Received 9 February 2023, accepted 22 February 2023, date of publication 27 February 2023, date of current version 2 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3249190

RESEARCH ARTICLE

Group Tracking for Video Monitoring Systems: A Spatio-Temporal Query Processing Approach

HYUNSIK YOON, DALSU CHOI, AND YON DOHN CHUNG, (Member, IEEE)
Department of Computer Science and Engineering, Korea University, Seoul 02841, Republic of Korea

Corresponding author: Yon Dohn Chung (ydchung@korea.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) funded by the Korean Government [Ministry of Science and ICT (MIST)] under Grant NRF-2020R1A2C2013286, in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2021R1A6A1A13044830, and in part by the MSIT under the Information and Communications Technology (ICT) Creative Consilience Program Supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP) under Grant IITP-2023-2020-0-01819.

ABSTRACT Recently, many video monitoring systems utilize deep learning technologies to recognize locations and trajectories of people in video data. In video monitoring systems, a fast discovery of human groups is an important task for several applications, for example, crime surveillance, contact tracing, and customer behavior analysis. To tackle the demand, we propose a group tracking method. First, we propose a spatial proximity definition and define a novel query type, a *group tracking query* that considers characteristics of video data. A group tracking query retrieves the groups that travel for more than a certain amount of video frame within a certain distance. We propose an efficient query processing method that exploits the spatio-temporal characteristics of groups. Through extensive experiments using real-world datasets, we verify the efficiency and effectiveness of our query definition and query processing method.

INDEX TERMS Spatio-temporal query processing, spatial data management, spatial databases, video query processing, video monitoring systems.

I. INTRODUCTION

With the recent advances in deep learning, various information can be extracted from an image. Especially, deep learning models can detect locations for objects (i.e., object detection) and also track object trajectories (i.e., object tracking). Even in a video that contains a lot of people, figuring out the trajectory of people becomes possible based on the deep learning techniques. Such object tracking data (crowd tracking data) can be produced quickly with a pretrained deep-learning model [1]. A model proposed in the work [1] produces crowd tracking data containing tens of people in a frame at the rate of 27-30 frames per second (fps). Accordingly, many video monitoring systems utilize object detection and tracking on a video that contains a lot of people (crowded video). For example, Briefcam [2] and Viisights [3] provide video analytics services for crowded video (e.g., surveillance systems for smart cities, safety and security management for crowded stadiums, etc.).

In a video monitoring system for crowds, the fast discovery of human groups is important. The example scenarios are as follows.

Scenario (a) [Crime surveillance]. An investigator has information that the three suspects of a crime move together. Accordingly, the investigator wants to find groups as soon as possible from various video sources (CCTVs). The investigator can arrest the suspects by utilizing the found groups.

Scenario (b) [Contact tracing]. During the recent COVID-19 pandemic, finding out close-contacts who contact to a super-spreader is important for early quarantine. When a person *A* contacts an infected person *B* longer than 15 minutes, *A* is classified as a close-contact who is likely to be infected [4]. In this case, close-contacts can be identified by finding groups that include the super-spreader.

Scenario (c) [Customer behavior analysis]. In commercial facilities that a lot of people visit, analyzing the behaviors

The associate editor coordinating the review of this manuscript and approving it for publication was Junchi Yan.

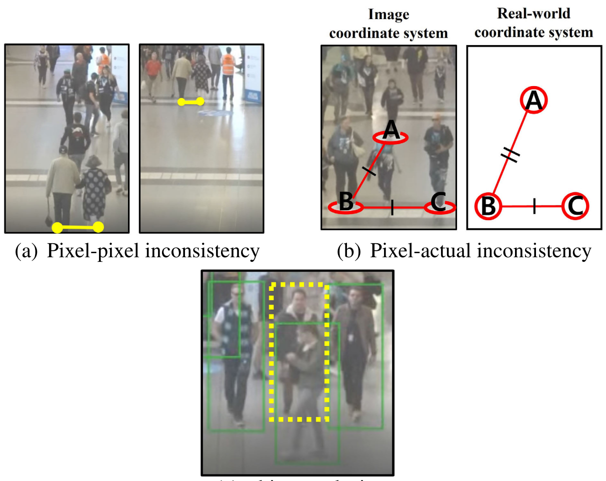


Image coordinate system
Real-world coordinate system
A
B
C
A
B
C
(a) Pixel-pixel inconsistency
(b) Pixel-actual inconsistency
(c) object occlusion

FIGURE 1. **Challenges in finding groups over crowd tracking data: In Figure 1(a), pixel distances between two objects vary according to the object locations while actual distances are fixed. In Figure 1(b), although the pixel distances between object A and B, and B and C are the same, the corresponding actual distances are different. In Figure 1(c), although the object bounded with a dashed box exists in the image, it is not detected by a machine learning model.**

of customers in groups is required to establish marketing and business strategies (e.g., determination of the locations for services or commodities). To analyze the customer groups, finding out the groups should be preceded.

There have been a lot of researches related to the above tasks, which are classified into two categories: (1) Learning-based approach and (2) spatio-temporal query processing approach. Although machine learning has recently developed very fast and solved a lot of problems in practice, it has inherent limitations as follows in solving the target application Scenarios (a)-(c). Learning-based methods need the ground truths of not only an individual human trajectory but also human groups in a video. Since the ground truths for the human and human groups need to be labeled manually, learning-based methods are not realistic in Scenarios (a)-(c). Even if we train a model of learning-based methods with manually labeled data, the model takes longer inference time than the running time of a video [5]. Besides inference time, a model also needs training time. Therefore, learning-based methods cannot be applied to practical on-line video monitoring systems where the fast discovery of groups is important. Because of these limitations, we try to discover the groups by a spatio-temporal query processing approach. With the approach, the query results can also be utilized as training data for the learning-based methods. However, the traditional spatio-temporal query processing approach has failed to support our target applications in the following aspects.

Challenge 1 [Perspective projection]. Video data is a projection of the three-dimensional real-world into a two-dimensional space (image), called perspective projection. In this environment, capturing spatial proximity between two objects and temporal consistencies of the proximity are challenging due to the following reasons. First, for two objects, even if a distance in the real-world (actual distance) does not change, the projected distance in an image plane (pixel distance) in the video can be changed, or vice versa. In Figure 1, pixel distances between two objects vary according to the object locations while actual distances are fixed. We call this *pixel-pixel inconsistency*. In Figure 1(b), although the pixel distances are the same, the corresponding actual distances are different, called *pixel-actual inconsistency*.

Challenge 2 [Object occlusion]. Object occlusion is a phenomenon that an object disappears in crowd tracking data while the object still exists in an image. In Figure 1(c), although the object bounded with a dashed box exists in the image, it is not detected by a machine learning model. A main cause of occlusion is the overlapping of objects in video data. Occlusion of an object may cause occlusion of groups.

In this paper, we propose a novel type of query, *group tracking query* and an efficient group tracking query processing method for video monitoring systems. Our query targets crowd tracking data from cameras which have fixed angles and locations (e.g., CCTV). The query processing method consists of two steps for each frame: (1) group detection and (2) group track maintenance. In a group detection step, we preprocess the crowd tracking data and find groups based on our group definition. In a group track maintenance step, we capture temporal consistency of the groups and return the query answers frame by frame. Our contributions are summarized as follows.

- We propose a novel spatial proximity definition that takes perspective projection into account.
- We propose a novel group tracking query for crowd tracking data.
- We propose an efficient algorithm with a pruning strategy considering proximity conditions for a group detection step.
- We propose an efficient method for capturing temporal consistency of groups for a group track maintenance step.
- We conduct extensive experiments to show efficiency and effectiveness of the proposed methods.

The rest of paper is organized as follows. In Section II, we introduce related works. In Section III, we describe terms and problem definitions including proximity definition and a group. In Section IV, we introduce a process to determine proximity considering perspective projection. In Section V, we propose a maximal group detection algorithm with pruning strategies. Section VI introduces an efficient method for the group track maintenance step. Section VII shows a workflow of group tracking query processing. We present experimental evaluations in Section VIII, and conclude the paper in Section IX.

II. RELATED WORK

A. LEARNING-BASED GROUP TRACKING

In the computer vision research area, several learning-based methods that find groups over crowd tracking data have been proposed [1], [5], [6], [7], [8], [9], [10], [11], [12]. Among them, the work [12] achieved the highest accuracy by exploiting the general moving patterns of a group. However, the inference time of its model was much longer than the total video running time. Accordingly, the work [5] proposed a fast group tracking method while achieving comparable accuracy with the work [12]. However, learning-based methods are still not appropriate for video monitoring systems for the following reasons. First, in our scenarios (e.g., Scenarios (a)-(c)), the fast discovery of human groups is important. However, the inference time of the model in the work [5] is still from 1 to 4.5 times longer than the total running time of a video. Second, to train a model, manually labeled ground truths are needed for both individual objects and groups. Moreover, to consider perspective projection in crowd tracking data, the ground truths should be mapped to a Euclidean-spaced real-world coordinate system via homographic transformation, which is not practical in video monitoring systems.

B. QUERY PROCESSING-BASED GROUP TRACKING

In the spatio-temporal database research area, several query processing-based methods that track groups of objects have been proposed [13], [14], [15], [16]. Their target data is a series of coordinates of moving objects (e.g., a series of coordinates of GPS devices). However, since they defined spatial proximities for groups in Euclidean spaces, perspective projection was not considered. To capture the temporal consistency of groups, the works [13], [14], [15], and [16] used the *clustering-and-intersection* algorithm. We described the algorithm in Section VI.

C. PERSPECTIVE PROJECTION AND OCCLUSION HANDLING

In existing works, there are two ways to capture spatial proximity between two objects with taking perspective projection into account. One is mapping of an image plane to a Euclidean spaced, real-world plane via homographic transformation [5], [12]. However, the mapping needs to know the mapping parameters between the two planes. Because a video monitoring system may include a lot of cameras that take different scenes, it is not practical to obtain every mapping parameter for the cameras in the video monitoring system. The other is exploiting the linear relationship between the length of a line segment and its y coordinate [17]. With the linear relationship, variation of a distance threshold becomes possible. This approach can handle pixel-pixel inconsistency in Figure 1(a). However, pixel-actual inconsistency in Figure 1(b) cannot be handled.

Perspective projection also causes object occlusion which affects temporal consistency of groups in crowd tracking data. There was a research that considers object occlusion when checking temporal consistency of groups [18]. They proposed an evaluation method for a temporal query over video tracking data. An example of a temporal query is as follows:

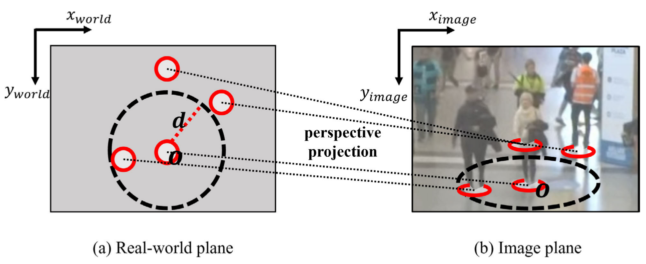


FIGURE 2. An example for perspective projection of a circle. When a circle (in Figure 2a) is observed by a camera that has a fixed position, it is projected into an ellipse as in Figure 2b.

'search all the video segments with the length of 10 seconds in which two people A and B appear continuously'. For the temporal query processing, they use the modified clustering-and-intersection algorithm to be tolerant to object occlusion. However, their method considers all the objects in a frame as a single group.

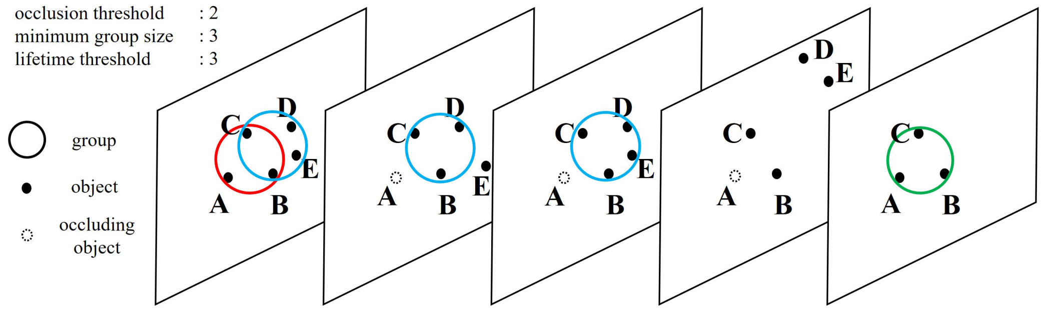
III. TERMS AND PROBLEM DEFINITIONS

A. INPUT DEFINITION

A frame of a video is an image, and a coordinate system for a frame is defined by a horizontal x-axis and a vertical y-axis based on the origin at the left-top corner of the frame. The coordinate system of a video frame is a projection of the Euclidean-spaced and real-world coordinate system. A track ot_i of a human o_i with a unique id i is defined as $\{d_i^{f_{start}}, \ldots, d_i^{f_{end}}\}(0 \leq start \leq end \leq N-1$, the integer N is the number of frames in a video), where a detection $d_i^{f_n} = (x, y, width, height)$ represents the coordinates and size of the unique rectangular bounding box for o_i at Frame f_n. We use $(x + width/2, y + height)$ as the coordinate of o_i. Because of object occlusion, a missing detection may exist in ot_i. Given a video $V = \{f_0, \ldots, f_{N-1}\}$, which is a sequence of frames, crowd tracking data of V is a set T of tracks $\{ot_0, \ldots, ot_{M-1}\}$ for M humans that appear in V. We denote a human as an object for the ease of description.

B. PROXIMITY DEFINITION WITH VARIABLE ELLIPSE

The pixel distance inconsistencies in Figure 1(a) and 1(b) make a fixed pixel distance threshold useless in crowd tracking data. Therefore, we use the concept of proximal boundary for an object. In the real-world plane, as shown in Figure 2(a), a circle centered object o with radius d contains all the spatial points within actual distance d from o. When we observe the circle from a camera that has a fixed position (e.g., CCTV), as shown in Figure 2(b), it is projected into an ellipse. This property is well-known and widely used in camera calibrations [19], [20]. However, finding the shape of the ellipse exactly matched with the circle in the real-world plane is also a homographic transformation which is not practical in a video monitoring system. Accordingly, we approximate ellipse-shaped proximal boundaries based on a user input for a sample object, which will be described in Section IV.



	Frame 0	Frame 1	Frame 2	Frame 3	Frame 4
$\{A, B, C\}$	appears		disappears	-	-
$\{B, C, D\}$	appears	appears	appears		disappears
$\{B, C, D, E\}$	appears		appears		disappears
$\{A, B, C\}'$	-	-	-	-	appears

FIGURE 3. An example for a group tracking query.

*Definition 1 (**Object Proximity**): Given two objects o_i and o_j and two corresponding ellipses e_{o_i} and e_{o_j} centered the two objects, o_i and o_j are proximal when o_j lies inside e_{o_i} and o_i lies inside e_{o_j}.*

C. GROUPS AND GROUP TRACKS

*Definition 2 (**Group and Maximal Group (MG)**): Given a set O of objects $\{o_0, \dots o_k\}$ and the minimum group size n, a **group** g represents a set of objects, such that (1) $g \subset O$, (2) $|g| \geq n$, and (3) $\forall o_i, o_j \in g$, o_i and o_j are proximal. Given a set G of groups, we say that a group mg is **maximal** among G, such that $mg \in G$, $\forall g_i \in G - \{mg\}$, $mg \not\subset g_i$.*

*Definition 3 (**Group Appearance, Occlusion, and Disappearance**): Given a set G of groups at Frame f_m, we say that a group $g_i \in G$ **appears** at Frame f_m. After the first appearance of a group g_i, when g_i does not appear at Frame f_n, we say that g_i **occludes** at f_n. There are two situations that occlusion of g_i occurs: (1) an object in g_i occludes and (2) an object in g_i becomes not proximal to $\exists g_j \in G$. Given an integer occlusion threshold $thres_{occ}$ (≥ 0) and Frame f_n, if g_i occludes in the recent $thres_{occ}$ frames ($\{f_{n-thres_{occ}+1}, \dots, f_n\}$), we say that g_i **disappears** at Frame f_n. When the members of g_i forms a group again after the disappearance, the group is regarded as a new group g_j.*

Figure 3 depicts five consecutive frames of crowd tracking data. We assume that the occlusion threshold is 2, the minimum group size is 2, and the lifetime threshold is 3. At Frame 0, the groups $\{A, B, C\}$, $\{B, C, D\}$, and $\{B, C, D, E\}$ appear. Therefore, *MG* among the three groups are $\{A, B, C\}$ and $\{B, C, D, E\}$ at Frame 0. At Frame 1, $\{A, B, C\}$ occludes due to occlusion of object A, and $\{B, C, D, E\}$ occludes because E becomes not proximal to the other objects in $\{B, C, D, E\}$. At Frame 2 $\{A, B, C\}$ disappears because it occludes during the occlusion threshold ($= 2$).

*Definition 4 (**Group Track and Maximal Group Track**):* A **group track (GTR)** $t_g = (f_{start}, f_{end})$ $(0 \leq start \leq end \leq N-1)$ is a tuple that consists of the first appearance frame (start frame), f_{start} and the last appearance frame (end frame), f_{end} of a group g. The lifetime of g is defined as $f_{end} - f_{start} + 1$. t_g should be expired at the disappearance of g. According to the definition of group disappearance, in frame interval $[f_{start}, f_{end}]$, a sequence of occluding frames of which length is less than the occlusion threshold may exist. Given a set T of *GTR*s, a *GTR* $t_{g_i} \in T$ is a **maximal group track (MGTR)** when there is no group track $t_{g_j} \in T$ that satisfies $g_i \subset g_j$, $t_{g_j}.f_{start} \leq t_{g_i}.f_{start}$ and $t_{g_i}.f_{end} \leq t_{g_j}.f_{end}$.

D. GROUP TRACKING QUERY

*Definition 5 (**(s, t, n)-Group Tracking Query**): Given a set of object tracks $OT = \{ot_0, \dots ot_{M-1}\}$ over a video, a user specifies the following predicates for a (s, t, n)-group tracking query.*

- *Spatial predicate $s = (y_{pivot}, r_{long}, r_{short})$, which is a tuple of three integers: y coordinate of a pivot object pivot in an image plane, long radius r_{long}, and short radius r_{short} for the pivot ellipse centered on pivot.*
- *Temporal predicate $t = (thres_{life}, thres_{occ})$, which is a tuple of two integer thresholds: the lifetime threshold $thres_{life}$ and the occlusion threshold $thres_{occ}$.*
- *Group size predicate n, an integer that represents the minimum group size.*

An output of a (s, t, n)-group tracking query at Frame f is a set of MGTRs among the GTRs, each of which is $t_{g_i} = (f_{start}, f_{end})$ such that (1) $t_{g_i}.f_{end} - t_{g_i}.f_{start} + 1 \geq thres_{life}$, and (2) $f - t_{g_i}.f_{end} \geq thres_{occ}$.

At Frame 4 in Figure 3, the *GTR*s, $t_{\{B,C,D\}} = (0, 2)$ and $t_{\{B,C,D,E\}} = (0, 2)$ should be expired because they occlude for the occlusion threshold (=2). Both *GTR*s have a longer lifetime than the lifetime threshold (=3). However, the only *MGTR* between them is $t_{\{B,C,D,E\}}$ because their lifetimes are the same, and $\{B, C, D\}$ is a subset of $\{B, C, D, E\}$. Therefore, the output of a (s, (3, 2), 3)-*group tracking query* at Frame 4 is $t_{\{B,C,D,E\}} = (0, 2)$. We describe the spatial predicate s in Section IV.

In Scenarios (a), (b) and (c) in Section I, groups can exist in various formations. Therefore, a group definition should allow the flexible formation of a group. Meanwhile, strong connections among the members in a group are also needed to prevent a group from being formed over entire crowds. Therefore, we make strong connections via temporal consistency and spatial proximity, and allow flexible formations by considering occlusion.

IV. ELLIPSE APPROXIMATION

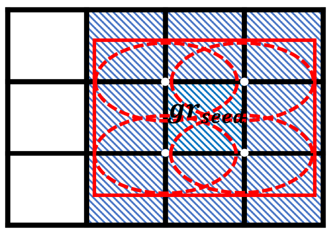
An ellipse can be specified with three parameters: a center point cp, a long radius r_{long}, and a short radius r_{short}. As we mentioned in Section III-B, it is challenging to find the exact ellipse on an image plane which is a projection of a circle on the real-world plane. Therefore, we approximate the shape of an ellipse by user-specified parameters. An approximation process is as follows. Given a sampled frame in a video, a user selects a pivot object as cp and determines the two radiuses, r_{long} and r_{short}. To resolve the pixel-pixel inconsistency, we should consider the different sizes of an ellipse according to the various locations of center points in the image plane. The work [17] made a hypothesis that the width of an object is proportional to the y coordinate of the object. They statistically verified the hypothesis for various video data with linear regression over minimum bounding box width and its y coordinates. In the same context, we conduct linear regression over the widths of an object and y coordinates in a crowd tracking data. We change the ellipse size proportionally using the coefficient of the linear regression. For the y coordinate of an ellipse, we use the center point of the ellipse. To the best of our knowledge, there is no study that defines spatial proximity using an ellipse to consider perspective projection. Using an ellipse, spatial proximity can be determined properly according to the relative positions between two objects.

V. MAXIMAL GROUP DETECTION

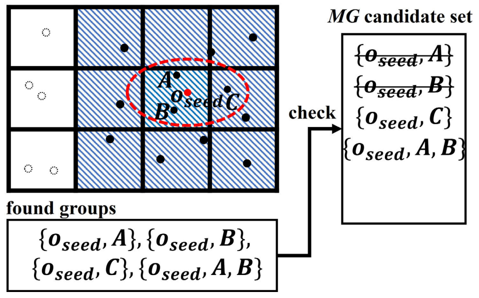
An appearance of group g_i implies appearance of g_j such that $g_j \subsetneq g_i$. That is, finding groups which are not maximal is unnecessary. In this section, given a set O of objects in a frame, we propose an efficient method to find maximal groups over O. The method is based on the definition of maximal group (MG) and pruning strategies.

A. A Naïve MAXIMAL GROUP DETECTION BASED ON GRID-BASED PARTITION

We next introduce a naïve method for our group detection step, which is based on a grid-based partition.



(a) A proximal grid cell computation



(b) A group computation and checks for an *MG* candidate set

FIGURE 4. Examples of a naïve maximal group detection. In Figure 4(a), the minimum bounding rectangle of the ellipses contains all the points that have a possibility to be proximal to an object in gr_{seed} because the ellipses are the largest in gr_{seed}. Therefore, the grid cells overlapping with the rectangle (proximal grid cells) should be examined. Figure 4(b) shows the (1) find of the groups in the proximal grid cells and (2) *MG* candidate set update with the groups found in (1).

1) GRID-BASED PARTITION

According to the definition of a group, an object in a group should be proximal to all the other objects in the group. In a naïve manner, to find groups over a set of objects in a frame, we should check all the possible combinations of the objects, which is inefficient. To resolve the inefficiency, we first preprocess crowd tracking data with a grid-based partition which consists of same-sized square grid cells on an image plane. The ids of grid cells are allocated in the row-first order, and the left-top grid cell is set to 0. With a grid-based partition, given a grid cell C, we can determine the set of grid cells which have a possibility to contain objects that are proximal to an object in C. We call such grid cells *proximal grid cells*. Proximal grid cells of a grid cell C include C itself.

2) PROXIMAL GRID CELL COMPUTATION

In Figure 4(a), the proximal grid cells of a grid cell gr_{seed} are filled with diagonal lines. Assume that a virtual ellipse which has the same shape with an ellipse centered the largest y coordinate in gr_{seed} exists on every vertex of gr_{seed}. That is, the ellipses are the largest in gr_{seed}. A minimum bounding rectangle can be obtained for four ellipses. The rectangle contains all the points that have a possibility to be proximal to an object in gr_{seed}. A grid cell overlapping with the rectangle is a proximal grid cell.

3) MAXIMAL GROUP CANDIDATE SET

To find *MG* among a set of groups, we maintain a set which contains candidates of *MG* (*MG candidate set*). An *MG* candidate set is always *checked* when a new group is found. The check with a new group g is processed as follows. For each group e in the MG candidate set, if $e \subsetneq g$, e is removed from the MG candidate set. If there is no group e such that $g \subset e$, g is added to the MG candidate set.

4) A Naïve *MG* DETECTION

Figure 4 represents an example of finding groups with the minimum group size 2, by a naïve method with a grid-based partition. Given a set O of objects in a frame of crowd tracking data, an MG detection process is the same as follows. (1) An image plane is partitioned into a regular grid. (2) After designating a grid cell as a seed grid cell gr_{seed} in an ascending order of cell ids, a set PG of proximal grid cells of gr_{seed} is computed (Figure 4(a)). (3) An object in gr_{seed} is designated as a seed object o_{seed}. Among the objects in PG, a set PO of proximal objects is computed such that $\forall po \in PO, po$ and o_{seed} are proximal. PO does not include o_{seed}. In Figure 4(b), the objects in PO are represented by the points which are A, B and C. (4) Among all the possible combinations of objects in $PO \cup \{o_{seed}\}$ which include o_{seed}, a combination that satisfies group definition can be a group. Assume that the objects A and B are proximal, and C is not proximal to both A and B. Then, found groups are presented in Figure 4(b). (5) The MG candidate set is checked with the found groups. If the MG candidate set is empty, $\{o_{seed}, C\}$ and $\{o_{seed}, A, B\}$ remain after the checks. (6) For each objects in gr_{seed}, (3)-(5) are repeated. (7) For each grid cell, (2)-(6) are repeated. (8) The groups in the MG candidate set are returned as maximal groups.

5) INCREMENTAL OBJECT COMBINATION FOR THE Naïve METHOD

In the naïve method, some proximity computations may be redundant. For example, in Figure 4(b), the proximity between o_{seed} and A is computed twice in $\{o_{seed}, A, B\}$ and $\{o_{seed}, A\}$. To resolve this inefficiency, we compute object combinations incrementally in the naïve method. Given a seed object o_{seed}, the minimum group size 2, and a set PO of proximal objects $\{A, B, C\}$ whose elements are proximal to o_{seed}, incremental object combination can be presented as a computational graph in Figure 5. Starting from a set $\{o_{seed}\}$, the incremental object combination tries to make all the possible combinations in a length-first order. The combinations are conducted by adding an object in the order in PO. We assume that the elements of PO are sorted in the order of found. Accordingly, if we assume that A, B, and C are proximal to each other, the order of creation is shown in Figure 5. Because proximity checks for the combinations are conducted incrementally, we can resolve the redundant proximity checks for common prefix. If an object cannot satisfy the proximity condition in a process, the process does not add the object and tries to add the next object in PO. When there is no object left to add and current combination satisfies the minimum group size 2, the combination is returned as a group. For example, in Figure 5, if B is not proximal to $\{o_{seed}, A\}$, $\{o_{seed}, A\}$ does not create the combination $\{o_{seed}, A, B\}$ and

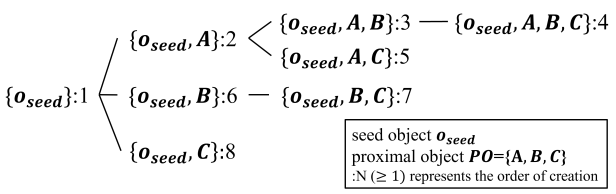


FIGURE 5. An example computational graph of incremental object combination. Starting from a set $\{o_{seed}\}$, object combinations are created in the order of the computational graph.

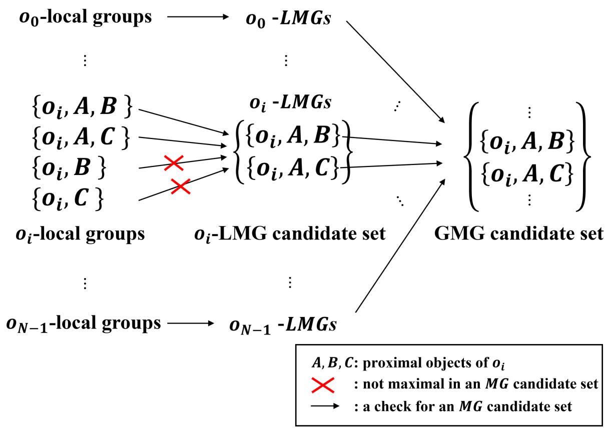


FIGURE 6. An example of an LG-MGD process. When all the o_i-local groups are checked o_i-*LMG* candidate set, the o_i-*LMG* candidate sets are merged into the *GMG* candidate set.

tries to add C. If C is proximal to $\{o_{seed}, A\}$, $\{o_{seed}, A, C\}$ returned as a group because there is no object left to add while satisfying the minimum group size. If C is not proximal to $\{o_{seed}, A\}$, $\{o_{seed}, A\}$ is returned because there is no object left to add while satisfying the minimum group size. We apply an incremental object combination to the naive method.

B. LOCAL-GLOBAL MAXIMAL GROUP DETECTION

We propose an efficient MG detection method called *Local-Global Maximal Group Detection (LG-MGD)* which reduces the number of checks and computational cost of checks for an MG candidate set. In addition, we introduce a pruning strategy for *LG-MGD*. At last, we describe the entire algorithm of *LG-MGD*.

1) LOCAL *MG* DETECTION

First, o_i*-local groups* are defined as the groups which are found from the incremental object combination in Section V-A5 for a seed object o_i. Because the proximal object set of o_i is the domain of o_i-local groups, a group in o_i-local groups is likely to have inclusion relationships with the other groups in o_i-local groups. Therefore, if we find the MGs among o_i-local groups in advance, the number of groups that are compared to the other local groups is reduced. We call the MGs among o_i-local groups o_i*-local maximal groups* (o_i-*LMG*s). We maintain a separate MG candidate set for each seed object o_i which is called o_i*-LMG candidate set*. As shown in Figure 6, after the o_i-*LMG* candidate set is checked with the o_i-local groups, groups in the o_i-*LMG* candidate set are returned as o_i-*LMG*s. Note that the checks for a group are conducted in the same manner as the naïve method. From the computational order of an incremental object combination in Section V-A5, there is a property about inclusion relationships among local groups. In an incremental object combination, because groups are returned when there is no object to add after all the trials to add objects, a group returned later cannot be a proper superset of a group returned before. With the property, the o_i-*LMG* candidate set is naturally filled with o_i-*LMG*s without removal of an existing group in the set. This property makes an incremental object combination more efficient in an *MG* detection. Note that the check of the o_i-*LMG* candidate set for a new group is still necessary to determine whether the new group is maximal in the o_i-*LMG* candidate set.

2) GLOBAL *MG* DETECTION

To find *MG*s at a frame, we should find *MG*s among the union of *LMG*s. Therefore, we maintain another candidate set, a global *MG* candidate set (*GMG* candidate set). When o_i-*LMG*s for a seed object o_i is returned, the *GMG* candidate set is checked with o_i-*LMG*s as shown in Figure 6. After the *GMG* candidate set is checked by all the *LMG*s, the groups in the *GMG* candidate set are returned as the *MG*s for the current frame. We call this entire *MG* detection process Local-Global Maximal Group Detection (*LG-MGD*). In *LG-MGD*, because a group that is not maximal is likely to be filtered in each *LMG* detection, computational cost to determine whether a group g is maximal in the *GMG* candidate set is reduced.

From the spatial constraint in group definition, we can acquire a proposition about the inclusion relationship between two groups, included in different *LMG*s.

Proposition 1: Given a set O of objects $\{o_0, o_1, \ldots, o_{N-1}\}$ in a frame and a set of local maximal groups $\{o_0$-LMG$, \ldots, o_{N-1}$-LMG$\}$, it always hold that $\forall g_i \in o_i$-LMG, $\forall g_j \in o_j$-LMG, $i \neq j$, g_i is not a proper superset of g_j, and g_j is not a proper superset of g_i.

Proof: Assume that g_i is a proper superset of g_j. Because g_j is an element of o_j-*LMG*, g_j includes o_j. Therefore, g_i also includes o_j. According to the definition of a group, it means that all the elements of g_i is proximal to o_j. Therefore, g_i or a proper superset of g_i should be an element of o_j-*LMG*. It means that a proper superset of g_j exists in o_j-*LMG*, which contradicts that g_j is an element of o_j-*LMG*. ■

With Proposition 1, a removal of subsets does not occur in a *GMG* candidate set. That is, all the final *MG*s that include o_{seed} are computed in o_{seed}-*LMG* detection. Therefore, we can remove o_{seed} from the corresponding grid cell after o_{seed}-*LMG* detection. Accordingly, when an object o_i which is proximal to o_{seed} becomes a seed, the number of proximal objects are reduced because o_{seed} is removed. Since it makes less object combinations for o_i, computational costs are reduced in both *LMG* and *GMG* detection.

Algorithms 1 and 2 describe *LG-MGD* process at a frame. Given a set *GR* of grid cells that partitions objects in a frame,

```
Algorithm 1 LG-MGD
 Input  : grid cells $GR$ for a frame, the minimum
          group size $n$
 Output: $GMG$ candidate set $GMG$
1 for each seed grid cell $gr_{seed}$ in $GR$ do
2     $PG \leftarrow$ proximal grid cells of $gr_{seed}$;
3     for each seed object $o_{seed}$ in $gr_{seed}$ do
4         get a set of proximal objects $PO$ which are
           proximal to $o_{seed}$ within $PG$;
5         $lmg \leftarrow \emptyset$;
6         LMGDetection($\{o_{seed}\}, PO, n, lmg$);
7         check $GMG$ with $lmg$;
8         remove $o_{seed}$ from $gr_{seed}$ ;      // pruning with
           Proposition 1
9 return $GMG$;
```

```
Algorithm 2 LMGDetection
 Input  : current object combination $O_{current}$,
          remaining proximal objects $O_{left}$, the
          minimum group size $n$, $LMG$ candidate set
          $lmg$
 Output: $LMG$ candidate set $lmg$
1  for each object $o_i$ in $O_{left}$ do
2      $O_{left} \leftarrow O_{left} - \{o_i\}$;
3      if $o_i$ and all the elements in $O_{current}$ are proximal
        then
4          $O_{current} \leftarrow O_{current} \cup \{o_i\}$;
5          if $O_{left}$ is not empty then
6              LMGDetection($O_{current}, O_{left}, n, lmg$);
7  if $|O_{current}| \geq n$ then
8      sort $O_{current}$ in the order of object id;
9      if $O_{current}$ is maximal in $lmg$ then
10         $lmg \cup O_{current}$;
```

a grid cell in *GR* is designated to a seed grid cell gr_{seed}, and the set of proximal grid cells *PG* of gr_{seed} is obtained (Lines 1-2, Algorithm 1). The method to get the proximal grid cells is the same as the naïve method, as introduced in Section V-A2. In gr_{seed}, an object is designated to a seed object o_{seed} and the set of proximal objects *PO* of o_{seed} in *PG* is obtained (Lines 3-4, Algorithm 1). An *LMG* candidate set *lmg* is initialized to empty set (Line 5, Algorithm 1). The algorithm calls an *LMGDetection* with the inputs which are a set $\{o_{seed}\}$ as a current object combination $O_{current}$, *PO* as a set O_{left} of remaining proximal objects, the minimum group size n, and *lmg* (Line 6, Algorithm 1). The *LMGDetection* performs o_{seed}-*LMG* detection.

For each element in O_{left}, Algorithm 2 removes o_i from O_{left} (Lines 1-2, Algorithm 2). If o_i and all the elements of $O_{current}$ are proximal, o_i is added to $O_{current}$ and if O_{left} is not empty, the recursive call of *LMGDetection* is conducted (Lines 4-6, Algorithm 2). After checking all the objects in O_{left}, the algorithm checks whether the size of $O_{current}$ is not smaller than the minimum group size n (Line 7, Algorithm 2). If the condition of the minimum group size is satisfied, $O_{current}$ becomes a group. We sort the elements of $O_{current}$ in the order of object id for the efficient comparisons of groups (Line 8, Algorithm 2). If there is no superset in *lmg*, $O_{current}$ is added to *lmg* (Lines 9-10, Algorithm 2).

The *GMG* candidate set, *GMG* is checked with the returned o_{seed}-*LMG*s (Line 7, Algorithm 1). After that, we remove o_{seed} from the grid cell gr_{seed} (Line 8, Algorithm 1). When all the objects in the current frame are processed, the groups in the *GMG* candidate set are returned as the *MG*s for the current frame.

The time complexity of *LG-MGD* is $O(NP! + NM^2)$, where N is the number of objects in a frame, P is the number of proximal objects of a seed object, and M is the number of *LMG*s of a seed object. $O(NP!)$ is the time complexity for an incremental object combination, and $O(NM^2)$ is the time complexity for the check of the *GMG* candidate set.

C. DISCUSSION

LG-MGD exploits the property of incremental object combination to reduce the number of comparisons of groups. The property of incremental object combination comes from the proposed group definition in this paper. By Proposition 1, we show the efficiency of the incremental object combination in *LG-MGD*. In addition, we acquire additional efficiency via the removal of seed object after a combination. With the aspects above, *LG-MGD* is a novel approach that finds maximal groups over crowd tracking data.

VI. GROUP TRACK MAINTENANCE

The maximal groups (*MG*s) from a group detection step at Frame f represent appearances of groups. To capture temporal consistency of the appearances, we maintain group tracks (*GTR*s), each of which contains the first and last appearance frames of a group. *GTR* maintenance includes the creation, update (for appearance frames), and expiry of a *GTR*. In this section, we propose an efficient *GTR* maintenance method which reduces unnecessary maintenance. In addition, we use a set-trie data structure to accelerate the proposed method. When two groups g and e, and two *GTR*s t_g and t_e that contain the appearance frames of g and e are given, we denote that both (1) the intersection between t_g and e and (2) intersection between t_g and t_e represent that the intersection between g and e. For the other set operations, the same manner of notations are also applied for the ease of descriptions.

A. A Naïve METHOD

As a naïve method, we use the modified clustering-and-intersection algorithm of the work [18] mentioned in Sections II-B and II-C. For each Frame f, given a set T of *GTR*s and a set G of *MG*s, an *MG* $g \in G$ conducts *GTR* maintenance as follows.

- Self Maintenance
  - **Self Update**: If $t_g \in T$, $t_g.f_{end}$ is updated to f.
  - **Self Creation**: If $t_g \notin T$, $t_g = (f, f)$ is created. If there is $t_e \in T$ such that $g \subsetneq e$, because g exists since $t_e.f_{start}$, $t_g.f_{start}$ is updated to $t_e.f_{start}$.
- Subset Maintenance
  - **Subset Update**: For each $t_e \in T$, such that $e \subsetneq g$, $t_e.f_{end}$ is updated to f.
  - **Subset Creation**: For each $t_e \in T$, such that $e \not\subset g$, if $t_{g \cap e} \notin T$, $t_{g \cap e} = (t_e.f_{start}, f)$ is created.

Because the intersection $g \cap e$ between $g \in G$ and each $t_e \in T$ is needed for Subset Creation, the existence check of t_g for Self Maintenance and subset search of g for Subset Update are conducted utilizing $g \cap e$. That is, if $g \cap e = g$ ($g \subseteq e$), one of Self Maintenance is conducted. If $g \cap e \subsetneq g$, one of Subset Maintenance is conducted. The intersection for the *GTR*s which are created at the current frame is not conducted. After *GTR* maintenance for every *MG*s at Frame f, if a group g disappears, t_g is expired. Given a set $T_{expired}$ of expired *GTR*s at Frame f, a *GTR* $t_g \in T_{expired}$, such that (1) the lifetime of t_g is longer than the lifetime threshold, and (2) t_g is a maximal group track in $T_{expired}$, is returned as an answer for a group tracking query.

Figures 7(a), (b) and (c) are the three examples of *GTR* maintenance with the naïve method. For all the examples in Figure 7, we assume that the occlusion threshold is 2, the lifetime threshold is 3 and the minimum group size is 2. A box that is located below a Frame f that is filled represents the appearance of a group that has been maintained. A box that is fully filled represents that the group is an *MG* at that frame. A box that is filled with diagonal lines represents that the group is a subset of an *MG* at that frame. The creation frame of each *GTR* is denoted with the bold character '**c**'. An empty frame of a *GTR*, t_g, represents one of three: (a) g has not appeared yet, (b) g occludes, and (c) g appears but t_g does not contain the appearance. A red X marker represents the expiry of a marked track.

In Figure 7(a), at Frame 0, the *MG* $\{A, B\}$ creates $t_{\{A,B\}} = (0, 0)$ by Self Creation. At Frame 1, the *MG* $\{A, B, C\}$ creates $t_{\{A,B,C\}} = (1, 1)$ by Self Creation, and updates $t_{\{A,B\}}.f_{end}$ to Frame 1 by Subset Update. At Frame 2, the *MG* $\{A, B, C\}$ updates both $t_{\{A,B,C\}}.f_{end}$ and $t_{\{A,B\}}.f_{end}$ to Frame 2 by Self Update and Subset Update, respectively.

In Figure 7(b), at Frame 0, the *MG* $\{A, B, C\}$ creates $t_{\{A,B,C\}} = (0, 0)$ by Self Creation. At Frame 1, the *MG* $\{A, B\}$ creates $t_{\{A,B\}} = (1, 1)$ and updates $t_{\{A,B\}}.f_{start}$ to $t_{\{A,B,C\}}.f_{start}$ (= 0) by Self Creation. At Frame 2, the *MG* $\{A, B\}$ updates $t_{\{A,B\}}.f_{end}$ to Frame 2 by Self Update. After that, since $t_{\{A,B,C\}}$ occludes during the occlusion threshold (=two frames), it is expired.

In Figure 7(c), at Frame 0, the *MG* $\{A, B, C, D\}$ creates $t_{\{A,B,C,D\}} = (0, 0)$ by Self Creation. At Frame 1, an *MG* $\{A, B, C, D\}$ updates $t_{\{A,B,C,D\}}.f_{end}$ to Frame 1 by Self Update. The other *MG* $\{A, B, C, E\}$ creates $t_{\{A,B,C,E\}} = (1, 1)$ by Self Creation, and create $t_{\{A,B,C,E\} \cap \{A,B,C,D\} = \{A,B,C\}} = (0, 1)$ by Subset Creation. At Frame 2, the *MG* $\{A, B, C, E\}$

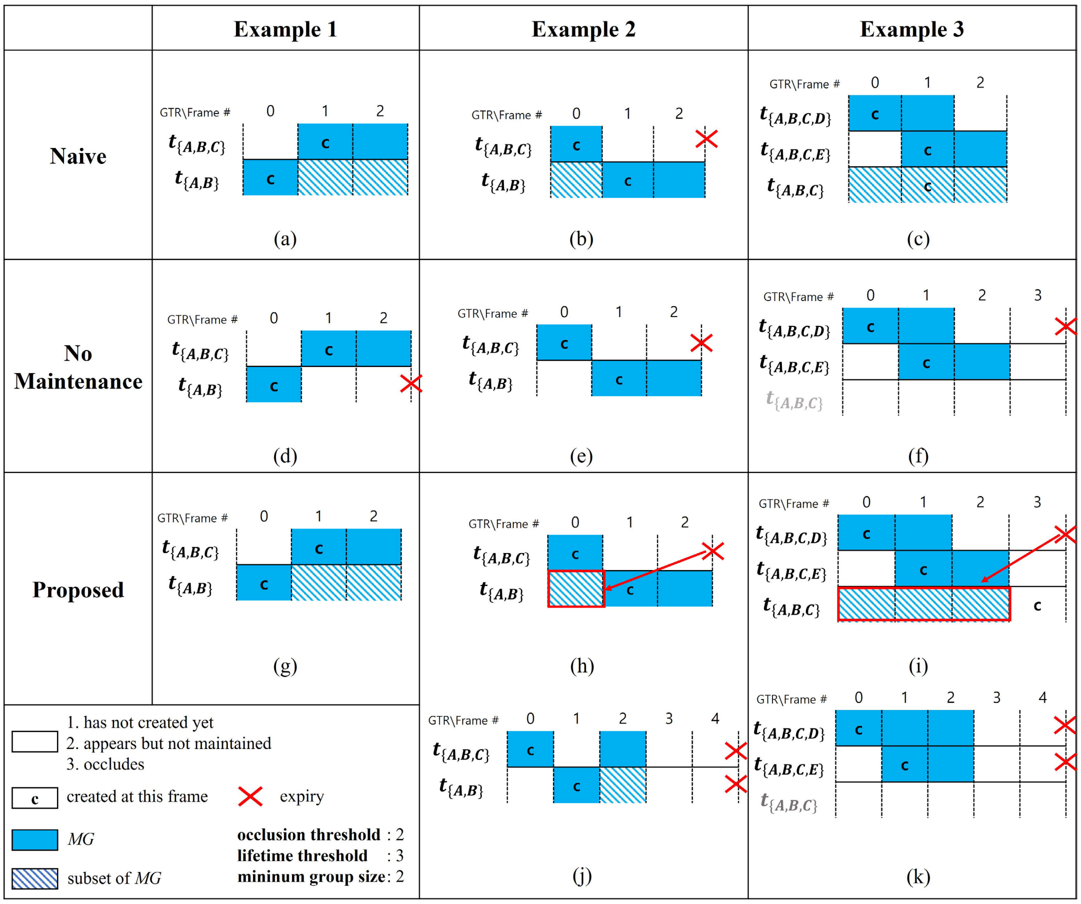


FIGURE 7. **Examples for group track maintenance with various methods.**

updates both $t_{\{A,B,C,E\}}.f_{end}$ and $t_{\{A,B,C\}}.f_{end}$ to Frame 2 by Self Update and Subset Update, respectively.

1) INTUITIONS

In crowd tracking data, a person may form a lot of temporary groups with people who pass by, which may cause a lot of Self Creation and Update. The majority of the groups are unlikely to be answers of a group tracking query. Moreover, the temporary groups also cause Subset Update and Subset Creation. If we have knowledge about the appearances of *GTR*s, we may prune unnecessary maintenance. Here, we get an intuition for making maintenance being patient. Because the appearances of a group are lost after the expiry of a *GTR*, if we perform proper maintenance at the expiry, the appearances are not lost. At the same time, both f_{start} and f_{end} of *GTR*s can be exploited to determine whether maintenance should be performed. To do that, we should determine which maintenance would become patient.

B. OPPORTUNITIES TO BE PATIENT IN THE Naïve METHOD

This subsection, for each type of maintenance in the naïve method, we assume that a single type of maintenance is not conducted and introduce observations. With the observations, we determine which type of maintenance becomes patient.

1) SELF UPDATE

In Figure 7(d), assume that the *MG* at Frame 1 and 2 is not $\{A, B, C\}$, but $\{A, B\}$. Then, the only existing *GTR* at Frame 2 would be $t_{\{A,B\}} = (0, 2)$ as shown in Figure 7(a). However, because Self Update is not conducted, $t_{\{A,B\}}$ would be $(0, 0)$ and is expired at Frame 2.

a: OBSERVATION

At Frame 2, because we assume that the *MG* at Frame 1 and 2 is not $\{A, B, C\}$, but $\{A, B\}$, there is no *GTR* that contains the appearances of $\{A, B\}$ at Frame 1 and 2. Therefore, Self Update cannot be patient.

2) SELF CREATION

Given an *MG* g, there are two steps in Self Creation. First, the creation of t_g, and second, the update of $t_g.f_{start}$ to the start frame of a superset of g. If the creation of t_g is not conducted at Frame f, Self Update after Frame f also cannot be conducted. Because Self Update cannot be patient, the creation of t_g also cannot be patient. Therefore, we assume that the update of $t_g.f_{start}$ is not conducted. With the assumption, in Figure 7(e), the appearance of $\{A, B\}$ at Frame 0 is lost because $t_{\{A,B\}}.f_{end}$ is not updated to Frame 0, and $t_{\{A,B,C\}}$ is expired at Frame 2.

a: OBSERVATION

At Frame 2, if $t_{\{A,B,C\}}$ updates $\{A, B\}.f_{start}$ to Frame 0, $t_{\{A,B\}}$ does not loss the appearance. To get the update, $t_{\{A,B\}}$ should have been created before Frame 2. Therefore, in Self Creation, only the update for the start frame can be patient.

3) SUBSET UPDATE

In Figure 7(d), because Subset Update is not conducted, the *MG* $\{A, B, C\}$ does not update $t_{\{A,B\}}.f_{end}$ to Frame 1 and 2. Therefore, $t_{\{A,B\}}$ is expired at Frame 2.

a: OBSERVATION

Because $t_{\{A,B\}}$ can be expired earlier than the superset of $t_{\{A,B\}}$, $t_{\{A,B\}}$ should check all the supersets frequently to get the appearances. For example, if $\{A, B, C\}$ appears continuously after Frame 2, $t_{\{A,B\}}$ should check all the existing supersets in every two frames (=the occlusion threshold). Accordingly, Subset Update would be better not to be patient.

4) SUBSET CREATION

In Figure 7(f), because Subset Creation is not conducted, $t_{\{A,B,C\}}$ is not created and the appearance of $\{A, B, C\}$ at Frame 0 is lost because $t_{\{A,B,C,D\}}$ is expired at Frame 3.

a: OBSERVATION

At Frame 3, if $t_{\{A,B,C,D\}}$ creates $t_{\{A,B,C\}}$ and updates appearances of $\{A, B, C\}$, $t_{\{A,B,C\}}$ would not loss the appearance at Frame 0. Therefore, Subset Creation can be patient.

According to the observations, we conduct Subset Creation and the start frame update in Self Creation at *GTR* expiry. Our *GTR* maintenance consists of two phases: (1) an appearance update phase, and (2) an expiry phase. We introduce our appearance update phase first.

C. APPEARANCE UPDATE PHASE

GTR maintenance in an appearance update phase is conducted as follows: For each *MG* g, if t_g exists, Self Update is conducted. If t_g does not exist, Self Creation is conducted. Different from the naïve method, Self Creation at Frame f creates $t_g = (f, f)$ without the update of the start frame. In addition, for each existing subset of g, Subset Update is conducted. In the naïve method, *MG*s are intersected with all the existing *GTR*s for Subset Creation. However, because we conduct Subset Creation at an expiry phase, there is no necessity for intersection in an appearance update phase.

Figure 7(g), (h), and (i) depict three examples of our *GTR* maintenance. At Frame 0 in Figure 7(g), the *MG* $\{A, B\}$ creates $t_{\{A,B\}} = (0, 0)$ by Self Creation. At Frame 1, the *MG* $\{A, B, C\}$ creates $t_{\{A,B,C\}} = (1, 1)$ by Self Creation, and updates $t_{\{A,B\}}.f_{end}$ to Frame 1 by Subset Update. At Frame 2,

Algorithm 3 Group Track Maintenance for Each Frame

```
Input  : current Frame $f$, set of $MG$s $G$, the occlusion
         threshold $thres_{occ}$, the lifetime threshold
         $thres_{life}$, the minimum group size $n$, set of
         tracks $TRACK$
Output: updated tracks $TRACK$, set of $MGTR$
         candidates $mgtr$
   // Appearance update phase;
1  for each $MG$ $g$ in $G$ do
2      if $t_g$ exists in $TRACK$ then
3          $t_g.f_{end} = f$;
4      else
5          a new track $t_g = (f, f)$ is added to $TRACK$;
6      update $t_e.f_{end}$ to $f$ such that $e \subsetneq g$, $t_e \in TRACK$;
   // Expiry phase;
7  for each track $t_g$ in $TRACK$ do
8      if $g$ disappears at Frame $f$ then
9          if lifetime of $t_g$ is no shorter than $thres_{life}$ then
10             if $t_g$ is a maximal group track in $mgtr$ then
11                 $t_g$ is added to $mgtr$;
12         for each track $t_e$ $(e != g)$ in $TRACK$ do
13             if $t_e$ is not expired at Frame $f$ and
               $|g \cap e| > n$ then
14                 if $t_{g \cap e}$ exists in $TRACK$ then
15                     $t_{g\cap e}.f_{start} = min(t_g.f_{start}, t_{g\cap e}.f_{start})$
                        ; // min() returns the smaller value
16                 else
17                     a new track $t_{g\cap e} =$
                        $(min(t_g.f_{start}, t_e.f_{start}), t_e.f_{end})$ is
                        added to $TRACK$;
18         $t_g$ is removed from $TRACK$;
```

the *MG* $\{A, B, C\}$ updates $t_{\{A,B,C\}}.f_{end}$ Frame 2 by Self Update, and updates $t_{\{A,B\}}.f_{end}$ to Frame 2 by Subset Update.

Figure 7(h), at Frame 0, the *MG* $\{A, B, C\}$ creates $t_{\{A,B,C\}} = (0, 0)$ by Self Creation. At Frame 1, the *MG* $\{A, B\}$ creates $t_{\{A,B\}} = (1, 1)$ by Self Creation. Note that $t_{\{A,B\}}.f_{start}$ is not updated to Frame 0 because the start frame update in Self Creation is conducted at an expiry phase. At Frame 2, the *MG* $\{A, B\}$ updates $t_{\{A,B\}}.f_{end}$ to Frame 2 by Self Update.

In Figure 7(i), at Frame 0, the *MG* $\{A, B, C, D\}$ creates $t_{\{A,B,C,D\}} = (0, 0)$ by Self Creation. At Frame 1, an *MG* $\{A, B, C, D\}$ updates $t_{\{A,B,C,D\}}.f_{end}$ to Frame 1 by Self Update. The other *MG* $\{A, B, C, E\}$ creates $t_{\{A,B,C,E\}} = (1, 1)$ by Self Creation. Note that $t_{\{A,B,C\}}$ is not created at Frame 1 because Subset Creation is conducted at an expiry phase. At Frame 2, the *MG* $\{A, B, C, E\}$ updates $t_{\{A,B,C,E\}}.f_{end}$ to Frame 2 by Self Update.

Lines 1-6 in Algorithm 3 describe an appearance update phase. For each *MG* g at Frame f, the algorithm checks whether t_g exists in a set of *GTR*s, *TRACK* (Lines 1-2). If t_g exists, $t_g.f_{end}$ is updated to f (Self Update, Line 3). If t_g does not exist, the algorithm creates $t_g = (f, f)$ (Self Creation without update, Line 5). For each existing t_e ($e \subsetneq g$), $t_e.f_{end}$ is updated to f (Subset Update, Line 6).

D. EXPIRY PHASE

After an appearance update phase, given a set T of existing *GTR*s, the occlusion threshold $thres_o$ and Frame f, $t_g \in T$, such that $f - t_g.f_{end} \geq thres_o$ (i.e., g disappears), is expired. The expired t_g conducts *GTR* maintenance with the following rules.

Expiry Rule 1. The intersection between t_g and each $t_e \in T$ which is not expired at Frame f is conducted.

- **Expiry Rule 1-1.** If $t_{g\cap e} \in T$, $t_{g\cap e}.f_{start}$ is updated to the minimum value between $t_g.f_{start}$ and $t_{g\cap e}.f_{start}$.
- **Expiry Rule 1-2.** If $t_{g\cap e} \notin T$, $t_{g\cap e}$ is created. In addition, $t_{g\cap e}.f_{start}$ is updated to the minimum value between $t_g.f_{start}$ and $t_e.f_{start}$, and $t_{g\cap e}.f_{end}$ is updated to $t_e.f_{end}$.

Expiry Rule 2. If the lifetime of t_g is no shorter than the lifetime threshold, t_g is added to a candidate set of maximal group track (*MGTR* candidate).

By Expiry Rule 1-1, an expired *GTR* t_g update each subset $t_e \in T$, such that $t_g.f_{start} < t_e.f_{start}$. Therefore, the start frame update in Self Creation is handled as shown in Figure 7(h). At Frame 2, the expired *GTR* $t_{\{A,B,C\}}$ is intersected with $t_{\{A,B\}}$ according to Expiry Rule 1. Because the intersection $t_{\{A,B\}}$ exists, according to Expiry Rule 1-1, $t_{\{A,B\}}.f_{start}$ is updated to the minimum value 0 between $t_{\{A,B,C\}}.f_{start}$ and $t_{\{A,B\}}.f_{start}$. Therefore, the appearance of $\{A, B\}$ at Frame 0 is not lost. When the two *GTR*s are expired at the same time as shown in Figure 7(j), the update is unnecessary because $t_{\{A,B\}}$ cannot be an *MGTR*. Even if $t_{\{A,B\}}.f_{start}$ is updated to Frame 0, because the start and end frame of both $t_{\{A,B,C\}}$ and $t_{\{A,B\}}$ are same as Frame 0 and 2, respectively, the *MGTR* is $t_{\{A,B,C\}}$. Accordingly, even if the appearance of $\{A, B\}$ at Frame 0 is not updated, the answer of group tracking query can be returned correctly. The reduction of unnecessary updates makes our *GTR* maintenance efficient.

By Expiry Rule 1-2, Subset Creation is handled as shown in Figure 7(i). At Frame 3, $t_{\{A,B,C,D\}}$ is expired. According to Expiry Rule 1, $t_{\{A,B,C,D\}}$ is intersected with $t_{\{A,B,C,E\}}$ which is not expired at Frame 3. Because the intersection $t_{\{A,B,C\}}$ does not exist, according to Expiry Rule 1-2, $t_{\{A,B,C\}} = (0, 3)$ is created. The naïve method creates $t_{\{A,B,C\}} = (0, 1)$ at Frame 1. Therefore, the appearance of $\{A, B, C\}$ at Frame 2 is updated by Subset Update. However, our *GTR* maintenance creates $t_{\{A,B,C\}}$ after Frame 2 (i.e., Frame 3). Therefore, we should conduct not only the creation of $t_{\{A,B,C\}} = (0, 1)$, but also the update of $t_{\{A,B,C\}}.f_{end}$ to Frame 2. As a result, $t_{\{A,B,C\}}$ is maintained without any appearance loss. When both $t_{\{A,B,C,D\}}$ and $t_{\{A,B,C,E\}}$ are expired at the same time as shown in Figure 7(k), the creation of $t_{\{A,B,C\}}$ is unnecessary. With the naive method, $t_{\{A,B,C\}}$ is created at Frame 1 and expired at Frame 4. However, in our *GTR* maintenance, we know that the creation of $t_{\{A,B,C\}}$ is unnecessary because it cannot be an *MGTR*. Even if $t_{\{A,B,C\}}$ is created, because the start and end frame of both $t_{\{A,B,C,D\}}$ and $t_{\{A,B,C\}}$ are same as Frame 0 and 2, respectively, the *MGTR* is $t_{\{A,B,C,D\}}$. Accordingly, even if $t_{\{A,B,C\}}$ is not created, the answer of group tracking query can be returned properly. The reduction of unnecessary creation makes our *GTR* maintenance efficient.

Lines 7-18 in Algorithm 3 show a pseudo-code for an expiry phase. At the current Frame f, for each existing track t_g, if $f - t_g.f_{end} \geq thres_{occ}$ (i.e., g disappears at Frame f), t_g is expired (Lines 7-8). If the lifetime of t_g is no shorter than the lifetime threshold and t_g is a maximal group track among the *MGTR* candidates found so far, t_g is added to set of *MGTR* candidates (Expiry Rule 2, Lines 9-11). For each existing *GTR* t_e ($e \neq g$), if t_e is not expired at Frame f, and $|g \cap e|$ is no smaller than the minimum group size n, then the existence check of $t_{g\cap e}$ is performed (Expiry Rule 1, Lines 12-13). When $t_{g\cap e}$ exists in the set of *GTR*s, *TRACK*, $t_{g\cap e}.f_{start}$ is updated to the minimum value between $t_g.f_{start}$ and $t_{g\cap e}.f_{start}$ (Expiry Rule 1-1, Lines 14-15). If $t_{g\cap e}$ does not exist, $t_{g\cap e}$ is created. In addition, $t_{g\cap e}.f_{start}$ is updated to the minimum vujpmm alue between $t_g.f_{start}$ and $t_e.f_{start}$, and $t_{g\cap e}.f_{end}$ is updated to $t_e.f_{end}$. (Expiry Rule 1-2, Lines 16-17). Because t_g is expired, t_g is removed from *TRACK* (Line 18). The time complexity of Algorithm 3 is $O(mNT + T^3)$, where m is the maximal size of a group, N is the number of *MG*s, and T is the number of existing *GTR*s. $O(mNT)$ is the time complexity for an appearance update phase, and $O(T^3)$ is the time complexity for an expiry phase.

E. DISCUSSION

As we mentioned in VI-A1, crowd tracking data contains a lot of temporary groups. That is, the groups in a certain frame are unlikely to be answers of a group tracking query. The existing studies to capture temporal consistency of groups [13], [14], [15], [16], [18] use the clustering-and-intersection algorithm which is inefficient because they do not consider temporary groups. To the best of our knowledge, there is no study that resolves the inefficiency caused by temporary groups.

F. FAST SET OPERATIONS WITH SET-TRIE

Our *GTR* maintenance includes a lot of set operations which are checking the existence of groups, and searching the subsets of groups. Therefore, we can apply a set-trie index data structure [21], [22] to accelerate *GTR* maintenance. Set-trie is a tree composed of nodes labeled with an element in the domain of [1,N]. Figure 8 presents an example of a set-trie. The root node is an empty set, and its children are labeled with an integer in the domain. A node labeled i cannot have a child labeled j such that $i > j$. A node with a square means the existence of a group which consists of a sequence of labels from the root to the current node. As shown in Figure 8, a set-trie indexes *GTR*s. When we want to find the subsets of the *GTR* $t_{\{1,2,3,4\}}$, for example, the subsets $t_{\{1,2\}}$ and $t_{\{1,2,3\}}$ can be found in a single traversal. Since objects in a group is sorted in the order of object id (Line 8, Algorithm 2), there is

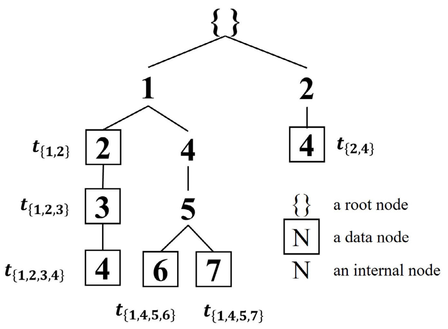


FIGURE 8. An example of set-trie that indexes group tracks. A data node contains a group that consists of a sequence of labels from the root to the current node. In a single traversal, several sets in inclusion relationships can be found.

no need for additional computations to apply a set-trie. The time complexity of our *GTR* maintenance with a set-trie is $O(m^2cN + mbT)$, where m is the maximal size of a group, c is the maximum number of children of a node in the set-trie, N is the number of *MG*s at the current frame, and T is the number of existing tracks. $O(m^2cN)$ is the time complexity for an appearance update phase, and $O(mbT)$ is the time complexity for an expiry phase.

VII. A WORKFLOW OF GROUP TRACKING QUERY PROCESSING

Figure 9 depicts a workflow of group tracking query processing for a frame. The workflow proceeds in the order of circled numbers in the figure. Given Frame f, ① Frame f is grid-partitioned. ② A maximal group detection on Frame f is conducted with the proposed *LG-MGD* process. ③ Group track maintenance is conducted for the maximal groups in Frame f. ④ Among the expired group tracks, the group tracks satisfying the predicates of a group tracking query are returned. ⑤ After the expiry phase, the process is repeated for the next frame. When all the frames are processed, group tracking query processing is finished.

VIII. EXPERIMENTAL EVALUATIONS

A. EXPERIMENTAL SETUP

1) SETTINGS

Our evaluations were conducted on a server with Intel(R) Xeon(R) CPU E5-2698 v4 @2.20GHz, 256GB memory, and four Tesla V100 GPUs. The operating system for the server was Ubuntu 18.04.6 LTS. Evaluated algorithms were implemented using C++ 14. Experimental results were averaged for 10 executions. For the visualization of results, we used OpenCV and Python 3.6.

2) DATASETS

We used MOT (Multi Object Tracking) [23], [24] dataset which is one of the most popular benchmarks for a multi-object tracking task. MOT dataset provides pairs of a real-world video and a ground truth of individual object tracking data for the video. Among them, MOT20-02,

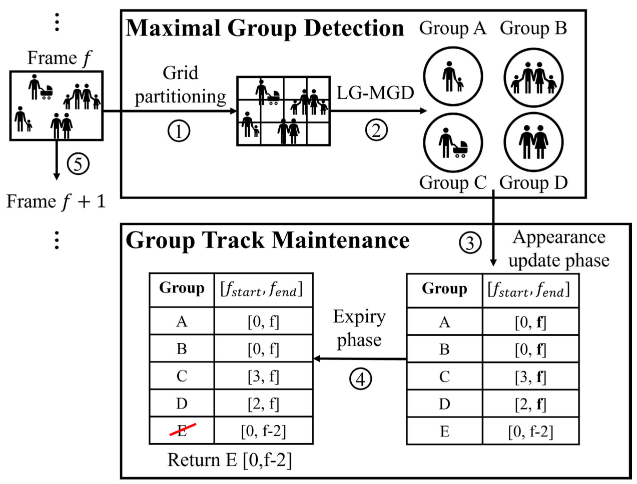


FIGURE 9. An example workflow of group tracking query processing. Given Frame f, a query is processed in the order of label ①-⑤.

TABLE 1. Dataset specifications.

dataset	MOT02GT	MOT02IF	MOT03GT	MOT03IF	MOT04GT	MOT04IF
total # of frames	2782 (111 seconds)		2405 (96 seconds)		5389 (179 seconds)	
resolution (pixels)	1920x1080		1173x880		1920x1080	
frames per second	25		25		30	
total # of unique objects in a video	296		735		83	
average # of objects per frame	59.9	46.1	139.6	108.5	39.3	41.2

MOT20-03, and MOT16-04 (denoted as MOT02, MOT03, and MOT04, respectively) datasets were used. The example frames for the datasets are presented in Figure 10. Because the ground truths of object tracking data are labeled manually for the entire frames, it does not include object occlusion. Therefore, we generated datasets utilizing the inferences of deep-learning based object detection and tracking models. We adopted Yolov5-crowdhuman [25] as an object detection model and DeepSort [26] as an object tracking model. We denote the ground truths and generated data (inferenced data) by appending GT and IF, respectively (e.g., MOT02GT, MOT04IF). The specifications of the datasets are presented in Table 1. In the average number of objects per frame, the differences between ground truths and inferenced data come from detection errors in object detection models.

3) PREDICATES AND PARAMETERS

The default predicates of (s, t, n)-group tracking query are determined to get reasonable groups. When a user wants to impose different constraints (e.g., stricter spatial proximity, looser temporal consistency, etc.) on a group, the user can vary the predicates.

In the grid-based partition that introduced in Section V-A, grid granularity represents the number of grid cells in a row (x-axis). The number of grid cells in a column (y-axis) is depending on the grid granularity because a grid cell is a square. We empirically set the granularity which shows the best performance in the naïve method as a default. The default predicates and parameters are presented in Table 2.

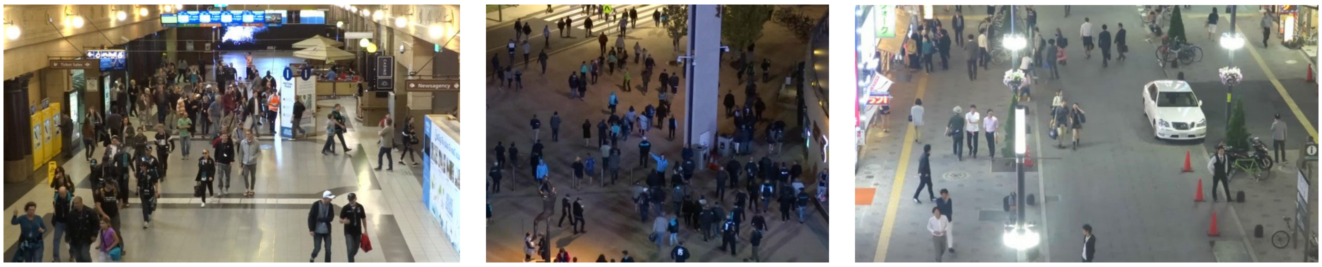

(a) MOT02 (b) MOT03 (c) MOT04

FIGURE 10. The sample frames of three datasets (MOT02, MOT03, and MOT04).

TABLE 2. The default parameters for each dataset.

dataset	MOT02GT	MOT02IF	MOT03GT	MOT03IF	MOT04GT	MOT04IF
pivot ellipse radius (pixels)	300, 100		120, 60		140, 70	
pivot object size (pixels)	128, 347		61, 130		103, 241	
lifetime threshold (frames)	500	125	500	125	125	100
minimum group size	2	2	2	2	2	2
occlusion threshold (frames)	125	25	125	50	25	50
grid granularity	6	7	9	8	5	5

4) EVALUATED ALGORITHMS

We evaluated the performances of the methods introduced in each query processing step (group detection and group track maintenance). The methods are as follows.

- Group detection
  - ***N-DET***: Incremental object combination with a grid-based partition in Section V-A
  - ***P-DET***: *LG-MGD* with the pruning strategy in Section V-B
- Group track maintenance
  - ***N-TRK***: The clustering-and-intersection algorithm in Section VI-A
  - ***P-TRK***: The proposed method which conducts some *GTR* maintenance at expire phases, and accelerated by set-trie in Sections VI-C and VI-D

For *N-TRK*, we also applied an additional optimization which was introduced in the work [18]. The optimization makes connections among the group tracks in inclusion relationship. When an existing group track t_e is intersected with an *MG*, if the intersection is empty, Subset Creation is not conducted. Therefore, the connected subsets of t_e also do not need to be intersected to conduct Subset Creation.

B. EXPERIMENTAL RESULTS

1) PERFORMANCE EVALUATIONS FOR THE GROUP TRACKING QUERY

This section introduces the group track query processing performances of the methods with varying predicates. The predicates are the occlusion threshold, the minimum group size, and pivot ellipse size. Because the lifetime threshold does not affect a query performance, the evaluation for the lifetime threshold was omitted. When a predicate varied, the other predicates were set to the default values in Table 2. Figures 11-13 show the execution time of group tracking queries with the different predicates. In all the graphs, *P-DET+P-TRK* outperformed the other methods. The performance gain obtained via (1) pruning of local groups and seed objects in group detection step, and (2) reduction of unnecessary *GTR* maintenance and acceleration of set operations in *GTR* maintenance step. From the comparison of *N-DET+N-TRK*, *P-DET+N-TRK*, and *N-DET+P-TRK*, we can know that a *GTR* maintenance step affected query execution time more than a group detection step.

The query execution times were much smaller than the total running time of a video. For example, in MOT02GT dataset with the total running time of 111 seconds, *P-DET+P-TRK* processed a query with the default parameters in 0.7 seconds. Even if the predicates varied, the longest query execution time was 4.9 seconds which was about 22.3 times shorter than the total video running time. These results support the efficiency of our query processing approach for group tracking in video monitoring systems.

Figure 11 shows execution times according to pivot ellipse radiuses. We changed radiuses keeping a proportion between short radius and long radius of the default ellipse. The long radiuses of pivot ellipses were presented in the graphs. In the entire graphs, the execution time increased exponentially with the increments of pivot ellipse radius. It was because the area of an ellipse increased exponentially as the radius increased, the number of proximal objects from an object tended to increase exponentially.

In Figure 12, the execution times increased when the occlusion threshold increased. When the occlusion threshold became larger, the *GTR*s for temporary groups were likely to be maintained longer. It means that increments of (1) Subset Updates and (2) the intersections for an expired *GTR*. Because *P-TRK* reduced unnecessary *GTR* maintenance, performance gaps between *N-DET+N-TRK* and *N-DET+P-TRK* became larger as the occlusion threshold became larger. Figures 12(e)-12(f) show different trends with the Figures 12(a)-12(d). In case of *N-DET+P-TRK* and *P-DET+P-TRK*, execution time did not increase linearly. It was because (1) MOT04GT contained fewer groups in a frame and (2) temporary groups are fewer than the other datasets because the distance between people was relatively

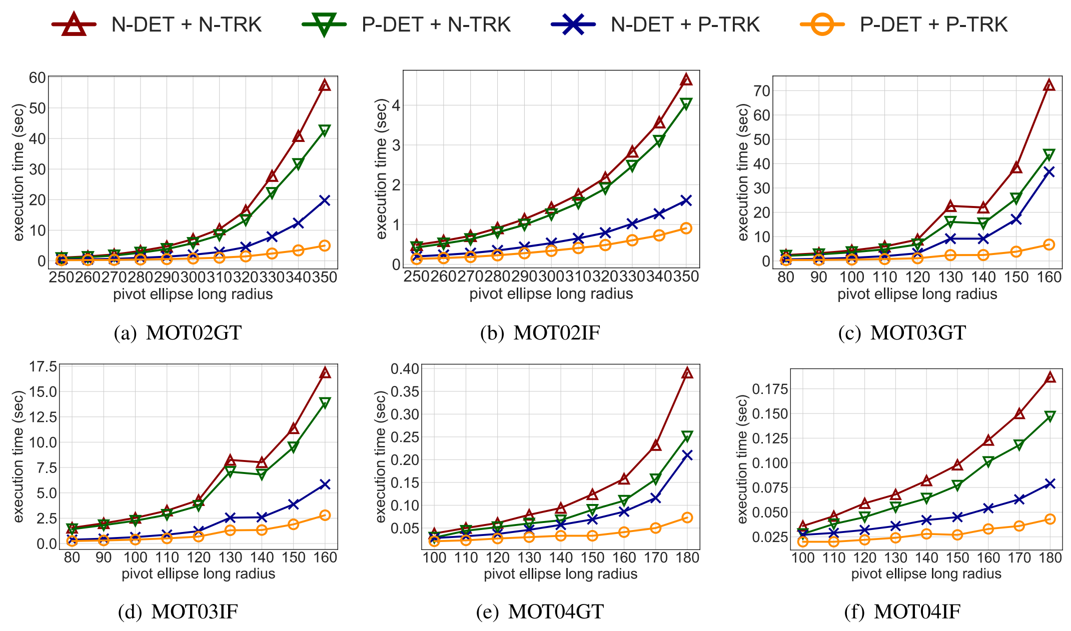


FIGURE 11. Query execution time with various ellipse sizes.

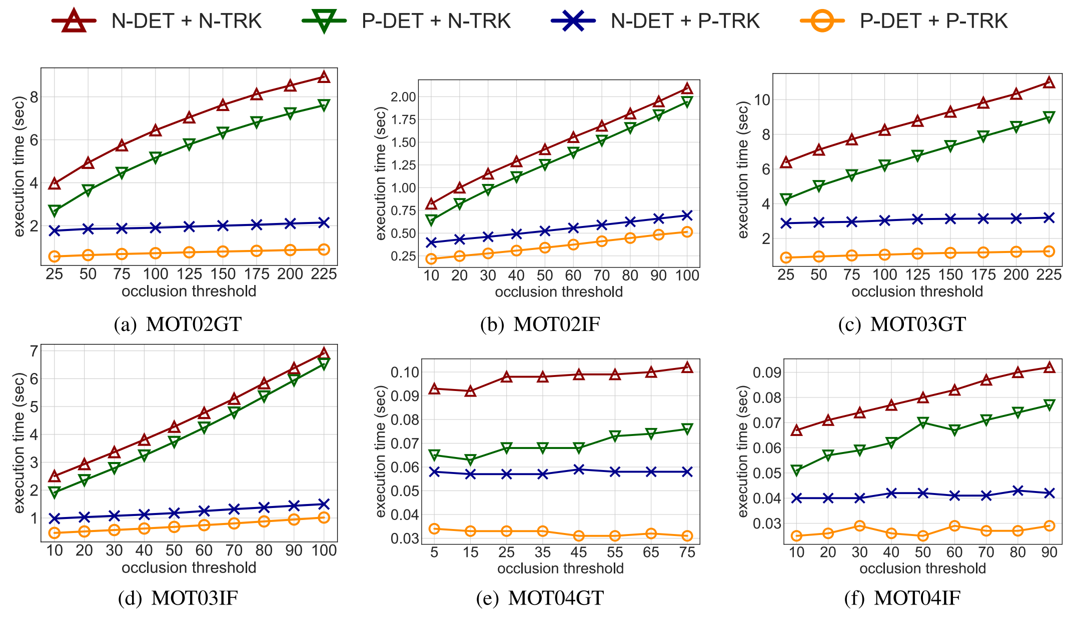


FIGURE 12. Query execution time with various occlusion threshold.

large. Therefore, the number of *GTR*s was smaller. As a result, in the small occlusion threshold, the performance gain of set-trie was smaller than the overhead of set-trie.

In Figure 13, as the minimum group size increased, the performance gain of the proposed became smaller. Execution time of *N-DET*+*P-TRK* became slower than *P-DET*+*N-TRK* as the minimum group size became larger. When the minimum group size became larger, detected groups became smaller, and the number of *GTR*s also became smaller. Since the performance gains of *P-DET* and *P-TRK* came from reducing the number of detected groups and *GTR*s, respectively, the performance gains became smaller.

2) EFFECTS OF GRID GRANULARITY ON GROUP DETECTION STEP

In this section, we analyzed the effect of grid granularity on the group detection time. We measured and compared

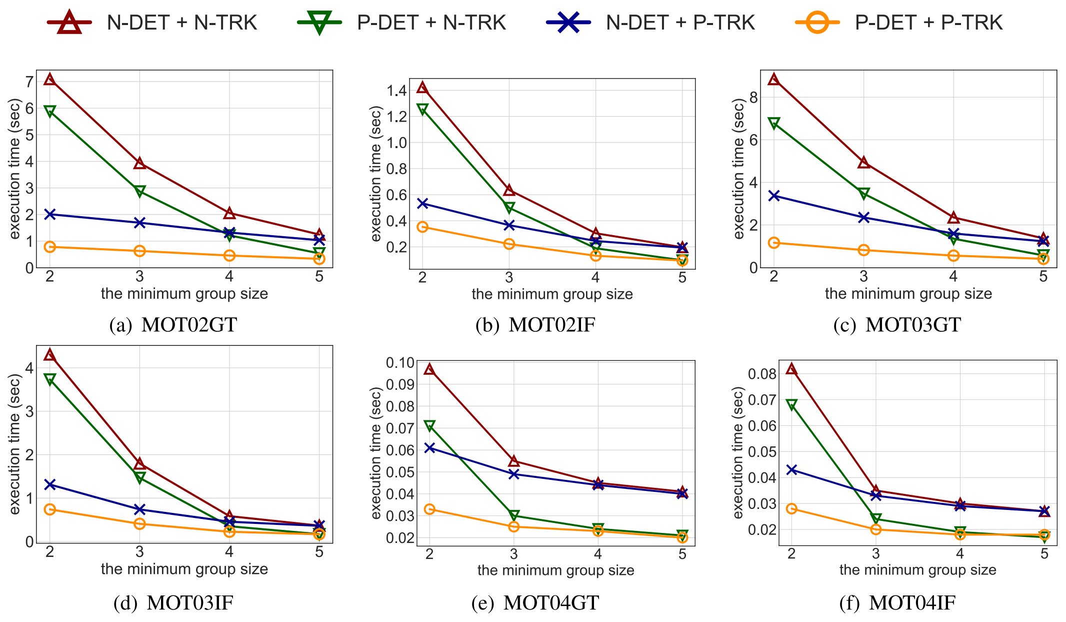


FIGURE 13. Query execution time with various minimum group size.

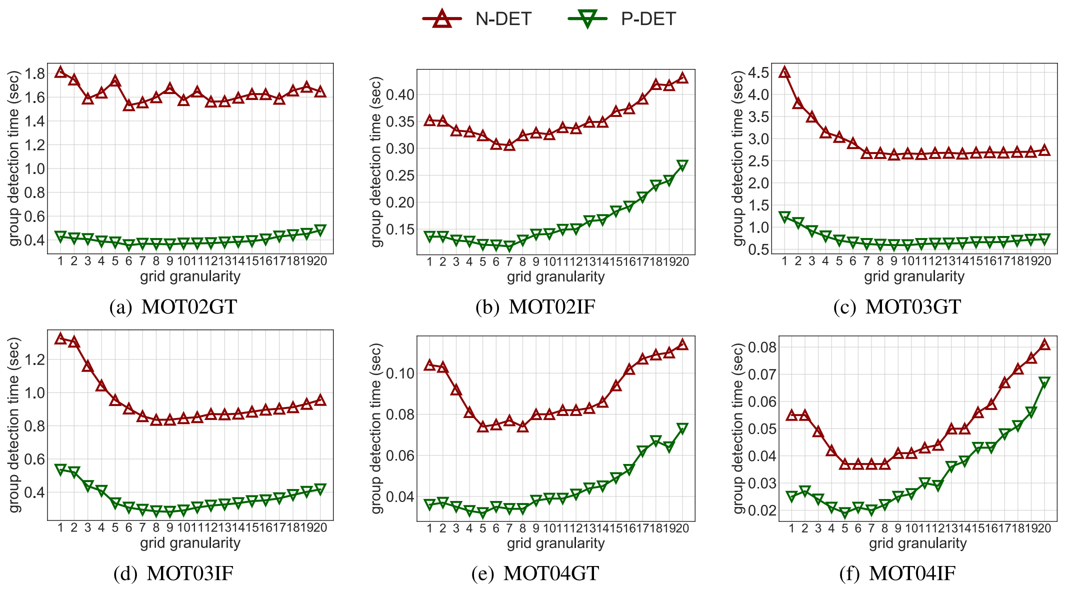


FIGURE 14. Group detection time with various grid granularity.

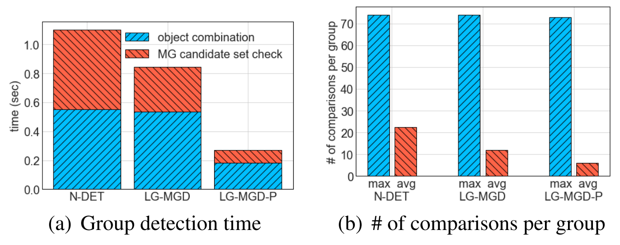


FIGURE 15. Group detection analysis. The results of the ablation study on the group detection time and the number of comparisons in *MG* candidate sets are presented.

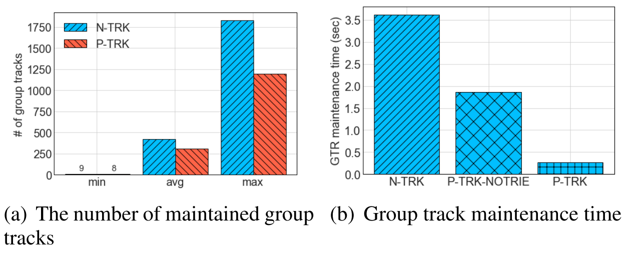
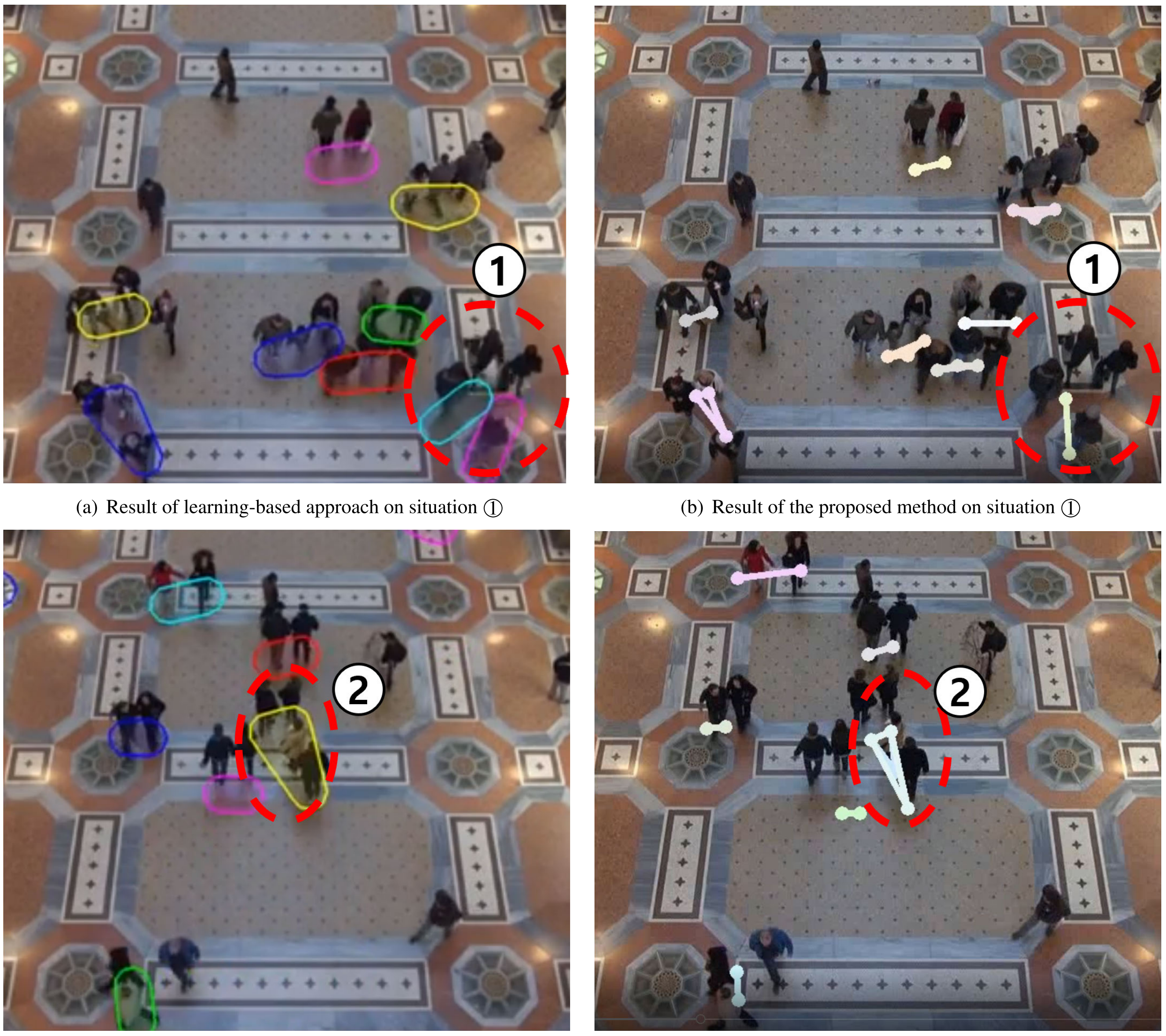


FIGURE 16. Group track maintenance analysis. The results of the ablation study on the number of maintained group tracks and maintenance time are presented.

group detection times of two group detection methods: (1) *N-DET* and (2) *P-DET*. Figure 14 shows group detection times according to the different granularity. Grid granularity 1 means no partitions. As shown in Figure 14, grid granularity



(a) Result of learning-based approach on situation ①

(b) Result of the proposed method on situation ①

(c) Result of learning-based approach on situation ②

(d) Result of the proposed method on situation ②

FIGURE 17. The sample frames for the visualized results of group tracking query and learning-based approach for the same dataset. A pair of figures with the same circled number shows different results for the same situation.

that was too small or large was not good. It was due to the tradeoff between the number of objects that are pruned and the number of grids that should be checked. We set granularity which shows the best performance in the naïve method as a default.

3) ABLATION STUDIES FOR GROUP DETECTION STEP

To further analyze our group detection method, we conducted an ablation study for each component of *P-DET*, which consists of *LG-MGD* and the pruning strategy that removes a seed object *o* after *o-LMG* detection in Section V-B. The evaluation was conducted over MOT02GT dataset. The compared methods were *N-DET*, *LG-MGD* and *LG-MGD* with the pruning strategy (*LG-MGD-P*). Figure 15(a) shows the maximal group detection time of each method, and Figure 15(b) shows the number of comparisons for a group at which checks of *LMG* and *GMG* candidate sets. Each bar in Figure 15(a) consists of incremental object combination time (bottom) and *MG* candidate set check time (top). Times to check *MG* candidate sets for *LG-MGD* was less than *N-DET*. It means that the computation of local maximal group was effective to reduce the average number of comparisons for groups as shown in Figure 15(b). *LG-MGD-P* significantly reduced the group detection time compared to *LG-MGD* as shown in Figure 15(a). It was because by removing a seed object, the number of proximal objects for the other seeds was reduced. Accordingly, as shown in Figure 15(b), the average number of comparisons was also reduced. Meanwhile, when a group was not filtered in an *LMG* detection, the group should be compared with all

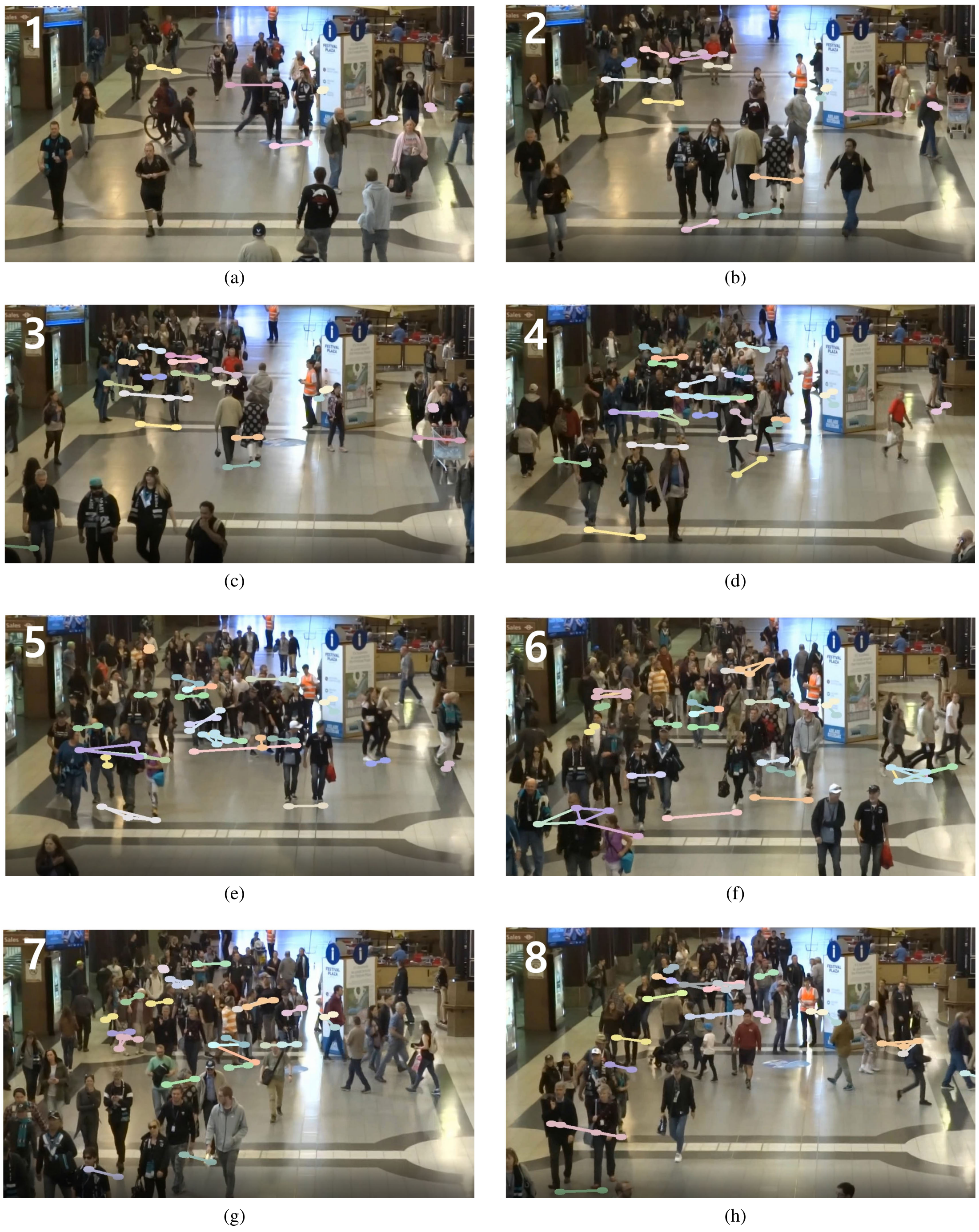


FIGURE 18. The sample frames for the visualized results of a group tracking query over MOT02GT. The people in the same group are connected by same-colored line segments.

the groups in the *GMG* candidate set. Therefore, the maximal number of set operations was similar in all the methods.

4) ANALYSIS FOR A GROUP TRACK MAINTENANCE STEP

Figure 16(a) shows the minimum, average and maximum number of *GTR*s in a *GTR* maintenance step for the methods *N-TRK* and *P-TRK*. The number of *GTR*s was measured after the expiry phase in each frame. Because the number of *GTR*s tended to increase exponentially as frames are processed, relatively few *GTR*s existed at the first frame of a video. Therefore, The minimum numbers of *GTR*s were similar. In the average and the maximum number of *GTR*s, *P-TRK* shows about 36% and 53% fewer *GTR*s than *N-TRK*, respectively. It was because *N-TRK* made a lot of unnecessary *GTR*s. Meanwhile, *P-TRK* reduced the creation of unnecessary *GTR*s by exploiting the appearance information of a group track. Accordingly, in Figure 16(b), *P-TRK* without a set-trie (*P-TRK-NOTRIE*) showed about 49% lower maintenance time than *N-TRK*. Our *GTR* maintenance already achieved significant performance gain without a set-trie. *P-TRK* reduced the maintenance time about seven times compared to *P-TRK-NOTRIE* by accelerating set operations via a set-trie in *GTR* maintenance.

5) COMPARISONS WITH LEARNING-BASED APPROACH

We conduct visual comparisons for group tracking results between a learning-based approach and the proposed method. For the learning-based approach, we used the work [12] which shows the highest accuracy among the works [1], [5], [6], [7], [8], [9], [10], [11], [12]. Figure 17(a) and 17(c) are captured images from a demo video [27] of the work [12]. The demo video visualized group tracking results on GVEII dataset [28]. GVEII dataset contains 2400 frames of video and ground truths for both groups and individuals. The average number of objects per frame is 49, which is similar to MOT02 dataset. For the proposed method, the predicates of group tracking query are set to get similar results to those of the learning-based method. The images in Figure 17(b) and 17(d) are visualized results of the group tracking query on GVEII dataset. Figure 17(b) and 17(d) depict the same frame as 17(a) and 17(c), respectively. Overall, the two methods find similar groups except for two situations (red dashed circle) that are labeled with the numbers ① and ②. In situation ①, a group of two people in column formation passes through between two people which are not in a group. The group tracking query considered the situation, while the learning-based approach did not. In situation ②, four people in a group walk in a large formation. As opposed to situation ①, the group tracking query found the results in several small groups, while the learning-based approach found the group with four people. As shown in the two situations, although the two methods showed similar group tracking results, failure cases differed according to the situations in a video. Therefore, it is challenging to determine the superiority of effectiveness between the two methods.

In the aspect of efficiency, the models of learning-based approaches have much longer inference time than the total running time of a video as we mentioned in Section II. Besides the inference time, labeling for ground truths, feature (e.g., motion patterns of groups, etc.) extraction, and training time are also needed. On the other hand, query processing time is much shorter than the total running time of a video as shown in Section VIII-B1. Therefore, for video monitoring systems, group tracking query is more practical than the learning-based approaches because fast discovery of groups is required.

6) VISUALIZATIONS OF GROUP TRACKING QUERY RESULT

In Figure 18, we visualized the results of a group tracking query with default parameters on MOT02GT dataset. Because it was difficult to identify groups in the whole video frame, the frames were cropped and enlarged. The frames are captured in approximately five seconds intervals (125 frames interval) and labeled with the integer at the left top corner of a frame in the order of frame number. A person in Figure 18 is depicted by a circle at the feet point, and the people in the same group are connected by same-colored line segments. In the figure, found groups are usually reasonable regardless of (1) group locations at a frame and (2) object overlaps that cause object occlusion and temporary groups.

IX. CONCLUSION

Recently, as video monitoring systems utilize object tracking on crowded videos, a lot of crowd tracking data are produced. Over crowd tracking data, there is a necessity for finding human groups rapidly. However, to the best of our knowledge, there is no existing work that finds groups over crowd tracking data. We proposed a novel type of query, *group tracking query*, which retrieves the groups that satisfy spatial proximity, temporal consistency, and the minimum group size. By the proximity definition based on a variable ellipse, a group tracking query considers perspective projection. Moreover, the groups which occlude temporarily can be found because a group tracking query considers the occlusion threshold. We also proposed efficient algorithms for a group tracking query processing. The proposed algorithms significantly reduced the computational cost for both group detection and group track maintenance step. Through the extensive experimental evaluations, the efficiency and effectiveness of a group tracking query were verified. A retrieval of groups over crowd tracking data which have changing camera location and angle can be the future work.

REFERENCES

[1] A. Bera and D. Manocha, "Realtime multilevel crowd tracking using reciprocal velocity obstacles," in *Proc. 22nd Int. Conf. Pattern Recognit.*, Aug. 2014, pp. 4164–4169.

[2] *BriefCam*. Accessed: Jan. 31, 2023. [Online]. Available: https://www.briefcam.com

[3] *Viisight*. Accessed: Jan. 31, 2023. [Online]. Available: https://www.viisights.com

[4] *CDC Close-Contact*. Accessed: Jan. 31, 2023. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/php/contact-tracing/contact-tracing-plan/appendix.html

[5] J. Shao, N. Dong, and Q. Zhao, "A real-time algorithm for small group detection in medium density crowds," *Pattern Recognit. Image Anal.*, vol. 28, no. 2, pp. 282–287, Apr. 2018.

[6] M. Bendali-Braham, J. Weber, G. Forestier, L. Idoumghar, and P.-A. Müller, "Recent trends in crowd analysis: A review," *Mach. Learn. Appl.*, vol. 4, Jun. 2021, Art. no. 100023.

[7] W. Ge, R. T. Collins, and B. Ruback, "Automatically detecting the small group structure of a crowd," in *Proc. Workshop Appl. Comput. Vis. (WACV)*, Dec. 2009, pp. 1–8.

[8] R. Mazzon, F. Poiesi, and A. Cavallaro, "Detection and tracking of groups in crowd," in *Proc. 10th IEEE Int. Conf. Adv. Video Signal Based Surveill.*, Aug. 2013, pp. 202–207.

[9] J. Shao, N. Dong, and Q. Zhao, "An adaptive clustering approach for group detection in the crowd," in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Sep. 2015, pp. 77–80.

[10] M. Chen, Q. Wang, and X. Li, "Anchor-based group detection in crowd scenes," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 1378–1382.

[11] Q. Ma, Q. Zou, N. Wang, Q. Guan, and Y. Pei, "Looking ahead: Joint small group detection and tracking in crowd scenes," *J. Vis. Commun. Image Represent.*, vol. 72, Oct. 2020, Art. no. 102876.

[12] F. Solera, S. Calderara, and R. Cucchiara, "Socially constrained structural learning for groups detection in crowd," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 5, pp. 995–1008, Aug. 2016.

[13] M. R. Vieira, P. Bakalov, and V. J. Tsotras, "On-line discovery of flock patterns in spatio-temporal data," in *Proc. 17th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, Nov. 2009, pp. 286–295.

[14] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen, "Discovery of convoys in trajectory databases," 2010, *arXiv:1002.0963*.

[15] L.-A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C.-C. Hung, and W.-C. Peng, "On discovery of traveling companions from streaming trajectories," in *Proc. IEEE 28th Int. Conf. Data Eng.*, Apr. 2012, pp. 186–197.

[16] P. Kalnis, N. Mamoulis, and S. Bakiras, "On discovering moving clusters in spatio-temporal data," in *Proc. Int. Symp. Spatial Temporal Databases*. Cham, Switzerland: Springer, 2005, pp. 364–381.

[17] Q. Zou, H. Ling, Y. Pang, Y. Huang, and M. Tian, "Joint headlight pairing and vehicle tracking by weighted set packing in nighttime traffic videos," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1950–1961, Jun. 2018.

[18] Y. Chen, X. Yu, N. Koudas, and Z. Yu, "Evaluating temporal queries over video feeds," in *Proc. Int. Conf. Manage. Data*, Jun. 2021, pp. 287–299.

[19] V. Fremont and R. Chellali, "Direct camera calibration using two concentric circles from a single view," in *Proc. Int. Conf. Artif. Reality Telexistence*, 2002, pp. 93–98.

[20] S. Su, Y. Luo, K. Yang, Z. Gao, Y. Zhao, X. Zhao, and G. Song, "A novel camera calibration method based on multilevel-edge-fitting ellipse-shaped analytical model," *IEEE Sensors J.*, vol. 20, no. 11, pp. 5818–5826, Jun. 2020.

[21] I. Savnik, "Index data structure for fast subset and superset queries," in *Proc. Int. Conf. Availability, Rel., Secur.* Cham, Switzerland: Springer, 2013, pp. 134–148.

[22] I. Savnik, M. Akulich, M. Krnc, and R. Škrekovski, "Data structure *set-trie* for storing and querying sets: Theoretical and empirical analysis," *PLoS ONE*, vol. 16, no. 2, Feb. 2021, Art. no. e0245122.

[23] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "MOT20: A benchmark for multi object tracking in crowded scenes," 2020, *arXiv:2003.09003*.

[24] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," 2016, *arXiv:1603.00831*.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[26] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3645–3649.

[27] F. Solera. (Aug. 12, 2014). *Structured Learning for Social Group Detection in Crowdsgveii*. Accessed: Jan. 31, 2023. [Online]. Available: https://www.youtube.com/watch?v=wl6Js91HXOc

[28] S. Bandini, A. Gorrini, and G. Vizzari, "Towards an integrated approach to crowd analysis and crowd synthesis: A case study and first results," *Pattern Recognit. Lett.*, vol. 44, pp. 16–29, Jan. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865513003760

HYUNSIK YOON received the B.S. degree from the Department of Computer Science and Engineering, Korea University, Seoul, South Korea, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering. His research interests include spatio-temporal query processing, spatial data management, in-memory data management, and machine learning for databases.

DALSU CHOI received the B.S. degree from the Department of Computer Science and Engineering, Korea University, Seoul, South Korea, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering. His research interests include distributed query processing, spatial data management, in-memory data management, array data management, and machine learning for databases.

YON DOHN CHUNG (Member, IEEE) received the B.S. degree in computer science from Korea University, Seoul, South Korea, in 1994, and the M.S. and Ph.D. degrees in computer science from KAIST, Daejeon, South Korea, in 1996 and 2000, respectively. He was an Assistant Professor with the Department of Computer Engineering, Dongguk University, Seoul, from 2003 to 2006. He joined as a Faculty Member of the Department of Computer Science and Engineering, Korea University, in 2006, where he is currently a Professor. His research interests include database systems, spatial databases, and data privacy.