# Optimal Camera Network Configurations for Visual Tagging

Jian Zhao, *Student Member, IEEE*, Sen-Ching (Samson) Cheung, *Senior Member, IEEE*, and Thinh Nguyen, *Member, IEEE*

*Abstract*—Proper placement of cameras in a distributed smart camera network is an important design problem. Not only does it determine the coverage of the surveillance, but it also has a direct impact on the appearance of objects in the cameras which dictates the performance of all subsequent computer vision tasks. In this paper, we propose a generic camera placement model based on the visibility of objects at different cameras. Our motivation stems from the need of identifying and locating objects with distinctive visual features or "tags." This is a very common goal in computer vision with applications ranging from identifying soccer players by their jersey numbers to locating and recognizing faces of individuals. Our proposed framework places no restriction on the visual classification tasks. It incorporates realistic camera models, self occlusion of tags, and occlusion by other moving objects. It is also flexible enough to handle arbitrary-shaped three-dimensional environments. Using this framework, two novel binary integer programming (BIP) algorithms are proposed to find the optimal camera placement for "visual tagging" and a greedy implementation is developed to cope with the complexity of BIP. Extensive performance analysis is performed using Monte Carlo simulations, virtual environment simulations, and real-world experiments. We also demonstrate the usefulness of visual tagging through robust individual identification and obfuscation across multiple camera views for privacy protection.

*Index Terms*—Binary integer programming, camera placement, multiple view geometry, privacy protection, smart camera network, visual tags.

## I. INTRODUCTION

**O**NE of the most important tasks in a distributed camera network is to identify and track common objects across disparate camera views. It is a difficult problem because image features like corners or scale-invariant feature transform (SIFT) may vary significantly between different camera views due to disparity, occlusions, and variation in illumination. One possible solution is to utilize semantically rich visual features based either on intrinsic characteristics such as faces or gaits, or artificial marks like jersey numbers or special-colored tags. We call the problem of identifying distinctive visual features on an object the "visual tagging" problem.
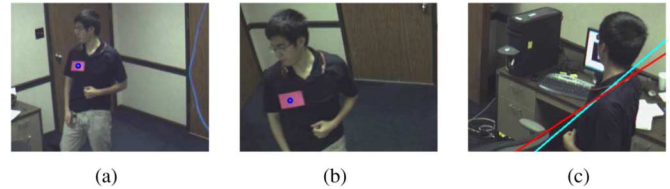
Fig. 1. Tag is visible only in two camera views Cam1 and Cam2 but not in Cam3, but its location in Cam3 can be uniquely determined by the intersection of the two epipolar lines. (a) Cam1 original. (b) Cam2 original. (c) Cam3 epipolar lines.

Even though visual tagging requires sophisticated classifiers for tracking and possible cooperation from surveillance subjects, it has a wide range of important applications. For example, using distinctive biometric features, visual tagging allows tracking of terrorist suspects across a large area like an airport that already has a network of video cameras in place. Automatic tracking of jersey numbers of players in a football field can be used to assist coaches in the study of different tactics and strategies. A recent application of visual tagging is to use special tags to identify individuals whose privacy needed to be protected in a video surveillance network [1]. Once a person is identified to possess a certain tag, his/her images in all cameras will be obfuscated to protect the identity. The ideal tag should be small, light, and easy to carry. This application of visual tagging on privacy protection is particularly challenging as the goal is to obfuscate the images of an individual in *all the cameras*, regardless of whether the small tag is visible to a particular camera. Provided that a tag is visible in two or more camera views, its 3-D location can be determined by a third camera by projecting the corresponding epipolar lines to the new view as shown in Fig. 1. To ensure proper localization of all the tags in the environment, a careful placement of cameras and a distributed protocol to exchange geometry information among cameras are indispensable. These are the topics addressed in this paper.

In this paper, we analyze the problem of visual tagging and propose the optimal design methodology of a smart camera network for visual tagging. The main contributions of the paper are as follows.

1) We present a novel stochastic visibility model to measure the performance of any camera configuration in triangulating visual tags, each of which is modeled as an oriented object in an arbitrary shaped 3-D environment. Our visibility model is based on the observed size of the object at different camera views, which is the key to the success of any appearance-based object identification scheme. Self-occlusion, occlusion by both environment and dynamic objects can be modeled using our framework.

2) Using binary integer programming, we develop two different algorithms for designing the optimal camera network configuration for visual tagging. The first formulation determines the number, the position, and the orientation of the cameras to achieve a target level of performance. The second formulation, under the constraint of a fixed number of cameras, computes the position and the orientation of the cameras to achieve the optimal level of performance. In order to cope with the high computational complexity, we also present a greedy strategy to approximate the solution.

3) With the optimal camera network configuration in place, we develop a distributed computation framework to locate human subjects wearing a small colored tag in a surveillance environment, and to protect the privacy of these subjects by visually erasing their presence in all camera views regardless of the visibility of the tags in specific cameras.

The rest of this paper is organized as follows. In Section II, we briefly review the state-of-the-art in camera placement problem and privacy protection schemes. In Section III, we present a generic model for measuring the performance of a particular camera placement. Section IV specializes the generic model to define a metric for the "visual tagging" problem based on the probability of observing a tag from multiple cameras. Using this metric, we formulate in Section V the search of the optimal camera placements as binary integer programming problems. With the optimal camera configuration in place, we describe how we use visual tagging to enhance privacy protection in Section VI. Experimental results using both simulations and real videos are presented in Section VII. We conclude this paper by discussing future work in Section VIII.

## II. RELATED WORK

The problem of finding the optimal camera placement has been studied for a long time. The earliest investigation can be traced back to the "art gallery problem" in computational geometry. This problem is the theoretical study on how to place cameras in an arbitrary-shaped polygon so as to cover the entire area [2]–[4]. Although Chvátal has shown in [5] that the upper bound of the number of cameras is $\lfloor n/3 \rfloor$, determining the minimum number of cameras turns out to be an NP complete problem [6]. While the theoretical difficulties of the camera placement problem are well understood, few solutions can be directly applied to realistic computer vision problems. Camera placement has also been studied in the field of photogrammetry for building the most accurate 3-D model. Various metrics such as visual hull [7] and viewpoint entropy [8] have been developed and optimization are realized by various types of *ad-hoc* searching and heuristics [9]. These techniques assume very dense placement of cameras and are not applicable to wide-area wide-baseline camera networks.

Recently, Ramakrishnan *et al.* propose a framework to study the performance of sensor coverage in wide-area sensor networks [10]. Unlike previous techniques, their approach takes into account the orientation of the object. They develop a metric to compute the probability of observing an object of random orientation from one sensor, and use that to recursively compute the performance for multiple sensors. While their approach can be used to study the performance of a fixed number of cameras,

it is not obvious on how to extend their scheme to find the optimal number of cameras as well as how to incorporate other constraints such as the visibility from more than one camera.

More sophisticated modeling pertinent to visual sensor networks are recently proposed in [11]–[13]. The sophistication in their visibility models comes at a high computational cost for the optimization. For example, the simulated annealing scheme used in [12] takes several hours to find the optimal placements of four cameras in a room. Other optimization schemes such as hill-climbing [11], semi-definite programming [13], and evolutionary approach [14] all prove to be computationally intensive and prone to local minima.

Alternatively, the optimization can be tackled in the discrete domain: Horster and Lienhart develop a flexible camera placement model by discretizing the space into grid and denoting the possible placement of camera as a binary variable over each grid point [15]. The optimal camera configuration is formulated as an integer linear programming problem which can incorporate different constraints and cost functions pertinent to a particular application. A similar idea was also proposed in [16]. While our approach follows a similar optimization strategy, we develop a probabilistic approach to capture the uncertainty of object orientation and mutual occlusion. In addition, our visibility model is far more realistic: unlike [15] in which the field of view of a camera is modeled as a 2-D fixed-size triangle, ours is based on measuring the image size of the object as observed by a pinhole camera with arbitrary 3-D location and pose. Our motivation is based on the fact that the image size of the object is key to the success of any appearance-based object identification scheme. While the optimization scheme described in [15] can theoretically be used for triangulating objects, their results as well as those from [16] are limited to maximizing sensor coverage. Our approach, on the other hand, directly tackles the problem of visual tagging in which each object needs to be visible by two or more cameras.

The topic of tracking objects in a multiple-camera system is also heavily studied in recent years [17]–[20]. Most tracking systems use a visual tracker for each camera and integrate the results from different cameras through a subsequent registration procedure. In [17], Foresti *et al.* survey on the low-level signal processing techniques suitable for these kinds of multiple-camera tracking applications. For multiple-camera registration, the majority of the existing techniques adopt a calibration procedure to build either an image-ground homography or a direct mapping between cameras [18], [19]. In [20], the authors combine visual tracking and camera calibration to reposition a set of active cameras in focusing and tracking the objects of interest. A common theme among all of these work is their reliance on the robustness of visual tracking, which certainly depends on the placement of cameras. The impact of different camera placements on visual tracking, however, has not been carefully studied. Our contribution here is to provide a flexible probabilistic model in studying the performance of different camera networks and finding the optimal configuration for various vision tasks, including object tracking and identification.

The application of visual tagging in privacy protected video surveillance is first proposed by Schiff *et al.* [1]. They use an Adaboost classifier to identify hard hats and apply particular filtering to track them through time. The privacy of an individual wearing such a hat is protected by having his/her face

covered by a black box. The choice of hard hats is to provide a significant target for tracking and recognition and to minimize occlusion. On the other hand, its prominent presence may be singled out in certain environments. In addition, their scheme does not incorporate any cues from multiple cameras. While the scheme proposed by Schiff *et al.* is the first to address the identification problem in privacy protected video surveillance, many others have proposed schemes to obfuscate identity information in video. Chen *et al.* [21] present a system obscuring the human body while preserving the structure and motion information. Newton *et al.* develop a face modification algorithm to counter face recognition [22]. Wickramasuri *et al.* use RFID to track individuals and visually replace them with a static background [23]. Our previous work demonstrates an efficient video in-painting algorithm to erase individuals for privacy protection [24] and presents a data hiding scheme to preserve privacy information in compressed video [25]. All of the above schemes can benefit from the optimal camera configuration and the communication protocols of sharing objects location information proposed in this paper.

In [26], we present an adaptive binary integer programming method for camera placement and preliminary experimental results of visual tagging. Based on the foundation established in [26], we have improved our methods in the following aspects: first, instead of restricting only to two-dimensional convex environments, our visibility model can now cope with three-dimensional environments with arbitrary shapes and obstacles. Second, our new model has incorporated the effect of occlusion of the tag from other objects in the environment. Third, our new optimal camera placement formulation supports a much larger search space so that we can obtain the same level of performance with fewer cameras. Fourth, we significantly reduce the computational complexity by developing a greedy algorithm whose performance rivals that of the exhaustive search. Fifth, in building the privacy protected surveillance network, we improve the performance in locating color tags using epipolar geometry across multiple camera views. Finally, additional experimental results are provided to demonstrate the performance of our algorithms.

## III. General Visibility Model

In this section, we outline a general model to compute the visibility of a single tag centered at $P$ in a confined three-dimensional environment. We assume that the 3-D environment has vertical walls with piecewise linear contours. Obstacles are modeled as columns of finite height and polyhedral cross sections. Whether the actual tag is the face of a subject or an artificial object, it is reasonable to model each tag as a small flat surface perpendicular to the ground plane. We further assume that all the tags are of the same square shape with known edge length $2w$. Without any specific knowledge of the height of individuals, we assume that the centers of all the tags lie on the same plane $\Gamma$ parallel to the ground plane. This assumption does not hold in the real world as individuals are of different height. Nevertheless, as we will demonstrate in Section VII-A, such height variation does not much affect the overall visibility measurements. While our model restricts the tags to be on the same plane, we place no restriction on the 3-D positions, yaw, and pitch angles of the cameras in the camera network.
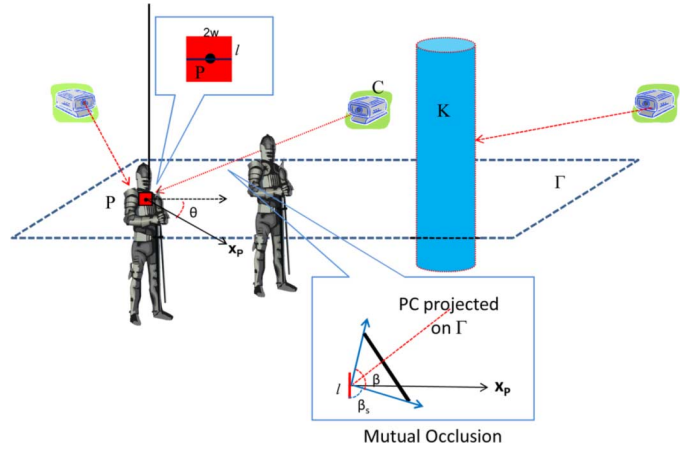


Fig. 2. Three-dimensional visibility model of a tag with orientation $\mathbf{x_P}$; $\Gamma$ is the plane that contains the tag center $P$. Mutual occlusion is modeled by a maximum block angle $\beta$ and its location $\beta_s$. $K$ describes the obstacles and wall boundaries. Cameras of arbitrary yaw and pitch angles can be placed anywhere in the 3-D environment.

Given the number of cameras and their placement in the environment, we define the visibility $V$ of a tag using *an aggregate measure of the projected size of a tag on the image planes of different cameras*. The projected size of the tag is very important as the image of the tag has to be large enough to be automatically identified at each camera view. Due to the camera projection of the 3-D world to the image plane, the image of the square tag can be an arbitrary quadrilateral. While it is possible to precisely calculate the area of this image, it is sufficient to use an approximation for our visibility calculation. Thus, we measure the projected length of the line segment $l$ at the intersection between the tag and the horizontal plane $\Gamma$. The actual 3-D length of $l$ is $2w$, and since the center of the tag always lies on $l$, the projected length of $l$ is representative of the overall projected size of the tag.

Next we identify the set of random and fixed parameters that affects $V$. The fact that we have chosen to measure the projected length of $l$ instead of the projected area of the tag greatly simplifies the parametrization of $V$. Given a camera network, the visibility function of a tag can be parameterized as $V(P, \mathbf{v_P}, \beta_s \,|\, w, K)$, where $P, \mathbf{v_P}, \beta_s$ are random parameters about the tag; $K$ and $w$ are fixed environmental parameters. These parameters are defined in the sequel and illustrated in Fig. 2.

$P$ defines the 2-D coordinates of the center of the tag on the plane $\Gamma$. $\mathbf{v_P}$ is the pose vector of the tag. As we assume the tag is perpendicular to the ground plane, the pose vector $\mathbf{v_P}$ lies on the plane $\Gamma$ and has a single degree of freedom—the orientation angle $\theta$ with respect to a reference direction. Note the dependency of $V$ on $\mathbf{v_P}$ allows us to model self-occlusion—the tag is being occluded by the person who is wearing it. The tag will not be visible to a camera if the pose vector is pointing away from the camera. While self-occlusion can be succinctly captured by a single pose vector, the modeling of mutual occlusion involves the number of neighboring objects, their distances to the tag and the positions of the cameras. The precise modeling of mutual occlusion can be extremely complicated. In our model, we choose the worst-case approach by considering a fixed occlusion angle

$\beta$ measured at the center of the tag on the $\Gamma$ plane. Mutual occlusion is said to occur if the projection of the line of sight on the $\Gamma$ plane falls within the range of the occlusion angle. In other words, we model the occluder as a cylindrical wall of infinite height around the tag partially blocking a fixed visibility angle of $\beta$ at random starting position $\beta_s$. $w$ is half of the edge length of the tag which is a known parameter. The shape of the environment is encapsulated in the fixed parameter set $K$ which contains a list of oriented vertical planes that describe the boundary wall and obstacles of finite height. It is straightforward to use $K$ to compute whether there is a direct line of sight between an arbitrary point in the environment and a camera. The specific visibility function suitable for visual tagging will be described in Section IV.

To correctly identify and track any visual tag, a typical classification algorithm would require the tag size on the image to be larger than a certain minimum size, though a larger projected size usually does not make much difference. For example, a color tag detector needs a threshold to differentiate the tag from noises, and a face detector needs a face image large enough to observe the facial features. On the other hand, the information gain does not increase as the projected object size increases beyond a certain value. Therefore, the threshold version can represent our problem much better than the absolute image size. Assuming that this minimum threshold on image size is $T$ pixels, this requirement can be modeled by binarizing the visibility function as follows:

$$V_b(P, \mathbf{v_P}, \beta_s \,|\, w, K, T)$$
$$= \begin{cases} 1, & \text{if } V(P, \mathbf{v_P}, \beta_s \,|\, w, K) > T \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Finally, we define $\eta$, the *mean visibility*, to be the metric for measuring the average visibility of $P$ over the entire parameter space

$$\eta = \int V_b(P, \mathbf{v_P}, \beta_s \,|\, w, K, T) \cdot f(P, \mathbf{v_P}, \beta_s) \, dP \, d\mathbf{v_P} \, d\beta_s \quad (2)$$

where $f(P, \mathbf{v_P}, \beta_s)$ is the prior distribution that can incorporate prior knowledge about the environment—for example, if an application is interested in locating faces, the likelihood of the head positions and poses are affected by furnishings and attractions such as television sets and paintings. Except for the most straightforward environment such as a single camera in a convex environment discussed in [26], (2) does not admit a closed-form solution. Nevertheless, it can be estimated by using standard Monte Carlo sampling and its many variants. The details of our Monte Carlo sampling strategy is discussed in Section VII.

## IV. VISIBILITY MODEL FOR VISUAL TAGGING

In this section, we present a visibility model for the visual tagging problem. This model is a specialization of the general model in Section III. The goal is to design a visibility function $V(P, \mathbf{v_P}, \beta_s \,|\, w, K)$ that can measure the performance of a camera network in capturing a tag in multiple camera views. We will first present the geometry for the visibility from one camera in Section IV-A and then show a simple extension to create $V(P, \mathbf{v_P}, \beta_s \,|\, w, K)$ for arbitrary number of cameras in Section IV-B. Table I provides a quick summary of all the symbols used in our derivation.

| | |
|---|---|
| $P$ | Tag center |
| $\mathbf{v_P}$ | Tag pose vector with $\|\mathbf{v_P}\| = 1$ |
| $\Gamma$ | Horizontal Plane where $P$ lies |
| $\beta$ | Occlusion angle measured at $P$ on $\Gamma$ |
| $\beta_s$ | Starting position of the occlusion angle |
| $\mathbf{v_V}$ | Normal of $\Gamma$ with $\|\mathbf{v_V}\| = 1$ |
| $l$ | Line segment at the intersection between the tag and $\Gamma$ |
| $P_{l1}, P_{l2}$ | Two end points of $l$ |
| $C$ | Camera's Center of Projection |
| $\mathbf{v_C}$ | Camera's direction of the projection or pose vector with $\|\mathbf{v_C}\| = 1$ |
| $\alpha$ | Angle between $\mathbf{v_P}$ and the vector from $P$ to $C$ |
| $\Pi$ | Image plane, a finite-size rectangle with normal $\mathbf{v_C}$ and its distance from the center of projection defines the focal length. |
| $O$ | Center of the $\Pi$ plane |
| $P', P'_{l1}, P'_{l2}, l'$ | Projection of $P$, $P_{l1}$, $P_{l2}$ and $l$ on $\Pi$ |
| $K$ | Fixed environment parameters about the walls and obstacles |
| $\Upsilon$ | A point in the camera space parameterized by $C$ and $\mathbf{v_C}$, along with other derived quantities like $\Pi$ and $O$ |
| $\Lambda$ | A point in the tag space parameterized by $P$, $\mathbf{v_P}$ and $\beta_s$ |

### A. Visibility for Single Camera

Given a single camera with the camera center at $C$, it is straightforward to see that a tag at $P$ is visible at $C$ if and only if the following four conditions hold.

1) The tag is not occluded by any obstacle or wall. (environmental occlusion)
2) The tag is within the camera's field of view. (field of view)
3) The tag is not occluded by the person wearing it. (self-occlusion)
4) The tag is not occluded by other moving objects. (mutual occlusion)

Thus, we define the visibility function for one camera to be the projected length $\|l'\|$ on the image plane of the line segment $l$ across the tag if the above conditions are satisfied, and zero otherwise. In the sequel, we demonstrate how the projected length is calculated and show how we check each of the four conditions.

Fig. 3 shows the projection of $l$, delimited by $P_{l1}$ and $P_{l2}$, onto the image plane $\Pi$. Based on the assumptions that all the tag centers has the same elevation and all tag planes are vertical, we can analytically derive the formulae for $P_{l1}, P_{l2}$ as follows: as $l$ is perpendicular to both the unit pose vector of the tag $\mathbf{v_P}$ and the unit normal vector $\mathbf{v_V}$ to the plane $\Gamma$, we have $P_{l1,2} = P \pm w(\mathbf{v_P} \times \mathbf{v_V})$. Their projections $P'_{l1}$ and $P'_{l2}$ lie on the intersections between the image plane $\Pi$ and the light rays $CP_{l1}$ and $CP_{l2}$, respectively. For $i = 1, 2$, any point $X$ on $\Pi$ must satisfy $\langle \mathbf{v_C}, X - O \rangle = 0$, where $\langle \cdot, \cdot \rangle$ indicates inner product, and any point $X$ on the line $CP_{li}$ must satisfy $X = C + \lambda(P_{li} - C)$. Thus, $P'_{li}$ for $i = 1, 2$ can be calculated as follows:

$$P'_{li} = C - \frac{\langle \mathbf{v_C}, O - C \rangle}{\langle \mathbf{v_C}, P_{li} - C \rangle} (P_{li} - C). \quad (3)$$

The projected length $\|l'\|$ is simply $\|P'_{l1} - P'_{l2}\|$.

After computing the projected length of the tag, we proceed to check the four visibility conditions as follows.

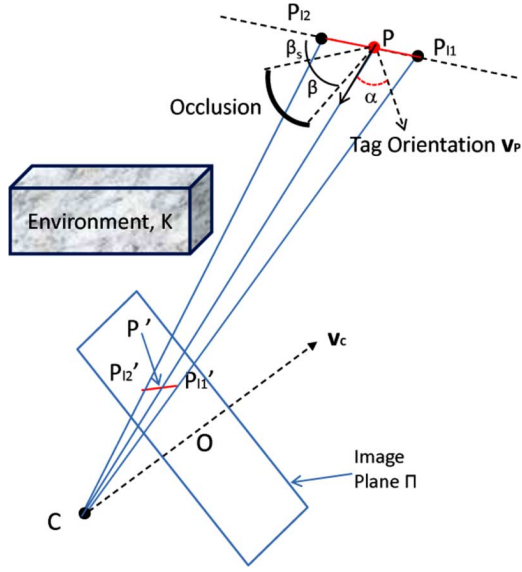1) **Environmental occlusion:** We assume that environmental occlusion occurs if the line segment connecting camera

Fig. 3. Projection of a single tag onto a camera.

center $C$ with the tag center $P$ intersects with some obstacle. While such an assumption does not take into account of partial occlusion, it is adequate for most visual tagging applications where the tag is much smaller than its distance from the camera. We represent this requirement as the following binary function:

$$\text{chkObstacle}(P, C, K)$$
$$= \begin{cases} 1, & \text{No obstacles intersect} \\ & \text{with line segment PC} \\ 0, & \text{else.} \end{cases} \quad (4)$$

Specifically, the obstacles are recorded in $K$ as a set of oriented vertical planes that describe the boundary wall and obstacles of finite height. Intersection between the line of sight $PC$ and each element in $K$ is computed. If there is no intersection within the confined environment or the points of intersection are higher than the height of the camera, no occlusion occurs due to the environment.

2) **Field of view:** Similar to determining environmental occlusion, we declare the tag to be in the field of view if the image $P'$ of the tag center is within the finite image plane $\Pi$. Using a similar derivation as in (3), the image $P'$ is computed as follows:

$$P' = C - \frac{\langle \mathbf{v_C}, O - C \rangle}{\langle \mathbf{v_C}, P - C \rangle}(P - C). \quad (5)$$

We then convert $P'$ to local image coordinates to determine if $P'$ is indeed within $\Pi$. We encapsulate this condition using the binary function $\text{chkFOV}(P, C, \mathbf{v_C}, \Pi, O)$ takes camera intrinsic parameters, tag location, pose vector as input, and returns a binary value indicating whether the center of the tag is within the camera's field of view.

3) **Self-occlusion:** As illustrated in Fig. 3, the tag is self-occluded if the angle $\alpha$ between the light of sight to the camera $C - P$ and the tag pose $\mathbf{v_P}$ exceeds $(\pi/2)$. We can represent this condition as a step function $U((\pi/2) - |\alpha|)$.

4) **Mutual occlusion:** In Section III, we model the worst-case occlusion using an angle $\beta$. As illustrated in Fig. 3, mutual occlusion occurs when the tag center or half the line segment $l$ is occluded. The angle $\beta$ is suspended at $P$ on the $\Gamma$ plane. Thus, occlusion occurs if the projection of the light of the sight $C - P$ on the $\Gamma$ plane at $P$ falls within the range of $[\beta_s, \beta_s + \beta]$. We represent this condition using the binary function chkOcclusion $(P, C, \mathbf{v_P}, \beta_s)$ which returns one for no occlusion and zero otherwise.

Combining both $\|l'\|$ and the four visibility conditions, we define the projected length of an oriented tag with respect to camera $\Upsilon$ as $I(P, \mathbf{v_P}, \beta_s | K, \Upsilon)$ follows:

$$\begin{aligned} I&(P, \mathbf{v_P}, \beta_s | w, K, \Upsilon) \\ &= \|l'\| \cdot \text{chkObstacle}(P, C, K) \\ &\quad \cdot \text{chkFOV}(P, C, \mathbf{v_C}, \Pi, O) \\ &\quad \cdot U\left(\frac{\pi}{2} - |\alpha|\right) \\ &\quad \cdot \text{chkOcclusion}(P, C, \mathbf{v_P}, \beta_s) \quad (6) \end{aligned}$$

where $\Upsilon$ includes all camera parameters including $\Pi, O$ and $C$. As stated in Section III, a threshold version is usually more convenient

$$\begin{aligned} I_b&(P, \mathbf{v_P}, \beta_s | w, K, \Upsilon, T) \\ &= \begin{cases} 1, & \text{if } I(P, \mathbf{v_P}, \beta_s | w, K, \Upsilon) > T \\ 0, & \text{otherwise.} \end{cases} \quad (7) \end{aligned}$$

### B. Visibility for Visual Tagging

To extend the single-camera case in Section IV-A to multiple cameras, we note that the visibility of the tag from one camera does not affect the other, and thus, each camera can be treated independently. The visual tagging problem requires that the tag must be visible by at least two cameras. Given $N$ cameras $\Upsilon_1, \Upsilon_2, \ldots, \Upsilon_N$, we define the visibility function $V(P, \mathbf{v_P}, \beta_s | w, K)$ for visual tagging to be the *second largest projected tag size among all the cameras*

$$\begin{aligned} V&(P, \mathbf{v_P}, \beta_s | w, K) \\ &= \max_{i \in \{1,2,\ldots,N\}, i \neq k} I(P, \mathbf{v_P}, \beta_s | w, K, \Upsilon_i) \quad (8) \end{aligned}$$

or the accompanying threshold version $V_b(P, \mathbf{v_P}, \beta_s | w, K, T)$

$$\begin{aligned} V_b&(P, \mathbf{v_P}, \beta_s | w, K, T) \\ &= \begin{cases} 1, & \text{if } V(P, \mathbf{v_P}, \beta_s | w, K) > T \\ 0, & \text{otherwise} \end{cases} \quad (9) \end{aligned}$$

where $I(P, \mathbf{v_P}, \beta_s | w, K, \Upsilon_i)$ is the visibility of tag $P$ with respect to camera $\Upsilon_i$ as defined in (6) and $i = k$ is the camera that captures the largest image of the tag or

$$k = \arg \max_{j \in \{1,2,\ldots,N\}} I(P, \mathbf{v_P}, \beta_s | w, K, \Upsilon_j).$$

Using this definition of visibility function, we can then compute the mean visibility $\eta$ as defined in (2) as a measure of the average likelihood of a random tag being observed by two or more cameras.

Even if the environment is densely covered with cameras, there is no guarantee that a tag at an arbitrary position will be visible to two cameras—a tag next to and facing the wall is only visible if there are two cameras right in front of the tag. In the actual design of camera networks, we would like to avoid such pathological cases and adopt the design if most of the environment is perfectly visible. We call the area in the environment a *perfect zone* in which a tag of half-length $w$, regardless of its pose, is visible to two or more cameras. In other words, the perfect zone can be defined as

$$
\begin{aligned}
\text{perfect zone} \\
= \{P : V(P, \mathbf{v_P}, \beta_s \,|\, w, K) > T \text{ for all } \mathbf{v_P}\} \\
= \{P : V_b(P, \mathbf{v_P}, \beta_s \,|\, w, K, T) = 1 \text{ for all } \mathbf{v_P}\}. \quad (10)
\end{aligned}
$$

The identification of the perfect zone is important in choosing the grid points used in the optimal camera placement algorithms which we will introduce in the next section.

## V. OPTIMAL CAMERA PLACEMENT

The goal of an optimal camera placement is to identify, among all possible camera network configurations, the one that maximizes the visibility function given by (9). As (9) does not possess an analytic form, it is very difficult to apply conventional continuous optimization strategies such as variational techniques or convex programming. As such, we follow a similar approach as in [15] by finding an approximate solution over a discretization of two spaces—the space of possible camera configurations and the space of tag location and orientation. The optimization problem over the discrete spaces is formulated as a binary integer programming (BIP) problem in which binary variables are used to indicate whether a camera is placed at a specific grid point and whether a tag is observable at a particular location and orientation.

After describing the discretization of our parameter spaces in Section V-A, we describe three different algorithms in computing the optimal camera configuration. The first algorithm MIN_CAM, introduced in Section V-B, finds the minimum number of cameras so that all the tag grid points must be observed by at least two cameras. This algorithm is efficient due to its progressive refinement in grid density, and it is useful for those applications where the visual tagging requirement in the environment needs to be strictly enforced. This stringent visibility requirement, on the other hand, may inflate the number of cameras needed. In many practical situations, limited resource dictates the number of cameras to be used. In Section V-C, we introduce FIX_CAM which maximizes the performance under a fixed number of cameras by relaxing some of the visibility requirements. Various cost functions can be used in FIX_CAM to customize the emphasis on either visibility or the number of cameras. At the heart of both MIN_CAM and FIX_CAM is the application of integer programming, a NP-hard problem [27], whose exact solution on a large number of grid points can take an inordinate amount of time to compute. In Section V-D, we introduce GREEDY, a greedy algorithm that can be used with either FIX_CAM or MIN_CAM. While no precise error bound is provided in this manuscript, experimental results show that
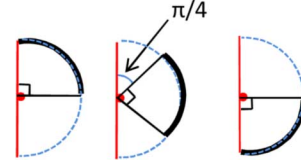


Fig. 4. Discretization to guarantee occlusion less than $\beta = \pi/4$ at any position will be covered in one of the above three cases: $[0, (\pi/2))$, $[(\pi/4), (3\pi/(2)))$, and $[(\pi/2), \pi)$.

the approximate solutions are very close to the exact ones in small-scale problems.

### A. Discretization of Camera and Tag Spaces

The design parameters for a camera network include the number of cameras, their 3-D locations, as well as their yaw and pitch angles. The number of cameras is either an output discrete variable or a constraint in our formulation. The elevation of the cameras is usually constrained by the environment. As such, our optimization does not search for the optimal elevation but rather has the user input it as a fixed value. For simplicity, we assume that all cameras have the same elevation, but it is a simple change in our code to allow different elevation constraints to be used in different parts of the environment. The remaining 4-D camera space: the 2-D location, yaw, and pitch angles are discretized into a uniform lattice *gridC* of $N_c$ camera grid points, denoted as $\{\Upsilon_i : i = 1, 2, \ldots, N_c\}$.

The unknown parameters about the tag in computing the visibility function (9) include the location of the tag center $P$, the pose of the tag $\mathbf{v_P}$, and the starting position of the worse-case occlusion angle $\beta_s$. Our assumptions stated in Section III have the tag center lied on a 2-D plane and the pose restricted to a 1-D angle with respect to a reference direction. As for occlusion, our goal is to perform the worst-case analysis so that as long as the occlusion angle is less than a given $\beta$ as defined in Section III, our solution is guaranteed to work no matter where the occluder is. As such, a straightforward quantization of the starting position $\beta_s$ of the occlusion angle will not work—an occlusion angle of $\beta$ starting anywhere between grid points will occlude additional view. To simultaneously discretize the space and maintain the guarantee, we select a larger occlusion angle $\beta_m > \beta$ and quantize the starting position of the occlusion angle using a step-size of $\beta_\Delta = \beta_m - \beta$. The occlusion angles considered under this discretization will then be $\{[i\beta_\Delta, i\beta_\Delta + \beta_m) : i = 0, \ldots, N_\beta - 1\}$, where $N_\beta = \lceil (\pi - \beta_m)/\beta_\Delta \rceil$. This guarantees that any occlusion angle less than or equal to $\beta$ will be covered by one of the occlusion angles. Fig. 4 shows an example of $\beta = \beta_\Delta = \pi/4$ and $\beta_m = \pi/2$. Combining these three quantities together, we discretize the 4-D tag space into a uniform lattice *gridP* with $N_p$ tag grid points $\{\Lambda_i : i = 1, 2, \ldots, N_p\}$.

Given a camera grid point $\Upsilon_i$ and a tag grid point $\Lambda_j$, we can explicitly evaluate the threshold single-camera visibility function (7) which we now rename as $I(\Lambda_j \,|\, w, T, K, \Upsilon_i)$ with $\Lambda_j$ representing the grid point for the space of $P$, $\mathbf{v_P}$ and $\beta_s$; $w$ the size of the tag; $T$ is the visibility threshold; $K$ is the environmental parameter; and $\Upsilon_i$ is the camera grid point. The numerical values of $I(\Lambda_j \,|\, w, T, K, \Upsilon_i)$ will then be used in formulating cost constraints in our optimal camera placement algorithms.

### B. MIN_CAM: Minimizing the Number of Cameras for a Target Visibility

MIN_CAM estimates the minimum number of cameras which can provide a mean visibility $\eta$ equal to or higher than a given threshold $\eta_t$. There are two main characteristics of MIN_CAM: first, $\eta$ is computed not on the discrete tag space but on the actual continuous space using Monte Carlo simulation. As such, the measurement is independent of the discretization. Furthermore, if the discretization of the tag space is done with enough prior knowledge of the environment, MIN_CAM can achieve the target using very few grid points. This is important as the complexity of BIP depends greatly on the number of constraints which is proportional to the number of grid points. Second, the visual tagging requirements are formulated as constraints rather than the cost function in the BIP formulation of MIN_CAM. Thus, the solution will guarantee the chosen tag grid points be visible at two or more cameras. While this is useful to those applications where the visual tagging requirement in the environment needs to be strictly enforced, they may inflate the number of cameras needed to capture some poorly chosen grid points. Before describing the details of how we handle this problem, we first describe the BIP formulation in MIN_CAM.

We first associate each camera grid point $\Upsilon_i$ in *gridC* with a binary variable $b_i$ such that

$$b_i = \begin{cases} 1, & \text{if a camera is present at } \Upsilon_i \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

The optimization problem can be described as the minimization of the number of cameras

$$\min_{b_i} \sum_{i=1}^{N_c} b_i \quad (12)$$

subjected to the following two constraints: first, for each tag point $\Lambda_j$ in *gridP*, we have

$$\sum_{i=1}^{N_c} b_i \cdot I_b(\Lambda_j \,|\, w, T, K, \Upsilon_i) \geq 2. \quad (13)$$

This constraint represents the requirement of visual tagging that all tags must be visible at two or more cameras. As defined in (7), $I_b(\Lambda_j \,|\, w, T, K, \Upsilon_i)$ measures the visibility of tag $\Lambda_j$ with respect to camera at $\Upsilon_i$. In other words, $\Lambda_j$ satisfying the constraint (13) must be in the perfect zone. Second, for each camera location $(x, y)$, we have

$$\sum_{\text{all } \Upsilon_i \text{ at } (x,y)} b_i \leq 1. \quad (14)$$

These are a set of inequalities guaranteeing that only one camera is placed at any spatial location. The optimization problem in (12) with constraints (13) and (14) forms a standard BIP problem.

The solution to the above BIP problem obviously depends on the selection of grid points in *gridP* and *gridC*. While *gridC* is usually predefined according to the constraint of the environment, there is no guarantee that, as alluded to in Section IV-B, a tag at a random location can be visible by two cameras even if there is a camera at every camera grid point. Thus, tag grid points must be placed intelligently—tag grid points away from obstacles and walls are usually easier to observe. On the other hand, focusing only on areas away from the obstacles may produce a subpar result when measured over the entire environment. To balance the two considerations, we solve the BIP repeatedly over a progressively refined *gridP* over the spatial dimensions until the target $\eta$, measured over the entire continuous environment, is satisfied. One possible refinement strategy is to have *gridP* started from a single grid point at the middle of the environment, and grew uniformly in density within the interior of the environment but remains at least one interval away from the boundary. If the BIP fails to return a solution, the algorithm will randomly remove half of the newly added tag grid points. The iteration terminates when the target $\eta_t$ is achieved or all the newly added grid points are removed. The above process is summarized in Algorithm 1.

**Input**: initial grid points for cameras $gridC$ and tag
$\quad\quad\quad gridP$, $\eta_t$, maximum grid density $maxDensity$
**Output**: Camera placement $camPlace$
Set $\eta = 0$, $newP = \emptyset$;
**while** $\eta \leq \eta_t$ **do**
$\quad$ **foreach** $\Upsilon_i$ *in* $gridC$ **do**
$\quad\quad$ **foreach** $\Lambda_j$ *in* $gridP \cup newP$ **do**
$\quad\quad\quad$ Calculate $I_b(\Lambda_j | w, T, K, \Upsilon_i)$;
$\quad\quad$ **end**
$\quad$ **end**
$\quad$ Solve
$\quad$ $newCamPlace = \text{BIP\_solver}(gridC, gridP, I_b)$;
$\quad$ **if** $newCamPlace == \emptyset$ **then**
$\quad\quad$ **if** $|newP| == 1$ **then**
$\quad\quad\quad$ break, return failure ;
$\quad\quad$ Randomly remove half of the elements from
$\quad\quad$ $newP$;
$\quad$ **else**
$\quad\quad$ $camPlace = newCamPlace$;
$\quad\quad$ $gridP = gridP \cup newP$;
$\quad\quad$ $newP =$ new grid points created by halving the
$\quad\quad$ spatial separation;
$\quad\quad$ $newP = newP \setminus gridP$;
$\quad\quad$ Calculate $\eta$ for $camPlace$ by Monte Carlo
$\quad\quad$ Sampling;
$\quad$ **end**
**end**

### C. FIX_CAM: Maximizing the Visibility for a Given Number of Cameras

A drawback of MIN_CAM is that it may need a large number of cameras in order to satisfy the visibility of all tag grid points. If the goal is to maximize the average visibility, a sensible way to reduce the number of cameras is to allow a small portion of the tag grid points not being observed by two or more cameras. As long as the tag grid is dense, such "blind spots" will be rare as guaranteed by a high average visibility. FIX_CAM is the algorithm that does precisely that.

We first define a set of binary variables on the tag grid $\{x_j : j = 1, \ldots, N_p\}$ indicating whether a tag on the $j$th tag point in *gridP* is visible at two or more cameras. In order to maximize the visibility, the objective function for BIP becomes

$$\max_{b_i} \sum_{j=1}^{N_p} x_j. \quad (15)$$

The relationship between the camera placement variables $b_i$'s as defined in (11) and visibility performance variables $x_j$'s can be

described by the following constraints. For each tag grid point $\Lambda_j$, we have

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j \mid w, T, K, \Upsilon_i) - (N_c + 1)x_j < 1 \qquad (16)$$

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j \mid w, T, K, \Upsilon_i) - 2x_j \geq 0. \qquad (17)$$

These two constraints effectively define the binary variable $x_j$: if $x_j = 1$, inequality (17) becomes

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j \mid w, T, K, \Upsilon_i) \geq 2$$

which means that a feasible solution of $b_i$'s must have the tag visible at two or more cameras. Inequality (16) becomes

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j \mid w, T, K, \Upsilon_i) < N_c + 2$$

which is always satisfied—the largest possible value from the left-hand size is $N_c$ corresponding to the case when there is a camera at every grid point and every tag point is observable by two or more cameras. If $x_j = 0$, inequality (16) becomes

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j \mid w, T, K, \Upsilon_i) < 1$$

which implies that the tag is not visible by two or more cameras. Inequality (17) is always satisfied as it becomes

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j \mid w, T, K, \Upsilon_i) \geq 0.$$

Two additional constraints are needed to complete the formulation: as the cost function focuses only on visibility, we need to constrain the number of cameras to be less than a maximum number of cameras as follows:

$$\sum_{j=1}^{N_c} b_j \leq m. \qquad (18)$$

We also keep the constraint in (14) to ensure only one camera is used at each spatial location.

Unlike MIN_CAM, the feasible solution set for FIX_CAM is non-empty—no matter how dense we set the discretization, the trivial case of no tag being observed will always satisfy the constraints. As such, we can simply run FIX_CAM on a fixed dense grid without any refinement of the tag space. FIX_CAM is more computationally intensive than MIN_CAM as there are two constraints for each tag grid point and usually a denser grid is used. Since this algorithm is more complex and requires the specification of a target number of cameras, a possible strategy is to use the MIN_CAM to estimate the approximate number of cameras and gradually reduce the number of cameras using FIX_CAM until the mean visibility falls below the target. Alternatively, we

can incorporate the minimization of the number of cameras by modifying the cost function defined in (15) to the following:

$$\max_{b_i} \sum_{j=1}^{N_p} x_j - \sigma \sum_{i=1}^{N_c} b_i \qquad (19)$$

where $\sigma$ is a user-defined parameter for balancing the maximization of visibility and minimization of the number of cameras. Experimental results using FIX_CAM will be shown in Section V-B.

### D. GREEDY: Greedy Algorithm to Speed Up BIP

BIP is a well-studied NP-hard combinatorial problem with plenty of approximation schemes such as branch-and-bound already implemented in software libraries such as lp_solve [28]. However, even these algorithms can be quite intensive if the search space is large. In this section, we introduce a simple greedy algorithm GREEDY that can be used for both MIN_CAM and FIX_CAM. Besides experimentally showing the effectiveness and the error margin of GREEDY in Section VII, we believe that the greedy approach is an appropriate approximation strategy due to the similarity of our problem to the set cover problem.

In the set cover problem, items can belong to multiple sets and the optimization goal is to minimize the number of sets to cover all the items. While finding the optimal solution to set covering is a NP-hard problem [27], it has been shown that the greedy approach is essentially the best that one can do to obtain an approximate solution [29]. We can draw the parallel between our problem with the set cover problem by considering each of the tag grid point as an item "belonging" to a camera grid point if the tag is visible at that camera. The set cover problem then minimizes the number of cameras needed, which is almost identical to MIN_CAM except for the fact that *visual tagging requires each tag to be visible by two or more cameras*. The FIX_CAM algorithm further allows some of the tag points not to be covered at all. It is still an open problem on whether these properties can be incorporated into the framework of set covering; our experimental results demonstrate that the greedy approach is a reasonable solution to our problem. The GREEDY algorithm is described in Algorithm 2.

In each round of the GREEDY algorithm, the camera grid point that can see the most number of tag grid points is selected and all the tag grid points visible at two or more cameras are removed. When using GREEDY to approximate MIN_CAM, we no longer need to refine the tag grids to reduce the computational efficiency. We can start with a fairly dense tag grid and set the camera bound $m$ to infinity. The GREEDY algorithm will terminate if the estimated mean visibility reaches the target $\eta_t$. When using GREEDY to approximate FIX_CAM, $\eta_t$ will be set to one and the GREEDY algorithm will terminate when the number of cameras reaches the upper bound $m$ as required by FIX_CAM.

## VI. VISUAL TAGGING FOR PRIVACY PROTECTED VIDEO SURVEILLANCE

In this section, we describe a system that uses visual tagging to protect privacy of selected individuals in a multiple-camera video surveillance network. The idea of privacy protected video

**Input**: initial grid points for cameras $gridC$ and tags
$gridP$, target mean visibility $\eta_t$ and the
maximum number of cameras $m$
**Output**: Camera placement $camPlace$
Set $U = gridC$, $V = \emptyset$, $W = gridP$, $camPlace = \emptyset$;
**while** $|V| < \eta_t \cdot |gridP|$ *or* $|camPlace| \leq m$ **do**
  $c =$ the camera grid point in $U$ that maximizes the
  number of visible tag grid points in $W$;
  $camPlace = camPlace \cup \{c\}$;
  $S =$ the subset of $grid$ that are visible by two or
  more cameras in $camPlace$;
  $V = V \cup S$;
  $W = W \setminus S$;
  Remove $c$ and all camera grid points in $U$ that share
  the same spatial location as $c$;
  **if** $U == \emptyset$ **then**
    $camPlace = \emptyset$;
    return;

**end**
Output $camPlace$
**Algorithm 2**: GREEDY: greedy search camera placement
algorithm

surveillance is to design algorithms for each camera to identify the trusted individual and obfuscate his/her identity using various image processing techniques. The main challenge lies in the simultaneous identification of the trusted individual in *all* camera views. As we have illustrated in Section I, visual tagging provides a reasonable solution and a camera network constructed using our optimal camera placement algorithms is ideal for constructing a privacy protected video surveillance network.

In our design, individuals whose privacy needed to be protected will be wearing small rectangular-shaped color tags. Each tag has a unique color for which we have prepared a specific color classifier. Our current implementation uses a Gaussian mixture model (GMM) classifier on the hue and saturation of each color. Using these classifiers, each camera identifies all pixels matching these colors, performs component grouping on pixels with the same color, and computes the centroid of each group. The related geometric information and their corresponding colors along with the camera's unique ID are sent to all other cameras using IP multicast protocol [30]. The advantage of using IP multicast is that adding a new camera amounts to subscribing to the multicast addresses of all existing cameras which do not need to flood the network with unwanted information or keep track of the status of other cameras. In order to localize occluded tags, fundamental matrices are estimated for each pair of cameras and epipolar lines of detected tags are broadcast to all cameras. Each camera then infers the locations of the tags by intersecting the epipolar lines. Except for the pathological case when multiple tags and the optical centers of the cameras they are visible to all lie on the same plane, all the intersections among epipolar lines within the image plane of each camera uniquely identify the locations of the tags. Objects associated with any directly-observed or occluded tags are then erased from the video by using an efficient object-template-based in-painting scheme [24].

## VII. EXPERIMENTAL RESULTS

Our framework of measuring and optimizing the design of camera configurations is flexible and has a great number of modeling parameters. It is impossible to exhaustively test all possible combinations. As such, we focus our effort into three areas

to demonstrate the validity and applicability of our proposed framework. First, we show various properties of MIN_CAM and FIX_CAM by varying different model parameters. We also provide experimental results to show how well GREEDY can approximate the results of FIX_CAM in Section VII-A. Second, we compare the optimal camera configurations computed by our techniques with other camera configurations in Section VII-B. Finally, we demonstrate the use of visual tagging for privacy protection in Section VII-C.

Before describing the experimental results, we provide an overview of the three different approaches we use for measuring visual tagging performance of our camera networks. They are Monte Carlo simulations, virtual environment simulations, and real-life experiments. The Monte Carlo simulation is a direct estimation of mean visibility $\eta$ defined in (2). Our Monte Carlo simulation uses several order more simulation points than the discrete grid points used in our optimization. As a result, $\eta$ can be robustly estimated. Furthermore, we can visualize the coverage of the environment by calculating the local average visibility at different spatial locations, such as the one depicted in Fig. 5(e). The gray-level of each pixel represents the visibility at each local region—the brighter the pixel, the higher the probability that it is visible at two or more cameras. A location is in the perfect zone if it is completely white.

Among all the evaluation approaches, Monte Carlo simulation is the easiest to execute because it is based on the same environment input and the same visibility model used by the optimization algorithms. This is also a drawback because we cannot use this technique to validate the assumptions made in our visibility model. Real-life implementation of the optimal camera configuration, on the other hand, requires a great of planning and installation effort. As an intermediate step before real-life testing, we use the OpenGL package provided in [31] to create a virtual 3-D environment that mimics the environment input and insert random-walking humanoids in it with some of them wearing red-color tags. $\eta$ is then estimated by the fraction of frames that have multiple views of the tag. The last type of evaluation is to physically construct a camera network, again with one person wearing a special color tag and other people moving around to create occlusion. The mean visibility is similarly measured except we need to resort to a more robust color detector. Examples of the both can be seen in Fig. 6.

### A. Optimal Camera Placement Experiments

All the simulations in this section assume a room of dimension 10 m × 10 m with a single obstacle and a square tag with edge length $w = 20$ cm long. For the camera and lens models, we assume a pixel width of 5.6 $\mu$m, focal length of 8 cm, and the field of view of 60 degrees. These parameters closely resembles the real cameras that we use in the real-life experiments. The threshold $T$ for visibility is set to five pixels which we find to be an adequate threshold for our color-tag detector.

We first study how MIN_CAM estimates the minimum number of cameras for a target mean visibility $\eta_t$ through tag grid refinement. For simplicity, we keep all the cameras at the same elevation as the tags and assume no mutual occlusion. The target mean visibility is set to be $\eta_t = 0.90$ and the algorithm reaches this target in four iterations. The output at each iteration are shown in Fig. 5. Fig. 5(a) and (e) show the first iteration.
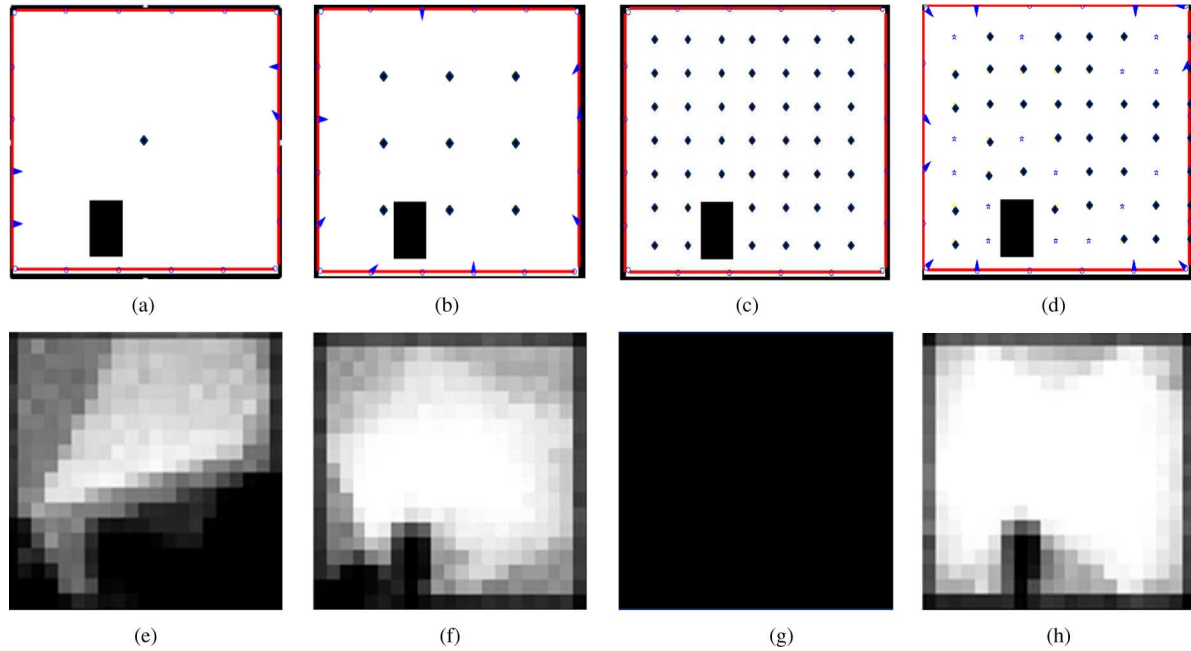
Fig. 5.   Four iterations of MIN_CAM: Figs. 5(a)–(d) show the grid points, the optimal camera pose, and location if applicable. The tag grid points are represented by black dots and randomly-discarded tag points as blue dots. The camera grid points are restricted at regular intervals along the red boundary of the environment. The blue arrowheads show the computed camera position and pose after each iteration. The corresponding spatial distribution of average visibility and the overall mean visibility $\eta$ are shown in Fig. 5(e) (Iteration one), 5(f) (Iteration two), 5(g) (Iteration three) and 5(h) (Iteration four). Note that $\eta = 0$ for Iteration three because there is no feasible solution. (a) Iteration one, (b) Iteration two, (c) Iteration three, (d) Iteration four, (e) $\eta = 0.4743$, (f) $\eta = 0.7776$, (g) $\eta = 0$, (h) $\eta = 0.9107$.

Fig. 5(a) shows the environment with one tag grid point (black dot) in the middle. The camera grid points are restricted at regular intervals along the red boundary of the environment and remain the same for all iterations. The blue arrows indicate the output position and pose of the cameras from the BIP solver. Fig. 5(e) shows the Monte Carlo simulation results. The mean visibility $\eta$ over the environment is estimated to be 0.4743. Since it is below the target $\eta_t$, the tag grid is refined as shown in Fig. 5(b) with the corresponding Monte Carlo simulation shown in Fig. 5(f). When the number of cameras increases from four to eight, $\eta$ increases to 0.7776. The next iteration shown in Fig. 5(c) grows the tag grid further. With so many constraints, the BIP solver fails to return a feasible solution. MIN_CAM then randomly discards roughly half of the newly added tag grid points. The discarded grid points are shown as blue dots in Fig. 5(d). With fewer grid points and hence fewer constraints, a solution is returned with 11 cameras. The corresponding Monte Carlo simulation shown in Fig. 5(h) gives $\eta = 0.9107$ which exceeds the target threshold and MIN_CAM terminates.

In the second experiment, we want to demonstrate the difference between FIX_CAM and MIN_CAM. Using the same environment as in Fig. 5(c), we run FIX_CAM to maximize the performance with 11 cameras. MIN_CAM fails to return a solution under this dense grid and after randomly discarding some of the tag grid points, outputs $\eta = 0.9107$ using 11 cameras. On the other hand, without any random tuning of the tag grid, FIX_CAM returns a solution of $\eta = 0.9205$ and the results are shown in Fig. 6(a) and (b). When we reduce the number of cameras to ten and rerun FIX_CAM, we manage to produce $\eta = 0.9170$ which still exceeds the results from MIN_CAM. This demonstrates that we can use FIX_CAM to fine-tune the

approximate result obtained by MIN_CAM. The camera configuration and the visibility distribution of using ten cameras are shown in Fig. 6(c) and (d), respectively.

Using the same setup, we repeat our FIX_CAM experiments using the GREEDY implementation. Our algorithm is implemented using MATLAB version 7.0 on a Xeon 2.1-Ghz machine with 4 Gigabytes of memory. The BIP solver inside the FIX_CAM algorithm is based on lp_solve [28]. We have tested both algorithms using 11, ten, nine, and eight maximum number of cameras. While changing the number of cameras does not change the number of constraints, the search space becomes more restrictive as we reduce the number of cameras. As such, it is progressively more difficult to prune the search space, making the solver resemble that of an exhaustive search. The results are summarized in Table II. For each run, three numerical values are reported: the total number of tag points visible to two or more cameras which is the actual minimized cost function, the running time, and the mean visibility estimated by Monte Carlo simulations. At eight cameras, GREEDY is 30 000 times faster than lp_solve but only 3% fewer visible tag points than the exact answer. It is also worthwhile to point out that the lp_solve fails to terminate when we refine the tag grid by halving the step-size at each dimension, while GREEDY uses essentially the same amount of time. The placement and visibility maps of the GREEDY algorithm that mirror those from FIX_CAM are shown in the second row of Fig. 6.

Armed with an efficient greedy algorithm, we can explore various modeling parameters in our framework. An assumption we made in the visibility model is that all the tag centers are in the same horizontal plane. This does not reflect the real world due to the different height of individuals. In the following exper-
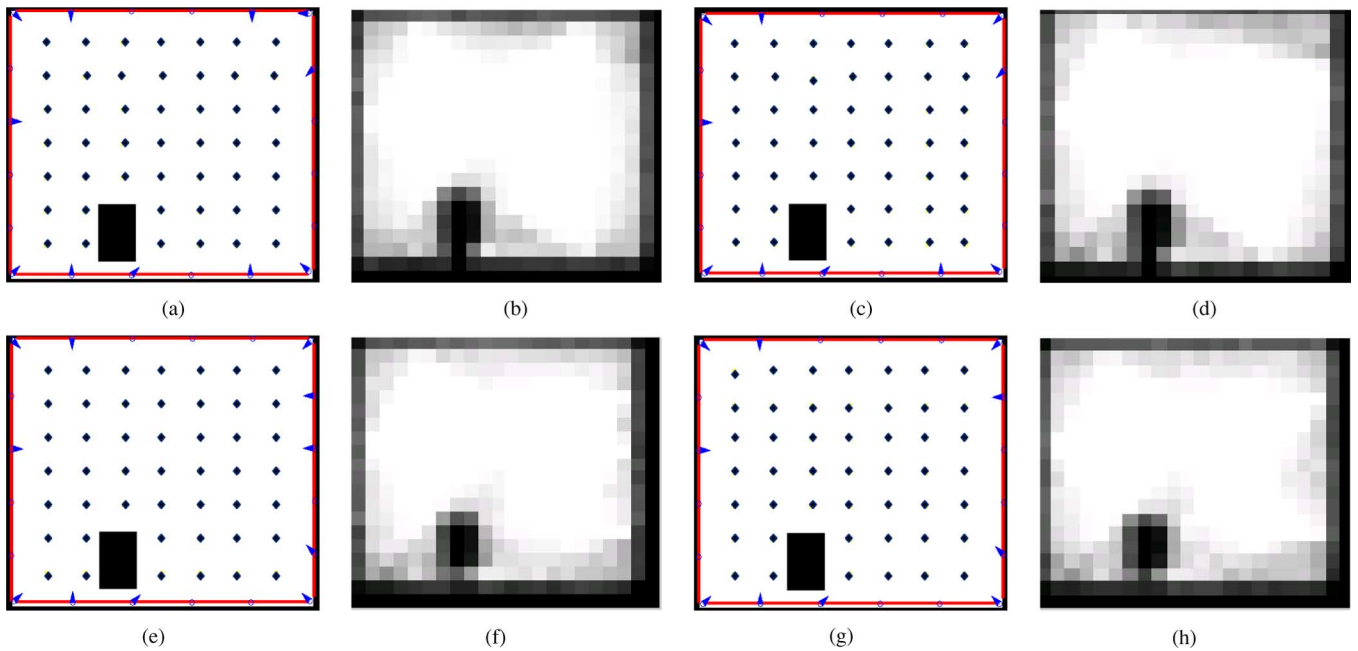
Fig. 6. Figs. 6(a)–(d) show the results of using FIX_CAM. Compared with the final iteration result in Fig. 5(h), we can see that FIX_CAM produces better $\eta$ than MIN_CAM using the same or fewer number of cameras. Figs. 6(e)–(h) show the same set of experiments using GREEDY as an approximation to FIX_CAM. (a) FIX_CAM: 11 cameras, (b) FIX_CAM: $\eta = 0.9205$, (c) FIX_CAM: 10 cameras, (d) FIX_CAM: $\eta = 0.9170$, (e) GREEDY: 11 cameras, (f) GREEDY: $\eta = 0.9245$, (g) GREEDY: 10 cameras, (h) GREEDY: $\eta = 0.9199$.

TABLE II
COMPARISON BETWEEN LP_SOLVE AND GREEDY

| No. of cameras | Lp_solve | | | Greedy | | |
|---|---|---|---|---|---|---|
| | Visible Tags (Total= 376) | Time(s) | $\eta$ | Visible Tags (Total= 376) | Time(s) | $\eta$ |
| Eleven | 373 | 1.20 | .9205 | 370 | 0.01 | .9245 |
| Ten | 370 | 46.36 | .9170 | 368 | 0.01 | .9199 |
| Nine | 365 | 113.01 | .9029 | 363 | 0.01 | .8956 |
| Eight | 362 | 382.72 | .8981 | 352 | 0.01 | .8761 |

TABLE III
EFFECT OF HEIGHT VARIATION ON $\eta$

| height model | +20 | -20 | +10 | -10 | Random |
|---|---|---|---|---|---|
| Change in $\eta$ | $-3.8\%$ | $-3.3\%$ | $-1.2\%$ | $-1.5\%$ | $-1.3\%$ |

iment, we examine the impact of the variation in height on the performance of a camera placement. Using the camera placement in Fig. 6(g), we simulate five different scenarios: the height of each person is 10 cm or 20 cm taller/shorter than the assumed height, as well as heights randomly drawn from a bi-normal distribution based on U.S. census data [32]. The changes in the average visibility are shown in Table III. They range from $-3.8\%$ to $-1.3\%$ which indicate that our assumption does not have a significant impact on the measured visibility.

Next, we consider the elevation of the cameras. In typical camera networks, cameras are usually installed at elevated positions to mitigate occlusion. The drawback of the elevation is that it has a smaller field of view when compared with the case when the camera is at the same elevation as the tags. By adjusting the pitch angle of an elevated camera, we can selectively move the field of view to various part of the environment. As we now add one more additional dimension of pitch angle, the optimization becomes significantly more difficult and GREEDY
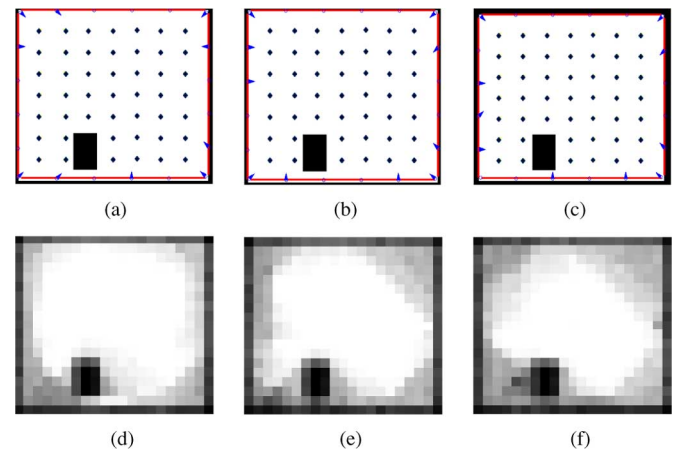


Fig. 7. Camera planning and Monte Carlo simulation results when the cameras are elevated to be 0.4, 0.8, and 1.2m above the tags. (a) Elevation $= 0.4$m, (b) Elevation $= 0.8$m, (c) Elevation $= 1.2$m, (d) $\eta = 0.9019$, (e) $\eta = 0.8714$, (f) $\eta = 0.8427$.

algorithm must be used. Fig. 7 shows the result for $m = 10$ cameras with three different elevations above the $\Gamma$ plane on which the centers of all the tags are located. As expected, the mean visibility reduces as we raise the cameras. The visibility maps in Fig. 7(d)–(f) show that our algorithm can adjust the pitch angles of the cameras so that the perfect region remains at the center of the environment to guarantee visibility from multiple cameras.

For the last experiment in this section, we present simulation results to show how our framework deals with mutual occlusion. Recall that we model occlusion as an occlusion angle of $\beta$ at the tag. Similar to the experiments on camera elevation, our occlusion model adds an additional dimension to the tag grid and thus we have to resort to the GREEDY algorithm. We would
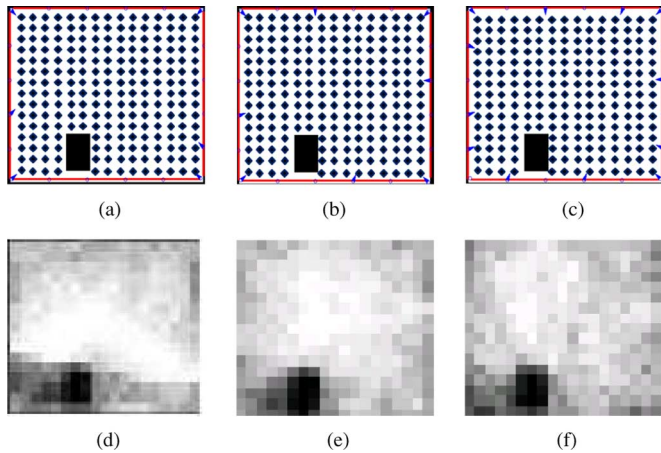
(a)                    (b)                    (c)

(d)                    (e)                    (f)

Fig. 8.   As the occlusion angle increases from $0°$ in Fig. 8(a)–(c), the required number of cameras increases as well when using GREEDY to achieve a target performance of $\eta_t = 0.8$. Fig. 8(d)–(f) are the correspondent visibility map under different occlusion angles. (a) $\beta = 0$; six cameras, (b) $\beta = 22.5°$; eight cameras, (c) $\beta = 45°$; twelve cameras, (d) $\eta = 0.8006$, (e) $\eta = 0.7877$, (f) $\eta = 0.7526$.

like to investigate how occlusion affects the number of cameras and the camera positions of the output configuration. As such, we use GREEDY to approximate MIN_CAM by identifying the minimum number of cameras to achieve a target level of visibility. We use a denser tag grid than before to minimize the difference between the actual mean visibility and that estimated by GREEDY over the discrete tag grid. The tag grid we use is $16 \times 16$ spatially with 16 different orientations. We set the target to be $\eta_t = 0.8$ and test different occlusion angle $\beta$ at $0°$, $22.5°$, and $45°$. As explained earlier in Section V-A, our discretization uses a slightly larger occlusion angle to guarantee worst-case analysis—we use $\beta_m = 32.5°$ for $\beta = 22.5°$ and $\beta_m = 65°$ for $\beta = 45°$. In the Monte Carlo simulation, we put the occlusion angle at a random position of each sample point. The results are shown in Fig. 8. We can see that even with increasing the number of cameras from six to eight to 12, the resulting mean visibility reduces slightly when the occlusion angle increases. Another interesting observation from the visibility maps in Figs. 8(d)–(f) is that the perfect region, indicated by the white pixels, dwindles as occlusion increases. This is reasonable because it is difficult for a tag to be visible at all orientation in the presence of occlusion.

### B. Comparison With Other Camera Placement Strategies

In this section, we compare our optimal camera placements with two different placement strategies. The first one is uniform placement—assuming that the cameras are restricted along the boundary of the environment, the most intuitive scheme is to place them at regular intervals on the boundary, each pointing towards the center of the room. The second one is based on the optimal strategy proposed in [15]. It is difficult to fully compare our scheme with that in [15] due to the many differences between them including the following.

- **Visibility model:** The camera model in [15] is a 2-D coverage triangle with no modeling of camera elevation, projected size of tag on image planes, tag pose for self-occlusion, and mutual occlusion. It is possible to use the formulation in [15] to model visibility from two or more cameras
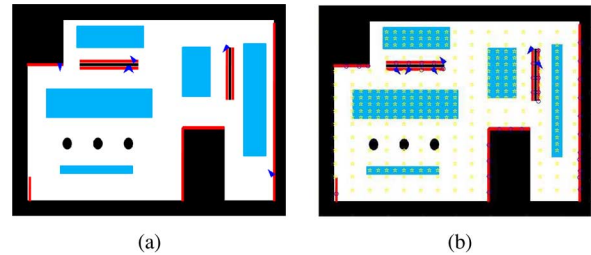
though all results in [15] are based on single-camera visibility.
- **Cost model:** While a variety of cost models for different cameras can be incorporated in [15], our formulation deals with a single type of camera.
- **Efficient implementation:** Though both schemes use greedy search, [15] focuses on random sampling of spaces while ours use a fixed grid.

Among all the differences, we believe that the visibility model is the most important and interesting difference between our schemes. In order to have a fair comparison between the two schemes, we have re-implemented the visibility model in [15] that supports both single and multiple camera visibility in the context of our own cost model and greedy implementation. As a sanity check, we first compare the optimal camera placement under single camera visibility between our implementation of their scheme and one of their results published in [15]. The computed placements over an environment used in [15] are shown in Fig. 9. The blue regions are emphasized areas in which more sampling points are placed. Out of the six cameras used, three of them have exactly the same pose as those in [15]. Among the remaining three, two of them are on the left half plane off by two grid points and the last one is on the right half, moved from the bottom corner in the original to the area between the two important regions in ours. Our computation of visibility shows that both configurations produce the same coverage, meaning that these are both solutions to the same optimization problem. This confirms that the differences in our implementations are small and should not affect the overall results.

To test the differences in visibility models, it is unfair to use Monte Carlo simulations which use the same model as the optimization. As a result, we resort to virtual environment simulations by creating a virtual 3-D environment that mimics the actual 10 m $\times$ 10 m room used in Section VII-A. We then insert a random-walking humanoid wearing a red tag. The results are based on the visibility of the tag in two or more cameras. The cameras are set at the same height as the tag, and no mutual occlusion modeling is used. The optimization is performed with respect to a fixed number of cameras. To be fair to the scheme in [15], we run their optimization formulation to maximize the visibility from two cameras. The measurements of $\eta$ for the three schemes with the number of cameras varied from five to eight are shown in Table IV. Our proposed FIX_CAM performs the best followed by the uniform placement. The scheme in [15] does not perform well as it does not take into account the orientation of the tag. As such, the cameras do not compensate each other when the tag is in different orientations.



(a)                    (b)

Fig. 9.   Comparison between our simulation of [15] and the published results in [15]. (a) Placement in [15]. (b) Placement in our simulation of [15].
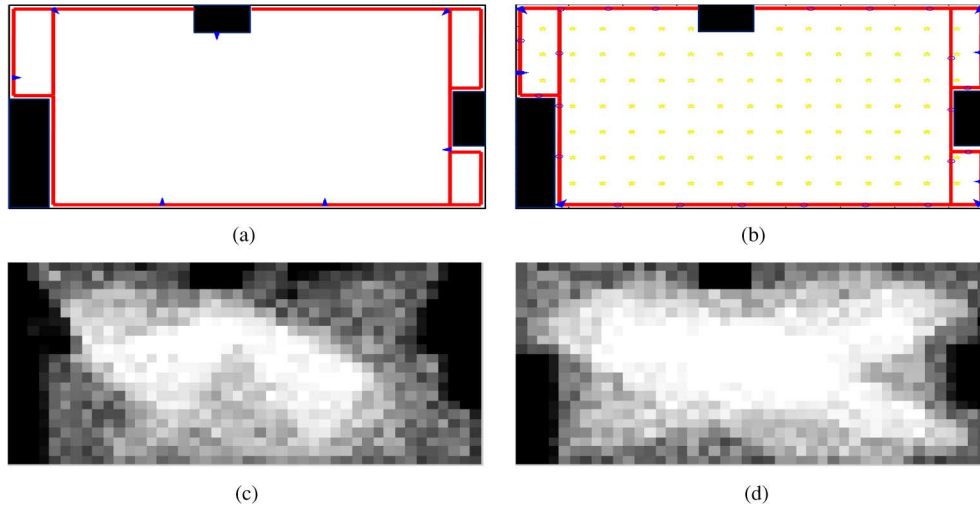
Fig. 10. Camera placement in our office environment. (a) Uniform placement, $\eta = 0.3801$. (b) Optimal placement, $\eta = 0.5325$. (c) Uniform placement: $\eta = 0.3801$. (d) Optimal placement: $\eta = 0.5325$.
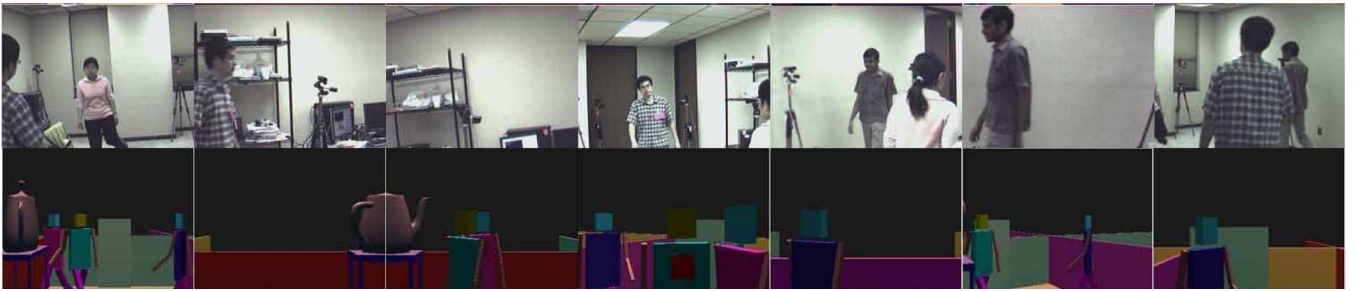


Fig. 11. Seven camera views from uniform camera placement.

TABLE IV
$\eta$ MEASUREMENTS AMONG THE THREE SCHEMES
USING VIRTUAL SIMULATIONS

| Number of cameras | FIX_CAM | [15] | Uniform Placement |
|---|---|---|---|
| 5 | $0.614 \pm 0.011$ | $0.352 \pm 0.010$ | $0.522 \pm 0.011$ |
| 6 | $0.720 \pm 0.009$ | $0.356 \pm 0.010$ | $0.612 \pm 0.011$ |
| 7 | $0.726 \pm 0.009$ | $0.500 \pm 0.011$ | $0.656 \pm 0.010$ |
| 8 | $0.766 \pm 0.008$ | $0.508 \pm 0.011$ | $0.700 \pm 0.009$ |

We are, however, surprised by how close uniform placement is to our optimal scheme. Thus, we further test the difference between the two with a real-life experiment that incorporates mutual occlusion. We conduct our real-life experiments indoor in a room of 7.6 meters long, 3.7 meters wide, and 2.5 meters high. There are two desks and a shelf along three of the four walls. Seven Unibrain Fire-i400 cameras at elevation of 1.5 meters with Tokina Varifocol TVR0614 lens are used. Since they are variable focal-length lens, we have set them at a focal length of 8 mm with a vertical field of view of 45° and horizontal field of view of 60°. As the elevation of the cameras is roughly level with the position of the tags, we have chosen a fairly large occlusion angle of $\beta_m = 65°$ in deriving our optimal placement. Monte Carlo results between the uniform placement and the optimal placement are shown in Fig. 10. For the virtual environment simulation, we replace the desks and the shelf with tables and teapot, insert three randomly walking humanoids, and capture 250 frames for measurement. For the real-life experiments, we capture about 2 min of video from the

seven cameras, again with three persons walking in the environment. Figs. 11 and 12 show the seven real-life and virtual camera views from both the uniform placement and optimal placement, respectively. As shown in Table V, the optimal camera placement is better than the uniform camera placement in all three evaluation approaches. The three measured $\eta$'s for the optimal placement are consistent. The results of the uniform placement have higher variation most likely due to the fact that an excessive amount of occlusion makes detection of color tags less reliable.

### C. Application of Tag Localization for Privacy Protection

In this section, we show the localization of visual tags in a multiple camera network using epipolar geometry. We also present results of using visual tagging in privacy protection as described in Section VI. Our experiments are based on a subset of three adjacent cameras from the camera network derived in Section VII-B. This limitation is due to our use of a planar checkerboard for camera calibration—cameras that are far apart cannot be calibrated together as they cannot see the checker pattern at the same time. This problem can be overcome by using other calibration objects such as a laser pointer as described in [33].

We first show the tag tracking performance using both a stationary checkerboard with 120 corners as well as the color tag used for the actual privacy experiment. For the checkerboard, we use a corner detector to extract all corners in the three camera

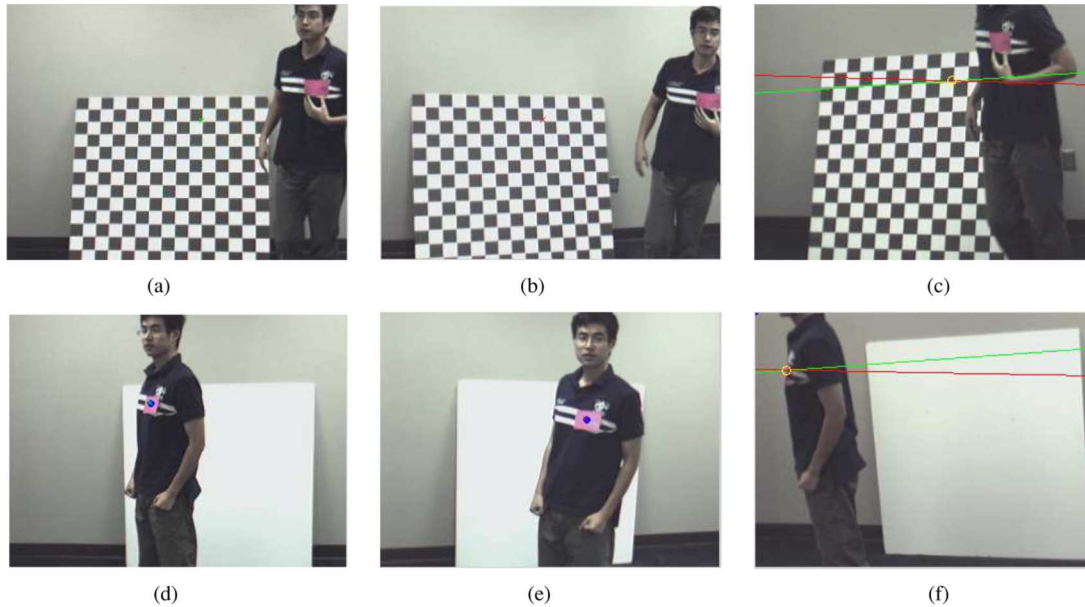Fig. 12.   Seven camera views from optimal camera placement.



Fig. 13.   (a) and (b) show the corresponding corners in the checkerboard in the original two camera views. After adjusting the image coordinates with the Sampson approximation, the green cross in (a) induces the green epipolar line in (c) and the red cross in (b) induces the red epipolar line. The intersection is very close to the actual corner in (c). (d) and (e) show the identified centroids of the tag. Each of the centroids induces an epipolar line in (f) and the intersection is the projected tag centroid in the third camera. (a) Corner detected in Cam1, (b) Corner detected in Cam2, (c) localization error, (d) Tag detected in Cam1, (e) tag detected in Cam2, (f) Epipolar line intersection.

TABLE V
$\eta$ MEASUREMENTS BETWEEN UNIFORM AND OPTIMAL CAMERA PLACEMENTS

| Camera Placement | Monte-Carlo Simulations | Virtual Simulation | Real-life Experiments |
|---|---|---|---|
| Uniform | 0.3801 | $0.4104 \pm 0.0153$ | $0.2335 \pm 0.0112$ |
| Optimal | 0.5325 | $0.5618 \pm 0.0156$ | $0.5617 \pm 0.0121$ |

TABLE VI
LOCALIZATION ERROR IN THE THIRD CAMERA VIEW

| Object Localized | Naive Intersection | Sampson Approximation |
|---|---|---|
| Corner | $1.0130 \pm 0.5669$ | $0.1898 \pm 0.1268$ |
| Color Tag | $23.842 \pm 6.4169$ | $4.5476 \pm 4.1244$ |

views. For the color tag, we adopt a GMM color detector and calculate the centroid in the image of the tag. Using the information from two camera views, we then infer the location of the corners in the third camera view using two variants of epipolar-line intersection. The "naive epipolar intersection" is simply the intersection of the two epipolar lines. The "Sampson approximation" uses a first-order approximation to select new image coordinates that are close to satisfying the epipolar constraint before computing the epipolar line intersection [34, Ch. 12]. The localization error, defined by the average distance in pixel between the actual corners in the third camera view and the corresponding projected corners, using the two approaches is shown in Table VI. We can conclude that the Sampson-approximated epipolar intersection provides results that are significantly better than the naive epipolar intersection. As for the

color tag experiment, we record 30 s of video from the three cameras which are synchronized using network timing protocol. The localization error using the two methods is also shown in Table VI. Although the errors are much larger this time, possibly due to the uncertainty introduced by centroid estimation and synchronization, the localization error reveals the same trend as in the corner detection and the tag is successfully located throughout the entire sequence. Fig. 13 shows the corner and tag localization on the third camera view.

The final experiment combines visual tagging with visual inpainting for privacy protection. Fig. 14 shows the pipeline of the process at three different camera views. The first image of each row shows the original video frame. Note that the tag is only visible in the top two but is largely occluded in the bottom one. As a result, our GMM color detector can only identify the centroid of
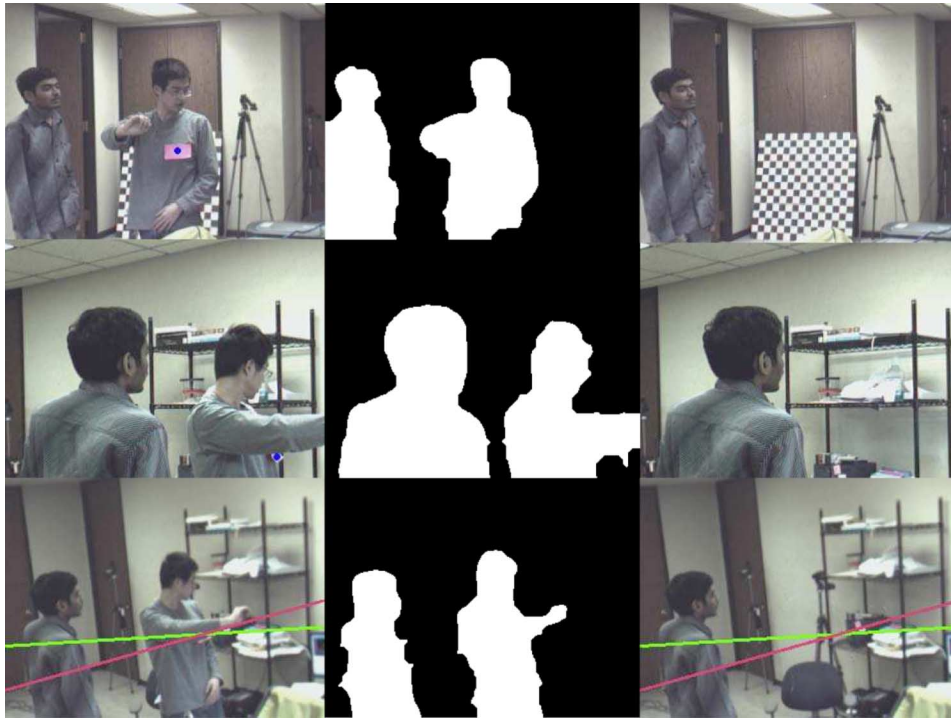
Fig. 14. Three rows of pictures show the privacy pipeline for three different camera views. The first column consists of the three views of the same scene with the tag detected only on the first two. Foreground blobs detected are shown in the second column and the inpainted frames shown in the last column. The bottom image uses epipolar geometry to infer the location of the occluded tag for the identification of the private individual.

the tag in the first two frames. Foreground moving blobs in each frame are shown in the middle column. To differentiate which blob to inpaint, the top two cameras identify the blob associated with the tag centroid to be the person whose privacy needed to be protected. The bottom camera receives the epipolar lines from the other two cameras and deduces the correct blob based on the inferred tag location. The inpainted results are shown in the last column of Fig. 14.

## VIII. Conclusions and Future Work

In this paper, we have proposed a framework in modeling, measuring, and optimizing placement of multiple cameras. Even though our framework is very general, our focus has been on triangulating visual objects for identifying different individuals in a privacy protected video surveillance network. By using a camera placement metric that captures both self- and mutual occlusion in 3-D environments, we have proposed two optimal camera placement strategies that complement each other using grid-based binary integer programming. To deal with the computational complexity of BIP, we have also developed a greedy strategy to approximate both of our optimization algorithms. Experimental results have been presented to verify our model and to show the effectiveness of our approaches. Equipped with an optimal camera placement, we have constructed a multiple-camera surveillance system capable of robustly identifying and obfuscating individuals for privacy protection.

There are many interesting issues in our proposed framework and visual tagging in general that deserve further investigation. Environmental factors such as prior knowledge of the movement of people, their interpersonal distances and configurations, as

well as the specifics of the back-end vision algorithms can and should be incorporated into the models to further improve the output camera placement. The incorporation of models for different visual sensors such as omnidirectional and PTZ cameras or even nonvisual sensors and other output devices such as projectors is certainly a very interesting topic. The optimality of our greedy approach can benefit from a detailed theoretical studies. The technical issues in combining wide-area calibration and visual tagging is also an important problem that we believe can be overcome in the very near future. Last but not the least, the use of visual tagging in other application domains such as immersive environments and surveillance visualization should be further explored.

## References

[1] J. Schiff, M. Meingast, D. Mulligan, S. Sastry, and K. Goldberg, "Respectful cameras: Detecting visual markers in real-time to address privacy concerns," in *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, 2007.

[2] J. O'Rourke, *Art Gallery Theorems and Algorithms*. Oxford, U.K.: Oxford Univ. Press, 1987.

[3] J. Urrutia, *Art Gallery and Illumination Problems*. Amsterdam, The Netherlands: Elsevier Science, 1997.

[4] T. Shermer, "Recent results in art galleries," *Proc. IEEE*, vol. 80, no. 9, pp. 1384–1399, Sep. 1992.

[5] V. Chvatal, "A combinatorial theorem in plane geometry," *J. Combinat. Theory Series B*, vol. 18, pp. 39–41, 1975.

[6] D. Lee and A. Lin, "Computational complexity of art gallery problems," *IEEE Trans. Inf. Theory*, vol. 32, pp. 276–282, 1986.

[7] D. Yang, J. Shin, A. Ercan, and L. Guibas, "Sensor tasking for occupancy reasoning in a camera network," in *Proc. IEEE/ICST 1st Workshop Broadband Advanced Sensor Networks (BASENETS)*, 2004.

[8] P.-P. Vazquez, M. Feixas, M. Sbert, and W. Heidrich, "Viewpoint selection using viewpoint entropy," in *Proc. Vision Modeling and Visualization Conf. (VMV01)*, Amsterdam, The Netherlands, IOS Press, 2001, pp. 273–280.

[9] J. Williams and W.-S. Lee, "Interactive virtual simulation for multiple camera placement," in *Proc. IEEE Int. Workshop on Haptic Audio Visual Environments and Their Applications*, 2006, pp. 124–129.

[10] S. Ram, K. R. Ramakrishnan, P. K. Atrey, V. K. Singh, and M. S. Kankanhalli, "A design methodology for selection and placement of sensors in multimedia surveillance systems," in *VSSN'06: Proc. 4th ACM Int. Workshop Video Surveillance and Sensor Networks*, New York, ACM Press, 2006, pp. 121–130.

[11] T. Bodor, A. Dremer, P. Schrater, and N. Papanikolopoulos, "Optimal camera placement for automated surveillance tasks," *J. Intell. Robot. Syst.*, vol. 50, pp. 257–295, Nov. 2007.

[12] A. Mittal and L. S. Davis, "A general method for sensor planning in multi-sensor systems: Extension to random occlusion," *Int. J. Comput. Vis.*, vol. 76, no. 1, pp. 31–52, 2008.

[13] A. Ercan, D. Yang, A. E. Gamal, and L. Guibas, "Optimal placement and selection of camera network nodes for target localization," in *Proc. IEEE Int. Conf. Distributed Computing in Sensor System*, 2006, vol. 4026, pp. 389–404.

[14] E. Dunn and G. Olaque, "Pareto optimal camera placement for automated visual inspection," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2005, pp. 3821–3826.

[15] E. Horster and R. Lienhart, "On the optimal placement of multiple visual sensors," in *VSSN'06: Proc. 4th ACM Int. Workshop Video Surveillance and Sensor Networks*, New York, ACM Press, 2006, pp. 111–120.

[16] K. Chakrabarty, S. Iyengar, H. Qi, and E. Cho, "Grid coverage of surveillance and target location in distributed sensor networks," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1448–1453, Dec. 2002.

[17] G. Foresti, C. Micheloni, L. Snidaro, P. Remagnino, and T. Ellis, "Active video-based surveillance systems: The low-level image and video processing techniques needed for implementation," *IEEE Signal Process. Mag.*, vol. 22, no. 2, pp. 25–37, 2005.

[18] W. Du, J.-B. Hayet, J. Piater, and J. Verly, "Collaborative multi-camera tracking of athletes in team sports," in *Proc. Workshop Computer Vision Based Analysis in Sport Environments*, 2006, pp. 2–13.

[19] K. Kim and L.-S. Davis, "Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering.," in *Proc. ECCV (3)*, 2006, pp. 98–109.

[20] C. Micheloni, G. Foresti, and L. Snidaro, "A network of cooperative cameras for visual-surveillance," *Vis., Image, Signal Process.*, vol. 152, no. 2, pp. 205–212, 2005.

[21] D. Chen, Y. Chang, R. Yan, and J. Yang, "Tools for protecting the privacy of specific individuals in video," *EURASIP J. Appl. Signal Process.*, vol. 2007, no. 1, pp. 107–107, 2007.

[22] E. N. Newton, L. Sweeney, and B. Main, "Preserving privacy by de-identifying face images," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 2, pp. 232–243, Feb. 2005.

[23] J. Wickramasuriya, M. Datt, S. Mehrotra, and N. Venkatasubramanian, "Privacy protecting data collection in media spaces," in *MULTIMEDIA'04: Proc. 12th Annu. ACM Int. Conf. Multimedia*, New York, ACM Press, 2004, pp. 48–55.

[24] S.-C. Cheung, J. Zhao, and M.-V. Venkatesh, "Efficient object-based video inpainting," in *Proc. IEEE Int. Conf. Image Processing (ICIP'06)*, 2006, pp. 705–708.

[25] W. Zhang, S.-C. Cheung, and M. Chen, "Hiding privacy information in video surveillance system," in *Proc. IEEE Int. Conf. Image Processing*, 2005, vol. 3, pp. II 868–II 871.

[26] J. Zhao and S.-C. Cheung, "Multi-camera surveillance with visual tagging and generic camera placement," in *Proc. ACM/IEEE Int. Conf. Distributed Smart Cameras, ICDSC07*, 2007.

[27] R. M. Karp, "Reducibility among combinatorial problems," *Complex. Comput. Computat.*, pp. 85–103, 1972.

[28] Introduction to lp_solve 5.5.0.10. [Online]. Available: http://lpsolve.sourceforge.net/5.5/.

[29] U. Feige, "A threshold of $\ln n$ for approximating set cover," *J. ACM*, vol. 45, no. 4, pp. 634–652, Jul. 1998.

[30] B. Quinn and K. Almeroth, Ip Multicast Applications: Challenges and Solutions, Tech. Rep., Sep. 2001, IETF RFC 3170.

[31] K. Agarway and P. Winston, Walke 1995. [Online]. Available: http://www.sgi.com/products/software/opengl/examples/glut.

[32] United States Census Bureau, Statistical Abstract of the United States: 1999 p. 155, Table 243.

[33] T. Svoboda, D. Martinec, and T. Pajdla, "A convenient multi-camera self-calibration for virtual environments," *PRESENCE: Teleoperat. Virtual Environ.*, vol. 14, no. 4, pp. 407–422, Aug. 2005.

[34] R. I. Hartley and P. Sturm, "Triangulation," *Comput. Vis. Understand.: CVIU*, vol. 68, no. 2, pp. 146–157, 1997.
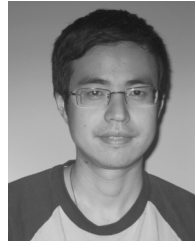
**Jian Zhao** (S'06) received the B.S. degree in electric engineering in Zhejiang University, Hangzhou, China, in 2005, the M.Sc. degree in electrical and computer engineering from the University of Kentucky, Lexington, in 2008, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

His research area includes vision network planning, computer vision, and image compression.

**Sen-ching (Samson) Cheung** (S'91–M'95–SM'07) received the B.S. degree (summa cum laude) from the University of Washington, Seattle, in 1992 and the Ph.D. degree from the University of California, Berkeley, in 2002.

He has been an Assistant Professor in the Department of Electrical and Computer Engineering of the University of Kentucky, Lexington, since 2004. Between 2002 and 2004, Samson was a computer scientist at the Sapphire Data Mining group in Lawrence Livermore National Laboratory. Between 1995 and 1998, he was a researcher at Compression Labs Inc. and later VTEL Corp. in San Jose, CA. He was also an active participant in the ITU-T H.263 (version 2) and ISO MPEG-4 standardization activities between 1995 and 1999.

Dr. Cheung was the winner of the Ralph E. Power Junior Faculty Award from Oak Ridge Associated Universities in 2005, and his work with the Sapphire Data Mining Group won the R&D 100 award in 2006. He has been a member of the Multimedia Systems and Applications Technical Committee of the IEEE Circuits and Systems Society since 2007. He has served in many technical program committees, including IEEE International Conference on Circuits and Systems (ISCAS), IEEE International Conference on Computer Vision (ICCV), IEEE International Conference on Multimedia and Expo (ICME), ACM Multimedia, and IEEE International Conference on Advanced Information Networking and Applications (AINA).

**Thinh Nguyen** (M'04) received the B.S. degree from the University of Washington, Seattle, in 1995 and the Ph.D. degree from the University of California, Berkeley in 2003.

He is an Assistant Professor at the School of Electrical Engineering and Computer Science of the Oregon State University, Corvallis. He has many years of experience working as an engineer for a variety of high-tech companies. He has served in many technical program committees. His research interests include multimedia networking and processing, wireless networks, and coding theory.

Dr. Nguyen is an associate editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS ON MULTIMEDIA, and the *Peer-to-Peer Networking and Applications*.