provide one of the earliest works to look specifically at realistic camera placements. They employ constraints on potentially visible of areas of a floor plan, the most likely resolution at which objects might be observed, and introduced regions where high object resolution is required. However, this approach was again based on a 2D floor plan.

Ram *et al.* [10] formally defined a design methodology for obtaining the optimal set of sensors and their placement. They first define a performance metric for accomplishing a particular task given to the sensors, taking into account their placement. Next, they determine all the combinations of sensors, along with their placement, whose performance metric meets or exceeds some limit, then finally determine and output the combination with the least cost. They outline a cost function that can be used to increase the number of frontal observations of objects , which is beneficial for some surveillance tasks, such as facial recognition.

Zhao and Cheung [17] document an alternate approach based on a concept called *visual tags*. A visual tag is distinctive tag physically attached to a person in the scene. The tag is designed to provide identification information easily, and allow privacy filtering to be automatically carried out on the video from the cameras. Each tag has an associated orientation . An optimal camera placement is achieved in two–dimensions by maximising the number of synthetic tags visible to the cameras, after being spread evenly across the scene. This system also optimises the number of cameras viewing the scene but the results presented are for small rooms.

Horster and Lienhart [5] developed a method that also uses a floor plan, similar to of [3], but allow the user to specify an importance distribution as an overlay of the observable space. This allows the focus to remain upon areas of higher importance or higher traffic. This system is still only two–dimensional, however, and the examples shown in the paper are relatively simple.

Yabuta and Kitazawa [15] present a fast method that optimises camera placement through the use of simplified, block based, scene model. Unlike previous methods, they use variable sized region blocks to determine visibility, and propose to weight these based upon their importance. This is both the major innovation and the major drawback of this method. The use of the block structure makes the method fast, but also render it unable to function in any scene with non–axis–aligned walls, which is very common in real buildings.

Yao *et al.* [16] extend the placement of cameras to consider not only coverage, but also overlap to allow for automated tracking processes. Similar to Erdem *et al.* [2] these are evaluated based upon a 2D floor plan, although they do consider tracking failures in their evaluation. The model of overlap employed in this paper is sophisticated, treating the edges of the camera views differently to allow explicit modelling of camera hand–off. Once again, the example placements shown in the paper are simple.

The previous work presented in this section has a range of different methods, each of which employs a constrained

scene model and, in general, was only demonstrated on small numbers of cameras. However, none of these have addressed nor employed the use of rich 3D models where there may be partial occlusions, such as those that occur due to roof mounted signs and fixtures, furniture or other ground based obstructions. The system presented in this report aims to exceed the state–of–the–art and overcome many of the limitations present within these methods.

3. Automatic Camera Placement

The first, and most important, step in the system is the loading and annotating of the 3D model. Once this step has been completed by the user, the automatic system takes over to produce an optimal camera placement.

3.1. Scene Models

Extracting important information, such as floor and ceiling location, directly from a 3D model is difficult, unless a large number of constraints are applied to the model. This system instead allows to user to supply metadata to assist the interpretation of the model. In particular, the scale and orientation of the model, along with the location of the floors and ceilings, are expected to be provided by the user.

In real buildings, there may be a number of architectural features which complicate the acquisition of model metadata. Stairs and ramps can both cause difficulties. Stairs must simply be handled by selecting each tread as part of the floor. Ramps are detected as part of the floor automatically, if their incline is within a selectable limit. The same applies to ceilings, allowing curved or cathedral ceilings to be incorporated.

Further, the model must be comprehensive, as it is relied upon during camera placement. This means that light fixtures, roof vents, signage, furniture, etc., are all represented in the model. If not, the system may place cameras on or behind one of these fixtures.

3.2. Parameterisation

A camera floating in the air, free of physical constraints, has six parameters to describe its location. Three of these describe its position, the remaining three describe its orientation. The orientation parameters can be seen as three angles: pan, tilt and roll. Pan and tilt are the most important of these, roll less so. A human installer would rarely mount a camera which wasn't level, so the roll parameter can be ignored for the sake of simplicity. The position of the camera can also be constrained since, for an internal surveillance network, cameras will generally be mounted on the ceiling. If the building has a number of levels, then the camera may reside on the ceiling of any one of them.

Cameras also have a number of internal, or intrinsic, parameters. For a surveillance camera, the most important of these is the zoom, or focal length. There are other as well, such as the sensor size of the camera, and the number of pixels the camera can capture. In the current system, the sensor size and number of pixels the camera can capture are chosen by the user.

The most straight–forward parameterisation for a camera position is a three vector, with x, y and z coordinates. Figure 1 shows a 2D example of this parameterisation. A simple solution to this problem is to use the three–vector approach, but correct the location each time it is changed so it lies on a valid surface.

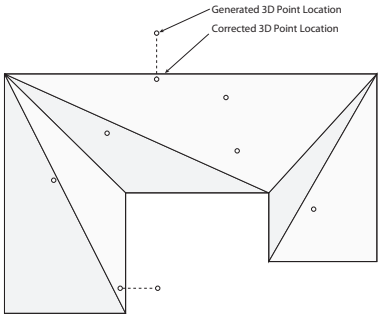


Figure 1. An example of the constrained 3D point parameterisation. Each point is represented by a full three–vector, but when it is generated, it is 'snapped' onto the nearest part of the roof.

This approach is simple and highly flexible, however it has two downsides. It again allows a large number of invalid locations, but these are removed via the correction. The second downside is that the correction itself can be computationally expensive if not implemented carefully. The correction is performed by projecting the point onto the plane of the triangle that contained the originating point, and it is then tested to see if it lies within the boundaries of the triangle. If so, the point is already on the surface, so nothing further needs to be done. If not, the triangle's neighbours are examined, a similar process is performed, and the closest point to the original uncorrected point is taken. An example of this is shown in Figure 2.

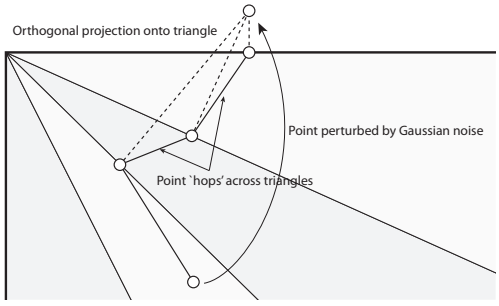


Figure 2. An example of the correction algorithm, when applied to a point that has moved outside the surface.

3.3. Cost Function

In order to estimate a camera placement, a function for evaluating the quality of a particular placement is required. Once such a function is available, an optimiser can be applied to adjust the parameters such that the quality of the solution increases.

To create a specification for this function, the qualities that an ideal solution must posses need to be enumerated:

1. A high degree of coverage of the area under surveillance
2. Overlap in camera views
3. Targets' image sizes fall within a specified range

These different properties can all be measured in a very natural way, that fits well in the problem space. If all the areas of the building where people can stand are identified, a set of markers can be spawned that cover this space. Two different distribution methods were trialled, with no discernible difference: a uniform sampling and a random sampling. The markers were designed to represent a person, and can either be a cube floating at head height, roughly the size of a human head, or a rectangular cuboid the size of a full human.

Each of the desired properties can then be expressed in terms of which of these markers are visible in the cameras. For each person, a counter is stored which records, for a particular camera configuration, how many cameras are able to see it. The coverage metric could simply obtained by counting how many markers are seen by at least one camera. The overlap metric could be considered as the number visible by two or more cameras. The pixel count is obtained by simply rendering the marker into each candidate camera, and only incrementing the counter if the number of pixels it covers is sufficiently high or low. The count can be computed easily using *occlusion queries* [8].

An occlusion query is executed by first generating a depth–map of the scene, shown in Figure 3, as seen by the candidate camera. This map is simply the depth of the scene for each pixel in the cameras view. Once this map has been acquired, an occlusion query is performed by comparing the depth of each imaged pixel in the marker to the depth map. Each imaged pixel that is closer than the corresponding depth map pixel is counted. If this count is greater than zero, the marker is visible.

The value of this count can be used to filter out small, or mostly occluded markers. The visibility test is simply modified to compare to a lower bound, $p_{\min}$. If the number of visible pixels, p, is above this bound, then the marker is visible. This idea can be extended to an upper bound, $p_{\max}$, as well – once a certain number of pixels are visible, it is no longer useful to obtain more. Thus, the final cost function takes on the form of a clamping function, shown below. In practice, ignoring small targets leads to cameras which focus too tightly on the tops of the markers. Thus, small targets are just given a reduced score.

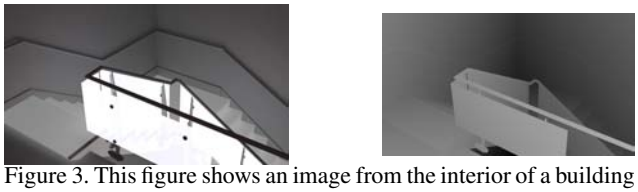


Figure 3. This figure shows an image from the interior of a building on the left, and its associated depth map on the right. The depth map provides a measure of the distance from the camera to each point in the scene.

$$c = (p - p_{\min})/(p_{\max} - p_{\min}) \quad (1)$$

$$\mathcal{C}_{\text{coverage}} = \begin{cases} p \le p_{\min}, & \frac{p}{p_{\min}} \\ p_{\min} \le p \le p_{\max}, & 0.9c + 0.1 \\ p \ge p_{\max} & 1 \end{cases} \quad (2)$$

The overlap metric is treated in a similar way. The number of cameras that can see each marker are counted, and then assigned an appropriate cost. In most cases, the only values of interest are two, three and four or more. If one is returned, then no overlap is present, so the value of the metric is zero. For the remaining cases, cost values were chosen empirically, and are currently given by:

$$\mathcal{C}_{\text{overlap}} = \begin{cases} v \le 1, & 0 \\ v = 2, & 1 \\ v = 3, & 1.5 \\ v \ge 4 & 1.75 \end{cases} \quad (3)$$

where v is the number of cameras viewing the given marker.

The two metrics must be combined into a single value, to provide the final cost for a particular camera configuration. Different scenarios will require camera configurations with more or less overlap, so the user is able to chose the extent to which overlap is favoured over coverage, via a simple slider. The slider produces a value, α, between zero and one, yielding a final cost of:

$$\mathcal{C} = (1 - \alpha)\mathcal{C}_{\text{coverage}} + \alpha\mathcal{C}_{\text{overlap}} \quad (4)$$

3.4. Optimiser

There are many different types of optimiser currently available, and the structure of the problem is the main guide as to which one is most appropriate. In this case, there is no initial solution available, so one must be generated. It is also unlikely that gradients can be computed since the cost function as stated is essentially discrete, and its evaluation is very complex, due to the use of occlusion queries.

Thus, the most appropriate class of optimiser is a search–based algorithm. A commonly choice in this area is a Genetic Algorithm (GA). GAs are able to search very complicated spaces with no prior information, other than the cost function itself. They are also ammenable to customisation, making them more tightly bound to the problem in question, and able to more quickly probe the solution space and locate optima.

For this project, customisation of the mutation and cross–over operators is critical to obtaining good performance. Each member of the population is a set of cameras, so the cross–over operator simply takes two such members and swaps a subset of the cameras from each. The mutation operator uses a Gaussian to perturb some of the camera parameters for any member chosen for mutation. The magnitude of the Gaussian noise is provided by the user, and is based on the range of each parameter.

4. Evaluation

The camera placement tool has been evaluated with a variety of real and synthetic 3D models, and with up to 400 cameras. The focus of testing was to validate the ability of the automatic placement tool to achieve optimal solutions, and also to verify that generated placements respond appropriately to variations in parameters related to the purpose of the surveillance network. Note that determining the true optimal camera placement is very difficult, and would likely require a time–consuming, brute–force search.

Figure 4 shows an example of a typical surveillance application – namely a small office. The results shown are drawn from a placement of eight cameras. The system has placed a camera in each office, and then staggered the remaining cameras throughout the space, resulting in excellent overlap and coverage. The placement of the two blue cameras is interesting, as it maximises the overlap in a very small region. The placement of the purple camera is also of note, as a human designer would be unlikely to select that location, which optimises overlap within the corridor.

The second example is a 3D model of a typical office building, with representative internal structure. Figure 5 shows an overview of the results from each level. The benefit of using the 3D model is apparent in this example as the system is able to both draw upon and work around the internal structure of the building. The resulting placement is very effective, composed of only 32 cameras. The black areas of the model are not visible, since they lie behind walls. An effective placement would likely be generated with fewer cameras, at the cost of reduced overlap.

5. Conclusion

Achieving a cost effective placement of the cameras in a surveillance network is challenging, particularly for large scale networks. It is also critical to the long term performance, effectiveness and maintainability of the network. The approach described in this report both significantly advances the state of the art and delivers a valuable tool that has been demonstrated to calculate automatic camera placements for a variety of situations. The major contribution of the approach is that it allows an operator to rapidly create a camera placement for a new surveillance network installation. Results reported for this new hybrid approach demonstrate its enhanced efficiency when compared to existing practice.

There is significant scope for further work on the algorithm. Selecting the number of cameras automatically is a high priority, along with selecting a more efficient minimiser. Even without these extensions, the algorithm represents a significant improvement over the state of the art.

References

[1] P. Bose, L. Guibas, A. Lubiw, M. Overmars, D. Souvaine, and J. Urrutia. The floodlight problem. *International Journal of Computational Geometry and Applications*, pages 153–163, 1997.

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

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

[4] R. Hill, A. van den Hengel, A. R. Dick, A. Cichowski, and H. Detmold. Empirical evaluation of the exclusion approach to estimating camera overlap, 2008. Proceedings of the International Conference on Distributed Smart Cameras.

[5] E. Horster and R. Lienhart. On the optimal placement of multiple visual sensors, 2006. International Workshop on Video Surveillance and Sensor Networks.

[6] L3-Communications. Camera placement service, 2009.

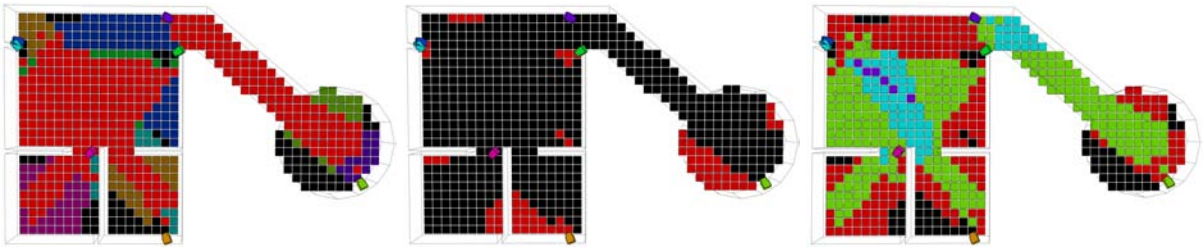
[7] A. Murray, K. Kim, J. Davis, R. Machiraju, and R. Parent. Coverage optimization to support security monitoring. *Computers, Environment and Urban Systems*, 31(2):133–147, 2007.

[8] Opengl, D. Shreiner, M. Woo, J. Neider, and T. Davis. *OpenGL(R) Programming Guide : The Official Guide to Learning OpenGL(R), Version 2 (5th Edition)*. Addison-Wesley Professional, August 2005.

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

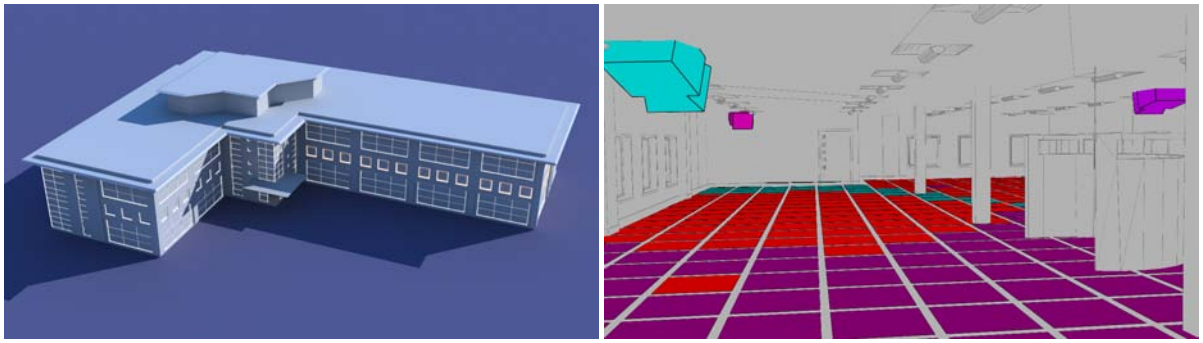
[10] S. Ram, K. Ramakrishnan, P. Atrey, V. Singh, and M. Kankanhalli. A design methodology for selection and placement of sensors in multimedia systems, 2006. International Workshop on Video Surveillance and Sensor Networks.

[11] E. Rieffel, A. Girgensohn, D. Kimber, and T. C. Q. Liu. Geometric tools for multicamera surveillance systems, 2007. International Conference on Distributed Smart Cameras.

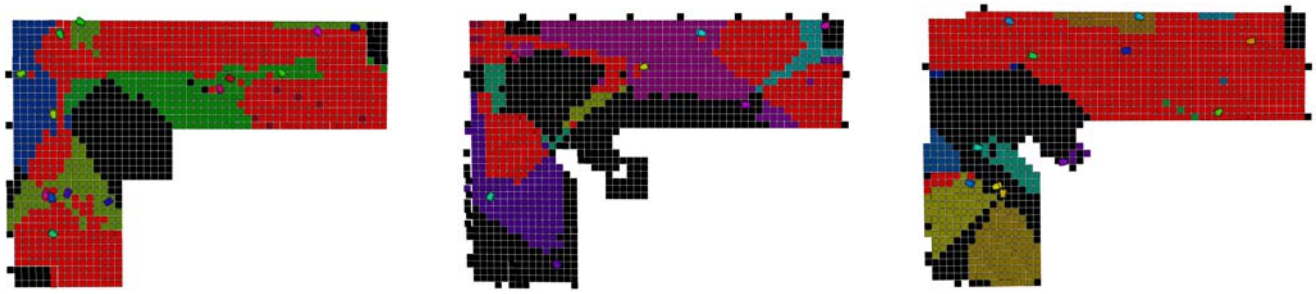


(a) Coloured by camera (b) Unobserved regions (c) Overlapped regions

Figure 4. This figure shows a synthetic building that was designed to resemble a typical small office. The first image shows the result, with a colour overlay showing which camera can view each marker. The red colour indicates overlap. The second figure shows areas that cannot be seen, whilst the final figure shows degrees of overlap.



(a) Model (b) Interior



(c) Level 1 (d) Level 2 (e) Level 3

Figure 5. The final, most realistic, test case is shown in this figure. It is a three storey office building, with modern interior design and no furniture. This figure show the result, as viewed from above, on each level of the office building model, along with an example interior view. The effect of the internal structure of the building can be clearly seen.

[12] M. Valera Espina and S. A. Velastin. Intelligent distributed surveillance systems: A review. *IEE Proceedings - Vision, Image and Signal Processing*, 152(2):192–204, April 2005.

[13] A. van den Hengel, A. R. Dick, H. Detmold, A. Cichowski, and R. Hill. Finding camera overlap in large surveillance networks, 2007. Proceedings of the 8th Asian Conference on Computer Vision.

[14] J. Williams and W. Lee. Interactive virtual simulation for multiple camera placement, 2006. IEEE Workshop on Haptic Audio Visual Environments and their Applications.

[15] K. Yabuta and H. Kitazawa. Optimum camera placement considering camera specification for security monitoring, 2008. International Symposium on Circuits and Systems.

[16] Y. Yao, C. Chen, B. Abidi, D. Page, B. Abidi, and M. Abidi. Sensor planning for automated and persistent object tracking with multiple cameras. In *IEEE Int. Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[17] J. Zhao and S. Cheung. Multi-camera surveillance with visual tagging and generic camera placement, 2007. International Conference on Distributed Smart Cameras.