Reproducibility Companion Paper: Swarical: An Integrated Hierarchical Approach to Localizing Flying Light Specks

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

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

Federico Cunico
federico.cunico@univr.it
University of Verona
Verona, Italy

Joshua Springer
joshua19@ru.is
Reykjavik University
Reykjavik, Iceland

Abstract

This companion paper provides artifacts and instructions on replicating the experiments in the ACM Multimedia 2024 paper entitled "Swarical: An Integrated Hierarchical Approach to Localizing Flying Light Specks." Swarm-based hierarchical, Swarical, is a localization technique that enables miniature drones, Flying Light Specks (FLSs), to accurately and efficiently localize and illuminate complex 2D and 3D shapes. It consists of two components, an offline planner and an online localization technique that executes on an FLS. The offline planner uses the FLS sensor specification for positioning to convert mesh files into swarms of FLSs. Some FLSs are dark and used only for localization. We reported the online localization technique to be fast and highly accurate. We describe how to reproduce this finding using our artifacts.

CCS Concepts

• **Computing methodologies** → **Distributed algorithms**; **Graphics systems and interfaces**; • **Information systems** → **Multimedia content creation**; • **Human-centered computing** → **Visualization design and evaluation methods**.

Keywords

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

ACM Reference Format:
Hamed Alimohammadzadeh, Shahram Ghandeharizadeh, Federico Cunico, and Joshua Springer. 2025. Reproducibility Companion Paper: Swarical: An Integrated Hierarchical Approach to Localizing Flying Light Specks. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), October 27–31, 2025, Dublin, Ireland.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3746027.3759199

1 Introduction

Swarical is designed for a swarm of Flying Light Specks (FLSs) to illuminate shapes [2, 6–12, 14–18, 21, 24–27]. An FLS is a small drone with some storage, processing power, networking capability to communicate with other FLSs, and sensors to compute its position relative to another FLS [1, 3–5, 15]. Our artifacts include complete software implementations, scripts, datasets, and instructions to reproduce the experimental results presented in [4]. Swarical is designed to work with any device that enables an FLS to compute its position relative to another FLS. We present one such device in [4], namely, a small inexpensive Raspberry Pi Camera Module 3. Hence, two separate GitHub repositories contain our artifacts:

- Swarical, https://github.com/flyinglightspeck/Swarical.
- Position estimation using ArUco markers and Raspberry Pi Camera Module 3, https://github.com/flyinglightspeck/aruco-pose-estimation.

The next two sections describe each repository in turn, presenting how to reproduce the results of [4].

2 Swarical

Figure 1 shows the offline and online components of Swarical. The planner is the offline component. Its input is a mesh file for a shape, swarm size G, and the physical characteristics of each FLS, including its sensor, to estimate its position relative to another FLS. Its output is the number of FLSs F, a mix of FLSs, the number of groups nG, a swarm-tree, and nG FLS-trees.

The execution of the Swarical localization technique by an FLS is the online component. The input to each FLS is: `fls_id`, `swarm_id`, `anchor_id`, `children_ids`, and `ground_truth_location`. We emulate an FLS as a software process. A deployment consists of F processes as computed by the planner. Typically, we run these processes on a cluster of servers with one process (FLS) per CPU core. Hence, a large experiment with F=1000 FLSs requires 1000 cores. We separate the running of a localization experiment from the visualization of its results. We may run an experiment once to produce its log files and visualize those log files many times without repeating the experiment.

The mesh file and its required number of FLSs define the scale of an experiment, i.e., its number of cores. We start by introducing small scale experiments involving point clouds that require 16 FLSs. These introduce the experimentalist to the different point clouds considered in [4], the use of dead reckoning that distorts a shape, and the visualization of the output files generated by experiments. We provide the log files from previously conducted thousand core

MM '25, Dublin, Ireland

ACM ISBN 979-8-4007-2035-2/2025/10
https://doi.org/10.1145/3746027.3759199

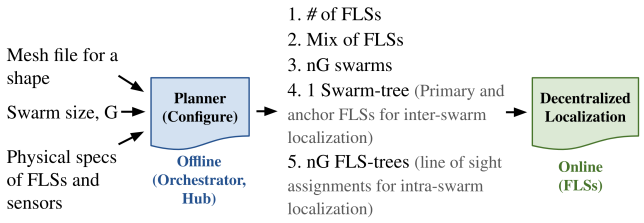

Figure 1: Swarical, a divide-and-conquer framework.

experiments for visualization. Section 2.2 describes how an experimentalist may run such an experiment to produce the log files.

2.1 Small Scale Experiments

Clone the Swarical repository:

```
git clone https://github.com/flyinglightspeck/Swarical.git
```

Open a browser to the Swarical GitHub link https://github.com/flyinglightspeck/Swarical and follow its README file. We recommend using Docker [19] to reproduce the results of this section. Depending on your operating system, install Docker Desktop or Engine. Once installed, ensure Docker is running before proceeding. From your desktop, open a terminal, change directory to the Swarical repository (`cd Swarical`), and execute its bash file (`bash init.sh`). The latter builds the Docker image required for reproducibility. It is simple to restart with a fresh environment by stopping and restarting the Docker daemon and executing the bash file. Once the bash file execution completes, it will show:

```
To access the server, open this file in a browser:
...
http://127.0.0.1:8888/tree?token=e8b9aa5a6e22952673c053fdf
```

The line starting with the `http://127.0.0.1:8888/...` will be different for your Docker instance. Copy and paste it into a browser. This opens an interface to the Jupyter Notebook running inside the Docker container. Open the notebook named `reproduction.ipynb`. Click the play button at the top of Jupyter to step through the software. The symbols next to Python software show whether the command is running (a *) or has completed execution (a number).

The software starts by introducing seven different shapes (i.e., datasets) used in the experiments of [4]. It instantiates the planner by specifying the radius of an FLS, and its camera's minimum and maximum range. The camera is the sensor an FLS uses to position itself relative to another FLS, see Section 3. Next, we load the planner with one of the shapes by specifying its mesh file. The planner reports the surface area of the mesh file in $meters^2$ and the number of required FLSs. Subsequently, it samples points from the mesh file and generates a point cloud. The user may visualize the resulting point cloud and how it may be illuminated with FLSs using dead reckoning [4]. With this technique, an FLS travels from a dispatcher to its destination with a random pre-specified angle of error not to exceed `alpha`. One may visualize the original point cloud, the ground truth, and the point cloud after the FLSs are dispatched to their assigned coordinate using dead reckoning.

Next, invoke the planner's compute_trees method to compute a swarm-tree consisting of multiple FLS-trees. The input is the approximate number of FLSs that should constitute a swarm. One may save the resulting swarm tree in a file, insert dark FLSs to provide connectivity between its disjoint swarms (or isolated FLSs), and visualize the swarm tree. All the above steps are fast and in total should not require more than one Second [1] to execute. One may generate a swarm tree for all seven shapes and a variety of swarm sizes in the order of a second or two.

The next section of the notebook, Decentralized Localization, highlights Swarical's online localization with a small scale experiment that one may execute on a laptop. It involves 16 FLSs (processes) illuminating a 2D rectangular array in a 4x4 grid. First, it shows the planner computing the swarm-tree. Next, it executes the localization technique. By default, it will execute for 30 seconds. It is computationally intensive for a small scale laptop or desktop. The program consists of one primary process, resembling the Orchestrator [15], and F=16 FLS processes. The FLS processes execute the decentralized localization technique ISR [4]. It requires FLSs to exchange messages and adjust their location continuously. We force it to terminate after 60 Seconds and to generate log files stored in the results directory. The files include the trajectory of each FLS and the messages they send and receive. The primary process uses these log files to compute the Hausdorff Distance and the Chamfer Distance as a function of time. These metrics show how closely the formation of FLSs matches the point cloud. Lower is better. The log files can be used to create a video visualization of FLSs moving to correct their position, converting the erroneous point cloud to the ground truth point cloud. FLSs that are members of the same swarm are shown with the same color. Make sure to copy the log file produced by Line 16 and paste it for use in Line 17. The process of creating the animation may require several minutes. To view the animation, use the browser to navigate to the Docker directory containing it and download the mp4 file. The Hausdorff distance is shown in the top left corner and drops below 1 cm in a few seconds.

The online experiments do not generate the exact same results each time due to the nondeterministic nature of their multitasking design. CPU scheduling, available resources, and the message passing delays vary the order in which the FLSs move, producing variation in the raw results, e.g, FLS trajectories. However, in each run, the results show Swarical is fast and quickly reduces Hausdorff and Chamfer Distances.

One may produce animations from an experiment's log files and view them unlimited times. Step 21 illustrates this for log files from thousand core experiments conducted using CloudLab and Amazon AWS. These experiments pertain to different variants of Swarical (HC, ISR, RSF) localizing the Skateboard. Their video generation may require tens of minutes. They correspond to our original experiments reported in Figures 13, 14, and 15, and the videos presented in the caption of Figure 13 (ISR, HC, and RSF). These files are automatically downloaded and placed in the results directory during the build process of the Docker image. A cell in the `reproduction.ipynb` is dedicated to generating the videos, and a Mathematica notebook, `swaricalplots.nb`, in the Swarical repository is provided to convert the raw results to the plots.

[1] On a MacBook Pro with M1 processor.

2.2 Decentralized Localization: Online Experiments

This section describes how to conduct the experiments that produced the log files visualized in Section 2.1. These were conducted using either CloudBank [20] or the Amazon AWS via CloudLab [23]. Our Swarical repository provides detailed instructions for each, see https://github.com/flyinglightspeck/Swarical/blob/main/CloudLab_README.md and https://github.com/flyinglightspeck/Swarical/blob/main/AWS_README.md, respectively. The choice of a cloud provider is not important because the results in [4] emphasize trends and comparisons instead of absolutes. What is important is for a cluster dedicated to an experiment to have either the same number of cores or a few more cores than the number of FLSs F required by illuminated shapes. (Recall that the planner computes F and reports its value for a shape, see Section 2.1.)

Our implementation uses the UDP broadcast protocol. It resembles FLSs in an area where they are communicating with one another. This must be manually set up with Amazon AWS, and the README file provides instructions for it. With the CloudLab, it is important for the PLATFORM in constants.py to be set to cloudlab. Our scripts use this variable to configure a deployment to limit the broadcast to a group of servers in the cluster. Otherwise, a deployment will broadcast messages throughout a data center, causing inefficiency for CloudLab and its users[2]. The README file provides instructions.

2.3 Datasets

We used five shapes from the Princeton Benchmark [22] to evaluate Swarical: a chess piece, a dragon, a palm, a skateboard, and a racecar. Planner converts them into a collection of 3D coordinates for spherical FLSs using Poisson disk sampling[13]. With Chess, we include both the original and a down-sampled version. We also include a kangaroo mesh file from free3d.com website. These files are placed in `assets/dataset/mesh` directory. Section 2.1 described how these are registered with the planner.

2.4 Reproducing Experimental Results

Figures 1, 5, 10, 11, and 12 of [4] were generated using the planner. The final plots for Figures 10, 11, and 12 were created using the Wolfram Mathematica software. A Mathematica notebook called `swaricalplots.nb` converts raw results to the plots.

3 Position Estimation Using a Camera

In [4], an FLS uses a Raspberry Pi Camera Module 3 NoIR[3] with autofocus and ArUco markers to compute its position relative to another FLS. The relative state between two FLSs u and v includes a position $\hat{\rho}_{u,v}$ and an orientation (roll, pitch, and yaw). Ideally, the accuracy of the position should be on the order of millimeters. The error in orientation should be less than 1° in each dimension.

The camera supports two types of lenses, regular and wide. Figure 6 of [4] shows the blind, sweet, and decaying range of the camera [5] with each lens. The camera does not detect an ArUco marker of a neighboring FLS when it is in its blind range, detects it with high accuracy when it is in its sweet range, and its detection becomes a function of distance when the marker is in the decaying range of the camera lens.

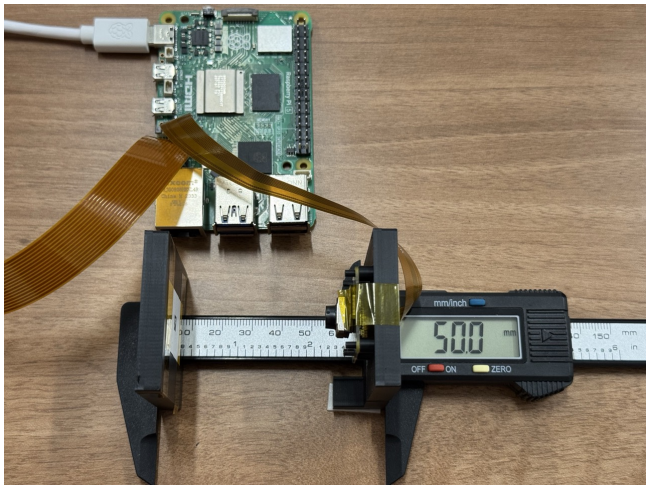

Figure 2: Position estimation using printed paper.

We evaluated two possibilities for tagging an FLS with an ArUco marker: 1) printing the ArUco marker on paper and pasting it on each FLS, and 2) displaying it on an LCD mounted on an FLS. Figure 2 shows our experimental setup with paper. It consists of:

- Raspberry Pi 5 with Raspberry Pi OS (bookworm) installed. This component hosts the software that uses the output of the camera to quantify a relative position.
- 3D printed parts for the Raspberry Pi Camera Module 3 (Wide or Regular) and the holder for the printed paper. The files for the 3D models are available in `assets/paper`.
- Caliper or ruler for measurements.

Figure 3 shows our setup with an LCD. It consists of the same caliper, Raspberry Pi 5, and camera as Figure 2 with the addition of:

- Waveshare 1.3 inch LCD Display Module IPS Screen 240x240.
- Arduino Uno Rev 3 to display the ArUco marker on the LCD.
- 3D printed parts for the Raspberry Pi Camera Module 3 (Wide or Regular), see `assets/lcd` for 3D printable models.

Figures 6 and 7 of [4] show the percentage error in the measured distances by the camera with both a wide and a regular lens using an ArUco marker on a printer paper with a 4.7 mm marker size. The same setup is used to quantify the orientation error reported in Figure 9 of [4]. Figure 8 of [4] highlights the difference between printing the ArUco marker on the paper versus displaying it on a small LCD. For each, it shows the percentage error in measured distance with either the regular or wide angle lens. Below, we describe how to reproduce these results. (Also, see https://github.com/flyinglightspeck/aruco-pose-estimation for the README file.)

With both setups, connect the camera to the Raspberry Pi (Pi for short) and turn it on. Clone and install the requirements on the Pi:

```
git clone https://github.com/flyinglightspeck/aruco-pose-estimation.git
cd aruco-pose-estimation
bash setup.sh
```

Verify that the Pi detects the camera using the following commands:

[2]Such broadcasts are considered a violation. Administrators may freeze the account of the violator and shut down the experiment.
[3]To function in the dark.

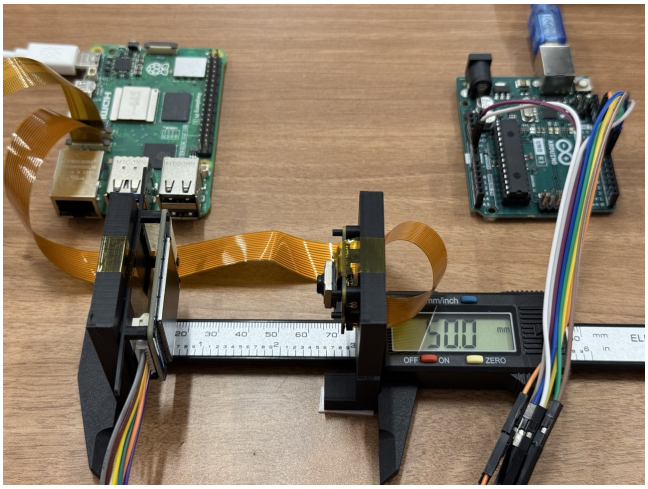

Figure 3: Position estimation using an LCD display.

```
rpicam-hello --list-cameras
rpicam-hello
```

Initiate the virtual environment:

```
source .env/bin/activate
```

Run the following command to start the program. It will run for 10 seconds and detect ArUco marker in each frame in real time:

```
python pi_pose_estimation.py -i pi3 -r 720p -t 10 --marker_size 0.0047 --live
```

The user may save the results using the following command:

```
python pi_pose_estimation.py -i pi3 -r 720p -t 10 --marker_size 0.0047 --save -e 80mm
```

The options `-i` and `-r` specify the camera type and its frame setting, respectively. (See Table 2 of [4] for details.) They utilize our provided calibration using the `calibration.py` file. The 720p setting was used for all experimental results reported in [4].

The README of the repository enumerates the possible arguments of pi_pose_estimation.py. By specifying the distance being measured using the `-e` option, the software computes and reports the observed error. This value should be set to the distance between the marker and the camera's image sensor. The 3d printed part has a placeholder for the camera's image sensor to facilitate adjusting the setup. One may conduct many experiments with different distances to populate a `/results` directory with result files and batch process them by issuing `python process.py`. This facilitates rapid data gathering with a final processing step. The code to generate the plots used in Figures 6-9 of the paper is in the `cameraplots.nb` Mathematica notebook.

4 Reproducibility Efforts

Our reproducibility study encompassed three experimental setups, using the public codebase identified in Section 1.

First, we used Linux servers to re-execute the small-scale experiments responsible for the illustrative figures of [4]. A subset of 3D meshes required minor, author-supplied code patches, and an early race condition in the parallel execution code produced transient error logs until corrected. Ultimately, all figures were successfully regenerated. Second, the large-scale experiments were also reproduced without numerical deviation, but only after a protracted installation and configuration process on an Amazon AWS cloud instance with large distributed computational power; this stage would include non-negligible economic costs.

Lastly, we re-conducted a subset of the experiments to quantify the error in visual pose estimation using ArUco fiducial markers. This took time because it required obtaining various hardware components, printing 3D models to reconstruct the experimental setup, and then re-conducting the experiment. We used a Raspberry Pi camera module 3 with a wide-angle lens and *with* IR filter, whereas the original paper tested both wide-angle and regular lenses – both *without* IR filters. This is because the camera of the original paper was unavailable during the reproduction. After clarifying several aspects of the experimental setup which were incorporated in this paper and the provided source code, we reproduced 4 of 5 experiments successfully, obtaining similar results with only small variations (likely due to manufacturing inconsistencies). Sharing the process via video, in addition to reading the paper, was crucial to accurately converging on the correct methodology. Both the original draft of this paper and its online documentations have been updated to incorporate constructive comments provided by the reviewers.

5 Conclusions

This paper describes how to reproduce the results of the Swarical localization technique [4] using a Raspberry Pi Camera Module 3. In caliberating this camera, one quickly discovers that it has a blind range. Using its wide lens, it cannot detect distances shorter than 50 mm. In [5], we introduce Swazure. It requires FLSs to cooperate by communicating with one another. This cooperation enables those FLSs that are closer than 50 mm to compute their relative position accurately. Swazure is a building block of Swarical when FLSs are required to operate in the blind range of their positioning sensor.

Acknowledgments

This research was supported in part by the NSF grants IIS-2232382 and CMMI-2425754. We gratefully acknowledge CloudBank [20] and CloudLab [23] for the use of their resources to enable all experimental results presented in [4] and their reproduction 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. doi:10.61981/ZFSH2301

[2] Hamed Alimohammadzadeh, Heather Culbertson, and Shahram Ghandeharizadeh. 2024. An Evaluation of Decentralized Group Formation Techniques for Flying Light Specks. In *Proceedings of the 5th ACM International Conference on Multimedia in Asia* (Tainan, Taiwan) *(MMAsia '23)*. Association for Computing Machinery, New York, NY, USA, Article 84, 7 pages. doi:10.1145/3595916.3626460

[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. doi:10.61981/ZFSH2302

[4] 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* (Melbourne VIC, Australia) *(MM '24)*. Association for Computing Machinery, New York, NY, USA, 6153–6161. doi:10.1145/3664647.3681080

[5] Hamed Alimohammadzadeh and Shahram Ghandeharizadeh. 2024. Swazure: Swarm Measurement of Pose for Flying Light Specks. In *The Second International*

Conference on Holodecks (Los Angeles, California) *(Holodecks '24)*. Mitra LLC, Los Angeles, CA, USA, 17–25. doi:10.61981/ZFSH2403

[6] Hamed Alimohammadzadeh and Shahram Ghandeharizadeh. 2025. Illuminating English Letters Using a Flying Light Speck. In *Proceedings of the 3rd International Workshop on UAVs in Multimedia: Capturing the World from a New Perspective* (Dublin, Ireland) *(UAVM '25)*. Association for Computing Machinery, New York, NY, USA, 5 pages. doi:10.1145/3728482.3757388

[7] 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. doi:10.1145/3587819.3592544

[8] Hamed Alimohammadzadeh, Shuqin Zhu, Jiadong Bai, and Shahram Ghandeharizadeh. 2024. Reliability Groups with Standby Flying Light Specks. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 1–11. doi:10.1145/3625468.3647606

[9] Hamed Alimohammadzadeh, Shuqin Zhu, and Shahram Ghandeharizadeh. 2025. Techniques to Conceal Dark Standby Flying Light Specks. *ACM Trans. Multimedia Comput. Commun. Appl.* (April 2025). doi:10.1145/3724399 Just Accepted.

[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. doi:10.61981/ZFSH2308

[11] Yang Chen, Hamed Alimohammadzadeh, Shahram Ghandeharizadeh, and Heather Culbertson. 2023. Towards Enabling Complex Touch-based Human-Drone Interaction.. In *IROS Workshop on Human Multi-Robot Interaction* (Detroit, USA).

[12] Yang Chen, Hamed Alimohammadzadeh, Shahram Ghandeharizadeh, and Heather Culbertson. 2024. Force-Feedback Through Touch-based Interactions With A Nanocopter. In *2024 IEEE Haptics Symposium (HAPTICS)*. 271–277. doi:10.1109/HAPTICS59260.2024.10520851

[13] 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. doi:10.1109/TVCG.2012.34

[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. doi:10.1145/3469877.3493698

[15] Shahram Ghandeharizadeh. 2022. Display of 3D Illuminations using Flying Light Specks. In *Proceedings of the 30th ACM International Conference on Multimedia* (Lisboa, Portugal) *(MM '22)*. Association for Computing Machinery, New York, NY, USA, 2996–3005. doi:10.1145/3503161.3548250

[16] Shahram Ghandeharizadeh. 2025. Flying Light Specks: Dronevision, Holodecks and Spatial Computing. In *Proceedings of the 3rd International Workshop on Multimedia Content Generation and Evaluation: New Methods and Practice (McGE '25)* (Dublin, Ireland) *(McGE '25)*. Association for Computing Machinery, New York, NY, USA, 2 pages. doi:10.1145/3746278.3759395

[17] Shahram Ghandeharizadeh and Luis Garcia. 2022. Safety in the Emerging Holodeck Applications. In *CHI 2022 Workshop on Novel Challenges of Safety, Security and Privacy in Extended Reality*.

[18] 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. doi:10.61981/ZFSH2304

[19] Dirk Merkel. 2014. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.* 2014, 239, Article 2 (March 2014).

[20] 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. doi:10.1145/3437359.3465586

[21] Trung Phan, Hamed Alimohammadzadeh, Heather Culbertson, and Shahram Ghandeharizadeh. 2023. An Evaluation of Three Distance Measurement Technologies for Flying Light Specks*. In *2023 International Conference on Intelligent Metaverse Technologies & Applications (iMETA)*. 1–8. doi:10.1109/iMETA59369.2023.10294597

[22] Philip Shilane, Patrick Min, Michael M. Kazhdan, and Thomas A. Funkhouser. 2004. The Princeton Shape Benchmark. In *2004 International Conference on Shape Modeling and Applications (SMI 2004), 7-9 June 2004, Genova, Italy*. IEEE Computer Society, 167–178. doi:10.1109/SMI.2004.1314504

[23] 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. doi:10.1145/844128.844152

[24] 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. doi:10.61981/ZFSH2309

[25] Nima Yazdani and Shahram Ghandeharizadeh. 2025. Integration of 3D FLS Displays with 3D Authoring Tools. In *Proceedings of the Third ACM International Workshop on Interactive Extended Reality* (Dublin, Ireland) *(IXR '25)*. ACM Press, New York, NY, 8 pages. doi:10.1145/3746269.3760418

[26] 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. doi:10.61981/ZFSH2303

[27] Shuqin Zhu and Shahram Ghandeharizadeh. 2024. Circular Flight Patterns for Dronevision. In *The Second International Conference on Holodecks* (Los Angeles, California) *(Holodecks '24)*. Mitra LLC, Los Angeles, CA, USA, 1–11. doi:10.61981/ZFSH2404