

# Task Scheduling in Large Camera Networks

Ser-Nam Lim<sup>\*</sup>, Larry Davis, and Anurag Mittal

<sup>1</sup> Cognex Corp., Natick, MA, USA

<sup>2</sup> CS Dept., University of Maryland, College Park, Maryland, USA

<sup>3</sup> CSE Dept., IIT, Madras, India

**Abstract.** Camera networks are increasingly being deployed for security. In most of these camera networks, video sequences are captured, transmitted and archived continuously from all cameras, creating enormous stress on available transmission bandwidth, storage space and computing facilities. We describe an intelligent control system for scheduling Pan-Tilt-Zoom cameras to capture video only when task-specific requirements can be satisfied. These videos are collected in *real time* during predicted temporal “windows of opportunity”. We present a scalable algorithm that constructs schedules in which multiple tasks can possibly be satisfied simultaneously by a given camera. We describe two scheduling algorithms: a greedy algorithm and another based on Dynamic Programming (DP). We analyze their approximation factors and present simulations that show that the DP method is advantageous for large camera networks in terms of task coverage. Results from a prototype real time active camera system however reveal that the greedy algorithm performs faster than the DP algorithm, making it more suitable for a real time system. The prototype system, built using existing low-level vision algorithms, also illustrates the applicability of our algorithms.

## 1 Introduction

Large scale camera network systems are being increasingly deployed for purposes that include security, traffic monitoring, etc. These systems typically consist of a large number of cameras, which can either be active (specifically, Pan-Tilt-Zoom or PTZ cameras) or static, transmitting in real time video streams to processing and/or storage systems. Our interest is in controlling these cameras to acquire video segments that satisfy task-specific constraints. For example, one may wish to acquire at least a few images of each person who enters a given region, capture video segments lasting  $k$  seconds and containing well-magnified facial images for facial recognition, or, capture  $k$  second long video segments of the side view of a person for gait modeling and recognition. By intelligently transmitting and storing only video segments satisfying task requirements, we can reduce the bandwidth requirements and storage space significantly and increase the efficiency and effectiveness with which the collected video segments can be processed.

The control of the cameras to collect these video segments is a challenging problem. The system must detect and track moving objects both within and between cameras in a *sensing stage*, a problem which is not fully solved yet. Papers such as [1–3] deal with tracking under occlusions, and other papers such as [4, 5] describe algorithms for tracking across non-overlapping views.

A second challenge is to predict, given a set of tracked targets, the time intervals during which video segments meeting the requirements of the tasks can be collected from available cameras. These requirements include ensuring (1) that the associated

---

<sup>\*</sup> This research was funded in part by the U.S. Government VACE program

object is unobstructed by other objects, (2) that it is moving in a direction suitable for the task, (3) that it can be captured in a field of view of the camera (PTZ) assigned to collect its video segments, and (4) that the collected video segments must satisfy task-specific minimum resolution and duration. For example, if the task is to collect facial images, then we must ensure that the video segments are collected only during time intervals when the person is predicted to be walking towards the camera and unobstructed by other moving objects. This can be done using the observed tracks of the person and other moving objects, *predicting their trajectories into the future*, and then identifying periods of crossings between the predicted trajectories with respect to each of the available cameras. The complements of these periods of crossings are visibility time intervals during which the person is unobstructed, and camera settings can be determined within these temporal visibility predictions to capture the person in a well-magnified frontal image or video sequence satisfying the four requirements above. This problem has been addressed in earlier work. [6] described the construction of so-called “Task Visibility Intervals” (TVIs) and “Multiple Task Visibility Intervals” (MTVIs), that represent time-varying camera setting ranges that can be used to collect video segments satisfying one (TVI) or multiple tasks simultaneously (MTVIs).

A TVI is a 6-tuple:

$$(c, (T, o), [r, d], Valid_{\psi, \phi, f}(t)), \quad (1)$$

where  $c$  represents a PTZ camera,  $(T, o)$  is a (task, object) pair -  $T$  is the index of a task to be accomplished and  $o$  is the index of the object to which the task is to be applied, and  $[r, d]$  is a future time interval during which task requirements can be satisfied using camera  $c$ . Then, for each time instance  $t \in [r, d]$ ,  $Valid_{\psi, \phi, f}(t)$  is the range of valid combinations of the pan angle ( $\psi$ ), tilt angle ( $\phi$ ) and focal length ( $f$ ) settings that camera  $c$  can employ to capture object  $o$  at time  $t$ . The tasks themselves are 3-tuples:

$$(p, \alpha, \beta), \quad (2)$$

where  $p$  is the required duration of the task,  $\alpha$  is the orientation of the object relative to the optical axis of the camera used to accomplish the task, and  $\beta$  is the minimum image resolution needed to accomplish the task.

[6] also described the composition of TVIs into MTVIs, time intervals during which collections of tasks can be satisfied simultaneously by one camera. A set of  $n$  TVIs, each represented in the form:

$$(c, (T_i, o_i), [r_i, d_i], Valid_{\psi_i, \phi_i, f_i}(t)),$$

for TVI  $i$  [Eqn. 1] can be combined into a valid MTVI, represented as:

$$(c, \bigcup_{i=1..n} (T_i, o_i), \bigcap_{i=1..n} [r_i, d_i], \bigcap_{i=1..n} Valid_{\psi_i, \phi_i, f_i}(t)), \quad (3)$$

such that:

$$\bigcap_{i=1..n} [r_i, d_i] \neq \emptyset, \quad (4)$$

i.e., there is some common time interval during which they can be scheduled, and:

$$\bigcap_{i=1..n} [r_i, d_i] \geq p_{max},$$

where  $p_{max}$  is the largest processing time among the tasks, and for all  $t \in \bigcap_{i=1 \dots n} [r_i, d_i]$ ,

$$\bigcap_{i=1 \dots n} Valid_{\psi_i, \phi_i, f_i}(t) \neq \emptyset, \quad (5)$$

i.e., the tasks can be captured with common PTZ settings.

Besides [6], other work that has focused on temporal analysis and planning for camera scheduling includes [7, 8], which discuss a dynamic sensor planning system, called the MVP system. They were concerned with determining occlusion-free viewpoints of a target. This involves handling occlusions between the target and the different moving objects in a scene, each of which generates a swept volume in temporal space. Using a temporal interval search, they divide the temporal intervals into halves while searching for a viewpoint that is not occluded in time by these sweep volumes. This is then integrated with other constraints such as focus and field of view in [8]. The culmination of this work is found in [7], where the algorithms are applied to an active robot work cell.

In this paper, we will address the problem of job scheduling given the TVIs and MTVIs generated as in [6]. In general, job scheduling problems are NP-hard, and approximation algorithms have to be employed. We first analyze the approximation factor of a greedy scheduling algorithm (as a function of the number of cameras), which reveals that its performance deteriorates significantly as the number of cameras increases. We then describe a Dynamic Programming (DP) approximation algorithm with an approximation factor that is much better than the greedy approach. The performance advantage of the DP algorithm is confirmed by simulations.

Finally, we describe a prototype real time active camera system. A scheduler controls PTZ cameras in real time to capture video segments based on automatically constructed TVIs and MTVIs. While the prototype system includes only a small number of cameras due to limited resources, the results illustrate the applicability of the algorithms for large scale camera networks.

## 2 Single-camera Scheduling

We first study the scheduling problem when only a single camera is used. This will be extended to the problem of multiple cameras in the next section. Also, we will limit our analysis to non-preemptive schedules in this paper.

We introduce the following theorems that make the single-camera scheduling problem tractable:

**Theorem 1.** *Let the slack for the  $i$ -th task be  $\delta_i = [t_{\delta_i}^-, t_{\delta_i}^+]$ , and define  $\delta_{max} = \max(|\delta_i|)$  and  $p_{min}$  as the smallest processing time among all (M)TVIs for some camera. Then, if  $|\delta_{max}| < p_{min}$ , then any feasible schedule for the camera is ordered by the slacks' start times.*

*Proof.* Consider that the slack  $\delta_1 = [t_{\delta_1}^-, t_{\delta_1}^+]$  precedes  $\delta_2 = [t_{\delta_2}^-, t_{\delta_2}^+]$  in a schedule and  $t_{\delta_1}^- > t_{\delta_2}^-$ . Let the processing time corresponding to  $\delta_1$  be  $p_1$ . Then  $t_{\delta_1}^- + p_1 > t_{\delta_2}^- + p_1$ . We know that if  $t_{\delta_1}^- + p_1 > t_{\delta_2}^+$ , then the schedule is infeasible. This happens if  $t_{\delta_2}^+ \leq t_{\delta_2}^- + p_1$  - i.e.,  $t_{\delta_2}^+ - t_{\delta_2}^- \leq p_1$ . Given that  $|\delta_{max}| < p_{min}$ ,  $t_{\delta_2}^+ - t_{\delta_2}^- \leq p_1$  is true.

Theorem 1 implies that if  $|\delta_{max}| < p_{min}$ , we can limit our attention to feasible schedules that are ordered by the slacks' start times. This is a reasonably close assumption in

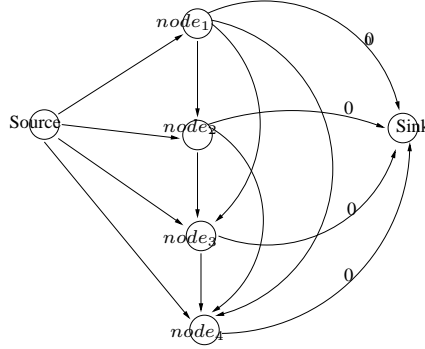
many cases since the time to move the cameras and capture an object is generally quite large compared to the slack times in crowded scenes, where such scheduling matters most. This assumption allows us to construct a Directed *Acyclic* Graph (DAG), where each (M)TVI is a node with an incoming edge from a common source node and outgoing edge to a common sink node, with the weights of the outgoing edges initialized to zero. An outgoing edge from one (M)TVI node to another exists iff the slack's start time of the first node precedes that of the second (Theorem 1), which can however be removed if it makes the schedule infeasible. Consider the following theorem and corollary:

**Theorem 2.** *A schedule - a sequence of  $n$  (M)TVIs each with slack  $\delta_i = [t_{\delta_i}^-, t_{\delta_i}^+]$ , where  $i = 1 \dots n$  represents the order of execution - is feasible if  $t_{\delta_n}^+ - t_{\delta_1}^- \geq (\sum_{i=1 \dots n-1} p_i) - (\sum_{i=1 \dots n-1} |\delta_i|)$ ,  $p_i$  being the processing time of the  $i^{\text{th}}$  (M)TVI in the schedule.*

*Proof.* For the schedule to be feasible the following must be true:  $t_{\delta_1}^- + p_1 \leq t_{\delta_2}^+$ ,  $t_{\delta_2}^- + p_2 \leq t_{\delta_3}^+$ , ...,  $t_{\delta_{n-1}}^- + p_{n-1} \leq t_{\delta_n}^+$ . Summing them up gives  $t_{\delta_1}^- + t_{\delta_2}^- + \dots + t_{\delta_{n-1}}^- + \sum_{i=1 \dots n-1} p_i \leq t_{\delta_2}^+ + t_{\delta_3}^+ + \dots + t_{\delta_n}^+$ , which can then be simplified as  $t_{\delta_n}^+ - t_{\delta_1}^- \geq (\sum_{i=1 \dots n-1} p_i) - (\sum_{i=1 \dots n-1} |\delta_i|)$ . The condition,  $t_{\delta_1}^- + p_1 \leq t_{\delta_2}^+$ ,  $t_{\delta_2}^- + p_2 \leq t_{\delta_3}^+$ , ...,  $t_{\delta_{n-1}}^- + p_{n-1} \leq t_{\delta_n}^+$ , is however only a sufficient condition for a feasible schedule.

**Corollary 1.** *Define a new operator  $\preceq$ , such that if  $\delta_1 (= [t_{\delta_1}^-, t_{\delta_1}^+]) \preceq \delta_2 (= [t_{\delta_2}^-, t_{\delta_2}^+])$ , then  $t_{\delta_1}^- + p_1 \leq t_{\delta_2}^+$ . Consider a schedule of (M)TVIs with slacks  $\delta_{i \dots n}$ . The condition:  $\delta_1 \preceq \delta_2$ ,  $\delta_2 \preceq \delta_3$ , ...,  $\delta_{n-1} \preceq \delta_n$ , is necessary for the schedule to be feasible. Conversely, if a schedule is feasible, then  $\delta_1 \preceq \delta_2$ ,  $\delta_2 \preceq \delta_3$ , ...,  $\delta_{n-1} \preceq \delta_n$ . Proof is omitted since it follows easily from Theorem 2.*

Due to Corollary 1, an edge between two (M)TVI nodes can be removed if it violates the  $\preceq$  relationship, since it can never be part of a feasible schedule.



**Fig. 1.** Single-camera scheduling: DAG formed from the set  $\{node_1 = \{T_1, T_2\}, node_2 = \{T_2, T_3\}, node_3 = \{T_3, T_4\}, node_4 = \{T_5, T_6\}\}$ . The weights between (M)TVI nodes are determined on the fly during DP. Assume that, in this example, the  $\preceq$  relationship is satisfied for the edges between the (M)TVI nodes.

Using such a DAG, a Dynamic Programming (DP) algorithm can be used to solve the single-camera scheduling problem. Consider the following set of (M)TVIs that have been constructed for a given camera, represented by the tasks  $(T_{1..6})$  they satisfy and sorted in order of their slacks' start times:  $\{node_1 = \{T_1, T_2\}, node_2 = \{T_2, T_3\}, node_3 = \{T_3, T_4\}, node_4 = \{T_5, T_6\}\}$ , where the set of nodes in the DAG in Figure 1 is given as  $node_{i=1..4}$ . DP is run by first initializing paths of length 1 starting from each of the (M)TVI nodes to the sink, all with "merit" 0. At each subsequent path length, the next node  $node_{next}$  chosen for a given node  $node_{curr}$  in the current iteration is:

$$node_{next} = \underset{n \in S_{curr2next}}{\arg \max} |S_n \cup Tasks(node_{curr})|, \quad (6)$$

where  $S_{curr2next}$  is the set of nodes that have valid paths starting from them in the previous iteration and for which  $node_{curr}$  has an outgoing edge to.  $S_n$  is defined as the set of tasks covered by the path (in the previous iteration) starting from  $n$ , and  $Tasks()$  gives the set of tasks covered by the (M)TVI associated with  $node_{curr}$ . So, for example, from  $node_1$ , paths of length 2 exist by moving on to either one of  $node_{2..4}$ , with the move to  $node_2$ ,  $node_3$  and  $node_4$  covering  $\{T_1, T_2, T_3\}$  (merits=3),  $\{T_1, T_2, T_3, T_4\}$  (merits=4) and  $\{T_1, T_2, T_5, T_6\}$  (merits=4) respectively. We choose the path of length 2 from  $node_1$  to  $node_3$ . Iterations are terminated when there is only one path left that starts at the source node or a path starting at the source node that covers all the tasks. In our example, the optimal path becomes  $node_1 \rightarrow node_3 \rightarrow node_4$ , terminated at paths of length 4 from the sink when all the tasks are covered.

### 3 Multi-camera Scheduling

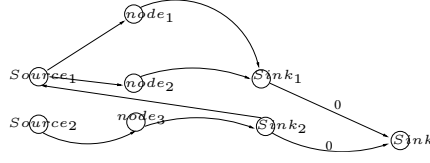
While single-camera scheduling using DP is optimal and has polynomial running time, the multi-camera scheduling problem is unfortunately NP-hard. Consequently, computationally feasible solutions can only be obtained with approximation algorithms. We consider both a simple greedy algorithm and a branch and bound-like algorithm.

#### 3.1 Greedy Algorithm

The greedy algorithm iteratively picks the (M)TVI that covers the maximum number of uncovered tasks, subject to schedule feasibility as given by Theorem 2. Under such a greedy scheme, the following is true:

**Theorem 3.** *Given  $k$  cameras, the approximation factor for multi-camera scheduling using the greedy algorithm is  $2 + k\lambda\mu$ , where the definitions of  $\lambda$  and  $\mu$  are given in the proof.*

*Proof.* Let  $G = \bigcup_{i=1..k} G_i$ , where  $G_i$  is the set of (M)TVIs scheduled on camera  $i$  by the greedy algorithm, and let  $OPT = \bigcup_{i=1..k} OPT_i$ , where  $OPT_i$  is the set of (M)TVIs assigned to camera  $i$  in the optimal schedule. We further define (1)  $H_1 = \bigcup_{i=1..k} H_{1,i}$ , where  $H_{1,i}$  is the set of (M)TVIs for camera  $i$ , that have been chosen by the optimal schedule but not the greedy algorithm and each of these (M)TVIs contains tasks that are not covered by the greedy algorithm in any of the cameras, (2)  $H_2 = \bigcup_{i=1..k} H_{2,i}$ , where  $H_{2,i}$  is the set of (M)TVIs for camera  $i$ , that have been chosen by the optimal schedule but not the greedy algorithm and each of these (M)TVIs contains tasks that are also covered by the greedy algorithm, and finally (3)  $OG = OPT \cap G$ .



**Fig.2.** Multi-camera scheduling: DAG formed from the set  $\{node_1 = \{T_1, T_2, T_3\}, node_2 = \{T_3, T_4\}\}$  for the first camera, and the set  $\{node_3 = \{T_1, T_2, T_3\}\}$  for the second camera.

Clearly,  $OPT = H_1 \cup H_2 \cup OG$ . Then, for  $h_{j=1\dots n_i} \in H_{1,i}$  where  $n_i$  is the number of (M)TVIs in  $H_{1,i}$ ,  $\exists g_{j=1\dots n_i} \in G_i$  such that  $h_j$  and  $g_j$  cannot be scheduled together based on the requirement given in Theorem 2, else  $h_j$  should have been included by  $G$ . If  $Tasks(h_j) \cap Tasks(g_j) = \emptyset$ , then  $h_j$  contains only tasks that are not covered by  $G$ . In this case,  $|h_j| \leq |g_j|$ , else  $G$  would have chosen  $h_j$  instead of  $g_j$ . Note that the cardinality is defined as the number of unique tasks covered. In the same manner, even if  $Tasks(h_j) \cap Tasks(g_j) \neq \emptyset$ ,  $h_j$  could have replaced  $g_j$  unless  $|h_j| \leq |g_j|$ . Consequently,  $|H_{1,i}| = |h_1 \cup h_2 \cup \dots \cup h_{n_i}| \leq |h_1| + |h_2| + \dots + |h_{n_i}| \leq |g_1| + |g_2| + \dots + |g_{n_i}|$ . Let  $\beta_j = \frac{|g_j|}{|G_i|}$  and  $\lambda_i = \max(\beta_j * n_i)$ . This gives  $|H_{1,i}| \leq \beta_1 |G_i| + \dots + \beta_{n_i} |G_i| \leq \lambda_i |G_i|$ . Similarly, we know  $|H_1| \leq \lambda_1 |G_1| + \dots + \lambda_k |G_k| \leq \lambda (|G_1| + \dots + |G_k|)$ , where  $\lambda = \max(\lambda_i)$ . Introducing a new term,  $\gamma_i = \frac{|G_i|}{|G|}$  and letting  $\mu = \max(\gamma_i)$ , we get  $|H_1| \leq k\lambda\mu |G|$ . Since  $|H_2| \leq |G|$  and  $|OG| \leq |G|$ ,  $|OPT| \leq (2 + k\lambda\mu) |G|$ .

### 3.2 Branch and Bound Algorithm

The branch and bound approach runs DP in a similar manner as single-camera scheduling, but on a DAG that consists of multiple source-sink pairs (one pair per camera), with the node of one camera's sink node linked to another camera's source node. An example is shown in Figure 2. Then, for a source node  $s$ , we define its "upper bounding set"  $S_s$  as:

$$S_s = \bigcup_{c \in S_{link}} S_c, \quad (7)$$

where  $S_{link}$  is the set of cameras for which paths starting from the corresponding sink nodes to  $s$  exist in the DAG, and  $S_c$  is the set of all tasks that are covered by some (M)TVIs belonging to camera  $c$ . Intuitively, such an approach aims to overcome the "shortsightedness" of the greedy algorithm by "looking forward" in addition to backtracking and using the tasks that can be covered by other cameras to influence the (M)TVI nodes chosen for a particular camera. Admittedly, better performance is possibly achievable if "better" upper bounding sets are used, as opposed to blindly using all the tasks that other cameras can cover without taking scheduling feasibility into consideration.

The algorithm can be illustrated with the example shown in Figure 2, which shows two cameras,  $c_1$  and  $c_2$ , and the following sets of (M)TVIs that have been constructed for them, again ordered by the slacks' start times and shown here by the tasks ( $T_{1\dots 4}$ )

they satisfy. For  $c_1$ , the set is  $\{node_1 = \{T_1, T_2, T_3\}, node_2 = \{T_3, T_4\}\}$  and for  $c_2$ ,  $\{node_3 = \{T_1, T_2, T_3\}\}$ . The DAG that is constructed has two source-sink pairs, one for each camera -  $(Source_1, Sink_1)$  belongs to  $c_1$  and  $(Source_2, Sink_2)$  to  $c_2$ . The camera sinks are connected to a final sink node as shown, with the weights of the edges initialized to zero. Weights between nodes in the constructed DAG are similarly determined on the fly like in the single-camera scheduling. Directed edges from  $Sink_2$  to  $Source_1$  connects  $c_1$  to  $c_2$ . The DP algorithm is run in almost the same manner as single-camera scheduling, except that at paths of length 3 from the final sink node, the link from  $Source_1$  to  $node_2$ , is chosen because the upper bounding set indicates that choosing the link potentially covers a larger number of tasks (i.e., the upper bounding set of  $Source_1$ ,  $\{T_1, T_2, T_3\}$  combines with the tasks covered by  $node_2$  to form  $\{T_1, T_2, T_3, T_4\}$ ).

The branch and bound algorithm can be viewed as applying the single-camera DP algorithm, camera by camera in the order given in the corresponding DAG, with the schedule of one camera depending on its upper bounding set. This allows us to derive a potentially better approximation factor than the greedy algorithm as follow:

**Theorem 4.** For  $k$  cameras, the approximation factor of the branch and bound algorithm is  $\frac{(1+k\mu(1+u))^k}{(1+k\mu(1+u))^k - (k\mu(1+u))^k}$ .  $\mu$  and  $u$  are defined as follow. Let  $G^* = \bigcup_{i=1\dots k} G_i^*$ , where  $G_i^*$  is the set of (M)TVIs assigned to camera  $i$  by the branch and bound algorithm. Then,  $\mu = \max(\frac{|G_i^*|}{|G^*|})$  and  $u = \max(u_i)$ , where  $u_i$  is the ratio of the cardinality of the upper bounding set of camera  $i$  to  $|G_i^*|$ .

*Proof.* Let  $\alpha$  be the approximation factor of the branch and bound algorithm. Then, assuming that schedules for  $G_1^*, \dots, G_{i-1}^*$  have been determined,  $|G_i^*| \geq \frac{1}{\alpha}(|OPT| - \sum_{j=1}^{i-1} |G_j^*|)$ . Adding  $\sum_{j=1}^{i-1} |G_j^*|$  to both sides gives:

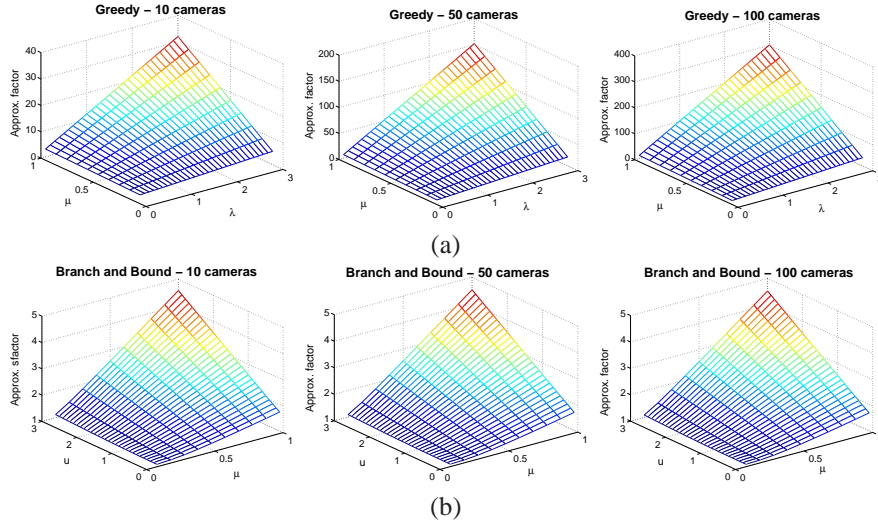
$$\sum_{j=1}^i |G_j^*| \geq \frac{OPT}{\alpha} + \frac{\alpha - 1}{\alpha} \sum_{j=1}^{i-1} |G_j^*|.$$

A proof by induction shows, after some manipulation:

$$\frac{\alpha^k}{\alpha^k - (\alpha - 1)^k} \sum_{j=1}^k |G_j^*| \geq |OPT|.$$

Let  $H = \bigcup_{i=1\dots k} H_i$ ,  $H_i$  being the set of (M)TVIs chosen by the optimal schedule on camera  $i$  but not the branch and bound algorithm. The condition  $|H_i| \leq |G_i^*| + u_i |G_i^*|$  is true; otherwise,  $H_i$  would have been added to  $G^*$  instead. Consequently,  $|H| \leq (|G_1^*| + \dots + |G_k^*|) + (u_1 |G_1^*| + \dots + u_k |G_k^*|) \leq k\mu |G^*| + ku\mu |G^*| \leq k\mu(1+u) |G^*|$ . Since  $OPT = OG \cup H$  (Theorem 3), we get  $|OPT| \leq 1 + k\mu(1+u) |G^*|$ . Thus,  $\alpha = 1 + k\mu(1+u)$ .

By expressing the approximation factors of the greedy and branch and bound algorithm as a function of the number of cameras, we see that the branch and bound algorithm *theoretically* outperforms the greedy algorithm substantially in terms of task coverage. This is illustrated in Figure 3, whereby the approximation factors of the greedy and branch and bound algorithm are plotted as the “distribution” parameters vary when different number of cameras are used. These distribution parameters refer to  $\lambda$  and  $\mu$  in Theorem 3, and  $\mu$  and  $u$  in Theorem 4. They represent how well the tasks are distributed among the cameras and (M)TVIs. The plots show that the greedy algorithm



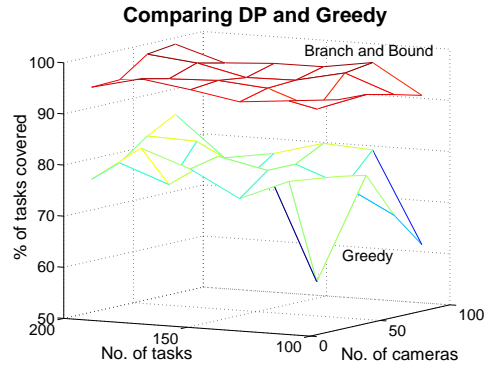
**Fig. 3.** (a) The approximation factor for the greedy algorithm using 10, 50 and 100 cameras respectively.  $\lambda$  and  $\mu$  here are as defined in Theorem 3. (b) The same plots for the branch and bound algorithm. Here, the approximation factor depends only on the distribution parameters and not on the number of cameras.  $u$  and  $\mu$  are as defined in Theorem 4.

is highly sensitive to the number of cameras, with the approximation factor becoming prohibitively high when the tasks are unevenly distributed. On the other hand, the performance of the branch and bound algorithm depends only on the distribution parameters and is not affected by the number of cameras.

Both the single-camera and branch and bound DP-based multi-camera algorithm have a computational complexity of  $O(N^3)$ ,  $N$  being the average number of (M)TVIs constructed for a given camera and used in the resulting DAG. On the other hand, the greedy algorithm takes only  $O(N^2)$  time, which could outweigh the benefits of better scheduling for very large camera networks.

## 4 Implementation and Experiments

Although we have theoretically found the approximation factors for the scheduling algorithms, it would be interesting for practical purposes to investigate the performance of the greedy algorithm relative to the DP algorithm under “normal” circumstances where we would expect “reasonable” task distribution. For this purpose, we conducted simulations using a scene of size  $200m \times 200m$ , and generated moving objects in the scene by randomly assigning to them different starting positions in the scene, sizes and velocities. Cameras are also simulated with calibration data from real cameras. The objects are assumed to be moving in straight lines at constant speeds, and the (M)TVIs for each camera are then constructed and utilized by the scheduler. We conducted simulations for 20, 40, 60, and 80 cameras and 100, 120, 140, 160, 180, and 200 objects, and plot the percentage of the total number of tasks that were captured by both the greedy



**Fig. 4.** The DP algorithm consistently covers more tasks than the greedy algorithm.

and DP algorithm. For each object, the task is to capture video segments in which the full body of the object is visible. Since there is only one task for each object, the total number of tasks equals the number of objects. The results are shown in Figure 4. The DP algorithm schedules more tasks than the greedy algorithm by a minimum of 13.55 percent and a maximum of 33.78 percent.

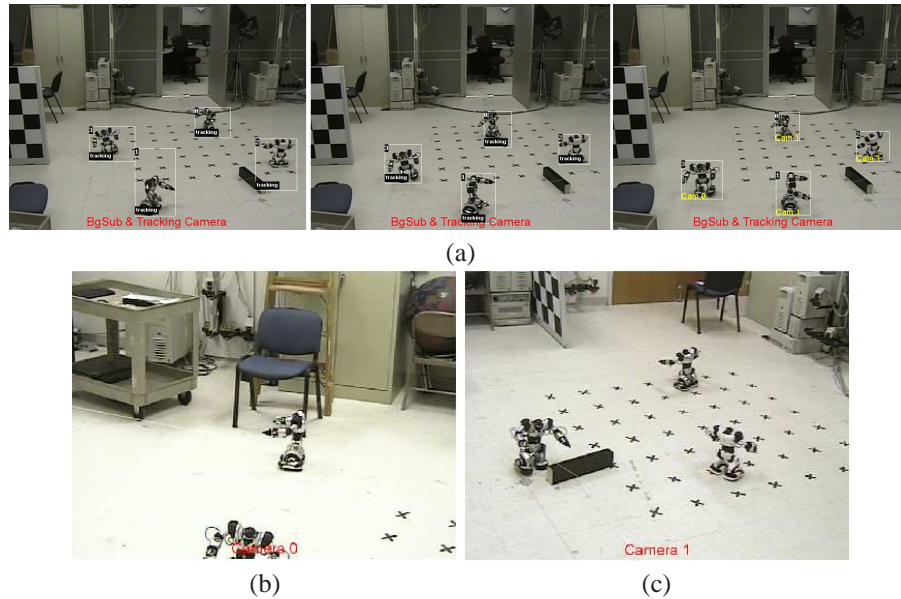
Finally, we test our algorithms in a small-scale *real time* image analysis system. Due to limited resources, building a system with large number of cameras was not possible. We developed a prototype multi-camera system consisting of four PTZ cameras synchronized by a Matrox four-channel card.

For running the experiments, one camera is kept static, so that it can be used for background subtraction and tracking in the sensing stage[9, 1]. From the detection and tracking, the system recovers an approximate 3D size estimate of each detected object from the ground plane and camera calibration. This is followed by the planning stage, during which the observed tracks allow the system to predict the future locations of the objects, and to use them for constructing (M)TVIs, which are then scheduled for capture. The predicted position of each detected object on the ground plane is mapped to the PTZ cameras, after which the 3D size estimate of the object is used to construct a rough 3D model of the object for the corresponding PTZ camera. Such a 3D model is utilized to determine valid ranges of PTZ settings during the construction of TVIs.

The experiments confirm that the greedy algorithm performs faster than the DP algorithm. This makes the greedy algorithm more suitable for our prototype system. A real time experiment using the greedy scheduler is illustrated in Figures 5. Four remote-controllable 12x14 inches robots move through the scene. Two PTZ cameras were needed to capture the four robots using a (one task) TVI and a three-task MTVI.

## 5 Conclusion

This paper addressed scheduling algorithms for smart video capture in large camera networks. We developed approximation algorithms for scheduling using a greedy and a DP based approach. While both theoretically and experimentally, the DP algorithm gives very good results, it is computationally more expensive than the greedy algorithm. A suitable algorithm can thus be chosen depending on the application scenario and computational resources available.



**Fig. 5.** (a) The robots are tracked (left and middle image), and the predicted tracks are used to construct the TVIs and MTVIs, which are then used by the scheduler to assign cameras to the (M)TVIs (annotated in the right image). Next, (b) camera 0 captures robot 3, and (c) camera 1 captures robots 0, 1 and 2 simultaneously.

## References

1. Isard, M., Blake, A.: Condensation - conditional density propagation for visual tracking. *International journal of computer vision* **29** (1998) 5–28
2. Mittal, A., Davis, L.: M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. In: *European Conference on Computer Vision, Copenhagen, Denmark*. (2002)
3. Zhao, T., Nevatia, R.: Bayesian human segmentation in crowded situation. In: *IEEE Conference on Computer Vision and Pattern Recognition*. (2003)
4. Kaucic, R., Perera, A.A., Brooksby, G., Kaufhold, J., Hoogs, A.: A unified framework for tracking through occlusions and across sensor gaps. In: *IEEE Conference on Computer Vision and Pattern Recognition, San Diego, CA*. (2005)
5. Rahimi, A., Dunagan, B., Darrell, T.: Simultaneous calibration and tracking with a network of non-overlapping sensors. In: *IEEE Conference on Computer Vision and Pattern Recognition, Washington DC*. (2004)
6. Lim, S.N., Davis, L.S., Mittal, A.: Constructing task visibility intervals for video surveillance. *ACM Multimedia Systems* (2006)
7. Abrams, S., Allen, P.K., Tarabanis, K.: Computing camera viewpoints in an active robot work cell. *International Journal of Robotics Research* **18** (1999)
8. Tarabanis, K., Tsai, R., Allen, P.: The.mvp sensor planning system for robotic vision tasks. *IEEE Transactions on Robotics and Automation* **11** (1995) 72–85
9. W.E.L.Grimson, C.Stauffer: Adaptive background mixture models for real-time tracking. In: *IEEE Conference on Computer Vision and Pattern Recognition*. (1999)