2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Assigning Cameras to Subjects in Video Surveillance Systems

Hazem El-Alfy David Jacobs Larry Davis
Department of Computer Science
University of Maryland at College Park
College Park, MD 20742
{helalfy,djacobs,lsd}@cs.umd.edu

Abstract— **We consider the problem of tracking multiple agents moving amongst obstacles, using multiple cameras. Given an environment with obstacles, and many people moving through it, we construct a separate narrow field of view video for as many people as possible, by stitching together video segments from multiple cameras over time. We employ a novel approach to assign cameras to people as a function of time, with camera switches when needed. The problem is modeled as a bipartite graph and the solution corresponds to a maximum matching. As people move, the solution is efficiently updated by computing an *augmenting path* rather than by solving for a new matching. This reduces computation time by an order of magnitude. In addition, solving for the shortest augmenting path minimizes the number of camera switches at each update. When not all people can be covered by the available cameras, we cluster as many people as possible into small groups, then assign cameras to groups using a minimum cost matching algorithm. We test our method using numerous runs from different simulators.**

I. Introduction

Surveillance systems are becoming common in modern facilities. The increasing need to provide safety has resulted in more demand for these systems. They typically consist of a network of cameras, monitored from a control room. The large number of cameras available and the rising cost of human resources make it prohibitively expensive to manually operate all of them. Recently, there has been a lot of interest in automating camera management in surveillance systems, to direct the operators' attention to "interesting" scenes.

A. Related Work

To cover large areas, cooperation schemes need to be developed amongst cameras. To maintain the visibility of as large an area as possible, several aspects of the problem have been addressed in the literature:

1) assigning camera locations;
2) controlling camera parameters, such as pan, tilt and zoom, and possibly position as well (for moving cameras) to track agents; and
3) switching tasks between cameras.

In scenes of known layout, such as lecture rooms [1], camera positions can be selected to focus on the lecturer and the audience, whose locations are known. In typical configurations, a wide-angle static camera is used to control the pan and tilt of an active camera that operates in high zoom mode. In surveillance applications, agents' activities are usually non-predictable. Wren et al. [2] control the pan, tilt and zoom of fixed high capability cameras using a network of low-cost sensors that measure activity.

Goradia et al. [3] present solutions to the problems of optimally deploying cameras and switching between them. They use dynamic programming to optimize the switching of multiple fixed cameras, tracking a single moving agent. Fiore et al. [4] present a system that reorganizes camera placement with scene changes, while Zhao and Cheung [5], in addition to other contributions, develop a method to determine both the optimal number and positions of cameras that achieve a desired level of visibility.

In the area of tracking agents across multiple cameras, Takemura and Miura [6] plan the panning and tilting of multiple fixed cameras to track as many moving agents as possible, in a room without obstacles. They use a motion model to predict people's future states. Collins et al. [7] associate cost functions with cameras to track a single moving object across multiple fixed cameras in an outdoor scene. Switching of cameras can occur to "hand-off" to another camera when an agent has moved to within its field of regard. It may occur between a wide-angle view camera and a highly zoomed-in one, to provide a better view of an agent. When gaps exist between cameras, Makris et al. [8] suggest a method to correspond the tracks of moving agents transiting through the "blind" regions of the camera network. An indoors surveillance system with multiple non-overlapping field-of-view cameras is presented in [9]. In this work, Porikli and Divakaran match objects between cameras to generate a key-frame-based video summary for each object during its presence within the system.

Another way to cover a large area with surveillance cameras is to use mobile cameras. Bandyopadhyay et al. [10] present a strategy to track a single target, moving amongst obstacles, using a single moving camera fixed on a robot. Tang and Özgüner [11] add several restrictions to that problem such as a limited number of robots with restricted fields of view, versus a larger number of moving targets. In this case, the problem becomes that of minimizing the average duration each target is not observed.

The problem of optimal sensor layout for visibility is classically known as the art gallery problem [12]. There are also a number of related problems that have drawn a lot of

The research described in this paper was supported in part by the US Government under the VACE program.

attention. For example, recently, Kloder et al. [13] solve the problem of minimizing the probability of undetected intrusion, in environments with obstacles. Kolling and Carpin [14] present an algorithm to find the minimum number of robots needed to detect possible intrusions, in indoor environments with many rooms connected by multiple doors. When the intruding targets are "clever", the problem becomes that of the pursuit-evasion game. Some passive results are to determine bounds on the numbers of targets hiding behind obstacles, based on observations by moving cameras [15]. The harder active problem is to calculate a pursuer path that keeps the evader in sight [16]. Interestingly, the inverse problem has also been attempted. Marinakis et al. [17] infer the relative positions of a set of sensors by observing the motions of objects between their non-overlapping fields of view.

B. Contributions

We consider an environment with multiple fixed cameras and multiple agents moving amongst fixed obstacles. We assume that people can be tracked using, for example, a wide field of view camera fixed to the ceiling. The cameras are controlled to acquire high resolution video of individual people or small groups. We assume that the cameras can pan and tilt fast enough so that latencies due to camera movement can be ignored. This is a fairly reasonable assumption with state-of-the-art pan speeds reaching up to 300° per second [18]. The problem is thus to assign cameras to people (or small groups) and select where to point each camera to produce the individual videos. Using our knowledge of people's positions, we employ bipartite matching to assign cameras to people over time to create a zoomed-in video, with as few camera switches as possible.

The contributions of the research presented in this paper are as follows:

1) We assign a "surveillance monitor" per subject (or small group), versus a monitor per camera. We solve the camera assignment problem in scenes with multiple subjects and multiple cameras with overlapping fields of regard.
2) We combine results from computer vision, computational geometry and algorithms to compute a continuously updated camera assignment that maximizes the total time subjects are imaged.

It is not always possible to assign a camera to every individual person; for example, there can be more people than cameras in the surveyed scene. Even if this is not the case, people might cluster in the field of regard of a single camera. We resolve this issue by controlling the focal length of the camera to capture a group of neighboring people with a single camera, while still having a close-up view of each person in the group. This is implemented by grouping people, then assigning cameras to groups using weighted bipartite matching. We show in our results the effect of this approach on the number of covered people.

The rest of the paper proceeds as follows: Section II defines the problem in more detail. Our approach to solving the problem is detailed in the ensuing three sections. Section III presents our modeling of the one person per camera problem as a bipartite graph matching algorithm, and section IV discusses the approach for the many people per camera situation. Section V tests and evaluates the method with multiple runs using two different simulators. Finally, we conclude in section VI.

II. Problem Definition

The first problem we solve is the following. A surveillance camera network with n_c cameras is set up in an environment containing obstacles. A group of n_p people walk freely in the room. Our goal is to construct a video for each person, generally using multiple cameras over time, that captures each person for as long as possible during their presence in the environment. This results in one (multi-camera) video per person rather than one video per camera being displayed to security operators, or further analyzed using computer vision methods.

The cameras are fixed but can pan, tilt and zoom (PTZ cameras). We call the volume of the environment that a camera can capture, for a specific PTZ setting, the *field of view* (FOV) of that camera. We define the *field of regard* (FOR) of a camera as the union of its FOV's for all possible PTZ settings reachable in negligible time. We assume that cameras have a large field of regard, but that a small field of view is generally required to image a person at the required spatial resolution.

When there are fewer cameras than people, the objective becomes covering as many people as possible. We accomplish this by assigning a camera simultaneously to several people, with the constraint of having a zoomed-in video that shows the covered people with high resolution.

Each camera is represented by its *visibility polygon*, the portion of the floor of the environment that lies inside its FOR. This enables us to solve the camera assignment problem in the two-dimensional space as shown in figure 1. By the earlier definition of a FOR, any person inside a specific camera's visibility polygon is viewable by that camera in a negligible time, using pan, tilt and zoom operations. The visibility computation algorithm we employ works for convex polygonal obstacles on the ground plane. Any non-convex obstacle is thus first decomposed into convex parts.

III. Bipartite Matching

A. Graph Modeling

We first compute the visibility polygons of cameras using a simple angular plane sweep algorithm, as the one presented by O'Rourke in [12]. We assume the locations of people $(x_i(t), y_i(t)), 1 \leq i \leq n_p$ are found, for example, using a wide field of view camera fixed to the ceiling. At each time instant t, we use the point-in-polygon test to determine the visibility polygons in which each person is located; in other words, the cameras that can view each person. This process is linear in the number of edges of all polygons. It is shown in [12] that the number of edges of a visibility polygon is linearly proportional to the number of edges of

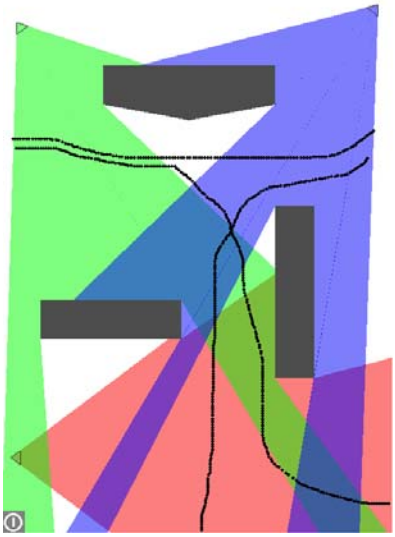

Fig. 1. Problem setup in 2D. People's trajectories are dotted and visibility polygons are in blended colors. The obstacles are shaded in black.

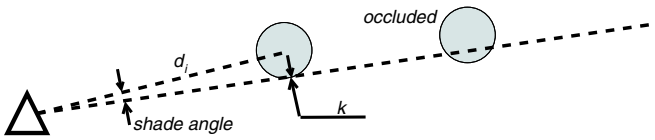

Fig. 2. Computing occlusions of subjects by one another.

obstacles (polygon holes), i.e. $O(n)$. This means that the process of determining the cameras that can view a person at any instant can be conducted in $O(n \cdot n_c)$ time. This process is done online, hence must be performed efficiently. When the number of people n_p and the number of cameras n_c are small, this can be computed in $O(n \cdot n_c \cdot n_p)$ for all people, at each time instant. As n_p grows, a more efficient approach is to divide the plane into cells that directly index visibility polygons, and build a two-dimensional data structure that efficiently assigns people to cells. An offline preprocessing stage will determine which cameras can view each cell.

We also suggest a 2D model for the occlusion of people by one another with respect to each camera. To compute the occlusion, we sort the subjects within each visibility polygon w.r.t. the angles they make with their respective cameras. Then, for each subject, we find its neighbors that are located farther from the camera, and mark them as occluded. Given a camera, a subject S_1 occludes a subject S_2 if S_1 is closer to that camera than S_2, and S_2 falls within the "shade" of S_1. The shade angle of a subject i is computed using the equation: $\alpha_i = \tan^{-1} \frac{k}{\sqrt{d_i^2 - k^2}}$ as shown in figure 2.

Finally, we model the camera assignment problem as matching in a bipartite graph $G = ((V,U),E)$ as shown in figure 3. The vertices V represent the people and U the set of cameras. An edge $e_{ij} \in E$, between v_i and u_j, exists if camera c_j can view person p_i, that is p_i falls inside the visibility polygon of c_j and is not occluded by another person. The problem is to match as many vertices

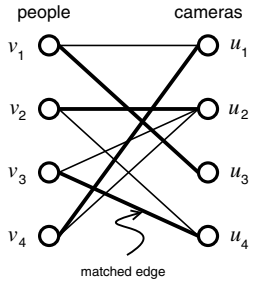

Fig. 3. An example of the problem model with a perfect matching.

as possible from V (people) to vertices in U (cameras), with the constraint that each camera is matched to at most one person. This is the classical bipartite matching problem [19].

B. Initial Matching

Once the bipartite graph is constructed, we compute an initial matching. Figure 3 shows an example of a *perfect* matching, where all vertices are matched. This is not always possible, and hence, we seek a *maximum cardinality* matching, in which we try to match as many vertices as possible. Specifically, we are interested in matching as many vertices of V (people) as possible. This proceeds by visiting each unmatched v-node, then using a modified breadth-first-search (BFS) algorithm to reach the closest unmatched u-node, and finally matching both. The BFS algorithm is modified so as to restrict visited edges to be sequences alternating between matched and non-matched edges. These sequences are known as *alternating paths*. When no more alternating paths can be found in the graph G, a maximum matching is found. It is proved (see [20] for example) that a standard matching algorithm (Edmonds' [19]) correctly solves the matching problem for a bipartite graph in $O(\min(|V|,|U|) \cdot |E|)$. In practice, $n_c \leq n_p$, so the initial matching complexity is $O(n_c^2 \cdot n_p)$.

C. Matching Update

As people move through the room, their positions within the fields of regard of cameras are updated. In the event that a person enters a visibility polygon, exits it, becomes occluded or becomes visible, the change results in updating the bipartite graph accordingly, and possibly recomputing the matching. There are four cases here, depending on the type of graph update that has occurred:

- If no edges included in the matching, called *matched edges*, are removed, nothing has to be done.
- If one or more matched edges are removed, we attempt to *augment* the matching.
- If one or more edges are added to an incomplete matching, we attempt to augment it.
- If one or more edges are added to a complete matching, nothing has to be done.

The matching algorithm is incremental by nature. The cost of each stage is $O(|E|) \simeq O(n_p \cdot n_c)$. This property reduces significantly the running time of subsequent matching computations, since practically, very few people change visibility

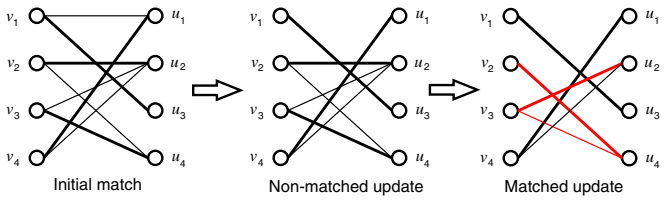

Fig. 4. A bipartite graph update that involves the two cases with edge removal. Matched edges are thickened. In the first update, a non-matched edge v_1u_1 is removed so nothing needs to be done. In the second time, the matched edge v_2u_2 is removed. The *alternating path* $v_2u_4v_3u_2$ (red) matches v_2u_4, unmatches v_3u_4 and matches v_3u_2.

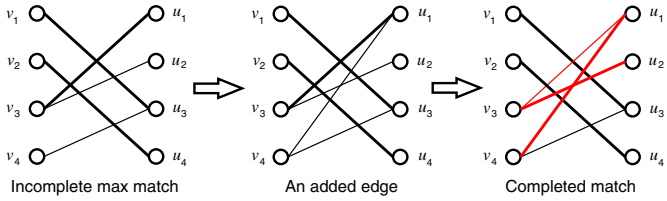

Fig. 5. A bipartite graph update that involves the addition of an edge to an incomplete matching. Matched edges are thickened. Initially, the maximum possible matching is incomplete. Later, an edge v_4u_1 is added. This permits completing the matching using the alternating path $v_4u_1v_3u_2$

regions at any time, as will be shown in the results section. Depending on the type of graph update that occurs, the search for an alternative matching proceeds as follows:

1) Determine a non-matched vertex $v \in V$ (person).
2) Span a tree rooted at v using modified BFS.
3) If non-matched leaf nodes in U (cameras) are reached, choose the closest one (u) to the tree root. The alternating path from v to u, forms the new *augmented* matching.
4) If no leaf nodes exist which are non-matched vertices in U, we conclude that v cannot be matched.

At each time, we only need to run this procedure for unmatched vertices. This represents an order of magnitude time savings over solving the matching problem for the entire graph at each update. In addition, matching the tree root to the *shallowest* non-matched leaf guarantees that we obtain the minimum number of re-assignments at each step. Figure 4 illustrates the two matching update cases that involve edge removal, and figure 5 shows how an incomplete matching is augmented when an edge is added.

IV. Minimum Cost Bipartite Matching

The previously described approach works efficiently when the number of people n_p and the number of cameras n_c are comparable. However, a non-uniform distribution of people over cameras can result in many people being uncovered while several cameras are unassigned. This can also happen when $n_p \gg n_c$. In the following, we update our algorithm to deal with these cases.

A. People Grouping and Graph Modeling

The first step is to cover as many people as possible with a small number of groups, where each group is represented

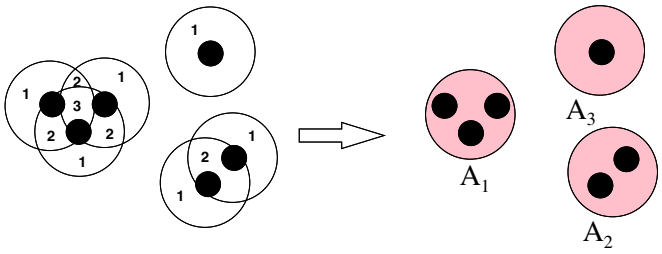

Fig. 6. The approach used to cluster people. The cardinalities of the intersections are marked to the left and the actual clusters to the right.

by a disk on the ground plane that covers the group. We constrain the groups (i.e. disks) to be of a small fixed size, since covering a large group would generally require a significant zoom-out, which would result in a video of low resolution. Such clustering problems are known to be NP-hard, see, for example, [21]. Several variants of the problem we address have been previously studied. In particular, Gonzalez [22] and Agarwal [21] suggest an $O(n^{\sqrt{k}})$-algorithm for the very similar problem of covering n points with fixed size squares, where k is the number of required clusters. This is impractical in most situations, such as ours. The authors also develop an approximate (not optimal) algorithm, that runs in $O(n \log k)$.

We employ a greedy approximate algorithm, that starts with the group (disk) that covers the maximum number of people, then the group with the next highest cardinality, and so on. This is done as follows (see figure 6):

1) Center a disk over each person. This represents the locus of centers of disks (groups) that can cover that person.
2) The plane is discretized using a grid, with its cells' values incremented by one for each disk that overlaps them.
3) Considering the disks in arbitrary order, if the next disk overlaps a previous one, the cardinalities of the grid cells forming the intersection region are incremented and the cells are inserted into a priority queue.
4) Dequeue a grid cell; it has maximum cardinality, say K.
5) Delete the K nearest subjects, assign them to a group, then undo the effects of steps 1) and 2).
6) Repeat 4) and 5) until only the grid cells with cardinalities 1 remain.

With n_p people and k clusters, this method requires $O(n_p)$ insertions in the priority queue, hence a complexity of $O(n_p \log k)$. Once the clusters are formed, we model the problem as previously using a bipartite graph $G = ((V, U), E)$, with V here representing *groups* of people $\mathcal{A}_i$ (possibly consisting of a single person) and U still representing the set of cameras. An edge $e_{ij} \in E$, between v_i and u_j, exists if camera c_j can view the group $\mathcal{A}_i$.

A special case exists when a group falls on the edge of a camera's visibility region or when some of its people are occluded by others. In that case, no single camera can view

all the people of that group, which defeats the purpose that grouping was originally developed for. We identify when this happens, then, using a variant of the grouping algorithm, the formed groups are split, into the largest subgroup that can be viewed entirely by a single camera, then the next largest, and so on. Our experiments show that slight improvements in the coverage rate of the algorithm is only achieved when the number of subjects is close to the number of cameras. For the rest of this paper, we do not use the splitting variant of the grouping algorithm.

B. Edge Weight Function

In order to view as many people as possible, priority is given to larger groups. This is achieved by adding weights to the graph edges. The *weighted bipartite matching* problem is that of finding a matching of G with the largest possible sum of edge weights [20], or the minimum sum of costs. The problem was initially solved by Kuhn [23] in $O((|V|+|U|)^3)$ time. Recently, an approximate algorithm that is an order of magnitude faster has been suggested by Schwartz for edge weights that satisfy certain properties [24].

To favor larger groups, we choose a cost function that assigns lower costs to edges associated with larger groups and find a match with minimal cost. A typical edge cost function is: $c_{ij} = W_{MAX} - w_{ij}$, where w_{ij} is the number of people in group i visible to camera j and W_{MAX} is larger than any w_{ij}. We implement a variant of Kuhn's algorithm that is proven to find the optimal solution. The algorithm proceeds by creating an auxiliary non-weighted graph which contains the minimum cost edges incident on each node. Then, a maximum cardinality (non-weighted) matching, similar to figure 3, is computed. If it is not complete, the next unused minimum cost edge is added and an augmented matching, similar to figure 5, is constructed. The process is repeated until all vertices of V (people) have been matched, or no more edges remain. This method works only with complete graphs that have $|V| = |U|$. This is achieved by completing the missing edges with ones with costs W_{MAX}, so they are not chosen during the optimization.

The use of a cost function that favors large groups of people results in small groups and individual people to be unmatched to any camera for extended periods of time. To balance the amounts of time each subject is covered, we add a time penalty to the edge cost function. The updated cost function is:

$$c_{ij} = W_{MAX} - w_{ij} - f(T_i) \tag{1}$$

where, for each person, a variable T_i accumulates the amount of time that person has not been assigned to any camera. For groups, we find the *minimum* positive T_i of its members. Our experiments have shown that using the *maximum* T_i in groups increases camera switches which produces a non-smooth video. At given time thresholds, the cost of an edge changes due to the time penalty, and the cameras are forced to switch. As a camera assignment changes, the variables T_i also change, which might reverse a previous assignment immediately, causing an infinite oscillation of cameras between two subjects. The function $f(\cdot)$ in equation (1) above is just used to prevent this from occurring, keeping the camera switches smooth. Similarly, other factors can also be included into the cost function, such as the orientation of a person w.r.t. camera, where a person facing a camera will have preference (lower cost) over a person facing away from it.

C. Matching Update

The changes occurring in groups, as people move through the environment, are more involved than in the one-to-one case presented earlier. We present below how these changes affect the graph topology, and how we efficiently update the solution.

1) Change in weights only, same graph topology: If a matching edge cost increases or a non-matching edge cost decreases, *augmenting* the solution is attempted. First, the match of the incident node is removed. Then, a tree rooted at this node is spanned using the modified BFS as previously. The branches of this tree are the potential alternating pathes. Along each path, the cost of an undone matching edge is subtracted, while the cost of a newly added matching edge is added. If a negative path is found, it means that we can find a smaller cost matching. The smallest cost branch is then used to augment the solution. If no negative paths are found, no better solution can be obtained.

If a matching edge cost decreases, nothing needs to be done. The intuition here is that the cost was already minimal using that edge. Now, it's even less. Similarly, if a non-matching edge cost increases, no new solution needs to be searched for. The current minimum cost solution did not use that edge and so obviously a new solution won't, when its cost has increased.

2) Change in graph topology: If an edge is added, this is equivalent to a cost decrease (from ∞ to c) of a non-matching edge. If an edge is removed, this is equivalent to an edge cost increase (from c to ∞): if it were a non-matching edge nothing has to be done, while if it were matching, an augmented solution is constructed. Finally, adding/removing a node to/from the graph (e.g., group split/merge) amounts to adding/removing at most n_c edges.

3) Time requirements: Since the solution is augmented here also using a modified BFS, the time complexity of each update is similarly $O(|E|) \simeq O(n_p \cdot n_c)$. When we have many more people than cameras ($n_p \gg n_c$), n_c can be considered a constant, and the update process is linear in the number of people n_p. By adding up the complexities of all the components of our algorithm, we can see that the bottleneck is in grouping people $O(n_p \log k)$. This represents the overall asymptotic complexity of the algorithm, if grouping is required. If not required, the overall complexity is linear in n_p.

V. Implementation and Results

To evaluate our method under different conditions, we constructed a simulator and, in addition, used the simulator developed by van den Berg et al. [25]. In our simulator,

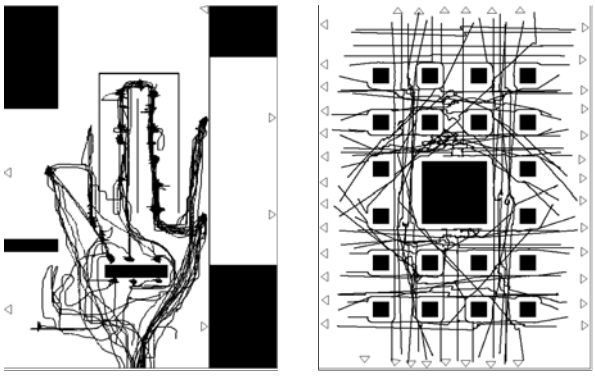

Fig. 7. Two example setups used in testing our method. Left: inside a bank, generated using our simulator (overlapping field-of-regard cameras); Right: a grid of obstacles, generated using van den Berg's simulator. The linear segments are people's trajectories.

obstacles' locations and people's trajectories and velocities are manually traced. This enables more realistic agents' trajectories. We have simulated two indoor surveillance scenarios: overlapping and non-overlapping fields-of-regard cameras. The scenarios correspond, loosely, to an idealized bank and an office interior, respectively, with about 25 people surveyed by up to 8 cameras. On the other hand, van den Berg's simulator easily generates larger numbers of trajectories. Using it, we generated up to 60 people with 40 surveillance cameras, moving among several setups of obstacles. This results in graphs with over 2000 edges. An example setup from each simulator is shown in figure 7.

Under each of the tested scenarios, we perform multiple runs, varying the number of cameras and people. For each subject, we compute its *coverage*, i.e. the fraction of frames that it was assigned to a camera during its presence in the scene. Then, we compute the mean and variance of the coverage, weighted by the subject's trajectory length.

Figure 8 shows the mean of the coverage for the two setups of figure 7, under different grouping strategies, as the number of people and cameras vary. Group diameters are measured in units of people's average sizes. In the bank environment (top), increasing the group size has a significant effect on coverage. The large amount of open space enables this to happen. However, in the obstacles grid environment (bottom), coverage increases with group size at a slower rate, because the narrow "hallways" that exist between obstacles do not allow the formation of large groups.

Adding a time penalty to the edge cost function, as mentioned in equation (1), distributes the coverage more evenly amongst the people who haven't been tracked in a while. We have experimented with different group sizes, in multiple environments and with different problem sizes (number of people, cameras). As expected, adding the time penalty to the cost function reduces the standard deviation of the coverage, due to a more even distribution. The standard deviation is further reduced when there is a much greater number of people than cameras. This is because in these situations, and in the absence of a time penalty, the cameras tend to remain assigned to the same subset of people to

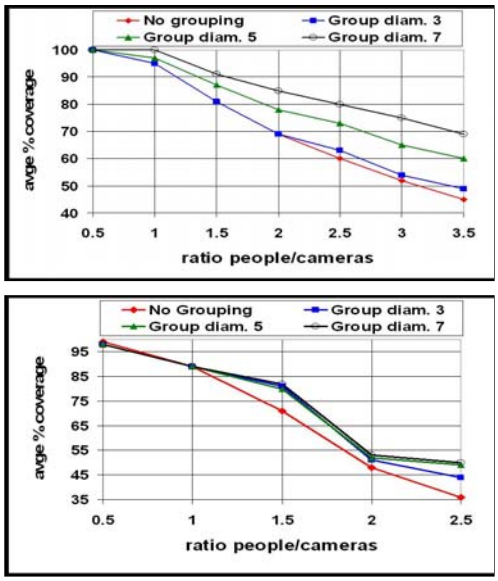

Fig. 8. Mean coverage as the group size (diameter) varies. Top: bank scenario from our simulator. Bottom: regular grid of obstacles from van den Berg's simulator.

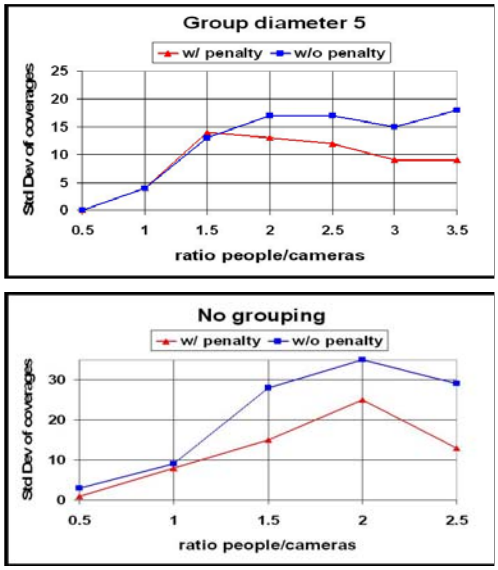

Fig. 9. Effect of adding a time penalty on the standard deviation ($= \sqrt{var}$) of the coverage. Top: bank scenario (our simulator) with a group size equal to 5 people. Bottom: regular grid of obstacles (van den Berg's simulator) without grouping.

reduce camera switches, rather than rotating to different people. Sample results for the two example setups are shown in figure 9.

We implemented the algorithms in C, on a Pentium D 2.8GHz processor with 1GB of memory. Computing the bipartite matching for multiple graph sizes is done in real time, the timings of which are shown in figure 10. These are much lower than the asymptotic bounds developed in sections III and IV. It shows that in practice, people stay for some time before changing visibility regions, which reduces the number of graph/matching updates.

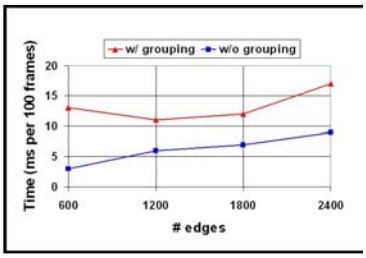

Fig. 10. Matching times for various graph sizes (van den Berg's simulator).

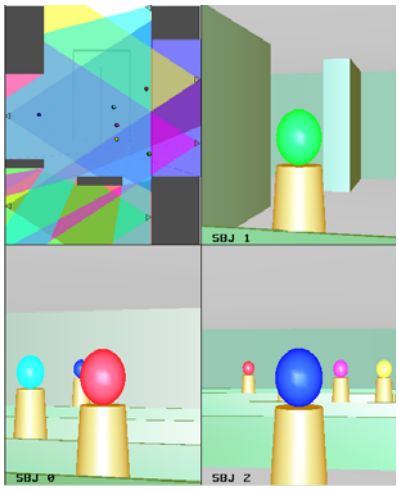

Fig. 11. Snapshot of our simulator: 2D top view with visibility regions along with 3D zoomed-in videos for the first 3 subjects.

VI. Conclusion

We have presented a novel approach to the problem of assigning multiple people moving amongst obstacles to multiple cameras, using the bipartite matching algorithm. We show how our method benefits from the incremental nature of the algorithm to update the matching of cameras to people online in linear time. When the number of cameras is comparable to the number of people in the surveyed scene, we have applied our method to construct individual zoomed-in videos for each person, as shown in figure 11. When we have many more people than cameras, we cluster people and attempt to cover as many as possible, without compromising their close-up view. Simulation results enable us to validate our method.

References

[1] Y. Rui, L. He, A. Gupta, and Q. Liu, "Building an intelligent camera management system," in *Proceedings of the Ninth ACM International Conference on Multimedia (ACM MM'01)*, Ottawa, Canada, Sept. 2001, pp. 2–11.

[2] C. R. Wren, U. M. Erdem, and A. J. Azarbayejani, "Automatic pan-tilt-zoom calibration in the presence of hybrid sensor networks," in *Proceedings of the Third ACM International Workshop on Video Surveillance & Sensor Networks (VSSN'05)*, Singapore, Nov. 2005, pp. 113–120.

[3] A. Goradia, Z. Cen, C. Haffner, N. Xi, and M. Mutka, "Switched video feedback for sensor deployment and target tracking in a surveillance network," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'07)*, Roma, Italy, Apr. 2007, pp. 3470–3475.

[4] L. Fiore, G. Somasundaram, A. Drenner, and N. Papanikolopoulos, "Optimal camera placement with adaptation to dynamic scenes," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)*, Pasadena, California, May 2008, pp. 956–961.

[5] J. Zhao and S. ching S. Cheung, "Multi-camera surveillance with visual tagging and generic camera placement," in *First ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC '07)*, Vienna, Austria, 2007, pp. 259–266.

[6] N. Takemura and J. Miura, "View planning of multiple active cameras for wide area surveillance," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'07)*, Roma, Italy, Apr. 2007, pp. 3173–3179.

[7] R. T. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1456–1477, Oct. 2001.

[8] D. Makris, T. Ellis, and J. Black, "Bridging the gaps between cameras," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, Washington, DC, 2004, pp. 205–210.

[9] F. Porikli and A. Divakaran, "Multi-camera calibration, object tracking and query generation," in *Proc. International Conference on Multimedia and Expo (ICME'03)*, Baltimore, MD, USA, July 2003, pp. 653–656.

[10] T. Bandyopadhyay, Y. Li, M. H. A. Jr, and D. Hsu, "A greedy strategy for tracking a locally predictable target among obstacles," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'06)*, Orlando, Florida, May 2006, pp. 2342–2347.

[11] Z. Tang and U. Özgüner, "Motion planning for multitarget surveillance with mobile sensor agents," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 898–908, Oct. 2005.

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

[13] S. Kloder and S. Hutchinson, "Partial barrier coverage: Using game theory to optimize probability of undetected intrusion in polygonal environments," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)*, Pasadena, California, May 2008, pp. 2671–2676.

[14] A. Kolling and S. Carpin, "Multi-robot surveillance: an improved algorithm for the GRAPH-CLEAR problem," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)*, Pasadena, California, May 2008, pp. 2360–2365.

[15] J. Yu and S. M. LaValle, "Tracking hidden agents through shadow information spaces," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)*, Pasadena, California, May 2008, pp. 2331–2338.

[16] R. Murrieta-Cid, R. Monroy, S. Hutchinson, and J.-P. Laumond, "A complexity result for the pursuit-evasion game of maintaining visibility of a moving evader," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'08)*, Pasadena, California, May 2008, pp. 2657–2664.

[17] D. Marinakis, G. Dudek, and D. J. Fleet, "Learning sensor network topology through monte carlo expectation maximization," in *Proc. IEEE International Conference on Robotics and Automation (ICRA'05)*, Barcelona, Spain, Apr. 2005, pp. 4581–4587.

[18] Toshiba America Information Systems, Inc. (2007) Toshiba IP Network PTZ Camera data sheet, IK-WB21A. [Online]. Available: www.toshibasecurity.com

[19] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of Mathematics*, vol. 17, pp. 449–467, 1965.

[20] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, N.J.: Prentice Hall, 1982.

[21] P. K. Agarwal and C. M. Procopiuc, "Exact and approximation algorithms for clustering," *Algorithmica*, vol. 33, no. 2, pp. 201–226, Dec. 2002.

[22] T. F. Gonzalez, "Covering a set of points in multidimensional space," *Information Processing Letters*, vol. 40, no. 4, pp. 181–188, Nov. 1991.

[23] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[24] J. Schwartz, A. Steger, and A. Weißl, "Covering a set of points in multidimensional space," *Experimental and Efficient Algorithms*, vol. 3503, pp. 476–487, May 2005.

[25] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin, "Interactive navigation of multiple agents in crowded environments," in *Proceedings of the 2008 symposium on interactive 3D graphics and games (I3D '08)*, Redwood City, CA, Feb. 2008, pp. 139–147.