J Intell Robot Syst (2009) 54:567–593
DOI 10.1007/s10846-008-9247-0

Active-Vision for the Autonomous Surveillance of Dynamic, Multi-Object Environments

Ardevan Bakhtari · Matthew Mackay · Beno Benhabib

Received: 13 February 2007 / Accepted: 9 June 2008 / Published online: 22 July 2008

Abstract This paper presents a novel method for active-vision-based sensing-system reconfiguration for the autonomous surveillance of an *object-of-interest* as it travels through a multi-object dynamic workspace with an a priori unknown trajectory. Several approaches have been previously proposed to address the problem of sensor selection and control. However, these have primarily relied on off-line planning methods and rarely utilized on-line planning to compensate for unexpected variations in a target's trajectory. The method proposed in this paper, on the other hand, uses a multi-agent system for on-line sensing-system reconfiguration, eliminating the need for any a priori knowledge of the target's trajectory. Thus, it is robust to unexpected variations in the environment. Simulations and experiments have shown that the use of dynamic sensors with the proposed on-line reconfiguration algorithm can tangibly improve the performance of an active-surveillance system.

Keywords Active vision · Surveillance · Sensor fusion · Multi-agent system

Mathematics Subject Classifications (2000) 1.0 · 10.1 · 103.0

A. Bakhtari · M. Mackay (✉) · B. Benhabib
Computer Integrated Manufacturing Laboratory,
Department of Mechanical and Industrial Engineering,
University of Toronto, 5 King's College Rd.,
Toronto, Ontario M5S 3G8, Canada
e-mail: mackay@mie.utoronto.ca

A. Bakhtari
e-mail: bakhtar@mie.utoronto.ca

B. Benhabib
e-mail: beno@mie.utoronto.ca

1 Introduction

The application of active sensing to surveillance problems has introduced *on-line sensing-system reconfiguration* (also, referred to as *sensor planning*) as a key research topic. Most recent work has focused on dynamic, multi-object environments, where the system can provide autonomous surveillance of an Object-of-Interest (OoI). Typically, these problems also include multiple obstacles (static or mobile) that may occlude the OoI. Thus, surveillance is defined herein as the data-acquisition and analysis process used for the recognition of the OoI or parameter estimation of the features of the OoI.

An introduction to the general area of surveillance is presented first below by examining past work on sensing-system reconfiguration for static and dynamic environments. In this context, the exact problem to be solved is defined, and our proposed solution is outlined in Section 2. Detailed simulation and experimental results validating the algorithm are presented in Sections 3 and 4, respectively.

1.1 Sensing-System Reconfiguration in Static Environments

Traditionally, sensing-system planning has been utilized to configure a set of sensors in a static surveillance environment. Such work has been categorized as either "generate-and-test" or "synthesis" [1]. In generate-and-test methods, sensor-placement plans are determined based on the task constraints by searching through a set of (discretized) sensing-system configurations. For example, in [2] a single robot moves a sensor to observe features on a stationary OoI. A discretized virtual sphere, created around the OoI, represents all the possible poses for the sensor. Only poses that are un-occluded and fit within the workspace of the robot are selected. Similarly, in [3], the sensing-system planner determines the minimum number of viewpoints (and, thus, sensor poses) to observe all features on an object. Herein, not only is the OoI's virtual sphere discretized, but, the surface of the OoI itself as well.

Synthesis methods determine sensing-system configurations by using the analytical relationship between task requirements and sensor parameters and, thus, are highly application specific. For example, in [4], the sensing-system planner synthesizes a region of viewpoints by first imposing a 3-D bound on the position of the camera by each of the task constraints (e.g., field of view and focus, resolution). The intersections of these bounds are considered to be regions of acceptable viewpoints. Similarly, in [5], the system automatically determines the viewing direction that allows the entire OoI to become visible while minimizing distortion in the image. The system works by taking points along the outer edge of the OoI and formulating uncertainties in sensor observations by data fusion and optimal sensor placement. Another example is given in [6], where optimal 2-D sensor placements are determined for a number of similar sensors. The algorithm positions sensors as close to the OoI as possible, while the orientations of the sensors are found through a closed-form solution that minimizes the area of the uncertainty ellipse associated with the sensors' observations. The system in [7] uses an off-line generate-and-test method with an on-line synthesis method to optimally place dissimilar sensors (range, intensity, and stereo cameras) for OoI inspection.

1.2 Sensing-System Reconfiguration in Dynamic Environments

1.2.1 Single-Object Environments

Recently, there has been interest in sensing-system reconfiguration in dynamic environments, [8–10]. Most current systems address this problem by utilizing methods developed for static environments. For example, the system proposed in [9] optimizes sensor configurations off-line by discretizing time and treating each time instant as a static case, utilizing the sensing-system reconfiguration method presented in [2]. This off-line approach requires the motion of the OoI to be known a priori with high accuracy. The system presented in [10] uses an off-line heuristics method to determine sensor motions in 2-D based on an a priori known OoI trajectory. An on-line controller is later used to readjust sensor motions to account for deviations in the actual OoI trajectory.

In contrast to the previous systems, the work proposed in [11] does not require a priori knowledge about the OoI's trajectory. The system discretizes the workspace into a number of sectors and, once the OoI enters a sector, the sensors assigned provide synchronous information about the OoI. The system in [12] utilizes multiple sensors through an agent-based sensing method, where each mobile sensor's path is independently determined through a triangulation method. The system in [13] also uses autonomous agents; however, unlike in [12], the agents negotiate to achieve the necessary level of coordination for accomplishing the given sensing task, while maximizing the amount of the target that can be observed at any given time. The system in [14] combines sensor-placement constraints with the shape and current pose of the OoI via a Bayesian network for task-specific sensing-system reconfiguration. The Bayesian network is reconstructed continuously to reflect changes in the pose of the target as determined by the active sensors. The derived sensing action maximizes the amount of target visible while minimizing the sensing cost (sensor movement).

1.2.2 Single-Target, Multi-Object Environments

A multi-object dynamic environment, while much more relevant to real-world applications, is considerably more complex than the previous examples. In a single-target, multi-object environment the system must perform sensing-system reconfiguration based on a single OoI trajectory and multiple obstacles. Examples of such systems were presented in [15] and [16]. Both works require the OoI and the surrounding environment to be model 3D polyhedrons, so that constraints (such as occlusions) can be determined at discretized time instants. Numerical optimization methods are used to determine sensor locations at each time instant. The methods presented in [17] and [18] use pre-determined constraints, such as occlusions, field of view, and travel limits, to dynamically plan the motion of the single robot-mounted camera. These sensing-system planners aim to avoid occlusions and maximize the camera's view of the target. Other systems, such as [19], use a combined algorithm, including closed-form solution to the sensor placement problem. In this work, SIFT (Scale Invariant Feature Transform) is used to detect the target object and to maximize the number of features in the field of view of the final sensor solution. Finally, other research has addressed the problem of shortest path planning. For example,

in [20], the authors present a genetic algorithm for optimal sensor placement, using the 'Christofides algorithm' for shortest path planning.

A detailed examination of the sensor-selection problem in multi-sensor systems was presented in [21]. There, the authors sought to minimize the error in estimating the position of a target using a generic sensor model and an approximation algorithm for sensor selection. It was determined that a relatively small subset of a larger number of sensors can reduce uncertainty in a sensing process effectively. Other issues, such as path planning for mobile robots (which can be viewed as active sensors), have also been examined in the context of multi-sensor systems [22, 23].

1.2.3 Multi-Target, Multi-Object Environments

In a multi-target, multi-object environment, sensing-system reconfiguration aims to maximize the number of targets that are observed while minimizing the uncertainty associated with the observations. Very little work has been done in this area, due to the inherent complexity of the problem.

This paper focuses on the problem of *single-target surveillance in multi-object environments*; and, thus, the problem of multi-target surveillance is beyond its scope.

1.3 Problem Definition

Surveillance was defined above as the observation of an OoI (or feature on an OoI), where one seeks to maximize visibility. More specifically, the sensing-system reconfiguration method must be able to cope with single-target (OoI), multi-object dynamic environments, where occlusions may be present and both the OoI and the obstacles may be static or moving at any time. In addition, the method should be general enough to be applied to a variety of applications with minimal adaptation and operate in an on-line mode. This paper proposes such a novel agent-based approach. Prior to a detailed description of our sensing-system reconfiguration method, several agent-based planning algorithms previously suggested in the literature are briefly discussed below.

1.3.1 Agent-Based Sensing-System Reconfiguration

Recently, a number of agent-based approaches have been proposed for the problem of real-time sensing-system planning. For example, in [24], the algorithm uses sensor agents to track multiple moving targets, where an agent is considered to be a Pan-Tilt-Zoom (PTZ) camera plus a dedicated computer. The agents scan the workspace for a target and, once one is detected, they share the OoI information. Each agent independently determines whether it should contribute to the surveillance of this target or search for a new target. In [25] and [26], multiple mobile sensors, modeled as separate agents, were used to detect and recognize targets. All agents start by searching for a target and, once one is detected, agents negotiate among themselves for assignment to the detected target. This algorithm, in contrast to that presented in [24], utilizes purely cooperative agents.[1] Performance is significantly

[1] Cooperative agents are those that work together to improve system performance rather than their own performance.

improved; however, complexity of the required conflict-management strategy is also increased.

In our work, an agent-based approach is also used for sensor selection and positioning in a multi-object environment. In contrast to the abovementioned systems, however, external virtual agents are used for conflict detection and management. The use of virtual agents improves global behavior of the multi-agent system and simplifies the conflict-management strategy. This can be accomplished while still maintaining near-optimal performance for object visibility. The virtual agents, as described later, are able to ensure this by providing a more logical division of the work within the system, better suiting the sensing-system reconfiguration process.

Finally, our system also differs from those described above in that the unused sensors are not utilized for target detection but are instead positioned in anticipation of future *service* requirements. This placement, as will be shown later, improves the likelihood that the unused sensors will be at poses with higher OoI visibility in the future. Thus, our algorithm follows a probabilistic framework, albeit a simple one. Overall, the goal is to maintain the quality of information for the task at hand while improving the responses to future OoI maneuvers.

2 Sensing-System Reconfiguration Strategy

The proposed sensing-system reconfiguration strategy is designed to provide estimates of an Object-of-Interest's (OoI) parameters at predetermined times along its trajectory. These predetermined times are referred to herein as *demand instants*, t_i. The spacing of demand instants is application-specific, but will generally coincide with acquisition instants for the sensor/camera. Care must be taken to ensure that the dynamic capabilities of each sensor are not overly restricted by a small demand instant spacing – some fractional multiple of the sensor acquisition rate may be used. Similarly, sufficient time to process a reasonable number of alternatives must be given. However, if the spacing is too large, response times will increase and stability will decrease. On the average, demand instant spacing will be uniform (equal spacing), with the period chosen a priori. However it would be possible for a system to adjust the spacing online, and to use uneven spacing, so long as the system is able to track processing/decision time accurately.

It is also assumed that the pose of the OoI at a particular demand instant is predicted from observations of the OoI motion, rather than known a priori. In general, the estimation of the OoI pose at a demand instant will change (and its corresponding uncertainty will diminish) as the prediction accuracy improves over time; however, the demand instant remains constant.

If the sensing system contains multiple redundant sensors, a subset of these may be sufficient to satisfy the sensing requirements of a demand instant. Namely, a sensor-fusion process does not need to combine the information from all of the sensors in the system. Instead, a subset of sensors, herein referred to as a *fusion subset*, may be selected to survey the OoI at a particular demand instant. This allows remaining sensors to be reconfigured in anticipation of future use. In this context, our previous work addressed this dispatching problem using heuristics and a blackboard approach [18]. In this paper, a novel agent-based approach is applied to the problem at hand. As will be shown, this approach offers an improved logical

division of the task at hand. The use of virtual agents is a natural extension, in that one attempts to 'describe' the overall behavior desired rather than directly solving the underlying dynamics. In order to allow for the generality of the method, the actual form of communication between the agents is not specified in the proposed general algorithm, as it does not need to be. Several alternatives will exist, depending on the physical implementation. In general, low latency, guaranteed delivery, and sufficient bandwidth are desirable properties in selecting a communication medium for this algorithm.

2.1 Quality of the Sensing Data

Integral to *sensor dispatching* is an estimate of the quality of data that each sensor can provide for the current demand instant and for the span of a rolling horizon of several demand instants. These estimates are used in our algorithm to select the specific sensors for inclusion in the fusion subset and determine their desired poses, given their current poses in the workspace and their motion capabilities.

A *visibility measure*, that is inversely proportional to the measurement uncertainty, is used as a quantitative measure. The model-based visibility measure provides a more robust basis for sensor selection than simple distance measures or a *line-of-sight* test. The visibility measure for the jth sensor servicing the ith demand instant is defined herein as,

$$v_{ji} \begin{cases} \dfrac{1}{\|\mathbf{R}\|} & If\ target\ is\ unoccluded \\ 0 & else \end{cases}, \tag{1}$$

where $\mathbf{R}$ is the covariance matrix associated with the sensor measurement.

For the cameras that are used in our experimental setup, $\mathbf{R}$ is a function of six variance parameters: three for the Cartesian position of the target OoI (x, y, z) and three for its orientation (n_x, n_y, n_z). Our variance-analysis experiments led to the conclusion that only two controlled sensor parameters significantly affect the measurement variances, Appendix A: the Euclidean camera-to-target distance, d, and the bearing of the camera with respect to the target, L:

$$d = \left\| \mathbf{o}_p - \mathbf{s}_p \right\|, \tag{2}$$

where $\mathbf{o}_p$ is the target position and $\mathbf{s}_p$ is the sensor position, and

$$\theta = cos^{-1} \left(\boldsymbol{o}_n \circ \boldsymbol{\tau} \right), \tag{3}$$

where $\boldsymbol{o}_n$ is the unit vector corresponding to the target orientation and $\boldsymbol{\tau}$ is the viewing direction of the sensor,

$$\boldsymbol{\tau} = \frac{\mathbf{o}_p - \mathbf{s}_p}{\left\| \mathbf{o}_p - \mathbf{s}_p \right\|}. \tag{4}$$

Thus, the visibility measure for a sensor over the span of a rolling horizon is defined as,

$$v_j = \sum_{i=1}^{m} a_i v_{ij,} \tag{5}$$

$$\sum_{i=1}^{m} a_i = 1, \tag{6}$$

where m is the number of demand instants in the rolling horizon and $0 \leq a_i \leq 1$ is the weight of the ith demand instant. The weight factor is constant for all the sensors and represents the uncertainty in the predictions of the future target poses. Although the weight factor a_i is user specified, and depends on the surveillance system at hand, in general itsvalue would decrease as predictions of target poses are made further into the future. The weights are normalized, as in Eq. 6. In order to choose these weights in a real-world application, one would perform controlled, repeatable experiments with the final system to determine the trade-off of future prediction necessary for best performance.

One possible way to ensure a minimum level of quality is to impose a constraint that *each target must be serviced with a combined visibility greater than a threshold*, v_{min}, *at every demand instant*. The actual value chosen would depend on workspace conditions, for example, the number of objects. Such a constraint can easily be imposed and monitored by the referee agent. It must be noted that this is just an example constraint, although it is representative of one used in an actual implementation.

2.2 Coordination Strategy

Dispatching can be accomplished using two complementary strategies: A *coordination strategy* to determine the subset of sensors to be used, and a *positioning strategy* to select the optimal pose of each sensor for any demand point being serviced.

The proposed agent-based system consists of multiple *sensor* agents, a *referee* agent, and a *judge* agent. Each sensor agent tries to maximize its own performance over the span of the rolling horizon. Although not directly controlled by a centralized controller, the sensor agents must abide the external rules of the environment monitored and enforced by two virtual agents. The rules are set to ensure the collective behavior of the sensor agents exhibits the desired system behavior.

2.2.1 Sensor Agents

The sensor agent is responsible for choosing the demand instants that the associated sensor will service and for determining its best poses in terms of the sensor's performance metric (i.e., visibility) over the span of the rolling horizon. If a demand instant is not serviced, the sensor would have zero visibility for that demand instant, however, it would allow more time for the sensor to maneuver for the next demand instant.

Each sensor agent searches through all possible combinations using a *depth-first approach* (e.g., [1, 1, 1] is a combination referring to servicing all demand instants

in a 3-demand-instant horizon). The total search space for a sensor agent is 2^m, where m is the number of demand-instants in the rolling horizon. However, certain combinations would always have a lower visibility than others and, therefore, might not have to be searched. For example, if combination [1, 1, 1] is achievable (not occluded), then, the combination [1, 1, 0] would not have any advantage for that sensor since it would have a lower visibility; thus, it does not have to be 'searched.'

At each combination searched, the sensor agent determines the best achievable poses to service the selected demand instants through the positioning strategy outlined in Section 2.3 below. Initial sensor poses (as well as the number and type of sensors, and their off-line placement) can be determined using traditional off-line methods, such as those summarized in [1]. Using these poses and the OoI's predicted locations, the sensor agent determines the expected, achievable visibility for each combination. The sensor agent, then, evaluates the combinations searched, to determine acceptable solutions. Acceptable solutions are constrained by the following two *internal* rules:

1. A demand instant cannot be serviced if it is occluded.
2. Combination [0, 0, 0, …, 0], representing a sensor not being assigned to any demand instant, is only considered if all other combinations are occluded.[2]

Next, the sensor agent ranks all acceptable combinations in a descending order of combined visibilities. The rth ranked acceptable solution for the jth sensor is denoted herein as $\boldsymbol{S}_{jr}$. The sensor agent sends the first ranked acceptable solution, $\boldsymbol{S}_{j1}$, to the referee agent. During this process, processing time is strictly tracked, and the search is terminated early, if necessary.

It is important to note that this ranking demonstrates the framework for the positioning of unassigned sensors. Let us consider the same example as above, with a 3-demand-instant horizon. Let us assume that a [1, 0, 1] combination is chosen and is achievable, which means that the sensor will service the first and third demand instants, but not the second. After achieving the pose necessary to service the current (first in the rolling horizon) demand instant, the sensor is unassigned for the next demand instant. Therefore, the sensor agent would 'look ahead' in the horizon and determine its next assigned demand instant, the third in this case. For this instant, an expected OoI position, $\boldsymbol{E}_x$, can be determined from the motion model. The predicted position can be determined internally (independently) by the sensor agent, or may be provided by an external tracking/prediction agent. The motion path during the unassigned demand instant would, therefore, be chosen as the one that will produce the maximum object visibility given the expected OoI position. Thus, for unused demand instants the system would seek to improve the potential (expected) visibility. Namely, the uncertainty would be directly proportional to the uncertainty of the underlying motion model. One can note that, this process is examining the expected rather than the actual visibility. If the assumption that expected visibility is equal to actual visibility at a future demand instant holds (i.e., the underlying motion model is accurate), then, we would obtain optimal behavior. In most cases, however, one would need to accept that the solution is actually a near-optimal one.

[2]The [0, 0, …, 0] combination allows the sensor to simply always follow the target so that it may be used in the future.

2.2.2 Referee Agent

The referee agent monitors the 'intentions' of the sensor agents and ensures that no *external* rules are violated. These external rules would depend on the surveillance task at hand and are, thus, user specified. If the referee agent detects a violation of the external rules, it initiates the judge agent in order to resolve the conflict. It is important to note the logical division that takes place. By abstracting and encapsulating the rules from the judge agent, we greatly reduce its complexity. More importantly, the generality of the algorithm is increased. The judge agent can be specified without a complete declaration of the external rules. Furthermore, once a system using this method is in place, its modifiability is greatly enhanced if the external rules of the system are not intertwined with those of the judge. A small change in the external rules does not necessitate a complete revision of the judge implementation. Obviously, this division is primarily a logical one; it does not mean that the agents need to be physically separate, thus, the communication overhead need not be overly affected.

As a simple example, in this work, the following external rule is defined to ensure the sensors are well distributed among the demand instants of the rolling horizon:

- *At least one sensor must be assigned to each demand instant.*

This was the only rule applied to the experiments presented in Sections 3 and 4. This rule shows the use of the referee agent without unnecessarily complicating the examples. A real-world application could contain a significantly more demanding set of rules, which are application specific. For example, a multi-target application might impose a rule that each unrecognized target be serviced by at least one sensor at every demand instant, or that every target must have a combined visibility greater than some minimum at every instant.

2.2.3 Judge Agent

Upon initiation, the judge agent sends a command to each sensor agent requesting the sensor agents' second-ranked acceptable solutions, $\boldsymbol{S}_{j2}$. Along with these alternate solutions, each sensor agent also sends the corresponding expected visibilities. The judge agent uses a depth-first approach to search through all possible permutations of 1st and 2nd ranked solutions for combinations that would resolve the conflict (an example combination of first and second ranked solutions of four sensor agents is $[S_{11}\ S_{21}\ S_{32}\ S_{41}]$). It is important to note that the Iterative Deepening Depth-First Search (IDDFS) [27] used in our work does not guarantee optimal computation time, nor an optimal visibility solution. However, it does offer excellent performance in terms of the trade-off between processing time and completeness of the search. The IDDFS is commonly used as a search method for this problem type due to the large potential search space and unknown search depth, which must be searched within a given time-limit.

During this search, the judge agent selects the combination with the highest visibility and informs the sensor agents of its decision. If no acceptable combination is found, the judge agent increases the depth of the search space by requesting the sensors agents' third-ranked solutions, as in Fig. 1. This process is repeated until an acceptable combination is found or the allowable search time has elapsed. In the event that no acceptable combination is found within the allowable search time, the

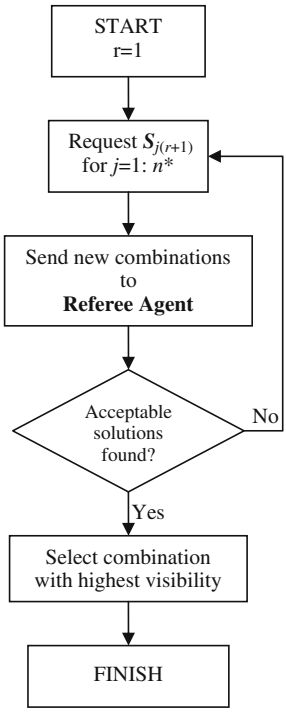

Fig. 1 Flowchart for judge agent's search for an acceptable solution

1st-ranked solutions (initial sensor agents' intentions) are used. This ensures that the system is not in a virtual deadlock if no solution exists that would satisfy the external rules.

If one considers all agents in a system to be peers, then, virtual deadlock is possible. Specifically, each agent may independently propose a first-ranked solution that is optimal for that agent, but, one is that sub-optimal in terms of the overall system performance. Since all agents are peers, no clear conflict resolution would be readily available and group consensus would have to be used to resolve it. This would entail extensive communications between the agents, unless a-priori set rules are used for resolution. This case, however, often results in a very poor resolution. This problem structure is very common (for example, computer-bus management) and the solution is almost invariably a master-slave or hub architecture, hence, the use of the judge agent in our work. The transmission of the ranked solutions, generally, incurs significantly less overhead than a completely distributed resolution, and a centralized set of rules improves the quality parameters of the system (maintainability, proper encapsulation, etc).

2.3 Positioning Strategy

In a single-target, multi-object environment the positioning strategy is not only based on the trajectory of the OoI (i.e., the target) but also on that of other *objects not of interest*. The first step in determining the best achievable pose is to determine the occluded regions in the workspace. In order to accomplish this, the pose of each

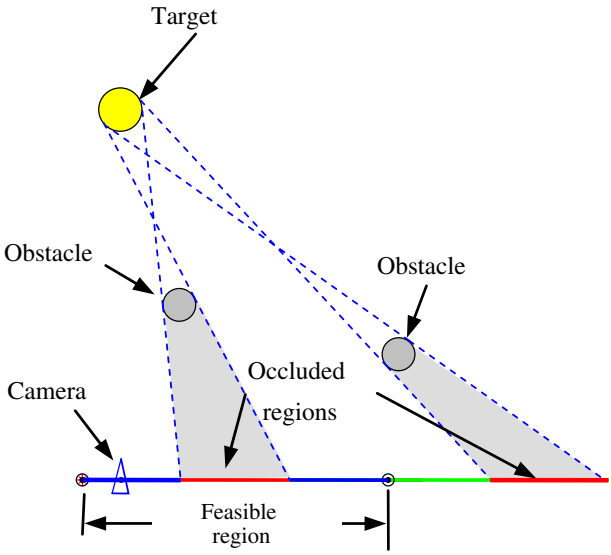

Fig. 2 An example of occluded regions of a sensor's workspace

object (OoI or obstacle) is predicted for the demand instant. Next, each object is modeled as a single geometric primitive (e.g., a sphere or cylinder), rather than as a collection of 3D polyhedra, in order to decrease computational complexity. Occluded regions of a sensor's workspace are determined by modeling the OoI as a light source and calculating the geometric shadow volumes cast by the obstacles, Fig. 2. The algorithm, subsequently, determines the region of the workspace that the sensor can travel to before the target reaches the demand instant, referred to herein as *feasible region*. This region is defined by the sensors' dynamic motion capabilities such as maximum velocity, $v_{\max}$, acceleration, a, as well as time to next demand instant, dt. For a sensor with one degree-of-freedom of translational mobility (along the x axis) the feasibility region, x_{feasible}, is defined as,

$$x_1 \leq x_{\text{feasible}} \leq x_r. \tag{7}$$

In Eq. 6, x_r is the right limit defined by

$$x_r = v_o\,(\mathrm{dt}a_1) + \frac{1}{2}a\,(\mathrm{dt}a_1)^2 + v_{\max}\,(\mathrm{dt}c_1) + v_o\,(\mathrm{dt}s_1) + \frac{1}{2}a\,(\mathrm{dt}s_1)^2\,, \tag{8}$$

where v_o is the current sensor velocity, $\mathrm{dt}a_1$, $\mathrm{dt}s_1$, and $\mathrm{dt}c_1$ are the times the sensor travels while accelerating, decelerating, and with a constant velocity, respectively, in order to get to the right travel limit, each defined by

$$\mathrm{dt}a_1 = \begin{cases} \dfrac{v_{\max} - v_o}{a} & if \quad \left(\dfrac{2v_{\max} - v_o}{a}\right) < dt \\ \dfrac{1}{2}\left(dt - \dfrac{v_o}{a}\right) & \text{else} \end{cases}, \tag{9}$$

$$\mathrm{dt}s_1 = \begin{cases} \dfrac{v_{\max}}{a} & if \quad \left(\dfrac{2v_{\max} - v_o}{a}\right) < dt \\ \dfrac{1}{2}\left(\mathrm{dt} + \dfrac{v_o}{a}\right) & \text{else} \end{cases}, \text{and} \tag{10}$$

$$\mathrm{dt}c_1 = \mathrm{dt} - (\mathrm{dt}a_1 + \mathrm{dt}s_1), \tag{11}$$

The x_1 in Eq. 6, is the left limit defined by

$$x_l = v_o(\mathrm{dt}a_2) - \frac{1}{2}a(\mathrm{dt}a_2)^2 - v_{\max}(\mathrm{dt}c_2) + v_o(\mathrm{dt}s_2) - \frac{1}{2}a(\mathrm{dt}s_2)^2, \tag{12}$$

where $\mathrm{dt}a_2$, $\mathrm{dt}s_2$, and $\mathrm{dt}c_2$ are the times the sensor travels while accelerating, decelerating, and with a constant velocity, respectively, in order to get to the left travel limit, each defined by

$$\mathrm{dt}a_2 = \begin{cases} \dfrac{v_{\max} + v_o}{a} & \text{if} \quad \left(\dfrac{2v_{\max} + v_o}{a}\right) < \mathrm{dt} \\ \dfrac{1}{2}\left(\mathrm{dt} + \dfrac{v_o}{a}\right) & \text{else} \end{cases}, \tag{13}$$

$$\mathrm{dt}s_2 = \begin{cases} \dfrac{v_{\max}}{a} & \text{if} \quad \left(\dfrac{2v_{\max} + v_o}{a}\right) < \mathrm{dt} \\ \dfrac{1}{2}\left(\mathrm{dt} - \dfrac{v_o}{a}\right) & \text{else} \end{cases}, \text{and} \tag{14}$$

$$\mathrm{dt}c_2 = \mathrm{dt} - (\mathrm{dt}a_2 + \mathrm{dt}s_2). \tag{15}$$

It should be noted that for sake of simplicity, the limits of the workspace have not been included in the equations above.

Lastly, the algorithm determines a sensor pose that would yield maximum visibility, which is both feasible and un-occluded (i.e., acceptable regions). This is done by discretizing the acceptable region into a pre-specified number of positions. The best pose is selected by evaluating the visibility metric at each discrete position. The number of discrete positions is a trade-off between resolution of the result and processing time - the exact choice can be determined through controlled experiments. In general, the number of discrete poses considered can be increased (thus increasing resolution of the result) until just before the controller cannot evaluate all poses (with some margin for variation in processing time). The coordination and positioning of each sensor is repeated continuously as new information, regarding the environment, becomes available. This ensures that new and more accurate target-pose predictions are utilized. Furthermore, as time approaches the demand instant (i.e., dt→0), the size of the acceptable region diminishes and, therefore, it would be more densely discretized resulting in more accurate sensor-pose determination.

Table 1 Summary of parameters for three systems

	Static system	Slow dynamic	Fast dynamic
Translational velocity	0 mm/s	2.5 mm/s	15 mm/s
Translational acceleration	0 mm/s^2	0 mm/s^2	30 mm/s^2
Rotational velocity	0 rad/s	0.1 rad/s	0.3 rad/s
Target velocity	5 mm/s	5 mm/s	5 mm/s
Obstacle velocity	7 mm/s	7 mm/s	7 mm/s

3 A Simulated Example

In order to demonstrate the proposed dispatching algorithm, a simulated example is briefly discussed in this section. The performance of the surveillance task is measured using three systems: static, slow dynamic, and fast dynamic. Parameters for these systems are summarized in Table 1. Each system has four cameras. The target was modeled as a 25 mm circle, and the obstacles as 60 mm circles. The OoI trajectories and the initial positions for each are shown in Fig. 3 (not to scale). The algorithm proposed in Section 2 was implemented, with individual agents represented as distinct software programs on a single physical system. Communication was thus simplified, using direct communication through message passing and temporary files.

In our simulations, six demand instants were considered. At each demand instant, the visibility attainable by each camera is calculated based on the camera model given in Appendix A. The visibilities of each camera for the static, slow, and fast systems are shown in Figs. 4, 5, 6, respectively, and the fused visibilities (the sum of the visibility metric for each sensor in the system) in Fig. 7. One can see that for all except Instant 6 (which shows a slight decrease in the slow system only), the fused visibility of the sensing system (fast or slow) is higher than traditional static placement. Similarly, the fast system (for all instants) shows the highest fused metric values of any of the three trials, as expected. As established in the literature review,

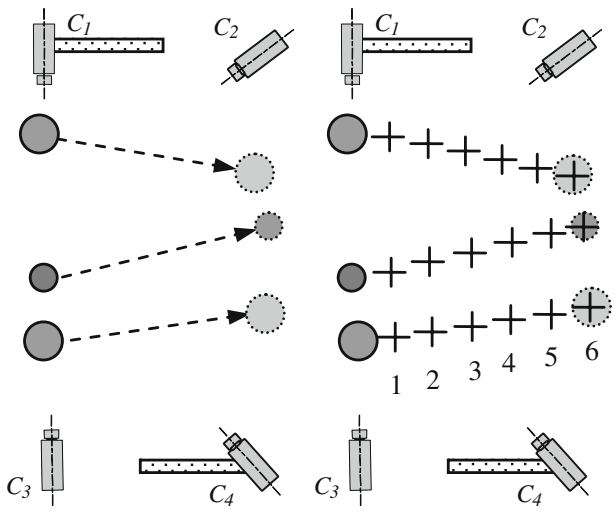

Fig. 3 *Left* Initial sensor poses and OoI trajectories. *Right* OoI/Obstacle poses at each demand instant

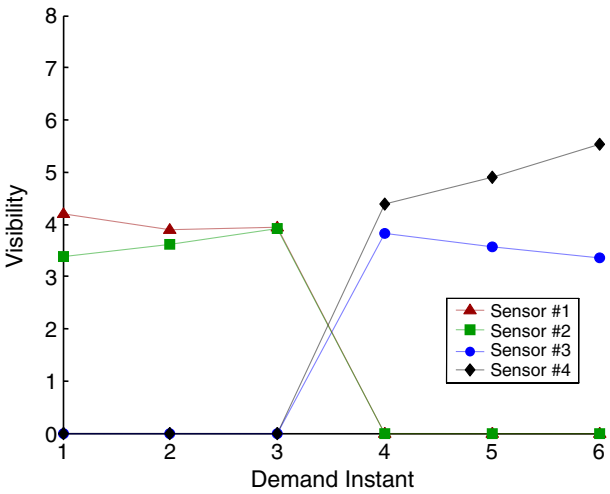

Fig. 4 Observed sensor visibilities for static system, in the presence of dynamic obstacles

increasing the visibility of the target directly leads to improved vision performance, and thus a tangible benefit.

4 An Experimental Example

4.1 Experimental Set-Up

An experimental setup was also devised to evaluate the performance of the proposed sensor-reconfiguration algorithm in a situation close to a real-world application. All results have been gathered from a real-world, physical setup using an implementation of the proposed algorithm.

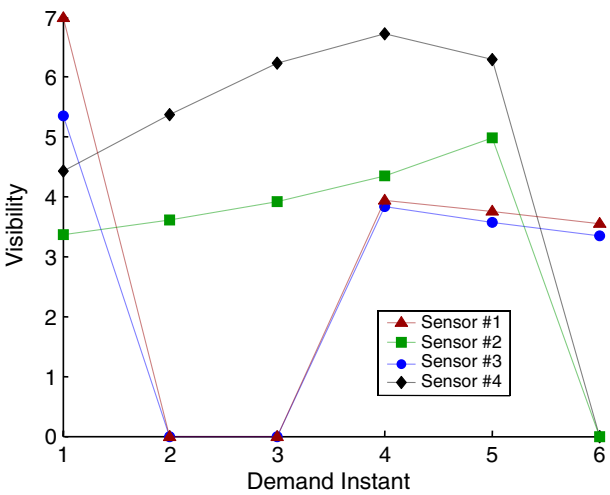

Fig. 5 Observed sensor visibilities for slow system, in the presence of dynamic obstacles

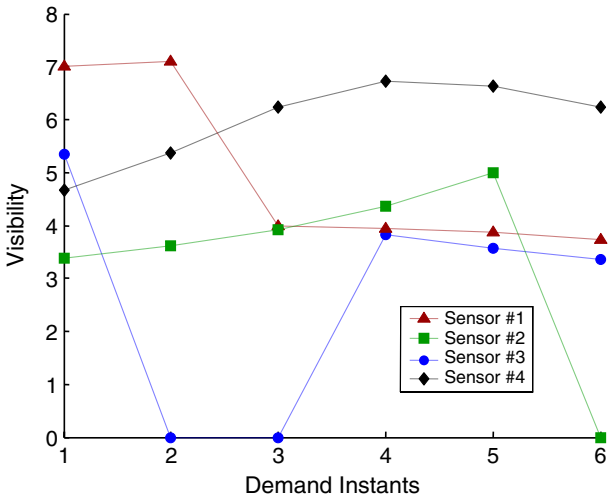

Fig. 6 Observed sensor visibilities for fast system, in the presence of dynamic obstacles

Hardware This system uses four mobile cameras and a single target, represented by a circular marker which maneuvers through the workspace on a planar trajectory, Fig. 8. Due to system limitations, only stationary obstacles could be considered during the experiments. A stationary, overhead camera (operating at 30 fps) is utilized to obtain general estimates of the target motion and obstacle locations. All cameras have one-DOF rotational capability (pan), while two of the cameras can also translate linearly (see Table 2 for component list).

Software The surveillance system's software consists of a collection of real and virtual agents; some of which were already described in Section 2 (i.e., *Sensor Agents*, *Referee Agent*, and *Judge Agent*). Others were added during the experiments in order to provide supporting functions (i.e., *Tracking and Prediction Agent* and *Sensor-*

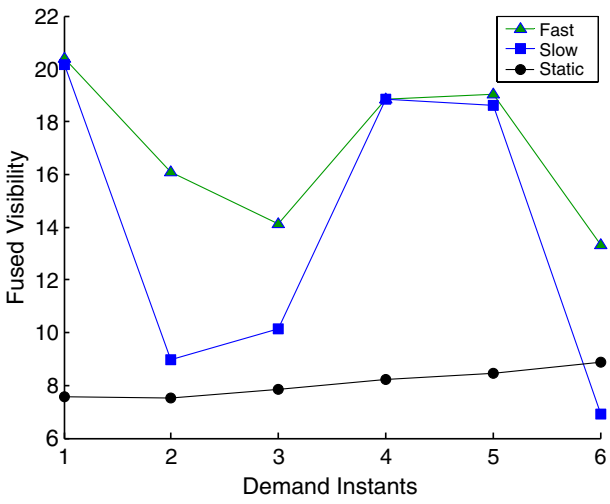

Fig. 7 Fused visibility of all three systems in the presence of dynamic obstacles

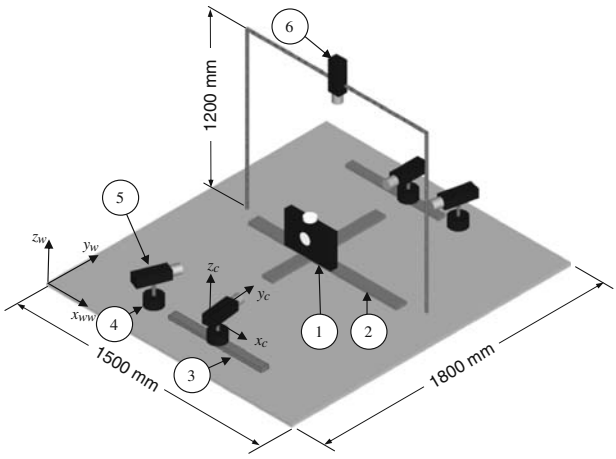

Fig. 8 System layout (see Table 2 below for hardware details)

Fusion Agent), and are described below, Fig. 9. A total of three physical systems were used, with communication provided by Ethernet-based client/server architecture. Uniform demand instant spacing was used, with a virtual period of 1/2 s – however, the algorithm has been tested in simulation to operate at speeds higher than the 30-Hz upper limit imposed by the target-tracking system.

4.1.1 Common Agents

Sensor Agents The sensor agent receives current and future OoI pose estimations from the tracking agent. It uses the information to choose the demand instants that the associated camera will service and to determine its optimum pose during data acquisition. A sensor agent's desired pose (if approved by the judge/referee agents) is sent to the associated motion controller in order to maneuver the sensor via its linear and rotary stages. Each sensor agent captures and processes images independently. The pose of the OoI is estimated using an analytical solution developed earlier in our

Table 2 Hardware specifications

Part no.	Hardware	Characteristic
1	Target	Matte black aluminum plate marked with white circular marker (diameter = 25 mm)
2	$x - y$ table	Range: 500 mm (x)/200 mm (y) Positional accuracy: 48 μm (x)/24 μm (y) Maximum velocity: 0.3 m/s
3	Two linear stages	Range: 300 mm Positional accuracy: 30 μm Maximum velocity: 0.3 m/s
4	Four rotary stages	Positional accuracy: 10 arc s Maximum velocity: $\pi/6$ rad/s
5	Four dynamic CMOS cameras	Resolution: 640 × 480 pixels Lens focal length: 25 mm
6	One static CCD camera	Resolution: 640 × 480 pixels Lens focal length: 12 mm

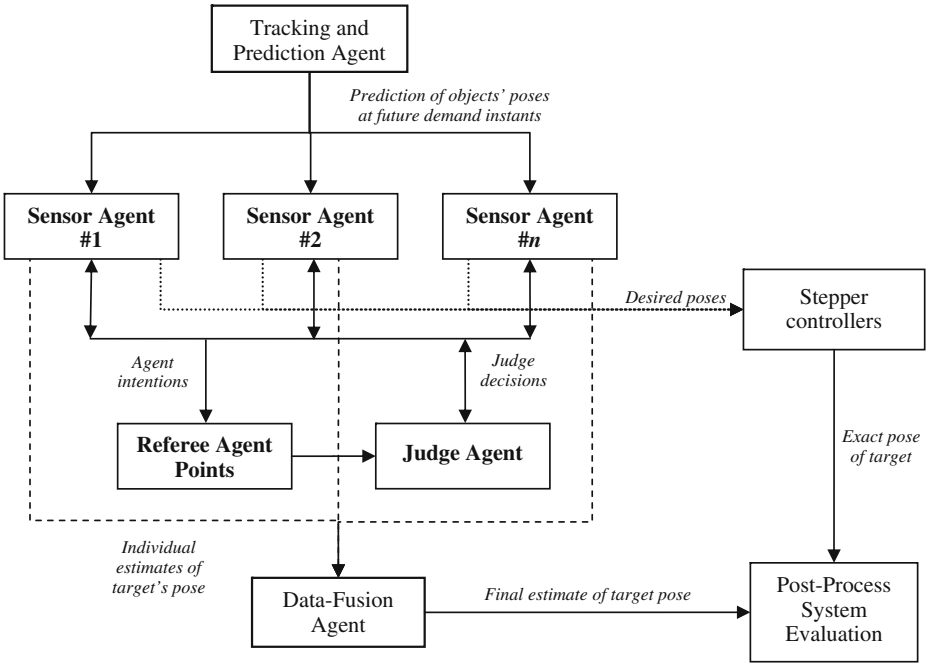

Fig. 9 Software architecture of the active surveillance system

laboratory [18], Appendix B. The data from individual sensor agents are, then, fused by a separate virtual agent in order to decrease the uncertainty in the final estimate of target's pose, Fig. 9.

Referee and Judge Agents As discussed in Section 2, these two virtual agents monitor the intentions of the sensor agents and ensure that a desirable global behavior is achieved.

4.1.2 Additional Agents

In addition to the basic agents of the proposed algorithm presented in Section 2, agents specific to the experimental setup were also implemented:

Tracking and Prediction Agents The purpose of the prediction agent is to determine estimates of the OoI's future positions by tracking a circular marker mounted on top of the OoI using a static overhead camera, Fig. 8. A center coordinate for each top marker is first determined in image coordinates, which are subsequently transformed to world coordinates, after calibration of the camera parameters using Tsai's camera-calibration technique [28]. The observed position of the marker is, then, fed into a recursive Kalman Filter (KF) [29, 30] for OoI-motion prediction. Namely, the agent uses the OoI's state model, maintained by the KF, to predict its future poses at the predefined demand instants.

Data-Fusion Agent Individual estimates of the OoI's poses are first transformed from their camera coordinate frames into a common world coordinate frame using the known camera poses. Since the center of the camera frame and its rotation axis may not coincide, there may exist an offset that must be accounted for when transforming the target position from the camera frame to the common world coordinate frame. This offset is determined through a moving camera-calibration method outlined in Appendix C. The aligned estimates are, then, fused by the data-fusion agent in order to determine a single estimate of the OoI's pose in world coordinates. The specific fusion algorithm, *Optimal Region* [31], was chosen for several reasons: (1) it requires a minimum amount of a priori knowledge about the target's trajectory, providing robustness to unexpected trajectory variations; (2) uncertainties in all cameras are estimated and considered to allow optimal fusion; (3) in the event of sensor malfunction, invalid data may be identified and discarded, allowing some degree of fault tolerance; and, (4) it is computationally efficient, enabling an on-line implementation.

A model developed in [32], which represents each sensor reading as a range containing the correct value of the variable, forms the basis of the Optimal-Region fusion algorithm. In order to distinguish between the model and the physical sensor, two terms were defined in [32]: *concrete sensor* and *abstract sensor*. A concrete sensor is the physical sensor with a single value reading, χ. An abstract sensor has a range of values, ρ, which includes the correct value of the physical variable being measured by the sensor. This is defined as,

$$\rho \in \Re, \chi - \delta < \rho < \chi + \delta, \tag{16}$$

where δ represents the accuracy of the sensor, calculated through a priori knowledge of the sensor's uncertainty. The Optimal-Region fusion algorithm requires an abstract sensor reading from each of the physical sensors (i.e., given n sensor readings, the algorithm will acquire n ranges, $\rho_1 \ldots \rho_n$). If a no-fault system is assumed (i.e.,

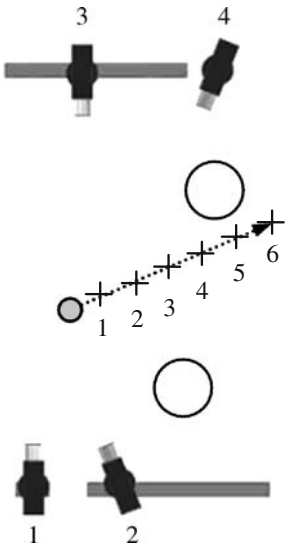

Fig. 10 Initial camera poses and OoI trajectory

all abstract sensors return a range that includes the correct value of the physical variable), then the range of values common to all abstract sensors contains the correct value of the physical variable. This common range, ρ_c, is referred to as the optimal region:

$$\rho_c = \rho_1 \cap \rho_2 \cap \ldots \cap \rho_n. \tag{17}$$

In real systems, however, faults may occur and, therefore, one or more sensors may not return a range that includes the correct value of the physical variable. The Optimal-Region algorithm will still return a range that includes the correct value if

$$n_f < n/2 \tag{18}$$

where n_f is the total number of faulty sensors and n is the total number of sensors used ([32] provides proof and further details). In order to accomplish this, the algorithm finds regions where $(n-n_f)$ of the n abstract sensors intersect, referred to as *probable regions*. The algorithm recursively uses range trees to return (as the optimal region) the smallest region that contains all probable regions. The final fused estimate, χ_{fused}, is the weighted average of the center of each probable region, q_j,

$$\chi_{\text{fused}} = \frac{\sum_j q_j s_j}{\sum_j s_j}, \tag{19}$$

where s_j is the number of sensors intersecting in the jth probable region.

4.2 Experimental Procedure and Results

A number of experiments were conducted in our laboratory to evaluate the performance of the proposed surveillance system, using the active vision set-up discussed above. The experiments verified that the performance of a surveillance system can be tangibly improved with the use of an effective dispatching strategy, under changing

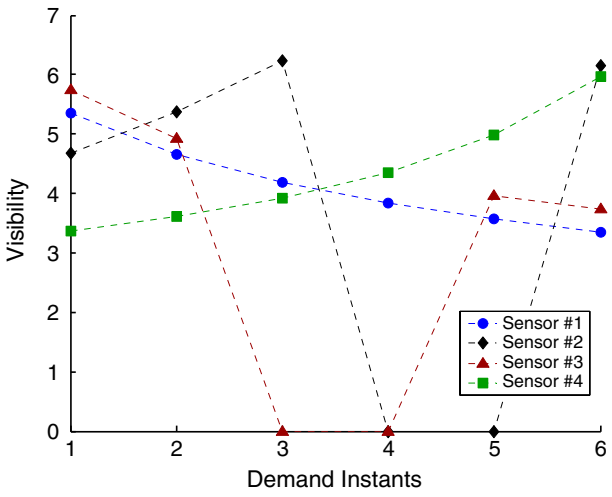

Fig. 11 Observed visibilities, slow system

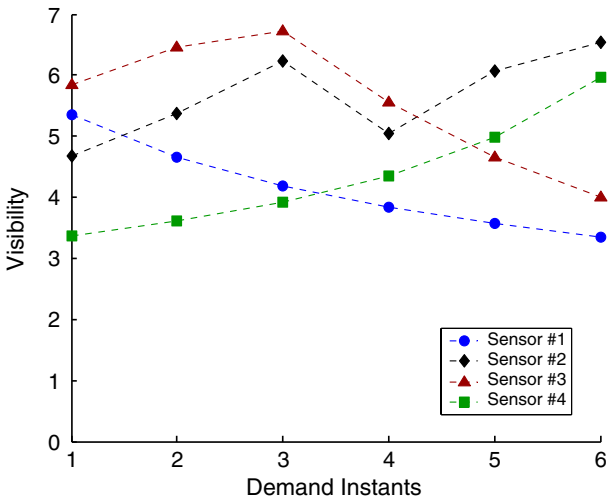

Fig. 12 Observed visibilities, fast system

OoI trajectories and sensor dynamics. This is primarily due to (1) increased robustness of the system (i.e., its ability to cope with a priori unknown target trajectories and presence of obstacles), (2) decreased uncertainty associated with estimating the target's pose through sensor fusion, and (3) increased reliability through fault tolerance.

Procedure For this experiment, the performance of the same fast and slow dynamic systems are compared. The system parameters are those outlined above and the target followed the trajectory shown in Fig. 10.

System evaluation was carried out using the visibility metric discussed in Section 2.1. Target visibility for a sensor is calculated using the expected variance in the measurements. This is a function of the Euclidean distance to the target and the angle of the camera's local axis to the OoI's surface normal. Evaluation is performed by the *Post-Process System Evaluation* agent,which finds exact errors in the real-time estimation of target pose. The first of these errors is the *absolute error*

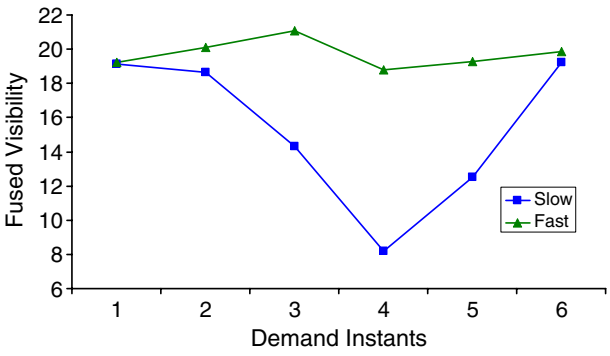

Fig. 13 Fused visibilities of both systems

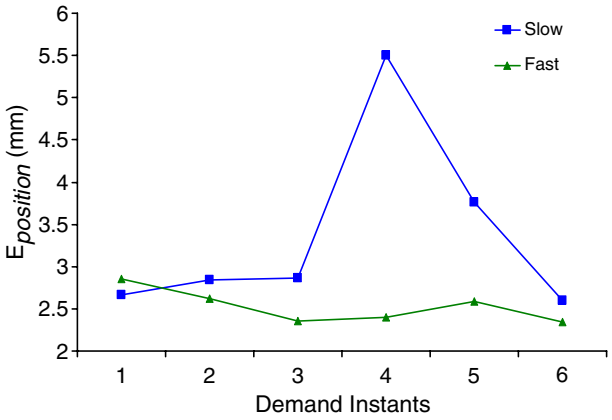

Fig. 14 Absolute errors in OoI position estimate

in position estimation, e_{position}, defined as the Euclidean distance between the true target position, $\boldsymbol{o}_p = (x_t, y_t, z_t)$, and the system's estimate of the target's position, $\chi_{\text{fused}} = (x_e, y_e, z_e)$:

$$e_{\text{position}} = \left\| o_p - \chi_{\text{fused}} \right\|. \tag{20}$$

Similarly, the absolute error in surface normal estimation, $e_{\text{orientation}}$, is the angle between the true OoI orientation, o_n, and the estimated surface normal, $o_{n_{\text{fused}}}$:

$$e_{\text{orientation}} = \cos^{-1}\left(o_n . o_{n_{\text{fused}}}\right). \tag{21}$$

Results The pose of the moving target was estimated at six demand instants. The visibilities of each sensor over the demand instants are given in Figs. 11 and 12. Fused visibilities for both systems are shown in Fig. 13. One can note that the fused target visibilities of the fast system are tangibly higher than those of the slow system. Again, by increasing the average visibility metric, one can infer that the performance of the system has increased. The corresponding absolute position errors are shown in Fig. 14 and the absolute errors in surface-normal estimations are given in Fig. 15. Despite the presence of random noise in both systems, the data confirms

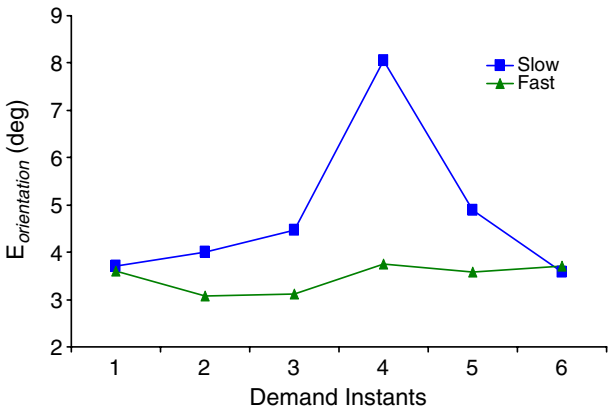

Fig. 15 Absolute errors in OoI surface-normal estimate

the improvement of system performance through the use of the fast system – it has a considerably lower overall absolute error. As such, one can conclude that increased mobility of the sensors allows our proposed methodology to further improve average visibility of the target.

5 Conclusions

A novel, generalized methodology is presented in this paper for the coordinated selection and positioning of groups of active sensors for the autonomous surveillance of a single target in a multi-object dynamic environment. The experiments and simulations presented have shown that tangible improvements in performance over the static case can be obtained through the use of multiple active sensors controlled by the proposed dispatching algorithm, and that increased sensor motion capabilities can improve performance further. While the overall reduction in uncertainty is significant, cases still exist where the limited motion capabilities of the system preclude the best solution. Future work will focus on addressing the limits imposed by the real-world motion capabilities of the sensors, and on extending the framework to multi-target environments.

Acknowledgements This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERCC). Ardevan Bakhtari would also like to acknowledge the financial support of the Ontario Graduate Scholarship (OGSST).

Appendix A: Sensor Modeling

Sensor modeling is an important part of optimal dispatching, where the objective is to estimate a sensor's performance, given a set of environmental conditions. For the cameras used in our experiments, there are six variance measurements that define the visibility metric. Three are for the target position (x, y, z) and three for orientation (n_x, n_y, n_z). Through variation analysis it was determined that only two controlled parameters significantly affect the measurement variances: the Euclidean camera-to-target distance, d, and the camera's bearing, θ, as in Fig. 16.

Two-factorial experiments were performed to determine the relationship between each measurement variance and the two controlled parameters, d and θ. As an example, the results for two variances (estimation along the y-axis and one surface

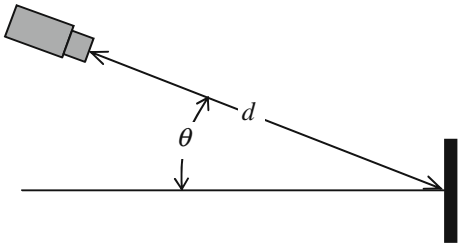

Fig. 16 Camera's distance, d, and bearing, θ

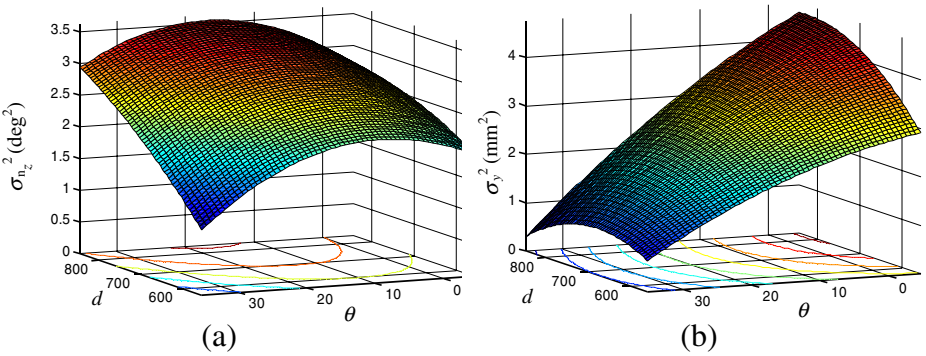

Fig. 17 Response surfaces of variances in (**a**) orientation estimation about z-axis, $\sigma_{n_z}^2$ (**b**) y-axis position, σ_y^2

normal) are shown in Fig. 17. One can note that the significant changes in target-localization performance. The shape of the elliptical projection of the circular marker (i.e., the target) is better viewed at an angle between 20–40° from the surface normal and reduced noise-to-signal ratios at close distances reduce variance. It should be also noted that these are not the only factors that contribute to the measurement variance. However, since only the camera's pose is dynamically adjusted (i.e., extrinsic parameters), these are the only controlled parameters that affect the measurement variance here. Other parameters, such as illumination, also affect the measurement variance but are not included in the visibility measure.

Appendix B: 3D Location Estimation

The process of 3D-location estimation [20] of a circular feature, using the estimated general parameters of the ellipse, is as follows:

1. Estimation of the coefficients of the general equation of the cone:

$$ax^2 + by^2 + cz^2 + 2fyz + 2gzx + 2hxy + 2ux + 2vy + 3wz + d = 0. \quad (22)$$

2. Reduction of the equation of the cone to:

$$\lambda_1 X^2 + \lambda_2 Y^2 + \lambda_3 Z^2 = 0, \quad (23)$$

where the XYZ-frame is the canonical frame of the conicoids.

3. Estimation of the coefficients of the equation of the circular-feature plane (plane intersecting the cone that would result in a perfect circle):

$$lX + mY + nZ = P \quad (24)$$

using the following transformation:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{-m}{\sqrt{l^2+m^2}} & \frac{-nl}{\sqrt{l^2+m^2}} & l & 0 \\ \frac{-l}{\sqrt{l^2+m^2}} & \frac{-mn}{\sqrt{l^2+m^2}} & m & 0 \\ 0 & \sqrt{l^2+m^2} & n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} \tag{25}$$

The transformation is defined such that Z' is normal to the plane axis defined by Eq. 28. The solution to determining l, m, and n depends on which of three possible cases occurs:

(a) $\lambda_1 < \lambda_2$,
(b) $\lambda_1 > \lambda_2$, or
(c) $\lambda_1 = \lambda_2$,

4. Estimation of the direction cosines of the surface normal with respect to the camera frame: l, m, n.
5. Knowing the radius of the circular feature (r) and its center ($X'_o\ Y'_o\ Z'_o$) in camera coordinates, ($X'\ Y'\ Z'$) can be found by solving the following system of equations (Fig. 18):

$$\begin{aligned} X'_0 &= -\frac{B}{A} Z'_0 \\ Y'_0 &= \frac{C}{A} Z'_0 \\ Z'_0 &= \pm \frac{Ar}{\sqrt{B^2 + C^2 - AD}}, \end{aligned} \tag{26}$$

where

$$\begin{aligned} A &\equiv \left(\lambda_1 l_1^2 + \lambda_2 l_2^2 + \lambda_3 l_3^2\right) \\ B &\equiv \left(\lambda_1 l_1 n_1 + \lambda_2 l_2 n_2 + \lambda_3 l_3 n_3\right) \\ C &\equiv \left(\lambda_1 m_1 n_1 + \lambda_2 m_2 n_2 + \lambda_3 m_3 n_3\right) \\ D &\equiv \left(\lambda_1 n_1^2 + \lambda_2 n_2^2 + \lambda_3 n_3^2\right). \end{aligned} \tag{27}$$

Appendix C: Moving-Camera Calibration

In order to determine the relationship between camera and world coordinates, the extrinsic camera parameters must be determined. These were found in our work through Tsai's calibration method [28]. However, in the case of active-vision, the cameras' extrinsic parameters are constantly changing due to rotations and translations. In order to account for these changes, the algorithm first applies a transformation matrix to return the rotated camera frame to the frame identical to

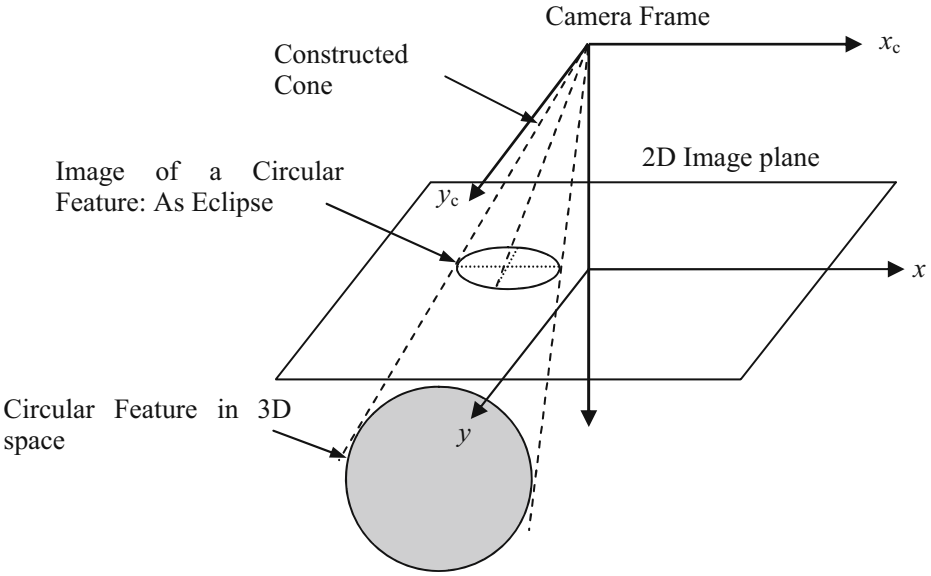

Fig. 18 A schematic representation of the ellipse created in the image plane from the circular feature

its original reference frame before rotation. It, then, applies a second transformation matrix to account for any translational movement of the camera since the initial calibration. At this stage, the original calibration matrix can be applied to transform the camera frame to the world frame.

In order to counter the effects of rotation, one must know the rotation axis, the rotation angle, and the center of rotation. We assume that the rotation axis the z-axis, due to the optical tables and high-precision rotary tables used in our experimental set-up. A secondary calibration method is used to determine the center of rotation in camera coordinates. First, the target is placed in the camera's field of view and

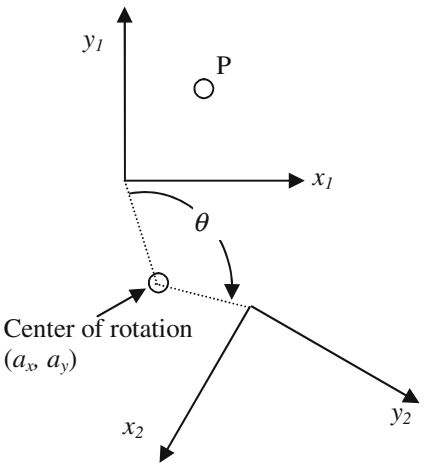

Fig. 19 Camera coordinate-frame transformation due to camera rotation

readings are taken on its position in camera coordinates. Multiple readings are taken and the results fused to minimize the effect of random noise. Without changing the target position, the camera is then rotated as much as possible while still keeping the target in its field of view. A second set of readings is taken on the target's position in the new camera coordinates.

We solve the following equations for the required offsets:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \sin(\theta) \end{bmatrix} \begin{bmatrix} Px_1 - a_x \\ Py_1 - a_y \end{bmatrix} + \begin{bmatrix} a_x \\ a_y \end{bmatrix} = \begin{bmatrix} Px_2 \\ Py_2 \end{bmatrix} \tag{28}$$

where (Px_1, Py_1) and (Px_2, Py_2) are the target locations in camera coordinates before and after rotation, respectively, θ is the rotation angle, and (ax, ay) is the center of rotation in camera coordinates, Fig. 19. The inverse of Eq. 28 transforms any camera-frame rotation to the original camera frame location.

References

1. Tarabanis, K.A., Allen, P.K., Tsai, R.Y.: A survey of sensor planning in computer vision. IEEE Trans. Robot. Autom. **11**(1), 86–104 (1995)
2. Sakane, S., Sato, T., Kakikura, M.: Model-based planning of visual sensors using a hand-eye action simulator: HEAVEN. In: Espiau, B. (ed.) Proc. Conf. on Advanced Robotics, pp. 163–174. Versailles, France (1987)
3. Tarbox, G.H., Gottschlich, S.N.: Planning for complete sensor coverage in inspection. Comput. Vis. Image Underst. **61**(1), 84–111 (1995)
4. Cowan, C.K., Kovesik, P.D.: Automated sensor placement from vision task requirements. IEEE Trans. Pattern Anal. Mach. Intell. **10**(3), 407–416 (1988)
5. Anderson, D.P.: Efficient algorithms for automatic viewer orientation. Trans. Comp. Graphics. **9**(4), 407–413 (1985)
6. Zhang, H.: Two-dimensional optimal sensor placement. IEEE Trans. Syst. Man. Cybern. **25**(5), 781–792 (1995)
7. Trucco, E., Umasuthan, M., Wallace, A.M., Roberto, V.: Model-based planning of optimal sensor placements for inspection. IEEE Trans. Robot. Autom. **13**(2), 182–194 (1997)
8. Sheng, W., Xi, N., Song, M., Chen, Y.: CAD-guided sensor planning for dimensional inspection in automotive manufacturing. IEEE-ASME Trans. Mechatron. **8**(3), 372–380 (2003)
9. Niepold, R., Sakane, S., Shirai, Y.: Vision sensor set-up planning for a hand-eye system using environmental models. In: Proceedings of the Society of Instrument and Control Engineers of Japan, vol. 7(1), pp. 1037–1040. Hiroshima, Japan, July (1987)
10. Matsuyama, T., Wada, T., Tokai, S.: Active image capturing and dynamic scene visualization by cooperative distributed vision. In: Nishio, S., Kishino, F. (eds.) Advanced Multimedia Content Processing, vol. 11(4), pp. 252–288. Springer, Berlin (1999)
11. Horling, B., Vincent, R., Shen, J., Becker, R., Rawlins, K., Lesser, V.: SPT distributed sensor network for real-time tracking. Technical Report 00-49. University of Massachusetts, Amherst, MA (2000)
12. Spletzer, J.R., Taylor, C.J.: Dynamic sensor planning and control for optimally tracking targets. Int. J. Rob. Res. **22**(1), 7–20 (2003)
13. Kamel, M., Hodge, L.: A coordination mechanism for model-based multi-sensor planning. In: Proceedings of the IEEE International Symposium on Intelligent Control, pp. 1143–1149. Vancouver, Canada (2002)
14. Zhou, H., Sakane, S.: Sensor planning for mobile robot localization using Bayesian network inference. J. Adv. Rob. **16**(8), 751–771 (2002)
15. Tsai, R.Y., Tarabanis, K.: Model-based planning of sensor placements and optical settings. In: Sensor Fusion II: Human and Mach. Strategies, pp. 936–944. Philadelphia, PA (1989)
16. Tsai, R.Y., Tarabanis, K.: Occlusion-free sensor placement planning. In: Freeman, H. (ed.) Machine Vision for Three Dimensional Scenes, pp. 349–356. Academic, Orlando, FL (1990)

17. Goodridge, S.G., Kay, M.G.: Multimedia sensor fusion for intelligent camera control. In: Proc. of IEEE/SICE/RSJ Multi-sensor Fusion and Integration for Intelligent Systems, pp. 934–940. Washington, D.C. (1996)
18. Merchand, E., Hager, G.D.: Dynamic sensor planning in visual servoing. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 1988–1993. Leuven, Belgium (1998)
19. Farag, A.A., Abdel-Hakim, A.E.: Image content-based active sensor planning for a mobile trinocular active vision system. In: International Conference on Image Processing (ICP04), pp. 2913–2916, Oct (2004)
20. Chen, S.Y., Li, Y.F.: Automatic sensor placement for model-based robot vision. IEEE Trans. Syst. Man. Cybern. B **34**(1), 393–408 (2004)
21. Isler, V., Bajcsy, R.: The sensor selection problem for bounded uncertainty sensing models. In: Proc. of the 4th Intl. Symposium on Information Processing in Sensor Networks. Information Processing in Sensor Networks, vol. 20, pp. 151–158. IEEE Press, Piscataway, NJ (2005)
22. Tang, Z., Ozguner, U.: Motion planning for multi-target surveillance with mobile sensor agents. IEEE Trans. Robot. **21**(5), 898–908 (2005)
23. Rekleitis, I., Meger, D., Dudek, G.: Simultaneous planning, localization, and mapping in a camera sensor network. Robot. Auton. Syst. **54**(11), 921–932 (2006)
24. Ukita, N., Matsuyama, T.: Real-time cooperative multi-target tracking by communicating active vision agents. In: Proc. of the Int. Conf. on Information Fusion, pp. 439–446. Queensland, Australia (2003)
25. Cook, D.J., Gmytrasiewicz, P., Holder, L.B.: Decision-theoretic cooperative sensor planning. IEEE Trans. Pattern Anal. Mach. Intell. **18**(10), 1013–1023, Oct (1996)
26. Cook, D.: Reconfiguration of multi-agent planning systems. In: Proc. Artificial Intelligence Planning Systems, pp. 225–230 (1994)
27. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Upper Saddle River, NJ, Prentice Hall (2003)
28. Tsai, R.: A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE J. Robot. Autom. **3**(4), 323–344 (1987)
29. Bakhtari, A., Eskandari, M., Naish, M.D., Benhabib, B.: A multi-sensor surveillance system for active-vision based object localization. IEEE, Conf. System, Man and Cybernetics, pp. 1013–1018. Washington, DC, October (2003)
30. Kalman, R.E.: A new approach to linear filtering and prediction problems. Trans. ASME J. Basic Eng. **82**(4), 35–45 (1961)
31. Brooks, R., Iyengar, S.S.: Multi-sensor fusion: Fundamentals and applications with software. Prentice Hall, Englewood Cliffs, NJ (1998)
32. Marzullo, K.: Tolerating failures of continuous-valued sensors. ACM Trans. Comput. Syst. **8**(4), 284–304 (1990)