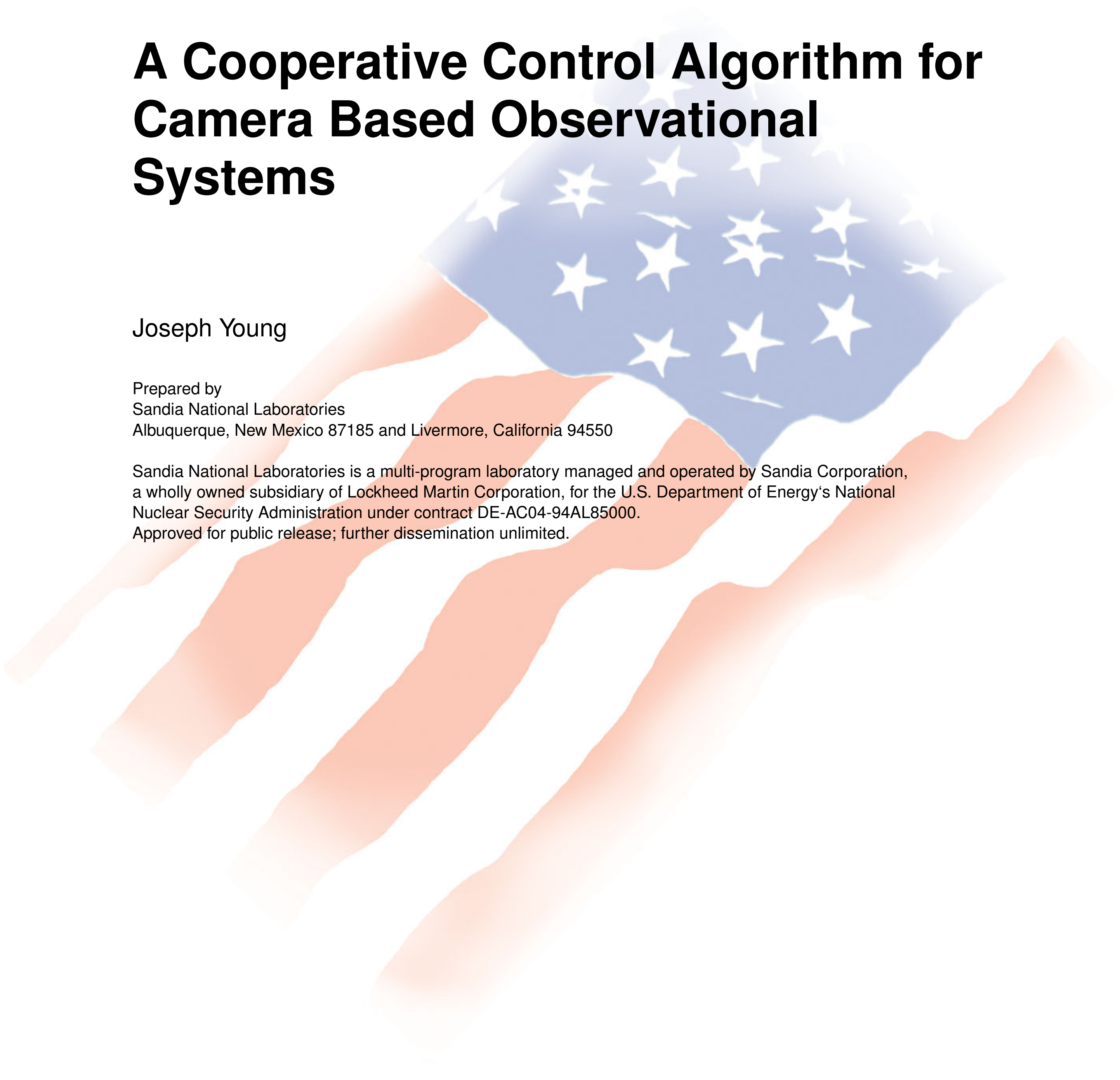
SANDIA REPORT

SAND2011-9324
Unlimited Release
Printed January, 2012

A Cooperative Control Algorithm for Camera Based Observational Systems



Joseph Young

Sandia National Laboratories

SAND2011-9324
Unlimited Release
Printed January, 2012

A Cooperative Control Algorithm for Camera Based Observational Systems

Joseph Young
Numerical Analysis and Applications
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1320
josyoun@sandia.gov

Abstract

Over the last several years, there has been considerable growth in camera based observation systems for a variety of safety, scientific, and recreational applications. In order to improve the effectiveness of these systems, we frequently desire the ability to increase the number of observed objects, but solving this problem is not as simple as adding more cameras. Quite often, there are economic or physical restrictions that prevent us from adding additional cameras to the system. As a result, we require methods that coordinate the tracking of objects between multiple cameras in an optimal way.

In order to accomplish this goal, we present a new cooperative control algorithm for a camera based observational system. Specifically, we present a receding horizon control where we model the underlying optimal control problem as a mixed integer linear program. The benefit of this design is that we can coordinate the actions between each camera while simultaneously respecting its kinematics. In addition, we further improve the quality of our solution by coupling our algorithm with a Kalman filter. Through this integration, we not only add a predictive component to our control, but we use the uncertainty estimates provided by the filter to encourage the system to periodically observe any outliers in the observed area. This combined approach allows us to intelligently observe the entire region of interest in an effective and thorough manner.

1 Introduction and Background

In recent years, there has been large growth in the application of automated camera based observation systems. The rate of the growth seems rivaled only by its breadth. For example, the wildlife and conservation community has made great use of remotely placed cameras in order to track wildlife. This includes using cameras to monitor the effectiveness of a highway underpass in Arizona [9], a camera system to monitor bears in the arctic circle in order to gauge behavior responses to ecotourists [29], and an automated system used to search for the critically endangered ivory-billed woodpecker [27, 1].

Within the sports community, there has also been growth in automated camera based systems. With respect to hockey, a camera based system tracks multiple players and attempts to classify their actions [14]. In another, multiple cameras are used to track the position of the hockey puck [6]. In baseball, multiple cameras are being used to collect a variety of new sports statistics [26].

In the public safety realm, there have been a variety of successful application of such systems. Of course, one of the most ubiquitous places such systems arise is in traffic monitoring [23, 13]. Alternatively, there has been work on the automatic detection of fires using satellites [7, 8]. Finally, there has been a vast amount of research on using automated closed circuit television systems (CCTVs) to monitor people. This includes work on monitoring and predicting crowd flow [5], tracking individual movements of people [11, 12], and a variety of other computer vision based tracking research [16, 17].

The unifying thread between all of the above research papers is their use of optical surveillance systems to gather and process information concerning their observations automatically. Generally, these systems involve multiple observations between several cameras. In some situations, these cameras move; in others, they do not. In either situation, we may ask ourselves whether or not our cameras are optimally oriented so that they may observe the area of interest at hand. Certainly, multiple cameras looking at the same location may not enhance the coverage of an area. In fact, a few well oriented cameras may observe an area of interest better than several poorly positioned devices. As a result, the proper placement and orientation of a set of cameras within some area is critical to our ability to observe the area of interest. In order to narrow our discussion, we focus on the situation of reorientable cameras mounted on pan/tilt devices and try to answer a very specific question. Can we coordinate the motion between several cameras so that they observe an area of interest more effectively according to some metric? In order to address this question, we investigate an optimal control algorithm formulated as a mixed integer linear program (MILP).

The type of algorithm that we pursue is most related to a class of algorithms used in the control of unmanned aerial vehicles (UAVs). In these algorithms, a receding horizon control applied to an optimal control problem modeled as a mixed integer linear program. The advantage of this approach is that certain discrete goals and constraints can be satisfied while simultaneously respecting the kinematics of the system. As an example of this methodology, Richards et. al developed a control algorithm for spacecraft where each craft coordinates with the others in order to avoid both collisions as well any plume that arises from using the vehicle's thrusters [22]. In a similar manner, Richards and How also developed a method to coordinate between multiple UAVs and avoid collisions [21]. Later, they created relatively general guidelines to applying this methodology to more generic control problems [20]. In a similar vein, Schouwenaars et. al developed a generic multivehicle path planning algorithm formulated as a mixed integer linear program [25]. Later, this algorithm was improved to coordinate the control of multiple helicopters that cooperate in order to insure a continuous line of communication to some ground station [24]. Bellingham et. al devised similar algorithms for coordinated action between fixed winged UAVs whose motion must respect a set of no fly zones [3]. Later, a stochastic element was added in order to account for the potential loss of multiple UAVs [4]. Also working on UAVs, Ademoye and Davari produced another mixed integer control algorithm for the coordinated control between multiple air vehicles [2]. Likewise, Reinl and von Stryk produced a slight variation on this general theme where they produced a mixed integer control for UAVs while adding constraints that insure a continuous line of communication between aircraft [19]. In a different application area, but using a similar approach, Earl and D'Andrea created a mixed integer control algorithm for a series of robots competing in the RoboFlag competition [10]. Also working with ground based robots, Thunberg and Ögren studied the pursuit evasion problem when modeled as a mixed integer control [28]. Finally, as a summary of many of these approaches, Murray produced a survey paper that discussed many of these results among others [18].

In the following sections, we develop a receding horizon control that coordinate actions between several different cameras working together to observe some area of interest. In order to accomplish this, we model our system as a mixed integer linear program which allows us to correctly model the kinematics of our cameras while accurately modeling a discrete number of observable objects in the area of interest. In addition, we augment our approach by integrating our model with a Kalman filter which not only improves the predictive capability of the approach, but assists in helping the cameras periodically observe outliers to the system so that the entire area is completely observed. Then, we verify the efficacy of our approach with a numerical experiment. Our contribution to the literature is the use of a mixed integer control to a new application area, a new formulation of a goal based optimization problem, and the integration of a Kalman filter with the mixed integer control.

2 Formulation of the Optimal Control

In the following formulation, we model a system where we have multiple cameras observe multiple objects. In this system, the cameras may be stationary or move as long as we assume that we know the position of each camera with respect to some global coordinate system. In addition to spatial movement, each camera may change its orientation via a pan/tilt unit. With respect to the objects of interest, each object may move, but we assume that we have an external mechanism that identifies the position of each object as well as predicts the path that it takes. One example of such a predictive system is a Kalman filter. Finally, our model and schedule remains discrete in time. Although the control of an individual camera is continuous in nature, we place constraints on the kinematics of the pan/tilt unit so that our discrete schedule may be realized in a continuous system.

Within a pan/tilt unit, the elevation angle describes how far up or down the camera points whereas the azimuth angle describes how far the camera has rotated to the left or right. Let $c_\phi^{(i)}$ and $c_\theta^{(i)}$ denote initial elevation and azimuth angles of camera i. Although we allow the cameras to rotate, we use these angles to fix a local coordinate system. In order to model the change in orientation, let $\phi^{(i)}[k]$ and $\theta^{(i)}[k]$ denote the offset in orientation. In this way, we determine the current orientation of camera i at time k with the quantity $(c_\phi^{(i)} + \phi^{(i)}[k], c_\theta^{(i)} + \theta^{(i)}[k])$.

Each camera also possesses a field of view angle which determines how much the camera can see. Let $\nu_\phi^{(i)}$ and $\nu_\theta^{(i)}$ denote the vertical and horizontal field of view angles respectively. Using the above notation, at time step k, each camera may view objects that lie between $c_\phi^{(i)} + \phi^{(i)}[k] - \nu_\phi^{(i)}/2$ and $c_\phi^{(i)} + \phi^{(i)}[k] + \nu_\phi^{(i)}/2$ vertically as well as $c_\theta^{(i)} + \theta^{(i)}[k] - \nu_\theta^{(i)}/2$ and $c_\theta^{(i)} + \theta^{(i)}[k] + \nu_\theta^{(i)}/2$ horizontally when considered from a local coordinate system.

Due to the physical limitations of a pan/tilt unit, we place constraints on how fast each unit may move. Let $v_\phi^{(i)}[k]$, $v_\theta^{(i)}[k]$, $a_\phi^{(i)}[k]$, and $a_\theta^{(i)}[k]$ denote the velocity and acceleration of camera i at time step k. Since a pan/tilt unit can not start and stop instantaneously, we place restrictions on the position, velocity, and acceleration of the units. Let $\phi_L^{(i)}$, $\phi_U^{(i)}$, $\theta_L^{(i)}$, $\theta_U^{(i)}$ denote the lower and upper bounds on how far the camera may rotate. Similarly, let $v_{\phi L}^{(i)}$, $v_{\phi U}^{(i)}$, $v_{\theta L}^{(i)}$, and $v_{\theta U}^{(i)}$ denote the lower and upper bounds on the velocity of the pan/tilt unit. In addition, let $a_{\phi L}^{(i)}$, $a_{\phi U}^{(i)}$, $a_{\theta L}^{(i)}$, and $a_{\theta U}^{(i)}$ denote bounds on the acceleration of the unit. Finally, based on these quantities, we relate the position, velocity, and acceleration of camera i using simple Newtonian physics and the equations

$$\phi^{(i)}[k+1] = \phi^{(i)}[k] + \Delta t v_\phi^{(i)}[k] + \frac{\Delta t^2}{2} a_\phi^{(i)}[k] \tag{1}$$

$$\theta^{(i)}[k+1] = \theta^{(i)}[k] + \Delta t v_\theta^{(i)}[k] + \frac{\Delta t^2}{2} a_\theta^{(i)}[k] \tag{2}$$

$$v_\phi^{(i)}[k+1] = v_\phi^{(i)}[k] + \Delta t a_\phi^{(i)}[k] \tag{3}$$

$$v_\theta^{(i)}[k+1] = v_\theta^{(i)}[k] + \Delta t a_\theta^{(i)}[k] \tag{4}$$

where Δt denote the size of the time step. In order to define the base terms for the above equations, we denote the initial state of the system as

$$\begin{array}{ccc} \phi^{(i)}[1] = \phi_0^{(i)} & v_\phi^{(i)}[1] = v_{\phi,0}^{(i)} & a_\phi^{(i)}[1] = a_{\phi,0}^{(i)} \\ \theta^{(i)}[1] = \theta_0^{(i)} & v_\theta^{(i)}[1] = v_{\theta,0}^{(i)} & a_\theta^{(i)}[1] = a_{\theta,0}^{(i)}. \end{array} \tag{5}$$

Next, we consider the position of objects. As we mentioned before, we assume we have some external mechanism that tells us the position of each object such as a Kalman filter. Typically, this is a set of Cartesian coordinates in a global coordinate system. Based on this, we need to transform each of these coordinates into a local coordinate system based on the elevation and azimuth angles

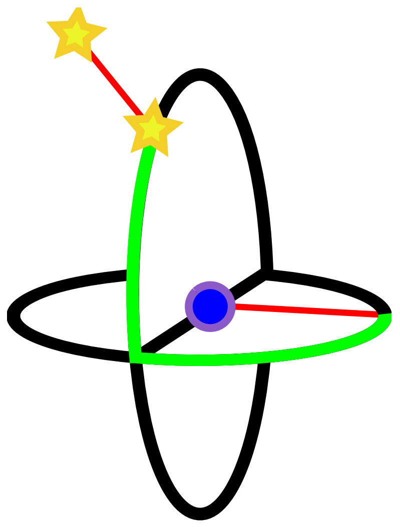


Figure 1. The above diagram indicates how the global coordinates are translated into a local spherical coordinate system. We denote the camera by the blue circle and the object in question as the yellow star. Then, we project the position of the object onto the unit sphere denoted in black. The green lines denote how much the camera would need to rotate in order to center its field of view on the object.

relative to each camera. We transform into a local coordinate system because this allows us to determine whether we can view a particular object using a system of linear inequalities. If we use a global coordinate system for our problem, the constraints become nonlinear and more difficult to solve. Based on this scheme, let us denote the position of the ith camera at the kth time step as the triple $(c_x^{(i)}[k], c_y^{(i)}[k], c_z^{(i)}[k])$. In a similar manner, let us describe the position of the jth object at the kth time step as the triple $(o_x^{(j)}[k], o_y^{(j)}[k], o_z^{(j)}[k])$. This allows us to determine the position of object j relative to camera i at time step k using the formulas

$$\delta_l^{(ij)}[k] = o_l^{(j)}[k] - c_l^{(i)}[k] \text{ for } l \in \{x, y, z\} \tag{6}$$

$$l^{(ij)}[k] = \sqrt{\delta_x^{(ij)}[k]^2 + \delta_y^{(ij)}[k]^2 + \delta_z^{(ij)}[k]^2} \tag{7}$$

$$\Delta_l^{(ij)}[k] = \delta_l^{(ij)}[k]/l \text{ for } l \in \{x, y, z\} \tag{8}$$

$$o_\phi^{(ij)}[k] = \arcsin(\Delta_y^{(ij)}[k]) - c_\phi^{(i)} \tag{9}$$

$$o_\theta^{(ij)}[k] = \text{arctan2}(\Delta_z^{(ij)}[k], \Delta_x^{(ij)}[k]) - c_\theta^{(i)} \tag{10}$$

where the notation $\delta_l^{(ij)}[k]$ for $l \in \{x, y, z\}$ allows us to abbreviate nearly identical formulas for $\delta_x^{(ij)}[k]$, $\delta_y^{(ij)}[k]$, and $\delta_z^{(ij)}[k]$. Another way to view these equations is that the point $(\delta_x^{(ij)}[k], \delta_y^{(ij)}[k], \delta_z^{(ij)}[k])$ represents the position of some object in space where the origin lies at the camera. Then, we project this point onto the unit sphere which gives the new point $(\Delta_x^{(ij)}[k], \Delta_y^{(ij)}[k], \Delta_z^{(ij)}[k])$. Once we have this point on the unit sphere, we determine the amount a camera would need to rotate in order to center on this object if the camera were pointing directly to the right down the positive part of the x-axis. Then, we modify these angles based on the fixed mount orientation of the camera. Figure 1 illustrates this scheme. We denote the object in question by a star and project it using a red line onto an outline of the unit sphere in black. Next, we denote the camera as a blue circle and draw a red line from the camera down the positive x-axis until we intersect the unit sphere. Then, we denote the amount the camera would need to rotate in order to center on the object in green. These angles give the position of the object relative to the camera.

Based on these local coordinates, we have a simple scheme to determine whether or not a camera is facing an object. Simply, we denote the amount the camera i needs to rotate in order to center on an object j at time step k with the equations

$$\epsilon_\phi^{(ij)}[k] = o_\phi^{(ij)}[k] - \phi^{(i)}[k] \tag{11}$$

$$\epsilon_\theta^{(ij)}[k] = o_\theta^{(ij)}[k] - \theta^{(i)}[k]. \tag{12}$$

When these quantities are zero, the object is centered with the field of view of the camera. Hence, camera i may view the object j at the time step k when the following inequalities are satisfied

$$|\epsilon_\phi^{(ij)}[k]| \leq \nu_\phi^{(i)}/2 \tag{13}$$

$$|\epsilon_\theta^{(ij)}[k]| \leq \nu_\theta^{(i)}/2. \tag{14}$$

As a minor point, since we are working within a spherical coordinate system, we also look directly at an object every time our camera makes a full revolution. In fact, there are infinitely many ways to center on an object due to the periodicity of the coordinate system and this is not reflected in the quantities $\epsilon_\phi^{(ij)}[k]$ and $\epsilon_\theta^{(ij)}[k]$ above. In general, ignoring these extra positions does not negatively impact the system performance unless the pan/tilt unit completes a full revolution during a single planning phase. In this case, we can address this difficulty directly with additional inequalities of the form

$$|\epsilon_\phi^{(ij)}[k] - 2\pi m| \leq \nu_\phi^{(i)}/2 \tag{15}$$

$$|\epsilon_\theta^{(ij)}[k] - 2\pi n| \leq \nu_\theta^{(i)}/2. \tag{16}$$

As long as these inequalities are satisfied for some combination of $m, n \in \mathbb{Z}$, where $\mathbb{Z}$ denotes the set of all integers, we say that an object lies within the field of view of some camera.

Of course, the field of view inequalities may or may not be satisfied for a particular combination of camera and object. This causes difficulty when introducing these inequalities directly into an optimal control formulation since the problem becomes infeasible when an object drops out of the field of view. As a result, we require a mechanism that denotes when a camera can see an object, but relaxes the above inequality when this is not possible. In order to accomplish this, we introduce a series of binary variables $b^{(ij)}[k]$ that denote whether camera i can see an object j at a time step k. Then, we modify the above field of view inequalities into

$$|\epsilon_\phi^{(ij)}[k]| \leq M + b^{(ij)}[k](\nu_\phi^{(i)}/2 - M) \tag{17}$$

$$|\epsilon_\theta^{(ij)}[k]| \leq M + b^{(ij)}[k](\nu_\theta^{(i)}/2 - M) \tag{18}$$

where M denotes some sufficiently large number that ϵ could never attain. For example, if we assume that the most a camera could rotate is 2π, a quantity of $M = 10$ should be sufficient. Let us consider these inequalities more closely. When camera i can *not* view object j at time step k, $b^{(ij)}[k] = 0$. Then, the inequalities become

$$|\epsilon_\phi^{(ij)}[k]| \leq M \tag{19}$$

$$|\epsilon_\theta^{(ij)}[k]| \leq M. \tag{20}$$

Essentially, since M is sufficiently large, the distance we need to rotate to center on the object is unrestricted. Alternatively, when camera i can view object j at time step k we have that $b^{(ij)}[k] = 1$ and the inequalities become

$$|\epsilon_\phi^{(ij)}[k]| \leq v_\phi^{(i)}/2 \tag{21}$$

$$|\epsilon_\theta^{(ij)}[k]| \leq v_\theta^{(i)}/2. \tag{22}$$

Thus, the distance necessary to center on a object must be half the view angle. As a final note, the absolute value function is nonlinear which increases the complexity of the model. We can linearize the above constraints by splitting the constraints into two pieces each where

$$\epsilon_{\phi}^{(ij)}[k] \leq M + b^{(ij)}[k](\nu_{\phi}^{(i)}/2 - M) \tag{23}$$
$$-\epsilon_{\phi}^{(ij)}[k] \leq M + b^{(ij)}[k](\nu_{\phi}^{(i)}/2 - M) \tag{24}$$
$$\epsilon_{\theta}^{(ij)}[k] \leq M + b^{(ij)}[k](\nu_{\theta}^{(i)}/2 - M) \tag{25}$$
$$-\epsilon_{\theta}^{(ij)}[k] \leq M + b^{(ij)}[k](\nu_{\theta}^{(i)}/2 - M). \tag{26}$$

As a final comment, even if we introduce these inequalities into an optimal control problem, we have not yet provided incentive for the system to satisfy them. The simplest way to accomplish this is to maximize the sum of the $b^{(ij)}[k]$. In order to achieve a value of one, rather than zero, for each variable forces the system to satisfy the field of view constraints. Of course, there are other, better ways to also accomplish this goal that we discuss below.

In order to obtain maximum coverage, we want to determine how many unique objects are viewed by all cameras and not just how many objects total are viewed. In order to determine this, let us introduce a set of new binary variables $\tilde{b}^{(j)}[k]$ that denote whether the object j can be viewed by any camera at time step k. Then, we introduce the relationship

$$\tilde{b}^{(j)}[k] \leq \sum_{i} b^{(ij)}[k]. \tag{27}$$

By maximizing some linear combination of the $\tilde{b}$, where we use positive weights, we create incentive for the system to satisfy the field of view inequalities, but we don't receive additional benefit when more than one camera views a particular object.

Finally, we have all the necessary pieces in order to formulate the problem in an optimal control formulation. In order to do this, we apply a receding horizon control which means that we plan ahead for a certain number of time steps and then execute the control for some number less than this. Each time we do this, we say we complete one plan and execution cycle. In order to determine the control for the planning period, we solve the following optimization problem

$$
\begin{array}{rl}
\displaystyle\max_{\phi,\theta,v_\phi,v_\theta,a_\phi,a_\theta,b,\tilde{b}} & \displaystyle\sum_{j=1}^{n}\sum_{k=1}^{p} w^{(j)}[k]\tilde{b}^{(j)}[k] \\
\text{st} & \displaystyle\tilde{b}^{(j)}[k] \le \sum_i b^{(ij)}[k] \\
\\
& o_\phi^{(ij)}[k] - \phi^{(i)}[k] \le M + b^{(ij)}[k](\nu_\phi^{(i)}/2 - M) \\
& -(o_\phi^{(ij)}[k] - \phi^{(i)}[k]) \le M + b^{(ij)}[k](\nu_\phi^{(i)}/2 - M) \\
& o_\theta^{(ij)}[k] - \theta^{(i)}[k] \le M + b^{(ij)}[k](\nu_\theta^{(i)}/2 - M) \\
& -(o_\theta^{(ij)}[k] - \theta^{(i)}[k]) \le M + b^{(ij)}[k](\nu_\theta^{(i)}/2 - M) \\
\\
& \phi^{(i)}[k+1] = \phi^{(i)}[k] + \Delta t v_\phi^{(i)}[k] + \frac{\Delta t^2}{2} a_\phi^{(i)}[k] \\
& \theta^{(i)}[k+1] = \theta^{(i)}[k] + \Delta t v_\theta^{(i)}[k] + \frac{\Delta t^2}{2} a_\theta^{(i)}[k] \\
& v_\phi^{(i)}[k+1] = v_\phi^{(i)}[k] + \Delta t a_\phi^{(i)}[k] \\
& v_\theta^{(i)}[k+1] = v_\theta^{(i)}[k] + \Delta t a_\theta^{(i)}[k] \\
\\
& \begin{array}{lll}
\phi^{(i)}[1] = \phi_0^{(i)} & v_\phi^{(i)}[1] = v_{\phi,0}^{(i)} & a_\phi^{(i)}[1] = a_{\phi,0}^{(i)} \\
\theta^{(i)}[1] = \theta_0^{(i)} & v_\theta^{(i)}[1] = v_{\theta,0}^{(i)} & a_\theta^{(i)}[1] = a_{\theta,0}^{(i)}
\end{array} \\
\\
& \begin{array}{ll}
\phi_L^{(i)} \le \phi^{(i)}[k] \le \phi_U^{(i)} & \theta_L^{(i)} \le \theta^{(i)}[k] \le \theta_U^{(i)} \\
v_{\phi L}^{(i)} \le v_\phi^{(i)}[k] \le v_{\phi U}^{(i)} & v_{\theta L}^{(i)} \le v_\theta^{(i)}[k] \le v_{\theta U}^{(i)} \\
a_{\phi L}^{(i)} \le a_\phi^{(i)}[k] \le a_{\phi U}^{(i)} & a_{\theta L}^{(i)} \le a_\theta^{(i)}[k] \le a_{\theta U}^{(i)}
\end{array}
\end{array}
\tag{28}
$$

where m denotes the number of cameras, n denotes the number of objects, and p denotes the number of time steps. We specify the domain of our variables as

$$
\begin{array}{rlrl}
\phi \in \mathbb{R}^{m\times p} & \text{Elevation position} & \theta \in \mathbb{R}^{m\times p} & \text{Azimuth position} \\
v_\phi \in \mathbb{R}^{m\times p} & \text{Elevation velocity} & v_\theta \in \mathbb{R}^{m\times p} & \text{Azimuth velocity} \\
a_\phi \in \mathbb{R}^{m\times p} & \text{Elevation acceleration} & a_\theta \in \mathbb{R}^{m\times p} & \text{Azimuth acceleration} \\
b \in \mathbb{B}^{m\times n\times p} & \text{Viewed by this camera} & \tilde{b} \in \mathbb{B}^{n\times p} & \text{Viewed by some camera.}
\end{array}
\tag{29}
$$

Similarly, we specify the domain of our constants as

$$
\begin{array}{rlrl}
o_\phi \in \mathbb{R}^{m\times n\times p} & \text{Local obj. elevation} & o_\theta \in \mathbb{R}^{m\times n\times p} & \text{Local obj. azimuth} \\
\nu_\phi \in \mathbb{R}^{m} & \text{View angle} & \nu_\theta \in \mathbb{R}^{m} & \text{View angle} \\
M \in \mathbb{R} & \text{Sufficiently large} & \Delta t \in \mathbb{R} & \text{Time step} \\
\phi_L \in \mathbb{R}^{m} & \text{Position bound} & \phi_U \in \mathbb{R}^{m} & \text{Position bound} \\
\theta_L \in \mathbb{R}^{m} & \text{Position bound} & \theta_U \in \mathbb{R}^{m} & \text{Position bound} \\
v_{\phi L} \in \mathbb{R}^{m} & \text{Velocity bound} & v_{\phi U} \in \mathbb{R}^{m} & \text{Velocity bound} \\
v_{\theta L} \in \mathbb{R}^{m} & \text{Velocity bound} & v_{\theta U} \in \mathbb{R}^{m} & \text{Velocity bound} \\
a_{\phi L} \in \mathbb{R}^{m} & \text{Acceleration bound} & a_{\phi U} \in \mathbb{R}^{m} & \text{Acceleration bound} \\
a_{\theta L} \in \mathbb{R}^{m} & \text{Acceleration bound} & a_{\theta U} \in \mathbb{R}^{m} & \text{Acceleration bound} \\
\phi_0 \in \mathbb{R}^{m} & \text{Initial orientation} & \theta_0 \in \mathbb{R}^{m} & \text{Initial orientation} \\
v_{\phi,0} \in \mathbb{R}^{m} & \text{Initial velocity} & v_{\theta,0} \in \mathbb{R}^{m} & \text{Initial velocity} \\
a_{\phi,0} \in \mathbb{R}^{m} & \text{Initial acceleration} & a_{\theta,0} \in \mathbb{R}^{m} & \text{Initial acceleration} \\
w \in \mathbb{R}_{++}^{n\times p} & \text{Observation weights} & &
\end{array}
\tag{30}
$$

where $\mathbb{R}_{++}$ denotes the positive real numbers. Finally, in order to calculate the object positions above, we require a preprocessing step where we compute

$$\delta_l^{(ij)}[k] = o_l^{(j)}[k] - c_l^{(i)}[k] \text{ for } l \in \{x, y, z\} \tag{31}$$

$$l^{(ij)}[k] = \sqrt{\delta_x^{(ij)}[k]^2 + \delta_y^{(ij)}[k]^2 + \delta_z^{(ij)}[k]^2} \tag{32}$$

$$\Delta_l^{(ij)}[k] = \delta_l^{(ij)}[k]/l \text{ for } l \in \{x, y, z\} \tag{33}$$

$$o_\phi^{(ij)}[k] = \arcsin(\Delta_y^{(ij)}[k]) - c_\phi^{(i)} \tag{34}$$

$$o_\theta^{(ij)}[k] = \text{arctan2}(\Delta_z^{(ij)}[k], \Delta_x^{(ij)}[k]) - c_\theta^{(i)} \tag{35}$$

where we have

$o_w \in \mathbb{R}^{n \times p}$	Global object position	$c_w \in \mathbb{R}^{n \times p}$	Global camera position
$c_\phi \in \mathbb{R}^m$	Fixed camera orientation	$c_\theta \in \mathbb{R}^m$	Fixed camera orientation

(36)

and $l \in \{x, y, z\}$.

3 Integration with a Kalman Filter

As mentioned before, we assume that we have an external mechanism that tells us the position of each object within the system. Certainly, technologies exist that reasonably locate objects moving within a system, but our model actually requires a prediction as to where these objects will move over the next planning step. In order to accomplish, we employ a Kalman filter. We use a Kalman filter for two reasons. First, their design and implementation is straight forward, fast, and well understood. Second, a Kalman filter gives us an estimate of the uncertainty in its prediction. We use this uncertainty in order to improve our model from before.

During the course of tracking an object j at time step k, a Kalman filter produces an *a priori* estimate covariance matrix, $P^{(j)}[k] \in \mathbb{R}^{3\times 3}$. As the amount of uncertainty in an estimate grows, this matrix grows as well. In order to distill this uncertainty into a single number, we have two options. First, we may compute

$$\kappa_m \det(P^{(j)}[k]^{-1/2}) \tag{37}$$

where κ_m denotes the area of the unit sphere in m dimensions. This quantity denotes the area of the uncertainty ellipsoid $\{x \in \mathbb{R}^m : x^T P^{(i)}[k]x \leq 1\}$. Hence, as the *a priori* estimate covariance matrix grows, this ellipse of uncertainty grows as well. As an alternative metric, we may compute the largest eigenvalue of the covariance matrix,

$$\lambda_{\max}(P^{(j)}[k]). \tag{38}$$

This quantity denotes the maximum distance from the origin to the edge of the ellipsoid described above. From a practical point of view, the second metric typically provides more valuable information. The largest eigenvalue of a matrix provides a bound on the determinant, and hence the area of the ellipsoid, but the converse is not true. Hence, in degenerate situations when the ellipsoid is very flat, but stretched in other directions, the largest eigenvalue still provides information that uncertainty in an object's position exists, but looking at the determinant does not. Finally, the largest eigenvalue is computationally cheaper to calculate than the determinant.

Based on the *a priori* estimate covariance matrix, we can immediate integrate information from the Kalman filter into the optimal control problem if we set the weights in the above problem to

$$w^{(j)}[k] = \lambda_{\max}(P^{(j)}[k]). \tag{39}$$

In this way, the system has more incentive to observe objects whose uncertainty is higher, or, more simply, objects that we have not viewed for some time. In affect, this scheme creates incentive for the cameras to observe all objects rather than focus on a select few.

Although this scheme improves the quality of our optimal control, it possesses a few problems that must be addressed. First, since we require an estimate of the uncertainty for all time steps in future, the weights above increase monotonically with time. If we solve the control problem for a large number of time steps, the problem may be poorly scaled and a solution more difficult to find. One possible resolution to this problem is to renormalize the above weights according to some scheme. For example, we could take the logarithm of the above weights. Alternatively, we could simply use the largest eigenvalue of the covariance matrix obtained at the last time step. In this way, objects not observed during previous plan and execution cycles would increase their observational weight between cycles, but not within the time scale of a single optimal control problem.

Second, since the Kalman filter and the optimal control problem are solved separately, the above model does not correctly account for when the system makes an observation. For example, if we observe an object at the third time step in our algorithm, we would expect the uncertainty corresponding to this object to be reduced. In the current formulation, this is not the case and there may be increased incentive to view this object at successive time steps. In fact, our current formulation only accounts for observations and updates the Kalman filter between the plan and execution cycles. In order to address this concern, we modify the above formulation by introducing additional constraints of the form

$$\sum_{k=1}^{p} \tilde{b}^{(j)}[k] \leq 1 \tag{40}$$

where p denotes the total number of time steps. This states that we receive benefit from observing an object only once during a particular plan and execution cycle. In addition, in conjunction with our above discussion on the weights derived from the Kalman filter, we set our weighting scheme to be

$$w^{(j)}[k] = (p + 1 - k)\lambda_{\max}(P^{(i)}[p]). \tag{41}$$

This creates increased incentive to observe objects with greater uncertainty in their position sooner in the planning cycle rather than later. It also avoids the scaling issues discussed above.

As a result, we see that the Kalman filter satisfies two desires. It not only allows us to predict the position of each object in the system, but it gives an indication of which objects have not been observed recently.

When we combine the MILP based control with the Kalman filter, we obtain a receding horizon control based on the following steps

1. Predict the position and the uncertainty in the position of each object for p time steps using the Kalman filter.
2. Solve the MILP described in the previous section using the weights

   $$w^{(j)}[k] = (p + 1 - k)\lambda_{\max}(P^{(j)}[p]).$$

   This gives a control for p time steps.
3. Execute the control for some number of time steps smaller than p.
4. During the execution of the control, continually update the *a posteriori* estimate of all objects in the system when possible.
5. Based on the current *a posteriori* estimates, return to step 1.

4 Algorithmic Complexity

Ideally, we would like to use our scheduling method within a real-time system. Therefore, we have three requirements of our algorithm in decreasing order of importance. First, it must always return a feasible solution. In other words, we can never accept a solution that violates the kinematics of our pan/tilt units. Second, we desire a solution that gives us good coverage of all objects found within our area of interest. Finally, we desire a solution that gives us optimal coverage of all objects found within our area of interest. We separate our last two requirements since, as we see below, it is very difficult to obtain an optimal solution within the time constraints that we require.

Our original formulation contains, $6mnp$ real variables and $(m+1)np$ binary variables where m denotes the number of cameras, n the number of objects, and p the number of time steps. Furthermore, we have $np+4mnp+n$ linear inequality constraints, $(p-1)m$ linear equality constraints, $12m(p-1)$ bound constraints, and $6m$ fixed variables. Since our problem contains a combination of discrete and continuous variables, but all functions involved are linear, we have modeled our scheduling problem as a mixed integer linear program (MILP.)

The advantage of modeling the scheduling problem as a MILP is that, given enough time, we can quantitatively measure the ideal performance of method. In other words, we have modeled not a heuristic, but a precise description of what we desire and the constraints involved. The disadvantage of this approach is that solving a MILP to optimality is an NP-hard problem. Furthermore, the large number of constraints involved mean that we have a very large NP-hard problem. As a result, it is unlikely that we can find an optimal solution during each planning steps.

Fortunately, the problem has good structure that we can exploit. The standard algorithm to solve a MILP is a process called branch, bound, and cut. The branch and bound process involves an intelligent combinatorial search over all possible integer solutions. By obtaining upper and lower bounds for the optimal solution, it becomes possible to prove that searching over certain combinations will never lead to an optimal solution. In this way, we search far fewer solutions than a full combinatorial search requires. A key part to this algorithm is the ability to obtain upper and lower bounds. An upper bound can always be obtained by solving a linear programming relaxation of the above problem. This can be done quickly. The lower bound generally requires a relatively expensive search for a feasible integer solution that we call an incumbent. In our case, we can obtain the incumbent for free. Simply, when b and $\tilde{b}$ are set to zero and all cameras do not move, we always have a feasible solution. Furthermore, the ability to quickly generate incumbent solutions means that we can always estimate the optimality gap. This gives us an estimate of how close our solution is to optimality. Based on these estimates, we can terminate the search for optimal solutions early and have an estimate for how good our schedule is. Finally, in addition to always having an incumbent solution, the search for an optimal solution is parallelizable. Therefore, we can devote considerable computational resources to the problem should it become necessary.

5 Computational Results

In the following numerical experiment, we track a variable number of objects with two cameras. Our goal in this exercise is to determine whether or not the error between the expected and actual object positions remains bounded or diverges. This gives us an idea of the robustness of the control algorithm.

We begin by placing two cameras randomly on a plane 100 units above the observation area where the position on this plane follows a two-dimensional normal distribution with mean $(0, 0)$ and variance $(30, 30)$. Then, we set the view angle for each camera to be a narrow $\pi/12$ radians (15 degrees). In terms of rotational speed, we set the maximum elevation and azimuthal angles to be $\pm 5\pi$, the maximum rotational velocities to be ± 10, and the maximum accelerations to be ± 100. We use dimensionless units, so these quantities represent the maximum the camera can rotate in a single unit of time.

Next, we create either 5, 10, 15, or 20 objects on a plane at 0 units above the observational area where the position of the objects are this plane follows a two-dimensional normal distribution with mean $(0, 0)$ and variance $(20, 20)$. Then, set set the objects to move along ballistic trajectories where the initial velocity of the object is given by (v_x, v_y, v_z). In this vector, v_x and v_z are normally distributed with mean 0 and variance 15. The quantity v_y is uniformly distributed between 30 and 60. In this way, each object initially moves in a random direction, but always up toward the cameras. In addition to the initial velocity, we apply an initial $(0, -9.8, 0)$ acceleration in order to simulate gravity.

Based on the above description, we represent the state of each object as

$$s = \begin{bmatrix} x \text{ position} \\ x \text{ velocity} \\ x \text{ acceleration} \\ y \text{ position} \\ y \text{ velocity} \\ y \text{ acceleration} \\ z \text{ position} \\ z \text{ velocity} \\ z \text{ acceleration} \end{bmatrix}. \tag{42}$$

This allows us to describe the state of each object at time k with the equation

$$s[k+1] = Ks[k] + \sum_{i=1}^{3} K_i x_i \tag{43}$$

where

$$K = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{44}$$

$$K_1 = \begin{bmatrix} \frac{1}{2}\Delta t^2 & \Delta t & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{45}$$

$$K_2 = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2}\Delta t^2 & \Delta t & 1 & 0 & 0 & 0 \end{bmatrix}^T \tag{46}$$

$$K_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2}\Delta t^2 & \Delta t & 1 \end{bmatrix}^T, \tag{47}$$

x_i are normally distributed random variables with mean 0 and variance 0.1, and Δt is 0.02. As a result, the objects move in standard ballistic trajectories, but the acceleration of the object is varied randomly in all directions. Finally, once an object reaches an elevation of 0, we remove the object and create a new object according to the scheme described above.

With respect to the Kalman filter, we assume that we can only observe each object's position, not its velocity and acceleration. As a result, we use an observation matrix

$$O = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{48}$$

In addition, we assume that we contain errors in our observations that are normally distributed with mean 0 and variance 10^{-3}.

Finally, we implement a receding horizon control based on the mixed integer linear programming formulation above. In this control, we plan for 30 time steps, but execute the control for only 15. In order to solve the MILP, we use GLPK [15] where we limit the time to solve the integer program to 10 seconds. When 10 second are completed, we take the best known incumbent solution as the control.

In the experiment, we simulate the system for a total of 500 time steps. A visualization of this experiment can be seen in Figure 2. Then, at each time step, we collect four metrics. First, we measure the average error in the predicted and actual object positions measured by

$$\frac{1}{n}\sum_{i=1}^{n} \frac{\|x_i^{(pred)} - x_i^{(actual)}\|_2}{1 + \|x_i^{(actual)}\|_2}. \tag{49}$$

where x_i denotes the position of the ith object in global (x, y, z) coordinates. Next, we measure the maximum error in the position at each time step measured by

$$\max_{i=1,\dots,n} \{\|x_i^{(pred)} - x_i^{(actual)}\|_2\}. \tag{50}$$

In a similar manner, we measure the average over the maximum eigenvalue of the *a priori* estimate of the covariance matrices from the Kalman filter

$$\frac{1}{n}\sum_{i=1}^{n} \lambda_{max}(P_i) \tag{51}$$

where P_i denotes the *a priori* estimate of the covariance matrix of the ith object. We also measure the maximum eigenvalue over all covariance matrices

$$\max_{i=1,\dots,n} \{\lambda_{max}(P_i)\}. \tag{52}$$

Then, we can see the performance of our simulation according to the above metrics in Figures 3, 4, 5, and 6.

These results indicate that the overall error and uncertainty between the predicted position of the objects in the system and the actual position remains small and bounded for 5, 10, and 15 objects, but not for 20. At 20 objects, both the error and the uncertainty in the objects' position becomes unbounded. Of course, this does not mean that we can not adequately observe a system with 20 objects. Rather, at 20 objects, 10 seconds is not enough time for the integer programming solver to improve the initial incumbent solution. If we allow the algorithm to run for more time, or use a higher performance integer programming solver, we observe the objects better.

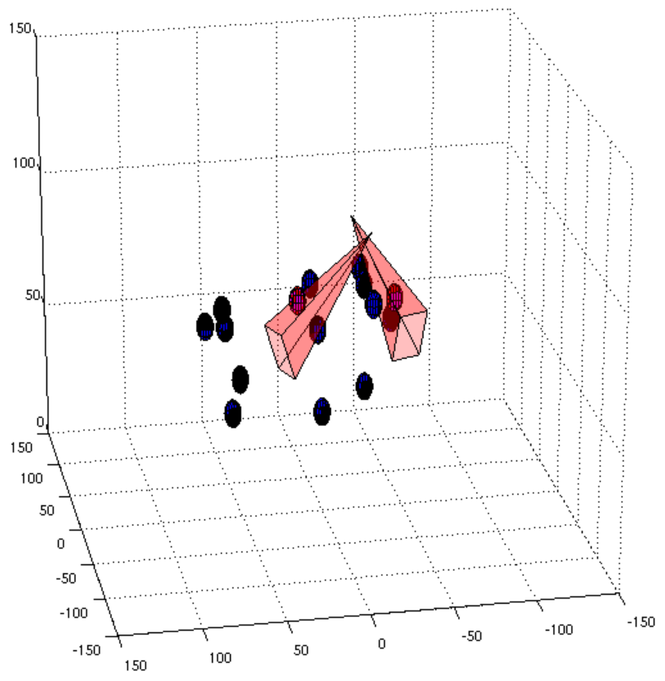


Figure 2. A visual depiction of the numerical experiment. Here, we use 2 cameras to track 15 objects moving in random ballistic trajectories. The translucent red pyramids denote the field of view of the two cameras whereas the red spheres denote observed objects, blue denote actual object positions, and black denote predicted object positions.

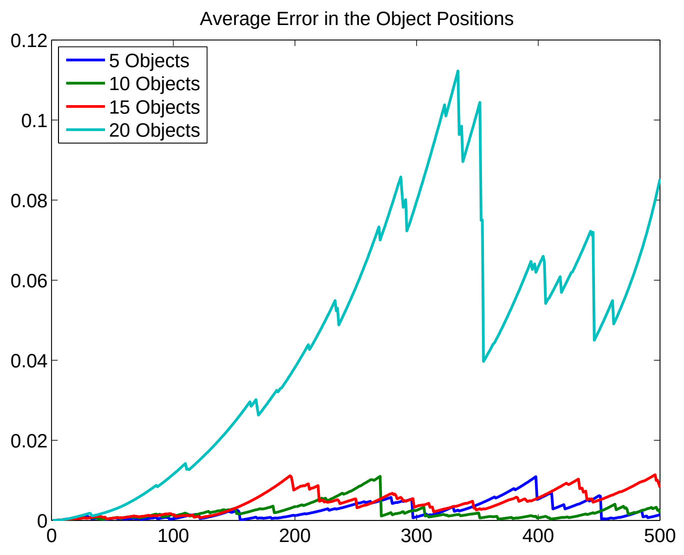


Figure 3. Average error in the objects' position measured by $\frac{1}{n}\sum_{i=1}^{n}\frac{\|x_i^{(pred)}-x_i^{(actual)}\|_2}{1+\|x_i^{(actual)}\|_2}$. Here, we see that the error remains relatively small and bounded for 5, 10, and 15 objects. However, the error grows unacceptably high for 20 objects.

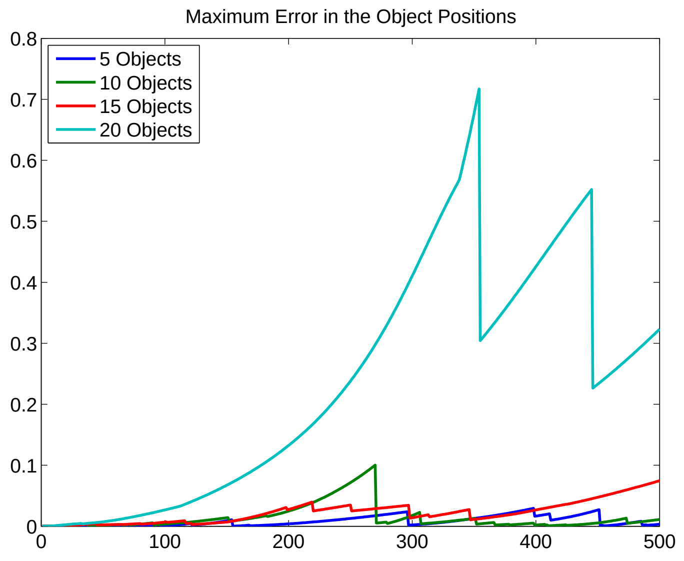


Figure 4. Maximum error in the objects' position measured by $\max\limits_{i=1,\ldots,n}\{\|x_i^{(pred)}-x_i^{(actual)}\|_2\}$. These results mirror those from the average error in the objects' position, which indicates that the overall error in the system is dominated by a small number of outliers. When these outliers become observed, the error in the system drops.

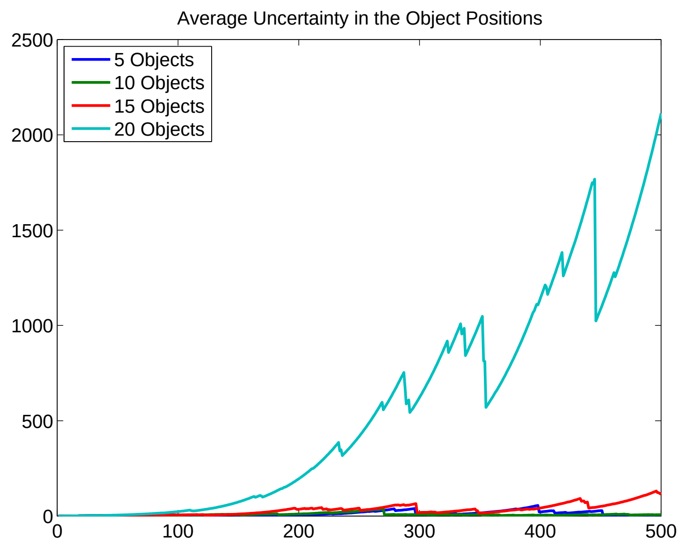


Figure 5. Average uncertainty in the objects' position measured by $\frac{1}{n}\sum_{i=1}^{n}\lambda_{max}(P_i)$ As with the average error in the objects' position, we see that the uncertainty in the system remains bounded for 5, 10, and 15 objects, but does not for 20 objects.

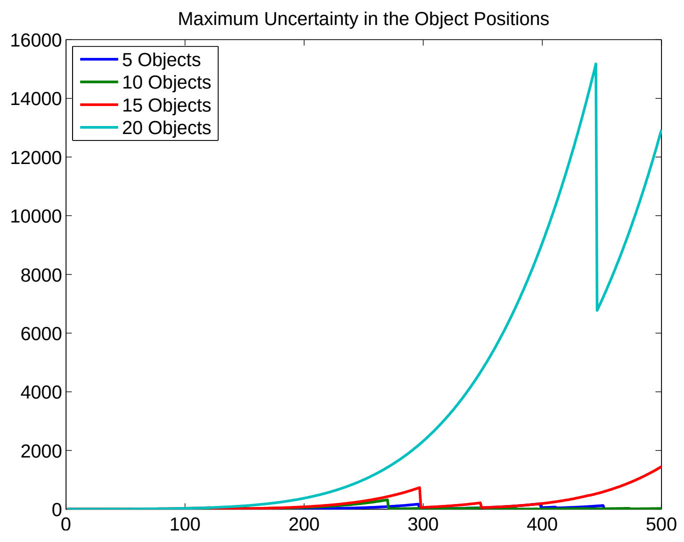


Figure 6. Maximum uncertainty in the objects' position measured by $\max_{i=1,\ldots,n}\{\lambda_{max}(P_i)\}$. These results mirror those from the average uncertainty in the objects' position. This indicates that a small number of outliers dominate the uncertainty in the system.

6 Conclusion

During the course this report, we have described an algorithm for the cooperative control of several cameras that are used to observe some area. This algorithm consists of three parts. First, we use a Kalman filter to predict the motion of all objects of interest as well as estimate the uncertainty in their position. Second, we solve a mixed integer linear program in order to determine the camera's optimal control. This formulation depends both on the kinematics of the cameras as well as the uncertainty estimates provided by the Kalman filter. Finally, we execute our control which allows us to make actual observations of the objects in the system. This allows us to update the Kalman filter and then solve for a new control.

Although seemingly difficult to solve, we have also discussed the algorithmic difficulties of the problem and discovered that it possesses good structure. Specifically, we can always generate an incumbent solution to the mixed integer linear program which gives us the ability to generate both upper and lower bounds to the solution in a quick and efficient manner. This allows us to not only find good controls quickly, but allows us to estimate the quality of these solutions.

In practice, the algorithm appears to perform well. The objects of interest are well observed and the number of outliers remains small. When the performance of the algorithm deteriorates, the lack of performance is directly due to not solving the integer program to optimality. Simply, we limit the amount of time used to solve the integer program and the amount of time required increases with the number of objects we track. If we increase the amount of computational effort in solving the integer program, the system performs better.

In short, our algorithm possesses the following three advantages. First, the algorithm is not heuristic; it models and solves a formal optimization problem. Second, we combine our overall goal of observation with the kinematics of our system into a single formulation that simultaneously considers all goals and constraints. Finally, we provide a tight coupling between our predictive mechanism, the Kalman filter, and our control algorithm which must account for future action. In this manner, we can effectively coordinate the actions between multiple cameras in order to observe an area of interest.

References

[1] Project ACONE: Automated birdwatching, 2007. http://www.c-o-n-e.org/acone/.

[2] T.A. Ademoye and A. Davari. Trajectory planning for multiple autonomous systems using mixed integer linear programming. In *System Theory, 2006. SSST '06. Proceeding of the Thirty-Eighth Southeastern Symposium on*, pages 175 –179, 2006.

[3] J. Bellingham, A. Richards, and J.P. How. Receding horizon control of autonomous aerial vehicles. In *American Control Conference, 2002. Proceedings of the 2002*, volume 5, pages 3741 – 3746 vol.5, 2002.

[4] J.S. Bellingham, M. Tillerson, M. Alighanbari, and J.P. How. Cooperative path planning for multiple uavs in dynamic and uncertain environments. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 3, pages 2816 – 2822 vol.3, December 2002.

[5] B.A. Boghossian and S.A. Velastin. Motion-based machine vision techniques for the management of large crowds. In *Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on*, volume 2, pages 961 –964 vol.2, September 1999.

[6] Rick Cavallaro. The foxtrax hockey puck tracking system. *IEEE Computer Graphics and Applications*, 17:6–12, March 1997.

[7] Evaristo Cisbani, Antonio Bartoloni, Marco Marchese, Gianluca Efisei, and Antonello Salvati. Early fire detection system based on multi-temporal images of geostationary and polar satellites. In *Geoscience and Remote Sensing Symposium, 2002. IGARSS '02. 2002 IEEE International*, volume 3, pages 1506 – 1508 vol.3, 2002.

[8] Mario Costantini, Massimo Zavagli, Evaristo Cisbani, and Bruno Greco. A technique for automatic fire detection from geostationary optical sensors and its validation on msg seviri data. In *Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. IEEE International Conference on*, pages 4153 –4156, August 2006.

[9] Norris L. Dodd, Jeffrey W. Gagnon, Amanda L. Mazo, and Raymond E. Schweinsburg. Video surveillance to assess highway underpass use by elk in arizona. *Journal of Wildlife Management*, 71(2):637–645, 2007.

[10] M.G. Earl and R. D'Andrea. Modeling and control of a multi-agent system using mixed integer linear programming. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, pages 107 – 111 vol.1, December 2002.

[11] L.M. Fuentes and S.A. Velastin. From tracking to advanced surveillance. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 3, pages III – 121–4 vol.2, September 2003.

[12] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82 – 98, 1999.

[13] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis. A survey of video processing techniques for traffic applications. *Image and Vision Computing*, 21(4):359 – 381, 2003.

[14] Wei-Lwun Lu, Kenji Okuma, and James J. Little. Tracking and recognizing actions of multiple hockey players using the boosted particle filter. *Image and Vision Computing*, 27(1-2):189 – 205, 2009. Canadian Robotic Vision 2005 and 2006.

[15] Andrew Makhorin. GNU linear programming kit, version 4.44, 2010. `http://www.gnu.org/software/glpk/glpk.html`.

[16] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[17] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision Image and Understanding*, 104:90–126, November 2006.

[18] Richard M. Murray. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):571–583, 2007.

[19] Christian Reinl and Oskar von Stryk. Optimal control of multi-vehicle-systems under communication constraints using mixed-integer linear programming. In *Proceedings of the 1st international conference on Robot communication and coordination*, RoboComm '07, pages 3:1–3:8, Piscataway, NJ, USA, 2007. IEEE Press.

[20] A. Richards and J. How. Mixed-integer programming for control. In *American Control Conference, 2005. Proceedings of the 2005*, pages 2676 – 2683 vol. 4, June 2005.

[21] A. Richards and J.P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, pages 1936 – 1941 vol.3, 2002.

[22] Arthur Richards, Jonathan How, Tom Schouwenaars, and Eric Feron. Plume avoidance maneuver planning using mixed integer linear programming. In *Proceedings of AIAA Guidance Navigation and Control Conference*, 2001. AIAA-2001-4091.

[23] E. Rowe. The Los-Angeles automated traffic surveillance and control (atsac) system. *Vehicular Technology, IEEE Transactions on*, 40(1):16 –20, February 1991.

[24] T. Schouwenaars, E. Feron, and J. How. Multi-vehicle path planning for non-line of sight communication. In *American Control Conference, 2006*, page 6 pp., June 2006.

[25] Tom Schouwenaars, Bart De Moor, Eric Feron, and Jonathan How. Mixed integer programming for multi-vehicle path planning. In *Proceedings of the European Control Conference*, pages 2603–2608, September 2001.

[26] Alan Schwarz. Digital eyes will chart baseball's unseen skills. *The New York Times*, July 2009. `http://www.nytimes.com/2009/07/10/sports/baseball/10cameras.html`.

[27] Dezhen Song, Ni Qin, Yiliang Xu, Chang Young Kim, D. Luneau, and K. Goldberg. System and algorithms for an autonomous observatory assisting the search for the ivory-billed woodpecker. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pages 200 –205, August 2008.

[28] J. Thunberg and P. Öandgren. An iterative mixed integer linear programming approach to pursuit evasion problems in polygonal environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5498 –5503, May 2010.

[29] Jens Wawerla, Shelley Marshall, Greg Mori, Kristina Rothley, and Payam Sabzmeydani. Bearcam: automated wildlife monitoring at the arctic circle. *Machine Vision and Applications*, 20:303–317, 2009. 10.1007/s00138-008-0128-0.

DISTRIBUTION:

1 MS 0899
RIM-Reports Management, 9532
(electronic copy)

Sandia National Laboratories