Exact Algorithms for Non-Overlapping 2-Frame Problem with Non-Partial Coverage for Networked Robotic Cameras

Yiliang Xu, Dezhen Song, and Jingang Yi

***Abstract*— We report our algorithmic development on the 2-frame problem that addresses the need of coordinating two networked robotic pan-tilt-zoom (PTZ) cameras for n, $(n > 2)$, competing rectangular observation requests. We assume the two camera frames have no overlap on their coverage. A request is satisfied only if it is fully covered by a camera frame. The satisfaction level for a given request is quantified by comparing its desirable observation resolution with that of the camera frame which fully covers it. We propose a series of exact algorithms for the solution that maximizes the overall 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 camera resolution levels, respectively. We have implemented all the algorithms and compared them with the existing work.**

I. Introduction

Networked Robotic pan-tilt-zoom (PTZ) cameras can cover a large region of remote scene with high resolution without requiring excessive network communication bandwidth. Consider a set of p $(p \geq 2)$ networked PTZ cameras in a large shopping mall for public surveillance, or in a deep forest for natural observation. There are n different rectangular observation requests proposed by multiple online users or initiated by *in situ* sensors. With usually much more competing requests than PTZ cameras available, an optimal set of p PTZ camera frames that best satisfies the requests needs to be computed. This is formulated as the p-frame problem in our previous work [1]. Fig. 1 illustrates a 2-frame problem instance.

II. Related Work

The p-frame problem relates to the p-center problem 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 [2]. Eppstein [3] proposes an $O(n \log^2 n)$ algorithm for the Euclidean 2-center problem. The rectilinear p-center problem is also NP-hard [2]. Bespamyatnikh and Kirkpatrick [4] propose a linear time algorithm for the rectilinear 2-center problem. 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.

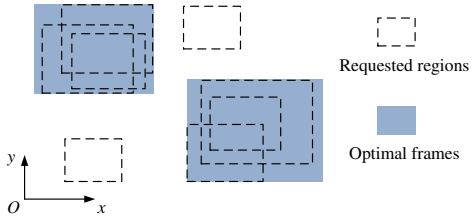

Fig. 1. An illustration of the non-overlapping 2-frame problem.

The p-frame problem can be applied to multiple camera surveillance systems, especially those with multiple active cameras. Fiore et al. [5] propose a dual-camera system with a wide-angle static camera and a PTZ camera for pedestrian surveillance. 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. El-Alfy et al. [6] model the subject-camera assignment issue for a PTZ camera network as a maximum matching problem in a bipartite graph and group the subjects using heuristics when there are less camera than subjects. 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.

Our group has been researching on developing intelligent vision systems and algorithms using robotic cameras for a variety of applications [7]. In [1], 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 [8]. 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. In this paper, we extend the single frame selection algorithm in [9] to the cases where $p = 2$ and propose a series of exact algorithms for the 2-frame problem with different camera configurations.

III. Problem Definition

A. Input and Output

As illustrated in Fig. 1, we assume all camera frames and requests are rectangular and each side of the rectangle is axis-

This work was supported in part by the National Science Foundation under IIS-0643298 and MRI-0923203.

Y. Xu and D. Song are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843. Email: ylxu@cse.tamu.edu and dzsong@cse.tamu.edu.

J. Yi is with MAE Department, Rutgers University, Piscataway, NJ 08854, USA. Email: jgyi@rutgers.edu.

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

B. 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 feasible solution (c_1, c_2) satisfies the Non-Overlapping Condition (NOC), i.e., the coverage regions of c_1 and c_2 do not overlap. 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.

C. Satisfaction Metric and Problem Formulation

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

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

where

$$I(c, r_i) = \begin{cases} 1 & \text{if} \quad r_i \subseteq c, \\ 0 & \text{otherwise}, \end{cases} \tag{2}$$

where we abuse the set operator $\subseteq$ to represent the 2D regional relationship between frame(s) and request(s) in the rest of this paper. Here, $r_i \subseteq c$ means that the region of r_i is fully contained in that of c.

The metric in (1) is an extension of the Resolution Ratio with Non-Partial Coverage (RRNPC) metric as in [1]. The indicator function in (2) considers the camera coverage over requests. It implies that the request is not satisfied if it is not or partially covered by a frame. The second term in (1) compares the resolution levels of camera frames and requests. The resolution ratio is truncated by 1 since finer camera resolution (smaller z) than desirable (z_i) does not increase the satisfaction level. This way, the satisfaction metric in (1) represents a coverage-resolution tradeoff. With the NOC assumption, we know $s(c_1, c_2) = s(c_1) + s(c_2)$.

Eq. (1) shows that the satisfaction of any candidate (c_1, c_2) can be computed in $O(n)$ time. Now we formulate the non-overlapping 2-frame problem as a maximization problem, $(c_1^*, c_2^*) = \arg\max_{(c_1, c_2) \in \mathbb{C}^2} s(c_1, c_2)$.

IV. Algorithms

A. Feasibility condition

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

Definition 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,

$$\begin{aligned} S_e^X(c_1, c_2) =& S_e^X(x_1 + \frac{4z_1}{2}, x_2 - \frac{4z_2}{2}) \\ & \cup S_e^X(x_2 + \frac{4z_2}{2}, x_1 - \frac{4z_1}{2}), \end{aligned} \tag{3}$$

$$\begin{aligned} S_e^Y(c_1, c_2) =& S_e^Y(y_1 + \frac{3z_2}{2}, y_2 - \frac{3z_2}{2}) \\ & \cup S_e^Y(y_2 + \frac{3z_1}{2}, y_1 - \frac{3z_1}{2}), \end{aligned} \tag{4}$$

as illustrated in Fig. 2. Intuitively, (3) and (4) define the "gap" between frames.

Lemma 1 (Feasibility condition): Given any feasible solution (c_1, c_2), it must have at least one non-empty separation as defined in (3) and (4), $S_e^X(c_1, c_2) \cup S_e^Y(c_1, c_2) \neq \phi$.

Lemma 1 is straightforward from the NOC. 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 paper. 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. 2 for the x-separable problem. Hence, (3) is simplified as, $S_e^X(c_1, c_2) = S_e^X(x_1 + \frac{4z_1}{2}, x_2 - \frac{4z_2}{2})$.

B. Optimality condition

Lemma 1 defines the necessary condition for any feasible solution. Unfortunately, there are infinite number of separa-

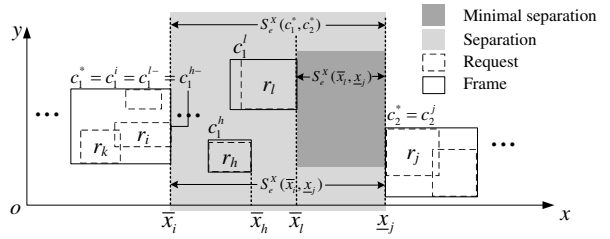

Fig. 2. An illustration of the optimal x-separable solution. At least one optimal solution $(c_1^*, c_2^*) = (c_1^i, c_2^j)$ corresponds to a separation $S_e^X(\overline{x}_i, \underline{x}_j)$. $S_e^X(\overline{x}_l, \underline{x}_j)$ is a minimal separation. r_l is the closest request on the left hand 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 (12).

tions. Next, we show how to reduce the problem to finite candidate separations to search for the optimal solution.

Given the optimal solution (c_1^*, c_2^*) as illustrated in Fig. 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., $x_1^* - \frac{4z_1^*}{2} = \underline{x}_k$), 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., $x_1^* + \frac{4z_1^*}{2} < \overline{x}_i$, 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 2 (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, $r_i \subseteq c_1'$ with $x_1' + \frac{4z_1'}{2} = \overline{x}_i$, and $r_j \subseteq c_2'$ with $x_2' - \frac{4z_j'}{2} = \underline{x}_j$. 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. 2, we have,

$$s(c_2^*) = \sum_{k=1}^{n} I(c_2^*, r_k) \min\left(\frac{z_k}{z_2^*}, 1\right). \tag{5}$$

Let R_2^* represent the set of requests which are fully enclosed by c_2^*. Then (5) is re-written as,

$$s(c_2^*) = \sum_{r_k \in R_2^*} I(c_2^*, r_k) \min\left(\frac{z_k}{z_2^*}, 1\right) = \sum_{r_k \in R_2^*} \min\left(\frac{z_k}{z_2^*}, 1\right). \tag{6}$$

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 $x_2' - \frac{4z_2'}{2} = x_2' - \frac{4z_2^*}{2} = \underline{x}_j$. Intuitively, c_2' is the frame similar to c_2^* except that its left side overlaps with line $x = \underline{x}_j$. Let R_2' be the set of requests that are completely enclosed by c_2', then

$$s(c_2') = \sum_{r_k \in R_2'} I(c_2', r_k) \min\left(\frac{z_k}{z_2^*}, 1\right) = \sum_{r_k \in R_2'} \min\left(\frac{z_k}{z_2^*}, 1\right). \tag{7}$$

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

$$\underline{x}_k \geq \underline{x}_j = x_2' - \frac{4z_2'}{2},\ \overline{x}_k \leq x_2^* + \frac{4z_2^*}{2} \leq x_2' + \frac{4z_2'}{2},$$

$$\underline{y}_k \geq y_2^* - \frac{3z_2^*}{2} = y_2' - \frac{3z_2'}{2},\ \overline{y}_k \leq y_2^* + \frac{3z_2^*}{2} = y_2' + \frac{3z_2'}{2}.$$

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

Comparing (6) and (7), 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 $x_1' + \frac{4z_1'}{2} = \overline{x}_i$ 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 2 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.

C. Exhaustive search

Based on Lemma 2, 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 overlapping with one boundary of the separation. We define these two constrained optimal frames,

$$c_1^i = \arg\max_{c=(x,y,z)} s(c),\ \text{s.t. } r_i \subseteq c \text{ and } x + \frac{4z}{2} = \overline{x}_i, \tag{8}$$

$$c_2^j = \arg\max_{c=(x,y,z)} s(c),\ \text{s.t. } r_j \subseteq c \text{ and } x - \frac{4z}{2} = \underline{x}_j. \tag{9}$$

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 (8) and (9), respectively. Algorithm 1 summarizes the exhaustive search approach.

Algorithm 1: Exhaustive Search Algorithm for x-Separable Non-Overlapping 2-Frame Problem (ES-XS-2)

Input: Request set R.
Output: (c_1^*, c_2^*)
1 **foreach** $S_e^X(\overline{x}_i, \underline{x}_j)$ $O(n^2)$
2 **do**
3 **if** $\overline{x}_i \leq \underline{x}_j$ **then**
4 Compute c_1^i as in (8); $T_1(n)$
5 Compute c_2^j as in (9); $T_1(n)$
6 **end**
7 **end**
8 **return** the best (c_1^i, c_2^j) pair; $O(1)$

It is noticed that in lines 4 and 5 of Algorithm 1, it requires the subroutines that solve the two sub-problems as in (8) and (9), 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 1 runs in $O(n^3) + O(n^2) \cdot T(n)$ time.

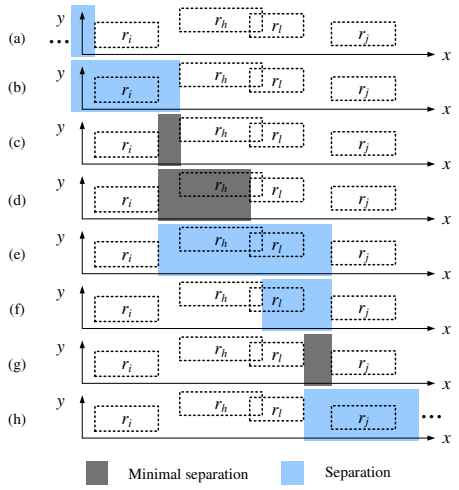

Fig. 3. An illustration of the sweeping of separation boundaries. During sweeping from left to right, if the separation is not a minimal separation, we contract the separation by moving its left boundary to its next candidate position and the optimal frame on its left hand side is computed as in (12). If the separation is a minimal separation, its right frame is computed as in (9), and forms a candidate solution with the optimal left frame maintained earlier.

D. 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. 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. 2). We define this separation with smallest non-negative width as the minimal separation.

Definition 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 $x = \underline{x}_j$, i.e., $l = \arg\min_{k\in P}(\underline{x}_j - \overline{x}_k)$ s.t. $\overline{x}_k \le \underline{x}_j$.

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. 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,

$$c_1^{l-} = \arg \max_{c_1^k,\ k\in P} s(c_1^k),\ \text{s.t. } x_1^k + \frac{4z_1^k}{2} \le \overline{x}_l. \tag{10}$$

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 (10) requires to compute $O(n)$ constrained optimal single frames as in (8) 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. 2,

$$h = \arg\min_{k\in P}(\underline{x}_j - \overline{x}_k),\quad \text{s.t. } \overline{x}_k \le \overline{x}_l \le \underline{x}_j. \tag{11}$$

Then the computations of c_1^{h-} and c_1^{l-} based on (10) only differ in computing $s(c_1^l)$. Therefore, we have,

$$c_1^{l-} = \begin{cases} c_1^{h-} & \text{if } s(c_1^{h-}) > s(c_1^l), \\ c_1^l & \text{otherwise.} \end{cases} \tag{12}$$

Eqs. (11) and (12) 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 (12).

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. 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 (12). Figs. 3(f) to 3(g) illustrate these operations.
- If the separation is a minimal separation. We compute the optimal frame on the right hand side of the separa-

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

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

```
Sort left sides of R : B = [b[1], ..., b[n]];                O(n log n)
Sort right sides of R : B̄ = [b̄[1], ..., b̄[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^- = φ; c_1^* = φ; c_2^* = φ;                              O(1)
u = 0; v = 1;                                                 O(1)
while v < n                                                   O(n)
do
    if b̄[u+1] > b[v]                          #Minimal separation
    then
        Find b[v] belongs to r_j;                             O(1)
        Compute c_2^j as in (9)                               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 b̄[u] belongs to r_l;                             O(1)
        Compute c_1^l as in (8);                              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)
```

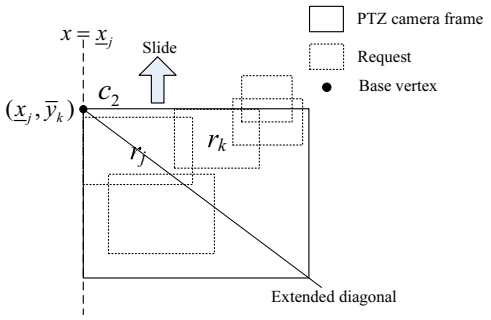

Fig. 4. An illustration of finding c_2^j as in (9) with fixed resolution. Slide the candidate frame c_2 along line $x = \underline{x}_j$ from a initial position. Whenever a horizontal frame side aligns with that of a request, the change in $s(c_2)$ can be computed in $O(1)$ time.

tion as in (9). 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 the right boundary to its next candidate position and a new sweeping step starts. The expansion from Fig. 3(d) to Fig. 3(e) illustrates these operations.

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

E. Algorithm complexity with different camera resolution configurations

We turn to the the subroutines for solving the subproblems as in (8) and (9), under different camera resolution configurations. Without loss of 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 (9).

1) 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. Recall c_2^j satisfies $x_2^j - \frac{4z_2^j}{2} = \underline{x}_j$ 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 metric in (1), 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 (1), 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 $y_2 + \frac{3z_2}{2} = \overline{y}_j$) 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 3, which runs in $O(n)$. This means when the cameras have a fixed resolution, $T_1 = O(n)$ and Algorithm 2 runs in $O(n^2)$ time.

Algorithm 3: Subroutine solving (9) with a fixed resolution

Input: Right separation boundary at $x = \underline{x}_j$;
Output: c_2^j;

```
Create candidate frame c_2;                                   O(1)
Set x_2 − 4z_2/2 = x_j, y_2 + 3z_2/2 = y_j;                   O(1)
Calculate s(c_2);                                             O(n)
while y_2 − 3z_2/2 < y_j                                      O(n)
do
    Slide c_2 upward along line x = 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)
```

2) 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 3 m times, each time for one resolution level, respectively. Therefore, when the cameras have m discrete resolution levels, Algorithm 2 runs in $O(n^2m)$ time.

3) 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 $x_2^j - \frac{4z_2^j}{2} = \underline{x}_j$. As shown in Fig. 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 [9]. According to the optimality condition in Lemma 2 of [9], one optimal frame c_2^j must have one corner coincident with a BV. Song et al. [9] 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 [9] 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 2 runs in $O(n^3)$ time.

Theorem 1: When cameras have a fixed, m discrete and continuous zoom level(s), Algorithm 2 runs in $O(n^2)$, $O(n^2m)$ and $O(n^3)$ times, respectively.

V. 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.

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

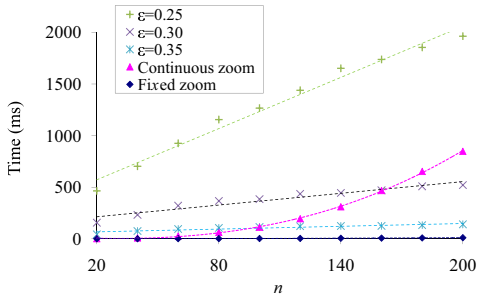

Fig. 5. Computation speed of algorithms with a fixed and continuous zoom level(s), respectively, and the comparison with the approximation algorithm in [1] with approximation bound $\epsilon = 0.35$, 0.30 and 0.25, respectively.

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. 5 illustrates the relationship between computation time and n for proposed algorithm with a fixed and continuous zoom level(s), respectively. It is shown 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 [1], which run in $O(n/\epsilon^3 + p^2/\epsilon^6)$ time, where ϵ is the approximation bound. We test the approximation algorithm with $\epsilon = 0.35$, 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 tracking and surveillance systems [10], [11] can handle much less than 100 objects at the same time while accurate object tracking/observation is required, which qualifies the proposed algorithm as a viable solution for these applications.

Fig. 6 shows two sample outputs of the algorithm with

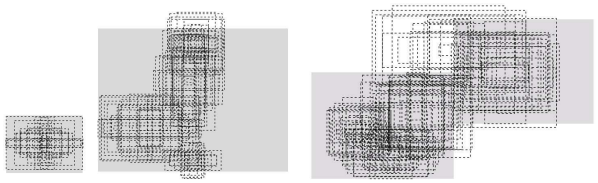

Fig. 6. Sample simulation results for random input. Dashed-line rectangles denote requests and grey rectangles are optimal frames. $n = 100$, $s_d = 5$.

continuous zoom and $n = 100$. In both cases, our algorithm reasonably locates 2 frames to cover most of the requests.

VI. Conclusion

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

Acknowledgement

We would like to thank N. Amato, J, Chen, F. van der Stappen, N. Papanikolopoulos, Z. Bing, R. Volz, and K. Goldberg for their insightful inputs, C. Kim, J. Zhang, A. Aghamohammadi, and W. Li, for their contributions to the Networked Robot Lab at Texas A&M University.

References

[1] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, May 2008, pp. 1011–1016.

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

[3] D. Eppstein, "Fast construciton of planar two-centers," in *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, January 1997, pp. 131–138.

[4] S. Bespamyatnikh and D. Kirkpatrick, "Rectilinear 2-center problems," in *Proceedings of the 11th Canadian Conference on Computational Geometry*, 1999, pp. 68–71.

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

[6] H. El-Alfy, D. Jacobs, and L. Davis, "Assigning Cameras to Subjects in Video Surveillance Systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 2009, pp. 837–843.

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

[8] Y. Xu and D. Song, "Systems and algorithms for autonomous and scalable crowd surveillance using robotic PTZ cameras assisted by a wide-angle camera," *Autonomous Robots*, vol. 29, no. 1, pp. 53–66, 2010.

[9] D. Song, A. F. van der Stappen, and K. Goldberg, "Exact algorithms for single frame selection on multi-axis satellites," *IEEE Transactions on Automation Science and Engineering*, vol. 3, no. 1, pp. 16–28, January 2006.

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

[11] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, 2006.