EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

Designing Camera Networks by Convex Quadratic Programming

Bernard Ghanem, Yuanhao Cao, and Peter Wonka

King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia

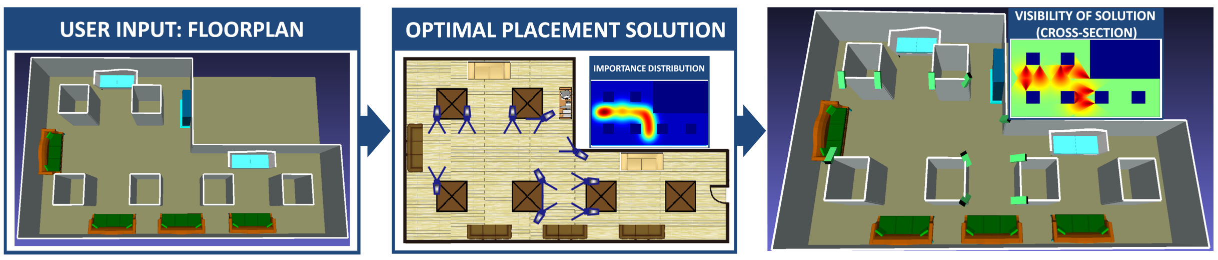
USER INPUT: FLOORPLAN
OPTIMAL PLACEMENT SOLUTION
IMPORTANCE DISTRIBUTION
VISIBILITY OF SOLUTION (CROSS-SECTION)

Figure 1: Example result of our proposed optimal camera placement framework. In a particular scenario, the user inputs a 3D floorplan that can be generated by processing an overhead 2D floorplan using the user-friendly GUI we developed. After setting certain camera parameters (e.g. field-of-view and depth-of-field), our approach computes a placement solution that can either maximize 3D floorplan coverage with a limited number of cameras or minimize the number of cameras needed to cover the entire floorplan. Unlike other placement methods, our approach is computationally efficient because it solves a constrained convex quadratic program. It also allows pairwise camera interactions to be directly encoded, which is quite useful for multi-view applications, such as 3D reconstruction and surveillance.

Abstract

In this paper, we study the problem of automatic camera placement for computer graphics and computer vision applications. We extend the problem formulations of previous work by proposing a novel way to incorporate visibility constraints and camera-to-camera relationships. For example, the placement solution can be encouraged to have cameras that image the same important locations from different viewing directions, which can enable reconstruction and surveillance tasks to perform better. We show that the general camera placement problem can be formulated mathematically as a convex binary quadratic program (BQP) under linear constraints. Moreover, we propose an optimization strategy with a favorable trade-off between speed and solution quality. Our solution is almost as fast as a greedy treatment of the problem, but the quality is significantly higher, so much so that it is comparable to exact solutions that take orders of magnitude more computation time. Because it is computationally attractive, our method also allows users to explore the space *of solutions for variations in input parameters. To evaluate its effectiveness, we show a range of 3D results on real-world floorplans (garage, hotel, mall, and airport).*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

We study the design of camera networks for applications in computer graphics and computer vision. The main motivation is the setup of surveillance systems for large buildings such as malls, airports, stadiums, factories, office buildings, or outdoor spaces, such as sidewalks and streets. In addition to the design of traditional surveillance systems that assume a human analyst processing the data, we are also interested in designing denser camera networks that would allow for automatic tracking, 3D reconstruction, change detection, or action recognition. We study the problem of automatic and *optimal* camera placement. Optimal camera placement is defined as the task of finding a set of cameras (e.g. defined by location, orientation, field-of-view (FoV), and depth-of-field (DoF)) to optimize a task-specific *objective* in a pre-defined *region-of-interest* (ROI).

There are two main ways to formulate the camera placement problem: **(i)** minimizing the number of cameras that completely cover the ROI [MD04, DOL06, ES06, RRA*06, BDSP07, YK08, GB09, SKR09, vdHHW*09, AA11, Ste12] and **(ii)** maximizing the coverage of the ROI with a limited number of cameras [HL06, YCA*08].

Our first major contribution is to propose a framework that tackles both of these questions as well as a novel extension that is unique to our work. This extension is a general model for camera relationships that encodes how well two cameras can work together. For example, for many tasks (e.g. 3D reconstruction and multi-camera tracking), it is beneficial to cover a region with more than one camera. However, these cameras should preferably observe the region from different views.

An important practical aspect of the camera placement problem is efficiency. A user typically wants to explore the solution space, rather than compute a single solution. For example, a user might decide that he can only afford seven cameras for surveillance of a store. However, after exploring the solution space, he might find that reasonable configurations can only be found by expanding the camera network to more cameras. Similarly, a user could have a list of hard constraints that unfortunately afford no solution. Therefore, the goal of this work is to derive an efficient optimization algorithm that enables the network designer to explore the space of optimal camera solutions for varying requirements and user-defined parameters. We observe that current methods, which model the problem as a binary linear program (BLP), are computationally expensive for medium and large scale problems, where hundreds of cameras need to be placed. In these cases, solving a BLP using the most efficient branch-and-bound methods has a runtime in the order of *hours*. This runtime might be acceptable to compute one placement, but it is infeasible to explore the solution space and experiment with different parameters.

Our second contribution is to propose a new mathematical formulation for the placement problem (namely a convex quadratic program with linear constraints), which lends itself amenable to efficient optimization methods. These methods are significantly faster than branch-and-bound techniques, without sacrificing solution quality as greedy methods.

Related Work

There are several variations of the camera placement problem in the literature, depending on the objective to be optimized, the type and nature of the ROI, the presence of static or dynamic obstalces/occluders in the ROI, and any strict constraints (e.g. limitation on the total number of cameras) to be satisfied. We refer the reader to literature surveys on sensor planning methods [TT95, MC13, MCT14].

Sparsest Camera Placement: Most often, the ROI is discretized spatially into a finite set of grid cells; however, there are exceptions that do address the much harder continuous placement problem [Ste12] and the well-known Art Gallery Problem [O'R87]. The majority of work on camera placement seeks to find the minimum number, layout, and settings of cameras needed to 'cover' a specific ROI, most often in the presence of static occluders (e.g. walls and pillars). The visibility of the ROI from a single camera is usually modeled as a binary profile (or occupancy grid map), where non-zero values indicate visible grid cells. In this case, the problem is a set coverage problem. Methods of this type vary according to their treatment of the underlying optimization problem. The method of Erdem *et al.* [ES06] addresses the problem on a 2D floorplan from a computational geometry point of view, models it as a linear binary program (LBP), and exploits conventional branch-and-bound methods to solve it. The computational cost of solving this LBP is reduced by a divide-and-conquer strategy using the Parisian evolutionary computation approach of Dunn *et al.* [DOL06]. This LBP can be extended to allow for different types of cameras (directional and omnidirectional) with different imaging models [GB09]. Ram *et al.* use a heuristic greedy method to approximate the LBP solution [RRA*06]. In the work of Sivaram *et al.*, the LBP is relaxed to an LP and solved using the conventional simplex method [SKR09]. In the previous methods, all grid cells of the ROI are considered to have equal importance. This assumption is relaxed by Yabuta *et al.* to distinguish essential from non-essential cells [YK08]. Also, independently moving dynamic occluders/obstacles can be incorporated to form a probabilistic formalism of the occupancy grid map [MD04] and the solution can be found using simulated annealing. Furthermore, trajectories generated by object tracking in video can be used to weigh the different grid cells [BDSP07]. Here, optimization is performed greedily, where the single best camera at each iteration is added to the solution. Some vision tasks (e.g. tracking and reconstruction) encourage higher degrees of overlap between cameras or proper camera handoff. This can be viewed as a structural constraint on the spatial layout of the network. Van Den Hengel *et al.* incorporate a simple version of this constraint in 3D floorplans and use a computationally expensive genetic programming method for optimization [vdHHW*09].

Limited Budget Placement: Another placement objective that is optimized is the coverage/visibility of an ROI under a strict limitation on the number of cameras (or equivalently a limited budget). Here, the entire ROI cannot usually be covered, so an optimal placement of the network is necessary to satisfy task-specific requirements. In the work of Horster *et al.*, this objective is minimized in 2D by solving the BLP greedily (one camera at a time) and by employing a heuristic divide-and-conquer strategy to reduce the problem size [HL06]. In the context of multi-camera persistent tracking, Yao *et al.* tackle a similar problem for a 3D ROI [YCA*08]. The discrete optimization method used here is very computationally expensive.

2. Methodology

In this section, we give a detailed description of our proposed camera placement framework (refer to Figure 1). In what follows, we discuss the various objectives and constraints that can be incorporated in this generic framework. We formulate and approximate the underlying problem as a convex quadratic binary program (QBP), which in turn can be relaxed using standard methods to a convex quadratic program (QP). Although the relaxed QP can include a large number of variables, it is amenable to efficient solutions over ranges of user-defined parameters. Also, we describe a GUI that allows the user to efficiently explore the solution space across ranges of parameters.

2.1. Floorplan Model

In this paper, we represent an ROI by a 3D floorplan. In this work, we expect a user to input a floorplan image, such as the one of a recreation room in Figure 2. We developed a user-friendly GUI that is used to draw points and polylines on the floorplan and to specify height information, e.g. to indicate where walls are, where cameras might be located, or to place arbitrary 3D objects. Similar to most previous work, the floorplan is discretized into grid cells with a user-defined spatial sampling rate in all three spatial dimensions. As this rate increases, it approximates the continuous camera placement problem. Following practical considerations, cameras cannot be located anywhere, so their locations are also discretized. They are shown as black circles in the GUI. The spacing between cameras is taken to be uniform and determined by a user-defined parameter, but the user also has the chance to add cameras manually to the floorplan. Once the locations and grid cells are defined, we discretize the set of possible cameras at each location by discretizing DoF, FoV, and orientation. These settings are determined by default parameters that can be easily changed in the GUI. Such a discretization scheme is valid in practice, where only a limited range of camera settings is usually plausible in a given surveillance setting. We also allow the user to add individual cameras using non-default parameters.

2.2. Camera Model

A camera is defined by the following parameters: location (3D), orientation (3D), horizontal and vertical FoV (2D), and DoF (1D). Given a camera's parameters, we can estimate its 3D visibility profile using a traditional quadratic decay function. In Figure 3 (*left*), we give an illustrative example of this profile (as a cross-section) for a single camera located at the corners of one of the pillars in the recreation room example of Figure 2. Given the camera's parameters, its visibility profile is defined as the line-of-sight visibility of each point in the floorplan as viewed from the camera. The profile is a pyramid whose dimensions are determined by the camera's FoV and orientation. In our experiments, we set the horizontal FoV=60^o and the vertical FoV=30^o. The visibility value at any point that is inside the cone and has a line-of-sight with the camera is a positive value in $[0,1]$, which decreases quadratically with distance from the camera's optical center and quadratically with angular distance from the camera's optical axis. This models two fundamental aspects of placement problems: the change of resolution with distance-to-camera, as well as, lens distortion at peripheries. Other constraints can be added to this camera model to encode other properties of real cameras. In fact, our proposed placement solution is generic enough to allow for different models of visibility, thus, making it applicable to generic sensor placement problems. Here, we note that most previous work considers only binary visibility values, thus, oversimplifying the problem. The visibility values of four points in the ROI are shown in Figure 3. Point (X_3,Y_3) is invisible to the camera, since it is occluded by the pillar. Opaque obstacles have a visibility of -1 and are excluded from the floorplan discretization. In this work, we use ray tracing to efficiently compute the visibility profile of any camera in the ROI.

2.3. Importance Distribution of Floorplan

Similar to the work by Yabuta *et al.* [YK08], we define an importance value for each grid cell in the floorplan. These values represent how crucial each grid cell's visibility is to the overall camera network. The distribution of these importance values is task- and floorplan-specific. For example, in high-security scenario where individuals are tracked across the network, some regions (e.g. doorways and exits) tend to be more security-sensitive (and thus more important) than others. Such a distribution is modelled as a pmf (probability mass function) across the grid cell centers. In the GUI, the user defines the distribution by adding to the floorplan reference points that have user-defined mass and extent (i.e. mean and covariance of a 3D normal distribution). The importance of any grid cell center is computed via kernel density estimation. Equivalently, when tracking is a requirement, the user can trace possible trajectories in the floorplan, from which importance values are estimated according to how repetitive these trajectories are in the spatial domain. Importance values are normalized to sum to 1. We show an example of a user-generated importance distribution in Figure 3 (*right*).

2.4. Mathematical Formulation

The inputs to our algorithm are a discrete set of cameras, where each camera is parameterized as a 4-tuple $c_i = (l_i,\theta_i,d_i,f_i)$, representing its 3D location, 3D orientation, 2D FoV, and 1D DoF respectively. We denote Ω as the universal set of all 4-tuples for a given floorplan. For every camera

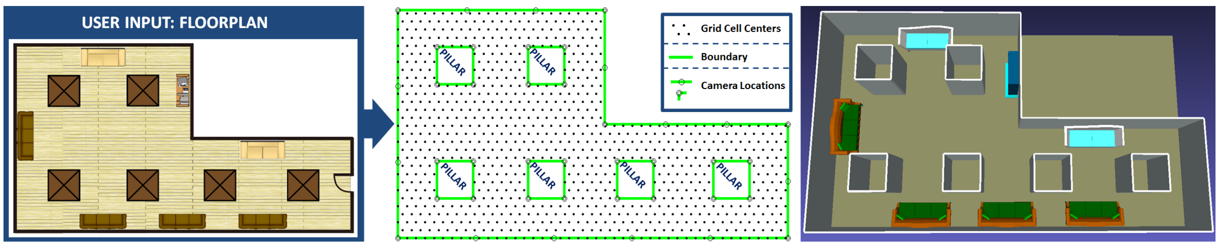

Figure 2: In our GUI, the user inputs a 2D floorplan (left) that can be manually digitized to indicate boundaries (in *green*), where cameras can be located. Using a specified wall height, the floorplan is discretized into a 3D uniform grid represented by black dots (middle). Camera settings (location, orientation, FoV, and DoF) can also be set. The floorplan can also be visualized and edited (right) to add general 3D obstacles, e.g., pillars, stairs, and furniture. By default, cameras are located at uniform distances on the boundaries. Orientations of cameras at the same location are also uniformly sampled.

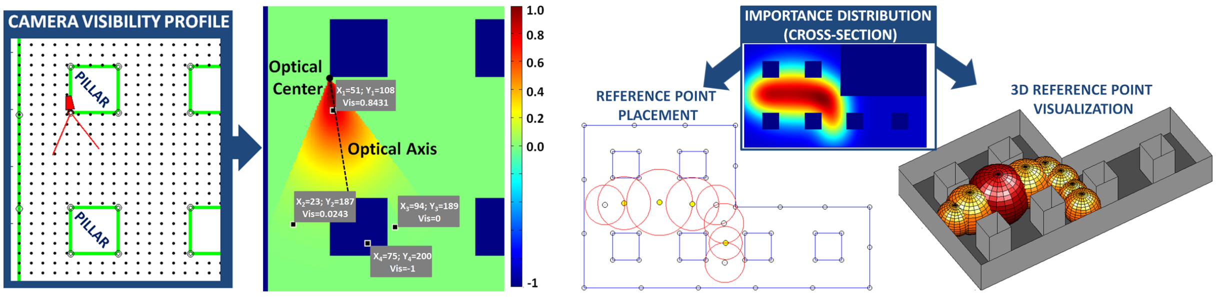

Figure 3: Left: Cross-section of a camera visibility profile. Visibility to the camera decreases quadratically with the distance from the optical center and the angular distance from the optical axis. (X_3,Y_3) is occluded by the pillar and is invisible to the camera. Right: Importance distribution across the recreation room, generated by clicking and weighing four reference points.

$c_i \in \Omega$, we define a binary variable $\mathbf{x}_i \in \{0,1\}$, which designates whether or not c_i exists in the network. Concatenating all these variables together forms the camera network vector $\mathbf{x} \in \{0,1\}^N$ where $N = |\Omega|$. Denoting M as the total number of grid cell centers in the floorplan, we define $\mathbf{a}_i$ as the importance value of the i^{th} grid cell and $\mathbf{a} \in \mathbb{R}_+^M$ as the vector of all importance values. Since $\mathbf{a}$ is a pmf, $\mathbf{1}^T\mathbf{a} = 1$. Furthermore, by denoting G as the number of distinct camera locations, we compile the location matrix $\mathbf{L} \in \{0,1\}^{N\times G}$ to group cameras belonging to the same location. The i^{th} column of $\mathbf{L}$ indexes cameras belonging to the i^{th} distinct location.

We generate the visibility profile for each $c_i \in \Omega$ at all M grid cells to form the *visibility matrix* $\mathbf{V} \in \mathbb{R}_+^{N\times M}$. The i^{th} row of $\mathbf{V}$ is the visibility profile of c_i. By using ray tracing, these profiles are computed efficiently (alternative solutions use BSP-trees, or ray bundle tracing). Next, we define the visibility function $v(j|\mathbf{x})$, which evaluates the visibility of point j in the ROI w.r.t. the camera network $\mathbf{x}$. Various forms of this function are conceivable. From a mathematical viewpoint, setting $v(j|\mathbf{x}) = \frac{1}{\mathbf{1}^T\mathbf{x}}\sum_i \mathbf{V}_{ij}\mathbf{x}_i$ (i.e. the average visibility value from all cameras in $\mathbf{x}$) simplifies the overall optimization. However, it does not adequately model the surveillance requirements of a camera network, since $v(j|\mathbf{x})$ can still be a low value even though a camera, which visualizes location j significantly well, does exist. Therefore, we resort to a more appropriate form, which has been utilized in previous work [HL06]. A point in the ROI can be surveyed successfully if that point's visibility from *at least* one of the cameras in the network exceeds a pre-defined minimum threshold. This threshold can be equivalent to a minimum resolution constraint. Therefore, we define $v(j|\mathbf{x})$ in Eq (1).

$$v(j|\mathbf{x}) = \max_i \left(\{\mathbf{V}_{ij}\mathbf{x}_i\}\right) \tag{1}$$

This visibility function will play a major role in formulating camera placement as a binary optimization problem that seeks to minimize an objective function $o(\mathbf{x})$ (and possibly a regularizer $r(\mathbf{x})$) subject to some task-specific constraints $g(\mathbf{x}) \geq \mathbf{0}$. The exact nature of the objective and equivalently the underlying constraints are task-specific. In what follows, we describe two objectives that are popularly considered in the camera placement literature and formulate their optimization mathematically.

2.4.1. Sparsest Camera Placement

The first popular form for camera placement is the maximum coverage problem, where a minimum number of cameras (or equivalently a minimum budget) is required to 'cover' a specified subset of the floorplan (possibly its entirety). Here, coverage refers to whether the visibility of a point in the ROI exceeds a threshold (e.g. minimum resolution). This is the case for high-sensitivity surveillance scenarios, where budget is secondary to performance. In this paper, we set $o(\mathbf{x}) = \mathbf{1}^T\mathbf{x}$, which is the number of cameras in the network. If cameras have different costs, the overall budget is minimized by replacing $\mathbf{1}$ in $o(\mathbf{x})$ with a cost vector $\mathbf{k}$.

We require $v(j|\mathbf{x}) \geq \varepsilon_j \; \forall j = 1,\cdots,M$. Here, the threshold ε_j is allowed to be a function of $\mathbf{a}_j$ to incorporate the importance value of grid cell j. Without loss of generality, we set $\varepsilon_j = \beta\mathbf{a}_j$ in this paper. Also, we set a limit on the total number of cameras that can be installed at each distinct camera location: $\mathbf{L}^T\mathbf{x} \leq \mathbf{n}$, where $\mathbf{n}_i$ is the limit at the i^{th} location. In most cases, $\mathbf{n} = \mathbf{1}$.

Unlike previous work that assumes independence between cameras in the network, we allow the user to constrain the overall camera layout by applying a pairwise regularizer $r(\mathbf{x}) = \mathbf{x}^T\mathbf{W}\mathbf{x}$, where $\mathbf{W} \in \mathbb{R}_+^{N\times N}$ is a weight matrix that measures the incompatibility of each pair of cameras in the network. To the best of our knowledge, we are the first to model such structural regularization on the camera layout. Defining $\mathbf{W}_{ij}$ is task-specific in general. For instance, since obtaining appearance information of an object from different views is crucial for robustness against occlusion in 3D reconstruction and multi-camera tracking, $\mathbf{W}_{ij}$ is expected to be high for cameras with little overlap in their visibility profiles and/or with optical axes pointing in the same direction (same view). Without loss of generality, we define $\mathbf{W}_{ij} = e^{-s_1(c_i,c_j)s_2(c_i,c_j)}$, where s_1 measures the intersection (or overlap) in profiles and s_2 the divergence from π of the angle between the optical axes. Other forms of $\mathbf{W}_{ij}$ are easily incorporated. Combining all these terms together, we formulate the sparsest camera placement problem in Eq (2). Note that when $\lambda = 0$, the problem degenerates to the case where camera network structure is insignificant.

$$\min_{\mathbf{x}} \; \mathbf{1}^T\mathbf{x} + \lambda\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{2}$$
$$\text{subject to: } \begin{cases} v(j|\mathbf{x}) \geq \beta\mathbf{a}_j \quad \forall j = 1,\cdots,M \\ \mathbf{L}^T\mathbf{x} \leq \mathbf{n}; \quad \mathbf{x} \in \{0,1\}^N \end{cases}$$

The visibility constraint in Eq (2) is nonlinear and nonconvex. A similar constraint was used previously [HL06], where an $M \times N$ auxiliary binary variable is added to linearize the constraint. Increasing the problem size in such a way quickly makes the optimization infeasible (especially at medium to large scales), so Horster *et al.* resorted to a greedy heuristic optimization strategy [HL06]. In this paper, we avoid this strategy by approximating the max function with a smooth approximation (known as the softmax function) as follows:

$$v(j|\mathbf{x}) \approx \sum_{i=1}^{N} \mathbf{V}_{ij}\mathbf{x}_i \frac{e^{\alpha\mathbf{V}_{ij}\mathbf{x}_i}}{\sum_{i=1}^{N} e^{\alpha\mathbf{V}_{ij}\mathbf{x}_i}}.$$

Since each $\mathbf{x}_i$ is binary, we have two identities: $e^{\alpha\mathbf{V}_{ij}\mathbf{x}_i} = (e^{\alpha\mathbf{V}_{ij}} - 1)\mathbf{x}_i + 1$ and $\mathbf{x}_i^2 = \mathbf{x}_i$. Using these identities, we formulate $v(j|\mathbf{x})$ in Eq (3), which is a fractional linear function of $\mathbf{x}$. Replacing this approximation in Eq (2), the original problem is reformulated as $P_1(\beta,\lambda)$ in Eq (4), where $\hat{\mathbf{V}}(\mathbf{b})$ is a matrix function such that each element of the result is defined as $\hat{\mathbf{V}}_{ij}(\mathbf{b}) = \frac{1}{N}[e^{\alpha\mathbf{V}_{ij}}(\mathbf{V}_{ij} - \mathbf{b}_j) + \mathbf{b}_j] \quad \forall i,j$. It is known that as $\alpha \to \infty$, the softmax approximation becomes exact. However, large values of α lead to very large values of $\hat{\mathbf{V}}$, which in turn lead to numerical instability in the optimization. After experimenting with different values of α, we observe that $\alpha = 5$ is a suitable empirical tradeoff between numerical stability and the fidelity of the approximation. We use this particular value for α in all our experiments.

$$v(j|\mathbf{x}) = \frac{\sum_{i=1}^{N}(\mathbf{V}_{ij}e^{\alpha\mathbf{V}_{ij}})\mathbf{x}_i}{N + \sum_{i=1}^{N}(e^{\alpha\mathbf{V}_{ij}} - 1)\mathbf{x}_i} \tag{3}$$

$$P_1(\beta,\lambda): \; \min \; \mathbf{1}^T\mathbf{x} + \lambda\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{4}$$
$$\text{subject to: } \begin{cases} \hat{\mathbf{V}}(\beta\mathbf{a})^T\mathbf{x} \geq \beta\mathbf{a} \\ \mathbf{L}^T\mathbf{x} \leq \mathbf{n}; \quad \mathbf{x} \in \{0,1\}^N \end{cases}$$

2.4.2. Limited Budget Placement

In this problem, there is a strict limit K on the number of cameras allowed (or equivalently on the budget). The goal is to find an optimal $\mathbf{x}$ that maximizes the coverage of grid cells. In other words, it maximizes the number of grid cells whose visibility w.r.t. the network exceeds a given threshold ε. We formulate this in Eq (5), where $\hat{\mathbf{v}}_j(\varepsilon)$ is the j^{th} column of matrix $\hat{\mathbf{V}}(\varepsilon)$ as defined before. We use the hinge loss function, defined as $\max(0, \varepsilon - \hat{\mathbf{v}}_j(\varepsilon)^T\mathbf{x})$, to penalize the j^{th} grid cell if its visibility falls below ε. Each hinge loss is weighted according to the importance of its grid cell (i.e. $\mathbf{a}_j$). By adding $\mathbf{t} \in \mathbb{R}^M$ as an auxiliary variable, Eq (5) is reformulated as $P_2(\varepsilon,\lambda,K)$ in Eq (6).

$$\min \sum_{j=1}^{M} \mathbf{a}_j \max(0, \varepsilon - \hat{\mathbf{v}}_j(\varepsilon)^T\mathbf{x}) + \lambda\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{5}$$
$$\text{subject to: } \begin{cases} \mathbf{1}^T\mathbf{x} \leq K \\ \mathbf{L}^T\mathbf{x} \leq \mathbf{n}; \quad \mathbf{x} \in \{0,1\}^N \end{cases}$$

$$P_2(\varepsilon,\lambda,K): \; \min \; \mathbf{a}^T\mathbf{t} + \lambda\mathbf{x}^T\mathbf{W}\mathbf{x} \tag{6}$$
$$\text{subject to : } \begin{cases} \mathbf{1}^T\mathbf{x} \leq K; \quad \mathbf{L}^T\mathbf{x} \leq \mathbf{n}; \quad \mathbf{x} \in \{0,1\}^N \\ \mathbf{t} \geq \mathbf{0}; \quad \mathbf{t} \geq \varepsilon\mathbf{1} - \hat{\mathbf{V}}(\varepsilon)^T\mathbf{x} \end{cases}$$

2.5. Optimization and Implementation Details

The two formulations $P_1(\beta,\lambda)$ and $P_2(\varepsilon,\lambda,K)$ are quite similar, so the same optimization strategy is used to solve both. Since they are generally NP-hard, exact solutions are computationally infeasible even for relatively small networks. It is worthwhile to note that when $\lambda = 0$, both problems have the BLP form addressed by previous work. This shows that previous formulations of camera placement can be incorporated into our generic framework. Also, an interesting property of the solution is that it is expected to be significantly sparse, since a reasonable discretization of camera parameters leads to a large M.

Convexity: For $\lambda > 0$, the convexity of both problems (and the computational cost of solving them) depends heavily on matrix $\mathbf{W}$. In general, $\mathbf{W}$ is not positive semi-definite (psd), so P_1 and P_2 are non-convex. However, since $\mathbf{x}_i \in \{0,1\}$ and $\mathbf{x}_i^2 = \mathbf{x}_i$, we have $\mathbf{x}^T\mathbf{x} = \mathbf{1}^T\mathbf{x}$ and the following identity: $\mathbf{x}^T\mathbf{W}\mathbf{x} = \mathbf{x}^T(\mathbf{W}-\sigma\mathbf{I})\mathbf{x} + \sigma\mathbf{1}^T\mathbf{x}$ for any σ. So, by setting σ to be smaller than the minimum negative eigenvalue of $\mathbf{W}$, i.e. $\sigma \leq \min(0, e_{\min}(\mathbf{W}))$, we reformulate P_1 and P_2 as convex binary quadratic programs (BQPs). In realistic cases, $\mathbf{W}$ is a large sparse matrix, since only a relatively small number of camera pairs are related to each other. Therefore, the minimum negative eigenvalue of $\mathbf{W}$ can be efficiently computed using conventional partial eigen-decomposition methods (e.g. the implicitly restarted Arnoldi method). Here, we note that BQPs also arise in the inference stage of an MRF. However, since MRF inference only considers unconstrained BQPs, they do not directly apply to the camera placement problem.

Relaxation to QP: Branch-and-bound methods exist for convex BQPs [BCL10]; however, they are computationally too expensive to solve individual large-scale problems and sets of these problems corresponding to varying user-defined parameters (e.g. changes in β, λ, ε, K, camera parameters (DoF or FoV), discretization of floorplan, etc.). To put this in perspective, the exact solution to a single P_1 problem using a BQP solver with $M = 800$ and $N = 250$ (small scale) has a runtime of more than an hour on an 8-core workstation running MATLAB. Such performance prohibits an interactive exploration of the solution space.

Instead of solving P_1 and P_2 exactly, we relax the BQP to a QP by replacing the binary constraint with a box constraint on the real-valued variable $\mathbf{x}$. The resulting (relaxed) convex QPs are stated in Eq (7)&(8), where $\sigma = \min(0, e_{\min}(\mathbf{W}))$. This relaxation is one of the conventional general-purpose relaxations for binary problems. We use it in this paper because of its simplicity and its memory and computational efficiency. Another popular relaxation technique transforms the binary problem into a semi-definite matrix problem (SDP). This type of relaxation has been previously used for camera placement [ZHYC11, EYEGG06] and other labeling problems [OEK07]; however, it suffers from two main drawbacks. For the camera placement problems in this paper, SDP relaxation is significantly more memory intensive and has a comparatively slower runtime than box-constraint relaxation. Moreover, adding smooth convex constraints to the BQP, as is the case in Eqs (4)&(5), cannot be trivially handled by SDP relaxation.

$$\hat{P}_1(\beta,\lambda): \ \min \ (1+\lambda\sigma)\mathbf{1}^T\mathbf{x} + \lambda\mathbf{x}^T(\mathbf{W}-\sigma\mathbf{I})\mathbf{x} \tag{7}$$

$$\text{subject to: } \begin{cases} \hat{\mathbf{V}}(\beta\mathbf{a})^T\mathbf{x} \geq \beta\mathbf{a} \\ \mathbf{L}^T\mathbf{x} \leq \mathbf{n}; \ \ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1} \end{cases}$$

$$\hat{P}_2(\varepsilon,\lambda,K): \ \min \ \mathbf{a}^T\mathbf{t} + \lambda\sigma\mathbf{1}^T\mathbf{x} + \lambda\mathbf{x}^T(\mathbf{W}-\sigma\mathbf{I})\mathbf{x} \tag{8}$$

$$\text{subject to: } \begin{cases} \mathbf{1}^T\mathbf{x} \leq K; \ \ \mathbf{L}^T\mathbf{x} \leq \mathbf{n}; \ \ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1} \\ \mathbf{t} \geq \mathbf{0}; \ \ \mathbf{t} \geq \varepsilon\mathbf{1} - \hat{\mathbf{V}}(\varepsilon)^T\mathbf{x} \end{cases}$$

Since σ is set to a value that maintains the convexity of both relaxed problems $\hat{P}_1$ and $\hat{P}_2$, the *global* minimum to these problems can be reached using conventional convex optimization methods starting from any random initialization. However, it is not necessary that the placement solution vector $\mathbf{x}^*$ that leads to this global minimum is unique across different random initializations. To minimize $\hat{P}_1$ and $\hat{P}_2$, we use an efficient QP solver [DG05] that guarantees global linear convergence for individual QP problems. It is well equipped to handle large-scale sparse problems. For a δ-accurate solution, its computational complexity is roughly $O(N(M+G)\log\delta)$ for $\hat{P}_1$ and $O((N+M)(M+G)\log\delta)$ for $\hat{P}_2$. Interestingly, this solver can efficiently minimize $\hat{P}_1$ and $\hat{P}_2$ even when $\lambda = 0$, i.e. even when traditional LP models of camera placement are considered and efficient LP methods can be used. In this case, solving the QP with $\lambda \ll 1$ (e.g. $\lambda = 10^{-8}$) converges to the same solution (up to a negligible tolerance) *faster* than solving the LP.

Since this solver is iterative and incorporates an augmented Lagrangian method, the solution to $\hat{P}_1(\beta+\Delta\beta, \lambda+\Delta\lambda)$, for reasonably sized increments $\Delta\beta$ and $\Delta\lambda$, can be initialized to the solution of $\hat{P}_1(\beta,\lambda)$ after projecting it into the feasible space. This is the main reason why solving these QPs across a *range* of user-defined parameters is computationally attractive. We show empirical evidence of this later. After solving the QP, the relaxed solution $\mathbf{x}^*$ is thresholded and subsequently projected unto the feasible space, whereby the threshold is adaptively set by clustering the values of $\mathbf{x}^*$ using a Gaussian mixture model with two components. For example, if the thresholded solution to $\hat{P}_2(\varepsilon,\lambda,K)$ contains more than K cameras, excess cameras are greedily removed such that the first removed camera increases the objective the least. Similar logic holds for the other constraints.

3. Experimental Results

In this section, we provide empirical evidence to validate the effectiveness and accuracy of our placement framework as compared to a greedy and exact baseline, its efficiency for ranges of user-tuned parameters, and its applicability to efficient exploration of the space of placement solutions.

3.1. User-Friendly GUI

We developed a simple GUI that allows users to load and discretize a floorplan in SVG format, localize, discretize, and parameterize different types of cameras, and set reference points (or trajectories) to determine the importance distribution. Placement solutions (for single or multiple sets of parameters) can be loaded to the floorplan, visualized, and exported to SVG format for use in conventional tools such as Adobe Illustrator.

3.2. Quality of Soft-Max Approximation

As mentioned earlier, we approximate the non-differentiable max function with the soft-max operator, which leads to the visibility function in Eq (3). The fidelity of this approximation is dominated by the α parameter. Ideally, the approximation is better when α takes on large values. However, in practice, large values of α lead to numerical instability that prevents the optimization algorithm from converging to the global minimum and significantly slows down its runtime. To evaluate the effect of α on the two placement problems, we consider the recreation room floorplan and solve $\hat{P}_1$ with $\beta = 0.05$ and $\hat{P}_2$ with $(\varepsilon = 0.01, K = 10)$ with increasing values of α. For both problems, we plot the objective that is obtained after convergence versus α in Figure 4. We also show the convergence time of the optimization for a sample set of α values. Small α values lead to sub-optimal results, since the soft-max operator leads to a loose approximation of the max function. Also, there exists a range of α values, which lead to the same minimum solution (representing faithful approximation). Although the solution is the same in this range, the runtime is higher for larger values of α. When α is set to very large values (greater than 100), numerical instability arises and the solution degrades with a substantial increase in runtime. To tradeoff approximation quality and runtime, we set $\alpha = 5$ in all our experiments.

3.3. Quantitative Comparison

We compare the accuracy and runtime of our optimization method to that of two baselines: **(i)** the exact discrete solution to $\hat{P}_1$ and $\hat{P}_2$ using a branch-and-bound method (available in MATLAB) and **(ii)** a greedy solution that iteratively identifies the feasible cameras (according to the constraints) in $\mathbf{x}$ that can be set to 1, ranks them according to the change in objective function, and greedily selects the one with highest rank (similar to the work of Krause *et al.*, which maximizes the mutual information between the selected cameras and those not selected [KSG08]) by setting its corresponding $\mathbf{x}_i = 1$. Since the implementation of prior work is not publicly available, we use the aforementioned baselines to emulate the types of optimization used in the literature.

For testing purposes, we consider the recreation room floorplan, whose discretization and camera parametrization are changed to vary M and N respectively. Using our proposed method and the two baselines, we solve P_1 with $\beta = 0.01$ and P_2 with $(\varepsilon = 0.01, K = 20)$. To simplify analysis, we set $\lambda = 0$ for both problems. In Figure 5 (*top*) and (*middle*), we increase the problem size $M \times N$ and plot the optimal objectives obtained by the greedy method and our own

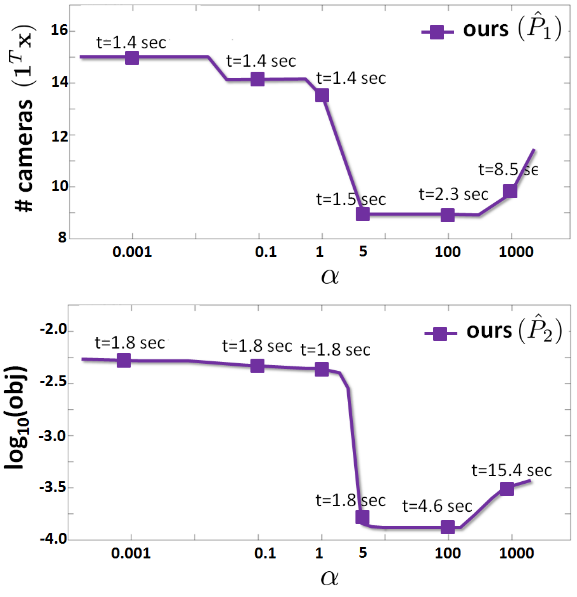

Figure 4: Evaluation of the effect of α on approximation quality. For both types of problems $\hat{P}_1$ and $\hat{P}_2$, our method converges to a global minimum solution for a range of α values. Smaller values lead to sub-optimal solutions because the soft-max operator does not adequately approximate the max function. Larger values lead to numerical instability that precludes proper convergence.

for both types of placement problems. Clearly, our proposed method produces a significantly smaller (24% on average) objective, i.e. a smaller number of cameras for sparsest camera placement and a larger number of covered grid cells for limited budget placement. At uniform intervals of $M \times N$, we report the runtime of the greedy baseline **(ii)** and our method in seconds (*bottom*). As expected, the runtime of both methods grows linearly with the the dimension of the problem. These runtime results validate our complexity analysis in Section 2.5. Although the greedy method is faster than ours in general, the runtime discrepancy decreases at larger values of $M \times N$. Moreover, these results show that our method is computationally feasible for mid- and large-scale placement problems, since the solution to a single problem of type $\hat{P}_1$ converges in 3.6 seconds and of type $\hat{P}_2$ in 6.8 seconds, when $MN \approx 2.3 \times 10^6$. Because of the slack variables in $\hat{P}_2$, its runtime is higher than that of $\hat{P}_1$. Due to its significantly high computational cost that increases exponentially with $M \times N$, it is infeasible to report the exact objective values for baseline **(i)** for these large values of $M \times N$. For smaller values (e.g. when $MN \approx 10^5$), the exact BLP method *does* outperform ours but only by 13% on average; however, its runtime exceeds 70 minutes. Here, the runtime of all methods does not include the time required to compute $\mathbf{V}$ and $\mathbf{W}$, which is negligible since the required computation is done efficiently and only once for each ROI and floorplan discretization. In

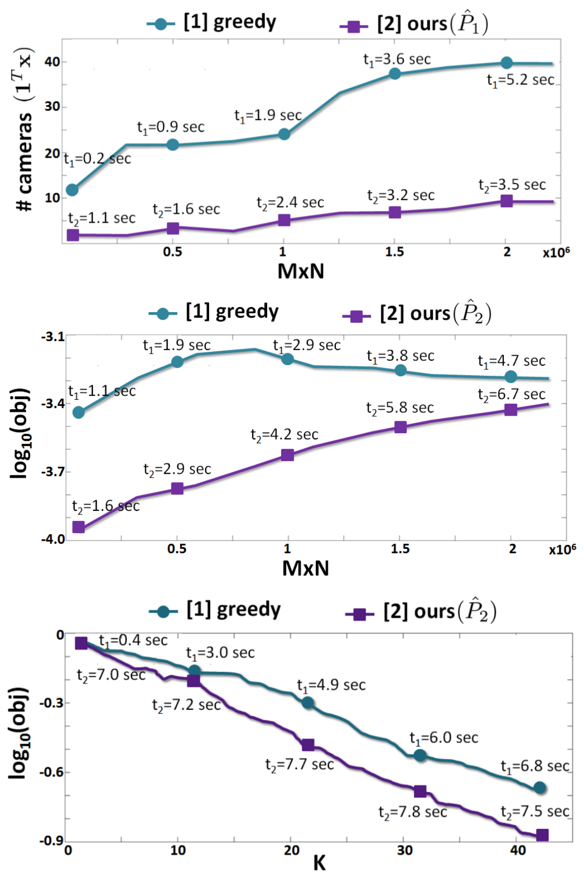

Figure 5: Comparison of performance and runtime between greedy baseline **(ii)** and our proposed method. For both types of placement problems $\hat{P}_1$ and $\hat{P}_2$, our method obtains a smaller cost, while remaining computationally competitive especially at large values of $M \times N$. A similar conclusion is reached when varying the parameters of the problem and keeping its size fixed. In the *bottom* plot, we vary K and report the performance and runtime for the two methods.

all our experiments, we run MATLAB 2011a on a 2.3GHz 16GB RAM machine.

To show the effect of the constraint space on the placement solution, we plot the performance and runtime of baseline **(ii)** and our proposed method for $\hat{P}_2$ problems with varying K (and fixed ε) in Figure 5 (*bottom*). Our method outperforms baseline **(ii)** especially at larger K values. More importantly, we note that the runtime of baseline **(ii)** grows linearly with K, since the feasible constraint space grows as K increases. In comparison, the runtime of our method is not affected much by K.

3.4. Exploring the Solution Space

For solution space exploration, we focus on the limited budget placement problem (i.e. $\hat{P}_2$). A very similar analysis can be done for $\hat{P}_1$. Since $\hat{P}_2$ is parameterized by (K,ε,λ), the user can explore this 3D parameter space to visualize how the solution changes with variations in these three parameters. To simplify presentation, we allow for 2D exploration, where λ is fixed to $\lambda_0 = \frac{1}{N}$ and the other two parameters are allowed to vary across a finite range of plausible values.

In Figure 6, we consider the recreation room example again with $MN \approx 2.3 \times 10^6$ and $\mathbf{a}$ determined by the given importance distribution. We vary ε (minimum visibility threshold) from 10^{-4} to 1 (20 discrete values) and K (maximum number of cameras) from 1 to $G = 42$ (number of distinct camera locations) in order to generate a total of 840 parameter combinations. For each pair (ε_0,K_0), we can solve $\hat{P}_2(\varepsilon_0,\lambda_0,K_0)$ and compute the percentage of the floorplan that is covered using the visibility threshold ε_0, i.e. the percentage of grid cells whose visibility satisfies $v(j|\mathbf{x}) \geq \varepsilon_0$, as defined in Eq (1). Clearly, the percentage of coverage decreases when ε increases for a fixed K and it increases when K increases for a fixed ε. This visualization allows the user to explore the solution space and, if required, to visualize the result of a particular solution. In Figure 6, the user selects parameter pair (K=10,$\varepsilon = 0.05$) with 54% coverage to visualize the underlying camera layout and the visibility of the floorplan w.r.t. the entire network.

As explained in Section 2.5, the iterative nature of our solver provides an efficient framework to solve a set of $\hat{P}_2$ problems with varying parameters. In fact, the 20×42 solution matrix shown in Figure 6 was generated in under 2 minutes, while naively solving each of the 840 problems independently runs in about 14 minutes. This result provides evidence of our solution's efficiency and validates our underlying formulation of the camera placement problem.

3.5. Qualitative Results

In each row of Figure 7, we show qualitative results of our placement solutions (as 2D cross-sections) to emphasize the effect of varying the different parameters in our framework. In the first two rows, $\hat{P}_2$ is solved with two possible ε values (one small and one large) for an increasing number of cameras K, while λ and $\mathbf{a}$ are kept fixed. When ε is small (the first row), the minimum visibility constraint is easily satisfied, so the camera layout is spread across the whole floorplan as K increases. This is to be contrasted with the results of the second row, where ε is 10 times larger. In this case, the camera layout is more compact, since the visibility constraint at each grid cell is harder to satisfy, especially at points with high importance. In the third row, we investigate the effect of λ on camera layout for a small camera network $K = 4$. When λ is very small, the cameras are treated independently, the structure of the overall layout is insignificant, and there is minimal overlap between any two cameras. As λ increases, the quadratic term in $\hat{P}_2$ has more impact on the optimization, thus, encouraging camera overlap and multi-view imaging of high importance regions in the floorplan. This behaviour continues till λ reaches its maximum value, when the quadratic term becomes dominant, the

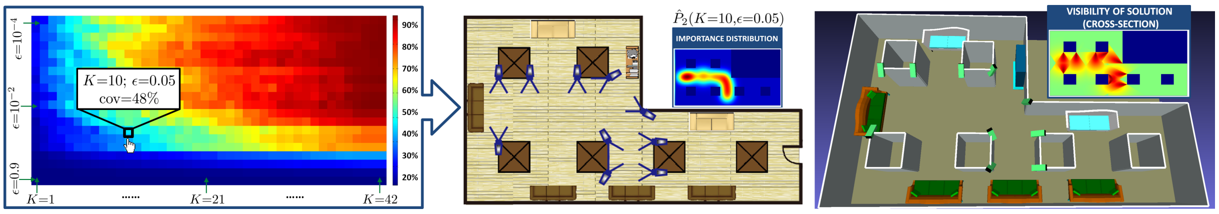

Figure 6: Exploring the solution space of $\hat{P}_2(\varepsilon,K)$. The solution matrix on the left visualizes the percentage of floorplan covered at a given (ε,K). This matrix was generated in just under two minutes. The user can select a particular parameter setting (e.g. ε=0.05 and K=10) and visualize the corresponding placement solution as a cross-section (in middle) or in 3D (on right), along with the resulting visibility map of the entire camera network (as a cross-section).

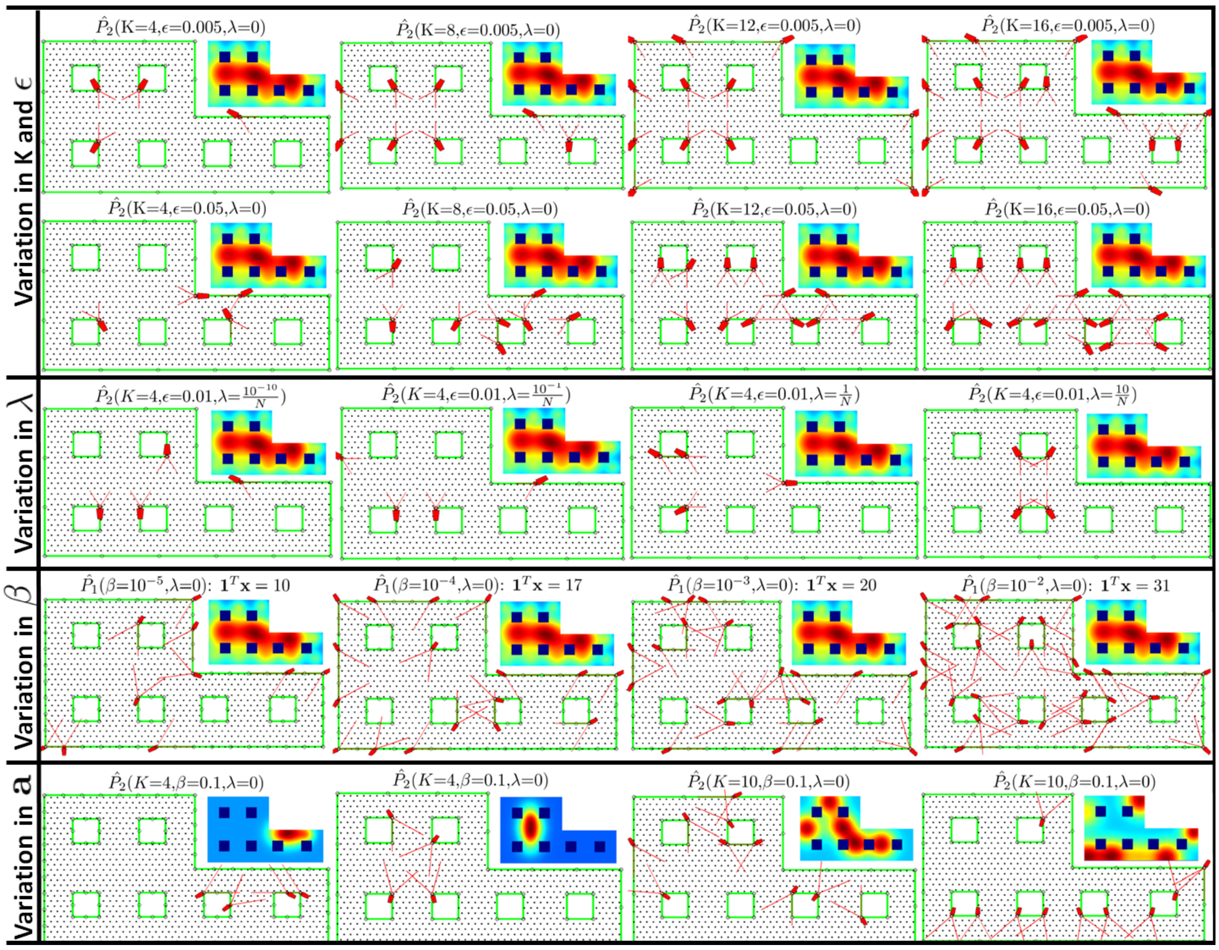
Variation in K and ϵ
$\hat{P}_2$(K=4,ϵ=0.005,λ=0)
$\hat{P}_2$(K=8,ϵ=0.005,λ=0)
$\hat{P}_2$(K=12,ϵ=0.005,λ=0)
$\hat{P}_2$(K=16,ϵ=0.005,λ=0)
$\hat{P}_2$(K=4,ϵ=0.05,λ=0)
$\hat{P}_2$(K=8,ϵ=0.05,λ=0)
$\hat{P}_2$(K=12,ϵ=0.05,λ=0)
$\hat{P}_2$(K=16,ϵ=0.05,λ=0)
Variation in λ
$\hat{P}_2(K=4,\epsilon=0.01,\lambda=\frac{10^{-10}}{N})$
$\hat{P}_2(K=4,\epsilon=0.01,\lambda=\frac{10^{-1}}{N})$
$\hat{P}_2(K=4,\epsilon=0.01,\lambda=\frac{1}{N})$
$\hat{P}_2(K=4,\epsilon=0.01,\lambda=\frac{10}{N})$
Variation in β
$\hat{P}_1(\beta=10^{-5},\lambda=0)$: $\mathbf{1}^T\mathbf{x}=10$
$\hat{P}_1(\beta=10^{-4},\lambda=0)$: $\mathbf{1}^T\mathbf{x}=17$
$\hat{P}_1(\beta=10^{-3},\lambda=0)$: $\mathbf{1}^T\mathbf{x}=20$
$\hat{P}_1(\beta=10^{-2},\lambda=0)$: $\mathbf{1}^T\mathbf{x}=31$
Variation in a
$\hat{P}_2(K=4,\beta=0.1,\lambda=0)$
$\hat{P}_2(K=4,\beta=0.1,\lambda=0)$
$\hat{P}_2(K=10,\beta=0.1,\lambda=0)$
$\hat{P}_2(K=10,\beta=0.1,\lambda=0)$

Figure 7: Solutions generated by our camera placement approach for $\hat{P}_1$ and $\hat{P}_2$ problems with varying parameters. In the first two rows, we study the effect of increasing the minimum visibility constraint in problem $\hat{P}_2$ defined by parameter ε. In the third row, we show the effect of increasing the quadratic tradeoff parameter λ. In the fourth row, the minimum visibility constraint in problem $\hat{P}_1$ is varied while fixing all other parameters. In the last row, we show how our placement solution adapts to dynamic changes in the scene, when the importance distribution (shown as a cross-section) of the 3D floorplan is changed. For a detailed description, refer to the text.

Table 1: A comparison of our algorithm with a greedy one. We list the number of entries in the visibility matrix $\mathbf{V}$ in millions ($M \times N$), the number of cameras in the scene (Cam), the objective value for $\hat{P}_2$ problems (Val) in thousands, and the run-time in seconds (Time). Our algorithm produces solutions with a smaller number of cameras for $\hat{P}_1$ problems (*garage*, *hotel*) and better coverage given a fixed number of cameras for $\hat{P}_2$ problems (*mall*, *airport*).

		Ours ($\hat{P}_1$)		**Greedy** ($\hat{P}_1$)	
Scene	$M \times N$	Cam	Time (sec)	Cam	Time (sec)
Garage	14.7	**30**	15.4	46	22.9
Hotel	6.1	**46**	4.1	53	5.6

		Ours ($\hat{P}_2$)			**Greedy** ($\hat{P}_2$)		
Scene	$M \times N$	Cam	Val	Time (sec)	Cam	Val	Time (sec)
Mall	6	20	**83**	31.7	20	95	7.3
Airport	6.6	26	**68**	12.6	26	79	6.1

cameras are placed opposite each other, and the floorplan coverage is minimal. In the fourth row, we analyse the effect of varying β (the minimum visibility constraint in $\hat{P}_1$) on the solution of $\hat{P}_1$. Here, we increase the default DoF of each camera and the number of camera locations to enable complete floorplan coverage. Smaller values of β lead to less cameras in the solution. As β increases, the minimum visibility constraint is harder to satisfy and more cameras need to be placed in the ROI. In fact, this is an example of how our proposed framework can easily adapt to different camera profiles, which can simulate different types of cameras including long-range or wide-angled surveillance cameras and even shorter range depth cameras (e.g. the Microsoft Kinect). As for the last row, we fix all optimization parameters and vary the importance distribution **a**. In these four examples, the doorway of the room, the couches, and other parts of the room are deemed important for surveillance and the cameras are placed accordingly.

In Figure 8, we show our placement solutions for four large-scale real-world ROIs, where $M \times N$ exceeds 6×10^6. The optimal camera layout is overlayed unto the original 2D floorplan image. For the *garage* and *hotel*, $\hat{P}_1$ problems are solved using the given importance distribution and parameters, while $\hat{P}_2$ problems are solved for the *mall* and *airport*. Renderings of the camera layout in 3D are shown besides the floorplans. High resolution images of all results are provided in the **supplementary material**. In Table 1, we report and compare the placement performance of our method and that of the greedy baseline **(ii)** when applied to these four ROIs. These results clearly show that our proposed method produces a *better* solution for both types of placement problems, within a reasonable runtime.

4. Conclusion

In this paper, we take a fresh look at the camera placement problem for computer graphics and computer vision applications. Our first contribution is to extend the problem statement to model camera-to-camera relationships, e.g. to prefer the placement of cameras that observe the same location from different views. Our second contribution is to derive an optimization strategy that that has a desirable trade off between speed (similar to a fast greedy algorithm) and quality (not much worse than an exact, high-quality binary optimization). In future work, we plan to extend our framework to moving cameras by integrating change detection and dynamic updates to the importance distribution.

5. Acknowledgments

We thank the anonymous reviewers for their valuable comments and suggestions. Special thanks goes to Yoshihiro Kobayashi and Christopher Grasso for generating the 3D renderings. Research reported in this publication was supported by competitive research funding from King Abdullah University of Science and Technology (KAUST).

References

[AA11] Amriki K., Atrey P.: Towards optimal placement of surveillance cameras in a bus. *IEEE International Conference on Multimedia and Expo* (2011). 2

[BCL10] Buchheim C., Caprara A., Lodi A.: An effective branch-and-bound algorithm for convex quadratic integer programming. In *Integer Programming and Combinatorial Optimization*, vol. 6080. 2010, pp. 285–298. 6

[BDSP07] Bodor R., Drenner A., Schrater P., Papanikolopoulos N.: Optimal Camera Placement for Automated Surveillance Tasks. *Journal of Intelligent and Robotic Systems 50*, 3 (2007), 257–295. 2

[DG05] Delbos F., Gilbert J. C.: Global linear convergence of an augmented Lagrangian algorithm for solving convex quadratic optimization problems. *Journal of Convex Analysis 12* (2005), 45–69. 6

[DOL06] Dunn E., Olague G., Lutton E.: Parisian camera placement for vision metrology. *Pattern Recognition Letters 27* (2006), 1209–1219. 2

[ES06] Erdem U., Sclaroff S.: Automated camera layout to satisfy task-speciïňĄc and floorplan-speciïňĄc coverage requirements. *Computer Vision and Image Understanding* (2006), 156–169. 2

[EYEGG06] Ercan A. O., Yang D. B., El Gamal A., Guibas L. J.: Optimal placement and selection of camera network nodes for target localization. In *IEEE International Conference on Distributed Computing in Sensor Systems* (Berlin, Heidelberg, 2006), DCOSS'06, Springer-Verlag, pp. 389–404. URL: http://dx.doi.org/10.1007/11776178_24, doi:10.1007/11776178_24. 6

[GB09] Gonzalez-Barbosa J.: Optimal camera placement for total coverage. *IEEE International Conference on Robotics and Automation* (2009), 844–848. 2

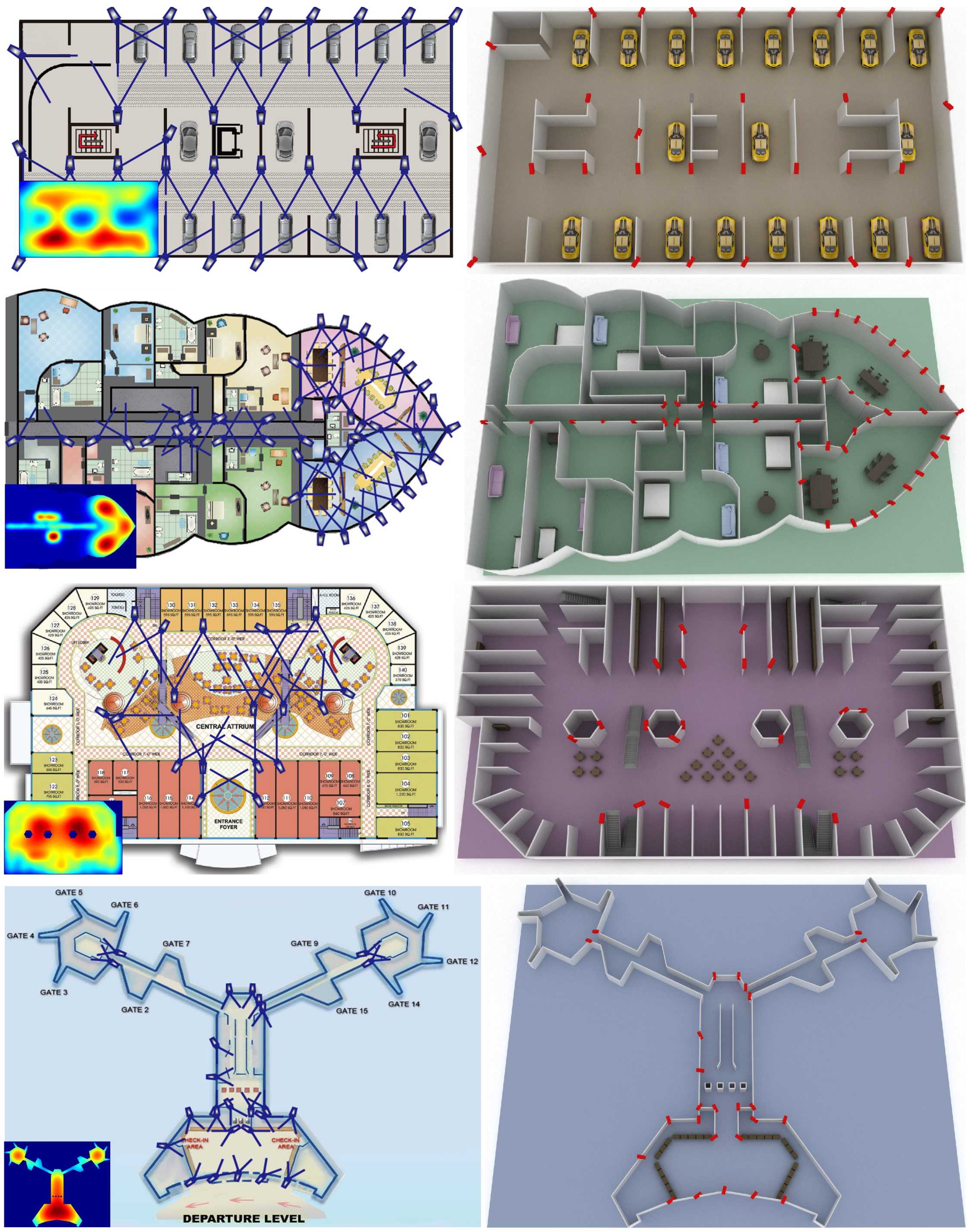

Figure 8: Results on real-world floorplans. We show solutions projected to 2D floorplans (*left*) and 3D renderings of the solutions (*right*). All high resolution results are provided in the **supplementary material**. In the first two rows, the sparsest camera placement problem, defined as $\hat{P}_1(\beta = 0.01, \lambda = \frac{1}{N})$, is solved for a *garage* and *hotel* floorplan respectively. In the third row, the limited budget problem, defined as $\hat{P}_2(K = 20, \varepsilon = 0.05, \lambda = \frac{1}{N})$, is solved for a *mall* floorplan, while a $\hat{P}_2(K = 26, \varepsilon = 0.05, \lambda = \frac{1}{N})$ problem is solved for the *hotel* floorplan. Note that a long-range surveillance cameras profile is used in the *garage* and *mall* examples to show the generic nature of our placement framework and its ability to adapt to different camera profiles.

[HL06] HORSTER E., LIENHART R.: On the optimal placement of multiple visual sensors. *International Workshop on Video Surveillance and Sensor Networks* (2006). 2, 3, 4, 5

[KSG08] KRAUSE A., SINGH A., GUESTRIN C.: Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research* (2008), 235–284. 7

[MC13] MAVRINAC A., CHEN X.: Modeling coverage in camera networks: A survey. *International Journal of Computer Vision 101*, 1 (2013), 205–226. URL: http://dx.doi.org/10.1007/s11263-012-0587-7, doi:10.1007/s11263-012-0587-7. 2

[MCT14] MAVRINAC A., CHEN X., TAN Y.: Coverage quality and smoothness criteria for online view selection in a multi-camera network. *ACM Transactions on Sensor Networks 10*, 2 (Jan. 2014), 33:1–33:19. URL: http://doi.acm.org/10.1145/2530373, doi:10.1145/2530373. 2

[MD04] MITTAL A., DAVIS L.: Visibility analysis and sensor planning in dynamic environments. *European Conference on Computer Vision* (2004). 2

[OEK07] OLSSON C., ERIKSSON A., KAHL F.: Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming. In *IEEE Conference on Computer Vision and Pattern Recognition* (June 2007), pp. 1–8. doi:10.1109/CVPR.2007.383202. 6

[O'R87] O'ROURKE J.: *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987. 2

[RRA*06] RAM S., RAMAKRISHNAN K., ATREY P., SINGH V., KANKANHALLI M.: A design methodology for selection and placement of sensors in multimedia systems. *International Workshop on Video Surveillance and Sensor Networks* (2006). 2

[SKR09] SIVARAM G. S. V. S., KANKANHALLI M. S., RAMAKRISHNAN K. R.: Design of multimedia surveillance systems. *ACM Transactions on Multimedia Computing, Communications, and Applications 5*, 3 (2009), 1–25. 2

[Ste12] STEINITZ A.: *Optimal Camera Placement*. Master's thesis, EECS Department, University of California, Berkeley, May 2012. 2

[TT95] TARABANIS P., TSAI R.: A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation* (1995). 2

[vdHHW*09] VAN DEN HENGEL A., HILL R., WARD B., CICHOWSKI A., DETMOLD H., MADDEN C., DICK A., BASTIAN J.: Automatic camera placement for large scale surveillance networks. *IEEE Workshop on Applications of Computer Vision* (2009), 1–6. 2

[YCA*08] YAO Y., CHEN C., ABIDI B., PAGE D., ABIDI B., ABIDI. M.: Sensor planning for automated and persistent object tracking with multiple cameras. *IEEE International Conference on Computer Vision and Pattern Recognition* (2008), 1–8. 2, 3

[YK08] YABUTA K., KITAZAWA H.: Optimum camera placement considering camera specification for security monitoring. *IEEE International Symposium on Circuits and Systems 2* (2008), 2114–2117. 2, 3

[ZHYC11] ZHAO J., HAWS D., YOSHIDA R., CHEUNG S.: Approximate techniques in solving optimal camera placement problems. In *IEEE International Conference on Computer Vision Workshop* (Nov 2011), pp. 1705–1712. doi:10.1109/ICCVW.2011.6130455. 6