SYSTEMS AND ALGORITHMS FOR AUTOMATED COLLABORATIVE OBSERVATION USING NETWORKED ROBOTIC CAMERAS

A Dissertation

by

YILIANG XU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2011

Major Subject: Computer Science

SYSTEMS AND ALGORITHMS FOR AUTOMATED COLLABORATIVE OBSERVATION USING NETWORKED ROBOTIC CAMERAS

A Dissertation

by

YILIANG XU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Dezhen Song
Committee Members,	Nancy Amato
	Jianer Chen
	Tamás Kalmár-Nagy
Head of Department,	Hank Walker

August 2011

Major Subject: Computer Science

ABSTRACT

Systems and Algorithms for Automated Collaborative Observation Using Networked Robotic Cameras. (August 2011)

Yiliang Xu, B.S., Zhejiang University; Ph.D., Nanyang Technological University

Chair of Advisory Committee: Dr. Dezhen Song

The development of telerobotic systems has evolved from Single Operator Single Robot (SOSR) systems to Multiple Operator Multiple Robot (MOMR) systems. The relationship between human operators and robots follows the master-slave control architecture and the requests for controlling robot actuation are completely generated by human operators.

Recently, the fast evolving advances in network and computer technologies and decreasing size and cost of sensors and robots enable us to further extend the MOMR system architecture to incorporate heterogeneous components such as humans, robots, sensors, and automated agents. The requests for controlling robot actuation are generated by all the participants. We term it as the MOMR++ system. However, to reach the best potential and performance of the system, there are many technical challenges needing to be addressed. In this dissertation, we address two major challenges in the MOMR++ system development.

We first address the robot coordination and planning issue in the application of an autonomous crowd surveillance system. The system consists of multiple robotic pan-tilt-zoom (PTZ) cameras assisted with a fixed wide-angle camera. The wide-angle camera provides an overview of the scene and detects moving objects, which are required for close-up views using the PTZ cameras. When applied to the pedestrian surveillance application and compared to a previous work, the system achieves increasing number of observed objects by over 210% in heavy traffic scenarios. The key issue here is given

the limited number (e.g., p $(p > 0)$) of PTZ cameras and many more (e.g., n $(n \gg p)$) observation requests, how to coordinate the cameras to best satisfy all the requests. We formulate this problem as a new camera resource allocation problem. Given p cameras, n observation requests, and ϵ being approximation bound, we develop an approximation algorithm running in $O(n/\epsilon^3 + p^2/\epsilon^6)$ time, and an exact algorithm, when $p = 2,$ running in $O(n^3)$ time.

We then address the automatic object content analysis and recognition issue in the application of an autonomous rare bird species detection system. We set up the system in the forest near Brinkley, Arkansas. The camera monitors the sky, detects motions, and preserves video data for only those targeted bird species. During the one-year search, the system reduces the raw video data of 29.41TB to only 146.7MB (reduction rate 99.9995%). The key issue here is to automatically recognize the flying bird species. We verify the bird body axis dynamic information by an extended Kalman filter (EKF) and compare the bird dynamic state with the prior knowledge of the targeted bird species. We quantify the uncertainty in recognition due to the measurement uncertainty and develop a novel Probable Observation Data Set (PODS)-based EKF method. In experiments with real video data, the algorithm achieves 95% area under the receiver operating characteristic (ROC) curve.

Through the exploration of the two MOMR++ systems, we conclude that the new MOMR++ system architecture enables much wider range of participants, enhances the collaboration and interaction between participants so that information can be exchanged in between, suppresses the chance of any individual bias or mistakes in the observation process, and further frees humans from the control/observation process by providing automatic control/observation. The new MOMR++ system architecture is a promising direction for future telerobtics advances.

DEDICATION

To my parents and Yanliang

ACKNOWLEDGMENTS

I extend my gratitude to those who helped me get through my Ph.D. program at Texas A&M.

Special thanks to Dezhen Song for patiently guiding me during the last five years. Dez is my advisor and my best friend. Dez is always willing to spend time with me and inspire me to explore new problems. Not only in research, but also other aspects of life, Dez set a great model for me to follow.

Thanks to Nancy Amato, Jianer Chen, and Tamás Kalmár-Nagy for serving as my dissertation committee members and their insightful suggestions and inspiring discussions. I am fortunate to have such great committee.

Thanks to Jingang Yi and Frank van der Stappen for being great research collaborators. Thanks to Changyoung Kim for being my great roommate, labmate and friend.

Thanks to N. Qin, H. Lee, H. Wang, J. Zhang, Z. Bing, W. Li, Y. Lu for their continuous help and contributions to my Ph.D. projects.

TABLE OF CONTENTS

Page

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

1. INTRODUCTION

A telerobot is a robot remotely controlled by a human operator for interacting with a remote physical environment [1]. In most practice, the high-level robot planning or cognitive decisions are by the human operator while the robot is responsible for the mechanical implementation. The research on telerobotics has been an important aspect in the field of robotics for a long history and it has found many applications such as space exploration [2], health care [3], and natural observation [4].

In traditional telerobotic systems, a human operator and a robot communicate by transmitting control commands and state feedback through a dedicated communication medium. According to the taxonomy proposed by Chong *et al.* [5], this class of systems belongs to Single Operator Single Robot (SOSR) systems. Most existing telerobotic systems can be modeled by this master-slave architecture. In 1898, Nicola Tesla [6] first demonstrated a radio-controlled vessel in the New York City. In 1950's, Goertz [7] developed systems which are directly controlled by human to handle radioactive materials behind shield walls. However, under this architecture, the control commands and decisions are made by individual human operator. Therefore, the quality of the control and operation is significantly affected by the individual human operator, which limits its accessibility to only trained specialists and experts.

The development of network technology allows a new communication medium between the local control site and the remote robot site and thus opens up new possibilities in system architecture. In 1994, the telegarden [8] (Fig.1.1) became the first robot on web that allows World Wide Web (WWW) users to control a remote robot in a garden filled with living plants. Since the system queues the users to access the robot sequentially, it essentially still belongs to SOSR category through it greatly extends system's accessibility to general public including both amateurs and experts through WWW.

This dissertation follows the style of *IEEE Transactions on Image Processing*.

Using computer network as the communication medium, Song [4] extended the system architecture by including multiple human operators to access the robot simultaneously. The control commands of the robot are generated by combining the the requests for control from multiple operators. In [9], even a human actor with cameras and microphones, replace the role of robot and navigates and performs actions in the remote environment, collaboratively controlled by multiple online users. This class of systems is categorized as Multiple Operator Single Robot (MOSR) systems. MOSR architecture especially with network as communication medium greatly extends system's accessibility to general public. Furthermore, since multiple human users share the control of the robot, it helps improve the system's reliability due to the collaboration between users. However, since there is only one robot, the system's capability especially in tasks such as search, surveillance, and exploration etc. is limited.

Recently, Multiple Operator Multiple Robot (MOMR) systems have emerged [10, 11]. It allows multiple human operators to control multiple (heterogeneous) robots. For example, Liu and his colleagues [12], developed a competitive MOMR system with two robotic arms controlled by two human users respectively under a game setting. However, since most existing systems still allow only one operator to directly control one robot, it is still the master-slave control. Also the robots control commands are still based on human inputs.

1.1 MOMR++: Networked Automated and Collaborative Observation System

The limits of the existing systems motivate us to extend the system architecture. Recently, the fast evolving advances in network and computer technologies and decreasing size and cost of sensors and robots enable us to further extend the MOMR system architecture to incorporate heterogeneous participating components such as humans, robots, sensors, and automated agents. It allows multiple human users, sensors, automated agents to access multiple robots. The requests for controlling robot actuation are generated by all the participants. We term it as MOMR++ system. The MOMR++ systems consist of:

Fig. 1.1. Telegarden [8] is the first robot on web that allows World Wide Web (WWW) users to control a remote robot in a garden filled with living plants.

- Participants: humans, sensors, and automated agents.
- Network communication medium
- Web server: a computer running a web server software.
- Robots.

Fig. 1.2 shows an example of MOMR++ system: networked automated and collaborative observation system. The system allows multiple internet users to observe the remote physical environment using multiple robotic cameras. Users designate regions of interest by drawing rectangles in the display of the scene. Meanwhile, automatic agents may request to periodically check certain regions and the optical and/or wireless sensors detect motions and anomalies and may request the cameras to follow up. Therefore, control commands of the robotic cameras are generated by combining all these observation requests from heterogeneous participants instead of just human inputs. This class of systems has many applications such as surveillance, nature observation, education, journalism, and entertainment.

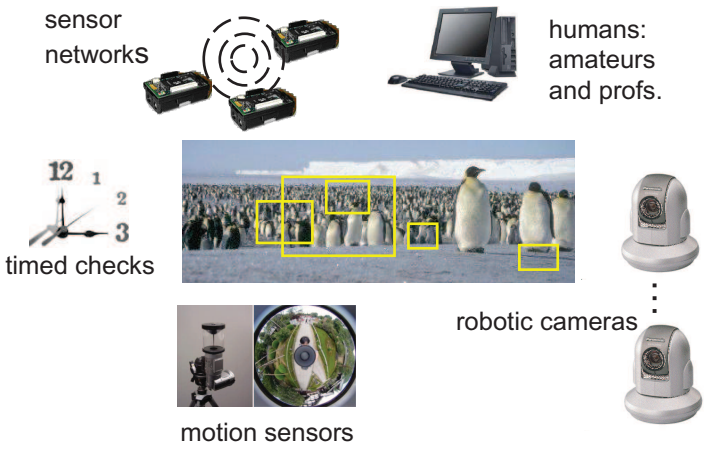


Fig. 1.2. The networked automated and collaborative observation system extends the traditional telerobotic system architecture to incorporate heterogeneous participating components such as humans, robots, sensors, and automated agents.

The new system enables much wider range of participants, such as a large group of students or researchers, to access to valuable resources (e.g., expensive robotic cameras), so that the utility of the these resources are improved. It also enhances the collaboration and interaction between participants so that information can be exchanged in between, which is very useful in learning and education domains. Since the cameras' control are shared with all participants, it suppresses the chance of any individual bias or mistakes in the observation process. Since automatic agents and sensor can control the robot and perform the observation task without human inputs, it further frees humans from the observation process by providing automatic observation. It also reduces the human workload by reducing the volume of data that requires human verification, since only interesting content is preserved by automated agents and sensors.

However, by incorporating various heterogeneous components, the relationship between these components becomes more complicated. The conventional master-slave ar-

chitecture no longer holds. For example, human users could compete for or collaboratively share the control of robots; automated agents or even sensors may take over the control of the robots and complementarily execute the observation tasks for humans. As a result, to reach the best potential and performance of the system, there are many technical challenges need to be addressed. In this dissertation, we address two major challenges in MOMR++ system development.

1.1.1 MOMR++: Autonomous Crowd Surveillance System

We first address the robot coordination and planning issue in the application of an autonomous crowd surveillance system. The system consists of multiple robotic pan-tilt-zoom (PTZ) cameras assisted with a fixed wide-angle camera. The wide-angle camera provides an overview of the scene and detects moving objects, which are considered as objects of interests. Based on the output of the wide angle camera, the system generates spatiotemporal observation requests for each object, which are candidates for close-up views using the PTZ cameras. The system computes the control commands for the PTZ cameras to track and observe the objects of interest by computing the optimal PTZ cameras' frames that best satisfy these observation requests. We implement the system and test it for pedestrian surveillance in a university campus environment. When compared to a previous work, the system achieves increasing number of observed objects by over 210% in heavy traffic scenarios.

The key issue in the autonomous crowd surveillance system is given limited number (e.g., p $(p > 0)$) of PTZ cameras and much more (e.g., n $(n \gg p)$) observation requests, how to coordinate the cameras to best satisfy all the requests. we formulate the camera planning problem as a new camera resource allocation problem. We propose a new similarity metric to measure the degree of satisfaction for each request. We focus on the development of scalable fast algorithms to solve this problem. We develop an approximation algorithm with guaranteed approximation bound, which provides tradeoff between the solution quality and speed. Given p cameras, n observation requests, and ϵ being ap-

proximation bound, the algorithm runs in $O(n/\epsilon^3 + p^2/\epsilon^6)$ time. We also develop an exact algorithm when $p = 2$, which runs in $O(n^3)$ time. This algorithm addresses the online computation requirements and fits many real-life applications.

We report this autonomous crowd surveillance system and algorithm development in Sections 2, 3, 4.

1.1.2 MOMR++: Rare Bird Species Detection System

We then address the automatic object content analysis and recognition issue in the application of an autonomous rare bird species detection system. We set up the system in the forest near Brinkley Arkansas for searching the thought-to-be-extinct ivory-billed woodpeckers. The camera monitors the sky, detects motions, and preserves video data for only those of targeted bird species. Without the human inputs, the system needs to autonomously distinguish and recognize the targeted object (i.e, the bird species here) from other moving objects and environmental noises and only preserves the interesting information for human verification. The system runs continuously from October 2006 to October 2007. During the one-year search, the system reduces the raw video data of 29.41TB to only 146.7MB (reduction rate 99.9995%).

The key issue in the bird detection system it to automatically recognize the flying bird species. We verify the bird body axis dynamic information by an extended Kalman filter (EKF), and compare the bird dynamic state such as body axis length and flying speed with the prior knowledge of the targeted bird species. However, due to significant measurement data noise and insufficient measurement data volume, a regular EKF fails to converge. To resolve this issue, we quantify the uncertainty in recognition due to the measurement uncertainty and develop a novel Probable Observation Data Set (PODS)-based EKF method. The new PODS-EKF algorithm searches the measurement error range for all probable observation data that ensures the convergence of the corresponding EKF, which guarantees to bound the true (noise-free) bird state. We then formulate the recognition problem as an optimization problem which searches in the PODS for the most likely observation

corresponding to the true (noise-free) bird state. In experiments with real video data, the algorithm achieves 95% area under the receiver operating characteristic (ROC) curve.

We report this bird detection system and corresponding algorithm development in Sections 5 and 6.

2. MOMR++ SYSTEM: AUTONOMOUS CROWD SURVEILLANCE SYSTEM

In this section, we report a example of the MOMR++ system: an autonomous crowd surveillance system with multiple pan-tilt-zoom (PTZ) cameras assisted by a fixed wide-angle camera. Consider a wide-angle camera is installed at an airport for human activity surveillance or in a forest for wildlife observation. The wide-angle camera can provide large, low resolution coverage of the scene. However, recognition and identification of humans and animals usually require close-up views at high resolution which needs PTZ cameras. The resulting autonomous observation system consists of a fixed wide-angle camera with multiple PTZ cameras as illustrated in Fig. 2.1. The wide-angle camera monitors the entire field to detect and track all moving objects. Each PTZ camera selectively covers a subset of the objects.

However there are usually more moving objects than the number of PTZ cameras. With these competing spatiotemporal observation requests, the major challenge is the control and scheduling of the PTZ cameras to maximize the "satisfaction" to the competing requests. The system design emphasizes the "satisfaction" to the requests which takes into account the 1) camera coverage over objects, 2) camera zoom level selection, and 3) camera traveling time. We approach the control and scheduling problem in two steps. First, a subset of the requests/objects is assigned to each PTZ camera. Second, each PTZ camera selects its PTZ parameters to cover the assigned objects. We formulate the problems in both steps as frame selection problems. We propose an approximation algorithm as in Section 3, and an exact algorithm as in Section 4 to solve them in real time. We implement the system and validate it in simulations and physical experiments. The experimental results show that our system outperforms an existing work by increasing the number of observed objects by over 210% in heavy traffic scenarios.

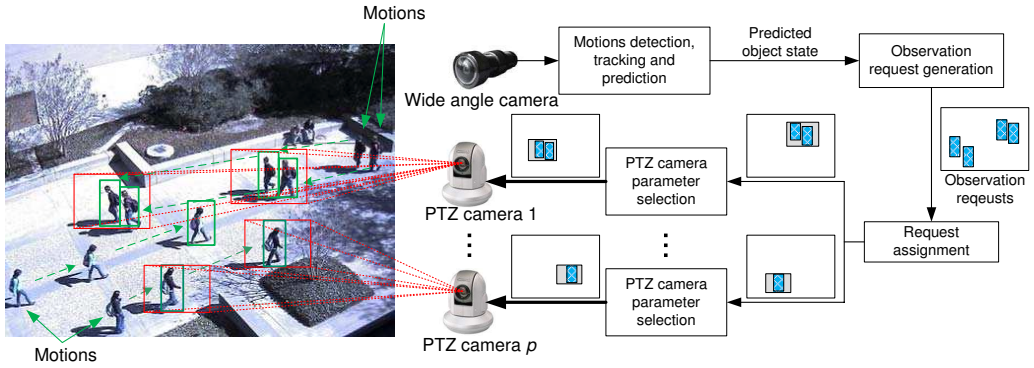


Fig. 2.1. System architecture. The solid green rectangles represent the moving objects and the dashed red rectangles indicate the selective coverage of the PTZ cameras.

2.1 Related Work

The proposed autonomous observation system relates to the existing works on motion detection and tracking, and multiple and active camera surveillance systems.

Our system critically relies on the motion detection and object tracking techniques in computer vision. Motion detection involves in detecting the moving objects and segmenting them out of the background from a video sequence in the same scene. To address the noise and changes in background, various background models have been proposed. Examples include temporal average [13], mean average deviation (MAD) [14], mixed Gaussian model [15], adaptive Gaussian estimation [16, 17], non-parametric model [18], Kalman filter compensation [19], and texture-based model [20]. A recent survey on motion detection can be found in [21]. Motion tracking usually builds on the motion detection. It predicts the trajectory of the objects by locating their position in every frame of the video sequence. Based on the representation of the object, existing tracking technologies can be categorized as point tracking [22], kernel tracking [23] and silhouette tracking [24]. A variety of fundamental techniques have been proposed for tracking, such as support vector machine (SVM) [25], active contour evolution [26] and Hough transform [27] etc. A comprehensive survey on object tracking can be found in [28]. Recently, due to its flexible

field of view (FOV) and variable resolution, pan-tilt-zoom (PTZ) camera has been used for tracking purpose. Unlike the work in [29], where each PTZ camera tracks only single object, our diagram is able to use only single PTZ to cover the multiple moving objects simultaneously.

In the recent decade, multiple camera surveillance systems, especially those with both static and active cameras have attracted growing attention of research. Most of the works are master-slave camera configuration [30]. The master static camera(s) provide the general information about the wide-angle scene while the slave active cameras acquire the localized high-resolution imagery of the regions of interest. This is a relatively new research area with many directions to explore. Early representative works include Stillman *et al.* [31], which addresses the camera-object assignment problem and Greiffenhagen *et al.* [32], which proposes a dual camera surveillance system consisting of a ceiling mounted omnidirectional camera and a PTZ camera. Our work belongs to this category.

Most works in this category schedule the active cameras based on straightforward heuristic rules. Zhou *et al.* [30] choose the object closest to the current camera setting as the next observation object. Hampapur *et al.* [33] adopt the simple round robin sampling. Bodor *et al.* [34] and Fiore *et al.* [35] propose a dual-camera system with one wide-angle static camera and a PTZ camera for pedestrian surveillance. Human activities (walking, running, etc.) are prioritized based on the preliminary recognition by the wide-angle camera. The PTZ camera focuses to the activity with the highest priority for further analysis. Costello *et al.* [36] are the first to formulate the single camera scheduling problem based on network packet scheduling methods. The authors propose and compare several greedy scheduling policies. With different assumptions towards the observation scene and objects, various scheduling formulation and schemes are proposed. In Lim *et al.* [37], the scheduling problem is formulated as a graph matching problem. In Bimbo and Pernici [38], the continuous scheduling problem is truncated by a predefined observation deadline and each truncated camera scheduling problem is formulated as an online dynamic vehicle routing problem (DVRP). Qureshi and Terzopoulos [39] propose a virtual environment simula-

tor to test various camera senor network frameworks. However these methods assign only one object to one active camera. Our system assigns multiple objects to individual cameras by selecting PTZ camera parameters such that the camera coverage-resolution tradeoff is achieved. This also enables group watching with scalability.

Very few work considers the selection of the zoom level of active cameras and assigns multiple objects to individual cameras. Lim *et al.* [40] construct the observation task for each single object as a "task visibility interval" (TVI) based on its predicted states and corresponding camera settings. When TVIs have non-empty intersection, they are grouped to form a "multiple task visibility interval" (MTVI). Based on the order of the starting time of (M)TVIs, a directed acyclic graph (DAG) is constructed. The scheduling problem is formulated as a maximal flow problem. A greedy algorithm and a dynamic programming scheme are proposed to solve it. Zhang *et al.* [41] construct a semantic saliency map to indicate the observation requests. An exhaustive algorithm finds the optimal single frame that minimizes the information loss. Sommerlade and Reid [42] use an information-theoretic framework to study how to select a single active camera's zoom level for tracking a single object to balance the chances of loosing the tracked object and that of loosing trace of other objects. In contrast to these works, our approach dose not require accurate long-term motion prediction. The assignment of multiple objects to individual PTZ cameras is carried out by selecting the camera parameters to achieve the tradeoff between coverage and resolution.

Evaluations of these scheduling strategies are usually done by simulation. Qureshi and Terzopoulos [43] propose a virtual environment simulator to test various camera senor network frameworks. Other related works on active camera sensor network include [44,45] which addresses the automatic calibration in the hybrid camera network. [46] address how to determine active camera settings based on predicted object motion.

In contrast to the existing works in this category, we propose a framework that supports each PTZ camera to cover multiple objects simultaneously. We formulate the camera scheduling problem as a sequence of frame selection problem so that the overall sat-

isfaction to the observation requests is maximized over time. We don't require accurate long-term prediction of object motion and camera calibration. It will be shown that our formulation of the problem is essentially the generalization of many existing works.

Our group focuses on developing intelligent vision systems and algorithms using robotic cameras for a variety of applications such as construction monitoring, distance learning, panorama construction and natural observation [4]. In the context of using PTZ camera for the collaborative observation, competing observation requests need to be covered by camera frame(s) to maximize the overall observation reward. This issue is formulated as a single frame selection (SFS) problem [47]. A series of algorithms for the single frame selection problem have been proposed [47, 48]. Song *et al.* [49] propose an autonomous observation system in which a single PTZ camera is used to fulfill competing spatiotemporal observation requests. In this section, multiple PTZ cameras are used to increase the observation coverage. We formulate the problem of coordinating the p camera frames as the p-frame problem and propose an approximation algorithm and an exact algorithm in the next two sections for solving it.

2.2 System Overview

Fig. 2.1 shows the architecture of the system. The system consists of p $(p \geq 1)$ PTZ cameras and a wide-angle camera. All cameras are calibrated. The wide-angle camera detects and labels all moving objects in the scene. The states of the objects (e.g., size, position and velocity) are tracked and predicted. Based on the prediction, the observation request generation module generates the competing spatiotemporal observation requests (shadowed rectangles) for all objects. Then the request assignment module groups requests and assigns a subset of the objects/requests to each PTZ camera by computing the p-frame settings that best satisfy the requests. Each PTZ camera tracks the objects assigned to it by selecting the PTZ parameter settings that best satisfy these requests to capture high resolution images/videos of the objects.

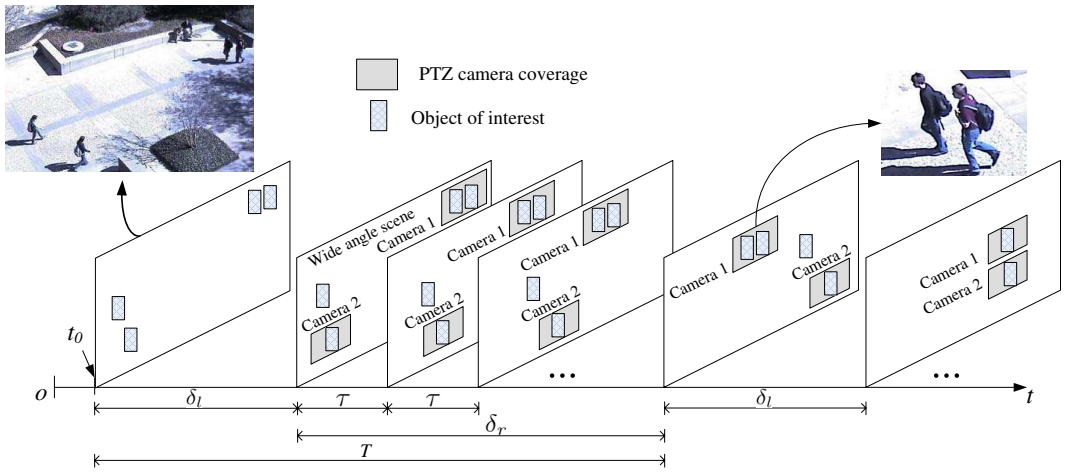


Fig. 2.2. System timeline. An observation cycle starts at $t = t_0$. Within each cycle time $T = \delta_l + \delta_r$, all PTZ cameras first take no more than δ_l time to adjust the PTZ parameters so that each PTZ camera is assigned with a subset of the objects. Then each PTZ camera micro-adjusts its parameters within interval τ to track the assigned subset of objects. This tracking lasts δ_r time until a new observation cycle starts.

Fig. 2.2 shows the timeline of the system. An observation cycle starts at time $t = t_0$. The states of the objects at time $t = t_0 + \delta_l$ are predicted, where δ_l is termed as "lead time." Based on the predicted states, the system generates the observation request at time $t = t_0 + \delta_l$ for each object. A subset of these objects is then assigned to each PTZ camera. Then the system starts to adjust the PTZ cameras according to the request assignment. The camera traveling time is bounded below the "lead time" δ_l so that the cameras can intercept the objects at time $t = t_0 + \delta_l$. After that, each PTZ camera tracks its object subset for time δ_r until the beginning of the next observation cycle. δ_r is termed as "recording time" and is evenly divided into n_r intervals with each of length τ. Based on the state prediction, the PTZ camera parameter selection module computes each camera's setting at the end of each interval. Then each camera micro-adjusts its settings for up to τ time and prepares for the next interval. By capturing images/videos for δ_r time, the request assignment module re-initiates and the operations above repeat. $T = \delta_l + \delta_r$ is called one observation cycle.

The stationary camera we use is a Arecont Vision AV3100 with a Computar lens whose focal length ranges from 4.5mm to 12.5mm. The camera runs at 11 frames per second (fps) with high resolution of 1600×1200. The PTZ cameras we use are Panasonic HMC280. The camera uses the MPEG4 compression and runs at up to 30 fps with resolution of 640×480. It has a 350° pan range and a 120° tilt range. It can pan and tilt up to 300° per second and 200° per second, respectively. It has $21\times$ motorized zoom with zoom-changing speed up to 5 levels per second. The system is programmed using Microsoft Visual C++.

2.3 Camera Scheduling

For p PTZ cameras, there are usually much more objects/requests. With the competing spatiotemporal requests, we need to control and schedule the PTZ cameras to capture sequences of images/videos that best satisfy the requests.

2.3.1 Observation Request Generation

The wide-angle camera detects moving objects and tracks them continuously. Each object is represented by its minimal iso-oriented bounding rectangular region which is determined by a 4-parameter vector,

$$[u, v, a, b]^T, \tag{2.1}$$

where (u, v) indicates the center of the rectangle in the image space; a and b denote the width and height of the rectangle, respectively. Thus the state of the object at time t can be represented by

$$x(t) = [u(t), v(t), a(t), b(t), \dot{u}(t), \dot{v}(t)]^T, \tag{2.2}$$

where $(\dot{u}(t), \dot{v}(t))$ indicates the velocity of the rectangle center in the image space at time t.

A non-parametric Gaussian background subtraction model [18] is used to detect and label any moving objects. For tracking and predicting the object state, each labeled object is assigned with a Kalman filter. A commonly used constant velocity model is adopted. The Kalman filter is also able to handle short-term occlusion by predicting the object motion. It is worth mentioning that a lot of other existing tracking algorithms [28] can be applied here and the tracking itself is not the focus of our work.

Given the predicted state of i-th object at time t is

$$\hat{x}_i(t) = [\hat{u}_i(t), \hat{v}_i(t), \hat{a}_i(t), \hat{b}_i(t), \hat{\dot{u}}_i(t), \hat{\dot{v}}_i(t)]^T,$$

we define the spatiotemporal observation request as,

$$r_i(t) = [T_i(t), z_i, \omega_i(t)]^T, \tag{2.3}$$

where $T_i(t) = [\hat{u}_i(t), \hat{v}_i(t), \hat{a}_i(t), \hat{b}_i(t)]$ represents the rectangular request region determined by $\hat{u}(t)$, $\hat{v}(t)$, $\hat{a}_i(t)$ and $\hat{b}_i(t)$ in the same way as u, v, a and b in (2.1); z_i indicates the desirable resolution, which is in the range of $Z = [\underline{z}, \overline{z}]$. We set z_i as the resolution of the minimal camera frame that contains $T_i(t)$. $\omega_i(t)$ is the temporal weight, which indicates the emergency/importance level of the i-th object at time t. $\omega_i(t)$ plays an important role in balancing the observation service across all the objects and will be discussed in details later in Section 2.3.4. Given there are n objects, we generate a set of n requests,

$$R(t) = \{r_i(t) | i = 1, 2, ..., n\}.$$

2.3.2 Request Assignment

As shown in Fig. 2.2, at the beginning of each recording time δ_r, we need to coordinate p PTZ cameras so that each camera is assigned a subset of the objects. We choose the p-frame settings that best satisfy all the requests at that time. We formulate the request

assignment issue as an optimization problem, which maximizes the overall "satisfaction" of the requests and we term this problem as a p-frame problem. The input of the p-frame problem is the request set $R(t) = \{r_i(t)|i = 1, 2, ..., n\}$. A solution to the p-frame problem is a set of p PTZ camera frames. Given a fixed aspect ratio (e.g. 4:3), a camera frame can be defined as $c = [x, y, z]^T$, where the pair (x, y) denotes the center point of the rectangular frame and $z \in Z$ specifies the resolution level of the camera frame. Here we consider the coverage of the camera as a rectangle according to the camera configuration space. Therefore, the width and height of the camera frame can be represented as $4z$ and $3z$ respectively. We define any candidate solution to the p-frame problem as $C^p = (c_1, c_2, ..., c_p) \in \mathcal{C}^p$, where $c_i, i = 1, 2, ..., p$, indicates the i-th camera frame in the solution. The objective of the p-frame problem is to find the optimal solution $C^{p*} = (c_1^*, c_2^*, ..., c_p^*)$ that best satisfies the requests:

$$C^{p*} = \arg\max_{C^p} s(C^p), \tag{2.4}$$

where $s(\cdot)$ is the satisfaction metric which will be introduced in details in Section 3.

2.3.3 PTZ Camera Parameter Selection

After each camera is assigned a subset of objects by solving the p-frame problem, the camera tries to track these objects for the recording time δ_r. This requires to select the camera parameter setting such that the satisfaction is maximized for each recording interval. Given each recording interval is represented as $[t - \tau, t)$ and the i-th camera is assigned a subset of objects with predicted states at time t, $\hat{X}_i(t) = \{\hat{x}_1(t), \hat{x}_2(t), ...\}$. The corresponding observation requests are generated $R_i(t) = \{r_1(t), r_2(t), ...\}$. The camera setting at time t, $c^*(t)$, is then determined by maximizing the satisfaction to $R_i(t)$,

$$c^*(t) = \arg\max_{c} \sum_{r_i(t) \in R_i(t)} s(c, r_i(t)). \tag{2.5}$$

This problem can be solved using the approximation algorithm in [48] with running time $O(|\hat{X}_i|/\epsilon^3)$, where $|\hat{X}_i|$ is the cardinality of $\hat{X}_i$ and ϵ is the approximation bound. However, (2.5) does not consider the fact that within time τ, the PTZ camera can only micro-adjust within a limited setting range. We assume the pan, tilt and zoom motion of the camera are independent. The reachable ranges for pan, tilt and zoom settings within time τ are α, β and γ, respectively. Then we rewrite (2.5) as,

$$c^*(t) = \arg \max_{c \in \alpha \times \beta \times \gamma} \sum_{r_i(t) \in R_i(t)} s(c, r_i(t)). \tag{2.6}$$

It is worth mentioning that most PTZ cameras' pan and tilt motion is fast enough to follow most objects in the scene. For example, recall the transition speed of the Panasonic HCM 280 camera is 300° per second for pan, 200° per second for tilt and 5 levels per second for zoom, respectively. Considering the camera has $21\times$ zoom levels and only less than 50° FOV, the time for changing pan and tilt settings is much less than the time for changing the camera zoom. Changing the zoom level when the camera is moving also creates significant motion blurring and often requires more than 1-2 seconds for re-focusing. Therefore, in practice, we usually search for the pan and tilt settings in $\alpha \times \beta$ while maintain the same zoom level for each recording period.

2.3.4 Dynamic Weighting

If we keep the request weight in (2.3) unchanged, the system will create a "biased frame selection" model that always prefers certain objects instead of balancing the camera resource for all objects. We address this issue by carefully designing the temporal weight $\omega_i(t)$ based on two intuitions: 1) object exiting FOV sooner is of more importance and 2) object less satisfied in history is of more importance. The first intuition is derived from the earliest deadline first (EDF) policy [36]. The policy addresses the emergency of the

requests. The second intuition addresses sharing the camera resource for all objects to achieve balanced observation over time. We define,

$$\omega_i(t) = \mu_i(t) \cdot \nu_i(t),$$

where $\mu_i(t)$ and $\nu_i(t)$ address the first and second intuitions, respectively. One candidate form of $\mu_i(t)$ is,

$$\mu_i(t) = \rho^{(\hat{d}_i - t)}, \tag{2.7}$$

where $\hat{d}_i$ is the predicted deadline for i-th object to exit the FOV and $0 < \rho < 1$ is a parameter that controls how quick the emergency increases. Because we only observe objects in the FOV, $t \leq \hat{d}_i$. When $t \to \hat{d}_i$, $\mu_i(t) \to 1$, as maximum.

To design $\nu_i(t)$, we need to first define the accumulative unweighted satisfaction (AUS) $\eta_i(t)$,

$$\eta_i(t) = \sum_{j=1}^{p} \sum_{t_k < t} \frac{s(c_j(t_k), r_i(t_k))}{\omega_i(t_k)}, \tag{2.8}$$

where the variable t_k refers to the discrete times when cameras take frames. The AUS essentially reflects how well an object is satisfied in history. We design $\nu_i(t)$ as,

$$\nu_i(t) = \max(1 - \frac{\eta_i(t)}{n_e}, 0), \tag{2.9}$$

where n_e is a parameter indicating the extent to which an object need to be observed. When $\eta_i(t) \geq n_e$, $\nu_i(t)$ is zero and we contend the object is fully satisfied and needs no observation any longer. Both $\mu_i(t)$ and $\nu_i(t)$ are bounded in range $[0, 1]$, which keeps the satisfaction metric in (4.2) a standardized metric.

2.4 Experiment

We have implemented the system using Microsoft Visual C++ 2005. The computer used is a Windows XP desktop PC with 2.0 GB RAM, 300 GB hard disk space and an

Intel Pentium(R) Dual Core 3.2 GHz CPU. We carry out a simulation to compare the camera scheduling of our system with an existing work based on the overall number of objects being observed. Finally, a physical experiment for crowd surveillance using real video data is reported.

2.4.1 Evaluating System by Simulation

We carry out a simulation for evaluating the scheduling method of the system based on random inputs. The results are compared with an existing scheduling algorithm.

Simulation setup

As shown in Fig. 2.3, a simulated 80×60 m^2 scene is constructed. Each object enters the scene through one side and maintains a constant velocity. Seven random numbers are needed to characterize each object. First, a random integer number ranging from 1 to 4 is generated to indicate which side the object enters through. Then a random real number in $[0, 1]$ is generated to indicate the entering point along the side. After that, the orientation of the object is determined by a random angle within the range $[-40°, 40°]$ with respect to the perpendicular of the side. The object speed is generated from a truncated Gaussian with a mean of 1.5 m/s and a standard deviation of 0.5 m/s, which is basically the speed of a walking people. The width and height of the rectangle that represents the object are randomly generated from a range $[1.5, 2.5]$ m. Finally, the desirable resolution of the object is generated from a range $[1, 21]$ (level), which is also the Panasonic HCM280 camera zoom range. The cameras run in 10 fps, which means $\tau = 0.1\ s$. Then $\alpha = 30°$ and $\beta = 20°$. 5000 objects arrive in the scene following a Poisson process with arrival rate λ, which represents the congestion level of the scene. We set the lead time $\delta_l = 4$ seconds, which guarantees that in the request assignment phase, camera adjustment is completed before cameras intercept the objects. We set $\delta_r = 6$ seconds, which is equivalent to

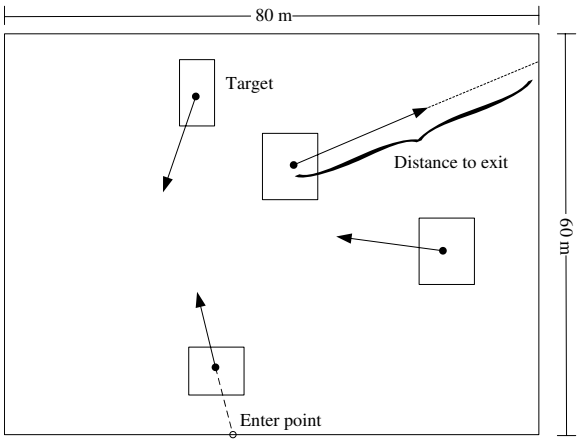


Fig. 2.3. An illustration of the simulated scene. Each object is represented as a rectangle and enters the scene from one of the four sides following a Poisson process. The orientation is within $[-40^\circ, 40^\circ]$ with respect to the norm of the side. The object maintains constant velocity and its time to exit is predicted.

$n_r = 60$ frames. We set the parameter $n_e = n_r$ in (2.9) and $\rho = 0.5$ in (2.7) and $\epsilon = 0.25$ in the p-frame approximation algorithm. Two PTZ cameras are used, i.e., $p = 2$.

Metric and results

We compare our scheduling scheme with the earliest deadline first (EDF) policy proposed in [36]. EDF is a heuristic scheme where the camera always picks the object with earliest deadline. With each congestion setting, 20 trials are carried out for average performance. We first compare the two schemes based on the ratio of number of objects that are observed for at least $n_r/2$ times to the total number of objects pass through the scene. We term this metric as M_n. This metric essentially indicates how many objects the system can capture and observe for a period of time. Fig. 2.4 shows the comparison result. It is shown that when the Poisson arrival rate λ is small, i.e., there are few objects in the scene, both scheduling schemes can reach almost best possible ratio (100%). When λ increases, i.e., the traffic in the scene becomes heavy, the performance of EDF deteriorates significantly

quicker than our method. In the heavy traffic scenario, our method outperforms the EDF by over 210%.

We also compare our method with EDF based on the satisfaction to the objects since it takes into account not only the times that an object is observed, but also the resolution of the observation. As mentioned earlier, the AUS as defined in (2.8) indicates how well an object is satisfied. We define the second metric M_s as the ratio of average AUS to the maximum possible satisfaction for each object (i.e., n_e). Fig. 2.5 summarizes the comparison based on M_s. It is shown that our method outperforms EDF as λ increases. In the heavy traffic scenario, our method outperforms the EDF by 250 %. This is not surprising since in heavy traffic situations, objects tends to be close to each other, where multi-object coverage has great advantage.

Close-up analysis reveals that our satisfaction formulation in (4.2) is actually a generalization of many existing scheduling schemes. For example, if we tune parameter ρ in (2.7) to approach to zero, then the change in $\mu_i(t)$ dominates the change in the overall weight. That means we extremely care the emergency of the request and thus the scheduling converges to the earliest deadline first (EDF) policy [36]. Also, if we set extremely high requested resolution (i.e., extremely small z_i), it implies that we extremely care the resolution of the image frame. As a result, the algorithm would tend to produce smaller frames (higher resolution) to cover fewer requests at a price of (possibly) losing coverage of other requests. In the extreme case, to obtain the best resolution, it would only assign one request to one PTZ camera, which is exactly the scheduling scheme as in almost all existing works.

2.4.2 Physical Experiment

We carry out a physical experiment to validate our system using real video data. Our camera is mounted on the 6th floor of the Evans Library of Texas A&M University to monitor the crowd entering and leaving the library. In the experiment, we set $t_0 = \delta_l = 1.5$ seconds, $\delta_r = 2$ seconds and $p = 3$. The camera runs at 10 fps. In the submission, we

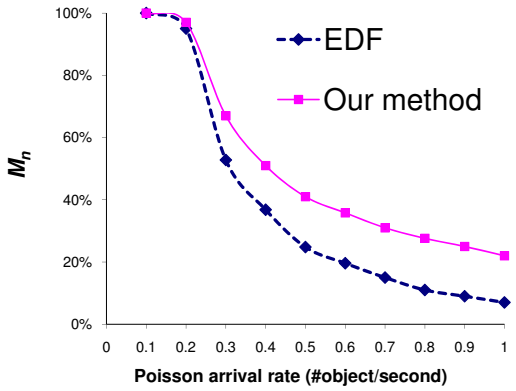


Fig. 2.4. Comparison of scheduling policies based on M_n.

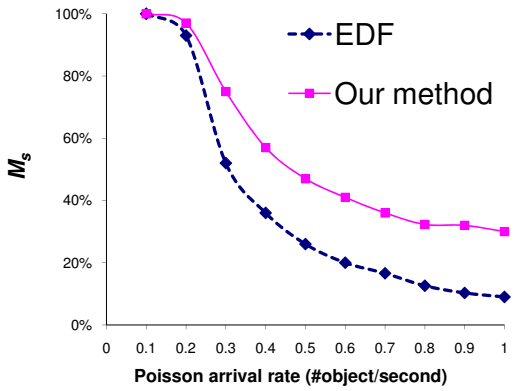
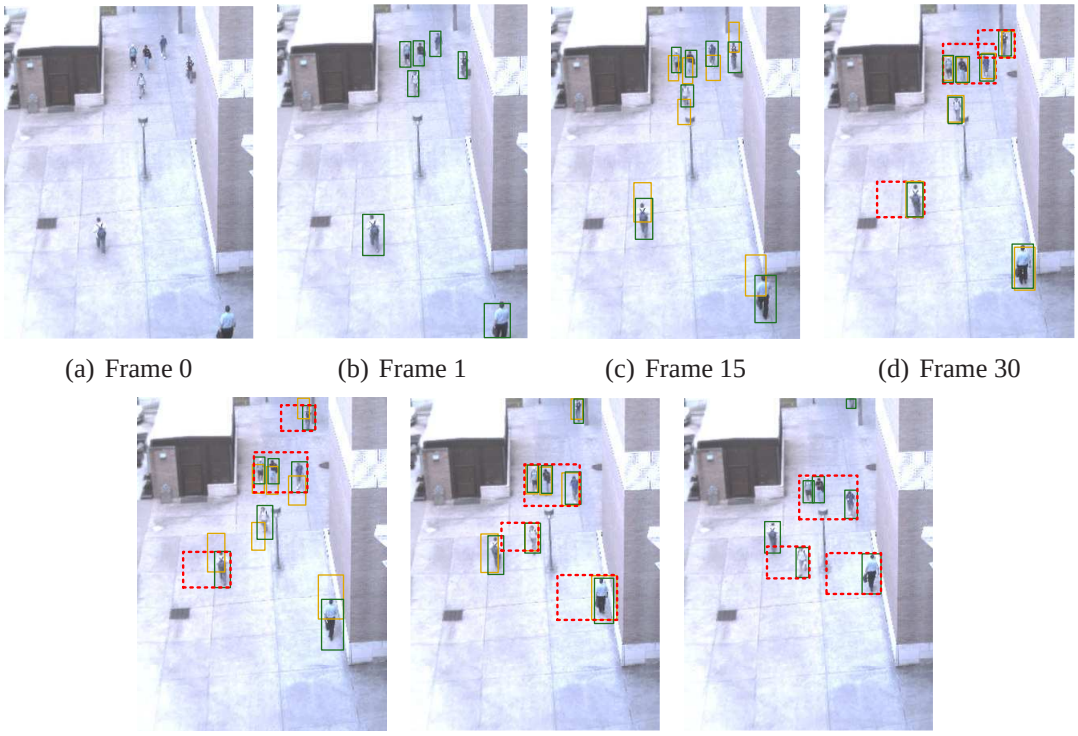


Fig. 2.5. Comparison of scheduling policies based on M_s.

attach a video clip that records a representative observation operation which contains two consecutive observation cycles at 17:25 on May 4th, 2009. The corresponding key frames are presented in Fig. 2.6. It is shown that the request assignment module is capable of partitioning the objects and assigning each PTZ camera with a subset of the objects. The PTZ camera parameter selection module ensures the assigned objects are covered for the duration of the observation cycle. Between the observation cycles, the system also shows



(a) Frame 0 (b) Frame 1 (c) Frame 15 (d) Frame 30

(e) Frame 50 (f) Frame 65 (g) Frame 85

Fig. 2.6. Key frames in a representative surveillance cycle. (a) At time $t = 0$, there are 7 people. (b) The system starts to track the people, who are represented by green rectangles. (c) At time $t = t_0$, the states of the people at time $t = t_0 + \delta_l$ are predicted, which are represented by yellow rectangles. (d) At time $t = t_0 + \delta_l$, each PTZ camera is assigned a subset of the people. The optimal PTZ camera settings are represented by red dashed rectangles. (e) At time $t = t_0 + T$, one observation cycle finishes and the system predicts the states of the people at time $t = t_0 + T + t_l$ for the next cycle. (f) At time $t = t_0 + T + t_l$, each PTZ camera is again assigned a subset of the people. The better satisfied objects in the previous cycle are deprioritized through the dynamic weighting. (g) At time $t = t_0 + 2T$, the second observation cycle finishes.

the ability to adjust the priority of the objects through the dynamic weighting so that every moving object is evenly observed.

2.5 Conclusions

We presented an autonomous vision system that consists of multiple robotic PTZ cameras and a fixed wide-angle camera for observing multiple objets simultaneously. We presented the system with observation request generation, request assignment and PTZ camera parameter selection modules. We formulated the PTZ camera scheduling as a sequence of request assignment and camera parameter selection problems with objective of maximizing the satisfaction to requests. We validated the system by both simulation and physical experiments. The comparison with an existing work based on simulation has shown that our system significantly enhances the observation performance especially in heavy traffic situations.

In the future, we will investigate how different frame selection formulation would impact the system performance and how they fit human user need in practice. Another interesting extension is to consider the camera traveling time within the request assignment. Intuitively, asynchronized observation by multiple PTZ cameras would further enhance the system performance. The camera content delivery through internet would be another interesting topic especially when number of camera increases.

In the next two sections, we will introduce an approximation algorithm and an exact algorithm, respectively, for the p-frame problem.

3. MOMR++ ALGORITHM: APPROXIMATION ALGORITHM FOR CROWD SURVEILLANCE SYSTEM

In Section 2, we introduce an autonomous crowd surveillance system. In the system design, it assigns n observation requests to p cameras by solving the p-frame problem. In this section, we focus on formal formulation of the p-frame problem and propose an approximation algorithm for solving the problem. Fig. 3.1 illustrates the p-frame problem: how to identify optimal p frames that best satisfy the n different polygonal requests.

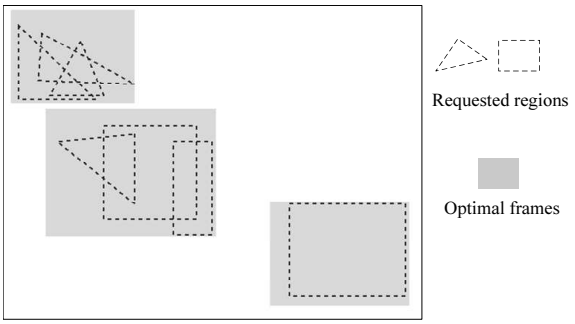


Fig. 3.1. An illustration of the least overlapping 3-frame problem.

We assume that the p frames have the least overlap (will be formally defined later) on the coverage between the frames and a request is satisfied only if it is fully covered by one of the p frames. Under the assumptions, we propose a Resolution Ratio with Non-Partial Coverage (RRNPC) metric to quantify the satisfaction level for a given request with respect to a set of p candidate frames. Hence the p-frame problem is to find the optimal set of (up to p) frames that maximizes the overall satisfaction. Building on the results in [48], we propose a lattice-based approximation algorithm. The algorithm builds on an induction-like approach that finds the relationship between the solution to the $p-1$ frame problem and the solution to the p-frame problem. For a given approximation bound ϵ, the algorithm runs in $O(n/\epsilon^3 + p^2/\epsilon^6)$ time. We have implemented the algorithm and

experiment results are consistent with our complexity analysis. We will begin with the related work.

3.1 Related Work

The p-frame problem relates to networked robotics, the facility location problem in operations research, and the single frame selection problem.

The development of the Internet allows more users to access online resources. The p frames taken by p networked pan-tilt-zoom cameras can be viewed as a special case of networked tele-operation, where each robotic camera has 3 Degrees of Freedom (DOF). According to the taxonomy proposed by Chong *et al.* [5], this system belongs to Multiple Operator Multiple Robot (MOMR) systems. The low cost robot and sensor network makes the MOMR system a very popular research domain [10, 11, 50]. In [12, 51], Liu and his colleagues developed a competitive MOMR system under a game setting. Our work emphasizes on the geometric coverage attributes of the robotic camera and addresses the MOMR problem in an optimization framework.

The p-frame problem is structurally similar to the p-center facility location problem, which has been proven to be NP-complete [52]. Given n request points on a plane, the task is to optimally allocate p points as service centers to minimize the maximum distance (called min-max version) between any request point and its corresponding service center. In [53], an $O(n \log^2 n)$ algorithm for a 2-center problem is proposed. As an extension, replacing service points by orthogonal boxes, Arkin *et al.* [54] propose a $(1+\epsilon)$-approximation algorithm that runs in $O(n \min(\lg n, 1/\epsilon) + (\lg n)/\epsilon^2)$ for the 2-box covering problem. Alt *et al.* [55] proposed a $(1+\epsilon)$-approximation algorithm that runs in $O(n^{O(m)})$, where $\epsilon = O(1/m)$, for the multiple disk covering problem. The requests in these problems are all points instead of polygonal regions as those in our p-frame problem and the objective of the p-frame problem is to maximize the satisfaction, which is not a distance metric.

The p-frame problem also relates to the art gallery problem [56]. The art gallery problem is to minimize the number of security guards to guard an art gallery, which is usually represented by a polygon with n vertices. Each guard has a certain range of vision. The location of the guard can be represented by a point while the reachable region of the guard can be represented by any geometrical shapes. Agarwal *et al.* [57] consider a variation of the art gallery problem where the terrain is not planar and there are only two guards with minimal heights. They propose an exact algorithm that runs in $O(n^2 \log^4 n)$ time. In [58], Eppstein *et al.* propose the sculpture garden problem where each guard has only a limited angle of visibility. They prove that the upper bound is $n - 2$ and the lower bound is $n/2$ for the number of the guards needed. More results on the art gallery problem can be found in [59]. Unlike the art gallery problem, the p-frame problem does not need to cover all requests. However, the selection has to be made based on maximizing the level of satisfaction of covered requests.

Our group has worked on camera frame selection problems since 2002. We have addressed the Single Frame Selection (SFS) problem and its variations such as approximate solution with continuous zoom [60], approximate solution with fixed zoom [61], and exact solution with continuous zoom and rectangular requests with fixed aspect ratio [62] or variable aspect ratio [47]. Extending the results for SFS to the p-frame problem is non-trivial. Our work in this section is the first attempt to tackle the problem.

3.2 Problem Definition

In this section, we formulate the p-frame problem. We begin with the definition of the inputs and outputs. Assumptions are then presented. We establish the request satisfaction metric so that we can formulate the problem as a geometric optimization problem.

3.2.1 Input and Output

The input of the problem is a set of n requests $R = \{r_i | i = 1, 2, ..., n\}$. Each request is defined as $r_i = [T_i, z_i]$, where T_i denotes the polygonal requested region and $z_i \in Z$ specifies the desired resolution level, which is in the range of $Z = [\underline{z}, \overline{z}]$. The only requirement for T_i is that its coverage area can be computed in constant time.

A solution to the p-frame problem is a set of p camera frames. Given a fixed aspect ratio (e.g. 4:3), a camera frame can be defined as $c = [x, y, z]$, where pair (x, y) denotes the center point of the rectangular frame and $z \in Z$ specifies the resolution level of the camera frame. Here we consider the coverage of the camera as rectangular according to the camera configuration space. Therefore, the width and height of the camera frame can be represented as $4z$ and $3z$ respectively. The coverage area of the frame is $12z^2$. The four corners of the frame are located at $(x \pm 4z/2, y \pm 3z/2)$.

Given w and h are the camera pan-tilt ranges respectively, then $\mathcal{C} = [0, w] \times [0, h] \times Z$ defines the set of all candidate frames. Therefore, $\mathcal{C}^p$ indicates the solution space for the p-frame problem. We define any candidate solution to the p-frame problem as $C^p = (c_1, c_2, ..., c_p) \in \mathcal{C}^p$, where $c_i, i = 1, 2, ..., p$, indicates the i-th camera frame in the solution. In the rest of the section, we use superscription * to indicate the optimal solution. The objective of the p-frame problem is to find the optimal solution $C^{p*} = (c_1^*, c_2^*, ..., c_p^*) \in \mathcal{C}^p$ that best satisfies the requests.

3.2.2 Set Operators

We clarify the use of set operators such as "$\cap$", "$\subseteq$ " and "$\nsubseteq$" to represent the relationship between frames, frame sets, and requests in the rest of the section.

- When two operands are frames or requests (e.g., $r_i \in R, c_u, c_v \in \mathcal{C}$), the set operators represent the 2-D regional relationship between them. For example, $r_i \subseteq c_u$ represents that the region of r_i is fully contained in that of frame c_u while $c_u \cap c_v$ represents the overlapping region of frames c_u and c_v .

- When the operands are one frame (e.g., $c_i \in \mathcal{C}$) and one frame set (e.g., $C^k \in \mathcal{C}^k, k < p$), we treat the frame as an element of a frame set. For example, $c_i \notin C^k$ represents that c_i is not an element frame in the frame set C^k.

- When the operands are two frame sets, we use set operators. For example, $\{c_1\} \subset C^p$ means frame set $\{c_1\}$ is a subset of C^p. Frame set $\{c_1, c_2\} = \{c_1\} \cup \{c_2\}$ is different from $c_1 \cup c_2$. The former is the frame set that consists of two element frames and the later is the union area of the two frames.

3.2.3 Assumptions

We assume that the p-frames are either taken from p cameras that share the same workspace or taken from the same camera. Therefore, if a location can be covered by a frame, other frames can cover that location, too.

We assume that the solution C^{p*} to the p-frame problem satisfies the following condition.

Definition 3.2.1 (Least Overlapping Condition (LOC)) $\forall r_i,\ i = 1, ...n,\ \forall c_u \in C^{p*}$, $\forall c_v \in C^{p*}$, *and* $c_u \neq c_v$,

$$r_i \cap (c_u \cap c_v) = \phi. \tag{3.1}$$

The LOC means that the overlap between frames is so small that no request can be fully covered by more than one frame simultaneously. The LOC forces the overall coverage of a p-frame set $\cup_{j=1}^{p} c_j$ to be close to the maximum. This is meaningful in applications when the cameras need to search for unexpected events while best satisfying the n existing requests because the ability to search is usually proportional to the union of overall coverage. Therefore, the LOC can increase the capability of searching for unexpected events. The extreme case of the LOC is that there is no overlap between camera frames.

Definition 3.2.2 (Non-Overlapping Condition (NOC)) *Given a p-frame set $C^p = (c_1, c_2, ..., c_p) \in \mathcal{C}^p$ ($p \geq 2$), C^p satisfies the NOC, if*

$$\forall u = 1, 2, ..., p, \forall v = 1, 2, ..., p, u \neq v, c_u \cap c_v = \phi \ .$$

It is not difficult to find that the NOC is a sufficient condition to the LOC. The NOC yields the maximum union coverage and is a favorable solution to applications where searching ability is important.

3.2.4 Satisfaction Metric

To measure how well a p-frame set satisfies the requests, we need to define a satisfaction metric. We extend the Coverage-Resolution Ratio (CRR) metric in [47] and propose a new Resolution Ratio with Non-Partial Coverage (RRNPC).

Definition 3.2.3 (RRNPC metric) *Given a request $r_i = [T_i, z_i]$ and a camera frame $c = [x, y, z]$, the satisfaction of request r_i with respect to c is computed as*

$$s(c, r_i) = I(c, r_i) \cdot \min(\frac{z_i}{z}, 1), \tag{3.2}$$

where $I(c, r_i)$ is an indicator function that describes the non-partial coverage condition,

$$I(c, r_i) = \begin{cases} 1 & \textit{if} \ \ r_i \subseteq c, \\ 0 & \textit{otherwise}. \end{cases} \tag{3.3}$$

Eq. (4.3) indicates that we do not accept partial coverage over the request. Only the requests completely contained in a camera frame contribute to the overall satisfaction. From (4.2) and (4.3), the satisfaction of the ith request is a scalar $s_i \in [0, 1]$.

Based on (4.2), the satisfaction of r_i with respect to a candidate least overlapping p-frame set $C^p = (c_1, c_2, ..., c_p) \in \mathcal{C}^p$ is,

$$s(C^p, r_i) = \sum_{u=1}^{p} I(c_u, r_i) \cdot \min(\frac{z_i}{z_u}, 1), \tag{3.4}$$

where z_i, z_u indicate the resolution values of r_i and the u-th camera frame in C^p respectively. The LOC implies that although (3.4) is in the form of summation, at most one frame contains the region of request r_i and thus non-negative $s(C^p, r_i)$ has a maximum value of 1. Therefore, RRNPC is a standardized metric that takes both the region coverage and the resolution level into account.

To simplify the notation, we use $s(c) = \sum_{i=1}^{n} s(c, r_i)$ to represent the overall satisfaction of a single frame c. We also use $s(C^k) = \sum_{j=1, c_j \in C^k}^{k} s(c_j)$, to represent the overall satisfaction of a partial candidate k-frame set C^k, $k < p$.

3.2.5 Problem Formulation

Based on the assumption and the RRNPC metric definition above, the overall satisfaction of a p-frame set $C^p = \{c_1, c_2, ..., c_p\} \in \mathcal{C}^p$ over n requests is the sum of the satisfaction of each individual request $r_i, i = 1, 2, ..., n$,

$$s(C^p) = \sum_{i=1}^{n} \sum_{u=1}^{p} I(c_u, r_i) \cdot \min(\frac{z_i}{z_u}, 1). \tag{3.5}$$

Eq. (3.5) shows that the satisfaction of any candidate C^p can be computed in $O(pn)$ time. Now we can formulate the least overlapping p-frame problem as a maximization problem,

$$C^{p*} = \arg \max_{C^p \in \mathcal{C}^p} s(C^p). \tag{3.6}$$

3.3 Algorithm

Solving the optimization problem in (3.6) is nontrivial. To enumerate all possible combinations of candidate solutions by brute force can easily take up to $O(n^p)$ time. In this section, we present a lattice-based approximation algorithm beginning with the construction of the lattice. To maintain the LOC in the lattice framework, we introduce the Virtual Non-Overlapping Condition(VNOC). Based on the VNOC, we analyze the structure of the approximate solution and derive the approximation bound with respect to the optimal solution that satisfies the NOC . To summarize this, a lattice-based induction-like algorithm is presented at the end of the section.

3.3.1 Construction of Lattice

We construct a regular 3-D lattice, which is inherited from [48] to discretize the solution space $\mathcal{C}^p$. Let 2-D point set $V = \{(\alpha d, \beta d) | \alpha d \in [0, w], \beta d \in [0, h], \alpha, \beta \in \mathcal{N}\}$ discretize the 2-D reachable region and represent all candidate center points of rectangular frames, where d is the spacing of the pan and tilt samples. Let 1-D point set $\mathcal{Z} = \{\gamma d_z | \gamma d_z \in [\underline{z}, \overline{z} + 2d_z], \gamma \in \mathcal{N}\}$ discretize the feasible resolution range and represent all candidate resolution values for the camera, where d_z is the spacing of the zoom. Therefore, we can construct the lattice as a set of 3-D points, $L = V \times \mathcal{Z}$.

Each point $c = (\alpha d, \beta d, \gamma d_z) \in L$ represents the setting of a candidate camera frame. There are totally $(wh/d^2)(g/d_z) = |L|$ candidate points/frames in L, where $g = \overline{z} - \underline{z}$. We set $d_z = d/3$ for cameras with an aspect ration of $4 : 3$ according to [48]. It is worth noting that any candidate frame with center point close to the boundary of the 2-D reachable region and a large zoom level may have its coverage out of the reachable region and thus becomes an infeasible candidate frame. Therefore, the actual feasible solution space is in a pyramid-like shape.

What is new is that the spacing of the lattice d and d_z also depends on the size of the requested regions. For any request $r_i \in R$, there exists an Iso-oriented Bounding Box

(IBB) for each r_i. Let us define λ and μ as the smallest width and height across all IBBs, respectively. We choose d such that

$$d < \min(3\lambda/10, \mu/3). \tag{3.7}$$

This input-sensitive lattice setting can help us to establish the LOC on the lattice and will be discussed later in Section 3.3.2. From here on, we use symbol ˜ to denote the lattice-based notations. For example, $\tilde{C}^p$ denotes a p-frame set on lattice L.

Definition 3.3.1 *For any camera frame $c \in \mathcal{C}$,*

$$\tilde{c}' = \min \tilde{c},\ \textit{s.t.}\ \tilde{c} \in L\ \textit{and}\ c \subseteq \tilde{c}.$$

Hence $\tilde{c}'$ is the smallest frame on the lattice that fully encloses c.

In the rest of the section, we use symbol $'$ to denote the corresponding smallest frame(s) on the lattice. For any camera frame $c = [x, y, z]$ and its corresponding $\tilde{c}' = [\tilde{x}', \tilde{y}', \tilde{z}']$, we define their bottom-left corners as (x^l, y^b) and $(\tilde{x}'^l, \tilde{y}'^b)$, and their top-right corners as (x^r, y^t) and $(\tilde{x}'^r, \tilde{y}'^t)$, respectively.

From the results of [48], we have

$$\begin{aligned} x^l - \tilde{x}'^l \le 5d/3,\ \ \tilde{x}'^r - x^r \le 5d/3, \\ y^b - \tilde{y}'^b \le 3d/2,\ \ \tilde{y}'^t - y^t \le 3d/2. \end{aligned} \tag{3.8}$$

3.3.2 Virtual Non-Overlapping Condition

The NOC defined in Definition 4.3.1 guarantees the LOC. However, due to the limitation of lattice spacing, it is very difficult for candidate frames on the lattice to follow the NOC. Actually, it is unnecessary (though sufficient) to follow the NOC to satisfy the LOC. It is possible to allow a minimum overlap that is controlled by the lattice spacing and mean-

while guarantee that the LOC is still satisfied, which yields the Virtual Non-Overlapping Condition (VNOC).

Definition 3.3.2 (Virtual Non-Overlapping Condition(VNOC)) *Given any j-frame set $C^j = (c_1, c_2, ..., c_j) \in \mathcal{C}^j, j = 2, 3, ..., p$ and any two frames $c_u, c_v \in C^j$, then C^j satisfies the VNOC, if* $\min(x_u^r - x_v^l, x_v^r - x_u^l) \leq 10d/3$ *or* $\min(y_u^t - y_v^b, y_v^t - y_u^b) \leq 3d$.

Corollary 1 *Given any two frames $c_1, c_2 \in \mathcal{C}$, if $\{c_1, c_2\}$ satisfies the VNOC, then $\{c_1, c_2\}$ also satisfies the LOC.*

Proof From the definition of VNOC and the settings of λ and μ, we see that the size of the overlapping region $c_1 \cap c_2$, on either the x-axis or y-axis, is less than the size of the smallest request. This guarantees that no requested region is fully contained in the overlapping region. Therefore, the LOC is satisfied. ■

Lemma 1 *Given any two frames $c_1, c_2 \in \mathcal{C}$ such that $\{c_1, c_2\}$ satisfies the VNOC, then*

$$s(\{c_1, c_2\}) = s(c_1) + s(c_2). \tag{3.9}$$

Proof From Corollary 1, $\{c_1, c_2\}$ satisfies the LOC. From the definition of the LOC and the RRNPC satisfaction metric defined in (4.2), the conclusion follows. ■

3.3.3 Approximation Solution Bound

The construction of the lattice allows us to search for the best p frames on the lattice, which yields an approximation solution. Furthermore, the VNOC and Lemma 1 assist us in deriving the approximation bound.

Lemma 2 *For any two frames $c_1, c_2 \in \mathcal{C}$, if $\{c_1, c_2\}$ satisfies the NOC, then $\{\tilde{c}_1', \tilde{c}_2'\}$ satisfies the VNOC.*

The proof of the lemma is trivial based on the definition of VNOC and the settings of λ and μ.

Given the optimal solution $C^{p*} = (c_1^*, c_2^*, ..., c_p^*)$ for the optimization problem defined in (3.6) that satisfies the NOC, there is a solution on the lattice $\tilde{C}'^{p*} = (\tilde{c}_1'^*, \tilde{c}_2'^*, ..., \tilde{c}_p'^*)$ whose element frames are the corresponding smallest frames on the lattice that contain those of C^{p*}. Lemma 2 implies that $\tilde{C}'^{p*}$ exists and satisfies the VNOC. However, how good is this solution in comparison to the optimal solution? We define the approximation bound ϵ which characterizes the comparative ratio of the approximation solution to the optimal solution

$$s(\tilde{C}'^{p*})/s(C^{p*}) \geq 1 - \epsilon. \tag{3.10}$$

Based on Lemma 1 and Theorem 1 in [48], we have

$$s(\tilde{C}'^{p*})/s(C^{p*}) \geq 1 - \frac{2d_z}{\underline{z} + 2d_z}. \tag{3.11}$$

Let $\tilde{C}^{p*}$ denote the optimal p-frame set on the lattice. Since $\tilde{C}'^{p*}$ is one of the p-frame sets on the lattice, then we have

$$\frac{s(\tilde{C}^{p*})}{s(C^{p*})} \geq \frac{s(\tilde{C}'^{p*})}{s(C^{p*})} \geq 1 - \frac{2d_z}{\underline{z} + 2d_z}. \tag{3.12}$$

Eq. (3.12) implies that we can use the solution $\tilde{C}^{p*}$ as the approximate solution to the optimal solution. Let the approximation bound be

$$\epsilon = \frac{2d_z}{\underline{z} + 2d_z}. \tag{3.13}$$

Solving (3.13) and combining the upper bound value of d as in (3.7), we have

$$d = 3d_z = \min(\frac{3}{2}(\frac{\epsilon}{1-\epsilon})\underline{z}, \min(3\lambda/10, \mu/3)). \tag{3.14}$$

Eq. (3.14) indicates that when $\epsilon \to 0$,

$$d = 3d_z = \frac{3}{2}(\frac{\epsilon}{1-\epsilon})\underline{z}. \tag{3.15}$$

Eqs. (3.13) and (3.15) imply that we can control the quality of the approximate solution by tuning the lattice spacing d. On the other hand, based on the lattice structure and the definition of the approximation bound, we know that the number of all candidate ponints/frames on the lattice is,

$$|L| = O(1/\epsilon^3). \tag{3.16}$$

3.3.4 Lattice-based Algorithm

With the approximation bound established, the remaining task is to search $\tilde{C}^{p*}$ on L. We design an induction-like approach that builds on the relationship between the solution to the $(p-1)$-frame problem and the solution to the p-frame problem. The key elements that establish the connection are Conditional Optimal Solution (COS) and Conditional Optimal Residual Solution (CORS).

Definition 3.3.3 (Conditional Optimal Solution) *$\forall \tilde{c} \in L$, the COS, $\tilde{U}_j(\tilde{c}) = \{\tilde{C}^{j*} | \tilde{c} \in \tilde{C}^{j*}\}$, is defined as the optimal j-frame set, $j = 1, 2, ..., p$, for the j-frame problem that must include $\tilde{c}$ in the solution set. Also, $\tilde{U}_j(\tilde{c})$ satisfies the VNOC.*

Therefore, we can obtain the optimal solution, $\tilde{C}^{p*}$, on the lattice by searching $\tilde{c}$ over L and its corresponding COS,

$$\tilde{C}^{p*} = \tilde{U}_p(\tilde{c}^*), \tag{3.17}$$

where $\tilde{c}^* = \arg\max_{\tilde{c} \in L} s(\tilde{U}_p(\tilde{c}))$.

Definition 3.3.4 (Conditional Optimal Residual Solution) *Given any COS, $\tilde{U}_{j+1}(\tilde{c}), j = 0, 1, ..., p-1$, we define the j-frame CORS with respect to $\tilde{c}$ as: $\tilde{Q}_j(\tilde{c}) = \tilde{U}_{j+1}(\tilde{c}) - \{\tilde{c}\}$.*

Corollary 2 *$\tilde{Q}_j(\tilde{c})$ is the optimal j-frame set that satisfies,*

- $\tilde{c} \notin \tilde{Q}_j(\tilde{c})$,
- $\{\tilde{c}\} \cup \tilde{Q}_j(\tilde{c})$ *satisfies the VNOC.*

What is interesting is that CORS allows us to establish the relationship between $\tilde{Q}_j$ and $\tilde{Q}_{j-1}$.

Lemma 3

$$\tilde{Q}_j(\tilde{c}_u) = \tilde{Q}_{j-1}(\tilde{c^*}) \cup \{\tilde{c^*}\}, \tag{3.18}$$

where $\tilde{c^*} = \arg\max_{\tilde{c} \in L} s(\tilde{Q}_{j-1}(\tilde{c}) \cup \{\tilde{c}\})$, subject to the constraint that $\{\tilde{c}_u, \tilde{c}\} \cup \tilde{Q}_{j-1}(\tilde{c})$ satisfies the VNOC.

Proof We prove the lemma by contradiction. Notice that the right hand side of (3.18) returns one of the j-frame sets that satisfy the two conditions in Corollary 2, while the left hand side is defined to be the optimal j-frame set that satisfies the same two conditions. Therefore, if we assume (3.18) does not hold, the only possibility is,

$$s(\tilde{Q}_j(\tilde{c}_u)) > s(\tilde{Q}_{j-1}(\tilde{c^*}) \cup \{\tilde{c^*}\}). \tag{3.19}$$

Take an arbitrary frame $\tilde{c}_v \in \tilde{Q}_j(\tilde{c}_u)$ out of $\tilde{Q}_j(\tilde{c}_u)$, the result is $\tilde{Q}_j(\tilde{c}_u) - \{\tilde{c}_v\}$ and according to Lemma 1, we have,

$$s(\tilde{Q}_j(\tilde{c}_u) - \{\tilde{c}_v\}) = s(\tilde{Q}_j(\tilde{c}_u)) - s(\tilde{c}_v). \tag{3.20}$$

Take $\tilde{c}_v$ out of $\tilde{Q}_{j-1}(\tilde{c}_v) \cup \{\tilde{c}_v\}$, the result is $\tilde{Q}_{j-1}(\tilde{c}_v)$ and

$$s(\tilde{Q}_{j-1}(\tilde{c}_v)) = s(\tilde{Q}_{j-1}(\tilde{c}_v) \cup \{\tilde{c}_v\}) - s(\tilde{c}_v). \tag{3.21}$$

Based on (3.19) and the fact that

$$s(\tilde{Q}_{j-1}(\tilde{c^*}) \cup \{\tilde{c^*}\}) \geq s(\tilde{Q}_{j-1}(\tilde{c}_v) \cup \{\tilde{c}_v\}),$$

we have,

$$s(\tilde{Q}_j(\tilde{c}_u)) > s(\tilde{Q}_{j-1}(\tilde{c}_v) \cup \{\tilde{c}_v\}). \tag{3.22}$$

Take $\tilde{c}_v$ out of both sides and combine with (3.20) and (3.21) respectively, we have,

$$s(\tilde{Q}_j(\tilde{c}_u) - \{\tilde{c}_v\}) > s(\tilde{Q}_{j-1}(\tilde{c}_v)). \tag{3.23}$$

The frame set on the right hand side of (3.23), $\tilde{Q}_{j-1}(\tilde{c}_v)$, is defined to be the optimal $(j-1)$-frame set that satisfies the two conditions in Corollary 2 while the frame set on left hand side, $\tilde{Q}_j(\tilde{c}_u) - \{\tilde{c}_v\}$, is only one of the $(j-1)$-frame sets that satisfy the two conditions. Contradiction occurs.

■

It is worth mentioning that it takes $O(p)$ time to check if $(\{\tilde{c}_u, \tilde{c}\} \cup \tilde{Q}_j(\tilde{c}))$ satisfies the VNOC. Because $\{\tilde{c}\} \cup \tilde{Q}_j(\tilde{c}) = \tilde{U}_{j+1}(\tilde{c})$ satisfies the VNOC as defined in Definition 3.3.3 and thus we only need to check if $\{\tilde{c}_u\} \cup \tilde{U}_{j+1}(\tilde{c})$ satisfies the VNOC, which takes $O(p)$ time.

Eq. (3.17) implies that we can obtain the approximation solution $\tilde{C}^{p*}$ from $\tilde{U}_p$. Definition 3.3.4 indicates that we can obtain $\tilde{U}_p$ from $\tilde{Q}_{p-1}$. Now Lemma 3 implies that we can construct $\tilde{Q}_j$ from $\tilde{Q}_{j-1}$, $j = 1, 2, ..., p-1$. Considering the fact that $\tilde{Q}_0 = \phi$, this allows us to establish the algorithm using an induction-like approach. Algorithm 1 shows the complete lattice-based algorithm. Considering any candidate frame $\tilde{c} \in L$, we precalculate the satisfaction values for all the $|L|$ candidate frames and store the values in a lookup table to avoid redundant calculation. Given any candidate frame $\tilde{c}_u \in L$ as the input, the lookup function l returns the satisfaction value of $\tilde{c}_u$, $l(\tilde{c}_u) = s(\tilde{c}_u)$. We implement the lookup function using the array, $l[u] = s(\tilde{c}_u)$. From the pseudo code in Algorithm 1, it is not difficult to know that,

Theorem 1 *Algorithm 1 runs in $O(n/\epsilon^3 + p^2/\epsilon^6)$ time.*

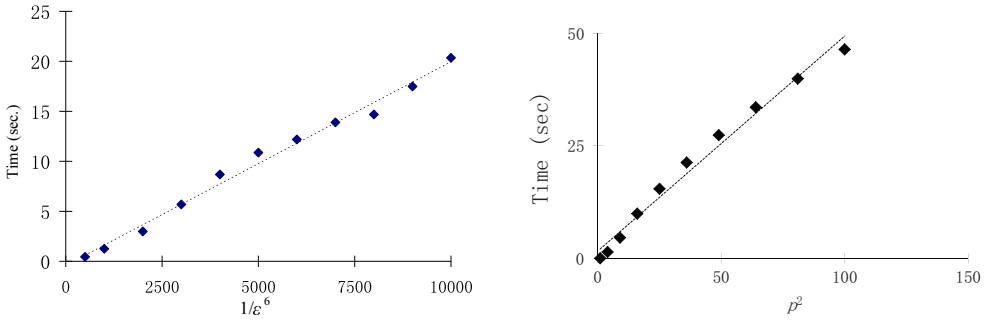
Algorithm 1: Lattice-based Algorithm

```
1  begin
2      for j ← 1 to |L| do                                              O(1/ε^3)
3          l[j] = s(c̃_j)                                                O(n)
4          Q̃_0(c̃_j) = ∅;                                                O(1)
5          s(Q̃_0(c̃_j)) = 0;                                            O(1)
6      end
7      for k ← 1 to p do                                                O(p)
8          C̃^{k*} = ∅;                                                  O(1)
9          s(C̃^{k*}) = 0;                                               O(1)
10         for u ← 1 to |L| do                          update C̃^{k*}, O(1/ε^3)
11             if s(C̃^{k*}) < s(Q̃_{k-1}(c̃_u)) + l[u] then
12                 C̃^{k*} = Q̃_{k-1}(c̃_u) ∪ {c̃_u};                       O(1)
13                 s(C̃^{k*}) = s(Q̃_{k-1}(c̃_u)) + l[u];                  O(1)
14             end
15         end
16         for u ← 1 to |L| do                          update Q̃_k(c̃_u), O(1/ε^3)
17             Q̃_k(c̃_u) = Q̃_{k-1}(c̃_u) ∪ ∅;                             O(1)
18             s(Q̃_k(c̃_u)) = s(Q̃_{k-1}(c̃_u));                           O(1)
19             for v ← 1 to |L| do                                      O(1/ε^3)
20                 if s(Q̃_k(c̃_u)) < s(Q̃_{k-1}(c̃_v)) + l[v] AND
21                 {c̃_u, c̃_v} ∪ Q̃_{k-1}(c̃_v) satisfies the VNOC        O(p) then
22                     Q̃_k(c̃_u) = Q̃_{k-1}(c̃_v) ∪ {c̃_v};                 O(1)
23                     s(Q̃_k(c̃_u)) = s(Q̃_{k-1}(c̃_v)) + l[v];            O(1)
24                 end
25             end
26         end
27     end
28     return C̃^{p*};
29 end
```

3.4 Experimental Results

We have implemented the algorithm using Java. The computer used is a desktop computer with an Intel Core 2 Duo 2.13GHz CPU and 2GB RAM. The operating system is Windows XP. In experiments, we test the algorithm speed with different parameter settings including the number of request n, the number of camera frames p, and the approximation bound ϵ.

In the experiments, both triangular and rectangular inputs are randomly generated. First, s_d points in V are uniformly generated across the reachable field of view. These points indicate the locations of interest and are referred to as seeds. Each seed is associated with a random radius of interest. To generate a request, we randomly assign it to one seed. For a triangular request, three 2-D points are randomly generated within the radius of the corresponding seed as the vertices of the triangle. For a rectangular request, a 2-D point is randomly generated as the center of the rectangular region within the radius of corresponding seed and then two random numbers are generated as the width and height of the request. Finally, the resolution value of the request is uniformly randomly generated across the resolution range $[\underline{z}, \overline{z}]$.

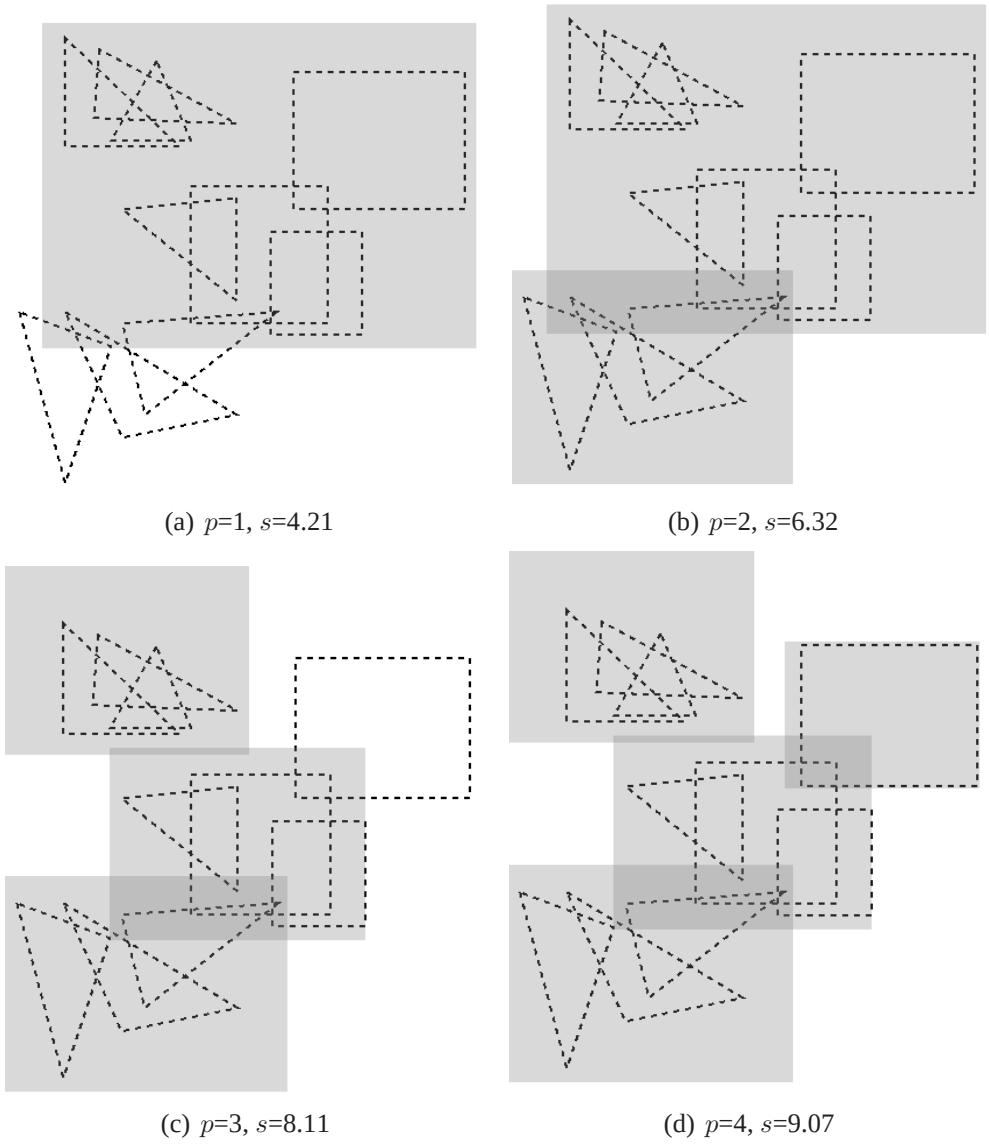


(a) The computation time vs. the approximation bound ϵ, (n=100, p=2). Note that the horizontal axis is $1/\epsilon^6$.

(b) The computation time vs. the number of frames p, (n=100, ϵ=0.25). Note the horizontal axis is p^2.

Fig. 3.2. Speed testing results.

Across the experiment, we set w=80, h=60, $\underline{z}$=5, $\overline{z}$=15 and s_d=4. For each parameter setting, 50 trials have been carried out for averaged performance. The simulation results indicate the linear relationship between the computation time and n. Fig. 3.2 illustrates the relationship between the computation time and the parameters p and ϵ. The results are consistent with our Big O notion complexity analysis. In Fig. 3.2(a), the computational time is linear to $1/\epsilon^6$. In Fig. 3.2(b), it even shows a trend of sub-linear with respect to p^2.



(a) p=1, s=4.21 (b) p=2, s=6.32

(c) p=3, s=8.11 (d) p=4, s=9.07

Fig. 3.3. Sample outputs when p increases for a fixed input set $n = 10$.

This may be due to the fact that when p is larger and frames have higher chance to violate the virtual non-overlapping condition, it takes less time to check if the frames satisfy the condition in lines 20 and 21 in Algorithm 1.

Fig. 3.3 shows how the output of the algorithm for a fixed set of inputs (n=10) changes when p increases from 1 to 4. It shows that our algorithm reasonably allocates camera frames in each case.

3.5 Conclusion

In this section, we have formulated the least overlapping p-frame problem with non-partial coverage as an optimization problem. A lattice-based approximation algorithm was proposed for solving the problem. Given n requests and p camera frames, the algorithm runs in $O(n/\epsilon^3 + p^2/\epsilon^6)$ time with the approximation bound ϵ. We have implemented the algorithm and tested it on random inputs. The experimental results are consistent with our theoretical analysis.

In future work, we will explore the new geometric data structures to improve complexity results. We will also develop algorithms for different variations of the problem such as allowing camera frames to overlap with each others.

The proposed approximation p-frame algorithm has been applied to the crowd surveillance system as in Section 2. However, the complexity of the algorithm is very sensitive to the approximation bound ϵ. When ϵ becomes small, the computation time increases dramatically. This prohibits the system's usefulness to applications where accurate solution is required. In the next section, we introduce an efficient exact algorithm when $p = 2$.

4. MOMR++ ALGORITHM: EXACT 2-FRAME ALGORITHM FOR CROWD SURVEILLANCE SYSTEM

4.1 Introduction

In Section 3, we introduce an approximation p-frame algorithm for the crowd surveillance system in Section 2. However, the speed of the approximation algorithm is very sensitive to the approximation bound ϵ. On the other hand, in real applications, it is rare to have many cameras due to various constraints such as cost, power supply, synchronization between cameras, and maintenance. This encourages us to explore online algorithms when p is small. In this section, we introduce an efficient exact algorithm when $p = 2$. Fig. 4.1 illustrates a 2-frame problem instance.

We assume the frames have no overlap on their coverage. We propose a series of algorithms to search for the solution that maximizes the overall request satisfaction. Our algorithms solve the 2-frame problem in $O(n^2)$, $O(n^2m)$ and $O(n^3)$ times for fixed, m discrete, and continuous resolution levels, respectively. We have implemented all the algorithms and compared them with our previous work in the last section. The experimental results are consistent with our complexity analysis. We begin with the related work.

4.2 Related Work

The 2-frame problem relates to the 2-center problem, networked robotics and multiple camera surveillance.

The p-frame problem is structurally similar to the p-center facility location problem. Given n request points in $\mathbb{R}^d, (d = 1, 2, ...)$, the task is to optimally allocate p points as service centers to minimize the maximum distance between points and their nearest service centers. The distance metric are usually Euclidean (l^2) or rectilinear (l^∞). The Euclidean p-center problem is NP-hard [63]. Eppstein [53] proposes an $O(n \log^2 n)$ algorithm for the Euclidean 2-center problem. Arkin *et al.* [54] replace the service points by orthogonal

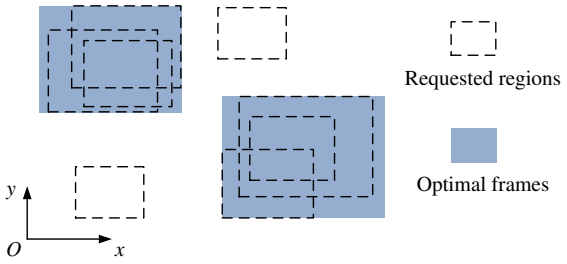


Fig. 4.1. An illustration of the non-overlapping 2-frame problem.

boxes in $\mathbb{R}^3$ and propose an approximation algorithm that runs in $O(n \min(\lg n, 1/\epsilon) + (\lg n)/\epsilon^2)$ for the Euclidean 2-box covering problem. Alt *et al.* [55] propose a $(1+\epsilon)$-approximation algorithm that runs in $O(n^{O(m)})$, where $\epsilon = O(1/m)$, for the multiple disk covering problem. The rectilinear p-center problem is also NP-hard [63]. Bespamyatnikh and Kirkpatrick [64] propose a linear time algorithm for the rectilinear 2-center problem. Ko *et al.* [65] propose a 2-approximation algorithm for solving the rectangular p-center problem and prove that factor 2 is optimal. The requests in these problems are all points instead of polygonal regions as those in the p-frame problem. The objective of the p-frame problem is to maximize the satisfaction, which is not a distance metric.

The task of p networked pan-tilt-zoom cameras taking p frames in the remote environment can be viewed as a special case of networked tele-operation, where each robotic camera has 3 Degrees of Freedom (DOF). Based on the taxonomy by Chong *et al.* [5], these systems belong to Multiple Operator Multiple Robot (MOMR) systems [10,11]. Liu *et al.*, [12] develop a competitive MOMR system where two operators control two multi-finger robotic hands, respectively, in a game setting. Huang *et al.*, [66] propose a criterion called degree of satisfaction (DOS) to evaluate the performance of competitive MOMR systems. Wang *et al.*, [67] propose an internet-based MOMR system that allows multiple students to control two robot soccer teams for competition. Our work emphasizes on the geometric coverage attributes of the robotic camera and addresses the MOMR problem in an optimization framework.

Table 4.1
Algorithm and system development for p-frame problems

Algorithm	Resolution	Complexity
p-frame approximate	Continuous	$O(n/\epsilon^3 + p^2/\epsilon^6)$
2-frame exact (this section)	fixed, m discrete, and continuous	$O(n^2)$, $O(n^2 m)$, and $O(n^3)$

The p-frame problem can be applied to multiple camera surveillance systems, especially those with multiple active cameras. Fiore *et al.* [35] propose a dual-camera system with a wide-angle static camera and a PTZ camera for pedestrian surveillance. The two cameras share the same point of view. While the wide-angle static camera monitors the scene and detects pre-defined individual human activities (e.g., loitering), the PTZ camera takes high-resolution images of the human for close-up observation. Lim *et al.* [40] propose a multiple camera system, which consists of one wide-angle static camera and multiple PTZ cameras. It constructs the observation task for each single object as a "task visibility interval" (TVI) based on accurate predicted states of the objects during their entire durations in the FOV. It solves the PTZ camera planning issue by modeling it as a maximum flow problem. A recent live system in this category can be found in [68]. Different from these existing work, the solution to the p-frame problem can be applied to optimally control PTZ camera parameters such that the camera coverage-resolution tradeoff is achieved by maximizing the satisfaction level of the observation to all objects. This also enables group watching which is very meaningful in many applications.

Our group has been researching on developing intelligent vision systems and algorithms using robotic cameras for a variety of applications [4]. In [69], we formulate the p-frame problem and propose an approximation algorithm that runs in $O(n/\epsilon^3 + p^2/\epsilon^6)$ time. An autonomous observation system that adopts this algorithm with multiple PTZ cameras has been introduced in [70]. However, the computation time of the algorithm is very sensitive to the approximation bound ϵ. It proves to be inviable for problems where exact or accurate solutions are required. As in [70], it can only handle less than 50 observation requests with $p = 2, \epsilon = 0.27$ in 0.5 second, which is equivalent to only 2 frames per

second. In this section, we extend the single frame selection algorithm in [47] to the cases where $p = 2$ and propose a series of exact algorithms for solving the 2-frame problem with different camera resolution configurations. Table 4.1 summarizes the current progress on p-frame problem.

4.3 Problem Definition

We begin with the definition of the inputs and outputs. Necessary assumptions are presented. Then we introduce the request satisfaction metric so that we can formulate the problem as a geometric optimization problem.

4.3.1 Input and Output

As illustrated in Fig. 4.1, we assume all camera frames and requests are rectangular and each side of the rectangle is axis-parallel. The i-th request is defined as $r_i = [\underline{x}_i, \underline{y}_i, \overline{x}_i, \overline{y}_i, z_i]$, where $(\underline{x}_i, \underline{y}_i)$ and $(\overline{x}_i, \overline{y}_i)$ denote the bottom-left and top-right corners of the rectangular requested region, respectively; $z_i \in Z$ specifies the desired resolution level, which indicates that each pixel in image corresponds to a $z_i \times z_i$ square area in the scene, and Z is the set of all possible resolution levels. Therefore, bigger $z \in Z$ indicates bigger camera frame coverage and thus can be interpreted as the reciprocal of the conventional concept of resolution. When the PTZ cameras have a fixed resolution level, $Z = \{z^0\}$, where z^0 is a constant; When cameras have m discrete resolution levels, $Z = \{z^1, z^2, ..., z^m\}$; Cameras can also have continuous resolution range $Z = [\underline{z}, \overline{z}]$, where $\underline{z}$ and $\overline{z}$ denote the lower and upper bounds of the resolution level, respectively. The input of the 2-frame problem is a set of n requests $R = \{r_i | i = 1, 2, ..., n\}$. We define the request index set as $P = \{1, 2, ..., n\}$.

A solution to the 2-frame problem consists of two camera frames. Assuming a fixed aspect ratio (e.g. 4:3), a camera frame can be defined as $c = [x, y, z]$, where (x, y) denotes the center point of the rectangular frame and $z \in Z$ specifies the resolution level of the

camera frame. Here we consider the coverage of the camera to be rectangular according to the camera configuration space. Therefore the width and height of the camera frame can be represented as $4z$ and $3z$, respectively. The four corners of the frame are located at

$$(x \pm \frac{4z}{2}, y \pm \frac{3z}{2}),$$

respectively.

Given w and h are the camera pan and tilt ranges, respectively, then $\mathcal{C} = [0, w] \times [0, h] \times Z$ defines the set of all candidate frames. Therefore, $\mathcal{C}^2$ indicates the solution space for the 2-frame problem. Let us define any candidate solution to the 2-frame problem as $(c_1, c_2) \in \mathcal{C}^2$. The objective of the 2-frame problem is to find the optimal solution $(c_1^*, c_2^*) \in \mathcal{C}^2$ that best satisfies the requests.

4.3.2 Assumptions

We assume that the two frames are either taken from two cameras that share the same workspace or taken from the same camera. Therefore, if a location can be covered by a frame, the other frame can cover that location, too.

We assume any solution (c_1, c_2) to the 2-frame problem satisfies the Non-Overlapping Condition.

Definition 4.3.1 (Non-Overlapping Condition (NOC)) *Given a 2-frame set $(c_1, c_2) \in \mathcal{C}^2$, it satisfies the NOC, if*

$$c_1 \cap c_2 = \phi \tag{4.1}$$

where we abuse the set operator "$\cap$" to represent the 2-D regional overlapping relationship between frames as a convention in the rest of this section. For example, in (4.1), $c_1 \cap c_2$ represents the overlapping region of frames c_1 and c_2.

The NOC increases the overall coverage of frames over requests since no request is redundantly covered by both frames and thus is a favorable solution to applications where searching ability is important.

4.3.3 Satisfaction Metric

For completeness, we briefly review the formulation of the objective function as proposed in [69]. We measure the "satisfaction" level of a request by comparing its requested resolution with that of the camera frame, which fully contains the region of the request. We define the Resolution Ratio with Non-Partial Coverage (RRNPC) metric. Given a request $r_i = [\underline{x}_i, \underline{y}_i, \overline{x}_i, \overline{y}_i, z_i]$ and a camera frame $c = [x, y, z]$, the satisfaction of request r_i with respect to c is computed as

$$s(c, r_i) = I(c, r_i) \cdot \min(\frac{z_i}{z}, 1), \tag{4.2}$$

where $I(c, r_i)$ is an indicator function that describes the non-partial coverage condition,

$$I(c, r_i) = \begin{cases} 1 & \text{if} \quad r_i \subseteq c, \\ 0 & \text{otherwise}, \end{cases} \tag{4.3}$$

where we abuse the set operator $\subseteq$ to represent the 2-D regional relationship between frame(s) and request(s) in the rest of this section. In (4.3), $r_i \subseteq c$ represents that the region of r_i is fully contained in that of c. Eq. (4.2) takes into account both camera coverage (first term in (4.2)) and camera resolution (second term in (4.2)) so that a coverage-resolution tradeoff is achieved.

From (4.2), the request satisfaction fulfilled by a frame c is calculated as

$$s(c) = \sum_{i=1}^{n} I(c, r_i) \cdot \min\,(\frac{z_i}{z}, 1), \tag{4.4}$$

where we overload the function $s(\cdot)$ by taking a frame as the input. Eq. (4.4) shows that evaluating a candidate frame takes $O(n)$ time.

4.3.4 Problem Formulation

With the NOC assumption, the overall satisfaction of n requests $(r_1, r_2, ..., r_n)$ served by a solution $(c_1, c_2) \in \mathcal{C}^2$ is,

$$\begin{aligned} s(c_1, c_2) &= \sum_{i=1}^{n} \sum_{j=1}^{2} I(c_j, r_i) \cdot \min(\frac{z_i}{z_j}, 1) \\ &= s(c_1) + s(c_2), \end{aligned} \tag{4.5}$$

where we overload the function $s(\cdot)$ by taking a 2-frame set as the input. Here we are interested in cases such that $s(c_1) > 0$ and $s(c_2) > 0$. If either $s(c_1) = 0$ or $s(c_2) = 0$, the 2-frame problem degenerates to a single frame problem.

Eq. (4.5) shows that the satisfaction of any candidate (c_1, c_2) can be computed in $O(n)$ time. Now we can formulate the non-overlapping 2-frame problem as a maximization problem,

$$(c_1^*, c_2^*) = \arg \max_{(c_1, c_2) \in \mathbb{R}^6} s(c_1, c_2).$$

4.4 Algorithms

4.4.1 Feasibility Condition

We start with analyzing the structural property of any feasible solution.

Definition 4.4.1 (Separation) *For any interval $[x_1, x_2]$, we define the 2-D point set*

$$S_e^X(x_1, x_2) = \{(x, y) \in \mathbb{R}^2 | x_1 \leq x \leq x_2\}$$

as an x-separation. Similarly, we define

$$S_e^Y(y_1, y_2) = \{(x, y) \in \mathbb{R}^2 | y_1 \leq y \leq y_2\}$$

as a y-separation for interval $[y_1, y_2]$.

For any feasible solution $(c_1, c_2) = ([x_1, y_1, z_1], [x_2, y_2, z_2])$, we define,

$$\begin{aligned} S_e^X(c_1, c_2) =& S_e^X(x_1 + \frac{4z_1}{2}, x_2 - \frac{4z_2}{2}) \\ & \cup S_e^X(x_2 + \frac{4z_2}{2}, x_1 - \frac{4z_1}{2}), \end{aligned} \tag{4.6}$$

$$\begin{aligned} S_e^Y(c_1, c_2) =& S_e^Y(y_1 + \frac{3z_2}{2}, y_2 - \frac{3z_2}{2}) \\ & \cup S_e^Y(y_2 + \frac{3z_1}{2}, y_1 - \frac{3z_1}{2}), \end{aligned} \tag{4.7}$$

as illustrated in Fig. 4.2. Intuitively, (4.6) and (4.7) define the "gap" between frames.

Lemma 4 (Feasibility condition) *Given any feasible solution (c_1, c_2), it must have at least one non-empty separation as defined in (4.6) and (4.7),*

$$S_e^X(c_1, c_2) \cup S_e^Y(c_1, c_2) \neq \phi.$$

Lemma 4 is straightforward from the non-overlapping condition.

Given the optimal solution (c_1^*, c_2^*), if $S_e^X(c_1^*, c_2^*) \neq \phi$, we call the problem is x-separable. Similarly, if $S_e^Y(c_1^*, c_2^*) \neq \phi$, we call the problem is y-separable. These two cases are not mutually exclusive. Without loss of generality, we focus on x-separable problem in the rest of this section.

As a convention from here on, we use c_1 to represent the "left" frame of a solution, and c_2 to represent the "right" frame as shown in Fig. 4.2 for the x-separable problem. Hence, (4.6) can be simplified as,

$$S_e^X(c_1, c_2) = S_e^X(x_1 + \frac{4z_1}{2}, x_2 - \frac{4z_2}{2}).$$

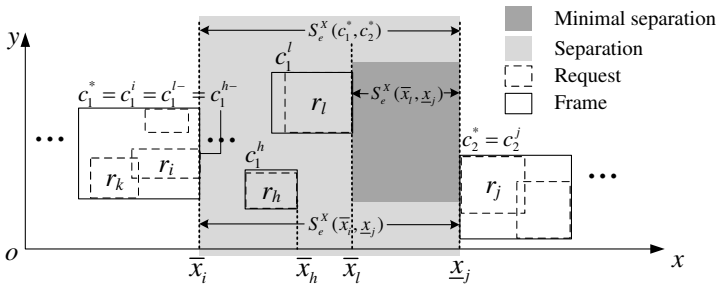


Fig. 4.2. An illustration of the optimal x-separable solution. At least one optimal solution $(c_1^*, c_2^*) = (c_1^i, c_2^j)$ corresponds to a separation is a minimal separation. r_l is the closest request on the left hand side of line x=$\underline{x}_j$. r_h is the second closest request on the left hand side of line x=$\underline{x}_j$. c_1^{l-} can be incrementally computed by comparing c_1^l and c_1^{h-}, as in (4.15).

4.4.2 Optimality Condition

Lemma 4 defines the necessary condition for any feasible solution. Unfortunately, there are infinite number of separations. Next, we show how to reduce the problem to finite candidate separations to assist the search for the optimal solution.

Given the optimal solution (c_1^*, c_2^*) as illustrated in Fig. 4.2, slightly sliding c_1^* to the right does not change its satisfaction level until its left side overlaps with that of r_k (i.e., $x_1^* - 4z_1^*/2 = \underline{x}_k$), because neither the camera resolution nor the camera-request coverage relationship changes. However, if we slide c_1^* slightly to the left so that its right side is on the left hand side of that of r_i, i.e., $x_1^* + 4z_1^*/2 < \overline{x}_i$, the satisfaction level decreases because the frame loses the complete coverage over request r_i. Similar arguments can apply to c_2^*. This tells us that at least one optimal solution is structurally defined by a separation, which corresponds to a pair of request sides.

Lemma 5 (Optimality condition) *For any x-separable problem, there must exist one optimal solution, $(c_1', c_2') = ([x_1', y_1', z_1'], [x_2', y_2', z_2'])$ and a non-empty separation $S_e^X(\overline{x}_i, \underline{x}_j)$, $i, j \in P$, such that*

$$r_i \subseteq c_1' \text{ and } x_1' + \frac{4z_1'}{2} = \overline{x}_i;$$
$$r_j \subseteq c_2' \text{ and } x_2' - \frac{4z_j'}{2} = \underline{x}_j.$$

Thus $S_e^X(\overline{x}_i, \underline{x}_j) = S_e^X(c_1', c_2')$ is the non-empty separation for this optimal solution.

Proof Given an optimal solution (c_1^*, c_2^*) as shown in Fig. 4.2, we have,

$$s(c_2^*) = \sum_{k=1}^{n} I(c_2^*, r_k) \min\left(\frac{z_k}{z_2^*}, 1\right). \tag{4.8}$$

Let R_2^* represent the set of requests which are fully enclosed by c_2^*. Then (4.8) is re-written as,

$$\begin{aligned} s(c_2^*) &= \sum_{r_k \in R_2^*} I(c_2^*, r_k) \min\left(\frac{z_k}{z_2^*}, 1\right) \\ &= \sum_{r_k \in R_2^*} \min\left(\frac{z_k}{z_2^*}, 1\right). \end{aligned} \tag{4.9}$$

Let $\underline{x}_j$ be the smallest x-coordinate of R_2^*,

$$\underline{x}_j = \min_{r_k \in R_2^*} \underline{x}_k.$$

For $c_2^* = [x_2^*, y_2^*, z_2^*]$, there exists a frame $c_2' = [x_2', y_2', z_2']$, such that $y_2' = y_2^*$, $z_2' = z_2^*$ and $x_2' - 4z_2'/2 = x_2' - 4z_2^*/2 = \underline{x}_j$. Intuitively, c_2' is the frame similar to c_2^* except that its left side overlaps with line $x = \underline{x}_j$. Define R_2' as the set of requests that are completely enclosed by c_2', we have,

$$s(c_2') = \sum_{r_k \in R_2'} I(c_2', r_k) \min(\frac{z_k}{z_2^*}, 1)$$
$$= \sum_{r_k \in R_2'} \min(\frac{z_k}{z_2^*}, 1). \quad (4.10)$$

Since $r_j \in R_2^*$, therefore $r_j \subseteq c_2^*$. We have $x_2' - 4z_2^*/2 = \underline{x}_j \geq x_2^* - 4z_2^*/2$, and thus $x_2' \geq x_2^*$. Therefore, $x_2' + 4z_2^*/2 \geq x_2^* + 4z_2^*/2$. For any $r_k = [\underline{x}_k, \underline{y}_k, \overline{x}_k, \overline{y}_k, z_k] \in R_2^*$, we have,

$$\underline{x}_k \geq \underline{x}_j = x_2' - 4z_2'/2,$$
$$\overline{x}_k \leq x_2^* + 4z_2^*/2 \leq x_2' + 4z_2'/2,$$
$$\underline{y}_k \geq y_2^* - 3z_2^*/2 = y_2' - 3z_2'/2,$$
$$\overline{y}_k \leq y_2^* + 3z_2^*/2 = y_2' + 3z_2'/2.$$

Therefore, $r_k \subseteq c_2'$ and $R_2^* \subseteq R_2'$.

Comparing (4.9) and (4.10), we have $s(c_2^*) \leq s(c_2')$. However, if $s(c_2^*) < s(c_2')$, we can replace c_2^* with c_2' to obtain a better non-overlapping solution, which contradicts the fact that (c_1^*, c_2^*) is optimal. Therefore, $s(c_2^*) = s(c_2')$ and c_2' is optimal. Similarly, we can find a frame c_1' with $\overline{y}_1' = \overline{y}_1^*$, $z_1' = z_1^*$, and $x_1' + 4z_1'/2 = \overline{x}_i$ for c_1^*. Therefore, (c_1', c_2') is an optimal solution. $S_e^X(\overline{x}_i, \underline{x}_j) = S_e^X(c_1', c_2')$ is the corresponding separation for (c_1', c_2'). ■

Lemma 5 defines the necessary condition for one optimal solution. Each non-empty separation $S_e^X(\overline{x}_i, \underline{x}_j)$ corresponds to a candidate solution. This leads to the exhaustive approach as follows.

4.4.3 Exhaustive Search

Based on Lemma 5, for each non-empty separation $S_e^X(\overline{x}_i, \underline{x}_j)$, we reduce the 2-frame problem to two single frame problems, each finding the optimal frame that has its one side

Algorithm 2: Exhaustive Search Algorithm for x-Separable Non-Overlapping 2-Frame Problem (ES-XS-2)

Input: Request set R.
Output: (c_1^*, c_2^*)

```
1  begin
2      foreach S_e^X(x̄_i, x_j)                                   O(n^2)
3      do
4          if x̄_i ≤ x_j then
5              Compute c_1^i;                                    T_1
6              Compute c_2^j;                                    T_1
7          end
8      end
9      return the best (c_1^i, c_2^j) pair;                      O(1)
10 end
```

overlapping with one boundary of the separation. We define these two constrained optimal frames,

$$c_1^i = \arg \max_{c=(x,y,z)} s(c), \text{ s.t. } r_i \subseteq c \text{ and } x + \frac{4z}{2} = \overline{x}_i, \tag{4.11}$$

$$c_2^j = \arg \max_{c=(x,y,z)} s(c), \text{ s.t. } r_j \subseteq c \text{ and } x - \frac{4z}{2} = \underline{x}_j. \tag{4.12}$$

We can find one optimal solution by exhaustively enumerating all $O(n^2)$ non-empty separations $S_e^X(\overline{x}_i, \underline{x}_j),\ i, j \in P$. For each $S_e^X(\overline{x}_i, \underline{x}_j)$, the corresponding candidate solution (c_1^i, c_2^j) can be obtained by solving the two single frame sub-problems as in (4.11) and (4.12), respectively. Algorithm 2 summarizes the exhaustive search approach.

It is noticed that in lines 5 and 6 of Algorithm 2, it requires the subroutines that solve the two sub-problems as in (4.11) and (4.12), respectively. Both subroutines run in $T_1(n)$ time. The implementation of the subroutines and $T_1(n)$ depend on different camera resolution configurations, which will be discussed in details later. The exhaustive search as in Algorithm 2 runs in $O(n^3) + O(n^2) \cdot T(n)$ time.

4.4.4 Sweeping of Separation Boundaries

However, further observation reveals a more efficient approach. Instead of enumerating all $O(n^2)$ separations $S_e^X(\overline{x}_i, \underline{x}_j),\ i,j \in P$, we only need to consider $O(n)$ special separations.

Given any non-empty separation $S_e^X(\overline{x}_i, \underline{x}_j)$ as shown in Fig. 4.2, we can always contract it to a smaller, non-negative width by moving the left separation boundary to the right, until the left boundary overlaps with a right request side, which is the closest to the right separation boundary (e.g., $\overline{x}_l$ in Fig. 4.2). We define this separation with smallest non-negative width as the minimal separation.

Definition 4.4.2 (Minimal separation) *Given any non-empty separation $S_e^X(\overline{x}_l, \underline{x}_j)$, defined by requests r_l and r_j, $l,j \in P$, we define it as the minimal separation with respect to r_j if r_l is the closest request to line $x = \underline{x}_j$ among those on the left hand side of $x = \underline{x}_j$,*

$$l = \arg\min_{k \in P} (\underline{x}_j - \overline{x}_k) \quad \textit{s.t. } \overline{x}_k \leq \underline{x}_j.$$

Given the optimal solution $(c_1^*, c_2^*) = (c_1^i, c_2^j)$ and its corresponding separation $S_e^X(\overline{x}_i, \underline{x}_j)$ as in Fig. 4.2, the corresponding minimal separation is $S_e^X(\overline{x}_l, \underline{x}_j)$ as illustrated by the striped area. It is obvious that $c_1^* = c_1^i$ is the optimal frame which is on the left hand side of both $S_e^X(\overline{x}_i, \underline{x}_j)$ and $S_e^X(\overline{x}_l, \underline{x}_j)$. We define the optimal frame on the left hand side of a separation as follows. Given any left separation boundary at $x = \overline{x}_l,\ l \in P$, we define frame c_1^{l-} as the optimal frame that is on the left hand side of the left separation boundary,

$$c_1^{l-} = \arg \max_{c_1^k,\ k \in P} s(c_1^k),\ \text{s.t. } x_1^k + \frac{4z_1^k}{2} \leq \overline{x}_l. \tag{4.13}$$

Therefore, we can find an optimal solution by enumerating all $O(n)$ minimal separations. For each minimal separation $S_e^X(\overline{x}_l, \underline{x}_j)$, we compute the corresponding c_1^{l-} and c_2^j.

The remaining question is how to efficiently compute c_1^{l-} for each minimal separation. Direct computation based on (4.13) requires to compute $O(n)$ constrained optimal single frames as in (4.11) and compare all of them. Given the minimal separation $S_e^X(\overline{x}_l, \underline{x}_j)$, let r_h be the second closest request left to line $x = \underline{x}_j$, as illustrated in Fig. 4.2,

$$h = \arg\min_{k \in P} (\underline{x}_j - \overline{x}_k), \text{ s.t. } \overline{x}_k \leq \overline{x}_l \leq \underline{x}_j. \tag{4.14}$$

Then the computations of c_1^{h-} and c_1^{l-} based on (4.13) only differ in computing $s(c_1^l)$. Therefore, we have,

$$c_1^{l-} = \begin{cases} c_1^{h-} \text{ if } s(c_1^{h-}) > s(c_1^l), \\ c_1^l \text{ otherwise}. \end{cases} \tag{4.15}$$

Eqs. (4.14) and (4.15) suggest an incremental approach to calculate $c_1^{l-},\ l \in P$. We search for all candidate left separation boundaries, which are defined by right request sides $\{\overline{x}_l,\ l \in P\}$, from left $(x = -\infty)$ to right $(x = \infty)$ and incrementally compute each $c_1^{l-},\ l \in P$, as in (4.15).

To search for all minimal separations, we sort all vertical request sides and sweep a separation, which is defined by the vertical request sides, from left to right as illustrated in Fig. 4.3. In each sweeping step, we either contract the separation by moving its left boundary toward right or expand the separation by moving its right boundary toward right.

- If the separation is not a minimal separation, we contract the separation by moving the left boundary to its next candidate position. The optimal frame on the left hand side of the new separation is computed as in (4.15). The contraction from Fig. 4.3(f) to Fig. 4.3(g) illustrates these operations.

- If the separation is a minimal separation. We compute the optimal frame on the right hand side of the separation as in (4.12). Since the optimal frame on the left hand side of the separation is maintained as described above, combining the two frames forms a candidate solution. After that, we expand the separation by moving

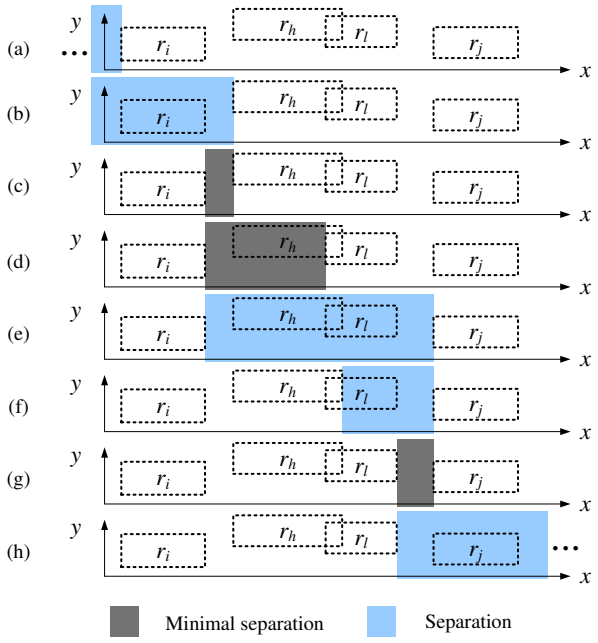


Fig. 4.3. An illustration of the sweeping of separation boundaries. During sweeping from left to right, if the separation is not a minimal separation, we contract the separation by moving its left boundary to its next candidate position and the optimal frame on its left hand side is computed as in (4.15). If the separation is a minimal separation, its right frame is computed as in (4.12), and forms a candidate solution with the optimal left frame maintained earlier.

the right boundary to its next candidate position and a new sweeping step starts. The expansion from Fig. 4.3(d) to Fig. 4.3(e) illustrates these operations.

We summarize the sweeping search algorithm for solving x-separable 2-frame problem in Algorithm 3. Since both the separation need to be contracted and expanded $O(n)$ times, respectively, the sweeping search as in Algorithm 3 runs in $O(n)T_1(n)$ time.

4.4.5 Algorithm Complexity with Different Camera Resolution Configurations

We turn to the implementations of the subroutines for solving the sub-problems as in (4.11) and (4.12), under different camera resolution configurations. Without loss of

Algorithm 3: Sweeping Search Algorithm for x-Separable 2-Frame Problem (SS-XS-2)

Input: Request set R;
Output: (c_1^*, c_2^*);

begin
  Sort left sides of $R : \underline{B} = [\underline{b}[1], ..., \underline{b}[n]]$; $O(n \log n)$
  Sort right sides of $R : \overline{B} = [\overline{b}[1], ..., \overline{b}[n]]$; $O(n \log n)$
  Sort top sides of R; $O(n \log n)$
  Sort bottom sides of R; $O(n \log n)$
  Sort requested resolutions of R; $O(n \log n)$
  $c_1^- = \phi$; $c_1^* = \phi$; $c_2^* = \phi$; $O(1)$
  $u = 0;\ v = 1$; $O(1)$
  while $v < n$ $O(n)$
  do
    if $\overline{b}[u+1] > \underline{b}[v]$ `#Minimal separation`
    then
      Find $\underline{b}[v]$ belongs to r_j; $O(1)$
      Compute c_2^j as in (4.12) $T_1(n)$
      if $s(c_1^*) + s(c_2^*) < s(c_1^-) + s(c_2^j)$ **then**
        $(c_1^*, c_2^*) = (c_1^-, c_2^j)$ $O(1)$
      end
      $v = v + 1$; $O(1)$
    end
    else
      $u = u + 1$; $O(1)$
      Find $\overline{b}[u]$ belongs to r_l; $O(1)$
      Compute c_1^l as in (4.11); $T_1(n)$
      if $s(c^-) < s(c_1^l)$ **then**
        $c_1^- = c_1^l$; $O(1)$
      end
    end
  end
  return (c_1^*, c_2^*) $O(1)$
end

generality, we only discuss the subroutine that calculates the optimal single frame on the right hand side of the separation, c_2^j, as in (4.12).

A fixed camera resolution

We first consider the case in which the cameras have a fixed resolution $z = z_0$. Given the right separation boundary at $x = \underline{x}_j$ as shown in Fig. 4.4. Recall c_2^j satisfies $x_2^j - 4z_2^j/2 = \underline{x}_j$ and $r_j \subseteq c_2^j$. Since the camera frame has a fixed size (resolution), we can align the left side of a candidate frame c_2 with line $x = \underline{x}_j$ and slide c_2 along the line $x = \underline{x}_j$ while maintaining $r_j \subseteq c_2$ to search for all candidate frames. Based on the RRNPC metric in (4.2), we know that $s(c_2)$ changes only at the moments when one horizontal side of c_2 overlaps with that of a request. Therefore, there are totally $O(n)$ candidate frames. Evaluating all of the candidate frames takes $O(n^2)$ time. However since we have sorted horizontal request sides, based on the RRNPC metric in (4.2), each change in $s(c_2)$ during the sliding can be determined in $O(1)$ time. Therefore, we can simply calculate the satisfaction of an initial candidate frame (e.g., the frame with $y_2 + 3z_2/2 = \overline{y}_j$) and update $s(c_2)$ by sliding c_2 upward along the line $x = \underline{x}_j$ while maintaining $r_j \subseteq c_2$. We summarize the subroutine in Algorithm 4.

The subroutine presented in Algorithm 4 runs in $O(n)$. This means when the cameras have a fixed resolution, $T_1 = O(n)$ and Algorithm 3 runs in $O(n^2)$ time.

Algorithm 4: Subroutine solving (4.12) with a fixed resolution

Input: Right separation boundary at $x = \underline{x}_j$;
Output: c_2^j;
begin
  Create candidate frame c_2; $O(1)$
  Set $x_2 - 4z_2/2 = \underline{x}_j$, $y_2 + 3z_2/2 = \overline{y}_j$; $O(1)$
  Calculate $s(c_2)$; $O(n)$
  while $y_2 - 3z_2/2 < \underline{y}_j$ $O(n)$
  do
    Slide c_2 upward along line $x = \underline{x}_j$ until one of its horizontal sides aligns with that of a request; $O(1)$
    Update $s(c_2)$; $O(1)$
  end
  return the best c_2; $O(1)$
end

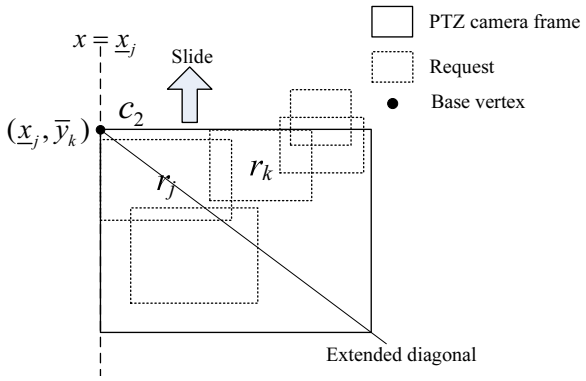


Fig. 4.4. An illustration of finding c_2^j as in (4.12) with fixed resolution. Slide the candidate frame c_2 along line x=$\underline{x}_j$ from an initial position. Whenever a horizontal frame side aligns with that of a request, the change in $s(c_2)$ can be computed in $O(1)$ time.

Discrete camera resolutions

Now we consider the cameras have m discrete resolution levels. In this case, for each right separation boundary, we just run the subroutine in Algorithm 4 m times, each time for one resolution level, respectively. Therefore, when the cameras have m discrete resolution levels, Algorithm 3 runs in $O(n^2m)$ time.

Continuous camera resolutions

Finally, we consider the cameras have continuous resolution range $[\underline{z}, \overline{z}]$. We already know the left side of c_2^j satisfies $x_2^j - 4z_2^j/2 = \underline{x}_j$. As shown in Fig. 4.4, the extended line of a horizontal request side $y = \overline{y}_k$ intersects with line $x = \underline{x}_j$ at vertex $(\underline{x}_j, \overline{y}_k)$. $(\underline{x}_j, \overline{y}_k)$ is defined as Base Vertex (BV) in [47]. According to the optimality condition in Lemma 2 of [47], one optimal frame c_2^j must have one corner coincident with a BV. Song *et al.* [47] propose a Base Vertex Incremental Computing with Diagonal Sweeping (BV-IC-DS) algorithm to find an optimal frame. The basic idea is to expand the candidate

frame along its extended diagonal by increasing the resolution. The satisfaction of the frame changes only at $O(n)$ number of critical resolution values and the changes between consecutive critical values can be determined in constant time. We apply a modified BV-IC-DS here. We skip the details and readers can refer to [47] for details.

BV-IC-DS runs in $O(n)$ for each BV and we have $O(n)$ BVs for each separation boundary. This means when cameras have continuous resolution levels, $T_1(n) = O(n^2)$ and Algorithm 3 runs in $O(n^3)$ time.

Theorem 2 *When cameras have a fixed, m discrete and continuous zoom level(s), Algorithm 3 runs in $O(n^2), O(n^2m)$ and $O(n^3)$ times, respectively.*

Table 4.2 summarizes the complexities for all algorithm variations.

Table 4.2
Summary of algorithm complexity

Zoom	Fixed	m discrete	Continuous
Exhaustive search	$O(n^3)$	$O(n^3m)$	$O(n^4)$
Sweeping search	$O(n^2)$	$O(n^2m)$	$O(n^3)$

It is worth mentioning that though we focus on rectangular requests here, our algorithm can also apply to problems with any polygonal requests. Based on the RRNC metric, a frame fully contains a polygonal request region if and only if the frame encloses its iso-oriented minimal bounding rectangle (MBR). We can reduce the problem with polygonal requests to the one with rectangular requests by replacing the polygonal request regions with their MBRs.

4.5 Experiments

We have implemented all the algorithms using Microsoft Visual C++ 2005. We test the algorithms on a desktop PC with a 3.2GHz Pentium(R) D CPU, 2 GB RAM, and a hard disk of 320 GB. We test the speed of the algorithms with different settings of n.

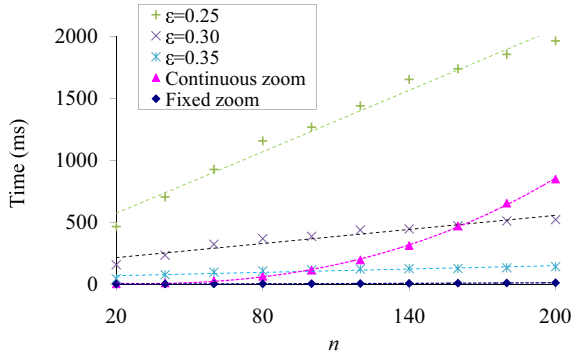


Fig. 4.5. Computation speed of algorithms with a fixed and continuous zoom level(s), respectively, and the comparison with the approximation algorithm in [69] with approximation bound $\epsilon = 0.35$, 0.30 and 0.25, respectively.

We use random input for testing. First, s_d 2-D points are uniformly generated across $[0, w] \times [0, h]$. Each point indicates a location of interest and is designated as "seed". Each seed is associated with a random radius of interest. To generate a request, we first randomly assign it to a seed. Then within the radius of the seed, a 2-D point is randomly generated as the center of the rectangular request region and two random numbers are generated as the width and height of the request. Finally, the resolution value of the request is randomly generated across the resolution range $[\underline{z}, \overline{z}]$.

Across the experiments, we set $w = 80$, $h = 60$, $\underline{z} = 5$, $\overline{z} = 15$ and $s_d = 5$. We set the fixed camera resolution as $z^0 = 8$. For each setting of n, 100 trials are carried out for averaged performance. Fig. 4.5 illustrates the relationship between computation time and n for proposed algorithm with a fixed and continuous zoom level(s), respectively. It is shown that the proposed algorithm with fixed zoom is very fast. It takes only 10 ms with $n = 200$, which is usually very large for most surveillance systems. Though the computation time of the algorithm with continuous zoom increases much faster as n increases, it takes only less than 900 ms with $n = 200$. Both curves are consistent with our complexity analysis.

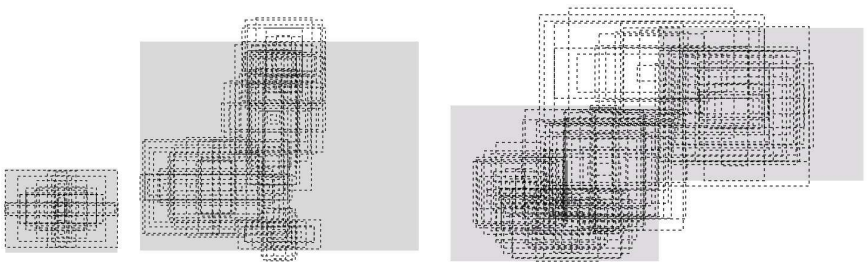
We also compare the proposed algorithm with the approximation algorithm in [69], which run in $O(n/\epsilon^3 + p^2/\epsilon^6)$ time, where ϵ is the approximation bound. We test the approximation algorithm with ϵ = 0.35, 0.30 and 0.25, respectively. It is shown that the approximation algorithm's speed performance deteriorates very quickly as ϵ increases. With $n \leq 200$, the approximation algorithm takes almost 2 seconds even if the approximation bound is considerably large as $\epsilon = 0.25$. When ϵ becomes even worse as 0.30 and 0.35, the approximation algorithm will eventually outspeed the proposed algorithm at n =160 and 100, respectively. It is also worth mentioning that the computation time of the approximation algorithm is proportional to the size of the problem space $[0, w] \times [0, h]$ while the speed of the proposed algorithm is independent of w and h.

These tell us that for applications where n is not very large but the problem space $[0, w] \times [0, h]$ is large, and the accuracy of the solution is a significant concern, the proposed algorithm outperforms the approximation algorithms in both speed and solution quality. If n is very large but the problem space $[0, w] \times [0, h]$ is small, and rough solution (e.g., $\epsilon \geq 0.25$) is acceptable, then the approximation algorithm is a faster alternative. In fact, most visual object detection/tracking/surveillaince systems [21, 28] can handle much less than 100 objects at the same time while accurate object tracking/observation is required, which qualify the proposed algorithm as a viable solution for these applications.

Fig. 4.6 shows two sample outputs of the algorithm with continuous zoom levels and $n = 100$. It is shown that in both cases, our algorithm reasonably locates 2 frames to cover most of the requests.

4.6 Conclusions

In this section, we formulate the non-overlapping 2-frame problem with non-partial coverage as an optimization problem. We propose a series of algorithms for solving the problem under different camera resolution configurations. For cameras with fixed, m discrete and continuous resolution level(s), we propose algorithms to solve the 2-frame



(a) (b)

Fig. 4.6. Sample simulation results for random input. Dashed-line rectangles denote requests and grey rectangles are optimal frames. $n = 100,\ s_d = 5$.

problem in $O(n^2)$, $O(n^2m)$ and $O(n^3)$ time, respectively. We have implemented all the algorithms and experimental results are consistent with our complexity analysis.

In future work, we will explore new algorithms for solving p-frame problems with $p \geq 3$. It is shown in Fig. 4.6(b) that some left area of the left frame and some right area of the right frame are wasted. It is due to the non-overlapping condition. We will relax the assumptions to allow camera frames to overlap in the future. We plan to apply the proposed algorithms to collaborative outdoor observation and surveillance in filed experiments.

In the last three sections, we have studied an example of the MOMR++ system: the autonomous crowd surveillance system and its corresponding frame selection algorithms. In the next section, we introduce a different MOMR++ system: the bird species detection system.

5. MOMR++ SYSTEM: RARE BIRD DETECTION SYSTEM

5.1 Introduction

In this section, we report another example of the MOMR++ system: the rare bird detection system, which is initially motivated for assisting the search for the though-to-be-extinct Ivory-Billed Woodpecker.

(a) (b)

Fig. 5.1. Our autonomous observatory system installed along Bayou DeView, a bottomland forest near Brinkley, Arkansas. (a) The installation site. (b) A high resolution video frame of a red-tailed hawk captured by the system on Dec. 13, 2006. The red-tailed hawk has a body length of 55 cm, close in length to the IBWO.

The Ivory-Billed Woodpecker (IBWO) is a magnificent creature that is of great interest to birdwatchers, ornithologists, and conservationists. The last confirmed U.S. sighting was in the early 1940s but a photo was taken in Cuba in 1948. In Feb. 2004, a credible eyewitness sighting was reported along Bayou DeView in eastern Arkansas, prompting a comprehensive and systematic search led by researchers at Cornell University and the Nature Conservancy. In Fall 2005, we joined the search effort by developing a high resolution robotic video system to observe the sky over an extended time period. Detailed high resolution video images are required to distinguish an IBWO from its cousin, the common Pileated Woodpecker.

Our goal is to develop a robust autonomous system that detects when birds fly into the field of view, keeping only the associated video segments for further species recognition. As illustrated in Fig. 5.1, the system has been installed in a clearing along Bayou DeView. This project is part of our larger effort to develop autonomous and networked systems for collaborative observation of natural environments [49].

We began with the following four design goals:

a) Sensitivity: the ability to detect and record video sequences of sufficiently high resolution to clearly distinguish between the IBWO, the Pileated Woodpecker, and other species with a low false negative rate ($< 20\%$),

b) Data reduction: the system records 198GB of high resolution video data per day. Due to greatly reduced networking bandwidth in the wilderness, we want to discard at least 99% of this while maintaining criterion a),

c) Accuracy: the system should maintain a low false negative rate, which means the system should not miss an IBWO flying by the camera. However, it is acceptable if the system has a relatively high false positive rate as long as criterion (b) is satisfied, and

d) Robustness: the ability to operate autonomously in harsh conditions over long periods (i.e. mean time between maintenance > 6 months.)

In this section, we report our system and preliminary algorithm development progress including hardware design, software architecture, and a bird filter that combines size filtering, nonparametric motion filtering, and temporal difference filtering. Our system has been deployed in two locations: Texas A&M campus from May - Aug 2006 and Bayou DeView, a swampy bottomland forest near Brinkley, Arkansas from Oct. 2006 to Oct. 2007. Initial results suggest that the system we describe has met these design criteria. Fig. 5.1 shows the system as deployed in Arkansas and a captured high resolution image of a red-tailed hawk.

5.2 Related Work

The IBWO is the third-largest woodpecker in the world. It has a distinctive ivory colored bill, white feathers under a black wing, and male birds have a red crest. A pair of birds may need 25km^2 or more of forest to feed. The loss of habitats due to the increasing human population and logging activities has greatly impacted the IBWO population in the past century. The last confirmed U.S. photos of IBWOs were taken by James Tanner in Louisiana in 1938. John Dennis took the last photos of this species in Cuba in April 1948.

Despite lack of conclusive evidence, the search for the legendary bird has never ceased. In 2005, the Cornell Laboratory of Ornithology and their colleagues reported the discovery of an IBWO in the Big Woods area of Arkansas [71] based primarily on a low-resolution video segment [72], so there is great interest in a high-resolution autonomous system.

Remote nature camera systems have been around since 1950s. Gysel and Davis [73] built an early video camera based on remote wildlife observation system to study rodents. Biologists use remote photography systems to observe nest predation, feeding behavior, species presence, and population parameters [74–79]. Commercial remote camera systems such as Trialmaster [74] and DeerCam have been developed since 1986 and have been widely used in wildlife observation. The Internet enables webcam systems that allow the general public to access remote nature cameras. Thousands of webcams have been installed around the world, for example, to observe elephants[1], tigers[2], bugs[3] and so on. However, most of cameras perform simple time sampled recordings, and it is difficult or impossible for human experts to reliably review the tens of thousands of images recorded.

Song and Goldberg have developed systems and algorithms for networked cameras for a variety of applications such as construction monitoring [80], distance learning [9], and panorama construction [81].

Motion detection segments the moving objects from a video sequence. Existing motion detection techniques can be classified into three categories: background subtrac-

[1] http://www.zulucam.org/
[2] http://www.tigerhomes.org/animal/web-cams.cfm
[3] http://bugscope.beckman.uiuc.edu/

tion [18, 82], temporal differencing [83], and optical flow [84, 85]. Background subtraction calculates the pixel-wise intensity difference between an input frame with a background reference model. To address the background noise, researchers propose many statistics-based background models such as temporal average [13], median absolute deviation (MAD) [14], adaptive Gaussian estimation [86], mixed Gaussian model, parameter estimation [87], non-parameter estimation [18], and Kalman filter compensation [19]. Temporal differencing calculates the pixel-wise intensity difference between two or three consecutive frames. Optical flow calculates the displacement flow vectors from a video sequence. A nature environment is noisy and unstructured. No single methodology can directly satisfy the four criteria in the IBWO search. During our system and software development, we carefully fine-tune the parameters to combine the strenghts of nonparametric estimation, temporal differencing, and connectivity checking.

5.3 Hardware

Our system design was based on input from the Cornell ornithologists and the conditions of the installation site. As illustrated in Figs. 5.1 and 5.2, the system is installed in a clearing in the swampy forest that is flooded by Bayou DeView in Arkansas. The clearing is a narrow corridor that is about 900 meters long and 50 meters wide. It was formed when the forest was cut to allow a high voltage line to run through it. The system is installed on an electric pole in this power line cut. A bird flying across the power line cut is clearly exposed to the sky, which makes this an ideal location for installing the system. The site was carefully selected by the Cornell ornithologists.

To provide good coverage of the region, we chose a two-camera system design with each camera facing upward in opposite directions along the corridor. We chose a camera lens with a 20° horizontal field of view and a 15° vertical field of view. Knowing that the bird often flies at tree-top height, which is about 10 meters above the tree, we setup the camera orientation to maximize coverage as illustrated.

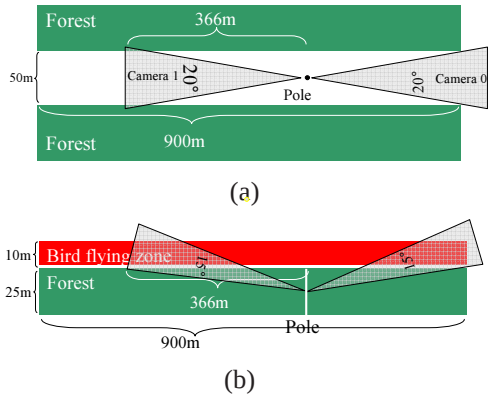
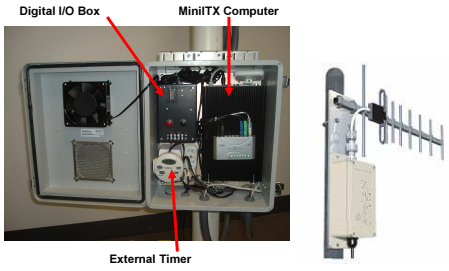


Fig. 5.2. Schematic of the system installation site and camera coverage. The camera has a 20° horizontal field of view and a 15° vertical field of view. (a) top view of system coverage; (b) side view of system coverage.

The Cornell ornithologists advised us that to serve as conclusive evidence, a bird image should be at least 25×25 pixels. We chose Arecont Vision 3100 3Mega-pixel high resolution networked video cameras as the imaging device. As illustrated in Fig. 5.3, other major components of the system include a MiniITX computer with 1.4 GHz CPU and 1GB RAM, a LinkSys wireless access point, an AW900 long range wireless adaptor with a 900Mhz directional Yagi antenna, an external timer, an external USB hard disk, and a digital I/O box with a set of relays and an LED array. To deal with the harsh swampy environment, the whole system is protected by weatherproof and thermal-controlled enclosures.

There are two separate networks in the system. The internal network is managed by the LinkSys access point that is both a wireless router and a four-port wired switch that allows the MiniITX computer to talk to the two cameras via the T3 local ethernet. The local 2.4Ghz wireless service is used to facilitate *in-situ* system debugging. The external network bridges the computer to the Internet by the AW900 long range wireless adaptor. Running at 1.5Mbps and 900Mhz carrier frequency, the AW900 long range wireless adaptor can reach a maximum distance of 40 miles if equipped with a 15dBi Yagi directional antenna. Since there is no interesting activity at night, the external timer powers off the



(a) (b)

Fig. 5.3. System hardware configuration: (a) the MiniITX computer, the external timer, and the digital I/O box are protected in a weatherproof box; (b) the AW 900 long range outdoor wireless adaptor and a 15dBi AW15 Yagi Antenna.

system each night. The external timer provides additional recoverability when the computer accidently crashes. Image data is stored in an external USB hard disk. Prior to the installation of the long range wireless network, one of us swapped the external hard disk every few weeks.

The customized digital I/O box has an LED array that displays the percentage of storage space left in the USB hard disk. The digital I/O box also controls a set of digital relays which can selectively power on or off individual cameras. This proves to be an important design choice because the camera firmware can crash and needs to be power-cycled from time to time. The digital I/O box is also equipped with a red push button that can power off the MiniITX, which has no keyboard or monitor. The simplified hardware interface makes it easy for non-experts to operate and maintain.

5.4 Software

To facilitate image acquisition, the MiniITX computer has a customized Microsoft Windows XP operating system. Due to the speed requirement, Microsoft Visual C++ has been chosen as the programming language in the development. As illustrated in Fig. 5.4, system software contains four main components: Bird Filter (BF), System Configuration

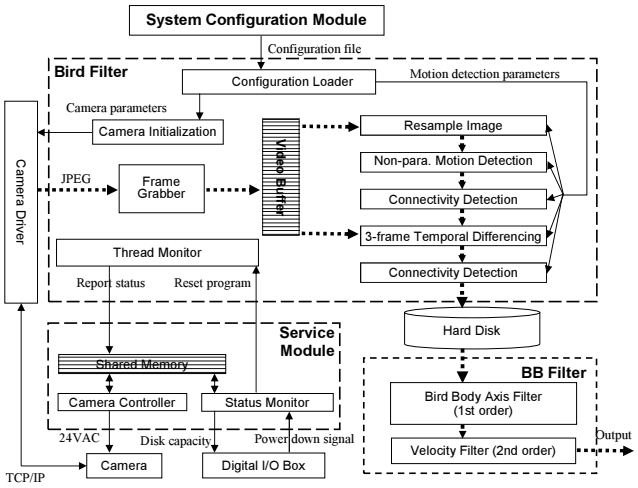


Fig. 5.4. System software diagram

Module (SCM), Service Module (SM), and Background Biometric (BB) filter. We will detail BF in the next section. The SCM is a configuration routine that allows us to adjust system parameters such as camera parameters, motion detection parameters, and on/off time on the field. The SM is a background process that monitors the whole system to detect if there is a software or hardware failure. The BB filter is still under development, it will be run offline to detect bird species automatically based on the biological information provided by the ornithologists.

5.5 Bird Filter

Based on what is known about the IBWO, the Bird Filter (BF) utilizes the information about the IBWO provided by the Cornell ornithologists:

Assumption 1

1. An adult IBWO has a body length of 48cm.
2. An IBWO can fly at $30 \sim 60$km/hr.

3. It takes a minimum size of 25×25 pixels to clearly distinguish the IBWO from the common pileated woodpecker.

5.5.1 Input and Output

The BF is a multi-threaded process that performs filtering on the acquired image in real time. The process decides whether to keep the video on the hard disk or to delete it. The filter makes the decision by filtering out images without motion and images with noisy motions. The noisy motions include the motions caused by vibrations of tree branches, moving clouds, sun positions, water reflections, dropping tree leaves, flying insects, and any moving objects smaller than 25×25 pixels in the image. As illustrated in Fig. 5.4, the BF acquires frames using the frame grabber thread. The frames are stored in a video buffer. Therefore, the input to the BF are image frames captured by the cameras and the output of the BF are image frames that contain fast-moving objects that are larger than 25×25 pixels.

5.5.2 Parameters

When the BF starts, it loads the configuration parameters such as camera parameters, regions of interest, and object size to initialize each relevant module. Camera parameters refer to camera auto iris gain that enables the camera to adapt itself to different lighting conditions in the outdoor environment. The image resolution is set to 1600×1200 pixels to ensure a good balance between frame rate and resolution. At this resolution, the Arecont vision camera runs at 11 frames per second (fps). Two cameras provide a total of 22fps to the system. To ensure the imaging of a fast-moving object, the camera exposure time is set to be less than 1/100 of a second. The regions of interest refer to where we perform bird detection on the image. It is stored as a binary map that can be defined at the installation site to facilitate the quick installation of the system.

5.5.3 Spatiotemporal Downsampling

Since the two cameras combined provide 22 fps at a resolution of 1600×1200 pixels each, it is impractical and unnecessary to analyze every image in real time. Therefore, we downsample video frame sequence spatiotemporally. We partition the continuous video sequence into sequential 7-frame video sequences. Define F to be a frame, the ith video sequence defined is,

$$F_i = \{F_{i1}, F_{i2}, ..., F_{ij}, ..., F_{i7}\}. \tag{5.1}$$

For each segment, we process its 4th frame F_{i4}, $i = 1, ..., \infty$, at a resolution of 400×300 with motion detection. In the downsampled image, we are interested in capturing motion objects that are bigger than 6×6 pixels, which is equivalent to the 25×25 pixels in the original size. There is a possibility that a bird might be missed due to the temporal downsampling. It takes a bird about 1 second to fly cross the power line cut, which should be sufficient time for the camera to capture 11 frames. However, there is a small chance that a bird might not appear on the 4th frame of the video sequence, and we could miss the bird completely. However, this is the natural limit imposed by the computation power and camera field of view. The downsampling operation can reduce noisy motions and increases computation speed.

5.5.4 Nonparametric Motion Filtering

To eliminate periodical noisy motions caused by vibrating tree branches and their shadows, we adopt the nonparametric background subtraction algorithm proposed by Elgammal et.al [18].

For every pixel at time t, Elgammal's algorithm updates a Gaussian model $N(\mathbf{0}, \Sigma)$ from its intensity values from the corresponding pixels in previous frames F_{i4}, $i = 1, ..., t$, where $\Sigma = \text{diag}\{\sigma_r^2, \sigma_b^2, \sigma_g^2\}$ is the variance-covariance matrix for three color channels. The Gaussian distribution updates itself as a new sample comes in. Therefore, for a periodic noise, the Gaussian model can characterize the periodic intensity change in its

variance if the algorithm has enough samples. The algorithm then predicts if a pixel is a foreground pixel based on probability thresholding. After extensive tests, we set the thresholding point to be the 98^{th} percentile.

This method has been proven to be robust in dealing with periodic noise. In our field test conducted on the Texas A&M campus, this method successfully filtered out the noisy background motions introduced by a rotating radio antenna. The output of nonparametric motions filtering is a binary map with white pixels as motion pixels, which is defined as B_{i4} for frame F_{i4}.

5.5.5 Connectivity Check

Unfortunately, the nonparametric filter cannot effectively filter out non-periodical noises such as moving clouds or dropping leaves. Further filtering is needed. We first perform a connectivity check to determine the size of the region that triggers the motion. Recall the required size in Assumption 1, we only keep the images with big moving objects. Recall that B_{i4} is a downsampled image. A size of 6×6 pixels is equivalent to the 25×25 pixels in the original image. If a B_{i4} contains a moving object that is bigger than 6×6 pixels, we proceed to the next step. Otherwise, we discard the entire segment F_i.

5.5.6 Temporal Differencing

Since a moving cloud can take on any shape or size, the downsampling and the non-parametric motion tracking cannot get rid of the false alarms triggered by moving clouds. On a cloudy day, the system might accumulate huge amounts of video data containing only moving clouds.

Observing the data, we notice that the velocity of a moving cloud is still relatively slow if compared with that of a flying bird. In adjacent frames, the displacement of a moving cloud is negligible if compared with the displacement of a flying bird. Therefore, for each motion frame F_{i4} detected by the nonparametric motion detector, we combine the motion

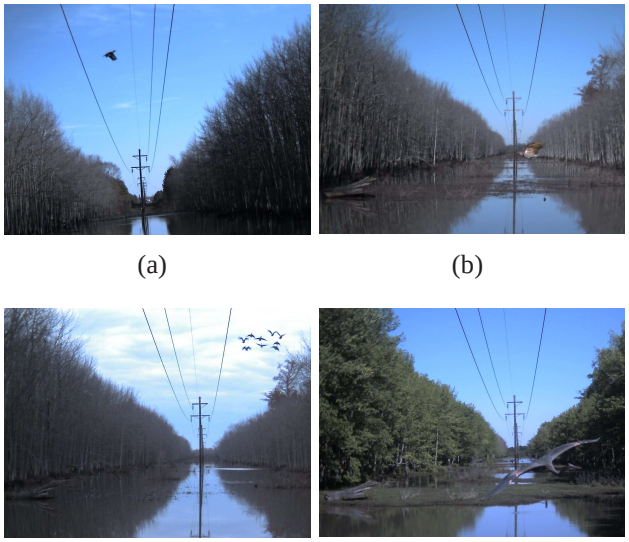
frame with two immediate adjacent frames F_{i2} and F_{i3} to judge the velocity difference. We know that motion on frame F_{i4} is detected using the previous frames $F_{i-1,4}$, $F_{i-2,4}$, $F_{i-3,4}$, ... $F_{1,4}$. For a slow moving object such as a cloud, although there exists an intensity difference $|F_{i4} - F_{i-1,4}|$ for the motion to be detected, the intensity difference between adjacent frames $|F_{i4} - F_{i3}|$ and $|F_{i5} - F_{i4}|$ should be much smaller than those of a fast moving object. Therefore the sum of $|F_{i4} - F_{i3}|$ and $|F_{i5} - F_{i4}|$ is a good thresholding function to judge if the moving speed of the object is fast enough. In our experiment, the threshold point is 30. We name it 3-frame temporal differencing. It is capable of filtering out objects that are significantly slower than the IBWO.

5.6 Experiments and Results

Two field tests have been conducted for the autonomous observation system. The system had been installed on the Texas A&M campus from May 2006 to October 2006 for the initial test. After 5 month-testing and tuning, the system was installed in Brinkley, AR to assist in the search for the IBWO from October 2006 to October 2007.

5.6.1 Sensitivity

Fig. 5.5 illustrates four species of birds imaged by our system in Arkansas. Among the samples, Fig. 5.5(a) is the closest cousin of the IBWO. Although the image is blurred, Cornell and U. Arkansas at Little Rock ornithologists were able to verify that it is a Pileated Woodpecker. A Pileated Woodpecker has a body length of 40 cm, which is just slightly smaller than that of the IBWO. The Northern Flicker in Fig. 5.5(b) is a smaller kind of woodpecker that has a size of 28-31 cm and a wingspan of 42-51 cm. Fig. 5.5(c) shows a flock of Canada Geese caught by the system. Fig. 5.5(d) is a Great Blue Heron with a wingspan of close to 2 m. Birds caught by the system can be either bigger or smaller than the IBWO and fly either faster or slower than the IBWO. This suggests that our system is capable of capturing conclusive images of an IBWO.



(a) (b)

(c) (d)

Fig. 5.5. Sample birds imaged by the system. (a) A Pileated Woodpecker (02/16/2007). (b) A Northern Flicker Woodpecker (02/27/2007). (c) A flock of Canada Geese (10/28/2006). (d) A Great Blue Heron (04/28/2007).

5.6.2 Data Reduction

As of September 4, 2007, the system has collected over 25 GB of images. A total of 113,836 images have been captured by the BF. Considering that there were a total of 245,520,000 images captured by the two cameras during the 310 days, the BF reduced the data by 99.9953%.

5.6.3 Accuracy

We consider both false negative and false positive rates. A false negative means that the system fails to detect when a bird flies by. Again, we tested the system using the data from both the Texas A&M campus and Brinkley, AR.

To test the false negative rate, we turn on the recording mode of the camera and sample every frame. Then we manually count the number of images containing a flying bird that

is bigger than 25×25 pixels. Comparing those with the algorithm output, we then get the false negative rate. A total of 80,000 image frames were collected over a 2-hour period on campus. There were three birds flying across the camera field of view in this 2-hour period and all have been detected by the BF. As mentioned earlier, the only reason a bird is missed by the system is the fact that it does not appear in F_{i4}, which is possible if the bird's flying trajectory is very close to the boundary of the camera field of view. The false negative test is actually the test of how many birds do not fly close to the center of the camera field of view. In the test data set, none of the birds fly close to the boundary of camera field view. We believe it could be less than perfect in the long run. Since the boundary of camera field of view is much smaller in comparison to overall field of view, the false negative rate should be a small value $(< 20\%)$. We are testing the false negative rate using the data from AR and will report the result in Section 6.

The false positive rate indicates the percentage of the images stored that are not triggered by bird motions. Since we perform motion detection computations on only the 4th frame of every 7-frame video segment, we collect the statistics only on the frame in which motion detection is performed. For the 1205 captured motion image files from the Texas A&M campus over a 6-day test period, the false positive rate is 32.9%. The false positive rate is 96% for the nine months of data collected in AR. The high false positive rate in AR is expected because we are more conservative in parameter settings. For example, our probability threshold in nonparametric motion filter is 99.9% for the experiment on the Texas A&M campus and is 98% for the experiment in AR. We purposefully lower the probability threshold to increase sensitivity. Also there are large numbers of insects in the forest that can trigger false alarms when they fly close to the lens. As long as the size of the files is not too big to be transferred, this false positive rate is acceptable.

5.6.4 Robustness

After one year in the Arkansas wilderness, the system has run continuously except for occasional power outages. The system has survived very large temperature variations

from winter to summer, severe weather changes, and has worked under high humidity conditions.

5.7 Conclusion and Future Work

This section reports our system and preliminary algorithm development for an autonomous observatory to assist the search for the IBWO. Data collected thus far suggests that the system achieves four design criteria: sensitivity, data reduction, accuracy, and robustness.

In Section 6, we will improve the filter efficiency by developing a more powerful filter that combine bird specific biological information such as size and velocity, and flying pattern.

6. MOMR++ ALGORITHM: PROBABLE OBSERVATION DATA SET-BASED EXTENDED KALMAN FILTER FOR BIRD SPECIES DETECTION

6.1 Introduction

In Section 5, we introduce a system to assist ornithologists to search for rare birds. The bird filtering algorithm in Section 5 is basically based on motion detection filtering. In this section, we further introduce a more powerful filtering algorithm that verifies the targeted bird biological information such as body size and velocity to further reduce the data volume for human without compromising the low false negative.

For the the rare bird searching task, three critical conditions must be met. First, a rare bird only appears in front of the fixed camera with very low occurrence (e.g., less than ten times per year) for very short durations (e.g., less than a fraction of a second), our algorithm must have a very low false negative (FN) rate. Second, since the final verification has to be performed by human experts, it is necessary to reduce the huge data volume to a manageable size, which also means that the filter can tolerate a less ideal false positive (FP) rate. Third, the system must be easy to be set up in the forest. Due to power and communication constraints, a single camera is preferred because it does not require the precise calibration and synchronization as dislocated stereo rigs would for distant fast-flying birds.

Fig. 6.1 shows the input of the problem is a short segmented motion sequence of an object. The output of the problem is to determine whether the motion sequence is caused by a targeted bird species. We verify the bird body axis information with the known bird flying dynamics. Since a regular extended Kalman filter (EKF) cannot converge due to the high measurement error and the limited observation data due to the high flying speed of the bird (e.g., the sample bird sequence in Fig. 6.1 only contains seven data points), we develop a probable observation data set (PODS)-based EKF and an approximate computation scheme. The new PODS-EKF searches the measurement error range for all probable observation data that ensure the convergence of the corresponding EKF outputs. The filtering

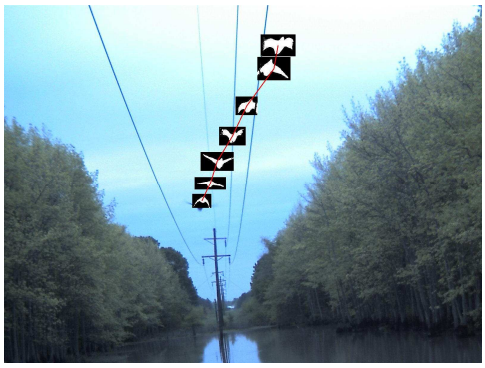


Fig. 6.1. An example of a video sequence of a flying bird that is captured in Bayou DeView in eastern Arkansas. The camera runs at 11 frames per second and the sequence is generated by superimposing the segmented bird images from consecutive video frames on the top of a background frame.

is based on whether the subset of PODS that guarantees EKF convergence is non-empty and the corresponding speed is within the known bird flying velocity profile. We show that the PODS-EKF filter theoretically ensures a zero FN rate.

We have evaluated the filtering algorithm using both the simulated data and field test data. Our algorithm has been applied for the search of IBWOs in eastern Arkansas. The physical experiment results show that the algorithm can reduce the video data for identification by over 99.9995% with close to zero FN. The rest of the section is organized as follows. Section 6.2 reviews the related works. The definition of the bird filtering problem is presented in Section 6.3. Sections 6.4 and 6.5 model the bird filtering problem and propose the PODS-EKF method followed by an algorithm in Section 6.6. The experimental results are presented in Section 6.7 before we conclude in Section 6.8.

6.2 Related Work

Detection of a flying bird relates to vision-based motion detection, image processing for animal detection and recognition, 3D structure inference with monocular vision, visual tracking, and especially Kalman filter-based visual tracking.

Recent development in vision-based motion detection has greatly advanced its robustness in noisy environments. Motion detection segments moving objects from their background using a video sequence. To address the background noises, researchers propose many statistics-based background models such as temporal average [13], median absolute deviation (MAD) [14], adaptive Gaussian estimation [86], mixed Gaussian models, parameter estimation [87], nonparametric estimation [18], and Kalman filter compensation [19]. Our work builds on the robust nonparametric background subtraction algorithm proposed in [18] to segment the moving foreground objects. Moreover, our algorithm advances the mere motion-detection to bird species detection by using bird flying dynamics.

Periodic motion detection [88, 89] assumes objects with periodic motion patterns and applies time-frequency analysis [88, 90] or image sequence alignment [91] to capture the periodicity. Applications of periodic motion detection have been found to vehicles, humans and even canines. However, recognizing birds is different because a bird flying pattern combines both gliding and wing-flapping and the periodic motion assumption does not apply.

Animal detection and recognition using video images has been an active research direction. Most of the existing approaches build appearance models of animals by feature points [92], silhouettes [93], contours [94], 2D kinematic chains of rectangular segments [95], and motion symmetry [96]. A known set of animal images are used to train and test the model using learning techniques such as neural networks [97], K-means [98], clustering [95], template matching [93] etc. A review of the image processing techniques for bird recognition can be found in [97]. However, these techniques require a large learning data set to train the model, which is not available in our applications.

Recently, the 3D structure inference using monocular vision has drawn increasing research attention. Ribnick *et al.* [99] propose an algorithm for estimating 3D parabolic trajectories of projectiles in monocular views. Saxena *et al.* [100] propose a learning algorithm that estimates 3D structures of a static scene based on a single still image. The work models the scene with sets of planes using Markov Random Field (MRF) and trains the

model based on depth cues such as texture variations and gradients, color, haze, and defocus etc. Hoiem *et al.* [101] propose a similar approach that models the static scene with "surface layout." Different from these works, our approach deals with a highly dynamic object (i.e., the bird) and its trajectory is not necessarily parabolic.

Visual tracking estimates trajectories of objects in 2D image space. State estimators such as Bayesian filters [102], particle filters [103, 104], sparse (extended) information filters [105] or (extended) Kalman filters [106] are often employed. When observation uncertainty presents, data association techniques such as multiple hypotheses based tracking [107] are usually used. A recent survey can be found in [28]. One key novelty of this work is that the existing works focus on the data association and state estimation problem from a large observation data set while our work focus on the state range estimation using minimal or even insufficient observation data set with relatively large noises.

The fundamental technique used in the bird detection is the extended Kalman filer. Kalman filter, extended Kalman filter, and their variations verify the detected motion information from video frames with the prior known dynamics. Since the methods utilize the information across consecutive video frames, their robustness to errors makes them ideal methods for poor illumination conditions and outdoor environments [108]. Hence, Kalman-filters have seen a wide range of applications such as simultaneous localization and mapping in robotics [109] and object recognition and tracking of vehicles [110], pedestrians [111], and even human eyes [112]. Most existing works assume rigid objects and ignore the convergence of Kalman filter because an ample amount of observation data are available. Unfortunately, these conditions do not hold for a high-speed flying bird.

Our group has developed systems and algorithms [4, 47, 48] for networked robotic cameras for a variety of applications such as construction monitoring [80], distance learning [9], panorama construction [81], and nature observation [49]. Our previous work [113] details how to build an autonomous nature observation system using motion detection. We learn that mere motion detection cannot save the biologists from the herculean task of image sorting, which inspires this work.

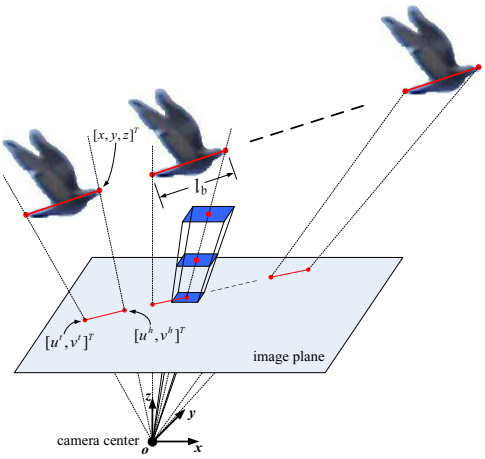


Fig. 6.2. An illustration of bird detection. When a bird flies across the camera FOV, the corresponding motion sequence can be used to extract a set of moving line segments that correspond to the body axis of the bird. The line segments are then verified using an EKF based on the known profile from the targeted species. The segmentation error of the end of body axis are uniformed distributed in the u-v image plane and can be represented as an inverse pyramid when the error range is back-projected from the camera center to the FOV volume.

6.3 Problem Description

Our system is a monocular vision system with a narrow field of view (FOV). The position of objects with respect to the camera is based on a 3D Cartesian camera coordinate system (CCS) with its origin at the camera center as shown in Fig. 6.2. The x-axis and y-axis of the CCS are parallel to the u-axis and the v-axis of the image coordinate system (ICS), respectively.

From the knowledge provided by ornithologists, we know that a flying bird is usually an adult bird. A bird does not change its size once reaching its adulthood. Birds of the same species share a similar size and flying speed range. This biological information allows us to distinguish the targeted species from other moving objects.

6.3.1 Assumptions

To establish the bird detection problem, we also have the following assumptions,

- A fixed and pre-calibrated camera is used. With a calibrated camera and without loss of generality, we can always transform camera intrinsic parameter matrix K_c to diag$(f, f, 1)$, where f is the focal length of the camera in units of pixel side length. ICS must have its origin located on the principal axis (z axis) of CCS. Hence we have perspective project matrix $P_c = [K_c|\mathbf{0}_{3\times1}]$.
- There is only one bird in the image sequence. If there are multiple flying birds in the scene, we assume each individual bird sequence has been isolated out using multiple object tracking techniques [28].
- The bird is flying along a straight line with a constant speed when captured by the camera. This assumption usually holds considering a fast flying bird can only stay in the FOV for less than a second.

6.3.2 Inputs and Output

The input of the problem is a sequence of n images which contain a moving object of any type. Each frame is time-stamped. Based on the information from ornithologists, we know the body length l_b and the flying speed range $\mathcal{V} = [v_{min}, v_{max}]$ of the targeted bird species. The output is to determine if the motion sequence is caused by the targeted bird species or not.

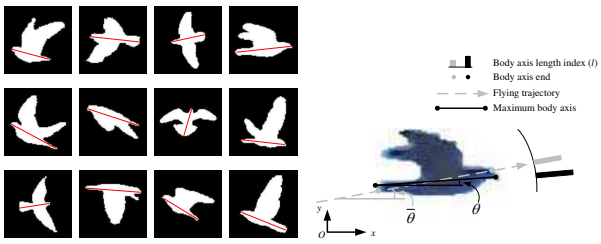
6.4 Modeling a Flying Bird

To develop a bird filter, the key is to extract the bird flying information from the segmented bird motion sequence and associate the information with the known flying models

and the prior information regarding the targeted species. Let us first observe the motion sequence of the flying bird to investigate how to extract the bird flying information.

6.4.1 Bird Body Axis Filter

As detailed in [113], we segment the moving object from its background and obtain a set of motion sequences. Fig. 6.3(a) illustrates different flying poses of a pigeon. At first glance, it is unclear how to utilize this information because bird poses are not a simple discrete set of states. The wing configurations of the bird vary dramatically from frame to frame. The shape of the bird changes significantly as well.



(a) (b)

Fig. 6.3. (a) Segmented bird flying poses. The white pixels in the binary map indicate the segmented salient motion zone. Bird body axes are overlaid on top of the segment image. (b) An illustration of the search for body axis length.

As we scrutinize the collected flying pose data, we find that a bird does not bend or extend its body during the flight as illustrated in Fig. 6.3(a). Hence, we have,

Observation 6.4.1 (Invariant Body Length) *A flying bird maintains a constant body length during flight.*

This observation has been confirmed by ornithologists and our image data collected (341 bird images from 61 motion sequences) from physical experiments. Except landing

and taking off, a bird extends its body straight to minimize the wind resistance during a normal flight. This is an important finding because it provides an entry point to attack the bird detection problem. The ornithologists also use the bird body length as an important index to classify birds because adult birds from the same species share the same body length with little variance. Hence the problem becomes how to extract the body axis orientation and length of a flying bird from the segmented motion sequence.

It is nontrivial to extract the bird body axis and length from the isolated poses in Fig. 6.3(a) because a bird is a non-rigid and deformable object. However, if we superimpose the segmented bird flying pose data to the background image as illustrated in Fig. 6.1, a new finding appears:

Observation 6.4.2 (Body Axis Orientation) *The orientation of the body axis of a flying bird is always close to the tangent line of its flying trajectory.*

To validate our observation, we analyze 61 bird motion sequences with a total of 341 segmented birds that we have collected in past years. The result confirms the observation. The mean orientation difference is 0.8° and the standard deviation is $\sigma_b = 8.3^\circ$. This observation inspires us to develop a bird body axis filter (BBAF) to extract bird body axes from the segmented motion zone.

Let us define the bird body line segment in the image frame as

$$\mathbf{z} = [u^h, v^h, u^t, v^t]^T, \tag{6.1}$$

where (u^h, v^h) is the head position and (u^t, v^t) is the tail position. From $\mathbf{z}$, we can compute the body axis orientation

$$\theta = \text{atan2}(v^h - v^t, u^h - u^t),$$

and the body axis length

$$l = \sqrt{(u^h - u^t)^2 + (v^h - v^t)^2}.$$

Note that l is different from l_b. l is the projection of l_b on the image plane and is in units of pixels.

We know that the slope of the tangent line of the trajectory can be extracted easily based on the position of the salient motion zone on the background image. The red line in Fig. 6.1 is the approximate trajectory generated by linking the geometric center of each motion zone. The tangent line of the approximate trajectory can serve as an initial solution for the bird body axis orientation. However, since the standard deviation $\sigma_b \neq 0$, further refinements are required.

Define B as the boundary pixel set of the motion zone (e.g., the boundary pixel set of the white pixels in each block in Fig. 6.3(a)). As illustrated in Fig. 6.3(b), any two points in B, as the body axis ends, form a candidate body axis with length l. Define $\bar{\theta}$ as the orientation of the corresponding tangent line of the flying trajectory. We find the bird body axis in image $\mathbf{z}$ by searching for its orientation in the range $[\bar{\theta} - 2\sigma_b, \bar{\theta} + 2\sigma_b]$ and the corresponding body axis ends in B to maximize l :

$$\mathbf{z} = \arg \max_{\substack{(u^h, v^h) \in B \\ (u^t, v^t) \in B}} l, \text{ subject to: } \theta \in [\bar{\theta} - 2\sigma_b, \bar{\theta} + 2\sigma_b]. \tag{6.2}$$

6.4.2 Bird Flying Dynamics

To determine whether the motion information is caused by the targeted species, we need to establish a bird flying model in the image frame. Let $\mathbf{p} = [x, y, z]^T$ denote the head position of the bird body axis and $\mathbf{v} = [\dot{x}, \dot{y}, \dot{z}]^T$ denote its velocity in the CCS. Since the bird flies along a straight line with a constant velocity, we have

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{v}} \end{bmatrix} = [\dot{x}, \dot{y}, \dot{z}, 0, 0, 0]^T = \begin{bmatrix} \mathbf{v} \\ \mathbf{0} \end{bmatrix}, \tag{6.3}$$

where the state variable $\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \end{bmatrix} \in \mathbb{R}^6$ describes the position and velocity of the bird head. Defining $\mathbf{x}_{tail} = [x^t, y^t, z^t]^T$ as the position of the bird tail, and we have

$$\mathbf{x}_{tail} = \begin{bmatrix} x - \dot{x} l_b / \|\mathbf{v}\| \\ y - \dot{y} l_b / \|\mathbf{v}\| \\ z - \dot{z} l_b / \|\mathbf{v}\| \end{bmatrix}. \tag{6.4}$$

As illustrated in Fig. 6.2, the relationship between the measurement data $\mathbf{z}$ defined in (6.1) and the corresponding state $\mathbf{x}$ can be described using the pin-hole camera model. Since $K_c = \text{diag}(f, f, 1)$, we have

$$\mathbf{z} = \begin{bmatrix} fx/z \\ fy/z \\ fx^t/z^t \\ fy^t/z^t \end{bmatrix} = \begin{bmatrix} fx/z \\ fy/z \\ f\frac{x\|\mathbf{v}\| - l_b\dot{x}}{z\|\mathbf{v}\| - l_b\dot{z}} \\ f\frac{y\|\mathbf{v}\| - l_b\dot{y}}{z\|\mathbf{v}\| - l_b\dot{z}} \end{bmatrix} + \mathbf{w} := h(\mathbf{x}) + \mathbf{w}, \tag{6.5}$$

where $h(\cdot)$ is usually called the measurement function and $\mathbf{w}$ represents the measurement noise.

6.5 Probable Observation Data Set-based EKF Method

6.5.1 Extended Kalman Filter

Eq. (6.2) provides the bird flying information extracted from images. Eq. (6.5) captures the prior known information regarding the targeted species. If the motion is caused by the targeted species, then the bird body axis information provided by (6.2) should follow the nonlinear dynamic system described by (6.5), which can be validated using an EKF.

Eqs. (6.3) and (6.5) can be re-written in a discrete-time form,

$$\mathbf{x}(k+1) = A(k+1)\mathbf{x}(k) + \mathbf{q}(k), \tag{6.6a}$$

$$\mathbf{z}(k) = h(\mathbf{x}(k)) + \mathbf{w}(k), \tag{6.6b}$$

where $\mathbf{q}(k) \in \mathbb{R}^6$ and $\mathbf{w}(k) \in \mathbb{R}^4$ represent the white Gaussian transition and measurement noises at time k with covariance matrix $Q(k) \in \mathbb{R}^{6\times6}$ and $W(k) \in \mathbb{R}^{4\times4}$, respectively,

$$\mathbf{q}(k) \sim \mathcal{N}(0, Q(k)), \;\; \mathbf{w}(k) \sim \mathcal{N}(0, W(k)),$$

and $A(k+1)$ is the state transition matrix at time $k+1$,

$$A(k+1) = \begin{bmatrix} \mathbf{I}_{3\times3} & \Delta T(k+1|k)\mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix},$$

where $\Delta T(k+1|k)$ is the time interval between time k and time $k+1$.

We define $P \in \mathbb{R}^{6\times6}$ as the covariance matrix for the state variable $\mathbf{x}$. The EKF for the system in (6.6) can be implemented as a state prediction step $\hat{\mathbf{x}}(k|k-1),\ \hat{P}(k|k-1)$ and measurement correction step $\hat{\mathbf{x}}(k|k),\ \hat{P}(k|k)$ recursively as follows,

$$\hat{\mathbf{x}}(k|k-1) = A(k)\hat{\mathbf{x}}(k-1|k-1), \tag{6.7a}$$

$$\hat{P}(k|k-1) = A(k)\hat{P}(k-1|k-1)A^T(k) + Q(k), \tag{6.7b}$$

$$K(k) = \frac{\hat{P}(k|k-1)H^T(k)}{H(k)\hat{P}(k|k-1)H^T(k) + W(k)}, \tag{6.7c}$$

$$\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1) + K(k)(\mathbf{z}(k) - h(\hat{\mathbf{x}}(k|k-1))), \tag{6.7d}$$

$$\hat{P}(k|k) = (\mathbf{I}_{6\times6} - K(k)H(k))\hat{P}(k|k-1), \tag{6.7e}$$

where $K(k) \in \mathbb{R}^{6\times4}$ is the "Kalman gain" at time k and $H(k) \in \mathbb{R}^{4\times6}$ is the Jacobian matrix of the function $h(\cdot)$ in (6.5) with respect to $\mathbf{x}$.

Recall that $\hat{\mathbf{x}}(k|k) = \begin{bmatrix} \hat{\mathbf{p}}(k|k) \\ \hat{\mathbf{v}}(k|k) \end{bmatrix}$. For the n-image motion sequence, the predicted $\hat{\mathbf{x}}(n|n)$ contains the bird velocity $\hat{\mathbf{v}}(n|n)$. The decision of accepting or rejecting the moving object as a member of the targeted species is defined as the following indicator function,

$$I(\mathbf{Z}^{1:n}) = \begin{cases} 1 \text{ (accept) if } \|\hat{\mathbf{v}}(n|n)\| \in \mathcal{V} \text{ and EKF converges}, \\ 0 \text{ (reject) otherwise}, \end{cases} \tag{6.8}$$

where $\mathbf{Z}^{1:n} = \{\mathbf{z}(1), \mathbf{z}(2), ..., \mathbf{z}(n)\}$ is the set of body axes across n frames. $\mathbf{Z}^{1:n}$ is also referred to as the observed data. Eq. (6.8) basically states that the moving object is a member of the targeted species if the EKF converges to the desired absolute velocity range $\mathcal{V}$.

6.5.2 Determining EKF Convergence

As indicated in (6.8), automatically determining whether the EKF converges or not is necessary. Define the estimated state set as $\mathbf{X}^{1:n} = \{\hat{\mathbf{x}}(1|1), \hat{\mathbf{x}}(2|2), ..., \hat{\mathbf{x}}(n|n)\}$. Since velocity convergence implies position convergence and $\hat{\mathbf{v}}(k|k)$ convergence means $\|\hat{\mathbf{v}}(k|k) - \hat{\mathbf{v}}(k-1|k-1)\| \to 0$, we determine the convergence of the EKF by inspecting

$$\varepsilon(\mathbf{X}^{1:n}) = \sum_{k=2}^{n} \omega(k) \|\hat{\mathbf{v}}(k|k) - \hat{\mathbf{v}}(k-1|k-1)\|,$$

where $\omega(k) > 0$ is the weighting factor at time k. $\omega(k)$ is a monotonically-increasing function of k, which gives more weight to later states. $\omega(k)$ is usually pre-generated using simulated random inputs across the entire possible parameter range without measurement error (i.e. $W(k) = \mathbf{0}_{4\times4}$). Setting $W(k) = \mathbf{0}_{4\times4}$ is to ensure EKF convergency, which will be explained later in the section. Denote $\|\hat{\mathbf{v}}\|$ as the speed of the bird known in each trial

of simulation. We repeat the EKF with randomized inputs for over 10^6 times to observe the quantity of

$$\frac{\|\hat{\mathbf{v}}\|}{\|\hat{\mathbf{v}}(k|k) - \hat{\mathbf{v}}(k-1|k-1)\|},$$

which is the inverse of the relative speed change at time k. We choose the weighting factor as

$$\omega(k) = E\left(\frac{\|\hat{\mathbf{v}}\|}{\|\hat{\mathbf{v}}(k|k) - \hat{\mathbf{v}}(k-1|k-1)\|}\right),$$

where function $E(\cdot)$ computes the expected value over all simulation trials for the targeted species. When the EKF converges, $\|\hat{\mathbf{v}}(k|k) - \hat{\mathbf{v}}(k-1|k-1)\|$ appears as a decreasing function of k after a few initial steps. Correspondingly, $\omega(k)$ is an increasing function of k. If $\|\hat{\mathbf{v}}(k|k) - \hat{\mathbf{v}}(k-1|k-1)\| \to 0$, then $\varepsilon(\mathbf{X}^{1:n})$ is smaller than that of the case $\|\hat{\mathbf{v}}(k|k) - \hat{\mathbf{v}}(k-1|k-1)\| \nrightarrow 0$. Therefore, to determine the EKF convergence, we employ a threshold δ on $\varepsilon(\mathbf{X}^{1:n})$ and introduce a new indicator variable,

$$I_{\text{EKF}}(\mathbf{X}^{1:n}) = \begin{cases} 1 \text{ (converge) if } \varepsilon(\mathbf{X}^{1:n}) < \delta, \\ 0 \text{ otherwise.} \end{cases} \tag{6.9}$$

Note that the threshold δ should be sufficiently small to ensure the convergence of EKF. The actual value of δ can be pre-determined in simulation. Then the decision-making in (6.8) is re-written as,

$$I(\mathbf{Z}^{1:n}) = \begin{cases} 1 \text{ (accept) if } \|\hat{\mathbf{v}}(n|n)\| \in \mathcal{V} \text{ and } I_{\text{EKF}}(\mathbf{X}^{1:n}) = 1, \\ 0 \text{ (reject) otherwise.} \end{cases} \tag{6.10}$$

The underlying condition for (6.10) to be an effective bird detection mechanism is that if observation $\mathbf{Z}^{1:n}$ is caused by the targeted bird species then the convergence of the EKF has to be guaranteed. Unfortunately, this condition usually does not hold due to two reasons: n is small and the measurement noise $\mathbf{w}(k)$ is too big. n is the number of images that contain the moving object. Due to its fast flying speed, the bird can only stay in the camera FOV for less than 1 second for most of the time. Actually, $n < 11$ for most cases in

our experiments. The measurement noise covariance matrix $W(k)$ is directly determined by the image segmentation error. Even at 1 pixel level, its relative range is 4% for a bird body length of 25 pixels. For the nonlinear deterministic discrete time system in (6.6), the large $W(k)$ means the EKF either fails to converge or converges very slowly according to [114]. The bird detection mechanism would have a close to 100% FN rate if the simple EKF implementation is used, which makes it useless.

6.5.3 Probable Observation Data Set-based EKF Method

Since simply applying EKF cannot address the bird detection problem, a new approach is required. Let us assume there is no measurement noise (i.e. $W(k) = \mathbf{0}_{4\times4}$) and no state transition noise $Q(k) = \mathbf{0}_{6\times6}$. At each time k, the EKF in (6.7) is a system of equations with four inputs, which is the dimensionality of $\mathbf{z}$, and six outputs, which is the dimensionality of $\mathbf{x}$. We also know that matrix A introduces two constraints: the constant speed and the linear trajectory. Therefore, the equation system can be solved within one step. The convergence of the EKF is not a problem when there is no noise provided that the bird does not fly in a degenerated trajectory (i.e. flying along the principal axis of the camera).

Although $Q(k) \neq \mathbf{0}_{6\times6}$ for most cases, the state transition noise $\mathbf{q}(k)$ is often very small, which leads to the following lemma,

Lemma 6 *The EKF described in (6.7) converges when* $W(k) = \mathbf{0}_{4\times4}$.

Proof We skip the proof because our system in (6.6) is a linear time-invariant discrete time system with a nonlinear observer. The convergence of its EKF can be proved by directly applying the results in [114]. ■

This is also confirmed in our experiments in which the EKF converges nicely under 7 periods (i.e. $n \leq 7$).

At first glance, this result is useless because we cannot get rid of the measurement noise. However, this result opens the door to a new approach. Define the observation data

without measurement error as $\mathbf{Z}^{1:n*} = [\mathbf{z}^*(1), \mathbf{z}^*(2), ..., \mathbf{z}^*(n)]^T$. Although we do not have $\mathbf{Z}^{1:n*}$, we know it is within the segmentation error range of $\mathbf{Z}^{1:n}$. For the k-th image, the measurement data is

$$\mathbf{z}(k) = [u^h(k), v^h(k), u^t(k), v^t(k)]^T.$$

Define the error-free measurement data at time k as

$$\mathbf{z}^*(k) = [u^{h*}(k), v^{h*}(k), u^{t*}(k), v^{t*}(k)]^T.$$

Given the segmentation error is within τ pixels, define

$$S_1(k) = [u^h(k) \pm \tau], \qquad S_2(k) = [v^h(k) \pm \tau],$$
$$S_3(k) = [u^t(k) \pm \tau], \qquad S_4(k) = [v^t(k) \pm \tau],$$

and the segmentation error range at time k as $\mathbb{S}(k)$. Hence,

$$\mathbf{z}^*(k) \in S_1(k) \times S_2(k) \times S_3(k) \times S_4(k) = \mathbb{S}(k). \tag{6.11}$$

We partition the entire segmentation error range set $\{\mathbb{S}(k), k = 1, 2, ..., n\}$ according to the convergence of the EKF using (6.9).

Definition 6.5.1 *Define the probable observation data set (PODS) $\mathbb{Z}^{1:n}$ as the set of observation data $\mathbf{Z}^{1:n}$ that satisfies the condition for the EKF convergence,*

$$\mathbb{Z}^{1:n} = \{\mathbf{Z}^{1:n} | \mathbf{z}(k) \in \mathbb{S}(k), k = 1, ..., n, \text{ and } \varepsilon(\mathbf{X}^{1:n}) \leq \delta\}. \tag{6.12}$$

Hence $\mathbf{Z}^{1:n*} \in \mathbb{Z}^{1:n}$. Each $\mathbf{Z}^{1:n}$ in the PODS is likely to be $\mathbf{Z}^{1:n*}$ and hence it is named as the probable observation data. For a given PODS $\mathbb{Z}^{1:n}$, there is a corresponding estimated state set $\mathbb{X}^{1:n}$, which contains a set of all possible estimated velocities at time n, which is defined as $\mathbb{V}$,

$$\mathbb{V} = \{\|\hat{\mathbf{v}}(n|n)\| \text{ such that } \mathbf{X}^{1:n} \in \mathbb{X}^{1:n}\}.$$

Then the decision making for our PODS-based EKF (PODS-EKF) method can be written as,

$$I(\mathbf{Z}^{1:n}) = \begin{cases} 1 \text{ (accept) if } \mathbb{V} \cap \mathcal{V} \neq \emptyset \text{ and } \mathbb{Z}^{1:n} \neq \emptyset, \\ 0 \text{ (reject) otherwise.} \end{cases} \tag{6.13}$$

Hence we have the following lemma,

Lemma 7 *If the non-degenerated observation data $\mathbf{Z}^{1:n}$ is triggered by a bird of the targeted species and the threshold δ for determining the convergence of EKF is sufficiently small, then $I(\mathbf{Z}^{1:n}) = 1$.*

Proof Since $\mathbf{Z}^{1:n}$ is triggered by the targeted species, its corresponding $\mathbf{Z}^{1:n*}$ ensures the convergence of the EKF according to Lemma 6.

Define $\mathbf{X}^{1:n*}$ as the corresponding estimated states for $\mathbf{Z}^{1:n*}$. Hence

$$\varepsilon(\mathbf{X}^{1:n*}) < \delta \rightarrow \mathbb{Z}^{1:n} \neq \emptyset,$$

because $\mathbf{Z}^{1:n*} \in \mathbb{Z}^{1:n}$.

Following our naming convention, $\hat{\mathbf{v}}^*(n|n)$ is the velocity component of $\hat{\mathbf{x}}^*(n|n) \in \mathbf{X}^{1:n*}$. Since the observation data is not degenerated, $\|\hat{\mathbf{v}}^*(n|n)\| \in \mathcal{V}$. We also know $\|\hat{\mathbf{v}}^*(n|n)\| \in \mathbb{V}$ by definition, $\mathbb{V} \cap \mathcal{V} \neq \emptyset$ holds. Since both conditions are satisfied, $I(\mathbf{Z}^{1:n}) = 1$. ■

Lemma 7 ensures that the PODS-EKF method theoretically has a zero FN rate in the bird detection, which is a very desirable property.

6.5.4 Approximate Computation for PODS-EKF

Computing the PODS $\mathbb{Z}^{1:n}$ is nontrivial. It is possible to use conventional searching methods such as a binary search. However, this would be very time consuming. Note that we actually do not need $\mathbb{Z}^{1:n}$ because all we need to know is whether the conditions

$\mathbb{V} \cap \mathcal{V} \neq \emptyset$ and $\mathbb{Z}^{1:n} \neq \emptyset$ hold or not. This allows an approximation method. For a given observation $\mathbf{Z}^{1:n}$, we define the following optimization problem,

$$\widetilde{\mathbf{Z}}^{1:n} = \arg \min_{\mathbf{z}(k) \in \mathbb{S}(k); k=1,...,n} \varepsilon(\mathbf{X}^{1:n}), \tag{6.14}$$

where $\widetilde{\mathbf{Z}}^{1:n}$ is the optimal solution to the minimization problem above. Actually, (6.14) is a typical nonlinear optimization problem with the error range $\mathbf{z}(k) \in \mathbb{S}(k); k = 1, ..., n$ and the EKF in (6.7) as constraints. There are many numerical methods from nonlinear programming that can be used here [115]. We apply a sequential quadratic programming (SQP) method [116]. Define $\widetilde{\mathbf{X}}^{1:n} = \{\widetilde{\mathbf{x}}(1|1), \widetilde{\mathbf{x}}(2|2), ..., \widetilde{\mathbf{x}}(n|n)\}$ as the estimated states corresponding to $\widetilde{\mathbf{Z}}^{1:n}$. We have the following lemma,

Lemma 8 $\varepsilon(\widetilde{\mathbf{X}}^{1:n}) > \delta \iff \mathbb{Z}^{1:n} = \emptyset$.

Proof Since (6.14) is a minimization problem, $\widetilde{\mathbf{X}}^{1:n}$ yields the minimal $\varepsilon(\mathbf{X}^{1:n})$, namely,

$$\varepsilon(\widetilde{\mathbf{X}}^{1:n}) > \delta \iff \varepsilon(\mathbf{X}^{1:n}) > \delta, \forall \mathbf{X}^{1:n} \in \mathbb{X}^{1:n} \tag{6.15}$$

$$\iff \mathbb{Z}^{1:n} = \emptyset. \tag{6.16}$$

■

It is worth mentioning that this method is an approximation in computation because the nonlinear programming solver often falls in a local minimum instead of a global minimum (see Remark 1).

Now we want to determine whether $\mathbb{V} \cap \mathcal{V} \neq \emptyset$. If we view the EKF output $\hat{\mathbf{v}}(n|n)$ as a function of $\mathbf{Z}^{1:n}$, it is continuous and differentiable with respect to each entry in $\mathbf{Z}^{1:n}$. Since $\mathbb{Z}^{1:n}$ is actually very small, the variance of the velocity in the set $\mathbb{V}$ is very small. Instead of comparing $\mathbb{V}$ to $\mathcal{V}$, we select a value in $\mathbb{V}$ to check if it is in $\mathcal{V}$. Define $\widetilde{\mathbf{v}}(n|n)$

as the velocity component of $\widetilde{\mathbf{x}}(n|n) \in \widetilde{\mathbf{X}}^{1:n}$. The chosen value is the $\|\widetilde{\mathbf{v}}(n|n)\|$ because it is readily available. Therefore, the approximation is

$$\|\widetilde{\mathbf{v}}(n|n)\| \in \mathcal{V} \iff \mathbb{V} \cap \mathcal{V} \neq \emptyset.$$

Remark 1 *Due to the approximation, the zero FN rate cannot be guaranteed. However, the FN rate is still very low (less than 5%) under the approximation as shown later in the physical experiment results. We conjecture that this is due to the fact that the nonlinearity of the problem is not very strong. For most of time, the SQP solver actually finds the global optimal. Therefore, the impact on the application is negligible. In practice, we can initiate the solver at different random starting points and run the solver multiple times, which can significantly increase the chance that the global optimal value can be found.*

6.5.5 Estimation of Initial States

The convergency and the performance of the EKF greatly depend on the accuracy of the initial state. Here we detail how to estimate the initial state of the flying bird,

$$\hat{\mathbf{x}}(0|0) = [\hat{\mathbf{p}}(0|0)^T, \hat{\mathbf{v}}(0|0)^T]^T \tag{6.17}$$

for each input, where $\hat{\mathbf{p}}(0|0)^T = [x(0|0), y(0|0), z(0|0)]$ and $\hat{\mathbf{v}}(0|0)^T = [\hat{\dot{x}}(0|0), \hat{\dot{y}}(0|0), \hat{\dot{z}}(0|0)]$.

We assume the bird speed is uniformly distributed across the range $\mathcal{V} = [v_{min},\ v_{max}]$. We set the initial speed of the bird as the mean speed: $\|\hat{\mathbf{v}}(0|0)\| = V = (v_{min} + v_{max})/2$. As shown in Fig. 6.4, given the image of the bird at the first observation $\mathbf{z}(1) = [u^h(1), v^h(1), u^t(1), v^t(1)]^T$, the body axis in image and the optical camera center form a plane. The 3D bird flying trajectory must be in this plane. Let us define the 3D coordinates of the bird head positions at discrete time $k,\ (k = 0, 1, ...)$, as

$$\hat{\mathbf{p}}(k|k) = \hat{\mathbf{p}}(0|0) + \hat{\mathbf{v}}(0|0)k\Delta T. \tag{6.18}$$

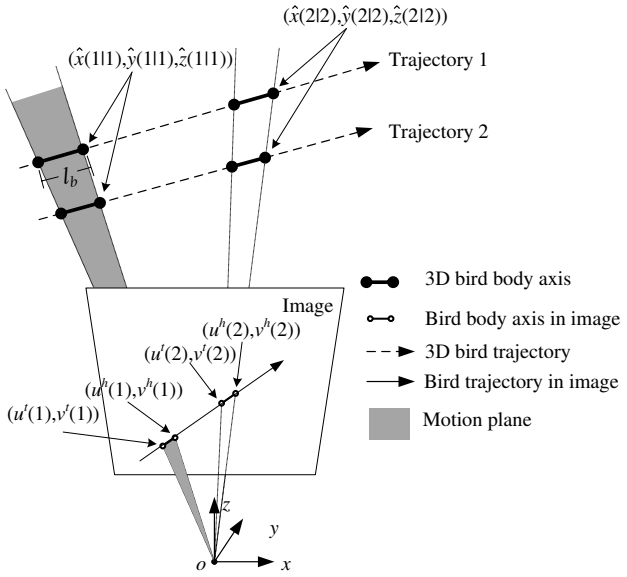


Fig. 6.4. An illustration of the initial state estimation for EKF.

Given the body axis length l_b and V, the position of the bird tail at time k is,

$$\hat{\mathbf{p}}(k|k) - \hat{\mathbf{v}}(k|k)\frac{l_b}{V} = \hat{\mathbf{p}}(0|0) + \hat{\mathbf{v}}(0|0)(k\Delta T - \frac{l_b}{V}). \tag{6.19}$$

Based on the pin-hole camera model, the bird head and tail positions project to the image at $(u^h(k), v^h(k))$ and $(u^t(k), v^t(k))$, respectively (see Fig. 6.4). Recall the perspective projection matrix $P_c = [K_c|\mathbf{0}_{3\times1}]$. Based on (6.18) and (6.19), this projection is represented in homogeneous coordinate system as,

$$\begin{aligned}\begin{bmatrix} u^h(k) \\ v^h(k) \\ 1 \end{bmatrix} &= \frac{1}{\hat{z}(k|k)} P_c \begin{bmatrix} \hat{\mathbf{p}}(k|k) \\ 1 \end{bmatrix} \\ &= \frac{P_c}{\hat{z}(0|0) + \hat{\dot{z}}(0|0)k\Delta T} \begin{bmatrix} \hat{\mathbf{p}}(0|0) + \hat{\mathbf{v}}(0|0)k\Delta T \\ 1 \end{bmatrix},\end{aligned} \tag{6.20}$$

and

$$\begin{bmatrix} u^t(k) \\ v^t(k) \\ 1 \end{bmatrix} = \frac{P_c}{\hat{z}(k|k) - \hat{\dot{z}}(k|k)\frac{l_b}{V}} \begin{bmatrix} \hat{\mathbf{p}}(k|k) - \hat{\mathbf{v}}(k|k)\frac{l_b}{V} \\ 1 \end{bmatrix}$$
$$= \frac{P_c}{\hat{z}(0|0) + \hat{\dot{z}}(0|0)k'} \begin{bmatrix} \hat{\mathbf{p}}(0|0) + \hat{\mathbf{v}}(0|0)k') \\ 1 \end{bmatrix}, \tag{6.21}$$

where $k' = k\Delta T - l_b/V$.

We have 6 unknowns as in (6.17). Each image data point has one bird head and one bird tail. Each body axis endpoint contributes two linear equations as shown in (6.20) and (6.21), respectively. Therefore, we only need the first 2 image data points (bird images) to form a system of 8 linear equations:

$$M_{8\times 6}\hat{\mathbf{x}}(0|0) = 0. \tag{6.22}$$

Obviously, (6.22) has non-zero solution. Actually, rank$(M_{8\times 6}) = 5$ and the solution to (6.22) is the null space of $M_{8\times 6}$, which can be represented as $\{\alpha \mathbf{x}_0\}$, where $\mathbf{x}_0$ is any non-zero solution to (6.22) and α is a scalar. This set of solutions correspond to an infinity number of parallel trajectories as shown in Fig. 6.4. Both trajectories 1 and 2 project back to the same points on the image. With a further constraint $\|\hat{\mathbf{v}}(0|0)\| = (v_{min} + v_{max})/2$, we obtain a unique initial state estimation $\hat{\mathbf{x}}(0|0)$.

6.6 Algorithm

We summarize our PODS-EKF based bird detection algorithm below in Algorithm 5. Note that the approximate computation of the PODS-EKF is used here.

Algorithm 5: PODS-EKF based Bird Detection Algorithm

for *the segmented motion block in i-th frame* **do**
  calculate the geometric center point C_i of the bird;
end
Connect $C_i,\ i = 1, 2, ..., n$ to generate a piecewise linear trajectory;
Obtain $\bar{\theta}$ from the trajectory;
for *the segmented motion block in i-th frame* **do**
  Obtain $\mathbf{z}(i)$ using the BBAF in (6.2);
end
Initialize the EKF using (6.20) and (6.21);
Solve the constrained nonlinear optimization problem in (6.14);
if $\|\widetilde{\mathbf{v}}(n|n)\| \in \mathcal{V}$ ***AND*** $\varepsilon(\widetilde{\mathbf{X}}^{1:n}) < \delta$ **then**
  return TRUE;
else
  return FALSE;
end

6.7 Experiments

We have implemented the PODS-EKF algorithm and tested the algorithm on both the simulated data and the real data from field experiments. The computer used in the test is a desktop PC with an Intel Core 2 Duo 2.13GHz CPU and 2GB RAM. The PC runs Microsoft Windows XP. The BBAF has been implemented using Microsoft Visual C++. The PODS-EKF filter has been implemented using Matlab v7.0. We choose Arecont Vision 3100 high resolution networked video cameras as imaging devices. The camera runs at 11 frames per second with a resolution of 3 mega pixels per frame. The lens for the camera is a Tamron auto-iris vari-focus lens with a focal length range of 10-40mm. We have adjusted the lens to ensure a 20° horizontal FOV.

6.7.1 Bird Body Axis Filter Test

We first verify whether the BBAF is capable of extracting the bird body axis from the noisy data. We have used two data sets in testing. The first data set has been collected from our campus and contains 61 bird motion sequences with a total of 341 segmented

birds which are mostly rock pigeons and American crows. The second data set has been collected from our test site in Arkansas and has a total of 88 images with 11 different species at 8 images per species. We compare the output of BBAF with the corresponding ground truth which is a human's choice in bird body axes. The difference between the BBAF output and the ground truth has means of 0.30° and 0.63°, and the same standard deviation of 3.7° for the first and the second data sets, respectively. The student $t-$test shows that the output of BBAF and human choices come from the same distribution for both data sets with statistic significance, which is satisfying.

6.7.2 Simulation

The second step is to test the performance of our PODS-EKF using the simulated inputs. The simulated inputs allow us to test the bird filtering performance under a full range of possible changes in the parameter settings, which are usually unavailable in physical experiments.

Random trajectory generation

$\mathbf{Z}^{1:n}$ needs to be generated from a random trajectory. First, four random numbers are generated as the coordinates of two points in the image plane. The two points determine a line in the image. The line and the camera center determine a motion plane in which the motion sequence will be generated. We know that the camera FOV is a pyramid with its top vertex at the camera center. The plane intersects with two faces of the pyramid. The fifth random binary number is generated to choose one of the two faces as the initial face through which the bird enters the camera FOV. The chosen face intersects with the motion plane and yields a line segment. We generate a point on this line segment using the sixth random number. The point is used as the initial position of the bird. This line segment's extension line divides the motion plane into two halves. We are interested in the half motion plane that intersects with the pyramid. The seventh random number in

Table 6.1
Species used in the experiments. The data sources are listed in the corresponding reference.

Species	l_b (cm)	$\mathcal{V}$ (km/h)
House Sparrow	15 [1]	[29, 40] [2]
Rock pigeon	33 [3]	[24, 56] [4]
Ivory-billed woodpecker	48 [5]	[32, 64] [6]
Red-tailed hawk	56 [7]	[32, 64][8]

the range of $[0, \pi)$ is generated as the pitch angle of the bird heading on the half motion plane. Finally, the eighth random number is used to generate the speed of the bird. Hence, 8 random numbers determine a complete trajectory of a flying bird. By projecting the trajectory back to the image plane with a preset bird body length, we obtain $\mathbf{Z}^{1:n}$.

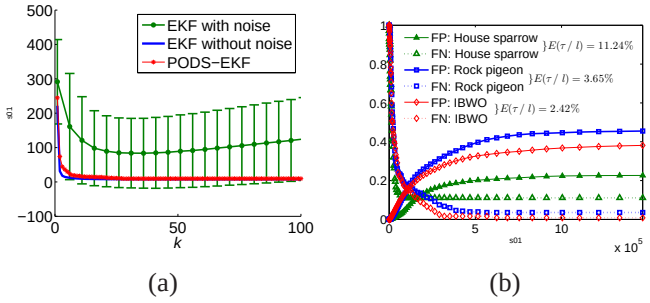
EKF convergence

An immediate step in the simulation is to verify if a regular EKF converges without measurement noise. Although Lemma 6 ensures the convergence in theory, it is unclear how many steps it would take. We simulate three types of birds in the test: house sparrows, rock pigeons, and IBWOs. House sparrows and rock pigeons are common birds in Texas and the IBWO is the rare bird which our system is used to search for in Arkansas. The three species represent small, medium, and large birds, respectively (see Table 6.1).

For each species, we generate 10^6 different sets of random inputs to test the regular EKF. Fig. 6.5(a) shows the EKF convergence for rock pigeons under different configura-

[1] http://en.wikipedia.org/wiki/House_Sparrow.
[2] http://www.garden-birds.co.uk/information/flight.htm
[3] http://www.allaboutbirds.org/guide/Rock_Pigeon/lifehistory
[4] http://www.ct.gov/DEP/cwp/view.asp?A=2723&Q=326076
[5] http://animals.nationalgeographic.com/animals/birds/ivory-billed-woodpecker.html
[6] http://news.mongabay.com/2007/0217-ibw.html
[7] http://www.nysite.com/nature/fauna/redhawk.htm
[8] http://www.nysite.com/nature/fauna/redhawk.htm

tions by tracking errors in speed $\|\hat{\mathbf{v}}(k|k) - \hat{\mathbf{v}}\|$, where $\hat{\mathbf{v}}$ is the true bird velocity known in simulation. It is shown that without image noise, the regular EKF nicely converges (the blue curve) as Lemma 6 predicts. With the image noise ($\tau = 1$ pixel), the regular EKF cannot converge and yields a big error variance (indicated as the green curve and vertical green line segments, respectively). We also show the output of our PODS-EKF (the red curve). Although not required, it is desirably close to the noise-free case.



(a) (b)

Fig. 6.5. (a) Convergence for different EKF configurations based on simulated rock pigeon data. (b) FP and FN rates with respect to δ in both simulation and physical experiments.

Performance of PODS-EKF under simulated inputs

Now we are ready to analyze the performance of PODS-EKF. We generate a set of random inputs to mimic three birds as in Table 6.1. We set a speed range from 15 to 85 km/h with an incremental step of 5 km/h and a bird size range from 10 to 60 cm with an incremental step of 2 cm. We set the segmentation error range $\tau = 1$ pixel. For each setting of the input data, 20 trials are carried out. The average computation time for each trial is 5.6 seconds. Fig. 6.5(b) demonstrates how the rates of FP and FN change according to δ. After $\delta > 1.0 \times 10^6$, the FN rates can be reasonably controlled to be less than 10%, 4% and 1%, for house sparrows, rock pigeons and IBWOs, respectively. This confirms that the approximation computation is reasonable. The reason PODS-EKF works worst for house

sparrows is that with the same FOV in the simulation, the smallest house sparrows lead to the highest noise-signal ratio, indicated as $E(\tau/l)$ in Fig. 6.5(b). Our PODS-EKF is not biased for a particular bird. To cope with small birds, we can increase the focal length to reduce $E(\tau/l)$. This test also tells us how to choose a proper lens for a targeted species in applications to ensure the best performance. The FP rates of the PODS-EKF are 23%, 45% and 38%, respectively, which are a little high. However, considering that we are comparing the targeted bird with birds similar in size and speed, this result is not surprising. In fact, the algorithm should behave better in real tests where noises from the moving objects have much larger range in both size and speed. Furthermore, the monocular system has difficulty in detecting objects with their trajectories close to the optical axis, which also contributes to the high FP rate.

6.7.3 Physical Experiments

We have conducted two field experiments: detecting flying rock pigeons, and assisting the search of the legendary IBWOs.

Data sets and ground truth

Since there is no existing data set or benchmark for the evaluation of bird detection. We have to use our data collected from both our campus and the experiment site in Arkansas for testing. The input data sets of our PODS-EKF filter are segmented motion sequences using a pre-filtering method detailed in [113], which is solely a salient motion detection method built on [18] by performing a connectivity check to eliminate small moving objects and periodic noises such as tree vibrations. The method pre-filters out small moving objects (less than 5×5 pixels) because they are too small for a human to positively identify a bird species at the end. The pre-filtering reduces noises when maintaining a zero FN rate. We have collected a total 1205 motion sequences after the pre-filtering.

The motion sequences used to test the PODS-EKF filter is the motion sequences containing more than 8 frames, which result in 119 out of the 1205 motion sequences. The reason we need at least 8 frames is due to the fact that even a noise-free EKF would need 7 steps to converge as shown in simulation (see Fig. 6.5(a)). The PODS-EKF filter works only if the corresponding noise-free EKF can converge. The ratio of 119/1205 is low because our camera frame rate is slow (11 fps) due to its high resolution. Better cameras would certainly improve that ratio and it is not a concern for our algorithm.

The surviving 119 motion sequences are the testing data set. Among them, 29 sequences are caused by rock pigeons, 21 sequences are caused by 10 difference species of birds including great blue herons, northern flickers, great egrets, America crows, red-tailed hawks, chimney swifts, Mississippi kites, purple Martins, pileated woodpeckers, belted kingfishers, and some un-identifiable birds. The remaining 69 motion sequences caused by noises such as moving clouds, falling leaves, flying insects, etc.

The ground truth is obtained by using human inputs on the same motion sequence that the PODS-EKF filter is tested.

Detecting a flying pigeon

Here the targeted species is rock pigeons since they are the dominating species in our data set.

Fig. 6.6 compares the potential outputs of regular EKFs and the output of the PODS-EKF using a sample rock pigeon sequence. The targeted species flying speed range is also overlaid on the figure. It is shown that the chance that the regular EKF would converge to the proper value is very small, which confirms the simulation results in Fig. 6.5(a). On the other hand, the PODS-EKF finds the optimal observation that ensures the EKF converges to the bird speed range.

Fig. 6.7(a) shows how the FN and FP rates of the PODS-EKF change according to δ. The convergence threshold is set as $\delta = 1.35 \times 10^6$. The outcome of the algorithm is summarized in Table 6.2.

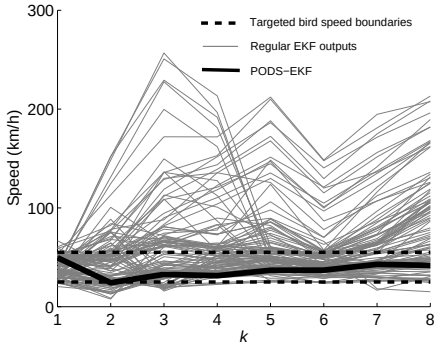


Fig. 6.6. Predicted bird speeds by a regular EKF with 200 possible observations in PODS and that by the PODS-EKF in detecting a rock pigeon.

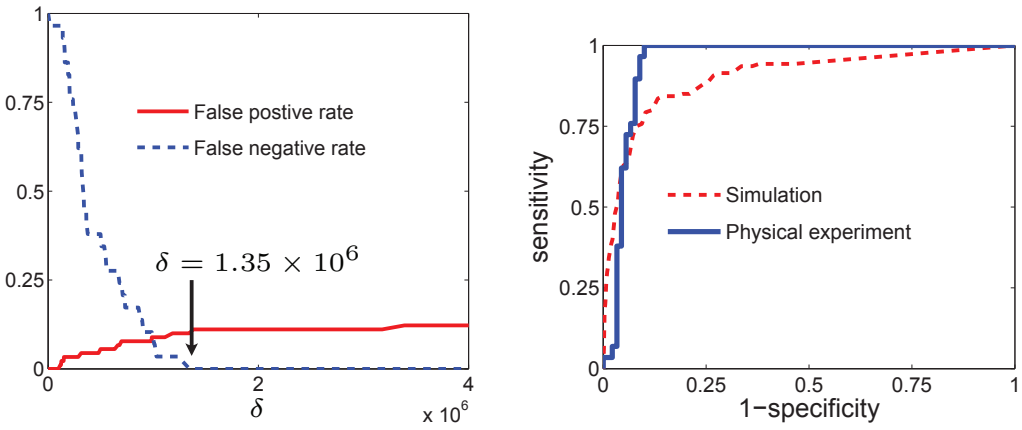
Table 6.2 indicates that our filtering algorithm can achieve extremely low FN rate ($0/29 = 0\%$). This is very important for the purpose of finding rare birds species. The FP rate is $9/90 = 10\%,$ which is better than that of the simulation results. This is due to the fact that it is much easier for the algorithm to distinguish the targeted species from noises such as flying insects and falling leaves in physical experiments rather than from other birds with similar body size and speed as in the simulation. Since the monocular vision system cannot provide depth information, the algorithm cannot achieve zero FP. Fortunately, this is allowable for our applications. The expectation of the algorithm is to reduce the video data for identification without compromising the FN rate.

Table 6.2
Experimental results from the rock pigeon filtering experiment.

	Pigeon	Not pigeon
Predicted pigeon	29	9
Predicted not pigeon	0	81

Fig. 6.7(b) illustrates the receiver operating characteristic (ROC) curves for both the simulation and physical experiments for rock pigeons. The convergence threshold ranges are $[4.6 \times 10^3, 1.5 \times 10^6]$ and $[1.8 \times 10^4, 3.3 \times 10^6]$ for the simulation and the physical

experiments, respectively. The areas under the ROC curve are 91.5% and 95.0% for simulation and the physical experiments, respectively, which again show that the algorithm performs much better in the physical experiments.



(a) (b)

Fig. 6.7. Physical experiment results for detecting a rock pigeon: (a) FN and FP rates w.r.t. δ and (b) The ROC curves for both the simulation and the physical experiments.

Assisting the search of the legendary IBWO in Arkansas

Since October 2006, our team have begun to assist the search for the thought-to-be-extinct IBWOs. The IBWO is the largest woodpecker in North America and was last seen over 60 years ago. Sporadic sightings have been reported in past decades but no definite evidence such as a clear picture of the bird is available. In October 2006, we installed a camera system in Bayou DeView wildlife refuge in Arkansas, where sightings of the bird were reported in 2004. Due to the low FN rate, our PODS-EKF algorithm is very desirable for this type of applications. Fig. 6.1 is taken from the camera. The system monitored the sky from Oct. 2006 to Oct. 2007. After initial motion detection filtering as in [113], we reduce the total 29.41TB video data to 27.42GB, which is still prohibitively huge for

human experts. After applying the PODS-EKF, we eventually reduce the data volume to 146.7MB (about 960 images), which is a reasonable amount of workload for a human expert to review to make the final identification. The overall reduction rate is 99.9995%. Unfortunately, no IBWO image has been captured yet.

However, our algorithm can also detect other birds such as red-tailed hawks in the region where our camera is installed. As shown in Table 6.1, a red-tailed hawk is a bigger bird than an IBWO but flies at about the same speed as IBWOs. The algorithm is able to successfully detect red-tailed hawks. Considering that our algorithm has successfully detected birds that are either bigger than IBWOs (red-tailed hawks) or smaller than IBWOs (rock pigeons), we are confident that if an IBWO flies cross the field of view of our camera, our system is able to capture the bird. Although no IBWO is detected, our system and algorithm design is successful.

6.8 Conclusions

We reported our development of a bird filtering algorithm to assist the search for rare bird species. We showed that a regular EKF cannot be directly applied because the EKF would not converge due to the high measurement error and the limited observation data due to the high flying speed of the bird. Instead, we developed a novel PODS-EKF method based on whether there exists a probable measurement in PODS with the corresponding speed in the flying speed range of the targeted species. The algorithm was extensively tested using both simulated inputs and physical experiments. The results were satisfying and the PODS-EKF bird filter reduced the video data by 99.9995% with a close to zero FN rate and 95.0% area under the ROC curve in physical experiments.

In the future, an immediate extension is to consider the case without the linear flying trajectory and/or the constant velocity. We will consider the simultaneous filtering of a flock of birds using a single camera or multiple cameras. It is also interesting to employ a robotic camera to combine tracking with filtering. A pan-tilt-zoom robotic camera can give a closer view of a flying bird, which reduces the measurement error at a price of

increasing the state transition error and the nonlinearity of the system. We will investigate how to achieve the best tradeoff.

7. CONCLUSION AND FUTURE WORK

In this dissertation, we extended the traditional telerobotic system architecture by including heterogenous components such as humans, robots, sensors, and automated agents. We term it as MOMR++ system. Since the relationship between various heterogeneous components are much more complicated than that in traditional systems, to reach the best potential and performance of the system, many technical challenges need to be addressed. We addressed two major challenges in the MOMR++ system by two automated collaborative observation systems, respectively.

7.1 Autonomous Crowd Surveillance System

7.1.1 System Development

We have developed an autonomous crowd surveillance system. It consists of p $(p > 0)$ robotic pan-tilt-zoom (PTZ) cameras assisted with a fixed wide-angle camera. The wide-angle camera provides an overview of the scene and detects n moving objects, which are considered as objects of interests. Based on the output of the wide angle camera, the system generates spatiotemporal observation requests for each object, which are candidates for close-up views using the PTZ cameras. The system controls the PTZ cameras to track and observe the moving objects by satisfying these observation requests. We have implemented the system and tested it for pedestrian surveillance in a university campus environment. Our system outperforms an existing work by increasing the number of observed objects by 210%.

7.1.2 Algorithmic Development

Since there are usually much more observation requests than the number of cameras, i.e., $p \ll n,$ coordinating and planing the cameras to best satisfy these requests is a chal-

lenge problem. I formulate the camera planning and control problem as an optimization problem: the p-frame problem which maximizes the overall satisfaction to observation requests by computing the optimal control command for the p frames. We use the satisfaction as the metric for measuring the control commands with participants' input requests. Each request is an iso-oriented rectangle with desirable resolution. The output are p rectangular frames as the camera control commands.

We have applied computational geometry and optimization theory to solve the p-frame problem. We have developed an approximation algorithm which runs in $O(n/\epsilon^3 + p^2/\epsilon^6)$ for n requests, p frames, and the approximation bound ϵ. I also developed an exact 2-frame algorithm which runs in $O(n^3)$.

7.2 Bird Species Detection System

7.2.1 System Development

We have developed an autonomous rare bird species detection system. We have set up the system in the forest near Brinkley Arkansas and it runs continuously for a year for searching the thought-to-be-extinct ivory-billed woodpecker. The cameras monitor the sky and detect any motion. The system autonomously distinguish the motion caused by the targeted species from other motion noises and only preserve the video data for the targeted species. During the one-year search, the system reduces the raw video data of 29.41TB to only 146.7MB (reduction rate 99.9995%).

7.2.2 Algorithmic Development

To recognize the targeted bird, I formulated the flying bird dynamics with a dynamic linear model. An EKF has been used to track the bird head and body axis length. The species decision is made by comparing the tracked bird state with prior profile of the particular bird species. It is showed that a regular EKF cannot be directly applied because the

EKF would not converge due to the high measurement error and the limited observation data due to the high flying speed of the bird. To tackle this issue, We quantified the uncertainty in the bird species recognition due to the uncertainty in the observation uncertainty. We developed a novel Probable Observation Data Set (PODS)-based EKF method. The new PODS-EKF algorithm searches the measurement error range for all probable observation data that ensures the convergence of the corresponding EKF, which guarantees to bound the true (noise-free) bird state. We then formulate the recognition problem as an optimization problem which searches in the PODS for the most likely observation corresponding to the true (noise-free) bird state. In experiments with real video data, the algorithm achieves 95% area under the ROC curve.

7.3 Future Work

The research on the MOMR++ system is still in its infancy. It can be viewed as a generalization of the MOMR systems by extending the range of control decision makers beyond just humans. Future research will further explore the relationship between the heterogeneous participants, such as competition and collaboration. Coordination of these heterogeneous participants will be one of the keys for a successful MOMR++ system. Another future direction is to further enhance the decision-making capability for the non-human components so that the system can be more autonomous.

7.3.1 Coordination of System Components: Extension of Frame Selection Problem

Overlapping frame selection

We have proposed the p-frame problem for coordinating the various system components with limited sensing resources. An immediate extension of the frame selection problem is to think of relaxing the assumptions to allow camera frames to overlap in the future. Allowing the frames to overlap requires a new satisfaction metric to measure the frames

with consideration of the possible redundant coverage over requests. It is interesting to investigate how different frame selection formulation would impact the system performance and how they fit human user need in practice.

Frame selection with traveling time

Another interesting extension is to consider the camera traveling time within the request assignment. We proposed a synchronized architecture in Section 2. Intuitively, asynchronized observation by multiple PTZ cameras would further enhance the system performance since it reduces the cameras' waiting time. A new metric that incorporates the camera traveling time into the satisfaction is needed. New algorithms such as fast incremental algorithm applied on the results of the p-frame algorithm may worth research.

7.3.2 Object Recognition: Extension of Bird Species Detection

Modeling bird dynamics

In MOMR++ system, sensors and automated agents are able to recognize and analyze the content of objects in remote environment. We developed the bird species recognition system. An immediate extension is to consider the case without the assumption of bird linear flying trajectory and/or the constant velocity. It requires to model the bird flying motion by a nonlinear dynamic model. It also requires to build a new belief estimator that captures the nonlinearity of the bird motion. Considering the image segmentation error remains significant, the recognition uncertainty caused by the measurement uncertainty under this nonlinear model needs to be formulated. Then the convergence issue of the new estimator and the corresponding recognition decision making will be another interesting issue.

Recognition of flock of birds

It is also interesting to consider the simultaneous detection and recognition of a flock of birds. Multiple object tracking approach is needed. Instead of looking into each individual bird, more interesting extension is to examine the group behavior pattern, such as the formation and the correlation between individual bird trajectory. We can use the group behavior pattern as the signature feature for bird species recognition. It also provides a lot more insights to understanding the behavior of particular bird species.

Signature features

It is also interesting to examine other signature features than the dynamics information for the bird species recognition. One promising feature is bird's wing flapping frequency. Preliminary study extracts bird's extreme point and tracks changes in the bird image. By comparing its frequency domain response with prior bird wing flapping frequency pattern, the bird species can be recognized. This approach is independent of the bird flying trajectory and requires least camera calibration.

Active bird detection

In Section 6, we use a static camera for detecting flying birds. It is natural to think of using active cameras, such as PTZ cameras to actively search for, track, and recognize the bird. By doing so, we gain more accurate observation with high-resolution images and longer observation duration, at the price of increased bird state transition uncertainty due to the uncertainty in the camera movements. It is interesting to look into the tradeoff between these two effects. An immediate challenge is to to track and segment the bird out of the background. A preliminary study suggests a panoramic background subtraction technique since it is robust to outdoor lighting conditions.

7.3.3 Scene Structure Understanding Panoramic Background Model

To support the active bird detection above, we are exploring to construct a panoramic background model using a PTZ camera for bird segmentation. Each pixel in the panoramic model captures its temporal color distributions. Using a single PTZ camera to construct the panoramic model, it requires to address the image alignment and registration under different scale or zoom.

A feature map is also required and maintained for each camera frame registering into the panorama. It will be interesting to research on the feature map storage and update approach with different scales, which supports efficient query without compromising the feature resolution.

REFERENCES

[1] G. Niemeyer, C. Preusche, and G. Hirzinger, "Telerobotics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. New York: Springer-Verlag, 2008, pp. 741–757.

[2] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl, "Sensor-based space robotics-ROTEX and its telerobotic features," *IEEE Trans. Robot. Autom.*, vol. 9, no. 5, pp. 649–663, Oct. 2002.

[3] J. Marescaux and F. Rubino, "Transcontinental robot-assisted remote telesurgery, feasibility and potential applications," in *Teleophthalmology*, K. Yogesan, S. Kumar, L. Goldschmidt, and J. Cuadros, Eds. New York: Springer-Verlag, 2006, pp. 261–265.

[4] D. Song, *Sharing a Vision: Systems and Algorithms for Collaboratively-Teleoperated Robotic Cameras*. New York: Springer-Verlag, 2009.

[5] N. Chong, T. Kotoku, K. Ohba, K. Komoriya, N. Matsuhira, and K. Tanie, "Remote coordinated controls in multiple telerobot cooperation," in *Proc. IEEE Int. Conf. Robotics and Automation*, San Francisco, CA, April 2000, pp. 3138–3343.

[6] N. Tesla, "Method of and apparatus for controlling mechanism of moving vessels or vehicles," Nov. 1898, US Patent 613,809.

[7] R. Goertz and W. Thompson, "Electronically controlled manipulator," *Nucleonics*, vol. 12, pp. 46–47, Nov. 1954.

[8] K. Goldberg, J. Santarromana, G. Bekey, S. Gentner, R. Morris, J. Wiegley, and E. Berger, "The telegarden," in *Proc. ACM SIGGRAPH*, Los Angeles, CA, Aug. 1995, pp. 135–1140.

[9] K. Goldberg, D. Song, and A. Levandowski, "Collaborative teleoperation using networked spatial dynamic voting," *Proc. IEEE*, vol. 91, no. 3, pp. 430–439, Mar. 2003.

[10] S. Sirouspour and P. Setoodeh, "Multi-operator/multi-robot teleoperation: an adaptive nonlinear control approach," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Edmonton, Canada, Aug. 2005, pp. 1576–1581.

[11] C. H. Fua and S. S. Ge, "Cobos: Cooperative backoff adaptive scheme for multirobot task allocation," *IEEE Trans. Robot.*, vol. 21, no. 6, pp. 1168–1178, Dec. 2005.

[12] J. Liu, L. Sun, T. Chen, X. Huang, and C. Zhao, "Competitive multi-robot teleoperation," in *Proc. IEEE Int. Conf. Robotics and Automation*, Barcelona, Spain, Apr. 2005, pp. 75–80.

[13] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," in *Proc. 13th Conf. Uncertainty in Artificial Intelligence*, Providence, RI, Aug. 1997, pp. 175–181.

[14] Y. H. Yang and M. D. Levine, "The background primal sketch: An approach for tracking moving objects," *Mach. Vis. Appl.*, vol. 5, no. 1, pp. 17–34, Dec. 1992.

[15] C. Stauffer and W. Grimson, "Adaptive background mixture modles for real-time tracking," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Ft. Collins, CO, Jun. 1999, pp. 246–252.

[16] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 780–785, Jul. 1997.

[17] A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Washington D.C., Jun./Jul. 2004, pp. 302–309.

[18] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Proc. 6th European Conf. Computer Vision*, vol. 2, Dublin, Ireland, Jun./Jul. 2000, pp. 751–761.

[19] C. Ridder, O. Munkelt, and H. Kirchner, "Adaptive background estimation and forground detection using kalman-filtering," in *Proc. Int. Conf. Recent Advances in Mechatronics*, Istanbul, Turkey, Aug. 1995, pp. 193–199.

[20] M. Heikkilä and M. Pietikanäinen, "A texture-based method for modeling the background and detecting moving objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 657–662, Apr. 2006.

[21] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, "Image change detection algorithm: A systematic survey," *IEEE Trans. Image Process.*, vol. 14, no. 3, pp. 294–307, Mar. 2005.

[22] C. J. Veenman, M. J. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 1, pp. 54–72, Jan. 2001.

[23] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577, May 2003.

[24] J. Kang, I. Cohen, and M. Medioni, "Object reacquisition using invariant appearance model," in *Proc. Int. Conf. Pattern Recognition*, Cambridge, UK, Aug. 2004, pp. 759–762.

[25] S. Avidan, "Support vector tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, Aug. 2004.

[26] M. Bertalmio, G. Sapiro, and G. Randall, "Morphing active contours," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 7, pp. 733–737, Aug. 2000.

[27] K. Sato and J. Aggarwal, "Temporal spatio-velocity transform and its application to tracking and interaction," *Computer Vision and Image Understanding*, vol. 96, no. 2, pp. 100–128, Nov. 2004.

[28] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, pp. 1–45, Dec. 2006.

[29] C. Micheloni and G. L. Foresti, "Zoom on target while tracking," in *Proc. IEEE Int. Conf. Image Processing*, New York, NY, Sep. 2005, pp. 117–120.

[30] X. Zhou, R. T. Collins, T. Kanade, and P. Metes, "A master-slave system to acquire biometric imagery of humans at distance," in *Proc. First ACM SIGMM Int. Workshop on Video Surveillance*, Berkeley, CA, Nov. 2003, pp. 113–120.

[31] S. Stillman, R. Tanawongsuwan, and I. Essa, "A system for tracking and recognizing multiple people with multiple cameras," in *Proc. Second Int. Conf. Audio-Vision Based Person Authentication*, Washington D.C., Mar. 1999, pp. 96–101.

[32] M. Greiffenhagen, V. Ramesh, D. Comaniciu, and H. Niemann, "Statistical modeling and performance characterization of a real-time dual camera surveillance system," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Hilton Head, SC, Jun. 2000, pp. 335–342.

[33] A. Hampapur, S. Pankanti, A. Senior, Y.-L. Tian, L. Brown, and R. Bolle, "Face cataloger: multi-scale imaging for relating identity to location," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, Miami, FL, Jul. 2003, pp. 13–20.

[34] R. Bodor, R. Morlok, and N. Papanikolopoulos, "Dual-camera system for multi-level activity recognition," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Edmonton, Canada, Sep./Oct. 2004, pp. 643–648.

[35] L. Fiore, D. Fehr, R. Bodor, A. Drenner, G. Somasundaram, and N. Papanikolopoulos, "Multi-camera human activity monitoring," *J. Intell. Robot. Syst.*, vol. 52, no. 1, pp. 5–43, May 2008.

[36] C. J. Costello, C. P. Diehl, A. Banerjee, and H. Fisher, "Scheduling an active camera to observe people," in *Proc. ACM 2nd Int. Workshop on Video Surveillance and Sensor Networks*, New York, NY, Oct. 2004, pp. 39–45.

[37] S. N. Lim, L. Davis, and A. Elgammal, "Scalable image-based multi-camera visual surveillance system," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, Miami, FL, Jul. 2003, pp. 205–212.

[38] A. D. Bimbo and F. Pernici, "Distant targets identification as an on-line dynamic vehicle routing problem using an active-zooming camera," in *Proc. IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, Beijing, China, Oct. 2005, pp. 97–104.

[39] F. Qureshi and D. Terzopoulos, "Smart camera networks in virtual reality," *Proc. IEEE*, vol. 96, no. 10, pp. 1640–1656, Oct. 2008.

[40] S. N. Lim, L. Davis, and A. Mittal, "Constructing task visibility intervals for video surveillance," *Multimedia Systems*, vol. 12, no. 3, pp. 211–226, Dec. 2006.

[41] C. Zhang, Z. Liu, Z. Zhang, and Q. Zhao, "Semantic saliency driven camera control for personal remote collaboration," in *Proc. IEEE 10th Workshop on Multimedia Signal Processing*, Cairns, Australia, Oct. 2008, pp. 28–33.

[42] E. Sommerlade and I. Reid, "Information-theoretic active scene exploration," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Anchorage, AK, Jun. 2008, pp. 1–7.

[43] F. Z. Qureshi and D. Terzopoulos, "Surveillance camera scheduling: a virtual vision approach," *Multimedia Systems*, vol. 12, no. 3, pp. 269–283, Dec. 2006.

[44] A. W. Senior, A. Hampapur, and M. Lu, "Acquiring multi-scale images by pan-tilt-zoom control and automatic multi-camera calibration," in *Proc. IEEE Workshop on Applications of Computer Vision*, Breckenridge, CO, Jan. 2005, pp. 433–438.

[45] C. R. Wren, U. M. Erdem, and A. J. Azarbayejani, "Functional calibration for pan-tilt-zoom cameras in hybrid sensor networks," *Multimedia Systems*, vol. 12, no. 3, pp. 255–268, Dec. 2006.

[46] U. Erdem and S. Sclaroff, "Look there! Predicting where to look for motion in an active camera network," in *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, Como, Italy, Sep. 2005, pp. 105–110.

[47] D. Song, A. F. van der Stappen, and K. Goldberg, "Exact algorithms for single frame selection on multi-axis satellites," *IEEE Trans. Autom. Sci. Eng.*, vol. 3, no. 1, pp. 16–28, Jan. 2006.

[48] D. Song and K. Goldberg, "Approximate algorithms for a collaboratively controlled robotic camera," *IEEE Trans. Robot.*, vol. 23, no. 5, pp. 1061–1070, Oct. 2007.

[49] D. Song, N. Qin, and K. Goldberg, "Systems, control models, and codec for collaborative observation of remote environments with an autonomous networked robotic camera," *Auton. Robots*, vol. 24, no. 4, pp. 435–449, May 2008.

[50] X. G. Wang, M. Moallem, and R. Patel, "An internet-based distributed multiple-telerobot system," *IEEE Trans. Syst. Man, Cybern., Part A*, vol. 33, no. 5, pp. 627–634, Sep. 2003.

[51] C. Zhao, J. Liu, Y. Li, T. Chen, and L. Sun, "Virtual repulsive force in competitive multi-robot teleoperation," in *Proc. 3rd Int. Conf. Information Technology and Applications*, Sydney, Australia, Jul. 2005, pp. 15–20.

[52] N. Megiddo and K. Supowit, "On the complexity of some common geometric location problems," *SIAM J. Computing*, vol. 13, pp. 182–196, Feb. 1984.

[53] D. Eppstein, "Fast construction of planar two-centers," in *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, New Orleans, LA, Jan. 1997, pp. 131–138.

[54] E. M. Arkin, G. Barequet, and J. Mitchell, "Algorithms for two-box covering," in *Proc. Sympos. Computational Geometry*, Sedona, AZ, Jun. 2006, pp. 459–467.

[55] H. Alt, E. M. Arkin, H. Brönnimann, J. Erickson, S. P. Fekete, C. Knauer, J. Lenchner, J. S. B. Mitchell, and K. Whittlesey, "Minimum-cost coverage of point sets by disks," in *Proc. Sympos. Computational Geometry*, Sedona, AZ, Jun. 2006, pp. 449–458.

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

[57] P. K. Agarwal, S. Bereg, O. Daescu, H. Kaplan, S. C. Ntafos, and B. Zhu, "Guarding a terrain by two watchtowers," in *Proc. Sympos. Computational Geometry*, Pisa, Italy, Jun. 2005, pp. 346–355.

[58] D. Eppstein, M. T. Goodrich, and N. Sitchinava, "Guard placement for efficient point-in-polygon proofs," in *Proc. Sympos. Computational Geometry*, Gyeongju, South Korea, Jun. 2007, pp. 27–36.

[59] T. Shermer, "Recent results in art galleries," *Proc. IEEE*, vol. 80, no. 9, pp. 1384–1399, Sep. 1992.

[60] D. Song, A. Pashkevich, and K. Goldberg, "Sharecam part II: Approximate and distributed algorithms for a collaboratively controlled robotic webcam," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots*, Las Vegas, NV, Oct. 2003, pp. 1087 – 1093.

[61] S. Har-Peled, V. Koltun, D. Song, and K. Goldberg, "Efficient algorithms for shared camera control," in *Proc. 19th ACM Sympos. Computational Geometry*, San Diego, CA, Jun. 2003, pp. 68–77.

[62] D. Song, A. van der Stappen, and K. Goldberg, "An exact algorithm optimizing coverage-resolution for automated satellite frame selection," in *Proc. IEEE Int. Conf. Robotics and Automation*, New Orleans, LA, May 2004, pp. 63–70.

[63] N. Megiddo, "On the complexity of some geometric problems in unbounded dimension," *J. Symb. Comput.*, vol. 10, no. 3-4, pp. 327–334, Sep./Oct. 1990.

[64] S. Bespamyatnikh and D. Kirkpatrick, "Rectilinear 2-center problems," in *Proc. 11th Canadian Conf. Computational Geometry*, Vancouver, Canada, Aug. 1999, pp. 68–71.

[65] M. Ko, R. Lee, and J. Chang, "An optimal approximation algorithm for the rectilinear m-center problem," *Algorithmica*, vol. 5, pp. 341–352, Jun. 1990.

[66] X. Huang, J. Liu, L. Sun, W. Sun, and T. Chen, "A criterion for evaluating competitive teleoperation system," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Beijing, China, Oct. 2006, pp. 5214–5219.

[67] Z. Wang, Y. Liu, D. Wang, Q. Li, T. Chen, Y. Liu, and Y. Jia, "Internet based robot competition and education," in *Proc. IEEE Int. Conf. Robotics and Biomimetics*, Sanya, China, Dec. 2007, pp. 285–290.

[68] N. Krahnstoever, T. Yu, S.-N. Lim, K. Patwardhan, and P. Tu, "Collaborative real-time control of active cameras in large scale surveillance systems," in *Proc. Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, Marseille, France, Oct. 2008, pp. 1–12.

[69] Y. Xu, D. Song, J. Yi, and F. van der Stappen, "An approximation algorithm for the least overlapping p-frame problem with non-partial coverage for networked robotic cameras," in *Proc. IEEE Int. Conf. Robotics and Automation*, Pasadena, CA, May 2008, pp. 1011–1016.

[70] Y. Xu and D. Song, "Systems and algorithms for autonomously simultaneous observation of multiple objects using robotic ptz cameras assisted by a wide-angle camera," in *Proc. IEEE/JRS Int. Conf. Intelligent Robots and Systems*, St. Louis, MO, Oct. 2009, pp. 3802–3807.

[71] J. W. Fitzpatrick, M. Lammertink, D. Luneau, T. W. Gallagher, B. R. Harrison, G. M. Sparling, K. V. Rosenberg, R. W. Rohrbaugh, E. C. H. Swarthout, P. H. Wrege, S. B. Swarthout, M. S. Dantzker, R. A. Charif, T. R. Barksdale, J. V. Remsen, S. D. Simon, and D. Zollner, "Ivory-billed woodpecker (campephilus principalis) persists in continental north america," *Science*, vol. 308, no. 5727, pp. 1460–1462, Jun. 2005.

[72] F. E. Hayes and W. K. Hayes, "The great ivory-billed woodpecker debate: Perceptions of the evidence," *Birding Magazine*, vol. 39, no. 2, pp. 36–41, Mar./Apr. 2007.

[73] L. W. Gysel and E. M. J. Davis, "A simple automatic photographic unit for wildlife research," *Journal of Wildlife Management*, vol. 20, pp. 451–453, 1956.

[74] T. E. Kucera and R. H. Barrett, "The trailmaster camera system for detecting wildlife," *Wildlife Society Bulletin*, vol. 21, pp. 505–508, 1993.

[75] K. Iida, R. Takahashi, Y. Tang, T. Mukai, and M. Sato, "Observation of marine animals using underwater acoustic camera," *Japanese Journal of Applied Physics*, vol. 45, no. 5B, pp. 4875–4881, May 2006.

[76] T. L. Cutler and D. E. Swann, "Using remote photography in wildlife ecology: a review," *Wildlife Society Bulletin*, vol. 27, no. 3, pp. 571–581, 1999.

[77] M. D. Sanders and R. F. Maloney, "Causes of mortality at nests of ground-nesting birds in the upper waitaki basin, south island, new zealand: a 5-year video study," *Biological Conservation*, vol. 106, no. 2, pp. 225–236, 2002.

[78] M. M. Stake and D. A. Cimprich, “Using video to monitor predation at black-capped vireo nests,” *BioOne*, vol. 105, no. 2, pp. 348–357, 2003.

[79] S. B. Lewis, P. DeSlmone, K. Titus, and M. R. Fuller, “A video surveillance system for monitoring raptor nests in a temperate rainforest environment,” *Northwest Science*, vol. 78, no. 1, pp. 70–74, 2004.

[80] D. Song, Q. Hu, N. Qin, and K. Goldberg, “Automating high resolution panoramic inspection and documentation of construction using a robotic camera,” in *Proc. IEEE Conf. Automation Science and Engineering*, Edmonton, Canada, Aug. 2005, pp. 172–177.

[81] N. Qin, D. Song, and K. Goldberg, “Aligning windows of live video from an imprecise pan-tilt-zoom robotic camera into a remote panoramic display,” in *Proc. IEEE Int. Conf. Robotics and Automation,*, Orlando, FL, May 2006, pp. 3429–3436.

[82] P. L. Rosin, “Thresholding for change detection,” in *Proc. 6th Int. Conf.Computer Vision*, Bombay, India, Jan. 1998, pp. 274–279.

[83] A. Lipton, H. Fujiyoshi, and R. Patil, “Moving target classification and tracking from real-time video,” in *Proc. 4th IEEE Workshop on Applications of Computer Vision*, Princeton, NJ, Oct. 1998, pp. 8–14.

[84] I. Cohen and G. Medioni, “Detecting and tracking moving objects for video surveillance,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Ft. Collins, CO, Jun. 1999, pp. 319–325.

[85] J. Barron, D. Fleet, and S. Beauchemin, “Performance of optical flow techniques,” *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 42–77, Feb. 1994.

[86] M. Kohle, d. Merkl, and J. Kastner, “Clinical gait analysis by neural networks: Issues and experiences,” in *Proc. IEEE Sympos. Computer-Based Medical Systems*, vol. 5, Maribor, Slovenia, Jun. 1997, pp. 138–134.

[87] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee, “Using adaptive tracking to classify and monitor activies in a site,” in *Proc. IEEE Sympos. Computer-Based Medical Systems*, vol. 5, Lubbock, TX, Jun. 1998, pp. 138–134.

[88] R. Cutler and L. S. Davis, “Robust real-time periodic motion detection, analysis, and applications,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 781–796, Aug. 2000.

[89] Y. Ran, I. Weiss, Q. Zheng, and L. S. Davis, “Pedestrian detection via periodic motion analysis,” *Int. J. Comput. Vis.*, vol. 71, no. 2, pp. 143–160, Feb. 2007.

[90] A. Briassouli and N. Ahuja, “Extraction and analysis of multiple periodic motions in video sequences,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 7, pp. 1244–1261, Jul. 2007.

[91] I. Laptev, S. J. Belongie, P. Pérez, and J. Wills, “Periodic motion detection and segmentation via approximate sequence alignment,” in *Proc. IEEE Int. Conf. Computer Vision*, Beijing, China, Oct. 2005, pp. 816–823.

[92] T. Burghardt and J. Calic, “Analysing animal behaviour in wildlife videos using face detection and tracking,” *IEE Proc. - Vision, Image, and Signal Processing*, vol. 153, no. 3, pp. 305–312, Jun. 2006.

[93] T. Sebastian, P. Klein, and B. Kimia, "Recognition of shapes by editing their shock graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 550–571, May 2004.

[94] M. Dunn, J. Billingsley, and N. Finch, "Machine vision classification of animals," in *Proc. IEEE Int. Conf. Mechatronics and Machine Vision in Practice*, Perth, Australia, Dec. 2003, pp. 157–163.

[95] D. Ramanan and D. Forsyth, "Using temporal coherence to build models of animals," in *Proc. IEEE Int. Conf. Computer Vision*, Beijing, China, Oct. 2003, pp. 338–345.

[96] J. Hayfron-Acquah, M. Nixon, and J. Carter, "Recognising human and animal movement by symmetry," in *Proc. Int. Conf. Image Processing*, Thessaloniki, Greece, Oct. 2001, pp. 290–293.

[97] U. D. Nadimpalli, R. R. Price, S. G. Hall, and P. Bomma, "A comparison of image processing techniques for bird recognition," *Biotechnology Progress*, vol. 22, no. 1, pp. 9–13, Jan. 2006.

[98] Y. Liu, J. Zhang, D. Tjondronegoro, and S. Geve, "A shape ontology framework for bird classification," in *Proc. 9th Biennial Conf. Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications*, Glenelg, Australia., Dec. 2007, pp. 478–484.

[99] E. Ribnick, S. Atev, and N. P. Papanikolopoulos, "Estimating 3d positions and velocities of projectiles from monocular views," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 938–944, May 2009.

[100] A. Saxena, M. Sun, and A. Ng, "Make3d: Depth perception from a single still image," in *Proc. AAAI Conf. Artificial Intelligence*, Chicago, IL, Jul. 2008, pp. 1571–1576.

[101] D. Hoiem, A. A. Efros, and M. Hebert, "Recovering surface layout from an image," *Int. J. Comput. Vis.*, vol. 75, no. 1, pp. 151–172, Oct. 2007.

[102] B. Han, Y. Zhu, D. Comaniciu, and L. Davis, "Kernel-based bayesian filtering for object tracking," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Diego, CA, Jun. 2005, pp. 227–234.

[103] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *Int. J. Comput. Vis.*, vol. 29, no. 1, pp. 5–28, Aug. 1998.

[104] C. Chang, R. Ansari, and A. Khokhar, "Multiple object tracking with kernel particle filter," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, San Diego, CA, Jun. 2005, pp. 566–573.

[105] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *Int. J. Robot. Res.*, vol. 23, no. 7-8, pp. 693–716, Aug. 2004.

[106] H. Nguyen and A. Smeulders, "Fast occluded object tracking by a robust appearance filter," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1099–1104, Aug. 2004.

[107] C. Yang, R. Duraiswami, and L. Davis, "Fast multiple object tracking via a hierarchical particle filter," in *Proc. IEEE Int. Conf. Computer Vision*, Beijing, China, Oct. 2005, pp. 212–219.

[108] G. L. Foresti, "Object detection and tracking in time-varying and badly illuminated outdoor environments," *Opt. Eng.*, vol. 37, no. 9, pp. 2550–2564, Sep. 1998.

[109] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. New York, NY: MIT Press, 2005.

[110] H. Veeraraghavan, P. Schrater, and N. Papanikolopoulos, "Switching kalman filter-based approach for tracking and event detection at traffic intersections," in *Proc. 13th Mediterranean Conf. Control and Automation*, Limassol, Cyprus, Jun. 2005, pp. 1167–1172.

[111] F. Xu, X. Liu, and K. Fujimura, "Pedestrian detection and tracking with night vision," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 1, pp. 63–71, Mar. 2005.

[112] W. Abd-Almageed, M. Fadali, and G. Bebis, "A non-intrusive kalman filter-based tracker for pursuit eye movement," in *Proc. American Control Conf.*, Anchorage, AK, May 2002, pp. 1443–1447.

[113] D. Song, N. Qin, Y. Xu, C. Y. Kim, D. Luneau, and K. Goldberg, "System and algorithms for an autonomous observatory assisting the search for the ivory-billed woodpecker," in *Proc. IEEE Int. Conf. Automation Science and Engineering*, Washington D.C., Aug. 2008, pp. 200–205.

[114] M. Boutayeb, H. Rafaralahy, and M. Darouach, "Convergence analysis of the extended kalman filter used as an observer for nonlinear deterministic discrete-time systems," *IEEE Trans. Autom. Control*, vol. 42, no. 4, pp. 581–586, Apr. 1997.

[115] M. Bazaraa, H. shelrali, and C. Shetty, *Nonlinear Programming: Theory and Algorithms*. New York, NY: John Wiley and Sons, Inc, 1993.

[116] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, "Procedures for optimization problems with a mixture of bounds and general linear constraints," *ACM Trans. Mathematical Software*, vol. 10, no. 3, pp. 282–298, Sep. 1984.

VITA

Mr. Yiliang Xu received his B.S. in electrical engineering from Zhejiang University, China, in 2002. He earned his first Ph.D. in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2006. He earned his second Ph.D. in computer science from Texas A&M University in 2011. His research interests include robotics, computer vision, and artificial intelligence.

Mr. Xu may be reached at the Department of Computer Science and Engineering, Texas A&M University, TAMU 3112, College Station, TX 77843-3112. His email is yiliang_xu@ieee.org.