Online Control of Active Camera Networks for Computer Vision Tasks

ADRIAN ILIE, University of North Carolina at Chapel Hill
GREG WELCH, University of North Carolina at Chapel Hill and University of Central Florida

Large networks of cameras have been increasingly employed to capture dynamic events for tasks such as surveillance and training. When using active cameras to capture events distributed throughout a large area, human control becomes impractical and unreliable. This has led to the development of automated approaches for online camera control. We introduce a new automated camera control approach that consists of a *stochastic performance metric* and a *constrained optimization method*. The metric quantifies the uncertainty in the state of multiple points on each target. It uses state-space methods with stochastic models of target dynamics and camera measurements. It can account for occlusions, accommodate requirements specific to the algorithms used to process the images, and incorporate other factors that can affect their results. The optimization explores the space of camera configurations over time under constraints associated with the cameras, the predicted target trajectories, and the image processing algorithms. The approach can be applied to conventional surveillance tasks (e.g., tracking or face recognition), as well as tasks employing more complex computer vision methods (e.g., markerless motion capture or 3D reconstruction).

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search; I.4.9 [**Image Processing and Computer Vision**]: Applications; J.7 [**Computers in Other Systems**]: Command and Control

General Terms: Algorithms, Experimentation, Measurement, Performance

Additional Key Words and Phrases: Camera control, active camera networks, computer vision, surveillance, motion capture, 3D reconstruction

ACM Reference Format:
Adrian Ilie and Greg Welch. 2014. Online control of active camera networks for computer vision tasks. ACM Trans. Sensor Netw. 10, 2, Article 25 (January 2014), 40 pages.
DOI: http://dx.doi.org/10.1145/2530283

1. INTRODUCTION

Many computer vision applications, such as motion capture and 3D reconstruction of shape and appearance, are currently limited to relatively small environments that can be covered using fixed cameras with overlapping fields of view. There is demand to extend these and other approaches to *large environments*, where events can happen in *multiple dynamic locations*, simultaneously. In practice, many such large environments are *sporadic*: events only take place in a few *regions of interest* (ROIs), separated by regions of space where nothing of interest happens. If the locations of the ROIs are static,

This work was supported by ONR grant N00014-08-C-0349 for Behavior Analysis and Synthesis for Intelligent Training (BASE-IT), led by Greg Welch (PI) at UNC, Amela Sadagic (PI) at the Naval Post-graduate School, and Rakesh Kumar (PI) and Hui Cheng (Co-PI) at Sarnoff. Roy Stripling, Ph.D., Program Manager.
Authors' addresses: A. Ilie, University of North Carolina at Chapel Hill, Department of Computer Science; email: adyillie@cs.unc.edu. G. Welch, University of North Carolina at Chapel Hill, Department of Computer Science; University of Central Florida Institute for Simulation & Training and Department of Electrical Engineering & Computer Science.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2014 ACM 1550-4859/2014/01-ART25 $15.00
DOI: http://dx.doi.org/10.1145/2530283

acceptable results can be obtained by straightforward replication of static camera setups used for small environments. However, if the locations of the ROIs are dynamic, coverage needs to be ensured throughout the entire volume. Using an increasing number of fixed cameras is impractical due to concerns over increased requirements in terms of computation and monetary cost, bandwidth and storage.

One practical solution to this problem is using *active cameras* to cover sporadic environments. Active cameras have been used in surveillance [Collins et al. 2000] and in computer vision fields such as motion capture [Davis 2002] and robotics [Davison and Murray 2002]. What makes them versatile is their capability to change their pan and tilt settings to aim in the direction of dynamic ROIs, and zoom in or out to best enclose the ROIs in their field of view. However, this versatility comes at a price: in order to capture dynamic events, active cameras need to be controlled online, in real time. Control decisions need to be made as events are happening, and to take into account *factors* such as target dynamics and camera capabilities, as well as *requirements* from the computer vision algorithms the images are captured for, such as preferred camera configurations, capture durations and image resolutions.

We present an approach that controls a network of active cameras online, in real time, such that they capture multiple events taking place simultaneously in a sporadic environment and produce the best possible images for processing using computer vision algorithms. We approach camera control as an *optimization problem* over the space of possible *camera configurations* (combinations of camera settings) and over time, under *constraints* derived from knowledge about the cameras, the predicted ROI trajectories and the computer vision algorithms the captured images are intended for. Optimization methods rely on objective functions that quantify the "goodness" of a candidate solution. For camera control, this objective function is a *performance metric* that evaluates dynamic, evolving camera configurations over time.

The rest of the article is organized as follows. In Section 2 we present a few performance metrics and touch on their suitability for use with our method. We also list some previous camera control methods encountered in surveillance applications. Section 3 details our performance metric, and Section 4 describes our control method. Section 5 briefly describes how we incorporate task requirements into our approach. In Section 6 we present experimental results. We discuss some future work and conclude the article in Section 7.

Note: This article is an extended version of the results presented at ICDSC 2011 [Ilie and Welch 2011], and builds on research conducted for a doctoral thesis [Ilie 2010]. It presents the progress made in the meantime, with an emphasis on the practical details needed to understand, duplicate and extend our results. Readers interested in more details on the theoretical aspects of our approach are referred to the thesis mentioned.

2. PREVIOUS WORK

2.1. Performance Metrics

Many researchers have attempted to express the intricacies of factors such as placement, resolution, field of view, focus, etc. into metrics that could measure and predict camera performance in diverse domains such as camera placement [Tarabanis et al. 1995], camera selection and view planning. We list a few performance metrics from these domains in the following.

Wu et al. [1998] use the the 2D quantization error on the camera image plane to estimate the uncertainty in the 3D position of a point when using multiple cameras. They model the quantization error geometrically, using pyramids, and the uncertainty region as an ellipsoid around the polyhedral intersection of the pyramids. The article presents a computational technique for determining the uncertainty ellipsoid for an

arbitrary number of cameras. Finally, the volume of the ellipsoid is used as a performance metric. Chen [2002] improves this metric by taking into account probabilistic occlusion, and applies it to optimally place cameras for motion capture. Davis [2002] uses the resulting fixed camera arrangement in combination with steering four pan-tilt cameras for mixed-scale motion recovery. He uses gradient descent to find nearby local minima and avoid large camera maneuvers, and prediction to alleviate the latency in camera response time.

Olague and Mohr [2002] present an approach for camera network design to obtain minimal errors in 3D measurements. Error propagation is analyzed to obtain an optimization criterion. The camera projection model is used to express the relationship between 2D and 3D points, and the error is assumed to come only from image measurements. The covariance matrix of the 3D points is approximated using a Taylor expansion, and the maximum eigenvalue is used as the optimization criterion. Optimization is performed using a genetic algorithm, incorporating geometric and optical constraints such as occlusion.

Chowdhury and Chellappa [2004] address the problem of many algorithms selecting and processing more data than necessary in an attempt to overcome unacceptable performance in their results. They introduce an information-theoretic criterion for evaluating the performance of a 3D reconstruction by considering the change in mutual information between a scene and its reconstructions.

Ram et al. [2006] propose a performance metric based on the probability of accomplishing a given task for placing sensors in a system of cameras and motion sensors. The task is capturing frontal information of a symmetric target moving inside a convex region. Tasks are first decomposed into two subtasks: object localization and image capture. Prior knowledge about the sensors is used to assess the suitability of each sensor for each subtask, forming a performance matrix. Interaction among sensors is decided using the matrix such that assigning sensors to subtasks leads to maximum overall performance. Camera performance is evaluated as the probability of capturing the frontal part of the symmetric object. Object orientation is modeled across a plane as a uniformly-distributed random variable. Motion sensors are transmitter-receiver pairs, placed on a grid. A trade-off between grid density and camera field of view is presented. A performance metric is computed as the capture probability at each grid point, averaged over the entire grid.

Bodor et al. [2005] compute the optimal camera poses for maximum task observability given a distribution of possible target trajectories. They develop a general analytical formulation of the observation problem, in terms of the statistics of the motion in the scene and the total resolution of the observed actions. An optimization approach is used to find the internal and external camera parameters that optimize the observation criteria. The objective function being optimized is directly related to the resolution of the targets in the camera images, and takes into account two factors that influence it: the distance from the camera to each target's trajectory and the angles that lead to forshortening effects.

Mittal and Davis [2004] compute the probability of visibility in the presence of dynamic occluders, under constraints such as field of view, fixed occluders, resolution, and viewing angle. Optimization is performed using cost functions such as the number of cameras, the occlusion probabilities, and the number of targets in a particular region of interest. Mittal and Davis [2008], introduce a framework for incorporating visibility in the presence of random occlusions into sensor planning. The probability of visibility is computed for all objects from all cameras. A deterministic analysis for the worst case of uncooperative targets is also presented. Field of view, prohibited areas, image resolution, algorithmic (such as stereo matching and background appearance) and viewing angle constraints are incorporated into sensor planning, then integrated

with probabilistic visibility into a capture performance metric. The metric is evaluated at each location, for each orientation and each given sensor configuration, and aggregated across space. The aggregated metric value is then optimized using simulated annealing and genetic algorithms.

Denzler et al. [Denzler and Brown 2001; Denzler et al. 2001] present an information theoretic framework for camera data selection in 3D object tracking and derive a performance metric based on the uncertainty in the state estimation process. Denzler et al. [2002] derive a performance metric based on conditional entropy to select the camera parameters that result in sensor data containing the most information for the next state estimation. Denzler et al. [2003] present a performance metric for selecting the optimal focal length in 3D object tracking. The determinant of the a posteriori state covariance matrix is used to measure the uncertainty derived from the expected conditional entropy given a particular action. Visibility is taken into account by considering whether observations can be made and using the resulting probabilities as weights. Optimizing this metric over the space of possible camera actions yields the best actions to be taken by each camera. Deutsch et al. [2004, 2005] improve the process by using sequential Kalman filters to deal with a variable number of cameras and occlusions, predicting several steps into the future and speeding up the computation. Sommerlande and Reid add a Poisson process to model the potential of acquiring new targets by exploring the scene [Sommerlade and Reid 2008b], examine the resulting camera behaviors when zooming [Sommerlade and Reid 2008a], and evaluate the effect on the performance of random and first-come, first-serve (FCFS) scheduling policies [Sommerlade and Reid 2008c]. The performance metric presented in Section 3 is similar to the metric by Denzler et al., but it uses a norm of the error covariance instead of entropy as the metric value, and employs a different aggregation method.

Allen [2007] introduces steady-state uncertainty as a performance metric for optimizing the design of multisensor systems. In previous work [Ilie et al. 2008] we illustrate the integration of several performance factors into this metric and envision applying it to 3D reconstruction using active cameras.

Application domains such as camera placement and selection make use of performance metrics custom-tailored to their requirements. While many existing metrics take into account several quality factors (such as image resolution, focus, depth of field, field of view, visibility, object distance and incidence angle) that have been shown to influence performance in a number of tasks, they are not easily generalized to apply to other tasks. Moreover, many previous approaches focus on just a few of these factors, and none explicitly describe how to account for all the factors, as well as other factors that are important in computer vision applications when using active cameras, such as issues due to the dynamic nature of active cameras (such as mechanical noise, repeatability and accuracy of camera settings), and specific requirements of each computer vision algorithm (such as preferred camera configurations). Our metric accounts for these factors, and is general enough to easily apply to both surveillance and computer vision tasks. In Section 3 we briefly mention where and how each factor is integrated into our metric. The interested reader can find a more detailed discussion on quality factors in Chapter 5 of Ilie [2010].

2.2. Camera Control Methods

Camera control methods are typically encountered in surveillance applications, and many are based on the adaptation of scheduling policies, algorithms and heuristics from other domains to camera control. We lists a few example methods here.

Costello et al. [2004] present and evaluate the performance of several scheduling policies in a master-slave surveillance configuration (a fixed camera and a PTZ camera). Their goal is to capture data for identifying as many people as possible, and it can

be broken into two objectives: capture high resolution images for as many people as possible and view each person for as long as possible. The proposed solution is to observe each target for an empirically-determined period of time, then move on to the next target, possibly returning to the first target if it is still in the scene. The camera scheduling problem is considered similar to a packet routing problem: the deadline and amount of time to serve become known once a target enters the scene. However, the deadlines are only estimated, serving a target for a preset time does not guarantee the task is accomplished, and a target can be served multiple times. This can be treated as a multiclass scheduling problem, with class assignments done based mainly on the number of times a target has been observed (other factors can be taken into account). The paper evaluates several greedy scheduling policies. The static priority (always choose from the highest class) policies analyzed are: random, first come, first serve (FCFS+), earliest deadline first (EDF+). Dynamic priority policies include EDF, FCFS and current minloss throughput optimal (CMTO). CMTO assigns a weight to each class, and tries to minimize the loss due to dropped packets. Scheduling is done by looking ahead to a horizon (cut) specified by the earliest time a packet will be dropped based on the packet deadlines. A list is formed with the highest weight packets with deadlines earlier than the cut, and the packet that results in the most weight served by the cut is selected. EDF+ is shown to outperform FCFS+ and CMTO in percentage of targets captured, but is worst in terms of the number of targets captured multiple times.

Qureshi and Terzopoulos [2005a, 2005b, 2007] present a Virtual Vision paradigm for the design and evaluation of surveillance systems. They use a virtual environment simulating a train station, populated with synthetic autonomous pedestrians. The system employs several wide field-of-view calibrated static cameras for tracking and several PTZ cameras for capturing high-resolution images of the pedestrians. The PTZ cameras are not calibrated. A coarse mapping between 3D locations and gaze direction is built by observing a single pedestrian in a preprocessing step. To acquire images of a target, a camera would first choose an appropriate gaze direction at the widest zoom, then fixate and zoom in after the target is positively identified. Fixation and zooming are purely 2D and do not rely on 3D calibration. Local Vision Routines (LVRs) are employed for pedestrian recognition, identification, and tracking. The PTZ controller is built as an autonomous agent modeled as a finite state machine, with free, tracking, searching and lost as possible states. When a camera is free, it selects the next sensing request in the task pipeline. The authors note that, while bearing similarities to the packet routing problem as described by Costello et al. [2004], scheduling cameras has two significant characteristics that set it apart. First, there are multiple "routers" (in this case, PTZ cameras), an aspect the authors claim is better modeled using scheduling policies for assigning jobs to different processors. Second, camera scheduling must deal with additional sources of uncertainty due to the difficulty estimating when a pedestrian might leave the scene and the amount of time for which a PTZ camera should track and follow a pedestrian to record video suitable for the desired task. Third, different cameras are not equally suitable for a particular task, and suitability varies with time. A weighted round-robin scheduling scheme with a FCFS+ priority policy is proposed in Qureshi and Terzopoulos [2005a] for balancing two goals: getting high resolution images and viewing each pedestrian for as long or as many times as possible. Weights are modeled based on the adjustment time and the camera-pedestrian distance. The danger of a majority of the jobs being assigned to the processor with the highest weight is avoided by sorting the PTZ cameras according to their weights with respect to a given pedestrian and assigning the free PTZ camera with the highest weight to that pedestrian. Ties are broken by selecting the pedestrian who entered the scene first. Other possible tie breaking options like EDF+ were not considered because they require an estimate of the exit times of the pedestrians from the scene, which

are difficult to predict. The amount of time a PTZ camera spends viewing a pedestrian depends upon the number of pedestrians in the scene, with a minimum set based on the number of frames required to accomplish the surveillance task. Weighted scheduling is shown to outperform nonweighted scheduling.

A common surveillance problem is the acquisition of high-resolution images of as many targets as possible before they leave the scene. A possible solution is to translate the problem into a real-time scheduling problem with deadlines and random new target arrivals. Del Bimbo and Pernici [2005] and Bagdanov et al. [2005] propose limiting the temporal extent of the schedules, due to the stochastic nature of the target arrivals and the requirement that a schedule be computed in real time. They also propose taking into account the physical limitations of PTZ cameras, specifically the fact that zooming is much slower than panning and tilting. The camera is modeled as an interceptor with limited resources (adjustment speeds), and the target dynamics are assumed known or predictable. The overall stochastic problem is decomposed into smaller deterministic problems for which a sequence of saccades can be computed. The problem of choosing the best subset of targets for a camera to intersect in a given time is an instance of Time Dependent Orienteering (TDO): given a set of moving targets and a deadline, find the subset with the maximum number of targets interceptable before the deadline. TDO is a problem for which no polynomial-time algorithm exists. The optimal camera tour for a set of targets is computed by solving a Kinetic Traveling Salesperson Problem (KTSP): given a set of targets that move slower than the camera and the camera's starting position, compute the shortest time tour that intercepts all targets. KTSP has been shown to be NP-hard. After limiting the schedule duration, KTSP is reformulated as a sequence of TDO problems. Targets are placed in a queue sorted on their predicted residual time to exit the scene, and an instance of TDO is solved by exhaustive search for the first 7–8 targets in the queue.

Naish et al. [Naish et al. 2001, 2003; Bakhtari et al. 2006] propose applying principles from dispatching service vehicles to the problem of optimal sensing. They first propose a method for determining the optimal initial sensor configuration, given information about expected target trajectories [Naish et al. 2001]. The proposed method improves surveillance data performance by maneuvering some of the sensors into optimal initial positions, mitigating measurement uncertainty through data fusion, and positioning the remaining sensors to best react to target movements. As a complement of this work, the authors present a dynamic dispatching methodology that selects and maneuvers subsets of available sensors for optimal data acquisition in real time [Naish et al. 2003]. The goal is to select the optimal sensor subset for data fusion by maneuvering some sensors in response to target motion while keeping other sensors available for future demands. Demand instants are known a priori, and scheduling is done up to a rolling horizon of demand instants. Sensor fitness is assessed using a visibility measure that is inversely proportional to the measurement uncertainty when unoccluded and zero otherwise. Aggregating the measurements for several sensors is done using the inverse of the geometric mean of the visibility measures for all the sensors involved. A greedy strategy is used to assign the best k sensors for the next demand instant, then to assign remaining sensors to subsequent demand instants until no sensors remain or the rolling horizon is reached. The sensor parameters are adjusted via replanning when the target state estimate is updated. In Bakhtari et al. [2006] describe an updated implementation using vehicle dispatching principles for tracking and state estimation of a single target with four PTZ cameras and a static overview camera.

Lim et al. [2005, 2007] propose solving the camera scheduling problem using dynamic programming and greedy heuristics. The goal of their approach is to capture images that satisfy task-specific requirements such as: visibility, movement direction, camera capabilities, and task-specific minimum resolution and duration. They propose the

concept of *task visibility intervals* (TVIs), intervals constructed from predicted target trajectories during which the task requirements are satisfied. TVIs for a single camera are combined into MTVIs (multiple TVIs). Single camera scheduling is solved using dynamic programming (DP). A directed acyclic graph (DAG) is constructed with a common source and a common sink, (M)TVIs as nodes, and edges connecting them if the slack start time of one precedes the other. DP starts from the DAG sink, adjusts the weights of the edges and terminates when all nodes are covered by a path. Multicamera scheduling is NP-hard, and is solved using a greedy approach, picking the (M)TVI that covers the maximum number of uncovered tasks. A second proposed approach uses branch and bound algorithm that runs DP on a DAG with source-sink subgraphs for each camera, connected by links from the sinks of some subgraphs to the sources of others. The greedy approach is shown to have significantly decreasing performance when the number of cameras increase.

Yous et al. [2007] propose a camera assignment scheme based on the visibility analysis of a coarse 3D shape produced in a preprocessing step to control multiple Pan/Tilt cameras for 3D video of a moving object. The optimization is then extended into the temporal domain to ensure smooth camera movements. The interesting aspect of this work is that it constructs its 3D results from close-up images of parts of the object being model, instead of trying to fit the entire object within the field of view of each camera.

Krahnstoever et al. [2008] present a system for controlling four PTZ cameras to accomplish a biometric task. Target positions are known from a tracking system with 4 fixed cameras. Scheduling is accomplished by computing plans for all the cameras: lists of targets to cover at each time step. Plans are evaluated using a probabilistic performance objective function to optimize the success probability of the biometric task. The objective function is the probability of success in capturing all targets, which depends on a quantitative measure for the performance of each target capture. The capture performance is evaluated as a function of the incidence angle, target-camera distance, tracking performance (worse near scene boundaries), and PTZ capabilities. A temporal decay factor is introduced to allow repeated capture of a target. Optimization is performed asynchronously, via combinatorial search, up to a time horizon. Plans are constructed by iteratively adding camera-target assignments, defining a directed acyclic weighted graph, with partial plans as nodes and difference in performance as edge weights. Plans that cannot be expanded further are terminal nodes and candidate solutions. A best-first strategy is used to traverse the graph, followed by coordinate ascent optimization through assignment changes. All plans are continuously revised at each time instant. New targets are added upon detection from monitoring a number of given entry zones.

Broaddus et al. [2009] present *ACTvision*, a system consisting of a network of PTZ cameras and GPS sensors covering a single connected area that aims to maintain visibility of designated targets. They use a joint probabilistic data association algorithm to track the targets. Cameras are tasked to follow specific targets based on a cost calculation that optimizes the task-camera assignment and performs hand-offs from camera to camera. They compute a "cost matrix" C that aggregates terms for target visibility, distance to maneuver, persistence in covering a target and switching to an already covered target. Availability is computed as a "forbidden matrix" F. They develop two optimization strategies: one that uses the minimum number of cameras needed, and another that encourages multiple views of a target for 3D reconstruction. The task to camera assignment is performed using an iterative greedy k-best algorithm.

Natarajan et al. [2012] propose a scalable decision-theoretic approach based on a Markov Decision Process framework that allows a surveillance task to be formulated as a stochastic optimization problem. Their approach covers m targets using n cameras, where $n \ll m$, with the goal of maximizing the number of targets observed. They

discretize both the space of target locations, directions and velocities, and the space of camera pan tilt and zoom settings. For each camera setting, they precompute the target locations within the camera's field of view and at the appropriate distance range to allow for biometric tasks to be performed on the captured images. Cameras are assumed independent of each other, as are targets. Target transition probability distributions, as well as target visibilities given all possible camera states are precomputed. These assumptions and precomputations result in an online computation time linear in the number of targets. Simulated (up to 4 cameras and up to 50 targets) and real (3 cameras and 6 targets) experiments are presented to validate the approach, which is compared to the approach in Krahnstoever et al. [2008].

Sommerlande and Reid [2010] present a probabilistic approach to control multiple active cameras observing a scene. Similar to our approach, they cast control as an optimization problem, but their goal is to maximize the expected mutual information gain as a measure for the utility of each parameter setting and each goal. The approach allows balancing conflicting goals such as target detection and obtaining high resolution images of each target. The authors employ a sequential Kalman filter for tracking targets in a ground plane. Experiments demonstrate the emergence of useful behaviors such as camera hand-off, acquisition of close-ups and scene explorations, without the use of handcrafted rules. A comparison is presented with independent scanning, FCFS, and random policies, using three metrics: resolution increase, new target detection latency and trajectory fragmentation. Under the assumption that no observation can be detrimental, they avoid the camera-target assignment problem by assigning all cameras to all targets. No attempt is made to reduce the size of the search space.

Another related area for camera control is distributed surveillance, where decisions are arrived at through contributions from collaborating or competing autonomous agents. Proponents of distributed approaches argue that overall intelligent behavior can be the result of the interaction between many simple behaviors, rather than the result of some powerful but complicated centralized processing. Examples of the some of the issues and reasoning behind distributed processing as implemented in 3^{rd} generation surveillance systems can be found in [Marcenaro et al. 2001; Oberti et al. 2001; Remagnino et al. 2003].

Matsuyama and Ukita [2002] describe a distributed system for real-time multitarget tracking. The system is organized in three layers (inter-agency, agency and agent). Agents dynamically interchange information with each other. An agent can look for new targets or and can join an agency which is already tracking a target. When multiple targets get too close to be distinguishable from each other, the agencies tracking them are joined until the targets separate.

Qureshi and Terzopoulos [2005b, 2007] apply their Virtual Vision paradigm for the design and evaluation of a distributed surveillance system. The Local Vision Routines (LVRs) and state model from the centralized system described in Qureshi and Terzopoulos [2005a] are still employed. However, cameras can organize into groups to accomplish tasks using local processing and intercamera communication with neighbors in wireless range. The node that receives a task request is designated as the supervisor and it broadcasts the request to its neighbors. While camera network topology is assumed as known, no scene geometry knowledge is assumed, only that a target can be identified by different cameras with reasonable accuracy. Each camera computes its own relevance for a task, based on whether it is free or not, how well it can accomplish the task, how close it is to the limits of its capabilities, and reassignment avoidance. The supervisor forms a group and greedily assigns cameras to tasks, giving preference to cameras that are free. Cameras are removed from a group when they cease to be relevant to the group task. Intergroup conflicts are solved at the supervisor of one of the conflicting groups as a constraint satisfaction problem, and each

camera is ultimately assigned to a single task. Communication and camera failures are accounted for, but supervisor failure is solved by creating new groups and merging old groups. No performance comparison is attempted between the centralized and distributed scheduling approaches.

In order to ensure adequate coverage of multiple events taking place in a sporadic, large environment, active cameras need to be controlled online, in real time, automatically. Many past surveillance approaches that deal with controlling active cameras are master-slave camera setups, aimed at specific surveillance tasks such as tracking or biometric tasks such as face recognition. These approaches work well in their domains, but are unable to provide the best imagery for complex computer vision tasks such as 3D reconstruction and motion capture, because they are not designed to take into account their specific requirements and the factors that influence their results. For example, typical surveillance applications usually coordinate cameras only to ensure proper hand-offs of targets between them or to prevent redundant assignments of multiple cameras to the same target. In contrast, our approach is designed for collaborative, simultaneous coverage of the same targets by multiple cameras to suit a specific computer vision task, but is general enough to be easily adapted to surveillance tasks. Also, while a few previous approaches take into account the time it takes cameras to change configurations (the *transition time*), none take into account the fact that some computer vision algorithms require precise camera calibrations, which require capturing for at least a minimum *dwell duration*.

When designing our approach, we started with a list of desirable features we wanted it to exhibit. The following list enumerates a few of the features of our approach, together with the approaches referenced in this section that also exhibit these features.

—Evaluate the performance of a camera configuration [Naish et al. 2001, 2003; Bakhtari et al. 2006; Lim et al. 2005, 2007; Qureshi and Terzopoulos 2005a, 2007; Krahnstoever et al. 2008; Broaddus et al. 2009; Matsuyama and Ukita 2002; Natarajan et al. 2012; Sommerlade and Reid 2010].
—Deal with static and dynamic occlusions [Lim et al. 2005, 2007; Broaddus et al. 2009; Sommerlade and Reid 2010].
—Attempt to minimize the time spent by cameras transitioning instead of capturing [Naish et al. 2001, 2003; Bakhtari et al. 2006; Bimbo and Pernici 2005; Bagdanov et al. 2005; Lim et al. 2005, 2007; Qureshi and Terzopoulos 2005a, 2007; Krahnstoever et al. 2008].
—Consider and compare present and future configurations Naish et al. 2001, 2003; Bakhtari et al. 2006; Lim et al. 2005, 2007; Krahnstoever et al. 2008].
—React to to changes in the ROI trajectories [Naish et al. 2001, 2003; Bakhtari et al. 2006; Lim et al. 2005, 2007; Costello et al. 2004; Krahnstoever et al. 2008; Broaddus et al. 2009; Matsuyama and Ukita 2002; Natarajan et al. 2012; Sommerlade and Reid 2010].
—Take into account the time it takes for camera settings to change [Naish et al. 2001, 2003; Bakhtari et al. 2006; Bimbo and Pernici 2005; Bagdanov et al. 2005; Lim et al. 2005, 2007; Qureshi and Terzopoulos 2005a, 2007; Costello et al. 2004; Krahnstoever et al. 2008; Broaddus et al. 2009].
—Deals with new targets entering the scene [Bimbo and Pernici 2005; Bagdanov et al. 2005; Lim et al. 2005, 2007; Costello et al. 2004; Krahnstoever et al. 2008; Broaddus et al. 2009; Matsuyama and Ukita 2002; Natarajan et al. 2012; Sommerlade and Reid 2010].
—Can assign one camera to view multiple targets [Costello et al. 2004; Broaddus et al. 2009; Matsuyama and Ukita 2002; Sommerlade and Reid 2010].

—Can assign multiple cameras to view a single target [Naish et al. 2001, 2003; Bakhtari et al. 2006; Lim et al. 2005, 2007; Qureshi and Terzopoulos 2005a, 2007; Krahnstoever et al. 2008; Broaddus et al. 2009; Matsuyama and Ukita 2002; Sommerlade and Reid 2010].

3. PERFORMANCE METRIC

For many computer vision applications, task performance of a camera configuration depends on its ability to resolve 3D features in the working volume. We measure this ability with our performance metric by using the uncertainty in the state estimation process. The metric is inspired by the performance metric introduced by Allen and Welch [2005], Allen [2007] and the pioneering work of Denzler et al. [Denzler and Zobel 2001; Denzler et al. 2001; Denzler and Brown 2001, 2002; Denzler et al. 2002]. In this section we provide short introductions to state-space models and the Kalman filter, then briefly describe the process by which we compute the uncertainty and arrive at a numeric value suitable for use in our optimization. The interested reader can find more details in [Welch et al. 2007; Ilie et al. 2008; Ilie and Welch 2011], and Chapter 5 of [Ilie 2010].

3.1. State-Space Models

State-space models [Kailath et al. 2000] are used in applications such as Kalman filter-based tracking to mathematically describe the expected target motion and the measurement system. In state-space models, variables (states, inputs and outputs) are represented using vectors, and equations are represented as matrices.

The internal state variables are defined as the smallest possible subset of system variables that can represent the entire system state at a given time. Formally, at time step t, the system state is described by the state vector $\bar{x}_t \in \mathbb{R}^n$. For example in the case of tracking, the user's 3D position is represented by the vector $\bar{x}_t = [x\ y\ z]^\mathsf{T}$. If orientation is also part of the state, the vector becomes $\bar{x}_t = [x\ y\ z\ \phi\ \theta\ \psi]^\mathsf{T}$, where ϕ, θ and ψ are roll, pitch and yaw Euler angles (rotation around the x-, y- and z-axis respectively). The state vector may also be augmented with hidden variables such as target speed and acceleration, if appropriate, depending on the expected characteristics of the target motion. Given a point in the state space, a mathematical *motion model* can be used to predict how the target will move over a given time interval. Similarly, a *measurement model* can be used to predict what will be measured by each sensor, such as 3D GPS coordinates or 2D camera image coordinates.

3.1.1. Motion Model. A motion model (also called *process model*) describes the expected target motion. Traditionally, such models have been described in terms of a physical parameter that stays constant over time, resulting in models for constant position (CP), constant velocity (CV) and constant acceleration (CA) [Chang and Tabaczyinski 1984]. These traditional stochastic models are shown in Figure 1.

In the process of stochastic estimation, integrated normally-distributed random noise is used to replace the constant component of each model. For example, the CV model becomes $x = x_0 + vt$, with velocity $v = \int a$, and acceleration a is a normally-distributed variable $a \sim \mathcal{N}(0, q)$. Incorporating this random component into each model results in the models known as Position (P), Position-Velocity (PV) and Position-Velocity-Acceleration (PVA), respectively [Welch et al. 2007]. Figure 2 uses integrals to illustrate the relation between position x with its temporal derivatives and the "driving" noise source $\mathcal{N}(0, q)$ for the *P*, *PV* and *PVA* models.

Choosing the right model for the expected motion plays a crucial role in obtaining good state estimates. The P model is most appropriate for situations where there is little to no motion. The PV model is used when the motion is fairly constant. The PVA

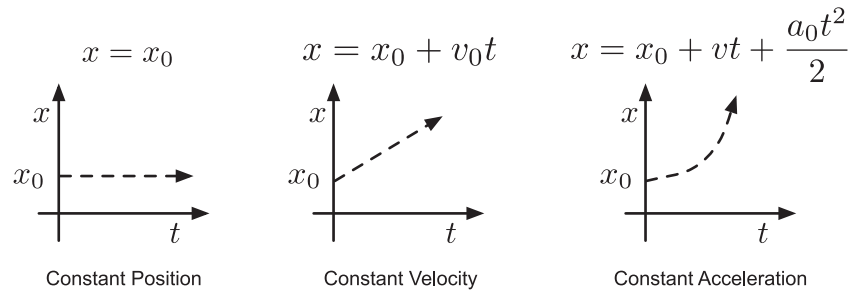

Fig. 1. Traditional motion models, from [Welch et al. 2007].

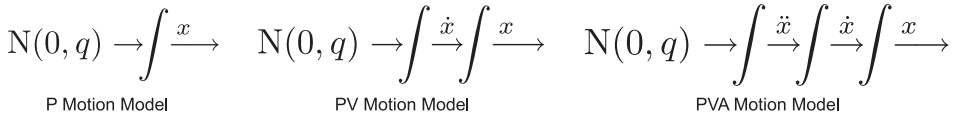

Fig. 2. Stochastic motion models, from [Welch et al. 2007].

model is used for situations in which there are sudden, rapid changes in speed and direction. In Section 3.3 of her thesis [Allen 2007], Allen presents a detailed discussion of these models and gives some examples where they are applied.

For a particular state vector $\bar{x}$, the change in state over time can be modeled using deterministic and random components as follows:

$$\bar{x}_{t+1} = f(\bar{x}_t) + \bar{w}. \tag{1}$$

The state transition function f is the deterministic component that relates the state at time step t to the state at time step $t+1$. The random variable $\bar{w} \sim \mathcal{N}(0, \boldsymbol{Q})$ is called process noise. In practice, f is linearized about the point of interest $\bar{x}$ in the state space by computing the corresponding Jacobian matrix A:

$$A = \frac{\partial}{\partial \bar{x}} f(\bar{x}) \Big|_{\bar{x}}. \tag{2}$$

This results in the following discrete-time linear equation:

$$\bar{x}_{t+1} = A\bar{x}_t + \bar{w}. \tag{3}$$

While such linearizations can lead to sub-optimal results, they provide a computationally efficient means for state estimation (see Allen [2007], Section 6.1.1).

The continuous-time equivalent of Equation (3) is the following:

$$\frac{d\bar{x}}{dt} = A_c\bar{x} + q_c. \tag{4}$$

Here A_c is an $n \times n$ continuous-time state transition matrix, and $\bar{q_c} = [0, \ldots, 0, \mathcal{N}(0, q)]^\mathsf{T}$ is an $n \times 1$ continuous-time process noise vector with corresponding $n \times n$ noise covariance matrix $\boldsymbol{Q_c} = E\{q_c\, q_c^\mathsf{T}\}$, where $E\{\}$ indicates expected value [Welch et al. 2007].

As the actual noise signal $\bar{w}$ in Equation (3) is not known, designers typically estimate the corresponding discrete-time covariance matrix $\boldsymbol{Q}$ instead, by integrating the continuous-time process in Equation (4). The solution to this integration is given in [Grewal and Andrews 1993] as:

$$\boldsymbol{Q} = \int_0^{\delta t} e^{A_c t} \boldsymbol{Q_c} e^{A_c^\mathsf{T} t} dt. \tag{5}$$

Table I. Parameters for the *P*, *PV* and *PVA* Models

Model	$\bar{x}$	A_c	Q_c	Q
P	$[x]$	$\begin{bmatrix} 0 \end{bmatrix}$	$\begin{bmatrix} q \end{bmatrix}$	$[q\,\delta t]$
PV	$\begin{bmatrix} x \\ \dot{x} \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix}$	$\begin{bmatrix} q\frac{\delta t^3}{3} & q\frac{\delta t^2}{2} \\ q\frac{\delta t^2}{2} & q\,\delta t \end{bmatrix}$
PVA	$\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & q \end{bmatrix}$	$\begin{bmatrix} q\frac{\delta t^5}{20} & q\frac{\delta t^4}{8} & q\frac{\delta t^3}{6} \\ q\frac{\delta t^4}{8} & q\frac{\delta t^3}{3} & q\frac{\delta t^2}{2} \\ q\frac{\delta t^3}{6} & q\frac{\delta t^2}{2} & q\,\delta t \end{bmatrix}$

Source: [Welch et al. 2007].

Using the corresponding parameters A_c and Q_c, matrix Q can be computed for the P, PV and PVA models [Welch et al. 2007]. Table I shows the continuous-time parameters (the state x, the transition matrix A_c and the process noise covariance matrix Q_c) and the discrete-time covariance matrix Q for the P, PV and PVA models. Welch and Bishop [2001] discuss the process of choosing q.

3.1.2. Measurement Model. Similarly to the process model, the measurements obtained from a sensor can be modeled using a deterministic and a random component. The observation at time step t is the measurement vector $z_t \in \mathbb{R}^m$. It is related to the state via the following equation:

$$\bar{z}_t = h(\bar{x}_t, \bar{a}_t) + \bar{v}. \tag{6}$$

The nonlinear measurement function h is the deterministic component that relates the state $\bar{x}_t$ to the measurement $\bar{z}_t$. The vector parameter $\bar{a}_t$ is the action taken at time step t, which comprises all parameters that affect the observation process. The action is considered performed before the measurement is taken. The random variable $\bar{v} \sim \mathcal{N}(0, R)$ represents the measurement noise. Just as with the state transition function f, the measurement function h is linearized about the point of interest $\bar{x}$ in the state space by computing the corresponding Jacobian matrix H:

$$H = \frac{\partial}{\partial \bar{x}} h(\bar{x}) \Big|_{\bar{x}}, \tag{7}$$

The measurement model becomes:

$$\bar{z}_t = H\bar{x}_t + \bar{v}. \tag{8}$$

In practice, Jacobian matrix H and measurement noise covariance matrix R are determined through sensor calibration.

The actual noise signal $\bar{v}$ is not known or even estimated. Instead, designers typically estimate the corresponding noise covariance matrix R, and use it to weight the measurements and to estimate the state uncertainty.

State-space models allow taking into account measurements from multiple, heterogeneous sensors. Allen describes such a system in her thesis (Allen [2007], Section 5.2.3). In Section 6, we present experimental evaluations using a hybrid system that takes measurements from multiple cameras and GPS sensors.

In the case of a GPS sensor, the measurement function h transforms a 3D point (latitude, longitude, altitude) into a local 3D coordinate system (x, y, z) used for tracking or 3D reconstruction. This transformation is a 3×3 linear transform H that can be used directly in Equation (8).

In the case of cameras, the measurement function h is embodied by the camera's projection matrix *Proj*, which projects a homogeneous 3D point $[x \;\; y \;\; z \;\; 1]^\mathsf{T}$ to a homogeneous 2D image pixel $[\, u' \;\; v' \;\; 1 \,]^\mathsf{T}$ as follows:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = Proj \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{9}$$

$$u' = u/w \tag{10}$$

$$v' = v/w. \tag{11}$$

The projection matrix Proj is typically determined through a geometric calibration process like the one in Zhang [1999]. One common way to define matrix *Proj* is as follows:

$$Proj = K\big[Rot|Tr\big]. \tag{12}$$

The 3×3 rotation matrix *Rot* and the 3×1 translation vector *Tr* represent the camera extrinsic parameters, and specify the transform between the world coordinate system and the camera's coordinate system. The 3×4 matrix $[Rot|Tr]$ is the concatenation of matrix *Rot* and vector *Tr*. The intrinsic parameters are represented by matrix K, a 3×3 matrix of the form:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{13}$$

f_x and f_y are the camera focal lengths, measured in pixels, in the x and y directions. s is the skew, typically zero for cameras with square pixels. c_x and c_y are the coordinates of the image center in pixels.

Since the camera measurement process embodied by the computation of u' and v' is not linear, the following Jacobian is used in Equation (8).

$$H = \begin{bmatrix} \frac{\partial u'}{\partial x} & \frac{\partial u'}{\partial y} & \frac{\partial u'}{\partial z} \\ \frac{\partial v'}{\partial x} & \frac{\partial v'}{\partial y} & \frac{\partial v'}{\partial z} \end{bmatrix}. \tag{14}$$

The measurement model is where some performance factors are integrated into the metric. The camera field of view and image resolution are integrated into the Jacobian H via the camera projection model. Noise due to focus, mechanical camera components, and target distance is added into the noise covariance matrix R. The interested reader is referred to Section 5.4 of Ilie [2010] for more details.

3.2. The Kalman Filter

The Kalman Filter [Welch and Bishop 2001] is a stochastic estimator for the instantaneous state of a dynamic system that has been used both for tracking and for motion modeling [Krahnstoever et al. 2001]. It can also be used as a tool for performance analysis [Grewal and Andrews 1993] when actual measurements (real or simulated) are available. This section provides a brief introduction.

3.2.1. Equations. The Kalman filter consists of a set of mathematical equations that implement a predictor-corrector type estimator. Its equations can be described using matrices A, H, Q and R defined in the state-space models in Section 3.1, and the initial state covariance P_0. The equations for the predict and correct steps are as follows.

(1) *Time Update* (Predict Step).
—Project state ahead:

$$\hat{x}_t^- = f(\hat{x}_{t-1},\, t-1). \tag{15}$$

$\hat{x}_t^- \in \mathbb{R}^n$ is the a priori state estimate at time step t, $\hat{x}_{t-1} \in \mathbb{R}^n$ is the a posteriori state estimate at time step $t-1$, given measurement z_{t-1}, and f is the state transition function.
—Project error covariance ahead:

$$P_t^- = A_t P_{t-1} A_t^{\mathsf{T}} + Q. \tag{16}$$

A_t is the Jacobian matrix of partial derivatives of the state transition function f with respect to x at time step t.

(2) *Measurement Update* (Correct Step) – can only be performed if a measurement is available.
—Compute Kalman gain:

$$K_t = P_t^- H_t^{\mathsf{T}} \left(H_t P_t^- H_t^{\mathsf{T}} + R\right)^{-1}. \tag{17}$$

H_t is the Jacobian matrix of partial derivatives of h with respect to x at time step t. Since h is a function of the selected action $\bar{a}_t$, both H_t and K_t are functions of $\bar{a}_t$.
—Update state estimate with measurement $\bar{z}_t$:

$$\hat{x}_t^+ = \hat{x}_t^- + K_t\left(\bar{z}_t - h\left(\hat{x}_t^-,\, a_t\right)\right). \tag{18}$$

The expression $K_t(\bar{z}_t - h(\hat{x}_t^-,\, \bar{a}_t))$ is called *innovation*, and quantifies the change in state over a single time step.
—Update error covariance:

$$P_t^+ = (I - K_t H_t)\, P_t^-. \tag{19}$$

Note that the a posteriori state covariance P_t^+ does not depend on the measurement $\bar{z}_t$. This allows evaluation of P_t^+ over time in absence of measurements.

3.2.2. Sequential Evaluation. The sequential Kalman filter is a sequential evaluation method for the Kalman filter. The time update (predict) phase is identical to the one in the standard Kalman filter. The sequential evaluation takes place in the measurement update (correct) phase. Each sensor $s = 1 \ldots c$ is given its own subfilter. The estimate $\hat{x}_t^-$, P_t^- from the predict phase becomes the a priori state estimate for the first subfilter:

$$\hat{x}_t^{-(1)} = \hat{x}_t^- \tag{20}$$

$$P_t^{-(1)} = P_t^-. \tag{21}$$

Each sensor s incorporates its measurement $\bar{z}_t^{(s)}$, as in Equation (18):

$$\hat{x}_t^{+(s)} = \hat{x}_t^{-(s)} + K_t^{(s)}\left(\bar{z}_t^{(s)} - h^{(s)}\left(\hat{x}_t^{-(s)},\, \bar{a}_t^{(s)}\right)\right). \tag{22}$$

The output state $\hat{x}_t^{+(s)}$ of each subfilter becomes the input state $\hat{x}_t^{-(s+1)}$ for the next subfilter:

$$\hat{x}_t^{-(s+1)} = \hat{x}_t^{+(s)}. \tag{23}$$

Equations (22) and (23) can be aggregated into a single expression for the entire set of c subfilters:

$$\hat{x}_t^{+(c)} = \hat{x}_t^{-(1)} + \sum_{s=1}^{c} K_t^{(s)}\left(\bar{z}_t^{(s)} - h^{(s)}\left(\hat{x}_t^{-(s)}, \bar{a}_t^{(s)}\right)\right). \tag{24}$$

Let $C_t^{(s)}$ be the contribution to the error covariance of each sensor s at time step t, computed as:

$$C_t^{(s)} = I - K_t^{(s)} H_t^{(s)}. \tag{25}$$

The error covariance can be updated with the contribution $C_t^{(s)}$, as in Equation (19):

$$P_t^{+(s)} = C_t^{(s)} P_t^{-(s)}. \tag{26}$$

The output covariance $P_t^{+(s)}$ of each subfilter becomes the input covariance $P_t^{-(s+1)}$ for the next subfilter:

$$P_t^{-(s+1)} = P_t^{+(s)}. \tag{27}$$

Equations (26) and (27) can be aggregated into a single expression for the entire set of c subfilters:

$$P_t^{+(c)} = \prod_{s=1}^{c} C_t^{(s)} P_t^{-(1)}. \tag{28}$$

If a sensor does not generate a measurement during a particular time step, the sequential Kalman filter allows simply skipping incorporating its contribution into Equations (24) and (28). However, the contribution $C_t^{(s)}$ of each sensor s at time step t depends on the a priori covariance of subfilter s, so the final a posteriori state $\hat{x}_t^{+} = \hat{x}_t^{+(c)}$ and covariance $P_t^{+} = P_t^{+(c)}$ depend on the order in which the subfilters are evaluated. In practice, the effects of ordering are usually ignored.

3.3. Estimating and Predicting Performance

We define the performance of a camera configuration as its ability to resolve features in the working volume, and measure it using the uncertainty in the state estimation process. Uncertainty in the state $\bar{x}$ can be measured using the error covariance P_t^{+} computed in the Kalman filter Equation (19).

In Ilie et al. [2008], we introduced the concept of *surrogate models* to allow evaluation of the metric in state-space only where needed: at a set of 3D points associated with each ROI. The metric values are aggregated over the state elements in the surrogate model of each ROI, over each ROI group and over the entire environment. At all aggregation levels, weights can be used to give more importance to a particular element. The choice of surrogate model is paramount, as it allows incorporating task requirements into the metric. Section 5 presents a few examples.

Due to the dynamic nature of the events being captured and the characteristics of the active cameras used to capture images, time needs to be considered as a dimension of the search space. Spatial aggregation of metric values over the environment for the current camera configuration is not sufficient, and future camera configurations need to be evaluated as well. This results in the performance metric evaluating a *plan*: a

temporal sequence of camera configurations up to a *planning horizon*. The difficulty is that at each time instant a camera's measurement can be successful or unsuccessful, depending on whether the ROI whose position is being measured is visible or not. Denzler et al. [2002] introduced a way to deal with visibility at each step. Deutsch et al. [2004] extended this approach to multiple steps into the future using a visibility tree, and then sped up the evaluation by linearizing the tree and extended the approach to multiple cameras using a sequential Kalman filter in Deutsch et al. [2006]. We employ a similar approach, but use a norm of the error covariance P_t^+ instead of entropy as our performance metric, and a different method to aggregate it over space and time. As the metric measures the uncertainty in the state, when comparing camera configurations smaller values are better.

Our metric computation works in tandem with the Kalman filter that is used to estimate the ROI trajectories. At each time instant, the filter incorporates the latest measurements from cameras and other sensors, and saves the current estimate (x_0^-, P_0^-). This estimate is the starting point for all metric evaluations. To evaluate a camera plan, we repeatedly perform sequential evaluations of the Kalman filter equations, stepping forward in time, while using process models to predict ROI trajectories and updating the measurement models with the corresponding planned camera parameters. When looking into the future, no actual measurements $\bar{z}_t$ are available at time t, but estimated measurements $\hat{z}_t = h(\hat{x}_t^-, a_t)$ can be used instead. Substituting $\hat{z}_t$ for $\bar{z}_t$ results in zero innovation. Equation (18) becomes simply:

$$\hat{x}_t^+ = \hat{x}_t^-. \tag{29}$$

At each time step, a camera measurement can be successful or not, depending on a variety of factors such as visibility, surface orientation, etc. If the measurement is assumed successful, the a posteriori state error covariance P_t^+ is computed as in Equation (19). If the measurement is assumed unsuccessful, the measurement update step cannot be performed, and $P_t^+ = P_t^-$. The two outcomes can be characterized by two distributions with the same mean $\hat{x}_t^+$ and covariances P_t^+ and P_t^-. Given the probability that a measurement is successful *ms*, these distributions can be considered as components of a Gaussian mixture $\mathcal{M}$ [Deutsch et al. 2006]:

$$\mathcal{M} = ms \cdot \mathcal{N}\left(\hat{x}_t^+, P_t^+\right) + (1 - ms) \cdot \mathcal{N}\left(\hat{x}_t^+, P_t^-\right). \tag{30}$$

The covariance of the Gaussian mixture $\mathcal{M}$ is:

$$P_t^{+'} = ms \cdot P_t^+ + (1 - ms) \cdot P_t^- = (I - ms \cdot K_t H_t)\, P_t^- \tag{31}$$

Since the two distributions have the same mean $\hat{x}_t^+$, $\mathcal{M}$ is unimodal and can be approximated by a new Gaussian distribution $\mathcal{M}'(\hat{x}_t^+, P_t^{+'})$, as shown in Deutsch et al. [2006]. It follows that in order to incorporate the outcome of an observation, one simply has to compute the success probability and replace the computation of the a posteriori error covariance in Equation (19) in the Kalman correct step with the one in Equation (31). The measurement success probability *ms* is where performance factors that affect visibility (such as occlusions and incidence angle) are integrated into the metric. The interested reader is referred to Section 5.4 of [Ilie 2010] for more details.

To aggregate over time, Deutsch et al. [2004, 2006] propose simply using the entropy value at the horizon. However, this value is very sensitive to the camera configurations and ROI positions during the last few time steps before the horizon. For example, when an ROI is occluded in camera view, the uncertainty increases to reflect the absence of measurements. Depending on the circumstances, such an increase during the last time steps before the horizon could end up penalizing plans that perform well during previous time steps. Conversely, a plan where the cameras become unoccluded during the last time steps before the horizon can end up favored over a plan that

ALGORITHM 1: Function $m = Metric(R, S, H)$

Input: set of ROIs R, set of sensors S, planning horizon H
Output: metric value m

```
m = 0;
for all ROIs r ∈ R do
    x_0^- = GetCurrentState(r);
    P_0^- = GetCurrentCovariance(r);
    m_r = 0;
    for t = 1...H do
        (x_t^-, P_t^-) = KalmanPredict(x_{t-1}^+, P_{t-1}^+);
        for all sensors s ∈ S do
            ApplySettings(s_plan, t);
            (x_t^+, P_t^+) = KalmanCorrect(s, x_t^-, P_t^-);
        end
        m_t = 0;
        for all points p in the model of r do
            m_t = m_t + SqrtMaxDiag(P_{t,p}^+) · w_p;
        end
        m_r = m_r + m_t · v_t;
    end
    m = m + m_r · u_r;
end
```

has the cameras unoccluded up until just before the horizon. To fully characterize the evolution of the metric value over time, our approach is to aggregate all the values up to the horizon instead. We use equal weights for all time steps, but different weights can be employed, for example, to emphasize the first few time steps, when trajectory predictions are more reliable.

In summary, our performance metric is computed by repeatedly stepping through the sequential Kalman filter equations and changing relevant state-space model parameters at each time step. The state is initialized using the current Kalman filter state estimate. Aggregation over space and time is performed using weighted sums, with weights being used to give more importance at various levels, such as to a point in a ROI's surrogate model, to a ROI, to a ROI group, or to a time instant. Equation (32) illustrates the general formula for the metric computation.

$$\mathcal{M} = \sum_{r=1}^{nROIs} u_r \left(\sum_{t=1}^{H} v_t \left(\sum_{p=1}^{N_r} w_p (SqrtMaxDiag(P_{t,p}^+)) \right) \right) \tag{32}$$

$nROIs$ is the number of ROIs, N_r is the number of points in the surrogate model of ROI r, H is the planning horizon. u_r, v_t and w_p are relative weights for each ROI r, time step t, and model point p, respectively. $P_{t,p}^+$ is the a posteriori covariance for model point p at time t. Algorithm 1 presents the process in detail.

To convert the error covariance into a single number, the function $SqrtMaxDiag()$ returns the square root of the maximum value on the diagonal of the portion of the error covariance matrix $P_{t,p}^+$ corresponding to the position part of the state. We chose the diagonal maximum because it results in the smallest position uncertainty in all three directions of the 3D space. One advantage of using the square root of the highest covariance in the metric function is that the measurement unit for the metric is the same as the measurement unit of the state space. For example, if the state consists of 3D point positions measured in meters, the metric value will also be in meters. This makes it more intuitive for a system user to specify application requirements such as

the desired maximum error in a particular area where important events take place. Entropy can also be used to convert the error covariance into a single number, but it is more expensive to compute, and does not have the same real-world units as the square root of the diagonal maximum.

The interested reader is referred to Chapter 5 of Ilie [2010] for a detailed discussion on how we arrived at our performance metric, how it differs from previous approaches, and how it incorporates quality factors known to influence the performance of camera configurations.

4. CONTROL METHOD

We define optimization in active camera control as the exploration of the space of possible solutions in search for the best solution as evaluated by the performance metric: the minimum state uncertainty. In [Ilie 2010], we showed that exhaustively exploring the space of combinations of camera pan, tilt and zoom settings is intractable, even when applying heuristics to reduce the search space size. Instead, we explore the space of camera-ROI *assignments*, and compute the best settings corresponding to each assignment using geometric reasoning: the best results are usually obtained when the ROI trajectories are enclosed in the camera fields of view as tightly as possible (see Section 4.4). Evaluating all possible combinations of plans for all cameras is intractable as well, but this search space features better opportunities to reduce its size. We performed a careful analysis of the search space complexity, revealing multiple heuristics that reduce the search space size, and conducted experiments using our metric to evaluate them. While necessarily limited in scope, the experiments confirmed that the heuristics were performing as expected. The interested reader is referred to Chapter 4 of Ilie [2010] for details on how and why we chose the specific set of heuristics in this section to reduce the size of the search space and ensure real-time performance.

During our analysis of the search space, we found that using proximity to decompose the optimization problem into subproblems and solving each subproblem independently was the heuristic most effective at reducing the search space size. As a result, our camera control method consists of two components: centralized *global assignment* and distributed *local planning*. The global assignment component groups ROIs into *agencies* based on proximity to each other and assigns the appropriate cameras to each agency. The local planning component is run at the level of each agency, and is responsible for finding the best plans for all the cameras assigned to that agency. The main advantage this strategy offers is that the subproblems associated with each agency can be solved independently, in parallel. Another advantage is the opportunity to run the two components of our approach at different frequencies: for example, the global assignment component can be run once every N cycles, while the local planning component can be run once per cycle at the level of each agency.

We perform a complete optimization during a *planning cycle*. For simplicity, and without loss of generality, we set the duration of a planning cycle to 1 second. Our current implementation, although not parallelized, still runs online, in real time (a complete optimization per second). During each cycle, the optimization process first predicts the ROI trajectories up to the planning horizon, then uses them to construct and evaluate a number of candidate plans for each camera. A plan consists of a number of *planning steps*, which in turn consist of a *transition* (during which the camera changes its settings) and a *dwell* (during which the camera captures, with constant settings). Candidate plans differ in the number and duration of planning steps up to the planing horizon. We set the planning horizon to the minimum between a predefined number of seconds (set to ensure real-time performance) and the duration of the longest possible camera capture, given the camera positions and capabilities and the predicted ROI trajectories.

4.1. Notation

Before presenting the algorithms that comprise our method, we introduce a few notation elements. We use standard sets notation: $\{\ldots\}$ is a set, $\emptyset$ is the empty set, $\subset$ represents a subset, $\in$ represents membership, $\equiv$ represents equality, $\neq$ represents inequality, $\setminus$ represents set difference, $\cup$ represents set union, and $\cap$ represents set intersection. Sets are denoted by capital letters like A, C, PS, R, R' and set members by small letters like a, c, p, r. A set member can be selected by its number. For example, $C\,[n]$ is the n-th camera in set C.

Additionally, we use several variables, including the following:

—a plan P is denoted as a set of configurations over time, to which standard set operations can be applied,
—$P\,[start \ldots stop]$ is plan P between times *start* and *stop*,
—P_a is the plan for all cameras in agency a, and $P_{a,c}$ is the plan for camera c in agency a,
—$a.Current\ Plan$ is the current plan for agency a,
—$r.Trajectory\,[t]$ represents the surrogate model of ROI r at time t, from which points can be selected, the *top* and *bottom* points in particular,
—R_a is the set of ROIs in agency a,
—C_a is the set of cameras assigned to agency a,
—$C_{avail.}$ is the set of available cameras,
—$c.Useful$ is a Boolean variable that specifies whether a camera c has been designated as useful or not,
—$c.TransitionDuration$ and $c.UninterruptibleDwellDuration$ are the transition and uninterruptible dwell durations for camera c,
—$c.AspectRatio$ is the aspect ratio of camera c,
—$c.Choices$ is the set of capture choices (start time and duration) for camera c.

4.2. Global Assignment

The global assignment component accomplishes two tasks: grouping ROIs into *agencies* and assigning cameras to each agency. We create agencies by clustering together ROIs that are close to each other and predicted to be heading in similar directions. We use predicted trajectories to cluster the ROIs into a minimum number of non-overlapping clusters of a given maximum diameter.

Standard clustering algorithms such as k-means ([Tan et al. 2005], Chapter 8) are not immediately applicable, because the ROI trajectories are dynamic. Additionally, the membership of all ROI clusters needs to exhibit hysteresis to help keep the assignments stable. When an agency's ROI membership changes, all cameras assigned to it need to reevaluate their current plans. Since available cameras currently assigned to other agencies might contribute more to the modified agency, their possible contributions need to be evaluated as well. If these evaluations result in changes in plans or assignments, cameras may need to transition, and end up spending less time capturing, which usually results in worse performance. To accommodate these requirements, we use a proximity-based *minimal change* heuristic, consisting of three steps.

(1) Assign each unassigned ROI to the agency closest to it if the agency would not become too large; form new agencies for remaining isolated ROIs.
(2) If any agency has become too large spatially, iteratively remove the ROI furthest from all other ROIs from it until the agency becomes small enough.
(3) If any two agencies are close enough to each other, merge them into a single agency.

Algorithm 2 presents the process in detail.

ALGORITHM 2: Function $A = ROIClustering(D, H, R, A)$

Input: cluster diameter D, horizon H, ROI set R, agencies set A
Output: agencies set A

```
PredictROITrajectories(T, H);
for all unassigned ROIs r ∈ R do
    d_min = D;
    for all agencies a ∈ A do
        d = Distance(r, a, H);
        if d < d_min then
            R_a = R_a ∪ {r};
            d_min = d;
            a_min = a;
        end
    end
    if d_min ≡ D then
        a = CreateAgency();
        A = A ∪ {a};
    end
    else
        a = a_min
    end
    R_a = R_a ∪ {r}
end
for all agencies a ∈ A do
    repeat
        {r_1, r_2} = MostDistant2ROIs(R_a, H);
        if Distance(r_1, r_2, H) > D then
            R' = R_a \ {r_1, r_2};
            if Distance(r_1, R', H) > Distance(r_2, R', H) then
                r = r_1
            end
            else
                r = r_2
            end
            R_a = R_a \ {r};
            AddToClosestAgency(r, D, H, A);
            if R_a ≡ ∅ then
                A = A \ {a}
            end
        end
    until Distance(r_1, r_2, H) ≤ D;
end
repeat
    {a_1, a_2} = Closest2Agencies(A);
    if Distance(a_1, a_2, H) ≤ D then
        a = Merge(a_1, a_2);
        A = A \ {a_1, a_2} ∪ {a};
    end
until Distance(a_1, a_2, H) > D;
```

Once agencies are created, we use a greedy heuristic to assign cameras to each agency, based on their potential contribution to it. Formal approaches such as linear programming are not easily applicable because they are not always guaranteed to provide a solution if the overall problem is infeasible. Only *available* cameras are taken into account. A camera is considered available if its state for the next planning

cycle is the start of a transition or an occlusion. Additionally, a camera's dwell can be interrupted if it has lasted longer than a specified minimum duration. The heuristic is also proximity-based: a camera-agency assignment is evaluated only if the camera is close enough to the ROIs in the agency. Proximity is evaluated in function $IsClose()$ to only try assigning cameras to nearby agencies. The heuristic tries assigning each available camera to all nearby agencies, searching for the camera-agency assignment that best improves the metric value for the agency. Improvement is measured using the ratio m_{after}/m_{before}, and the best improvement bi corresponds to the smallest ratio. The initial value for bi is $1+\epsilon$, where ϵ is a very small number. If a camera does not improve the metric for any agency, its settings are left unchanged for the duration of the current cycle. The resulting plans are compared with plans obtained by prolonging the current plans up to the planning horizon whenever possible, and the greedy assignments are only applied if they perform better. Algorithm 3 presents the process in detail.

ALGORITHM 3: Function $P_{greedy} = GreedyAssignment(A, C, H)$

Input: set of agencies A, set of cameras C, horizon H
Output: heuristic plan P_a
$C_{avail.} = FindAvailableAgents(C)$;
$P_{current} = \emptyset$;
for *all agencies* $a \in A$ **do**
  for *all cameras* $c \in C_a$ **do**
    $P_a = BuildPlan(c, a, 1, H, true)$;
    $P_{current} = P_{current} \cup P_a$;
  end
end
$P_{greedy} = P_{current}$;
for *all cameras* $c \in C_{avail.}$ **do**
  $bi = 1 + \epsilon$;
  $m_{before} = Metric(A, P_{greedy}, H)$;
  $c.Useful = false$;
  for *all cameras* $c \in C_{avail.}$ **do**
    if $\sim IsClose(c, a, H)$ **then**
      continue
    end
    $P_a = BuildPlan(c, a, 1, H, c.agency \equiv a)$;
    $AddCameraToAgency(c, a, P_a)$;
    $m_{after} = Metric(A, P_{greedy} \cup \{P_a\}, H)$;
    $RemoveCameraFromAgency(c, a)$;
    $i_{c,a} = m_{after}/m_{before}$;
    if $i_{c,a} \leq bi$ **then**
      $bi = i_{c,a}$;
      $a_{best} = a$;
      $P_{best} = P_a$;
      $c.Useful = true$;
    end
  end
  if $c.Useful$ **then**
    $AddCameraToAgency(c, a_{best}, P_{best})$;
    $P_{greedy} = P_{greedy} \cup P_{best}$;
  end
  $C_{avail.} = C_{avail.} \setminus \{c\}$;
end

The plans corresponding to each camera-agency assignment are generated heuristically by assuming the worst-case scenario: the camera is repeatedly set to transition, then capture for as long as possible, with a field of view as wide as possible. While other scenarios, in which the camera captures for shorter intervals, may result in better performance by using narrower fields of view, the goal of this heuristic is only to quickly assess the potential contribution a camera can have to the capture of the ROIs in the agency it is being assigned to. The heuristic also takes into account predicted static and dynamic occlusions, and plans transitions during occlusions whenever possible, in order to minimize the time when the camera is not capturing. Static occlusions are precomputed off-line for each camera. Dynamic occlusions are computed online, using the predicted ROI trajectories. Algorithm 4 presents the process in detail.

ALGORITHM 4: Function $P_a = BuildPlan(c, a, start, stop, prolong)$

Input: camera c, agency a, planning *start* and *stop* times, bool *prolong*
Output: heuristic plan P_a

```
t = start;
P_a = a.CurrentPlan;
if prolong then
    P_a = ProlongCurrentCapture(P_a);
    t = EndOfFirstCaptureTime(P_a);
end
P_a = P_a[1...t];
while t < stop do
    t_FoV = WidestFoVCaptureTime(c, a, t, stop);
    t_occl = NextOcclusionStart(c, a, t, stop);
    e = min(t_FoV, t_occl);
    P_a[t...e] = ComputeSettings(c, a, t, e);
    if t_FoV < t_occl then
        t = e;
    else
        t = NextOcclusionEnd(c, a, t, stop) − c.TransitionDuration;
    end
end
```

The number of assignments evaluated is $nAgencies \cdot nCams_{avail.}$. Note that the results of the assignment process are dependent on the order in which cameras are considered. If all possibilities were evaluated instead, the complexity would increase to $nAgencies \cdot nCams_{avail.} (nCams_{avail.} + 1)/2$, and a formula quadratic in $nCams_{avail.}$ is often likely to result in slower than real-time performance.

It is worth noting that two small changes in our global assignment algorithm make it general enough to apply to camera selection as well. A scenario likely to be encountered in practice is that the camera infrastructure installed at a site might have both F fixed cameras and A active cameras, as well as a number D of devices to record the video streams coming from the cameras. If there are not enough devices to record all the streams ($D < A + F$), a modified assignment approach can be used to decide which streams to record. The changes would simply be to return after assigning D cameras and to skip the planning of transitions and dwells for fixed cameras. Selecting a set of cameras to record would not preclude other algorithms that can run in real time (such as tracking) from running using all camera streams, or from potentially contributing their input to our method.

4.3. Local Planning

Local planning at the level of each agency is concerned with the locally-optimal capture of the ROIs in the agency. All cameras assigned to each agency capture all member ROIs, and no further camera-ROI assignment decisions are made at this level. The planning decisions made at this level are on when and for how long each camera should dwell (capture), and when it should transition to a new configuration.

The current configuration of each camera, corresponding to the first step in a plan, is arguably the most important: configurations corresponding to subsequent steps are revised during subsequent optimization cycles, when the planning method can take advantage of updated predictions of ROI trajectories. We take advantage of this and call the resulting planning process *myopic*: exhaustive exploration of all possible dwell durations is only done during the first planning step, and heuristic computations are used for subsequent steps up to the time horizon.

During each evaluation cycle, a decision is made whether to keep the current camera settings or to transition to a new configuration. We reduce the search complexity by having the planning process be *lazy*: only making this decision for the first cycle in a planning step. Subsequent cycles will simply leave in place the result of the decision made during the first cycle. The reasoning behind this heuristic is, again, that the predicted trajectories available during subsequent cycles are less precise, and the planning process will get a chance to make a decision for each cycle when it becomes the current cycle, and is provided with updated trajectory estimates.

The interested reader is referred to Chapter 4 of Ilie [2010] for a discussion on the impact of myopic, lazy, and other planning heuristics on the size of the search space. While significantly reducing the search space, they can still sometimes result in too many plans to evaluate for running in real time. We further reduce the number of candidate plans by planning transitions during occlusions and limiting how many planning steps of minimum length can fit before the planning horizon. The heuristic employed for completing plans beyond the first step is the same one used during global assignment: function *BuildPlan* (), described in Algorithm 4.

Static occlusions do not depend on the camera parameters, so they are precomputed for each camera in the variable *c.Visib.* and used to further reduce the number of possible plans in the *ComputeStaticVisibilities* () function. Other restrictions, such as how many planning steps of minimum length can fit before the planning horizon, can be taken into account as well in the *FindChoices* () function.

To achieve online, realtime control, the set of candidate plans can be sorted so that the most promising plans are evaluated first. One can use prior experimental observations to derive criteria for quickly judging a plan's potential without a costly metric evaluation.

While not a guarantee that the best plan would be chosen on time, we have found this combination of heuristics to closely approximate an exhaustive search. The final result of the planning heuristics is a set of candidate plans for each camera. All possible combinations of candidate plans for all cameras are explored exhaustively using backtracking. Algorithm 5 presents the local planning procedure, Algorithm 6 details the backtracking procedure, and Algorithm 7 shows a simple example of how the set of choices for the first planning step's end time can be computed.

4.4. Computing Camera Settings

Once the start cycle and capture duration are decided for a planning step in a camera's plan, we use geometric reasoning to compute the corresponding camera pan, tilt and zoom values that ensure optimal coverage of all the ROIs in the agency the camera has

ALGORITHM 5: Function $P_a = LocalPlanning\,(C_a, a, h)$

Input: camera set C_a, agency a, planning horizon h
Output: local plan P_a
$C_{avail.} = FindAvailableAgents\,(C_a)$;
for *all cameras* $c \in C_{avail.}$ **do**
    $c.Visib. = ComputeStaticVisibilities\,(c, a, 1, h)$;
    $P_{a,c}\left[1 \ldots h\right] = BuildPlan\,(c, a, 1, h, true)$;
    $c.Choices = FindChoices\,(c, a, 1, h)$;
end
$m_{best} = Metric\,(\{a\}, P_a, h)$;
$P_a = BackTrack\,(a, C_{avail.}, 0, 0, P_a, h)$;

ALGORITHM 6: Function $P_{best} = BackTrack\,(a, C, cn, n, P_{best}, h)$

Input: agency a, camera set C, current camera number cn, current choice number n, current best plan P_{best}, horizon h
Output: current local best plan P_{best}
$cn = cn + 1$;
if $cn < size\,(C)$ **then**
    $c = C\,[cn]$;
    for *all choices* $i \in c.Choices$ **do**
        $P_{best} = BackTrack\,(a, C, cn, i, P_{best}, h)$;
    end
else
    $P_a\left[1, c.Choices\,(n)\right] = ComputeSettings\,(c, a, 1, c.Choices\,(n))$;
    $P_a\left[c.Choices\,(n) + 1, h\right] = BuildPlan\,(c, a, c.Choices\,(n) + 1, h, true)$;
    $m = Metric\,(\{a\}, P_a)$;
    if $m < m_{best}$ **then**
        $m_{best} = m$;
        $P_{best} = P_a$;
    end
end

ALGORITHM 7: Function $S = FindChoices\,(c, a, start, stop)$

Input: camera c, agency a, planning *start* and *stop* times
Output: set of choices for the first planning step's end time S
$t = start + c.TransitionDuration + c.UninterruptibleDwellDuration$;
$t_{FoV} = WidestFoVCaptureTime\,(c, a, t, stop)$;
$S = \emptyset$;
while $t \leq t_{FoV}$ **do**
    if $\sim IsOccluded\,(c, t)$ **then**
        $S = S \cup \{t\}$;
    end
end

been assigned to. To speed up computation, we use a 2.5D approximation, illustrated in Figure 3. Algorithm 8 builds a 2D point set PS from the points at the bottom and the projections of the points at the top of the surrogate model of each ROI onto the xy plane as seen from the camera, and computes the camera parameters from 2D angles and distances in the plane. We found that this approximation provides satisfactory results in practice.

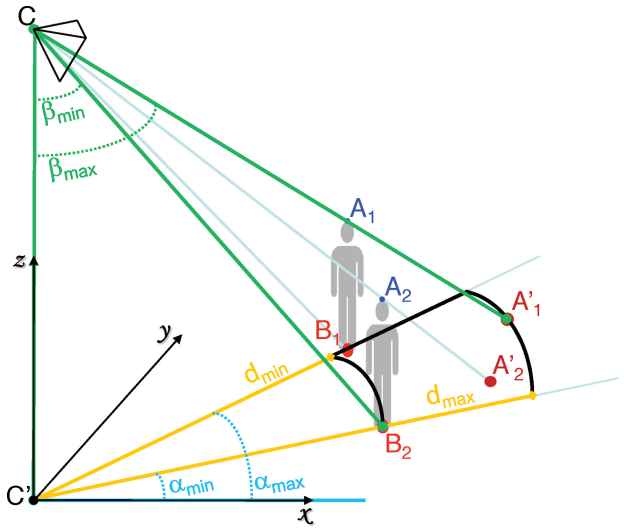

Fig. 3. Computing camera settings.

ALGORITHM 8: Function $PTZ = ComputeSettings\,(c, a, start, stop)$

Input: camera c, agency a, planning $start$ and $stop$ times
Output: PTZ settings for plan step P $[start \ldots stop]$
$PS = \emptyset$;
$t = start$;
while $t \leq stop$ **do**
  for *all ROIs* $r \in R_a$ **do**
    $p_1 = ProjectOntoPlane\,(r.Trajectory\,[t]\,.top)$;
    $p_2 = r.Trajectory\,[t]\,.bottom$;
    $PS = PS \cup \{p_1, p_2\}$;
  end
  $t = t + 1$
end
compute angles α_{min} and α_{max};
compute distances d_{min} and d_{max};
compute angles β_{min} and β_{max};
$HFoV = \alpha_{max} - \alpha_{min}$;
$VFoV = \beta_{max} - \beta_{min}$;
if $HFoV/VFoV > c.AspectRatio$ **then**
  $HFoV = c.AspectRatio * VFoV$;
end
$Pan = {}^{(\alpha_{max}+\alpha_{min})}/_{2}$;
$Tilt = {}^{(\beta_{max}+\beta_{min})}/_{2}$;
$Zoom = {}^{HFoV}/_{MaxHFoV}$;
$PTZ = \{Pan, Tilt, Zoom\}$;

5. INCORPORATING TASK REQUIREMENTS INTO OUR APPROACH

The computer vision algorithm that processes the captured images can be run in realtime, and provide feedback to the camera control method. An example of such an algorithm that can run in realtime is 3D tracking. Alternatively, constraints and requirements can be derived a priori, and more time-consuming computer vision algorithms can be run on the captured images long after the events have taken place. For

example, 3D reconstruction algorithms can be time-consuming, but rigorously specifying their requirements allows the capture of images that can still guarantee good results even without real-time feedback. Many task requirements become easy and intuitive to incorporate into our method due to the performance metric being in state-space. In this section, we give a few examples of how common task requirements can be incorporated, grouped by the application domain. The interested reader is referred to Chapters 5 and 6 of Ilie [2010] for a more detailed presentation of our performance metric and camera control method, including a discussion on applying them to a variety of tasks.

5.1. Tracking

The requirements of a tracking application are naturally expressed in state-space. For example, the user of a tracking system may specify the desired tracking accuracy for a particular region where events might be more important or more likely to happen, or for a particular ROI enclosing a person deemed more important. This importance appears as a weight in the aggregation part of the metric function, and affects the optimization process.

The state can be modeled as a single 3D point if the user is only interested in the ROI's position, or multiple points placed for example at the skeleton joints if the user desires motion capture. Models can be augmented with local surface orientation if the user wants to take into account self-occlusion. The state can also be reduced to 2D if the tracking system employs a ground plane assumption, in which case a single camera measurement is sufficient to determine the ROI position at any time.

5.2. Surveillance

Surveillance tasks have very diverse goals. Here we give two examples of surveillance tasks and how they can be accommodated by our approach.

(1) Following a person or object throughout the environment: enclose the person or object inside a ROI and set its importance higher than the importance of other ROIs. The camera control method will automatically follow the more important ROI and exhibit complex behaviors such as coordination and hand-offs between cameras.
(2) Capturing images of people's faces for biometric tasks: augment the surrogate model with points on the person's face and the person's movement and/or gaze direction, and incorporate the incidence angle into the metric computation as a factor in the probability of making a successful measurement ms in Equations (30) and (31). To ensure the capture of as many faces as possible, have a single successful capture drastically decrease the importance of the captured target's ROI. This will result in cameras capturing other faces that are comparably more uncertain. As time passes, repeated captures of the same person's face if still present in the environment can be ensured by having the importance of its ROI increase slowly over time.

5.3. 3D Reconstruction

When the application is a 3D reconstruction method, surrogate models can be adjusted to better fit the requirements of the particular 3D reconstruction approach. We present two examples.

(1) Surrogate models for stereo reconstruction could be augmented with local surface orientation, which can be included in an appropriate aggregation function that uses the angle between the camera ray and the surface normal. The resulting metric would give more weight to samples better suited for stereo matching. This change

results in the metric giving preference to camera configurations that have been shown to work better in stereo reconstruction: cameras placed relatively close to each other and aimed in a similar direction.

(2) The default approach of minimizing the uncertainty in the 3D position of a number of points gives good results in 3D volumetric reconstruction. However, a different kind of surrogate model for volumetric 3D reconstruction could instead consist of a medial axis-like representation: a set of 3D centroids and rays to the surface. The metric to be optimized could then be an aggregation of the uncertainties in the length of the rays together with the ray lengths themselves, since volumetric approaches typically seek a 3D model of minimum volume. Both approaches result in the metric giving preference to camera configurations that have been shown to work better in volumetric reconstruction: cameras placed uniformly around a ROI, in an outside-looking-in arrangement.

5.4. New Targets

There are multiple ways of incorporating new persons or objects entering the environment.

(1) One approach is suggested by previous work in surveillance such as Krahnstoever et al. [2008], and applies to enclosed environments with a limited number of entry points. In this case, previous approaches have scheduled cameras to periodically survey the entry points for any potential new targets. A similar approach can be straightforwardly applied to our method by having static ROIs enclose the entry points, and automatically creating new dynamic ROIs enclosing any new targets when they are detected entering the scene. Given appropriate surrogate models for each ROI, the camera control method would automatically plan available cameras to both survey the static ROIs enclosing the entry points and to track the newly created dynamic ROIs.

(2) Another approach is introduced by Sommerlande and Reid [2008a]. They model the probability of new targets entering the scene using a background Poisson process, and integrate the noise injected by the process into the performance metric from Deutsch et al. [2006]. Since our performance metric is in many respects similar to the one in Deutsch et al. [2006], this approach is easily applicable to it as well.

(3) A third approach is to have the state of new ROIs bootstrapped by input from an external system, for example GPS or a tracking system that uses a few static cameras with wide fields of view to cover the entire scene. This is another area where state-space models demonstrate their usefulness: besides helping to acquire new targets, measurements from these external devices can be incorporated into the sequential evaluation process described in Section 3.2.2, and thus contribute to reducing the uncertainty in the state.

6. EXPERIMENTAL RESULTS

6.1. Simulation-Based Experimental Results

6.1.1. Experiment Setup. A training exercise was performed by members of the United States Marines Corp, and captured on-site at the Sarnoff Corporation in Princeton, NJ. The exercise participants' trajectories were used to test how an implementation of our method can control six PTZ cameras to observe six exercise participants.

The exercise scenario was for a squad of four Marines to patrol, cross a danger zone between two buildings, cordon and search a civilian, neutralize a sniper threat, and move out to secure an area.

We used the game engine-based simulator from Taylor et al. [2007] to run multiple simulations using the same input data. The simulator provides means of controlling

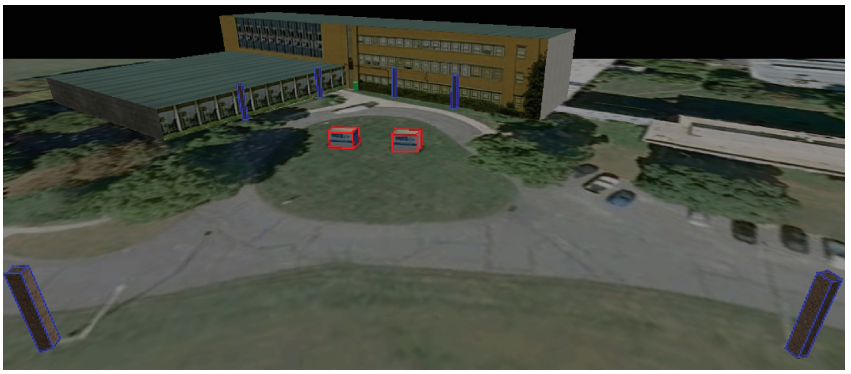

Fig. 4. Overview of the USMC training exercise site as modeled in the simulator.

a number of virtual cameras in a simulated environment and retrieving images from them.

Figure 4 shows an overview of the exercise site modeled inside the simulator. To keep the scenario as realistic as possible, we placed six virtual cameras on existing infrastructure (shown as pillars outlined in blue): four cameras along the building walls and two cameras on light poles in the parking lot. There were also two shipping containers, outlined in red, which served as "buildings" during the exercise and helped us test the impact of static occluders.

The simulator does not provide means of programmatically controlling the virtual characters, so we used a plug-in architecture from Casper [2007] to extend its functionality. We wrote a custom plug-in script to act as a server to which client applications can connect and send commands. We implemented commands to retrieve a list of available participants, move a participant to a new position, change a participant's orientation and retrieve the current position and orientation of a participant.

For each simulated camera image, the simulator also provides ground truth (silhouettes, bounding boxes, total and visible pixel counts) for the virtual characters in the image. This feature is aimed at testing image processing algorithms, but we used it to compute the visibility of a virtual character placed in a particular position as seen from each of the six cameras used in this experiment. We sequentially placed a virtual character at positions on a 2D regular grid spanning the area, aimed all cameras at it, and queried the simulator for the ground truth pixel counts of the virtual character as seen by each camera. Using this process, we precomputed the visibilities over the area where the exercise took place and stored them as a per-camera *visibility maps*. The visibility at a point (x, y) on the 2D grid for camera c was computed as:

$$V_c^{(x,y)} = \frac{PC_c^{visible}}{PC_c^{total}}. \tag{33}$$

where PC_c^{total} and $PC_c^{visible}$ are the counts of total and visible pixels occupied by the virtual character in camera c's image.

Figure 5 shows a top-down view of the exercise site. Camera locations are shown as blue circles, each accompanied by its visibility map shown as a gray-scale image. Brighter map values denote higher visibility. The aggregated visibility is also shown as a gray-scale image overlaid on the top-down satellite view of the site and aligned to match the area where the exercise took place. The two shipping containers which served as props during the exercise appear as dark spots of zero visibility that cast "shadows" in the visibility maps.

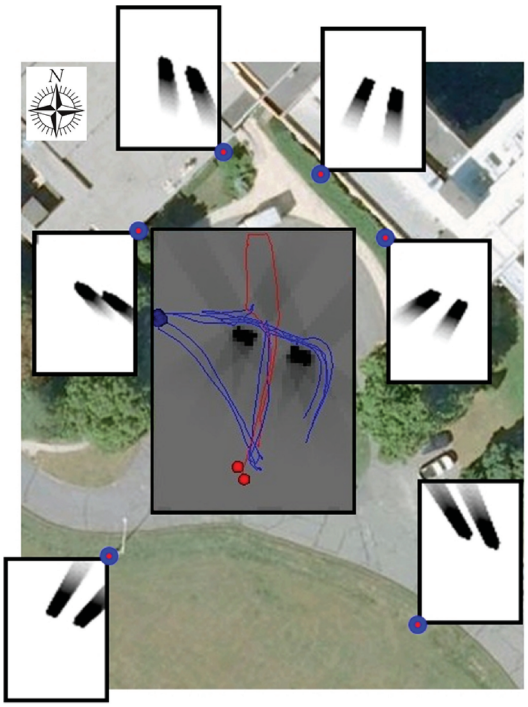

Fig. 5. Top-down view of the exercise site, with aggregated visibility map overlaid on top. Marine tracks are shown in blue, civilian and sniper tracks are shown in red. Camera positions are also highlighted, with visibility maps attached.

During the exercise, participant trajectories were captured as GPS (long., lat., alt.) measurements over time. The approach from Broaddus et al. [2009] was used to capture images from two PTZ cameras and refine the trajectory estimates. We filtered the trajectory data to reduce noise, sampled it at every second and used the samples as input for our implementation, which performed a complete optimization every second. During experiment runs, we used the 3D GPS points on these input trajectories to generate simulated 2D measurements in each virtual camera and we also incorporated them directly into the metric computation as simulated 3D measurements. We added noise (precomputed for repeatability) to each measurement before incorporating it into the performance metric computation.

The surrogate model for each exercise participant consisted of two cubic regions, $1m$ on the side, stacked on the vertical axis. The coordinates of the center of each region were included in the state, and a PV state model [Welch et al. 2007] was used in the Kalman filter-based performance metric evaluation process. To compute the visibility of each cubic region from each camera, we shot rays from the camera's center of projection through seven points associated with the region, and sampled the precomputed visibility map at the intersection point between the ray and the plane of the visibility map. The seven sample points were the region center and the six points at ± 0.5 meters in the x, y and z axis directions. The point visibilities were aggregated as follows: the center point had a weight of $1/2$, and the six exterior points each had a weight of $1/12$. The default visibility threshold used to determine whether an object was occluded was 0.75.

6.1.2. Results. We ran multiple simulations using the same input data and tuning the parameters of our control method. The results we obtained (the interested reader is

referred to Chapter 7 of Ilie [2010] for more details) led to several observations, some of which are summarized as follows.

(1) Using active cameras always performed better than using fixed cameras (simulated by zooming out the active cameras and aiming them at the center of the scene). The positive effect of higher resolution when controlling the cameras was more than enough to compensate for the interruptions in capture due to transitions.
(2) Decreasing the frequency at which GPS measurements were incorporated into the metric (to once every 5 seconds) and eliminating them altogether made the control method zoom out the cameras to compensate for the higher uncertainty, which resulted in worse overall performance.
(3) We treated ROIs with visibility less than a threshold as occluded. We varied the threshold from 0.1 to 1. Increasing the occlusion threshold led to more occlusions, resulting in plans with captures that were more fragmented by transitions, but the increased image resolution compensated for some of the performance loss.
(4) Decreasing the planning frequency (we varied it between once every 1 . . . 10 cycles) to allow for more comprehensive searches resulted in fewer disruptions in camera membership for each agency, but also in an increased number of times when wrong trajectory predictions led to some ROIs not being captured for some time intervals.
(5) Increasing the planning horizon length (we varied it from 10 to 28 cycles) and the clustering diameter (we varied it from 2.5 to 25 meters) led to the heuristic generating longer captures at larger fields of view and lower resolutions, which resulted in worse performance.

Figures 6–8 show how our camera control method automatically achieves desirable complex behaviors: reacting to predicted occlusions, adapting to changes in predicted ROI motion, and coordinating transitions. Participants are grouped into agencies, shown as circles with varying shades of gray. Cameras assigned to a particular agency have their base represented as a square of the same shade of gray. Camera orientation is shown as a triangle color-coded by the camera state: green when capturing, blue when in transition. When capturing, camera fields of view are shown as white lines.

In Figure 6, the four Marines are grouped into an agency, and the civilian and the sniper into a second agency.

At $Time = 11$, Camera 6 is capturing the Marines, which are approaching the first building.

At $Time = 15$, two of the Marines are already behind the first building, and the other two Marines are predicted to follow them, so Camera 6 becomes increasingly less useful and is set to transition to where it can provide better coverage.

At $Time = 17$, the transition has ended, Camera 6 is assigned to a different agency and capturing the civilian and the sniper.

In Figure 7, the four Marines are grouped into an agency, and the civilian and the sniper into a second agency.

At $Time = 21$, the Marines are about to exit the field of view of Camera 3, and the first Marine is moving fast to cross the danger zone between the two buildings.

To avoid losing track of the fast-moving Marine, at $Time = 24$, Camera 3 is set to zoom out in order to cover the predicted trajectory of the Marine.

After crossing the danger zone, the Marine stops and turns around to cover the other three Marines behind him. Since the new predicted trajectory of the Marine is shorter, there is no need for Camera 3 to be zoomed out. At $Time = 30$, Camera 3 is zoomed back in to cover the Marines at higher resolution.

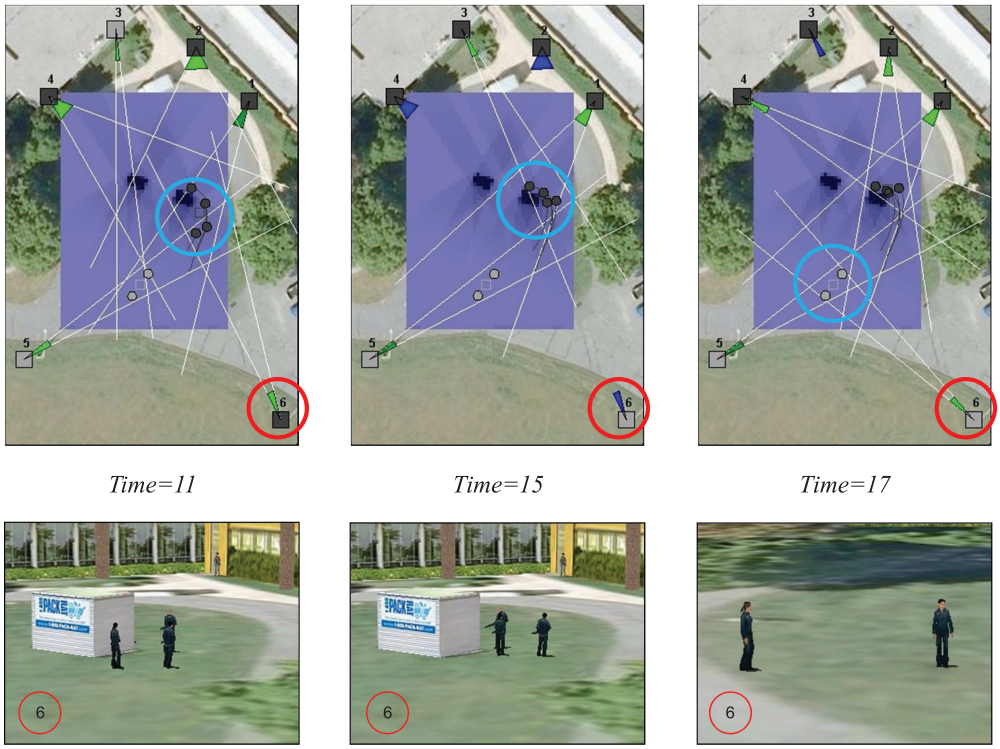

Fig. 6. Reacting to predicted occlusions. *Top row:* top-down views of cameras and exercise participants, overlaid on top of a satellite image and aggregated visibility map. *Bottom row:* simulated images for camera 6.

Figure 8 camera coverage of the exercise participants is the following.

At $Time = 42$, Cameras 5 and 6 cover the civilian and the sniper. As the civilian is heading toward the Marines for the cordon search, he exits the field of view of the cameras, but will soon be covered by the cameras covering the Marines. Cameras 5 and 6 can now zoom in to better cover the sniper. However, while either camera transitioned, it would leave the other camera covering the sniper by itself. Meanwhile, the Marines are covered by four cameras, one of which can help cover the sniper.

At $Time = 58$, Camera 2 has been reassigned from covering the Marines to covering the sniper, and Camera 6 can begin to zoom in.

At $Time = 65$, after Camera 6 has been zoomed in, Camera 5 can begin to zoom in as well.

Finally, at $Time = 68$, once both Cameras 5 and 6 are zoomed in and covering the sniper, Camera 2 is reassigned back to covering the Marines and the civilian.

One important result was that our planning heuristics can produce satisfactory results and serve as a fall-back when the search space is too large to be explored in realtime. We varied the number of candidate plans explored by the local planning component of our method between *heuristic* (simply using the worst-case, wide field of view plan generated heuristically by the global assignment, Algorithm 4), *selective* (heuristically generate and explore a small number of plans that break up long captures with wide fields of view into shorter captures with narrower fields of view) and *exhaustive* (explore all candidate plans). Selective planning was our default

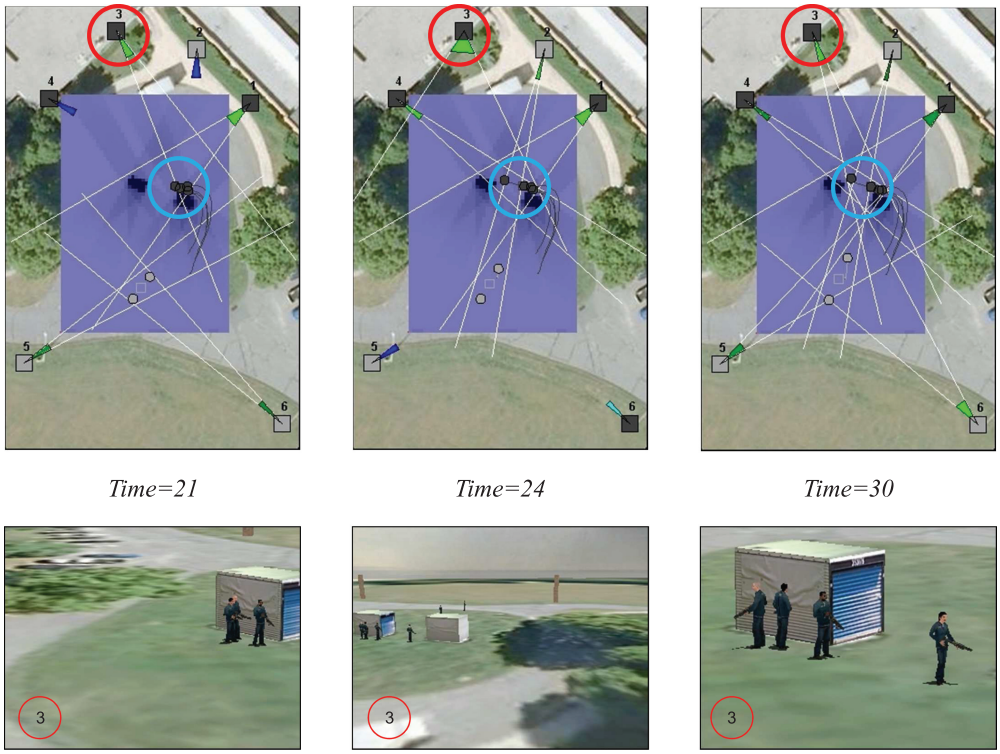

Fig. 7. Adapting to changes in predicted ROI motion. *Top row:* top-down views of cameras and exercise participants, overlaid on top of a satellite image and aggregated visibility map. *Bottom row:* simulated images for Camera 3.

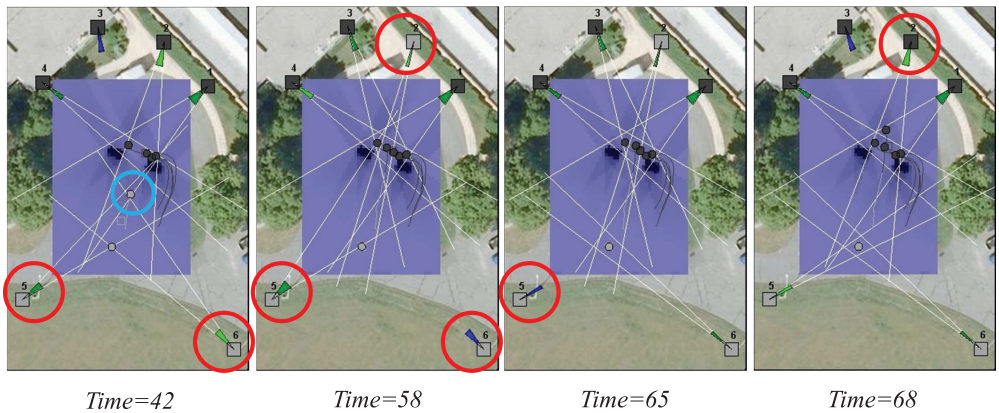

Fig. 8. Coordinating transitions. Top-down views of cameras and exercise participants, overlaid on top of a satellite image and aggregated visibility map.

exploration method, and it resulted in real-time performance. We designed heuristic planning as a fallback when the time budget for running in real time was close to exhausted, but we never had to resort to it in our experiments. Exhaustive exploration was performed offline when tractable—151 out of 170 planning cycles. Figure 9

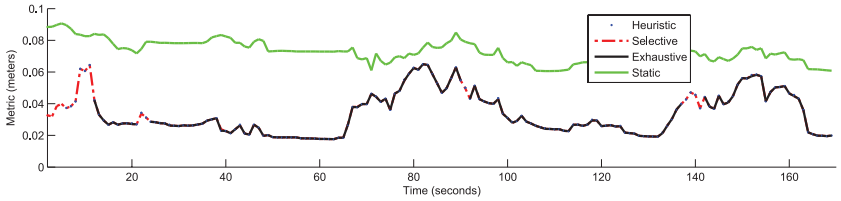

Fig. 9. Metric values over time for various local exploration methods.

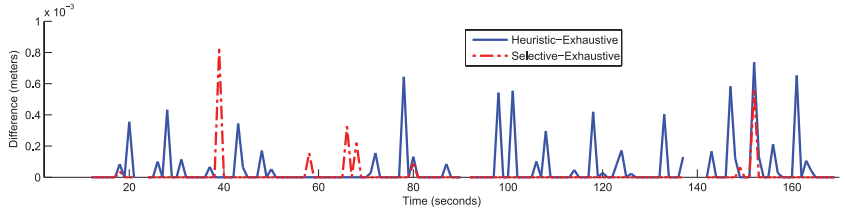

Fig. 10. Differences in metric values over time.

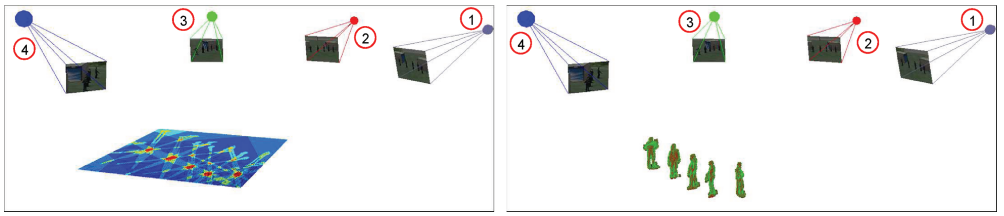

Fig. 11. 3D reconstruction results using method from [Guan 2010]. *Left:* rendering of a single slice of the reconstructed volume showing occupancy probabilities. *Right:* rendering of 3D reconstruction results obtained by thresholding. Also shown are the poses and images of the cameras involved.

shows the results, including the metric values when using static cameras, shown for comparison.

As expected, the metric values in the case of static cameras are significantly higher, but part of the price paid for the lower resolution is offset thanks to the fact that there are no transitions. The three curves corresponding to varying the number of candidate plans explored are very close to each other, to the point of being indistinguishable at the scale in Figure 9. Figure 10 shows the differences between the three curves at a larger scale. The fallback heuristic arrived at the same result as the exhaustive search for 109 out of 151 cycles, or 72% of the time, and the default selective search did so for 143 cycles, or 94% of the time. For the remaining times when the results were different, the average difference in the metric values was $0.01m$ for the fallback heuristic and $0.0087m$ for the default selective search.

The structure of the performance metric for this experiment, in terms of components such as the surrogate model and the aggregation function used, is suitable for volumetric 3D reconstruction approaches such as the one by Guan [2010]. Figure 11 shows reconstruction results obtained using simulated images from four cameras captured at $Time = 56$, during the cordon search. We generated perfect segmentations by providing the reconstruction approach with images taken with and without the exercise participants.

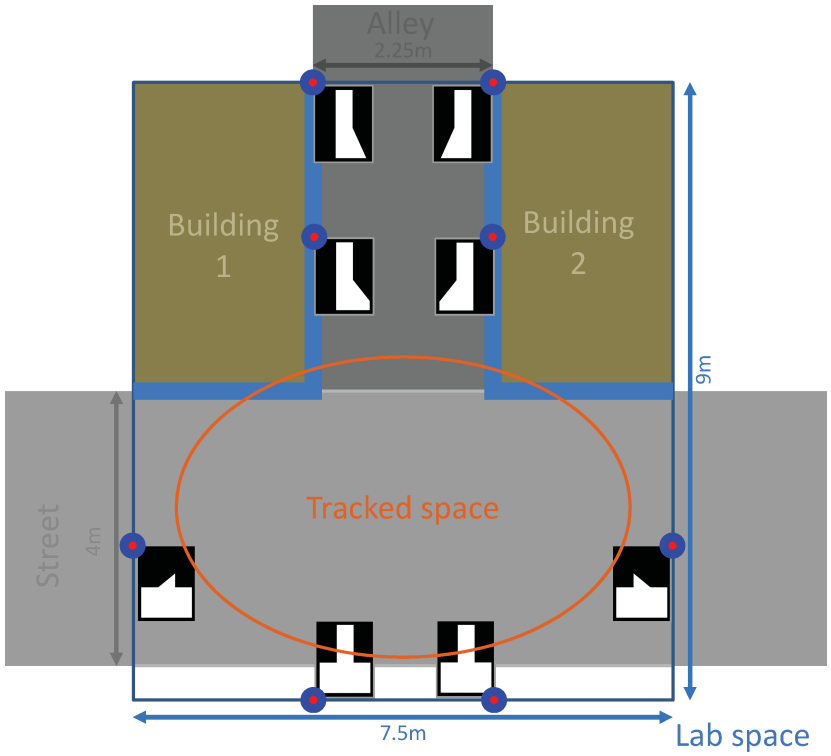

Fig. 12. The layout of the lab where we tested our camera control method. Camera positions are also highlighted, with visibility maps attached.

6.2. Laboratory-Based Experimental Results

6.2.1. Experiment Setup. We also applied our camera control method in a laboratory setup at UNC, using eight Axis 233D PTZ cameras. The environment was a $7.5m \times 9m$ room, shown in Figure 12. It contained part of a street and an alley between two buildings. Eight cameras were placed on the ceiling, $7m$ above the floor. Visibility maps were constructed manually for each camera, under the assumption that the building walls were tall enough to occlude everything behind them. The visibility threshold used to compute occlusions was 0.75.

The primary objective of this experiment was to verify our camera control method using real cameras. Consequently, we did not perform image processing to track the participants. An external tracker (NaturalPoint *OptiTrack*) was used to provide trajectories in the area outlined in orange in Figure 12, and trajectories were extrapolated using the last known orientation and speed when participants exited the tracked area. During the exercise, participant positions were captured by the tracker using head-worn tracker targets. Similarly to the experiments in Section 6.1, we used the resulting 3D points to generate simulated 2D measurements in each camera image and also incorporated them directly into the metric computation as 3D "GPS" measurements.

We had four members of the United States Marines Corp perform a training exercise similar to the one in Section 6.1. The exercise scenario was for the Marines to patrol the street, crossing the danger zone between the two buildings, and to cordon and search a civilian that was heading down the alley to join two other civilians who were waiting.

Fig. 13. An overview image taken during the training exercise. The tracked area is marked with white tape. Building walls were simulated using cloth attached to waist-high posts.

Figure 13 shows all seven participants in an overview image taken during the exercise when the civilian was being searched.

6.2.2. Camera Calibration. In our simulated experiments, the virtual cameras did not need to be calibrated: the pan, tilt and zoom setting values computed by our method using Algorithm 8 could be directly applied to them using nothing more than geometric transforms between the world and camera coordinate systems. In particular, panning the virtual cameras meant simply a rotation around the world z axis, and zooming only meant computing the zoom factor corresponding to a particular field of view. When using real cameras, this is no longer the case. The Axis 233D cameras are fairly sophisticated in that they accept pan and tilt values in degrees, and apply these values with great precision (repeatability). Experiments confirmed that the camera-reported pan and tilt setting values were accurate. Repeated geometric calibrations performed using the method in Zhang [1999] at various levels of zoom confirmed that reported zoom values were precise as well. Camera-reported zoom values were not simple $\times 1, \times 2, \ldots, \times n$ multipliers, but a straightforward linear transform applied to them yielded such values, and we found the transform to be repeatable across our six cameras.

The remaining problem was that the cameras' pan and tilt rotations were around arbitrary axes which could not be manually aligned with the corresponding world axes. We devised a procedure to calibrate the cameras by computing the rotation matrix between the camera and world coordinate systems. We first performed a geometric calibration to determine each camera's extrinsics, in particular its center of projection (COP) in world coordinates. We manually aimed each camera at a number of 3D points with known world coordinates, and recorded the camera-reported pan and tilt values, which we then mapped to a set of 3D points on an unit sphere centered at the camera's COP. We intersected the lines from each world 3D point to the COP with the same unit sphere, obtaining a second set of 3D points. The rotation matrix we needed was obtained by taking the two sets of 3D points on the unit sphere and computing the rotation around the COP required to align them.

This rotation matrix and the linear transform for the zoom values rendered the Axis 233D cameras as easy to control as the virtual cameras used in the experiments in Section 6.1.

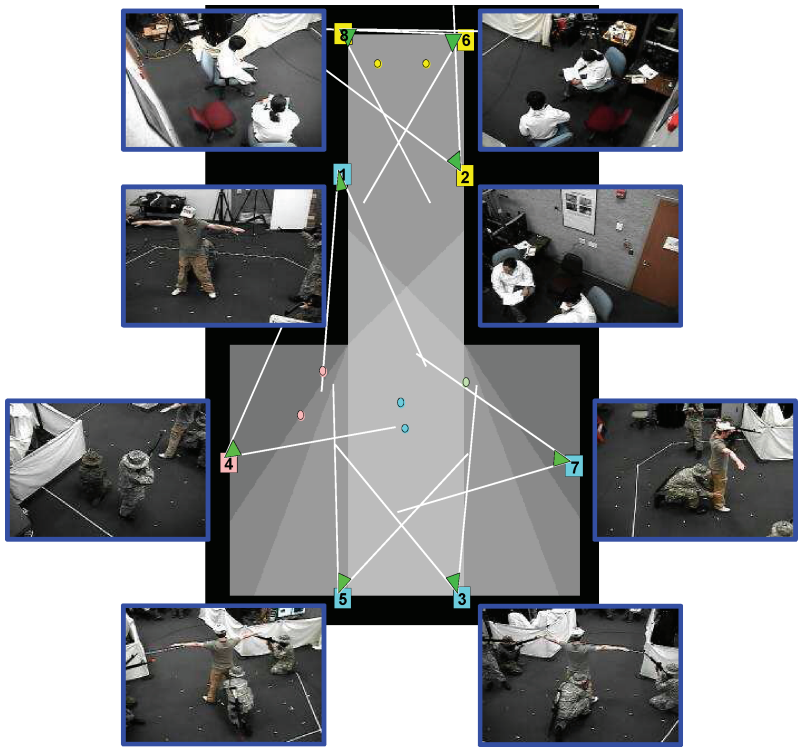

Fig. 14. Example camera images captured during the exercise, overlaid on top of a composite visibility map.

6.2.3. Results. While unable to run comparisons between results obtained using varying parameters like we did in Section 6.1, we performed multiple runs of the exercise scenario. Figure 14 shows example camera images taken during the exercise at the same time as the overview image in Figure 13, when the civilian was being searched. The images are overlaid on top of a composite visibility map, and attached to the icons representing the cameras that captured them. Participants are grouped into agencies, shown as circles with varying colors. Cameras assigned to a particular agency have their base represented as a square of the same color. Camera orientations are shown as triangles, and camera fields of view are shown as white lines.

We consistently observed desirable camera behaviors, including the ones summarized here.

(1) As soon as new participants entered the scene and were detected by the tracker, they were added to agencies and cameras were reallocated to cover them.
(2) Continuous agency coverage was maintained by appropriately adjusting the cameras' pan, tilt and zoom settings. Adjustments were done in a staggered fashion between cameras covering the same agency, such that the time intervals when the group was not being captured by as many cameras as possible were minimized.
(3) When an agency split, cameras were reallocated to ensure continuous coverage of the resulting agencies.
(4) Cameras were switched between agencies when their contribution to the coverage of an agency was better than their contribution to the agency they were currently covering.

(5) Continuous ROI coverage was maintained even when ROIs moved quickly or over relatively large distances by anticipating their trajectory and preemptively adjusting the cameras' pan, tilt and zoom settings.

7. CONCLUSIONS AND FUTURE WORK

We presented a new online camera control approach that treats camera control as an optimization problem. For the objective function, we developed a versatile performance metric that can incorporate both performance factors and application requirements. To reduce the size of the search space and arrive at an implementation that runs in real time, our camera control method breaks down the optimization problem into subproblems. We first employ a proximity-based minimal change heuristic to decompose the problem into subproblems and a greedy heuristic to assign cameras to subproblems based on evaluating candidate plans. We then solve each subproblem independently, generating and evaluating candidate plans as time allows.

We have discussed how our approach can be easily adapted to include requirements from various computer vision tasks. We demonstrated our approach in simulated and laboratory experiments, during which we have shown cameras exhibiting behaviors that facilitate the application of computer vision algorithms to the images they captured.

We plan on working on replacing the simulated camera measurements obtained from the external tracker with actual measurements from image tracking. We are especially interested in the situation where image tracking is to be performed with the same cameras that are controlled by our approach, which requires the camera control to balance the requirements of the computer vision application with the requirements of image tracking. We are also studying the impact of imprecise geometric calibration and camera settings precision and accuracy on our control method.

We are working on using knowledge about the environment to correct erroneous trajectory predictions. For example, if a Kalman filter-based trajectory prediction has a target passing through a wall, the prediction could be corrected to have the target stop in front of the wall or walk along it instead.

Our current implementation uses a proximity heuristic for grouping ROIs into agencies, and a greedy heuristic for assigning camera agents to agencies. We chose our heuristics after a careful analysis of the computational complexity of an exhaustive search and using our metric to evaluate alternatives, with the overarching goal of reducing search space size while still providing a reasonable solution. A thorough analysis of the impact of these heuristics on the solution, as well as a comparison with other approaches, are needed to fully gauge their effectiveness. It would be interesting to explore other clustering algorithms and evaluate the performance of our method when using them. A formal method such as linear programming could be used for assigning cameras to agencies, when feasible, and our heuristic could be run only when such methods fail. A promising direction to explore in clustering would be to allow overlaps, i.e. to allow an ROI to be a member of multiple agencies. Additionally, the current greedy assignment scheme exclusively assigns an agent to an agency. There may be situations when, with a small change in its settings, a camera could cover the ROIs in another agency nearby. In such situations, it may be beneficial to explore sharing an agent among multiple agencies. Sharing agents would require a protocol for agencies to cooperate when computing the camera parameters for agents that are shared with other agencies.

The main strategy our approach employs to reduce the search space size is decomposing the problem into subproblems and solving the subproblems at the level of each agency. Our current implementation solves the subproblems sequentially, in a single thread. We are working on a parallel implementation that would allow us to evaluate more candidate plans, increasing the chance that the provided solution is the global

optimum. Another possible research direction is decentralizing our approach further by having "smart" cameras locally estimate their own contribution to each ROI or agency. In the current implementation, the global assignment process is centralized. Distributing the assignment process by making it collaborative is likely to reduce computation loads in large camera networks, possibly at the expense of increased bandwidth required for collaboration, as well as possible additional system latency and weaker optimal performance guarantees.

REFERENCES

B. D. Allen. 2007. Hardware design optimization for human motion tracking systems. Ph.D. thesis, University of North Carolina at Chapel Hill.

B. D. Allen and G. Welch. 2005. A general method for comparing the expected performance of tracking and motion capture systems. In *Proceedings of the 12th ACM Symposium on Virtual Reality Software and Technology*. ACM, 201–210.

A. D. Bagdanov, A. Del Bimbo, and F. Pernici. 2005. Acquisition of high-resolution images through online saccade sequence planning. In *Proceedings of the 3rd ACM International Workshop on Video Surveillance & Sensor Networks*. 121–130.

A. Bakhtari, M. D. Naish, M. Eskandari, E. A. Croft, and B. Benhabib. 2006. Active-vision-based multisensor surveillance-an implementation. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* 36, 5, 668–680.

R. Bodor, P. Schrater, and N. Papanikolopoulos. 2005. Multi-camera positioning to optimize task observability. In *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance*. 552–557.

C. Broaddus, T. Germano, N. Vandervalk, A. Divakaran, S. Wu, and H. Sawhney. 2009. Activision: Active collaborative tracking for multiple ptz cameras. In *Proceedings of SPIE: Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*. Vol. 7345.

M. Casper. 2007. EventScripts Python. http://python.eventscripts.com.

C. B. Chang and J. A. Tabaczyinski. 1984. Application of state estimation to target tracking. *IEEE Trans. Autom. Control* 29, 2, 98–109.

X. Chen. 2002. Designing multi-camera tracking systems for scalability and efficient resource allocation. Ph.D. thesis, Stanford University.

A. K. R. Chowdhury and R. Chellappa. 2004. An information theoretic criterion for evaluating the quality of 3D reconstructions from video. *IEEE Trans. Image Process.* 13, 7, 960–973.

R. T. Collins, A. J. Lipton, et al. 2000. A system for video surveillance and monitoring. Tech. rep. CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

C. J. Costello, C. P. Diehl, A. Banerjee, and H. Fisher. 2004. Scheduling an active camera to observe people. In *Proceedings of the ACM 2nd International Workshop on Video Surveillance & Sensor Networks*. 39–45.

J. Davis. 2002. Mixed scale motion recovery. Ph.D. thesis, Stanford University.

A. J. Davison and D. W. Murray. 2002. Simultaneous localisation and map-building using active vision. *IEEE Trans. Pattern Anal. Mach. Intell.*

A. Del Bimbo and F. Pernici. 2005. Towards on-line saccade planning for high-resolution image sensing. *Pattern Recognit. Lett.* 27, 15, 1826–1834.

J. Denzler and C. Brown. 2002. An information theoretic approach to optimal sensor data selection for state estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 2, 145–157.

J. Denzler, C. M. Brown, and H. Niemann. 2001. Optimal camera parameter selection for state estimation with applications in object recognition. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*. Lecture Notes in Computer Science, vol. 2191, 305.

J. Denzler and M. Zobel. 2001. On optimal observation models for kalman filter based tracking approaches. Tech. rep., Lehrstuhl für Mustererkennung, Universität Erlangen-Nürnberg.

J. Denzler, M. Zobel, and H. Niemann. 2002. On optimal camera parameter selection in kalman filter based object tracking. In *Proceedings of the 24th DAGM Symposium on Pattern Recognition*. 17–25.

J. Denzler, M. Zobel, and H. Niemann. 2003. Information theoretic focal length selection for real-time active 3-d object tracking. In *Proceedings of the International Conference on Computer Vision*. Vol. 1, 400–407.

B. Deutsch, H. Niemann, and J. Denzler. 2005. Multi-step active object tracking with entropy based optimal actions using the sequential kalman filter. In *Proceedings of the IEEE International Conference on Image Processing*. Vol. 3, 105–108.

B. Deutsch, S. Wenhardt, and H. Niemann. 2006. Multi-step multi-camera view planning for real-time visual object tracking. In *Proceedings of the 26th DAGM-Symposium on Pattern Recognition.* Lecture Notes in Computer Science, vol. 4174, 536–545.

B. Deutsch, M. Zobel, J. Denzler, and H. Niemann. 2004. Multi-step entropy based sensor control for visual object tracking. In *Proceedings of the 28th DAGM-Symposium on Pattern Recognition.* Lecture Notes in Computer Science, vol. 3175, 359–366.

M. S. Grewal and A. P. Andrews. 1993. *Kalman Filtering Theory and Practice*. Information and System Sciences Series, Prentice Hall, Upper Saddle River, NJ.

L. Guan. 2010. Multi-view dynamic scene modeling. Ph.D. thesis, University of North Carolina at Chapel Hill.

A. Ilie and G. Welch. 2011. On-line control of active camera networks for computer vision tasks. In *Proceedings of the International Conference for Distributed Smart Cameras*.

A. Ilie, G. Welch, and M. Macenko. 2008. A stochastic quality metric for optimal control of active camera network configurations for 3D computer vision tasks. In *Proceedings of the Workshop on Multi-camera and Multimodal Sensor Fusion Algorithms and Applications*.

D. A. Ilie. 2010. On-line control of active camera networks. Ph.D. thesis, University of North Carolina at Chapel Hill.

T. Kailath, A. H. Sayed, and B. Hassibi. 2000. *Linear Estimation. Information and System Sciences Series*. Prentice Hall, Upper Saddle River, NJ.

N. Krahnstoever, M. Yeasin, and R. Sharma. 2001. Towards a unified framework for tracking and analysis of human motion. In *Proceedings of the IEEE Workshop on Detection and Recognition of Events in Video*.

N. Krahnstoever, T. Yu, S.-N. Lim, K. Patwardhan, and P. Tu. 2008. Collaborative real-time control of active cameras in large scale surveillance systems. In *Proceedings of the Workshop on Multi-Camera and Multi-Modal Sensor Fusion Algorithms and Applications*.

S.-N. Lim, A. Mittal, and L. Davis. 2005. Constructing task visibility intervals for a surveillance system. In *Proceedings of the 3rd ACM International Workshop on Video Surveillance & Sensor Networks*. 141–148.

S.-N. Lim, A. Mittal, and L. S. Davis. 2007. Task scheduling in large camera networks. In *Proceedings of the Asian Conference on Computer Vision*.

L. Marcenaro, F. Oberti, G. L. Foresti, and C. Regazzoni. 2001. Distributed architectures and logical-task decomposition in multimedia surveillance systems. *Proc. IEEE* 89, 10, 1419–1440.

T. Matsuyama and N. Ukita. 2002. Real-time multitarget tracking by a cooperative distributed vision system. *Proc. IEEE.* 90. 1137–1150.

A. Mittal and L. Davis. 2004. Visibility analysis and sensor planning in dynamic environments. In *Proceedings of the European Conference on Computer Vision*.

A. Mittal and L. S. Davis. 2008. A general method for sensor planning in multi-sensor systems: Extension to random occlusion. *Int. J. Comput. Vision* 76, 1, 31–52.

M. D. Naish, E. A. Croft, and B. Benhabib. 2001. Simulation-based sensing-system configuration for dynamic dispatching. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. 2964–2969.

M. D. Naish, E. A. Croft, and B. Benhabib. 2003. Coordinated dispatching of proximity sensors for the surveillance of manoeuvring targets. *Rob. Comput. Integr. Manuf.* 19, 3, 283–299.

P. Natarajan, T. N. Hoang, K. S. Low, and M. Kankanhalli. 2012. Decision-theoretic approach to maximizing observation of multiple targets in multi-camera surveillance. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*.

F. Oberti, G. Ferrari, and C. S. Regazzoni. 2001. Allocation strategies for distributed video surveillance networks. In *Proceedings of the International Conference on Image Processing*. Vol. 2. 415–418.

G. Olague and R. Mohr. 2002. Optimal camera placement for accurate reconstruction. *Pattern Recognit.* 35, 4, 927–944.

F. Qureshi and D. Terzopoulos. 2007. Surveillance in virtual reality: System design and multi-camera control. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 1–8.

F. Qureshi and D. Terzopoulos. 2005a. Surveillance camera scheduling: a virtual vision approach. In *Proceedings of the 3rd ACM International Workshop on Video Surveillance & Sensor Networks*. ACM Pres, 131–140.

F. Qureshi and D. Terzopoulos. 2005b. Towards intelligent camera networks: A virtual vision approach. In *Proceedings of the 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*. 177–184.

G. S. Ram, K. Ramakrishnan, P. Atrey, V. K. Singh, and M. S. Kankanhalli. 2006. A design methodology for selection and placement of sensors in multimedia surveillance systems. In *Proceedings of the 4th ACM International Workshop on Video Surveillance & Sensor Networks*.

P. Remagnino, A. Shihab, and G. Jones. 2003. Distributed intelligence for multi-camera visual surveillance. *Pattern Recognit.* 37, 4, 675–689.

E. Sommerlade and I. Reid. 2008a. Cooperative surveillance of multiple targets using mutual information. In *Proceedings of the Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*.

E. Sommerlade and I. Reid. 2008b. Information theoretic active scene exploration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

E. Sommerlade and I. Reid. 2008c. Information-theoretic decision making for exploration of dynamic scenes. In *Proceedings of the 5th International Workshop on Attention in Cognitive Systems*.

E. Sommerlade and I. Reid. 2010. Probabilistic surveillance with multiple active cameras. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

P.-N. Tan, M. Steinbach, and V. Kumar. 2005. *Introduction to Data Mining*. Addison-Wesley.

K. A. Tarabanis, R. Y. Tsai, and P. K. Allen. 1995. A survev of sensor planning in computer vision. *IEEE Trans. Rob. Autom.* 11, 1, 86–104.

G. R. Taylor, A. J. Chosak, and P. C. Brewer. 2007. OVVV: Using virtual worlds to design and evaluate surveillance systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1–8.

G. Welch, B. D. Allen, A. Ilie, and G. Bishop. 2007. Measurement sample time optimization for human motion tracking/capture systems. In *Proceedings of the Workshop on Trends and Issues in Tracking for Virtual Environments at the IEEE Virtual Reality Conference*. G. Zachmann, Ed.

G. Welch and G. Bishop. 2001. An introduction to the Kalman filter: SIGGRAPH 2001 course 8. In *Computer Graphics SIGGRAPH 2001 Course Pack.* ACM Press, Addison-Wesley Publishing Company, Los Angeles, CA.

J. J. Wu, R. Sharma, and T. S. Huang. 1998. Analysis of uncertainty bounds due to quantization for three-dimensional position estimation using multiple cameras. *Opt. Eng.* 37, 280–292.

S. Yous, N. Ukita, and M. Kidode. 2007. An assignment scheme to control multiple pan/tilt cameras for 3d video. *J. Multimedia* 2, 1, 10–19.

Z. Zhang. 1999. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the International Conference on Computer Vision*.

Received March 2012; revised October 2012; accepted January 2013