Swarical: An Integrated Hierarchical Approach to Localizing Flying Light Specks

Hamed Alimohammadzadeh
halimoha@usc.edu
University of Southern California
Los Angeles, California, USA

Shahram Ghandeharizadeh
shahram@usc.edu
University of Southern California
Los Angeles, California, USA

Abstract

Swarical, a Swarm-based hierarchical localization technique, enables miniature drones, Flying Light Specks (FLSs), to accurately and efficiently localize and illuminate complex 2D and 3D shapes. Its accuracy depends on the physical hardware (sensors) of FLSs used to track neighboring FLSs to localize themselves. It uses the specification of the sensors to convert mesh files into point clouds that enable a swarm of FLSs to localize at the highest accuracy afforded by their sensors. Swarical considers a heterogeneous mix of FLSs with different orientations for their tracking sensors, ensuring a line of sight between a localizing FLS and its anchor FLS. We present an implementation using Raspberry cameras and ArUco markers. A comparison of Swarical with a state of the art decentralized localization technique shows that it is as accurate and more than 2x faster.

CCS Concepts

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

Keywords

Localization, Flying Light Specks, Dronevision, Swarm, 3D Display

ACM Reference Format:
Hamed Alimohammadzadeh and Shahram Ghandeharizadeh. 2024. Swarical: An Integrated Hierarchical Approach to Localizing Flying Light Specks. In *Proceedings of the 32nd ACM International Conference on Multimedia (MM '24), October 28-November 1, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3664647.3681080

1 Introduction

A Flying Light Speck (FLS) [14] is a drone configured with light sources. A swarm of FLSs may illuminate complex 2D and 3D multimedia shapes in a fixed volume, a 3D multimedia display [15]. Each FLS is assigned a coordinate. A challenge is how cooperating FLSs may illuminate 2D and 3D shapes. GPS [27] is not an option due to the lack of a line of sight to GPS satellites in an indoor setting [3, 14]. An FLS may travel to its assigned coordinate using Dead Reckoning [6]. This technique may employ a drone's inertial measurement unit (IMU) to approximate its location. IMUs of a drone are known to be noisy, with the error in estimated location increasing as a function of traveled distance [3, 6, 17, 22]. Figure 1 shows a palm tree with different degrees of Dead Reckoning error.

(a) GT. **(b)** $\epsilon = 2°$. **(c)** $\epsilon = 5°$. **(d) Swarical.**

Figure 1: Palm tree with 725 FLSs. Ground truth (GT), Dead Reckoning with two different degrees of error ($\epsilon = 2°$ and $5°$), and Swarical using Dead Reckoning with $\epsilon = 5°$.

A localization framework may manipulate a design space consisting of hardware, software, and data. Consider each in turn: Hardware includes sensory devices mounted on an FLS. A framework has a host of hardware choices ranging from Ultra Wide Band (UWB) radios [10, 28, 29] to ultrasonic devices and cameras [20, 24–26]. The software includes algorithms that implement a localization technique. A framework may use the decentralized algorithm of SwarMer [3] that is executed by FLSs. Data refers to a 3D shape and its representation as a point cloud. An example of a 3D shape file is a polygon mesh file. It is a collection of vertices, edges, and faces that define a 3D shape. A framework may adjust the number of FLSs used to illuminate the faces of a mesh file. With different types of FLS hardware, the framework may use a mix of FLSs that enhance the accuracy of localization, which enables a swarm of drones to illuminate a shape with high accuracy.

In this paper, we present a Swarm-based hierarchical (Swarical) framework to localize FLSs. Swarical is an integrated approach that considers hardware, software, and data. It starts by selecting the hardware that enables FLSs to localize. It uses the specification of this hardware in combination with a mesh file to compute the number of FLSs that should illuminate the shape. This considers the range of sensors used to localize FLSs in combination with the characteristics of a mesh file. Given a heterogeneous mix of FLSs with different mountings of sensors (for line of sight), Swarical computes the right mix of FLSs to illuminate a shape. This mix ensures a localizing FLS has a line of sight with its anchor FLS.

Contributions of this paper include:

- Swarical as a framework that considers hardware, software, and characteristics of a mesh file (data) to compute a point

MM '24, October 28-November 1, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0686-8/24/10
https://doi.org/10.1145/3664647.3681080

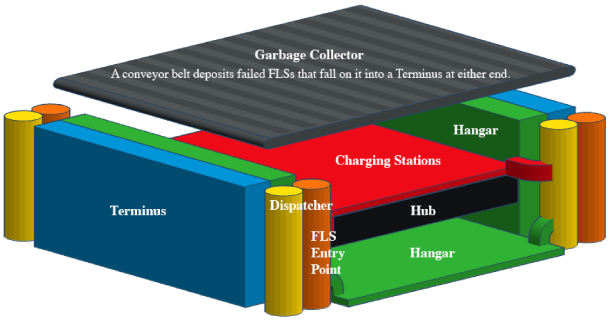


Figure 2: The yellow cylinders of the architecture of [15] are Dispatchers that deploy FLSs. The Hub is comparable to today's servers and hosts the Orchestrator process.

cloud for localization and illumination of a shape. (Sections 2 and 3.)
- Three online localization techniques with one, ISR, emerging as the superior technique. ISR enhances speed and accuracy compared to its counterparts. (Section 4.)
- An implementation of Swarcial using cameras and ArUco markers mounted on FLSs to track one another. (Section 5.)
- A comparison of Swarcial with a state of the art decentralized algorithm named SwarMer [3] shows Swarcial is more than 2x faster and equally accurate. (Section 5.4.)
- We open source our software implementations and its data set at https://github.com/flyinglightspeck/Swarcial.

Related work: The concept of 3D displays using FLS is presented in [1–5, 8, 9, 14–16, 24, 36, 37]. The most relevant is SwarMer, a decentralized localization technique that is fast and highly accurate. A qualitative and quantiative comparison of SwarMer with Swarcial is presented in Section 5.4. Obtained results show Swarcial is equally accurate and more than 2x faster than SwarMer.

The rest of this paper is organized as follows. Section 2 provides an overview of Swarcial and establishes the terminology used in this paper. While Section 3 introduces the planner component of Swarcial, Section 4 introduces several online decentralized localization techniques. We introduce an implementation of Swarcial in Section 5 and compare it with SwarMer [3]. Brief conclusions are presented in Section 6.

2 Overview and Terminology

This paper assumes the architecture of [1, 15], see Figure 2. It consists of a Hub and one or more Dispatchers to deploy FLSs. The Hub is a computer similar to today's servers. It hosts an Orchestrator process that executes the planner component of Swarcial, see Figure 4. The Orchestrator provides metadata to FLSs and deploys them using one or more Dispatchers. The FLSs travel to their assigned coordinates using Dead Reckoning. They localize relative to one another to illuminate 2D and 3D shapes.

An FLS may be configured with various sensors that enable it to localize relative to a neighboring FLS. Section 5.1 describes the use of cameras and ArUco markers [13]. A localizing FLS uses its camera to take a picture of its anchor FLS's ArUco marker and processes the picture to compute its relative pose to the anchor FLS. A challenge is how to mount cameras and ArUco markers on FLSs to ensure the camera of a localizing FLS has a line of sight with the ArUco marker of its anchor FLS. We address this challenge using a heterogeneous mix of FLSs with cameras mounted on their top, side, or bottom. See Figure 3.

(a) Top. (b) Side. (c) Bottom.

Figure 3: Three FLSs with different camera orientations/FoVs.

Swarcial is a divide-and-conquer technique. It partitions a shape into a collection of swarms. FLSs of a swarm localize relative to one another. This is intra-swarm localization. A swarm also localizes relative to another swarm. This is inter-swarm localization, stitching swarms together to illuminate a complex 2D/3D shape.

Swarcial consists of two distinct steps, see Figure 4. A centralized configuration planner and a decentralized localization process. The former is an offline process executed by the Orchestrator. The latter is an online technique executed by swarms of FLSs.

The input to the planner is a mesh file of a shape, the desired size of a swarm (G), and the available mix of FLSs with the specification of their sensors (e.g., range and orientation of a sensor). The planner processes the mesh file to compute both the number of FLSs and their correct mix to illuminate the shape using the specification of the localization device. It constructs groups of FLSs that are in close proximity to one another. The size of each group is approximately G.

The planner constructs two types of trees: FLS-trees and one swarm-tree. See Figure 5. An FLS-tree defines the anchor FLS for a localizing FLS in a swarm. The swarm-tree identifies a *primary* FLS in a child swarm that localizes relative to an anchor FLS in its parent swarm. The root of the swarm-tree is an exception. Both trees guarantee a localizing FLS has a line of sight with its anchor FLS.

When illuminating a shape, FLSs that constitute a swarm continuously localize relative to one another. The primary of a swarm (except for the root) will localize relative to the identified anchor FLS of its parent swarm. It computes a vector for its movement. Its entire swarm, including the primary, moves along this vector.

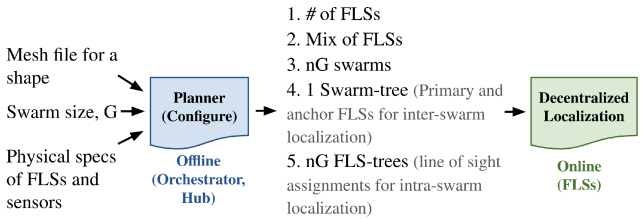


Figure 4: Swarcial, a divide-and-conquer framework.

Definition 1. *A* **swarm** *consists of one or more FLSs. Members of a swarm localize relative to one another continuously. A swarm-tree identifies the parent-child relationships between swarms. Except for the swarm that serves as the root of the tree structure, every swarm has a parent swarm and one FLS f_P designated as its primary. The primary f_P of a child swarm localizes relative to an anchor FLS of its parent swarm, computing a vector $\overrightarrow{V}$. f_P and all FLSs that constitute its swarm move along this vector $\overrightarrow{V}$.*

The output of the planner may be a large volume of data. However, each FLS requires a small fraction of this output to cooperate with the other FLSs by executing the decentralized localization technique. The Orchestrator provides this information to the FLSs.

For a given shape, the Orchestrator may execute the planner and store its output in a file. When a user requests the display of the shape repeatedly, the Orchestrator may read the file to provide each FLS with the required information [4]. The online FLS localization process is decentralized, fast, and continuous.

3 Planner

The planner consists of two sequential steps. First, it converts a mesh file into an FLS point cloud using the limits of a tracking device. Second, it fragments the resulting point cloud of F FLSs into nG swarms. Each swarm consists of approximately G FLSs. This step constructs one swarm-tree and nG FLS-trees, $nG = \lceil \frac{F}{G} \rceil$ swarms. Below, we describe the two steps in turn.

3.1 Step 1: Mesh File to FLS Illumination

FLSs must track one another to localize and illuminate a mesh file. The limits of the FLS tracking device in combination with the error tolerated by an application dictate the number of FLSs used to illuminate a face. To illustrate, consider an application that tolerates 5% error in the maximum difference between the estimated truth and the ground truth, i.e., Hausdorff distance [18]. The application uses the minimum and maximum range ($[T_{min}$-$T_{max}]$) of the tracking device that produces at most 5% error in measured distances to compute the density of FLSs in a face. Below, we present a general technique for computing this density. An implementation of it in the context of visual tracking using fiducial markers is presented in Section 5.

Consider a tracking device placed at the center of a spherical shaped FLS with a radius of R. An application tolerates e% error in the Hausdorff distance of an illumination. The planner identifies the minimum and maximum $[T_{min}$-$T_{max}]$ range of the tracking device with a percentage error less than or equal to e. Assume the radius R is less than or equal to T_{max}, $R \leq T_{max}$, the planner computes the min/max density of FLSs in a unit of area: $D_{min} = \frac{1}{\pi \times max(T_{max}/2,R)^2}$, $D_{max} = \frac{1}{\pi \times max(T_{min}/2,R)^2}$. By multiplying these by the area of a face, the planner estimates the minimum and maximum number of FLSs required to illuminate the face with e% error in Hausdorff distance.

There is extensive work on sampling a mesh file [31] to generate a point cloud. Section 5.2 uses the Constrained Poisson-disk sampling [11] by providing it with the number of FLSs computed using the above discussion. It is possible to use other techniques such as those in [35].

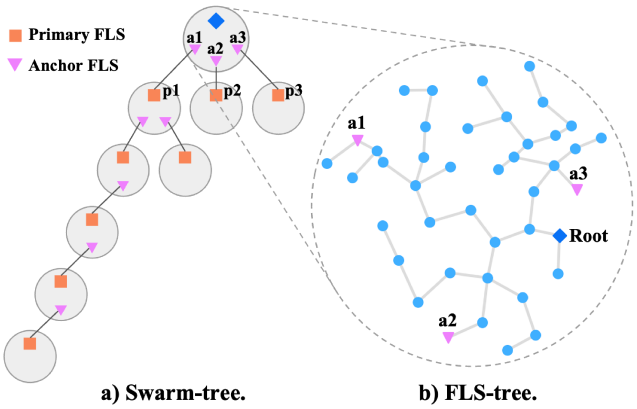


Figure 5: Swarm-tree and FLS-tree with the Chess Piece, G=50.

3.2 Step 2: FLS-Tree and Swarm-Tree

The planner constructs swarms with different mixes of FLSs to facilitate intra and inter swarm localization. Given a group size G and F FLSs, the planner constructs nG groups using the k-Means [21] algorithm, $nG = \lceil \frac{F}{G} \rceil$. Each resulting group will consist of approximately G FLSs. A group corresponds to a swarm.

The planner constructs a *swarm-tree* on the nG swarms, identifying one FLS of a swarm as its primary f_P that localizes relative to the nearest anchor FLS in a parent swarm. The planner constructs an *FLS-tree* on the G FLSs in a swarm, establishing the localizing and anchor relationship between the FLSs that constitute a swarm. Figure 5 shows the FLS-tree and swarm-tree of the Chess Piece.

The objective of the planner is to satisfy two constraints. First, the tracking device of a child FLS should have line of sight with its parent FLS. Second, the distance between the localizing FLS and its anchor FLS should respect the $[T_{min} - T_{max}]$ of the tracking device.

To realize its objectives, the planner uses the center of a swarm to construct a minimum-spanning tree [7, 19] across the swarms. This is the swarm-tree. Its vertices correspond to swarms of FLSs. The weight of an edge between two swarms is the Euclidean distance between their centers. The minimum spanning tree connects all the swarms together without any cycles and with the minimum possible total edge weight. The planner identifies the vertex with the highest number of edges as the root of the swarm-tree. It walks its children in a breadth first manner to establish the parent-child relationship between swarms. With a parent-child swarm, the planner selects an FLS from the parent swarm that is closest to an FLS in the child swarm. The latter is the primary FLS of the child swarm. The former is the anchor FLS of the parent swarm. The primary localizes relative to the anchor. The point cloud dictates the orientation of the primary relative to its anchor. The planner uses this information to assign one of the FLS types in Figure 3 to the primary with the objective of ensuring it has line of sight to its anchor.

Once the primary FLS of a swarm is identified, the planner computes a minimum spanning tree for the FLSs that constitute a swarm. This is the FLS-tree. Its vertices correspond to FLSs. The distance between two FLSs is computed using the Euclidean distance between their coordinates. The minimum spanning tree connects all the FLSs together without any cycles and with the minimum possible

total edge weight. The planner traverses this tree starting with the primary in a breadth first manner. It establishes the line of sight relationship from child to parent. The planner uses the orientation of an FLS in the point cloud to assign the child FLS one of the FLS types shown in Figure 3. The selection ensures a localizing (child) FLS has a line of sight with its anchor (parent) FLS.

With both the swarm-tree and *nG* FLS-trees, the planner ensures the distance between a localizing and anchor FLS is lower than T_{max}. If their distance exceeds T_{max}, the planner inserts dark FLSs to reduce the distance. These FLSs may serve as hot standbys to tolerate the failure of the illuminating FLSs [5].

4 Continuous Localization

This section describes three online localization techniques. All three assume an Orchestrator that allocates the correct mix of FLSs per output of the planner. The Orchestrator uses the FLS-trees and the swarm-tree to assign each FLS a coordinate in the 3D volume and provides it with its parent FLS and children FLSs. With an FLS designated as the primary of a swarm, f_P, the Orchestrator provides the FLS with the identity of its anchor FLS in its parent swarm (as computed by the planner).

The key difference between the localization techniques is the amount of concurrent movement by different FLSs in a swarm and across the swarms. We start with a highly concurrent technique. Subsequently, we describe two variants that limit the amount of concurrent movement. Our experimental results show the second technique, ISR, is faster and more accurate than the other two. It is also more energy efficient by minimizing the total distance traveled by FLSs.

Highly Concurrent, HC, allows the primary of a swarm (f_P) to localize relative to its anchor in the parent swarm while the anchor is localizing itself. This means all swarms may localize at the same time. Below, we describe intra-swarm localization, i.e., how FLSs in a swarm localize relative to one another. Subsequently, we describe inter-swarm localization, i.e., how two swarms localize relative to one another.

Using the ground truth, an FLS knows its position and orientation relative to its swarm members. The FLS-tree ensures a localizing FLS has a line of sight with its anchor FLS. The root of the tree is an exception. Consider localization for a child FLS and a root FLS in turn.

A child FLS u computes its pose relative to its parent v, $\hat{\rho}_{u,v}$ ($\hat{\rho}_{u,v} = -\hat{\rho}_{v,u}$). The pose, $\hat{\rho}_{u,v}$, is a position vector where v is the vector's head and u at the origin is its tail. FLS u broadcasts $\hat{\rho}_{u,v}$ to all its swarm members. A receiving FLS constructs an intra-swarm tree to maintain this information broadcasted by different FLSs. See the FLS-tree of Figure 5. An FLS i computes a relative pose for each reachable[1] FLS within the tree structure. This relative pose, denoted as $\hat{\rho}_{i,j}$, is determined by the sum of relative pose vectors $\hat{\rho}_{u,v}$ along the path from FLS i to FLS j. To correct its position relative to these FLSs, FLS i computes a correction vector v_{ij}, defined as $\rho_{i,j} - \hat{\rho}_{i,j}$, where $\rho_{i,j}$ represents the pose of FLS i to FLS j in the ground truth. This process is repeated for all reachable FLSs, resulting in a set of correction vectors. FLS i then moves along the average of these vectors, computed as $\frac{1}{N}\sum_{j\in N_T} v_{ij}$, where N_T is the reachable FLSs in the FLS-tree, including FLS i. It is possible for an FLS to compute a vector using only its parent FLS. This happens at the very beginning before the FLS receives a vector from other FLSs or when a swarm consists of only 2 FLSs.

Every time an FLS receives the relative pose from another FLS in its swarm, it repeats the process to localize itself. Should an FLS not receive information from its swarm members for 500 milliseconds, it localizes, computes a vector, broadcasts its pose relative to its parent to all its swarm members, and moves along the vector. An FLS clears its tree structure after each inter-swarm localization.

The root FLS also receives relative measurements from its children, grandchildren, and other descendant FLSs in the tree. It uses this information to compute its relative pose to them. It computes a vector to correct its position relative to each FLS. Next, it computes an average of these vectors. And moves along this average vector to localize.

An inter-swarm localization occurs once the length of the vector computed by all members of a swarm is smaller than a pre-specified threshold. Once the primary f_P of a swarm detects this condition, it localizes relative to its anchor in its parent swarm. The root swarm is an exception as it has no primary and will not localize relative to another swarm. f_P uses its pose relative to its anchor to compute a vector to correct its pose. Subsequently, f_P and its entire swarm moves along this vector. After this movement, the FLSs that constitute the swarm clear their tree structure of the relative pose information broadcasted by the FLSs in their swarm. Subsequently, they repeat their intra-swarm localization.

HC prevents a swarm from performing inter-swarm localization while its FLSs are localizing actively, i.e., their computed average vector is greater than a pre-specified threshold. Removing this requirement results in a variant with higher concurrency. It causes FLSs that constitute a swarm to move away from their primary, producing distorted shapes.

A small (large) threshold value implies a more (less) accurate relative poses. A large threshold value, say ∞, is not the same as not having a threshold all together. It orders intra and inter swarm localizations for a swarm to not overlap in time.

Inter-Swarm Rounds, ISR, limits the number of swarms that localize at a time. It requires the anchor FLS of f_P to be stationary prior to f_P localizing relative to it. It uses the swarm-tree to realizes this objective. Once the length of the correction vector computed by an FLS in the root swarm that serves as an anchor for a child swarm is smaller than a pre-specified threshold, the anchor informs its f_P to localize. The f_P waits until its correction vector relative to its swarm members is smaller than a pre-specified threshold. Subsequently, it localizes relative to its parent's anchor FLS, computes a vector, moves along this vector, and requires its entire swarm to move along this vector. Next, the anchor FLSs in the f_P's swarm notify their children's f_P to localize relative to them. This process continues until the children swarm at the leaves of the tree localize.

ISR's localization is continuous, starting with the root swarm. An anchor FLS in one swarm may send multiple notifications to its f_P to localize while the f_P waits for its correction vector to become smaller than the pre-specified threshold. In this case, the f_P drops

[1]Reachable means there is a path between the FLS and other FLSs in the tree with information about their relative pose. Either the Orchestrator may provide an FLS with the FLS-tree, or the network transmission of an FLS may include its id and its parent id to enable a receiving FLS to construct the FLS-tree.

the repeated messages. It localizes once after its correction vector is smaller than the pre-specified threshold.

Table 1: Raspberry camera module 3 NoIR specifications.

Lens	Resolution (px)	FoV (°)	Min Focus Range	Weight (g)	Price (USD)
Regular	4608 × 2592	D 75, H 66, V 41	100 mm	3.2	$25
Wide	4608 × 2592	D 120, H 102, V 67	50 mm	3.2	$35

The root swarm initiates the above process every time it receives a relative pose from a swarm member, which causes it to compute a correction smaller than the pre-specified threshold. The concept of a swarm member localizing every 500 milliseconds is present. Hence, in the worst case scenario, the root swarm initiates localization every 500 milliseconds.

Rounds across the Swarm-tree and FLS-trees, RSF, constrains the number of concurrent localizations within a swarm. An FLS in a swarm localizes relative to its anchor in rounds. These rounds are initiated by the root of the FLS-tree.

RSF is continuous, similar to the other techniques. Starting with the root swarm of the swarm-tree, the root FLS of its FLS-tree notifies each of its children FLSs to localize relative to it while it remains stationary. Subsequently, each child FLS notifies its children FLSs to localize relative to it while it remains stationary. This process repeats continuously.

Except for the swarms that are at the leaves of the swarm tree, a swarm has an anchor FLS for each of its children swarms. An f_P of these swarms localizes relative to their anchor. Once an anchor FLS completes its localization, it notifies the f_P to localize relative to it. This causes the entire child swarm containing f_P to move. Subsequently, f_P's children localize relative to it. This process repeats continuously. The root swarm initiates the above process, similar to ISR.

5 An Implementation and Evaluation

This section describes an implementation of Swarical using Raspberry cameras and ArUco markers. Section 5.1 presents a camera and characterizes its accuracy in measuring pose. Subsequently, Sections 5.2 and 5.3 present results from Swarical's planner and localization techniques, respectively. Finally, we compare Swarical with a state of the art decentralized localization technique named SwarMer [3] in Section 5.4.

5.1 FLS Tracking: Calibration

To localize relative to its neighbors, an FLS must track them. The ideal tracking mechanism should be:

- Accurate: An FLS should be able to quantify its relative state to a neighbor with a high accuracy. The relative state between two FLSs u and v includes a pose $\hat{\rho}_{u,v}$ and an orientation (roll, pitch, and yaw) [30, 33, 34]. Ideally, the accuracy of the position should be in millimeters. The error in a measured orientation should be less than 1 degree in each dimension.
- Acceptable range: An FLS should be able to measure its state relative to a neighbor at distances ranging from a few centimeters up to tens of centimeters.

Table 2: Camera's frame rate and marker detection performance with the regular/wide lens. The 720p setting is used for the numbers reported in this paper.

Resolution	Frames/ Second	Avg Camera Delay, milliseconds	Avg Processing Time, milliseconds
480p	59.3/44.8	10/15	6/7
720p	46.9/44.4	3/8	18/14
1080p	21.1/26.0	8/8	39/29

- Fast with a high refresh rate: An FLS should be able to quantify its relative state to a neighboring FLS in sub-milliseconds. Moreover, it should be able to refresh this information quickly at a frequency of 10 Hz.
- Robust: An FLS should be able to track a neighbor in an indoor setting with different lighting including no light, i.e., a dark room.

ArUco markers [13] with a Raspberry camera configured with IR lighting satisfy the above requirements[2]. The camera is small, lightweight, and ready for use with a drone. It has a regular and a wide lens with a minimum focus range of 10 and 5 cm, respectively. See Table 1. It supports three different resolutions. Table 2 shows these and our experimentally measured average camera delay and processing time. The average camera delay is the elapsed time from when the application requests a frame to the time the camera provides the frame. Processing time is the time required to measure position and orientation using Raspberry Pi 5. We designed our software to capture an image once it is done processing the current image. Hence, the reported accuracy is based on the latest image available. We use its 720p frame setting for the rest of this paper, see Table 2.

The maximum range of a camera for detecting a marker depends on the marker size. Figure 6 shows the detection rate with a 4.7 mm paper printed maker size. While the x-axis of this figure is the distance between the Raspberry camera and the marker, the y-axis is the detection rate. It highlights the minimum focus range of the different lenses in Table 1 with the detection rate becoming 100% at the reported minimums. With the wide angle lens, the detection rate drops to zero with 300 mm. With this marker size, Figure 7 shows the percentage error increases as a function of the distance[3]. The regular lens provides a lower error as a function of longer distances when compared with the wide lens.

Larger marker sizes reduce the error with both lenses. The wide lens has a lower error when compared with the regular lens. See Figure 8. In this figure, the x-axis is the marker size, and the y-axis is the percentage error in the measured distance. The reported errors are for measuring the minimum focus range of the two lenses, 5 and 10 cm, with wide and regular lenses, respectively. In general, paper provides a higher percentage error when compared with LCD.

Figure 9 shows the error in roll, pitch, and yaw as a function of paper printed marker size with the wide lens. The camera provides a higher accuracy for the yaw (rotation around the axis perpendicular

[2]We also considered Bitcraze's AI Deck and decided against its use due to its cost, $225 at the time of this writing.

[3]With both lenses, we report the percentage error with distances smaller than the advertised minimum as long as the camera detects the ArUco marker.

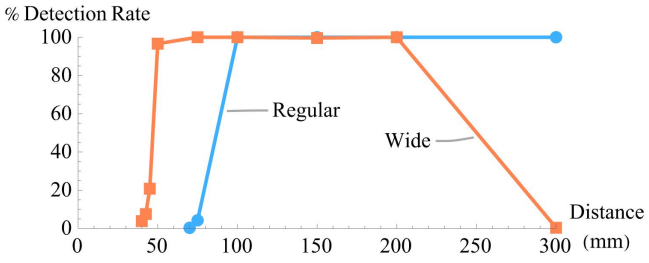


Figure 6: Detection rate as a function of distance between camera and marker. The paper printed marker size is 4.7 mm.

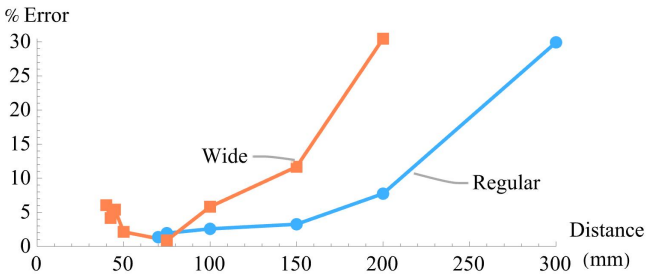


Figure 7: Percentage error of distance measurement as a function of distance between camera and marker. The paper printed marker size is 4.7 mm.

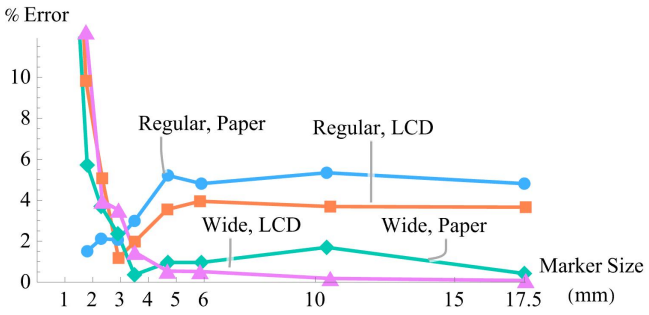


Figure 8: Percentage error of distance measurement as a function of marker size. The measured distance is 5 and 10 cm for wide and regular lens, respectively.

to the marker) than the roll (rotation around the length) and the pitch (rotation around the depth). This accuracy decreases with marker sizes smaller than 3 mm.

In darkness, an FLS may use the camera with IR light to capture an image of paper-printed markers for processing. In our experiments, IR lighting in the dark does not impact the accuracy of measurements and the detection rate.

5.2 Planner

We use the Raspberry camera in the [6,8] cm range as it provides the highest accuracy. With this range, the planner computes a point cloud of 1372 FLSs and 40 standby FLSs for the skateboard. The mix of FLSs with a camera mounted on their top, side, and bottom is

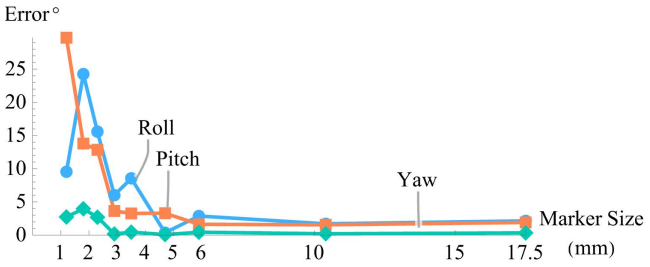


Figure 9: Error of orientation measurements in degrees as a function of marker size with wide lens and printed markers.

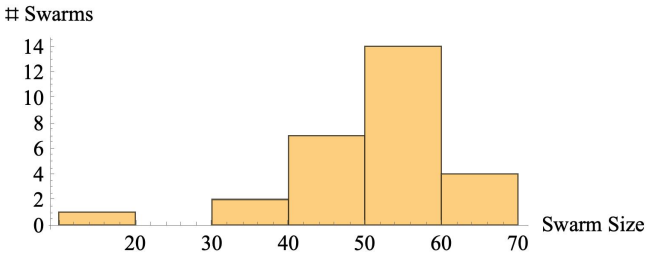


Figure 10: Distribution of swarm size, Skateboard, G=50.

142, 1137, and 133, respectively. The percentage of each variant is 10.1%, 80.5%, and 9.4%, respectively. This mix ensures a localizing FLS has a line of sight with the ArUco marker of its anchor FLS. With all the shapes, the percentage of FLSs with a camera mounted on their side is significantly higher than the others.

Figure 10 shows the distribution of swarm size with the Skateboard with G=50. Swarical uses k-Means to construct swarms. This clustering technique minimizes the Euclidean distance between the FLSs that constitute a swarm. However, it does not ensure swarms of the same size. As shown, the size of a swarm varies from 10 to 70. The same is true with the other shapes. The topology of a shape dictates the swarm sizes constructed by k-means.

Figure 11 shows the distribution of the distance between localizing and anchor FLSs within the swarms (FLS-trees) and across swarms (swarm-tree). This is for the Skateboard with different group sizes, G. We configured the planner to limit the distance between a localizing and an anchor FLS to [6-8] cm. After constructing the swarm-tree and FLS-trees for each value of G, it inserted 160, 132, 40, 23, and 18 dark FLSs for G=5, 10, 50, 150, and 200, respectively. Hence, the median is between 6 and 7 cm for all G values. There is no localizing anchor pair with a distance smaller than 6 cm. The variation in distance is greater for the swarm-tree with smaller group sizes, G=5 and 10. This is because there is a larger number of swarms. The inverse is observed with larger group sizes, G=150 and 200 because there are fewer swarms.

Figure 12 shows the distribution of the branching factor for the swarm-tree and the FLS-trees. This is the number of FLSs (swarms) that localize relative to one anchor FLS (swarm). The median is one. However, the outliers may be as high as 3 or 4. The minimum is zero. These correspond to FLSs (swarms) that are the leaves of an FLS-tree (swarm-tree).

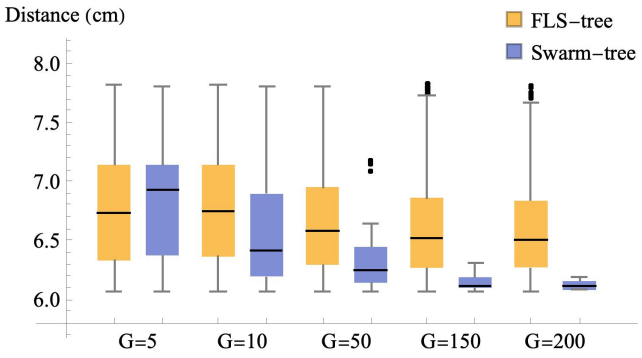


Figure 11: Distribution of distance between localizing and anchor FLSs within a swarm (FLS-tree) and across swarms (Swarm-tree), Skateboard.

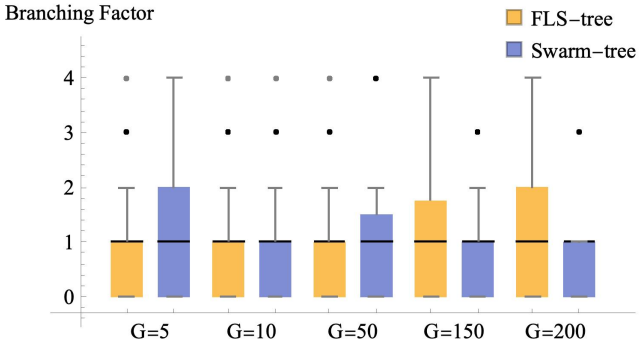


Figure 12: Distribution of the number of localizing FLSs (swarms) per anchor FLS, Skateboard.

5.3 Localization

All experiments reported in this section are conducted using a cluster of 20 Amazon AWS servers, c6a.metal, with 192 virtual cores. Each core is used to emulate an FLS. We use Hausdorff Distance (HD) [18] and Chamfer Distance (CD) [12] to compare the quality of localizations provided by HC, ISR, and RSF. These metrics compare the FLS coordinates obtained using a localization technique, i.e., the estimated truth E, with the FLS coordinates provided by the Planner, i.e., the ground truth P. After applying a translation, HD quantifies the maximum error in distance between E and P. CD quantifies the average error between E and P. Both techniques require a translation process because Swarical is a relative localization technique. Our implementation of the translation process computes the center of E and P. It aligns their centers prior to measuring the maximum/average error. A lower value is better, with zero reflecting a perfect match between E and P.

In general, HD is more strict than CD because it uses the maximum error. Both are useful in understanding the tradeoffs associated with the alternative techniques.

Figure 13 compares HC, ISR, and RSF with one another. The x-axis is the elapsed time from when the dispatcher deploys the first FLS. Once an FLS arrives at its assigned coordinate, it starts to localize. We assume a 5° dead reckoning error. The y-axis shows the

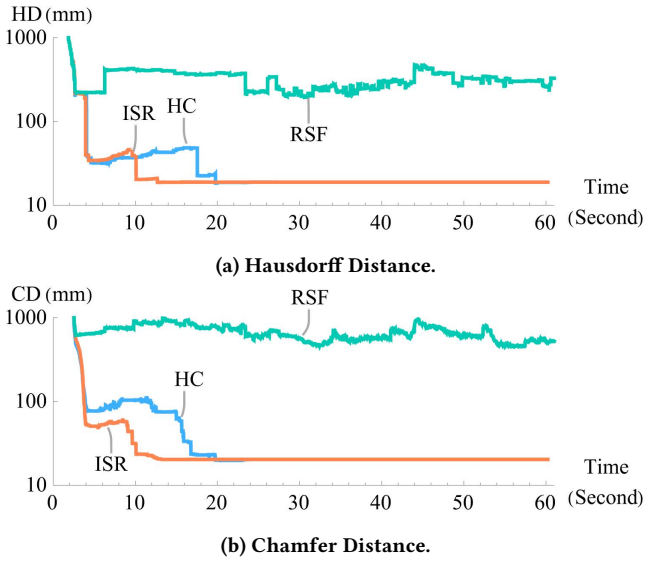


Figure 13: A comparison of Localization techniques, Skateboard, $G = 50$. Click ISR, HC, and RSF for a demonstration.

HD and the CD[4] in Figure 13a and 13b, respectively. Both figures are for the Skateboard with G=50. Similar trends are observed with the other shapes and values of G.

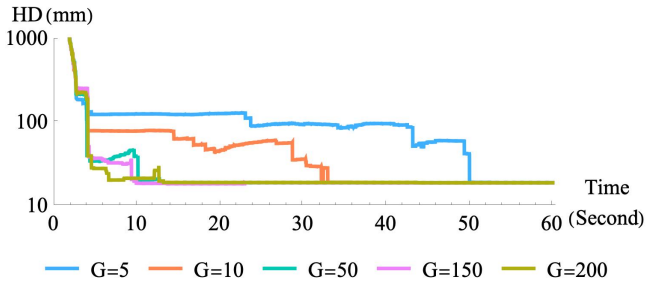
Figure 13 shows ISR is superior to HC and RSF. It enhances HD and CD, providing illuminations that resemble those computed by the Planner more accurately. RSF is significantly worse. It requires an FLS to compute its pose relative to another FLS (its anchor). HC and ISR require an FLS to compute an average correction pose. This averaging minimizes HD and CD as a function of time while RSF's HD and CD remain elevated.

In all these experiments, RSF causes the FLSs to travel a longer total distance when compared with ISR and HC. ISR reduces this metric slightly lower than HC. This slight improvement is consistent throughout our experiments.

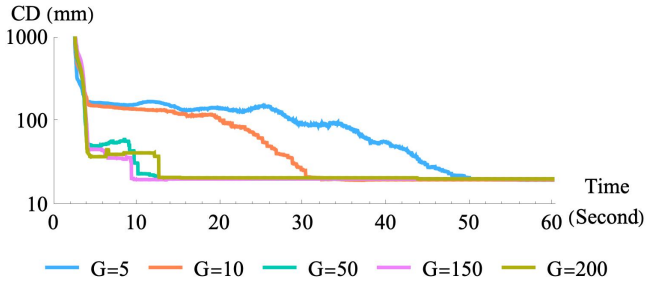
The select range of [6,8] cm corresponds to 0.9 to 1.2 mm error, see Figure 7. However, in Figure 13, HD levels off at 18.9 mm. This is 20x higher. If we considered only two points, we would observe the expected 0.9 to 1.2 mm error. However, with a point cloud, the error compounds as FLSs localize to magnify the error.

Figure 14 shows the HD and CD of the Skateboard with ISR and different swarm sizes (G). Small swarm sizes ($G \leq 10$) result in a higher HD and CD, i.e., a larger difference between the point clouds illuminated by ISR and the point cloud computed by the Swarical's planner. This is because they result in an unbalanced and deep swarm-tree with more swarms, 43 with G=5 and 38 with[5] G=10. The swarms close to the leaves of the swarm-tree require a longer time to localize because their anchor in a parent swarm has a higher probability of changing its location. This change is due to

[4]CD's value may be higher than HD because it computes the average distance between point clouds A and B, then computes it again by replacing A with B, for B and A, and then adds the two values [12]. See Equation: $\text{Chamfer}(A,B) = \frac{1}{|A|}\sum_{a\in A}\min_{b\in B}\|a-b\|_2^2 + \frac{1}{|B|}\sum_{b\in B}\min_{a\in A}\|b-a\|_2^2$.
[5]Depth decreases to 12 with G=50.



(a) Hausdorff Distance.



(b) Chamfer Distance.

Figure 14: Comparison of different swarm sizes (G) with the Skateboard and the ISR technique. Lower is better.

both intra-swarm and inter-swarm localization. An inter-swarm localization of a primary moves the entire swarm, including those FLSs that serve as anchors for other swarms. These result in high Hausdorff and Chamfer distances.

5.4 A Comparison with SwarMer

SwarMer [3] is a decentralized localization framework for FLSs. Individual FLSs localize relative to one another to form swarms. An FLS of one swarm localizes relative to an anchor FLS of another swarm to merge with it, forming a larger swarm. This process repeats until there is one swarm. Subsequently, SwarMer thaws the final swarm into individual FLSs and repeats the process.

Both SwarMer and Swarical are continuous techniques that use the concept of localizing and anchor FLSs. SwarMer constructs its swarms in an online manner. In contrast, Swarical constructs its swarms in an offline manner. SwarMer's swarms are seeded with 1 FLS that merge to construct larger swarms, ultimately growing into one swarm that includes all FLSs. Swarical's swarms are static. Swarical is an integrated approach that considers the range of sensors mounted on an FLS to track another FLS. This is reflected in its hierarchical swarm-tree and nG FLS-trees. These concepts are absent from SwarMer.

Figure 15 shows the HD with SwarMer and Swarical for the Skateboard. Swarical is configured with group size 50 (G=50) and the ISR technique. SwarMer does not consider the error associated with the range of an FLS's tracking device. Hence, we assume the tracking device is 100% accurate in measuring an FLS's pose with both techniques. Swarcial localizes the FLSs more than 2x faster than SwarMer. A similar observation is made with CD.

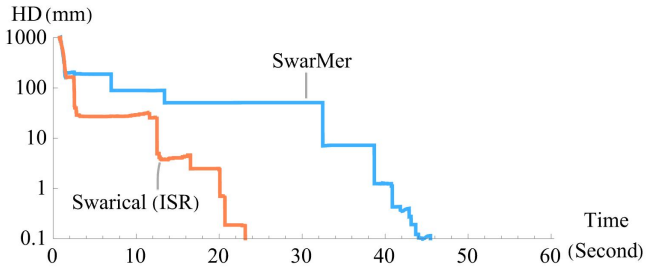


Figure 15: Comparison of Swarical with SwarMer for the Skateboard.

Swarical is faster than SwarMer for two reasons. First, FLSs exchange fewer messages. More specifically, SwarMer requires a challenge phase for a localizing FLS to identify its anchor FLS. This step is absent from Swarical; its Planner computes the localizing and anchor FLSs in an offline manner. Second, FLSs move a shorter distance with Swarical than SwarMer. In the experiments of Figures 15, on the average 8% less. The minimum distance moved by FLSs with Swarical is 12% shorter than SwarMer.

5.5 Discussion

Figure 14 shows a camera error of [0.9-1.2] cm resulting in an HD that is 20x higher. It is possible to model the relationship between the camera error and the observed HD. A system designer may use these analytical models to estimate the HD for a tracking device. Below, we describe the analytical models.

The camera error adds a positive percentage error to the distances measured by FLSs. Let D denote the average distance between FLSs, say D=7 cm. Moreover, let the average percentage error attributed to the camera be ϵ%, say ϵ=1.15%. When FLSs localize erroneously using distances that are ϵ percentage (1.15%) larger than the ground truth, the point cloud shrinks ϵ%. This is because a localizing FLS overestimates its distance to an anchor FLS, causing it to adjust its distance to be shorter than the ground truth.

To estimate the observed error, one may shrink a point cloud ϵ% and compare it with the original point cloud. The intuition here is that the localization error depends on how the distance between the matching points between the two point clouds changes as we scale one of the point clouds and align their centers. The results will approximate the HD expected with a camera that provides ϵ% error in its measurements. In our experiments with different shapes, the percentage error between the estimated and observed HD was lower than 2.5%.

6 Conclusions and Future Research

Swarical is a framework that considers the range of sensors mounted on FLSs to generate point clouds that enable FLSs to localize with a high accuracy. In turn, this renders highly accurate illuminations. The accuracy of Swarical is dictated by the sensor and its hardware used to localize. Swarical ensures localizing FLSs have a line of sight with their anchors. Simulation results show that Swarical is as accurate with scaled-down versions of drones, cameras, and ArUco markers. Our immediate research direction is to construct these candidate FLSs.

7 Acknowledgments

We thank the anonymous reviewers of the ACM MM 2024 for their valuable comments. This research was supported in part by the NSF grants IIS-2232382 and CMMI-2425754. We gratefully acknowledge CloudBank [23] and CloudLab [32] for the use of their resources to enable all experimental results presented in this paper.

References

[1] 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

[2] 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).

[3] 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

[4] 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

[5] Hamed Alimohammadzadeh, Shuqin Zhu, Jiadong Bai, and Shahram Ghandeharizadeh. 2024. Reliability Groups with Standby Flying Light Specks. In *ACM Multimedia Systems* (Bari, Italy).

[6] Martin Brossard, Axel Barrau, and Silvère Bonnabel. 2020. AI-IMU Dead-Reckoning. *IEEE Transactions on Intelligent Vehicles* 5, 4 (2020), 585–595. https://doi.org/10.1109/TIV.2020.2980758

[7] Bernard Chazelle. 2000. The Soft Heap: An Approximate Priority Queue with Optimal Error Rate. *J. ACM* 47, 6 (nov 2000), 1012–1027. https://doi.org/10.1145/355541.355554

[8] 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

[9] 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 pages.

[10] Pablo Corbalán, Gian Pietro Picco, and Sameera Palipana. 2019. Chorus: UWB Concurrent Transmissions for GPS-like Passive Localization of Countless Targets. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks* (Montreal, Quebec, Canada) *(IPSN '19)*. Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/3302506.3310395

[11] Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. 2012. Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes. *IEEE Transactions on Visualization and Computer Graphics* 18, 6 (2012), 914–924. https://doi.org/10.1109/TVCG.2012.34

[12] H. Fan, H. Su, and L. Guibas. 2017. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 2463–2471. https://doi.org/10.1109/CVPR.2017.264

[13] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. 2014. Automatic Generation and Detection of Highly Reliable Fiducial Markers under Occlusion. *Pattern Recognition* 47, 6 (2014), 2280–2292. https://doi.org/10.1016/j.patcog.2014.01.005

[14] Shahram Ghandeharizadeh. 2021. Holodeck: Immersive 3D Displays Using Swarms of Flying Light Specks. In *ACM Multimedia Asia* (Gold Coast, Australia). ACM Press, New York, NY, 1–7. https://doi.org/10.1145/3469877.3493698

[15] Shahram Ghandeharizadeh. 2022. Display of 3D Illuminations using Flying Light Specks. In *ACM Multimedia*. ACM Press, New York, NY, 2996–3005.

[16] 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

[17] Francisco Javier Gonzalez-Castano, Felipe Gil-Castineira, David Rodriguez-Pereira, Jose Angel Regueiro-Janeiro, Silvia Garcia-Mendez, and David Candal-Ventureira. 2020. Self-corrective Sensor Fusion for Drone Positioning in Indoor Facilities. *IEEE Access* 9 (2020), 2415–2427.

[18] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. 1993. Comparing Images Using the Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15, 9 (1993), 850–863. https://doi.org/10.1109/34.232073

[19] David R. Karger, Philip N. Klein, and Robert E. Tarjan. 1995. A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees. *J. ACM* 42, 2 (mar 1995), 321–328. https://doi.org/10.1145/201019.201022

[20] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. 2013. Towards a Swarm of Agile Micro Quadrotors. *Autonomous Robots* 35 (11 2013), 573–7527. https://doi.org/10.1007/s10514-013-9349-9

[21] 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

[22] K Nirmal, AG Sreejith, Joice Mathew, Mayuresh Sarpotdar, Ambily Suresh, Ajin Prakash, Margarita Safonova, and Jayant Murthy. 2016. Noise Modeling and Analysis of an IMU-based Attitude Sensor: Improvement of Performance by Filtering and Sensor Fusion. In *Advances in optical and mechanical technologies for telescopes and instrumentation II*, Vol. 9912. SPIE, 2138–2147.

[23] 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

[24] 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).

[25] 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

[26] Robin Ritz, Mark W. Müller, Markus Hehn, and Raffaello D'Andrea. 2012. Cooperative Quadrocopter Ball Throwing and Catching. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 4972–4978. https://doi.org/10.1109/IROS.2012.6385963

[27] Nel Samama. 2008. *Global Positioning: Technologies and Performance*. https://doi.org/10.1002/9780470241912

[28] K. Siwiak. 2001. Ultra-wide Band Radio: Introducing a New Technology. In *IEEE VTS 53rd Vehicular Technology Conference, Spring 2001. Proceedings (Cat. No.01CH37202)*, Vol. 2. 1088–1093 vol.2. https://doi.org/10.1109/VETECS.2001.944546

[29] Janis Tiemann and Christian Wietfeld. 2017. Scalable and Precise Multi-UAV Indoor Navigation using TDOA-based UWB Localization. In *2017 international conference on indoor positioning and indoor navigation (IPIN)*. IEEE, 1–7.

[30] Jiang Wang and William J Wilson. 1992. 3D Relative Position and Orientation Estimation using Kalman Filter for Robot Control. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*. IEEE Computer Society, 2638–2639.

[31] Zong-Sheng Wang, Jung Lee, Chang Geun Song, and Sun-Jeong Kim. 2020. Data-Driven Point Sampling with Blue-noise Properties for Triangular Meshes. In *Proceedings of the 3rd International Conference on Computer Science and Software Engineering* (Beijing, China) *(CSSE '20)*. Association for Computing Machinery, New York, NY, USA, 77–82. https://doi.org/10.1145/3403746.3403908

[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] Seong-hoon Peter Won, Wael William Melek, and Farid Golnaraghi. 2009. a Kalman/Particle Filter-based Position and Orientation Estimation Method using a Position Sensor/Inertial Measurement Unit Hybrid System. *IEEE Transactions on Industrial Electronics* 57, 5 (2009), 1787–1798.

[34] Hao Xu, Luqi Wang, Yichen Zhang, Kejie Qiu, and Shaojie Shen. 2020. Decentralized Visual-Inertial-UWB Fusion for Relative State Estimation of Aerial Swarm. In *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 8776–8782.

[35] Dong-Ming Yan, Jian-Wei Guo, Bin Wang, Xiao-Peng Zhang, and Peter Wonka. 2015. A Survey of Blue-Noise Sampling and Its Applications. *Journal of Computer Science and Technology* 30, 3 (2015), 439–452.

[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