Reliability Groups with Standby Flying Light Specks

Hamed Alimohammadzadeh, Shuqin Zhu, Jiadong Bai, Shahram Ghandeharizadeh
Los Angeles, California, USA

ABSTRACT

A Flying Light Speck, FLS, is a miniature sized drone configured with light sources to illuminate different colors and textures. A swarm of FLSs illuminates complex 3D multimedia shapes in a fixed volume, a 3D display. An FLS is a mechanical device. Its failure is the norm rather than an exception, causing a point of an illumination to go dark. In this paper, we use reliability groups with dark standby FLSs to minimize the duration of time a point remains dark. This study makes two novel contributions. First, it compares a centralized and a decentralized algorithm to form groups, demonstrating the superiority of the centralized technique. Second, it detects when the dark standby FLSs may obstruct the user's field of view and relocates them with minimal impact on their provided benefit.

CCS CONCEPTS

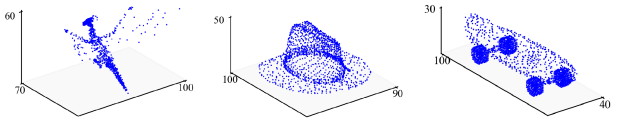
• **General and reference** → *Reliability*; • **Human-centered computing** → *Visualization design and evaluation methods*; • **Computing methodologies** → **Graphics systems and interfaces**.

ACM Reference Format:
Hamed Alimohammadzadeh, Shuqin Zhu, Jiadong Bai, Shahram Ghandeharizadeh. 2024. Reliability Groups with Standby Flying Light Specks. In *ACM Multimedia Systems Conference 2024 (MMSys '24), April 15–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3625468.3647606

1 INTRODUCTION

A Flying Light Speck, FLS, is a miniature sized drone equipped with one or more light sources to generate different colors and textures with adjustable brightness. It is battery powered, network enabled, has processing capability to implement decentralized algorithms, and some storage. It uses the latter to store data pertaining to its lighting responsibility at certain coordinates as a function of time. Synchronized swarms of FLSs will illuminate virtual objects in a prespecified 3D volume, an FLS display [17, 18]. An FLS display may be a cuboid that sits on a table or hangs on a wall, the dashboard of a self-driving vehicle, a room, etc. It will render (a) static 2D or 3D point clouds, see Figure 1, (b) slide shows consisting of a sequence of 2D or 3D point clouds, (c) motion illuminations consisting of a sequence of 2D or 3D point clouds with temporal constraints, and (d) interactive illuminations for games.

An FLS display may be used by diverse applications ranging from entertainment to healthcare, education, scientific devices (e.g., deep space telescopes) that produce 3D data among others. With healthcare, an FLS display may illuminate 3D MRI scans of a patient and their organs, empowering a physician to separate the different organs (illuminations) and examine them in real time. With entertainment, an FLS display may illuminate characters of multi-player games such as Minecraft and Fortnite to come alive and interact with one another on a table top. Visualization of 3D scientific data is a significant challenge. An immersive FLS display addresses the challenge by enabling a user to analyze the data from different perspectives including stepping into the data, e.g., examining the 3D voxels of a hippocampus in an MRI scan.



(a) Dragon, 760. **(b) Hat, 1562.** **(c) Skateboard, 1727.**

Figure 1: Three point clouds and their number of FLSs [4].

An FLS is a mechanical device that fails [17, 18]. It may encounter different kinds of failures ranging from propeller damage [23] and rotor loss [21] that prevent it from flying to hardware and software failures that prevent it from communicating and coordinating with other FLSs. An FLS that must leave its swarm to charge its battery may also be considered as a form of failure. To elaborate, STAG [18] constructs flocks with staggered flight times, allowing an FLS F_d with an almost depleted battery to leave its flock to charge its battery while a dispatcher deploys a new FLS F_f with a fully charged battery to substitute for it. If F_f arrives either prior to or at approximately the same time as F_d's departure then they may switch places gracefully. Otherwise, F_d's absence may be considered as a form of failure that degrades the Quality of Illumination (QoI).

An FLS failure causes a point in an illumination of a point cloud to go dark. Our objective is to minimize the duration of time for a replacement FLS to illuminate this dark point, i.e., the Mean Time to Illuminate a Dark point (MTID). One approach is for the display to deploy a replacement FLS. In this case, MTID is dictated by the distance from a dispatcher to the dark point and the speed of the FLS. An alternative is to deploy dark standby FLSs [18] in anticipation of a failure. These standbys are positioned in close proximity of an illuminating FLS. When an illuminating FLS fails, the standby FLS occupies its coordinates and resumes its lighting responsibility. By minimizing the standby's traveled distance, the MTID is improved.

Our objective is to construct reliability groups, each with G illuminating FLSs and C standby FLSs. The standbys are placed close to the group center. By minimizing the distance between the illuminating FLSs of a group, the travel time of a standby FLS to the coordinates of a failed illuminating FLS is reduced. Figure 2 shows the 95^{th} percentile[1] of the Skateboard's MTID as a function of time with no reliability groups, and reliability groups of G=3

[1]The 50^{th} percentile is almost identical to the 95^{th} percentile.

MMSys '24, April 15–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-0412-3/24/04.
https://doi.org/10.1145/3625468.3647606

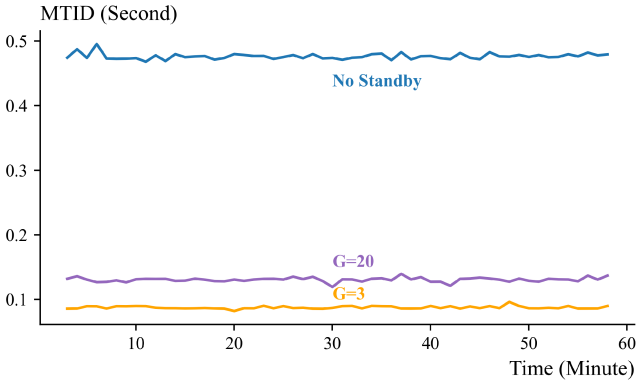


Figure 2: MTID of the Skateboard, 95^{th} percentile, λ=300 seconds, φ=3000, Speed=66.67Meters/Seconds. Lower is better.

and G=20. With G=3 (20), the number of dark standby FLSs is approximately 30% (5%) of the number of points in a point cloud. This is the overhead of using reliability groups. The benefit of the reliability groups is the significant reduction in MTID, see Figure 2.

A standby is dark and may obstruct the user's Field of View (FoV). Figure 3 shows the illuminating and standby FLSs for user's FoV from the bottom of the Dragon and the Skateboard. It shows the obstructing FLSs with G=3 and G=20. Obstructing dark standbys diminish the QoI. There are several possible approaches: Prevention[2], detection, and a hybrid. Due to lack of space, this paper presents detective techniques only.

The **contributions** of this paper include:

- An evaluation of a centralized (k-means [22]) and a decentralized (CANF [3]) technique to form reliability groups for a point cloud illuminated using a swarm of FLSs. k-means is significantly faster and provides a superior MTID when compared with CANF. (Section 4.)
- Insights into when reliability groups do not enhance MTID. We use a simple priority queue that enables a display to disable reliability groups dynamically when they are not beneficial. (Section 5.)
- We detect obstructing standby FLSs and present techniques that change their location to avoid obstruction. (Section 6.)
- We open source our software implementations, and the 3D data used in this study for use by the scientific community. See https://github.com/flyinglightspeck/StandbyFLSs.

Table 1 shows the notations used in this paper and their definitions. The rest of this paper is organized as follows. Section 2 provides related work. Section 3 presents an overview of a system and the terminology used in this paper. Sections 4-6 were detailed as contributions. Brief conclusions and future research directions are presented in Section 7.

2 RELATED WORK

The concept of FLSs and FLS displays are presented in [2–5, 10, 11, 17–19, 27, 36, 37]. The idea to form reliability groups and use standby FLSs is introduced in [18]. It presents analytical models to demonstrate the benefits of using standby FLSs. However, it does not describe techniques to form reliability groups. Our study is different because it formally introduces and quantifies QoI and the average duration of time a point is dark (MTID), introduces a centralized and a decentralized technique to form groups and quantifies their tradeoffs, identifies when reliability groups do not enhance MTID, a technique to detect dark standbys obstructing

[2]Computes a non-obstructing location for the dark standby FLSs.

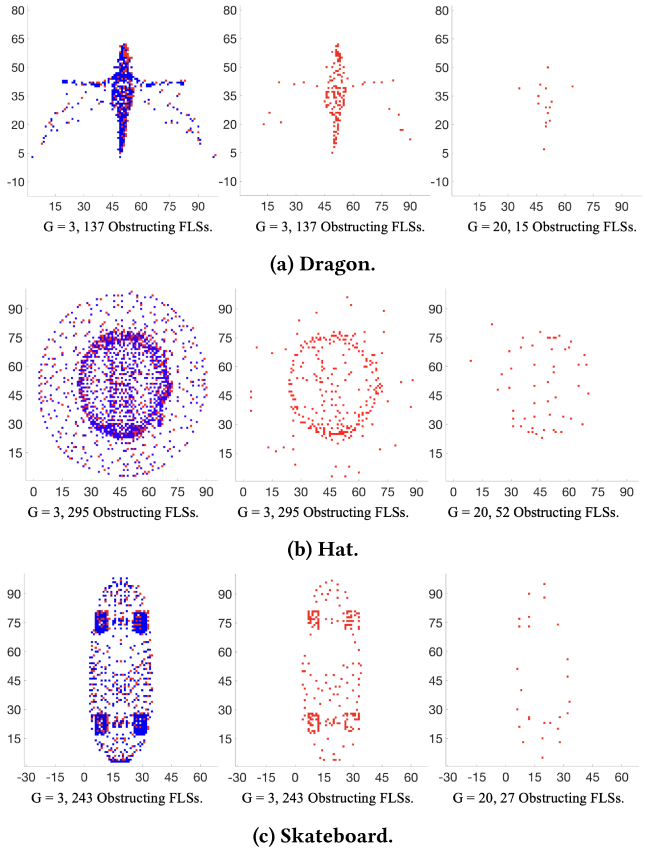


Figure 3: Bottom view of shapes. The leftmost figure shows illuminating (blue) FLSs with some obstructed by a dark standby (red) FLS. The other two figures show the obstructing standby (red) FLSs only.

Table 1: Notations and their definitions.

Notation	Definition
Q	Ratio of illumination cell to display cell.
D	Total number of dispatchers.
F	Total number of FLSs required by a point cloud.
G	Number of FLSs per reliability group.
C	Number of standby FLSs per reliability group.
n_G	Number of groups $n_G = \lfloor \frac{F}{G} \rfloor$.
MTID	Mean time to illuminate a dark point due to FLS failure.
MTTF	Average amount of time an FLS operates before it fails.
λ	Maximum lifetime of an FLS.
φ	Maximum rate of FLS deployment.
ϱ	Maximum rate of FLS failures, $\varrho = \frac{F}{\lambda}$.

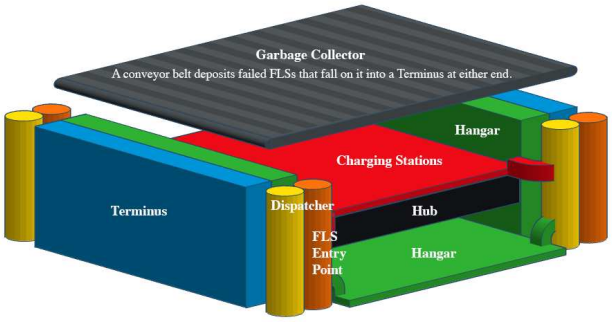


Figure 4: The yellow cylinders of the simple architecture [18] are dispatchers that deploy FLSs. A dispatcher deploys FLSs at a rate φ and a display consists of D dispatchers. Failed FLSs fall on a conveyor belt that deposits them in a Terminus.

the user's FoV, and how a standby changes its location to not obstruct the user's FoV. These novel contributions are absent from the aforementioned studies.

The concept of reliability groups [14, 15, 30, 31] and standbys dates back to arrays of memory modules and disks [20, 29]. With disk subsystems such as RAID [26], the hardware infrastructure dictates the characteristics of the reliability groups. FLS displays are different in that the data in the form of a point cloud dictates the characteristics of the reliability groups. Distance between the points that constitute a reliability group is an important consideration as it dictates MTID and QoI. This concept is absent from prior literature in the area of high data availability. Another difference is that, an FLS display may use reliability groups selectivity to enhance MTID. As detailed in Section 5, a simple priority queue enables an FLS display to not deploy standby FLSs, disabling the concept of a reliability group when it is not beneficial. Hardware and software RAIDs do not require this flexibility. Finally, data availability techniques do not have the concept of a standby obstructing a user's FoV. This important consideration is addressed in this paper.

3 TERMINOLOGY AND PROBLEM STATEMENT

We assume the architectures of [2, 18] consisting of a hub and one or more dispatchers to deploy FLSs. See Figure 4. The hub is comparable to today's server. It hosts an Orchestrator process that communicates with FLSs using wireless communication, detects failures, and coordinates the dispatchers to deploy FLSs. A dispatcher may be in the form of a cylinder on the sides of a display[3] [18]. The rate at which an architecture deploys FLSs impacts MTID.

Display cell and illumination cell are fundamental to this study. To render an illumination, an FLS display constructs a 3D mesh on the display volume. A cell of this mesh, a *display cell* [18], is dictated by the downwash of an FLS [7, 8, 16, 28, 35]. Assuming an FLS is a quadrotor, a cell may be an ellipsoid [7, 8, 28] or a cylinders [16, 35] that results in a larger separation along the height dimension. Each display cell has a unique Length, Height, and Depth (L,H,D) coordinate [18]. We use the L, H, D coordinate system instead of X, Y, Z to identify a cell because there is no consensus on one definition of the Y and Z axes. While the picture industry uses the Z axis as the depth, mathematicians use the Y axis for the depth. It is trivial to map our L, H, D coordinate system to either definition without ambiguity.

The light emitted from an FLS fills an *illumination* cell. We assume an illumination cell is either the same size or larger than a display cell. Without loss of generality and to simplify the discussion, we assume a display cell and an illumination cell are cubes. We denote the ratio of an illumination cell to a display cell along a dimension as Q. Assuming the same Q value across L, H, and D, the number of display cells contained in an illumination cell is Q^3. A maximum of Q^3 FLSs may occupy different cells of an illumination cell with one FLS illuminating the cell. We assume the remaining Q^3-1 FLSs remain invisible. This assumption is reasonable because these FLSs may render the same light and color as the illuminating FLS with lower intensity depending on the location of their occupied display cell.

An increase in the value of Q may be interpreted in different ways. One interpretation is that the display is increasing in size and the light source of each FLS is bright enough to illuminate an increasing number of display cells. For example, with Q=10, an FLS must illuminate 100 displays cells and the display is 10x larger along a dimension when compared with Q=1. The second interpretation is that the volume of downwash by an FLS has shrunk. Q=10 implies this volume shrunk 10x when compared with Q=1, enabling the display to construct a larger number of FLSs in an illumination cell. In this paper, we assume the first interpretation of Q. It is trivial to convert the results of this paper in the context of the second interpretation[4].

Definition 1. Quality of Illumination (**QoI**) of a point cloud is the percentage of its points illuminated by FLSs at an instance in time.

The best theoretical QoI value is 100%. The QoI of a rendering may be lower than 100% due to failure of the illuminating FLSs. The tolerable QoI is application dependent.

We use Mean Time To Failure (MTTF) [18, 26, 29], to quantify the average amount of time an FLS operates before it fails. A higher MTTF is desirable because it reflects a longer FLS lifespan. We construct *reliability* groups consisting of G illuminating FLSs and C standby FLSs. A *standby* is a dark FLS assigned to a group of G illuminating FLSs. The distance between the G FLSs should be minimized to enable the standby to substitute for a failed FLS as fast as possible, minimizing the MTID.

Definition 2. Mean Time to Illuminate a Dark point (**MTID**) is the average elapsed time from when a point of a point cloud goes dark due to the failure of its illuminating FLS and back to being illuminated by a replacement FLS.

Definition 3. A *group centroid* is the coordinate that identifies the center of a reliability group consisting of G FLSs. It is computed using the coordinates of its G FLSs: $[\frac{1}{G}\sum_{i=1}^{G} L_i, \frac{1}{G}\sum_{i=1}^{G} H_i, \frac{1}{G}\sum_{i=1}^{G} D_i]$, where $[L_i, H_i, D_i]$ is the coordinate of the i^{th} FLS in the group. The

[3] Or black tiles at the bottom of a Dronevision [2].

[4] The acceleration and deceleration components of the velocity model may impact the conversion of the results between the two interpretations.

Orchestrator assigns a standby to the group centroid as long as it is not occupied by an illuminating FLS, i.e., the coordinates of a point in a point cloud.

Problem Definition 1. *Minimize the MTID of an illumination given reliability groups with G illuminating FLSs and C standby FLS without obstructing the user's FoV.*

3.1 Failure types

We assume FLS failures are either mechanical, lighting related, or both. These prevent an FLS from either flying, rendering its illumination responsibilities, or both. In addition, we assume an FLS may detect these failures and use its wireless communication interface to notify its neighboring FLSs and the Orchestrator of its failure. We defer failure of electronics that prevent an FLSs from communicating to future work.

When an illuminating FLS fails, its assigned point (coordinate) goes dark. The display may illuminate this point by either requiring a dispatcher to deploy a new FLS to the coordinates of the dark point or use a nearby standby to fly to the coordinates of the dark point (failed FLS) to resume its lighting responsibilities. The latter minimizes MTID, motivating nG reliability groups with C standby FLSs per group.

With reliability groups, we assume an illuminating and a standby FLS are identical with the same MTTF. A reliability group may be in either the *normal* or a *degraded* mode of operation. In the normal mode, the group consists of G illuminating FLSs and C standbys. In degraded mode, either one or more illuminating FLSs have failed, one or more standby FLSs have failed or are missing (because they substituted for a failed illuminating FLS), or a combination of these possibilities.

When an illuminating FLS fails, it notifies its group members and the Orchestrator of its failure. Its message identifies the group that experienced the failure, G_f. If G_f has an idle standby FLS, this FLS changes its status to illuminating and flies to the coordinate of the failed FLS to assume its lighting responsibility. The Orchestrator dispatches a new standby to G_f. The coordinates of this standby is G_f's centroid.

When a standby FLS fails, it notifies the G illuminating FLSs and the $C-1$ standbys in its group and the Orchestrator of its failure. The illuminating FLSs maintain the absence of a standby and switch to degraded mode of operation. The Orchestrator dispatches a new FLSs to the coordinates identified by the group centroid as the replacement standby. In case the replacement standby fails mid-flight, it notifies the Orchestrator to deploy a new replacement FLS. Once a replacement standby is dispatched, it notifies the illuminating FLSs in its group of its presence. The illuminating FLSs record the presence of the standby and switch to normal mode of operation.

If G_f has no standby FLS and one of its illuminating FLSs fails, either the Orchestrator or the failed FLS may notify a mid-flight replacement standby for the group to change its destination to the coordinates of the failed illuminating FLS and resume its lighting responsibility. In essence, this mid-flight standby becomes an illuminating FLS on its way to illuminate its assigned point. Next, the Orchestrator deploys a new standby for G_f. The coordinates of this standby is G_f's group centroid.

With FLS failures, the number of dispatchers and the rate at which they deploy FLSs are important factors. With fully utilized dispatchers, standby FLSs may degrade both QoI and MTID. This is because standby FLSs fail similar to illuminating FLSs. Deployment of standby FLSs taxes the dispatchers and prevents them from deploying FLSs to illuminate points that improve QoI/MTID. Below, we formalize these concepts and highlight their interrelationships.

Failure Model: We assume a simple failure model with FLSs failing independent of one another. This model selects a time to live (TTL) between 1 second and a fixed constant λ. The life time of an FLS may not exceed λ, TTL $\leq \lambda$. Once this TTL expires, an FLS fails. The distribution used to select a TTL impacts QoI. We considered several distributions for a model. Due to lack of space, we present the model using a uniform distribution. With this model, an FLS selects a random value between 1 and λ and approximates a mean of the distribution close to $\frac{\lambda}{2}$.

4 GROUP CONSTRUCTION

Problem Definition 2. *Given a point cloud with F points (coordinates) with each point assigned to a different FLS and a required group size G, maximize the number of groups consisting of approximately G FLSs such that (1) an FLS participates in exclusively one group and all FLSs agree on participation in the same group, and (2) the euclidean distance between the group members is minimized.*

In this paper, the symmetry constraint that all G FLSs in a group agree to be a member of the same group is fundamental. We focus on a popular centralized group formation technique named k-means [22] and a decentralized technique named Closest Available Neighbor First, CANF [3]. With both, the number of groups nG is bounded by $\lceil \frac{F}{G} \rceil$. At the same time, the two techniques are different. While k-means constructs nG groups consisting of a variable number of illuminating FLSs, CANF constructs nG groups each with G FLSs[5].

4.1 Centralized Group Construction: k-Means

With a centralized technique, the Orchestrator processes a point cloud to compute nG groups and the location of each of the C standbys for each group. It uses k-means [22], a clustering technique that partitions a point cloud with F points into k groups defined by a centroid. Given the size of a reliability group G, n_G is defined as $nG = k = \lceil \frac{F}{G} \rceil$. k-means is an iterative heuristic that adjusts the nG new centroid locations in each iteration. The first iteration selects the nG centroids randomly [6]. Each subsequent iteration computes point-to-cluster distance for each centroid by iterating the points in the point cloud. It uses the average of these distances to compute nG new centroid locations. It repeats until either the cluster assignment does not change or a maximum number of iterations (1000) is reached[6]. The iterative portion of this implementation consists of two passes: batch and online. During batch updates, each iteration re-assigns points to their nearest cluster centroid followed by re-computation of cluster centroids. During the online

[5]One group is an exception when $\frac{F}{G}$ is not an integer.

[6]The geometry of a point cloud and the value of G dictates how quickly k-means constructs groups. In our experiments with the point clouds of Figure 1, the number of passes with G=3 (20) and the dragon, hat, and skatebaord is 6, 6, 4 (13, 14, 20), respectively.

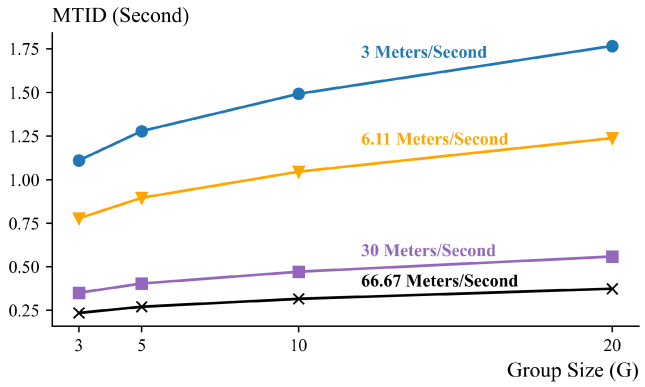


Figure 5: MTID of k-menas (Skateboard) as a function of different group sizes G. Faster FLSs result in lower MTIDs. Lower is better.

updates, individual points are analyzed and reassigned to a different centroid if this reassignment reduces the sum of distances. This pass analyzes all the points.

The Orchestrator schedules one or more dispatchers to deploy the FLSs. Failures are handled per discussion of Section 3.1. The speed of FLSs impacts the observed MTID. Figure 5 shows the MTID as a function of different group sizes, G, with different FLS speeds. Faster FLS speeds result in a lower MTID. The relationship is not linear because of the acceleration and deceleration of an FLS. For example, increasing the speed from 3 Meters/Second to 30 Meters/Second does not reduce the MTID by 10x because the standby FLS must accelerate from a speed of zero and slow down (decelerate) to arrive at its destination.

4.2 Decentralized Group Construction

A decentralized group construction technique requires the Orchestrator to first deploy FLSs. The Orchestrator assigns each FLS a point to illuminate, a coordinate in the ground truth. An FLS may use dead reckoning to arrive at its assigned coordinate and localize relative to its neighbors. In addition, they may exchange messages to implement a decentralized algorithm to form groups. They notify the Orchestrator of their group IDs. In turn, the Orchestrator computes a coordinate close to the center for each of the C standbys and deploys them to those locations. The Orchestrator ensures the assigned location of standby is not occupied by another FLS.

Several decentralized algorithms for group formation are described and presented in [3, 12, 13]. A comparison of SimpleR, CANF, VNS [9] and RS [12] with group sizes as large as 20, $G \leq 20$, is presented in [3]. It shows CANF constructs the most number of groups. With large group sizes, $G \geq 10$, CANF is also faster in constructing the groups[7]. In this paper, we use CANF.

Algorithm 1 shows the pseudo-code of CANF [3]. Each FLS f_i executes CANF independently. It constructs groups with the objective to minimize the distance between the G FLSs that constitute each group. We define $\Delta(f_i, f_j)$ as the distance between two FLSs f_i and f_j. The weight of any two FLSs f_p and f_q in R_i is the reciprocal of their distance: $w(f_p, f_q) = \frac{1}{\Delta(f_p, f_q)}$. This definition of weight is

[7]These prior studies do not compare the decentralized techniques with a centralized technique such as k-means.

Algorithm 1: CANF $(i, P, G, N(f_i))$

```
1  i is the unique identifier of this FLS
2  P is the coordinates of a 3D point of a point cloud assigned to f_i
3  G is the size of the group
4  N(f_i) is a set of f_i's neighboring FLSs
5  R_i ← {}
6  P_t ← 0.2
7  while forever do
8      C ← {}
9      if proper(f_i,R_i) then
10         C ← R_i
11     end
12     if N(f_i) changes or with probability P_t then
13         for f_n ∈ N do
14             if f_i ∈ R_n and
                 Score_i(R_n ∪ {f_n} − {f_i}) > Score_i(R_i) then
15                 C ← R_n ∪ {f_n} − {f_i}
16             end
17         end
18         N(f_i) ← Sort FLSs in N(f_i) in ascending distance in
             Euclidean space
19         C' ← {}
20         for f_u ∈ N(f_i) do
21             if |C'|=G-1 then
22                 break
23             end
               /* Check if f_u has a group or is ceded.   */
24             if R_u is the empty set then
25                 C' = C' ∪ f_u
26             end
27             else if f_i ∈ R_u then
28                 C' = C' ∪ f_u
29             end
30             else
31                 for f_k ∈ R_u do
32                     if Score_i(R_u ∪ f_u − f_k) > Score(R_i) then
33                         C' = C' ∪ f_k
34                     end
35                 end
36             end
37             if |C'| = G − 1 then
38                 break
39             end
40         end
41         if Score_i(C') > Score_i(C) then
42             C ← C'
43         end
44         R_i ← C
45         W_i ← Σ_{p=1}^{G−1} Σ_{q=p+1}^{G} w(f_p, f_q)
46         Transmit W_i to FLS f_j ∈ N(f_i)
47         Transmit R_i to FLS f_m ∈ R_i
48     end
49 end
```

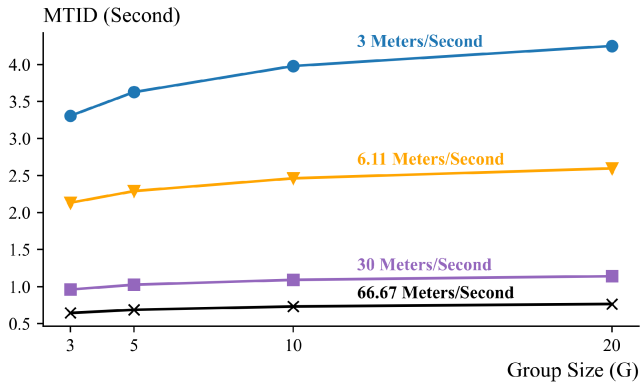


Figure 6: MTID of CANF (Skateboard) as a function of different group sizes G. Faster FLSs result in lower MTIDs. Lower is better.

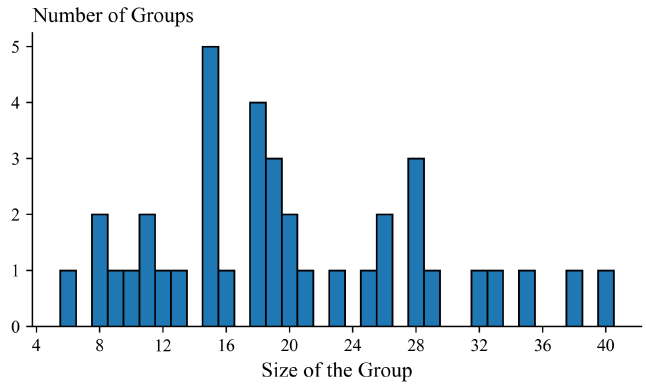


Figure 7: Histogram of k-mean's group sizes with the Dragon and G=20. CANF constructs 38 groups of size G=20.

symmetric. For each FLS f_i, CANF computes set R_i with the objective to maximize the sum of the weights between any 2 FLS in its group: $W_i(R_i \cup f_i) = \sum_{p=1}^{G-1} \sum_{q=p+1}^{G-1} w(f_p, f_q)$.

A group $R_i = \{f_1, \cdots, f_{G-1}\}$ is *proper* [13] if and only if for each neighbor f_j this set has a higher weight than all other groups found by that neighbor. If this is true for all FLSs of R_i then the predicate proper(f_i, R_i) returns true. A score for a set R_i is defined as its weight W_i if the set is proper. Otherwise, its score is zero.

To compute $Score(R_i)$, an FLS f_i exchanges messages with its neighbors, N(f_i). Its message contains its W_i, see Lines 46 and 47 of Algorithm 1. FLS f_i gathers this information from the messages transmitted[8] by its neighbors to compute its Score.

Lines 9-11 of the algorithm prevent an FLS from discarding a good group found in a previous round. They also ensure the protocol converges. Lines 13-17 require an FLS to use a neighbor's transmitted group information (Lines 46-47) to improve the score of its own group. If its score is higher than the existing group then it is stored as R_i, its weight is computed and shared with the neighboring FLSs, and the group R_i is shared with every FLSs that constitutes R_i, Lines 41-47.

Lines 18-40 implement the core functionality of CANF. They show CANF sorts f_i's neighbors in ascending distance using their assigned coordinates. It iterates this list starting with the closest FLS f_u, see Line 20. If f_u does not have f_i as a group member, CANF considers all of f_u's group members as its own, see Lines 31-35. It maintains a local R_u that denotes the proper group found thus far. It uses R_u about f_u to construct a group and decide if its grouping is proper and compute a score. It adds f_u as a candidate only if it results in a higher score.

In addition to the group size, the speed of FLSs impacts the MTID observed with CANF, see Figure 6. This is similar to k-means. Note that the scale of the y-axis with k-means (Figure 5) is more than 2x smaller than CANF (Figure 6), highlighting the superiority of k-means.

[8]These transmissions facilitate discovery of neighboring nodes.

4.3 A Comparison

We implemented k-means and CANF using the Python programming language. While k-means runs on a single server, we emulate the FLSs that execute CANF using a cluster of 16 servers each with 64 cores. In all these studies, we assume the number of standbys for a group is 1, C=1.

A key advantage of CANF is that it is decentralized, enabling FLSs to compute their groups dynamically. A quantitative analysis shows k-means is superior to CANF along several dimensions. First, it constructs groups orders of magnitude faster than CANF. Second, it enhances the overall QoI by minimizing MTID. It realizes this by minimizing the distance between the FLSs that constitute a group.

k-means enhances MTID by constructing nG groups without requiring all groups to have the same size G. For example, Figure 7 shows k-mean's number of constructed groups for different group sizes with the Dragon and G=20. It constructs groups as small as 6 FLSs and as large as 40 FLSs. Most groups consist of 15 FLSs, G=15. This flexibility enables k-means to construct groups with both sparse (Dragon) and dense (Race-car) point clouds effectively such that the FLSs that constitute a group are in close proximity of one another. In turn, a standby FLS substitutes for a failed illuminating FLS quickly to enhance QoI.

Execution time of CANF is orders of magnitude slower than k-means because of its message passing between FLSs, see Figure 8. In our implementation, FLSs communicate using the UDP protocol. k-means does not have this overhead.

k-means becomes slower with larger point clouds. With CANF, the topology of the point cloud dictates its execution time. For example, CANF constructs groups 13x faster with the Skateboard when compared with the Dragon (G=5) even though the Skateboard has more than 2x points.

While k-means becomes faster with larger group sizes, CANF becomes slower. The complexity of CANF increases with larger group sizes because a higher number of FLSs compete with one another to form larger groups. Figure 8 shows the execution time of each algorithm in log scale as a function of G. The slow-down (speedup) with CANF (k-means) is significant. This figure shows k-means is significantly faster than CANF.

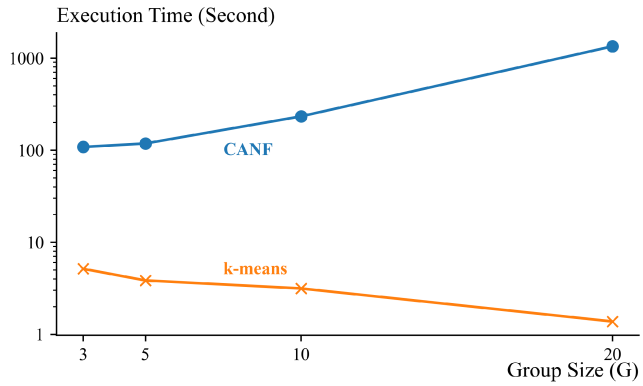


Figure 8: With the Skateboard, k-means is significantly faster than CANF in computing groups. This difference increases with larger group sizes. Y-axis is log scale. Lower is better.

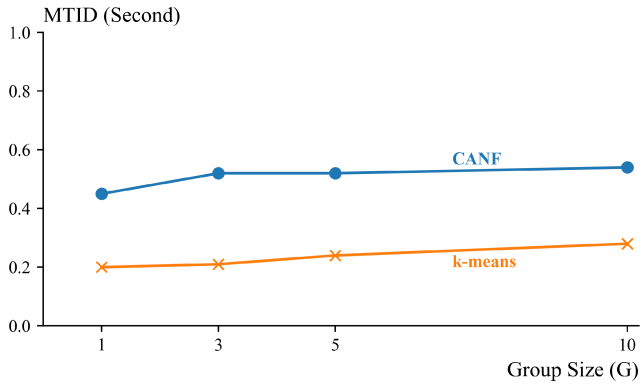


Figure 9: k-means provides a lower MTID when compared with CANF (Skateboard) for all group sizes G. FLS speed is 6.11 Meters/Second. Lower is better.

QoI and MTID of k-means and CANF is the same when the dispatcher is the bottleneck. In this case, the standby FLSs do not provide a benefit (see Section 5 for details) and the distance from the dispatcher to the individual illuminating FLSs (i.e., points of a point cloud) dictate the MTID. This distance is approximately the same with both k-means and CANF. Hence, their identical QoI. For example, with a dispatch rate of 30 FLSs/Second with a MTTF of 30 Seconds, the QoI of the Skateboard is approximately 50% with both CANF and k-means. Their MTID is 28.82 Seconds.

When the dispatcher is not the bottleneck, k-means is superior to CANF. Figure 9 shows k-means results in a lower MTID[9] with all group sizes for the Skateboard. k-means minimizes the distance between the centroid of a group and each FLS that constitutes the group, see Figure 10. This is true with all shapes.

QoI is also impacted by the time for the dispatcher to deploy an FLS to replace a failed standby. During this time, a second illuminating FLS may fail. In this case, there is no standby and the replacement FLS must arrive from the dispatcher. In our experiments, the average distance from a dispatcher to a group is approximately 57 display cells with both k-means and CANF. Hence, its impact is almost identical with both techniques.

[9]Time required for one standby FLS to recover one failure in a reliability group.

5 DISPATCHERS

The rate at which dispatchers deploy FLSs (φ) relative to the failure rate of FLSs (ϱ) dictates the QoI improvements observed with reliability groups. When $\varphi \leq \varrho$, reliability groups may degrade QoI as they require a dispatcher to deploy replacement FLSs for the failed standbys. However, standbys do not enhance QoI. QoI is enhanced when dispatchers deploy replacements for a failed illuminating FLS.

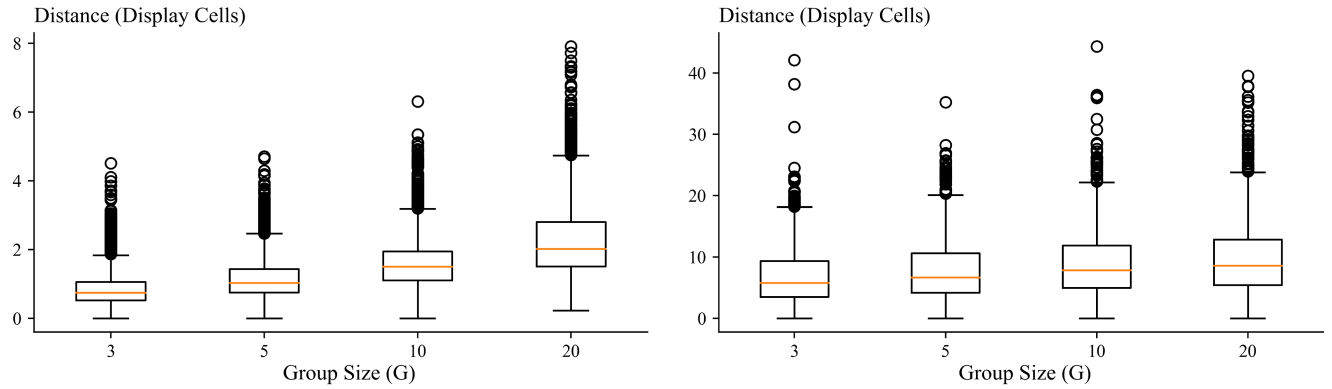
To illustrate, consider an architecture with 1 dispatcher and φ=35 FLSs/Second. With no reliability groups, its QoI is 57.24% and MTID is 21 Seconds. With G=3, QoI is reduced to 49.10% and the MTID increases to 30 Seconds. The replacements for the failed standby FLSs are 33% of the FLSs deployed by the dispatcher. They prevent the dispatcher from deploying replacement FLSs for the illuminating FLSs, resulting in a lower QoI.

We address this limitation using priority queues. We assign a higher priority to replacement FLSs that illuminate a point, moving them to the head of the queue. The replacement standby FLSs may be logical in a data structure maintained by the Orchestrator instead of physical FLSs. In our example, use of priority queues enhances QoI to 57.67% and reduces MTID to 21.3 Seconds. These numbers are comparable to those with no reliability groups. For the remainder of this paper, we assume dispatchers use priority queues.

5.1 Multiple Dispatchers

One may use D dispatchers to increase the overall rate of FLS deployment to $D \times \varphi$. A challenge is how the Orchestrator should select a dispatcher to deploy an FLS. We considered four policies: Random, Round-Robin (RR), Power-of-2 [24], and Shortest Distance First (SDF). With Random, the Orchestrator selects a dispatcher randomly. With RR, the Orchestrator uses a simple mod function by maintaining a monotonically increasing counter C. It selects the dispatcher with id (C *mod* D) and increments C by one. With Power-of-2 [24], the Orchestrator samples the queue of two randomly chosen dispatchers and selects the one with the shortest queue. With SDF, the Orchestrator computes the distance from a dispatcher to the coordinates of the FLS to be deployed, selecting the one with the shortest distance. It is possible to combine two or more techniques. For example, a hybrid may require the Orchestrator to select a dispatcher using SDF and, if it has a queue then, use power-of-2 to select a dispatcher.

We evaluated these policies with alternative shapes. Obtained results show the following lessons. When $\varphi \leq \varrho$, SDF is inferior to the other techniques. It may results in formation of a queue at a dispatcher while other dispatchers sit idle waiting for work. When $\varphi > \varrho$ and with no reliability group, the example hybrid technique provides a slightly better QoI ($\approx$ 1%), MTID ($\approx$ 1%), and a lower traveled distance (1 to 2 display cells) when compared with the other techniques. When $\varphi > \varrho$ and with reliability groups, except for SDF, other techniques provide comparable QoI and MTID. In most experiments, RR is slightly superior. In all experiments, SDF provided a significantly lower QoI and MTID.



(a) k-means. **(b) CANF.**

Figure 10: Box plot of the distance from a group centroid to an illuminating FLS in the group with the Skateboard. The orange line is the median, the box highlights the first and third quartiles, the whiskers are a distance of 1.5 times the quartile, and the rest are outliers. The y-axis scale with k-means is 5x smaller than CANF. Lower is better.

Below, we present results from the Skateboard and the Dragon to highlight the above lessons. We assume an architecture with D=3 dispatchers. Each dispatcher may deploy φ=10 FLSs/Sec and FLS speed is 6.11 Meters/Second. With the Dragon, the failure rate is 12 FLSs/Sec. To highlight the difference between the policies, we remove the standbys by conducting the experiment with no reliability groups. In this experiment, the hybrid technique provides the best QoI (MTID) of 95.39% (1.46). SD provides the worst QoI (MTID) 41.84% (37.96), resulting in formation of a queue that processes the highest number of concurrent dispatched FLSs with coordinates closest to that queue. The queuing delay increases MTID and reduces QoI. In this experiment, the average queuing delay of SD is 36.79 Seconds compared with ≈ 0.3 Second with other techniques.

With both the Dragon and the Skateboard and no reliability groups, the hybrid of SD with power-of-2 provided a slightly better QoI and MTID. For example, with the Skateboard, the hybrid provided a QoI (MTID) of 97.64% (1.30) versus power-of-2 with QoI (MTID) of 95.37% (1.33). In addition, its average traveled distance (32.69) was lower (34.45). This difference does not exist with the reliability groups. One explanation is that the different policies are heuristics and their observed performance benefit depends on the number of FLSs, the topology of the point cloud, and how FLSs fail.

6 OBSTRUCTING FLS

A dark standby may obstruct the user's FoV, see Figure 11.a. The number of obstructing FLSs is impacted by Q, i.e., the ratio of an illumination cell to a display cell. See Figure 14a and discussions of Section 3. Figure 12 shows the number of obstructing FLSs with G=3 and G=20 for different Q values for the top view of the Skateboard. A higher Q value increases the likelihood of a dark standby either being placed in a display cell of an illumination cell or eclipsed by several illuminating cells from different angles. Figure 13 highlights the obstructing FLSs with a red square for the different shapes from their right side with Q=10. The number of obstructing FLSs is higher with G=3 when compared with G=20 because it has approximately 7x more standby FLSs.

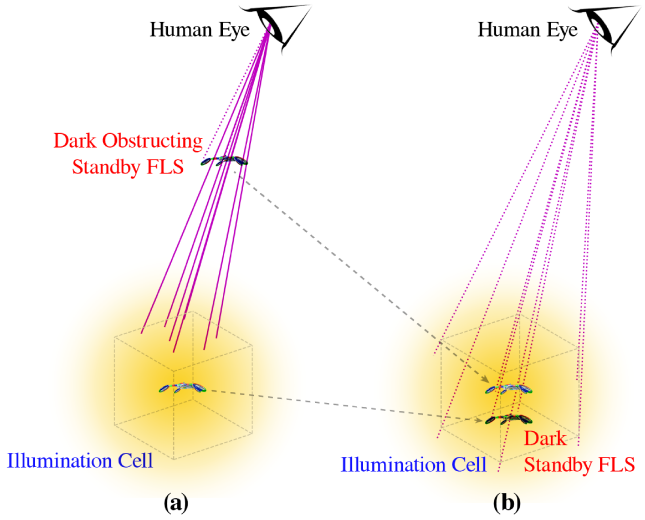


Figure 11: (a) A dark standby obstructs the user's FoV. (b) Suspend may move the obstructing FLS to the location of the illuminating FLS to assume its lighting responsibility. The illuminating FLS goes dark, moves one display cell behind the new illuminating FLS, and becomes the new standby.

An FLS display may track the human gaze using cameras [1, 34], computing the coordinates of the user's gaze U. Ray tracing [33] computes a vector V from U to an illuminating FLS f_i, see Figure 11.a. A dark standby that intersects V and falls in-between U and f_i is an obstructing FLS as long as it is not in a display cell of an illumination cell.

The Orchestrator may monitor the user's movement to compute a set of future locations $\{U_f\}$ relative to their current coordinate U. It may use different techniques to prevent the standby FLS from obstructing. This section presents two. First, the Orchestrator may dissolve the reliability group of an obstructing standby and require

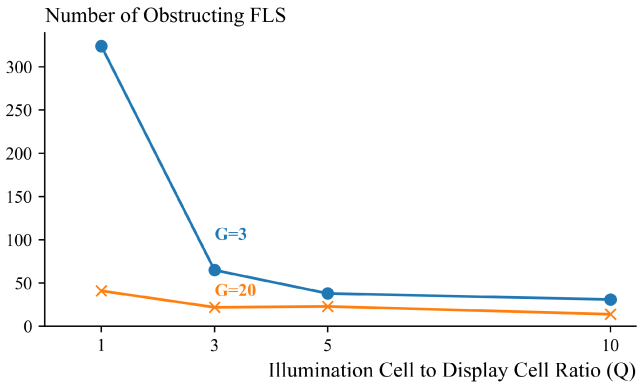
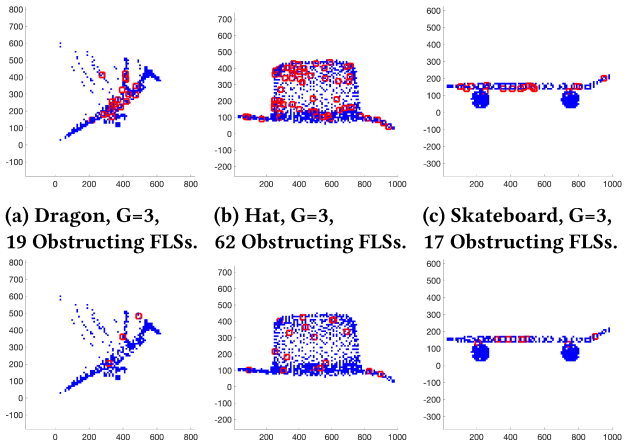


Figure 12: Number of Obstructing FLSs as a function of Q with different group sizes, Skateboard, top view.



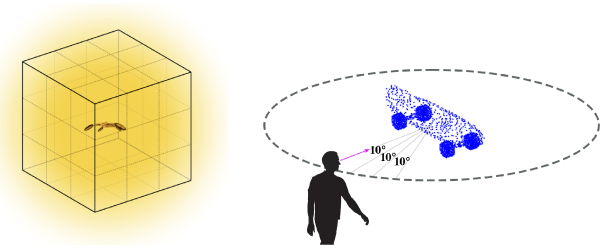
(a) Dragon, G=3, 19 Obstructing FLSs. **(b) Hat, G=3, 62 Obstructing FLSs.** **(c) Skateboard, G=3, 17 Obstructing FLSs.**

(d) Dragon, G=20, 3 Obstructing FLSs. **(e) Hat, G=20, 17 Obstructing FLSs.** **(f) Skateboard, G=20, 7 Obstructing FLSs.**

Figure 13: Three point clouds, user's FoV is from the right side, G={3, 20}, Q=10. The distance between the user and the shape is 100 display cells.

the standby to fly to a hangar for permanent storage. Second, the Orchestrator may suspend the reliability group of an obstructing FLS and reactivate it once the user moves such that the standby is no longer obstructing. Both increase MTID.

Below, we describe the two techniques in turn. This includes results from a scenario where a user at coordinate U walks around an illumination at 10° intervals, and completes a 360° circle by returning to their starting coordinate U, see Figure 14b. A key metric is the percentage increase in MTID when compared with the scenario where the obstructing FLSs are present. We compute this metric by quantifying the MTID with obstructing FLSs, $MTID_O$. Subsequently, we quantify the MTID after removing the obstructing standbys, $MTID_N$. The percentage increase is $100 \times \frac{MTID_N - MTID_O}{MTID_O}$.



(a) Q=3. **(b) 10° intervals.**

Figure 14: (a) An illumination cell with ratio Q=3 along each dimension, (b) a user circling an illumination at 10° intervals.

6.1 Dissolve

Dissolve is a technique that deactivates the reliability group of a standby FLS that obstructs the user's FoV, requiring the standby to fly to a hanger for permanent storage. The geometry of a shape and the value of Q impact the number of dissolved reliability groups. Figure 15a shows the percentage of reliability groups dissolved as a user walks around the Skateboard. The x-axis of this figure is the movement of the user at 10° increments. The y-axis is the cumulative percentage of dissolved reliability groups. A large percentage are dissolved with Q=1. The percentage is lower with Q=5 and 10 because there is a higher probability of a dark standby either occupying a display cell of an illumination cell or being eclipsed by an illumination cell from all angles.

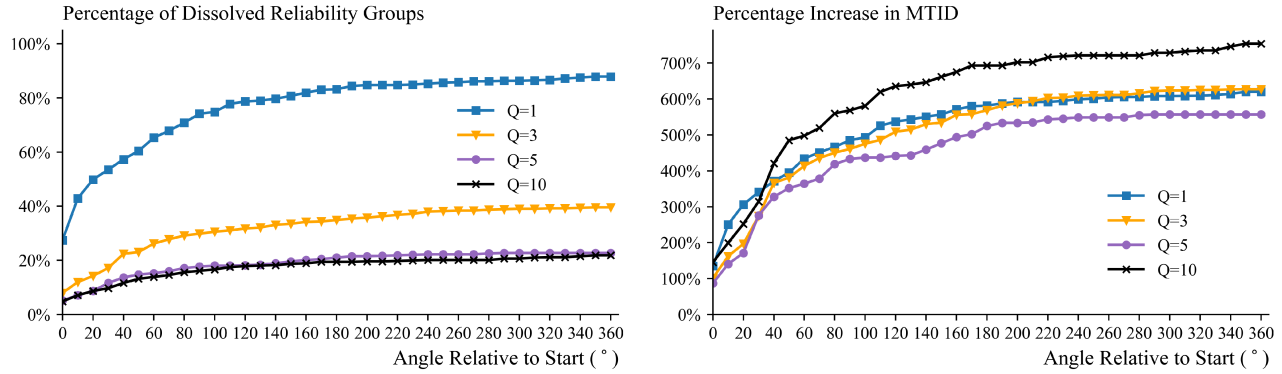
Dissolve increases MTID. When an illuminating FLS of a dissolved reliability group fails, its replacement must be deployed by a dispatcher. The replacement must travel a longer distance, resulting in a higher MTID. Figure 15b shows the percentage increase in MTID as a function of the user walking around the Skateboard. With Q=10, the percentage increase is higher because the display is larger, i.e., consists of Q^3 (1000x) more display cells than Q=1. This requires a replacement FLS to travel a longer distance (display cells) from a dispatcher.

6.2 Suspend

Suspend improves on Dissolve by suspending reliability groups pertaining to an obstructing standby instead of deactivating them permanently. Once the user moves such that the standby is no longer obstructing or anticipated to obstruct, a standby is restored for the group to re-activate the reliability group. It maintains a large number of reliability groups activated even with Q=1. This is the key strength of Suspend, resulting in a superior MTID when compared with Dissolve.

A disadvantage of Suspend is that a standby FLS may travel back-and-forth from a dispatcher several times as the user walks around an illumination. Some of the standbys restored by Suspend after the user moves (say from 20° to 30°) may become obstructing in a later movement of the user (say from 40° to 50°), forcing the standby to fly back to the hangar again.

Improving Suspend: With $Q \geq 3$, the obstructing standby FLS may fly to a different display cell instead of a hangar. This relieves the dispatchers from deploying standbys to reactivate a reliability



(a) Deactivated Reliability groups (%)

(b) Increase in MTID (%)

Figure 15: Dissolve and the resulting percentage increase in MTID, Skateboard, G=3. Lower is better.

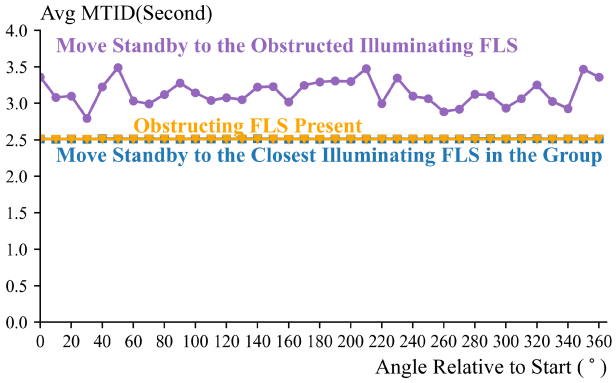


Figure 16: MTID with Suspend. Skateboard, G=3, Q=10.

group. This is advantageous when dispatchers have a limited deployment rate. Moreover, it minimizes the traveled distance by a standby to re-activate its reliability group, enhancing MTID.

A simple technique may assign the obstructing standby to a display cell of an illuminating FLS, hiding it from the user's FoV. This may be the closest illuminating FLS. This FLS may be a member of the reliability group of the standby and known to the standby[10]. Alternatively, it may be a display cell of the illuminating cell that is being obstructed. One technique to implement this is shown in Figure 11b. It requires the obstructing standby FLS to move along the vector V computed using Ray tracing away from the anticipated coordinates of user's gaze $\{U_f\}$ and towards the obstructed illuminating FLS[11]. Next, it assumes the lighting responsibility of the illuminating FLS and occupies its display cell while the illuminating FLS goes dark and hides behind the new illuminating FLS.

We implemented both techniques to compare[12] them with one another. Figure 16 shows their average MTID. As a comparison yardstick, it shows the average MTID with the obstructing standby FLS present. Obtained results show hiding the obstructing standby in the closest illuminating FLS is superior and almost as good as when the obstructing FLSs are present. There are two explanations for this. First, for any x-axis tick-mark, only a few standby FLSs change location. And, the *average* is almost identical to having these standbys at their original coordinates. Second, the distance from the standby to the illuminating FLS is short and the MTID is dominated by the acceleration and deceleration components of the velocity model (6.11 Meters/Second^2).

[10]With the shapes of Figure 1, the closest illuminating FLS is always a member of the standby's reliability group.

[11]When a standby obstructs multiple illuminating FLSs with $|U_f| \geq 2$, the standby moves towards the closest obstructing illumination cell.

[12]We employ the current coordinate of the user's gaze. Hence, the reported MTIDs are a theoretical lower bound. Predictive models of user movement are future work.

7 CONCLUSIONS AND FUTURE RESEARCH

In this study, we show standby FLSs may be used effectively to enhance the QoI in the presence of FLS failures. We compared a centralized (k-means) and a decentralized (CANF) algorithm to form reliability groups, showing the superiority of the centralized technique in minimizing distance and improving MTID. We detect a dark standby FLSs that may obstruct the user's FoV and present a technique that re-locates this standby to avoid obstruction. Once the user moves such that the standby is no longer obstructing, the standby FLS is restored to its original coordinates. In general, this technique (named Suspend) maximizes the benefits provided by the standbys. Its disadvantage is that it requires the standby FLSs to consume energy (battery power) to fly to new locations.

Our immediate research direction is to evaluate alternative user movement patterns and how to predict them to minimize the number of impacted standby FLSs and their total traveled distance. More longer term, we will implement our techniques using a swarm of off-the-shelf drones, e.g., Crazyflies.

8 ACKNOWLEDGMENTS

We thank Peter A. Kara and the anonymous MMSys 2024 reviewers for their constructive comments on earlier drafts of this paper. This research was supported in part by the NSF grant IIS-2232382. We gratefully acknowledge CloudBank [25] and CloudLab [32] for the use of their resources to enable all experimental results presented in this paper.

REFERENCES

[1] Amer Al-Rahayfeh and Miad Faezipour. 2013. Eye Tracking and Head Movement Detection: A State-of-Art Survey. *IEEE Journal of Translational Engineering in Health and Medicine* 1 (2013), 2100212 –2100212. https://doi.org/10.1109/JTEHM.2013.2289879

[2] Hamed Alimohammadzadeh, Rohit Bernard, Yang Chen, Trung Phan, Prashant Singh, Shuqin Zhu, Heather Culbertson, and Shahram Ghandeharizadeh. 2023. Dronevision: An Experimental 3D Testbed for Flying Light Specks. In *The First International Conference on Holodecks* (Los Angeles, California) *(Holodecks '23)*. Mitra LLC, Los Angeles, CA, USA, 1–9. https://doi.org/10.61981/ZFSH2301

[3] Hamed Alimohammadzadeh, Heather Culbertson, and Shahram Ghandeharizadeh. 2023. An Evaluation of Decentralized Group Formation Techniques for Flying Light Specks. In *ACM Multimedia Asia* (Taipei, Taiwan).

[4] Hamed Alimohammadzadeh and Shahram Ghandeharizadeh. 2023. SwarMer: A Decentralized Localization Framework for Flying Light Specks. In *The First International Conference on Holodecks* (Los Angeles, California) *(Holodecks '23)*. Mitra LLC, Los Angeles, CA, USA, 10–22. https://doi.org/10.61981/ZFSH2302

[5] Hamed Alimohammadzadeh, Daryon Mehraban, and Shahram Ghandeharizadeh. 2023. Modeling Illumination Data with Flying Light Specks. In *ACM Multimedia Systems* (Vancouver, Canada) *(MMSys '23)*. Association for Computing Machinery, New York, NY, USA, 363–368. https://doi.org/10.1145/3587819.3592544

[6] David Arthur and Sergei Vassilvitskii. 2007. K-Means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (New Orleans, Louisiana) *(SODA '07)*. Society for Industrial and Applied Mathematics, USA, 1027–1035.

[7] Senthil Hariharan Arul and D. Manocha. 2020. DCAD: Decentralized Collision Avoidance With Dynamics Constraints for Agile Quadrotor Swarms. *IEEE Robotics and Automation Letters* 5 (2020), 1191–1198. https://doi.org/10.1109/LRA.2020.2967281

[8] Daman Bareiss and Joran van den Berg. 2013. Reciprocal Collision Avoidance for Robots with Linear Dynamics using LQR-Obstacles. In *Proceedings - IEEE International Conference on Robotics and Automation*. 3847–3853. https://doi.org/10.1109/ICRA.2013.6631118

[9] Jack Brimberg, Nenad Mladenović, Dragan Urošević, and Eric Ngai. 2009. Variable Neighborhood Search for the Heaviest k-Subgraph. *Computers & Operations Research* 36, 11 (2009), 2885–2891. https://doi.org/10.1016/j.cor.2008.12.020

[10] Yang Chen, Hamed Alimohammadzadeh, Heather Culbertson, and Shahram Ghandeharizadeh. 2023. Towards a Stable 3D Physical Human-Drone Interaction. In *The First International Conference on Holodecks* (Los Angeles, California) *(Holodecks '23)*. Mitra LLC, Los Angeles, CA, USA, 34–37. https://doi.org/10.61981/ZFSH2308

[11] Yang Chen, Hamed Alimohammadzadeh, Shahram Ghandeharizadeh, and Heather Culbertson. 2024. Force-Feedback Through Touch-based Interactions With A Nanocopter. In *IEEE Symposium on Haptics* (Long Beach, California) *(Haptics '24)*. IEEE, Long Beach, CA, USA, 7.

[12] Anna Chmielowiec and Maarten van Steen. 2010. Optimal Decentralized Formation of k-Member Partnerships. In *IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. 154–163. https://doi.org/10.1109/SASO.2010.14

[13] Anna Chmielowiec, Spyros Voulgaris, and Maarten van Steen. 2014. Decentralized Group Formation. *Journal of Internet Services and Applications* 5, 1 (2014). https://doi.org/10.1186/s13174-014-0012-2

[14] David J. DeWitt, Shahram Ghandeharizadeh, Donovan Schneider, Allan Bricker, Hui-I Hsiao, and Rick Rasmussen. 1990. The Gamma Database Machine Project. *IEEE Transactions on Knowledge and Data Engineering* 1, 2 (March 1990).

[15] David J. DeWitt, Shahram Ghandeharizadeh, and Donovan A. Schneider. 1988. A Performance Analysis of the Gamma Database Machine. In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 1-3, 1988*, Haran Boral and Per-Åke Larson (Eds.). ACM Press, 350–360. https://doi.org/10.1145/50202.50245

[16] Eduardo Ferrera, Alfonso Alcántara, J. Capitán, Á. R. Castaño, P. Marrón, and A. Ollero. 2018. Decentralized 3D Collision Avoidance for Multiple UAVs in Outdoor Environments. *Sensors (Basel, Switzerland)* 18 (2018).

[17] Shahram Ghandeharizadeh. 2021. Holodeck: Immersive 3D Displays Using Swarms of Flying Light Specks. In *ACM Multimedia Asia* (Gold Coast, Australia). https://doi.org/10.1145/3469877.3493698

[18] Shahram Ghandeharizadeh. 2022. Display of 3D Illuminations using Flying Light Specks. In *ACM Multimedia*. 2996–3005.

[19] Shahram Ghandeharizadeh and Vincent Oria. 2023. Virtual Reality, Augmented Reality, Mixed Reality, Holograms and Holodecks. In *The First International Conference on Holodecks* (Los Angeles, California) *(Holodecks '23)*. Mitra LLC, Los Angeles, CA, USA, 38–40. https://doi.org/10.61981/ZFSH2304

[20] R. W. Hamming. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal* 29, 2 (1950), 147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x

[21] Zhou Liu and Lilong Cai. 2023. Simultaneous Planning and Execution for Safe Flight of Quadrotors Suffering One Rotor Loss and Disturbance. *IEEE Trans. Aerospace Electron. Systems* 59, 5 (2023), 5731–5747. https://doi.org/10.1109/TAES.2023.3265948

[22] S. Lloyd. 1982. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137. https://doi.org/10.1109/TIT.1982.1056489

[23] Jeffrey Mao, Jennifer Yeom, Suraj Nair, and Giuseppe Loianno. 2023. From Propeller Damage Estimation and Adaptation to Fault Tolerant Control: Enhancing Quadrotor Resilience. *CoRR* abs/2310.13091 (2023). https://doi.org/10.48550/ARXIV.2310.13091 arXiv:2310.13091

[24] Michael Mitzenmacher. 2001. The Power of Two Choices in Randomized Load Balancing. *IEEE Trans. Parallel Distrib. Syst.* 12, 10 (Oct. 2001), 1094–1104. https://doi.org/10.1109/71.963420

[25] Michael Norman, Vince Kellen, Shava Smallen, Brian DeMeulle, Shawn Strande, Ed Lazowska, Naomi Alterman, Rob Fatland, Sarah Stone, Amanda Tan, Katherine Yelick, Eric Van Dusen, and James Mitchell. 2021. CloudBank: Managed Services to Simplify Cloud Access for Computer Science Research and Education. In *Practice and Experience in Advanced Research Computing* (Boston, MA, USA) *(PEARC '21)*. Association for Computing Machinery, New York, NY, USA, Article 45, 4 pages. https://doi.org/10.1145/3437359.3465586

[26] David A. Patterson, Garth Gibson, and Randy H. Katz. 1988. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data* (Chicago, Illinois, USA) *(SIGMOD '88)*. Association for Computing Machinery, New York, NY, USA, 109–116. https://doi.org/10.1145/50202.50214

[27] Trung Phan, Hamed Alimohammadzadeh, Heather Culbertson, and Shahram Ghandeharizadeh. 2023. An Evaluation of Three Distance Measurement Technologies for Flying Light Specks. In *International Conference on Intelligent Metaverse Technologies and Applications (iMETA2023)* (Tartu, Estonia).

[28] James Preiss, Wolfgang Honig, Gaurav Sukhatme, and Nora Ayanian. 2017. Crazyswarm: A Large Nano-Quadcopter Swarm. In *IEEE International Conference on Robotics and Automation (ICRA)*. 3299–3304. https://doi.org/10.1109/ICRA.2017.7989376

[29] Kenneth Salem and Hector Garcia-Molina. 1986. Disk Striping. In *Proceedings of the Second International Conference on Data Engineering, February 5-7, 1986, Los Angeles, California, USA*. IEEE Computer Society, 336–342. https://doi.org/10.1109/ICDE.1986.7266238

[30] Donovan A. Schneider, David J. DeWitt, and Shahram Ghandeharizadeh. 1989. An overview of the Gamma Database Machine. In *Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage, COMPCON Spring 89, San Francisco, CA, USA, February 27 - March 3, 1989, Digest of Papers*. IEEE Computer Society, 162–166. https://doi.org/10.1109/CMPCON.1989.301920

[31] Martin Schulze, Garth Gibson, Randy Katz, and David A Patterson. 1989. How reliable is a RAID?. In *COMPCON Spring 89*. IEEE Computer Society, 118–119.

[32] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. 2002. An Integrated Experimental Environment for Distributed Systems and Networks. *SIGOPS Oper. Syst. Rev.* 36, SI, 255–270. https://doi.org/10.1145/844128.844152

[33] Amy Williams, Steve Barrus, R. Keith Morley, and Peter Shirley. 2005. An Efficient and Robust Ray-Box Intersection Algorithm. In *ACM SIGGRAPH 2005 Courses* (Los Angeles, California) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 9–es. https://doi.org/10.1145/1198555.1198748

[34] Xuehan Xiong, Zicheng Liu, Qin Cai, and Zhengyou Zhang. 2014. Eye Gaze Tracking Using an RGBD Camera: A Comparison with a RGB Solution *(UbiComp '14 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 1113–1121. https://doi.org/10.1145/2638728.2641694

[35] Yang Xu, Shupeng Lai, Jiaxin Li, Delin Luo, and Yancheng You. 2019. Concurrent Optimal Trajectory Planning for Indoor Quadrotor Formation Switching. *Journal of Intelligent & Robotic Systems* 94 (05 2019). https://doi.org/10.1007/s10846-018-0813-9

[36] Nima Yazdani, Hamed Alimohammadzadeh, and Shahram Ghandeharizadeh. 2023. A Conceptual Model of Intelligent Multimedia Data Rendered using Flying Light Specks. In *The First International Conference on Holodecks* (Los Angeles, California) *(Holodecks '23)*. Mitra LLC, Los Angeles, CA, USA, 38–44. https://doi.org/10.61981/ZFSH2309

[37] Shuqin Zhu and Shahram Ghandeharizadeh. 2023. Flight Patterns for Swarms of Drones. In *The First International Conference on Holodecks* (Los Angeles, California) *(Holodecks '23)*. Mitra LLC, Los Angeles, CA, USA, 29–33. https://doi.org/10.61981/ZFSH2303