


Article

Design of a Rapid Structure from Motion (SfM) Based 3D Reconstruction Framework Using a Team of Autonomous Small Unmanned Aerial Systems (sUAS)

Douglas Shane Smith, Jr. and Hakki Erhan Sevil * 

Intelligent Systems & Robotics, University of West Florida, Pensacola, FL 32514, USA

* Correspondence: hsevil@uwf.edu

Abstract: The aim of this research effort was to develop a framework for a structure from motion (SfM)-based 3D reconstruction approach with a team of autonomous small unmanned aerial systems (sUASs) using a distributed behavior model. The framework is composed of two major goals to accomplish this: a distributed behavior model for a team of sUASs and a SfM-based 3D reconstruction using team of sUASs. The developed distributed behavior model is based on the entropy of the system, and when the entropy of the system is high, the sUASs get closer to reducing the overall entropy. This is called the grouping phase. If the entropy is less than the predefined threshold, then the sUASs switch to the 3D reconstruction phase. The novel part of the framework is that sUASs are only given the object of interest to reconstruct the 3D model, and they use the developed distributed behavior to coordinate their motion for that goal. A comprehensive parameter analysis was performed, and optimum sets of parameters were selected for each sub-system. Finally, optimum parameters for two sub-systems were combined in a simulation to demonstrate the framework's operability and evaluate the completeness and speed of the reconstructed model. The simulation results show that the framework operates successfully and is capable of generating complete models as desired, autonomously.

Keywords: 3D reconstruction; structure from motion (SfM); small unmanned aerial systems (sUAS); distributed behavior



Citation: Smith, D.S., Jr.; Sevil, H.E. Design of a Rapid Structure from Motion (SfM) Based 3D Reconstruction Framework Using a Team of Autonomous Small Unmanned Aerial Systems (sUAS). *Robotics* **2022**, *11*, 89. <https://doi.org/10.3390/robotics11050089>

Academic Editors: Charalampos P. Bechlioulis and Panagiotis Vlantis

Received: 20 June 2022

Accepted: 31 August 2022

Published: 4 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As technology advances and computational power increases for onboard equipment, more tasks can be allocated on smaller mobile platforms. This translates well onto autonomous small unmanned aerial systems (sUASs), allowing for increasingly complex tasks to be assigned and completed without any human interaction. With the resources becoming more affordable and available, it is possible to accomplish tasks that, when conducted by humans, are cumbersome and in some cases impossible. These tasks can exist in a wide range of applications, such as search and rescue, reconnaissance, post weather damage assessment, and many other military and commercial applications. One of these tasks, 3D reconstruction, is the focus of this study. Developing a framework that can rapidly deploy, control, and build a model of a 3D environment provides the capability to investigate an area in depth without the need for human interaction. For instance, one example mission can be for a team of sUASs to go out and locate an area of interest, record video, and then convert the video to a 3D model. In this study, this example mission is broken down into two essential parts: (i) a distributed model for a team of sUASs, and (i) 3D reconstruction of the model by combining the data from individual agents in the team. By using a team of sUASs, the process is sped up to complete the recording of the area of interest and share the computational load between agents, providing the benefit of being able to work simultaneously on the same area of interest, focusing on different parts of it. All of these points result in an efficient rapid framework with a low computational load on individual

agents in the team. Hence, the focus of this study was to evaluate the developed framework for 3D reconstruction using a team of sUASs.

The original contribution of this paper is that a single framework has been designed and developed that has a pipeline including both SfM-based 3D reconstruction and the distributed behavior of multiple sUASs together. The paper includes (i) the development of a distributed behavior model with decentralized rules; (ii) the development of the algorithm based on the entropy of the system to switch between different behaviors, such as grouping and 3D reconstruction phases; (iii) implementation of 3D reconstruction using an SfM approach with image data inputs from different sUASs. Additionally, an evaluation of several parameters was used to maximize the benefit of using the open source software, COLMAP [1], while reducing the computation time required to accomplish this task. Several experiments were conducted in different situations to evaluate the optimum positioning and settings dependent on the scenario. Experiments were performed to evaluate the optimum entropy threshold values that allow the team of sUASs to navigate effectively without interfering with each other or failing to navigate along the desired course, and then perform a coordinated 3D reconstruction mission in a distributed fashion. The intention of this paper is to both present the framework design with extensive analysis, and to show proof-of-concept results of the entire pipeline in an Unreal Engine simulator (AirSim) that is very accurate at representing real-world applications. The demonstration of real-world results using the developed framework is not included in the scope of originally intended contribution in this article, and is left for a future work.

The remainder of the paper is organized as follows. In the next section, related work from the literature is provided. In Section 3, details of the methods utilized in this study are discussed, along with an explanation of why they were selected as part of the pipeline. In Sections 4 and 5, parameter analyses for the distributed behavior model and the 3D reconstruction method are presented, respectively. A validation model and experiment results are described in Section 6. In Section 7, future directions and discussion are provided. Finally, Section 8 presents the conclusion.

2. Related Work

In dynamic and extreme environments with various threats, obstacles, and restricted areas, it is particularly challenging for multiple agents to operate autonomously. Most of the contemporary distributed behavior technology is inspired by nature, which provides good solutions for managing groups—i.e., fish schools, ant swarms, animal packs, bird flocks, and so on—for applications including but not limited to precision agriculture, infrastructure monitoring, security, telecommunication, and 3D model reconstruction. Several examples of behavior-based methods are worth mentioning here. Formation testing with reactive formation control that was divided into avoiding obstacles, avoiding robots, moving to the goal, and maintaining formation was presented by Balch [2]. Lawton et al. introduced three behavior control strategies: coupled dynamics formation control, coupled dynamics formation control with passivity-based inter-robot damping, and saturated control [3]. Monteiro et al. used nonlinear attractor dynamics to create a framework for controlling a team of robots [4]. Xu et al. presented a variant of an initial formation and a subsequent formation controller [5]. Other studies on formation control strategies have been presented by Olfati-Saber [6], mixing potential field and graph-based ones, and, more recently, by Vásárhelyi [7]. One method involved modeling behaviors by defining the environment with attractors and repellers to maintain a triangle formation and avoid obstacles [8]. A vision-based method where general formation control with local sensing has also been presented in the literature [9].

Leader–follower and virtual leader–follower methods have also been presented in the literature [10,11]. Zhang et al. proposed a cooperative guidance control method based on backstepping that allows the desired formation and achieves a multi-UAV steady state [10]. A flocking-based control strategy for UAV groups was presented by Liu and Gao [11]. More similar to the developed method is an approach presented by Lee and Chwa that

is decentralized-behavior-based [12] and only uses relative distance information between neighbors and obstacles. Another similar method is consensus formation control [13]. Limited work on entropy control or analysis of multi-agent systems exists in the literature, aside from a recent paper on potentially using cross-entropy to determine the robustness of a multi-agent group [14]. Areas of active research for applications in formation control include, but are not limited to, the following: improvements towards autonomous farming with swarm robotics for agricultural applications [15], satellites in space [16], and natural disaster relief [17]. In [18,19], the researchers used a particle swarm optimization (PSO)-based algorithm to optimize the coverage of an area using UAVs. In [20], a method of labeling each UAV as it accomplishes tasks was used to track and coordinate movement using a modified genetic algorithm (GA). Wildermuth and Schneider investigated computer vision techniques to build a common coordinate system for a group of robots [21]. Researchers from Carnegie Mellon University used dynamic role assignment through artificial potential fields (APF) in [22] to coordinate the movement of a team for Robocup. Lakas et al. used a leader–follower control technique to navigate and maintain a swarm formation for an autonomous UAV swarm [23]. Moreover, a swarm formation method with a heterogeneous team including ground and aerial platforms is presented in [24]. Many of these have applications in surveillance and reconnaissance, as shown in [25–27]. The approach presented by MacKenzie allocates constrained subtasks, and during assignment, robots can incorporate the cost of meeting constraints due to each other [26]. Li and Ma introduced an optimal controller that is suitable for multi-agent cooperative missions based on the idea of leader–follower methods [27]. Unlike these complex and complicated methods, the main focus in this study was to implement entropy-based distributed behavior, which demonstrates a simple means of coordination and can be easily applied to a sUAS.

In the literature, various studies have been presented that are related to 3D modeling approaches. These studies can be grouped depending on their application areas and 3D modeling techniques. Some of the applications of 3D modeling presented in the literature include 3D building modeling [28], 3D city modeling [29], automatic registration for georeferencing [30], railroad center line reconstruction [31], automatic building extraction [32], scene parsing [33], elevation mapping [34], species recognition, height and crown width estimation [35], target localization and relative navigation [36], and visual localization [37]. Modeling techniques used in these studies involve automatic aerial triangulation, coarse-to-fine methods [28], digital surface nodes application [29], the iterative closest-point (ICP) algorithm [30], the random sample consensus algorithm [31], the binary space partitioning (BSP) tree [32], the Markov-random-field-based temporal method [33], fuzzy logic and particles [34], multi-scale template matching (MSTM) [35], dynamic bias estimation [36], and localization-by-recognition and vocabulary-tree-based recognition methods [37].

Structure from motion (SfM) is one of several methods of analyzing and reconstructing a 3D model, as described in [1,38], and it is capable of reconstruction using frames from unstructured imagery. Given that high-resolution cameras are now inexpensive and installed on nearly every commercially available Unmanned Aerial Vehicle (UAV), utilizing the methods described is a more practical solution. Furthermore, the methods described in [1,38] have been developed in open source software, COLMAP, that can be controlled via a graphical user interface (GUI) for human interaction and review, and Command Prompt commands that can easily be utilized in the software. It can also be built on various operating systems, which allows it to be easily applied to a host of possible onboard computational units. This ability to run the software from Command Prompt also allows it to be integrated as a sub-system into the general sUAS so that it can be executed autonomously. A comparison of several SfM methods using a wide variety of software is presented in [39–42] with a focus on comparing the speed and accuracy using a set number of photos. In this paper, another focus is finding the most efficient camera poses and using that information to reduce the time it takes to create a model.

One of the most relevant works was presented by Mentasti and Pedersini [43]. In that study, control strategies for a single UAV were provided for 3D reconstruction of large

objects through a planned trajectory. Contrary to that study, in this study, a framework for multi-agent sUAS is presented. The other most relevant work is demonstrated in [44]. It accomplishes a similar task as presented in this paper, but it uses different reconstruction software and object detection methods. More importantly, the framework presented in this paper uses the developed distributed behavior model, which allows a multi-agent system to coordinate their motion during 3D model reconstruction missions. Similarly, Daftry et al. focused on the 3D reconstruction of a building using micro aerial vehicles (MAV); however, their approach is not a complete autonomous system for reconstruction [45]. Gao et al. used a simulated environment to reconstruct a 3D model from images, which is explored in this research as well [46]. All of these methods contribute in some part, but there is no single example of the framework developed in this research.

3. Methodology

The developed framework has two major parts: the distributed behavior model and 3D model reconstruction. The general scheme of the framework can be described as follows. The developed distributed behavior model is based on the entropy of the system. In this model, the goal is to keep the entropy less than a predefined threshold value, and the sUASs, which have decentralized rule-based controller, reduce the entropy by grouping together. As sUASs begin their application, the first thing they do is to come close to each other until entropy is below the threshold (grouping phase). Once the entropy is less than the threshold, the sUASs begin their 3D reconstruction mission (mission phase). The sUASs are given the object in the location of interest, and they coordinate their motion and formation automatically. They can switch grouping phase if they fall apart, and then they can come closer and continue the 3D reconstruction mission. For 3D reconstruction, the sUASs complete a circular orbit while taking photos. Finally, those photos are fed into the SfM algorithm to create a 3D model of the object. The following sub-sections provide details of the dynamic model of sUASs, the developed distributed behavior model, and the 3D reconstruction approach.

3.1. Dynamic Model of sUASs

In the simulation experiments, the sUAS model was used [47]. The sUAS was defined as four connected vertices with thrust forces F_i , torques from the propellers τ_i , and control inputs u_i (Figure 1).

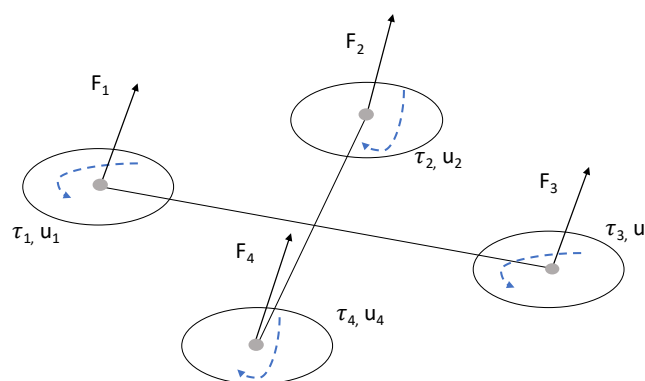


Figure 1. Representation of the sUAS model.

The forces and torques [47] are defined as

$$F_i = C_T \rho \omega_{max}^2 D^4 u_i \tag{1}$$

$$\tau_i = \frac{1}{2\pi} C_{pow} \rho \omega_{max}^2 D^5 u_i \tag{2}$$

where D is the propeller’s diameter, ρ is the air density, ω_{max} is the max angular velocity in revolutions per minute, C_T is the thrust coefficient, and C_{pow} is the power coefficient. In the

simulator, the magnitude ($|\cdot|$) of the linear drag force on the body is used, and it is defined as [47]:

$$|\mathbf{F}_d| = \frac{1}{2}\rho|\mathbf{v}^2|C_{lin}A \tag{3}$$

where A is the vehicle cross-section, C_{lin} is the linear air drag coefficient, \mathbf{v} is the velocity vector, and drag force acts in the opposite direction to the velocity vector. Consider an infinitesimal surface area ds in the sUAS body with the angular velocity ω . The linear velocity $d\mathbf{v}$ experienced by ds equals to $\mathbf{r}_{ds} \times \omega$; thus, the linear drag equation for ds becomes [47]

$$|d\mathbf{F}| = \frac{1}{2}\rho|\mathbf{r}_{ds} \times \omega|^2C_{lin}ds \tag{4}$$

where direction of $d\mathbf{F}$ is $-\mathbf{r}_{ds} \times \omega$. The drag torque can now be computed by integrating over the entire surface as $\tau_d = \int_S \mathbf{r}_{ds} \times d\mathbf{F}$, and the body for the drag force is approximated as a set of connected faces and then approximated as a rectangular box. Further, the net force is calculated as [47]

$$\mathbf{F}_{net} = \sum_i \mathbf{F}_i + \mathbf{F}_d \tag{5}$$

and the torque is calculated as

$$\tau_{net} = \sum_i |\tau_i + \mathbf{r}_i \times \mathbf{F}_i| + \tau_d \tag{6}$$

3.2. Distributed Behavior Model

Entropy was used as a form of distributive behavior model for the team of sUASs. Specifically, Tsallis entropy was utilized, which is defined as [48]

$$S_T = \frac{1 - \sum_i p_i^q}{q - 1} \tag{7}$$

When the sUAS are far apart, the entropy of the equation is high, which means that system’s stability is low. Low stability is an indicator that the probability of the system remaining cohesive is decreasing. This approach works for homogeneous and non-homogeneous systems alike, because it generalizes the model as a system of particles. Each agent has a case dependent variable, X , so that the probability density function is calculated using [25,49–52]:

$$p_i(X) = \frac{1}{\Gamma(k)\theta^k} X^{k-1} e^{-\frac{X}{\theta}} \tag{8}$$

where k and θ are shape and scale parameters, respectively. During execution, each agent calculates the entropy for itself and shares it with all neighboring agents. The goal with this method is to reduce the entropy of the system to the lowest value achievable while still remaining stable. The algorithm takes the current location and heading (x,y,θ) , the desired coordinate (x,y) , and the maximum velocity as inputs; and outputs command velocity and heading, per sUAS. Distances and angles between each sUAS and the destination are calculated, then the nearest and farthest agents are determined. Error is bounded between a maximum and a minimum distance, and then the entropy of the system is calculated by summing $\frac{e_i}{d_{max}^q}$ divided by the $q - 1$. The pseudo-code representation is shown in Algorithm 1.

More clearly explained, if the sUAS has a entropy value that is greater than the pre-defined threshold, it is urged to group with its closest neighbor. This phase is called “grouping” phase. Then, the formed sub-group(s) tries to get closer to the next farthest sUAS agent or other sub-group, if the entropy values are still greater than the threshold.

One safety step is checking whether the sUASs are too close; if so, the algorithm commands them to move apart. That allows sub-groups to come closer instead of agents inside a sub-group to come very close to each. Moreover, if entropy values are less than the threshold value, sUASs start moving to a global way-point. This phase is called the “mission” phase.

Algorithm 1: Pseudo-code representation of Tsallis-entropy-based distributed behavior

Input : $[X_{cur}, Y_{cur}, \theta_{cur}]$ for each sUAS, (X_{des}, Y_{des}) Global Waypoint, V_{max}

Output: V_{com}, θ_{com}

Distances and Angles are calculated between each sUAS and then to the desired waypoint

Distances between sUAS are compared and the closest and furthest sUAS are identified

if $d_{betweensUAS} > d_{max}$ **then**

$e = d_{max}$

else if $d_{betweensUAS} < d_{max}$ & $d_{betweensUAS} > d_{min}$ **then**

$e = d_{betweensUAS}$

else if $d_{betweensUAS} \leq d_{min}$ **then**

$e = d_{min}$

$$S_T = \frac{1 - \sum_i^{numsUAS} (\frac{e_i}{d_{max}})^q}{q-1}$$

$Threshold \leftarrow S_T$

if $S_T < Threshold$ **then**

if $d_{closestsUAS} < d_{min}$ **then**

$\theta_{com} = \theta_{waypoint} - \theta_{cur}$

$V_{com} = V_{max}$

else

$\theta_{com} = (\theta_{index} - \theta_{cur}) + \pi$

$V_{com} = V_{max}$

else if $d_{closestsUAS} > d_{min}$ **then**

$\theta_{com} = \theta_{index} - \theta_{cur}$

$V_{com} = 2 * V_{max}$

else if $d_{furthestUAS} > d_{min}$ **then**

$\theta_{com} = \theta_{index2} - \theta_{cur}$

$V_{com} = 2 * V_{max}$

else if $d_{closestsUAS}$ or $d_{furthestUAS} < d_{min}$ **then**

$\theta_{com} = (\theta_{index} - \theta_{cur}) + \pi$

$V_{com} = 1/2 * V_{max}$

Map all angles between $-\pi$ to π

To test this model, simulations were run in AirSim [47]. To ensure the robustness and scalability of the model, simulations were run with 3, 6, 9, 12, 15, 18, and 20 sUAS. Using this method, it was possible to collect enough data to find suitable values to complete the simulation missions. This capability led to the discovery that, along with the Tsallis entropy, the developed model needs to have two other parameters to effectively control the sUAS team. The additional parameters are:

- Minimum distance
- Entropy threshold

3.2.1. Minimum Distance

The minimum distance was added to avoid collisions between each sUAS in the team. Since the goal is to reduce the overall entropy, the sUASs move closer together, and all of them try to occupy the same physical space. This causes collisions and makes maneuvering challenging. To avoid this, the minimum distance was set. This minimum distance makes it possible for each sUAS to operate without interference from the others. Figure 2 represents

the three sectors within the minimum distance that the sUAS will monitor. If another sUAS exists in any of the sectors, the sUAS will attempt to separate, but the sector will determine how they will separate. If another sUAS is in the “front-left” sector, the sUAS will determine the angle relative to other sUAS and subtract 90 degrees; thus, it will make a right turn and avoid the other sUAS. Similarly, in the “front-right” sector case, the sUAS will determine the angle and add 90 degrees to turn left. If a sUAS is determined to be in the “behind” sector, 180 degrees will be added (and readjusted to between 0 and 360), making the sUAS move directly away from the other sUASs. The reason for introducing different sectors is that there are certain cases in which the distributed behavior model cannot be stabilized. Those cases are (i) the sUAS being on the edge of the entropy threshold, (ii) the sUAS being on the edge of the minimum distance threshold, and (iii) the sUAS being on a nearly parallel path with its nearest neighbor. Outside of those cases, if the sUASs perform a 180 degree maneuver, that leads to oscillating behavior and the whole system becomes unstable. Consider two sUAS in such a situation. Both sUAS will be stuck in a loop of turning 180 to get away from the minimum distance and then immediately turn back 180 to get within entropy threshold value. By assigning quadrants, it is possible to make smaller adjustments and let the sUAS “drift” closer together on their flight path.

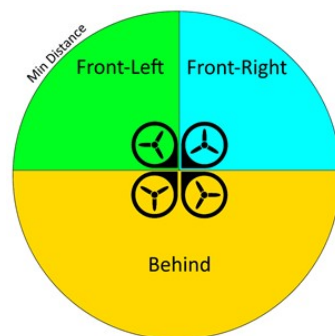


Figure 2. Diagram illustrating the sUAS minimum distance.

To operate effectively, each sUAS needs to monitor three different ranges, as shown in Figure 3. Within the minimum distance, no other sUAS should be present. If any sUASs do enter this range (depending on where they are), then the sUASs will make adjustments to separate, as shown in Figure 2. Within the maximum grouping distance, sUASs will attempt to place all the other sUASs in this range. If they are too close, they will separate, and if they are too far, they will move towards each other. Within the maximum distance, the sUASs will measure their distances and attempt maneuvers to move closer. Anything outside this distance is not weighted more heavily but will still contribute to high entropy.

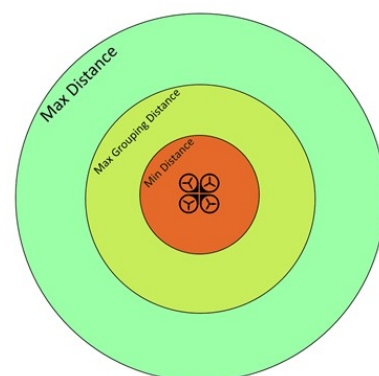


Figure 3. Diagram illustrating the different ranges that a single sUAS will monitor for other sUASs.

3.2.2. Entropy Threshold

Similar to the way that the physical size of the sUAS interferes with reducing the overall entropy, the physical size of the team interferes with occupying the required system entropy. To make the solution scalable, the entropy threshold needs to be changed based on the size of the team. The entropy threshold forms a “bubble” that the team must stay inside of. A “bubble” that works for a team of three sUAS is physically too small for a team of 20 sUAS to occupy. Therefore, the entropy threshold needs to be increased based on how many sUAS are in the team. Figure 4 represents this concept in a 2D example.

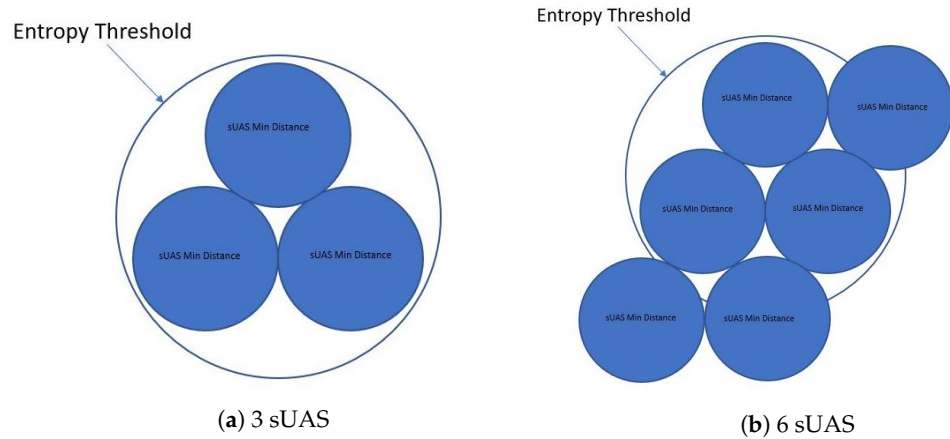


Figure 4. 2D representation of 3 and 6 sUAS teams occupying the same entropy “bubble”.

To determine a suitable entropy threshold, a simulation mission was run using different values for the entropy threshold. The simulation mission starts with the team in a straight line with each sUAS separated. They are initialized, and then we move into the “grouping” phase of the mission where the goal is to reduce the entropy below the threshold value. Once this is achieved, the team moves into the “mission” phase that consists of navigation through four waypoints. More details and results are provided in the parameter analysis section.

3.3. 3D Model Reconstruction

The structure from motion method uses a series of images and builds a 3D representation of the environment or object in interest. This is done by first taking a set of input images and extracting or detecting features in the images. Once these features are detected, they are then matched with the same features in other images. Once enough features are detected, it is possible to estimate both the 3D geometry of the point (structure) and the camera pose (motion) [53]. This is represented as

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^m \sum_{j=1}^n W_{ij} \cdot \left\| \mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2 \tag{9}$$

The solution of this equation is also known as bundle adjustment. There are several proven techniques that can be used to accomplish each of these steps. For the 3D reconstruction step of the framework, COLMAP [1] was adopted to complete the SfM-based reconstruction using unordered images. It is well documented and allows the user to incorporate it into a custom framework with minimal modifications. It is an end-to-end application capable of taking multiple photos or videos and generating a model based on some user defined parameters.

The method of implementing the SfM feature extraction process is improved by augmenting the scene graph with appropriate geometric relations. For each image I_i , algorithm detects sets $\mathcal{F}_i = \{(\mathbf{x}_j, \mathbf{f}_j) | j = 1 \dots N_F\}$ of local features at location $\mathbf{x}_j \in \mathbb{R}^2$ represented by an appearance descriptor \mathbf{f}_j . An initial fundamental matrix is calculated,

and if it is determined to have N_F inliers, then the images are considered geometrically verified. Next, the number of homographic inliers N_H is used to classify the transformation of the imaged pair. To approximate model selection methods such as the geometrical robust information criterion (GRIC), it is assumed that a moving camera in a general scene is $N_H/N_F < \epsilon_{HF}$. In the case of correct calibration and $N_H/N_F < \epsilon_{HF}$, the essential matrix is decomposed, the points from inlier correspondences are triangulated, and the median triangulation angle, α_m , is determined. α_m is used to differentiate between cases of pure rotation and planar scenes. For valid pairs, the scene graph is labeled with its model type (general, panoramic, planar) alongside the inliers of the model with maximum support N_H, N_E, N_F . Here, the model type is leveraged to seed the reconstruction only from the non-panoramic with a preference for the calibrated image pairs. Triangulation from the panoramic case is not calculated to avoid degenerative points and improve robustness of triangulation, and follow on image registrations.

The following step in the framework is to improve next best view selection by keeping track of the number of visible points and their distribution in each image. These two parameters are used to determine a score \mathcal{S} , where more distributed points and a more uniform distribution result in a higher score. This allows for images with the highest score to be registered first. This is achieved by discretizing each image into cells of K_I in both dimensions. Then, each cell is labeled either *empty* or *full*. This step is repeated with each cell being subdivided with the score \mathcal{S} being updated.

To effectively triangulate images and handle arbitrary levels of outlier contamination, multiview triangulation is performed using random sample consensus (RANSAC) [54]. The feature track is defined as $\mathcal{T} = \{T_n; n = 1 \dots N_T\}$, which is a set of measurements with prior unknown ratio ϵ of inliers. A measurement T_n consists of normalizing image observation $\bar{\mathbf{x}}_n \in \mathbb{R}^2$ and corresponding camera pose $\mathbf{P}_n \in \mathbf{SE}(3)$. This defines the projection from world to camera frame $\mathbf{P} = [\mathbf{R}^T \ -\mathbf{R}^T \mathbf{t}]$ with $\mathbf{R} \in \mathbf{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$. The objective is to maximize support of the measurement conforming with two-view triangulation.

$$\mathbf{X}_{ab} \sim \tau(\bar{\mathbf{x}}_a, \bar{\mathbf{x}}_b, \mathbf{P}_a, \mathbf{P}_b) \text{ with } a \neq b, \tag{10}$$

where τ is any triangulation method and \mathbf{X}_{ab} is the triangulation point. A well conditioned model must satisfy two constraints. First, a sufficient triangulation angle α :

$$\cos \alpha = \frac{\mathbf{t}_a - \mathbf{X}_{ab}}{\|\mathbf{t}_a - \mathbf{X}_{ab}\|_2} \cdot \frac{\mathbf{t}_b - \mathbf{X}_{ab}}{\|\mathbf{t}_b - \mathbf{X}_{ab}\|_2} \tag{11}$$

Second, positive depths d_a and d_b with respect to the views \mathbf{P}_a and \mathbf{P}_b with the depth defined as

$$d = [p_{31} \ p_{32} \ p_{33} \ p_{34}] \begin{bmatrix} \mathbf{X}_{ab}^T \\ 1 \end{bmatrix}^T, \tag{12}$$

where p_{mn} denotes the element in row m and column n of \mathbf{P} . A measurement T_n is considered to conform with the model if it has positive depth d_n and if its reprojection error,

$$e_n = \left\| \bar{\mathbf{x}}_n - \begin{bmatrix} x'/z' \\ y'/z' \end{bmatrix} \right\|_2 \text{ with } \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{P}_n \begin{bmatrix} \mathbf{X}_{ab} \\ 1 \end{bmatrix}, \tag{13}$$

is smaller than a threshold t . RANSAC maximizes \mathcal{K} iteratively and usually has a sample size of two. This makes it likely to sample the same minimum set multiple times, so COLMAP only generates unique samples. To ensure confidence of η that at least one outlier-free minimum has been sampled, RANSAC must run for at least K iterations. This process is run recursively until the consensus set is less than three.

At this point, bundle adjustment (BA) is performed when the growth model has reached a certain percentage. To account for potential outliers, the Cauchy function is utilized as the robust loss function. After the BA, any observations with large projection errors are filtered out, and retriangulation (RT) is performed to improve the completeness

of the reconstruction. BA and RT are performed iteratively until the amount of filtered out observations diminishes. The BA step is a major bottleneck for SfM, so to improve the speed of the pipeline, similar images are clustered together. Given the application used in this study, this is unlikely to be due to the fact that a very limited number of photos is used in an effort to decrease computation time. Details of the approach and all of the equations can be found in [1].

In this study, several trials were evaluated using different parameters. Those parameters were:

- Processing quality;
- Number of photos;
- Camera angle;
- Multiview formation;

Using these parameters, it was possible to generate 184 trials using two datasets. First, the benchmark dataset “Cat” was used from ToHoku University Multiview Stereo (THU-MVS) Datasets [55]. This dataset consists of 108 images that include three sets of 36 images. Each set has 36 images taken from a circumferential camera position separated by 10 degrees. These image locations are depicted in Figure 5.

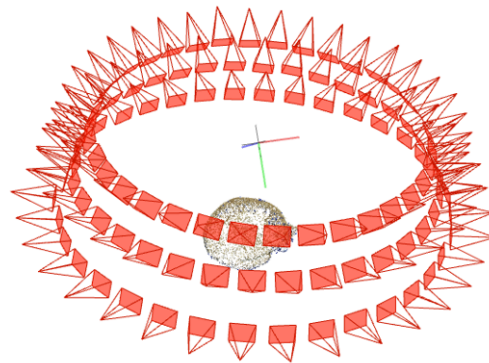


Figure 5. Camera Pose for benchmark dataset [55].

Using the observations from the benchmark dataset, a custom dataset consisting of 96 images was created. The created dataset used an outdoor setting and a camera phone. This new dataset was meant to represent a more realistic environment outside of a controlled setting. The camera pose for these images was adjusted similarly to 32 images separated by approximately 11.25 degrees. This dataset generated 124 trials that were each evaluated similarly to the benchmark dataset. The third dataset was generated in the AirSim environment [47]. Details of all datasets are given in more detail in the following sections.

In this study, COLMAP was utilized as a tool for reconstructing the 3D model. The focus was to optimize the developed framework by reducing the input to a functional level. To determine the minimum data required to build a suitable model, camera locations were changed and results were compared. The contribution in terms of 3D reconstruction was determining the optimal locations to generate a 3D model rapidly.

4. Distributed Behavior Model Parameter Analysis

For analysis of the developed distributed behavior model, four parameters were tested to characterize the behavior of the developed algorithm. In the analysis, three sUAS were used, and the goal was for sUAS to group first and then to go a pre-defined waypoint. The trials were run until all three sUAS were within 20 m of the waypoint. The following were the cost functions used to compare different results.

$$f_1 = \sum_i^{numUAVs} \sqrt{V_X^2 + V_Y^2} \quad (14)$$

$$f_2 = \sum_i^{numUAVs} \sqrt{\Delta X_{Total}^2 + \Delta Y_{Total}^2} \quad (15)$$

The sUAS were controlled via velocity control, so the cost function in Equation (14) characterizes the summation of the total control inputs per sUAS. The cost function in Equation (15) characterizes the summation of total distance traveled per sUAS. No exact parameter set was the best in all cases for parameter analysis. The exact application dictates which parameter set is preferred. The resulting cost function values and the parameters used for each trial are given in Tables 1 and 2.

The parameters tested were threshold, maximum velocity (V_{max}), maximum distance (d_{max}), and q (Table 2). Minimum distance (d_{min}) was manipulated in the latter half of the trials to prevent the entropy of the system from becoming negative. The q is the Tsallis entropy constant given in Equation (7). Maximum distance is the divisor for the summation of entropy of the system. Maximum velocity is the highest allowed velocity per sUAS. Threshold is the transition point in system entropy to switch from the grouping phase to the 3D reconstruction mission phase.

Trials 1–3 were performed by varying the threshold parameter. Depending on the threshold value, the entropy of the system kept decreasing as sUASs got closer together (grouping phase); then, they started going to the waypoint (mission phase). The sUASs ended up closest in Trial 1 before going to the waypoint (Table 1). This resulted in the highest cost for velocities, f_1 , and distance, f_2 , in comparison to Trials 2 and 3. As the threshold increased, the cost functions f_1 and f_2 decreased because less grouping resulted in lower velocities and lower overall distances traveled to the waypoint. Trials 4–6 varied the maximum velocity parameter. Higher maximum velocity resulted in less stable individual agent trajectories, but the system of agents still effectively reached the waypoint without colliding. Trial 4 resulted in a similar trajectory to Trial 1, as is evident in the parameter costs (Table 1). Trial 6 had the highest cost of f_1 of any trial and the highest average f_2 . Thus, the general trend is: as the maximum velocity increases, so do the cost functions f_1 and f_2 .

Trials 7–9 varied the maximum distance parameter. Changing the maximum distance had minor effects on the overall behavior because minimum distance must be increased to prevent negative entropy in the system. The largest distinction is that the calculated overall entropy of the system was smaller than in other parameter runs. Trials 10–12 varied the q parameter, which is given in Equation (7). The q parameter greatly affects the scale of the entropy of the system. With q of 0.6, 0.75, and 0.9, the highest entropy became 5, 8, or 20, respectively. With a higher q , the trajectories become less smooth, which is undesirable in most applications. As the parameter q increases, the cost functions f_1 and f_2 also increase.

Table 1. Parameter analysis results—cost function values, f_1 and f_2 , in 12 trials.

	f_1 sUAS1	f_1 sUAS2	f_1 sUAS3	f_2 sUAS1	f_2 sUAS2	f_2 sUAS3
Trial 1	1706	1706	1706	177	174	177
Trial 2	1632	1632	1632	168	166	170
Trial 3	1387	1387	1387	145	144	140
Trial 4	1747	1748	1748	180	176	181
Trial 5	1755	1755	1755	179	175	181
Trial 6	3105	3105	3105	237	219	249
Trial 7	1672	1672	1672	172	169	174
Trial 8	1778	1778	1778	180	168	181
Trial 9	1593	1593	1593	161	151	165
Trial 10	1730	1730	1730	178	175	180
Trial 11	2244	2179	2244	199	206	195
Trial 12	2856	2560	2865	248	188	245

Table 2. Parameter analysis results—parameter values used in 12 trials.

	Threshold	V_{max}	d_{max}	q	d_{min}
Trial 1	0.5	0.5	100	0.5	12
Trial 2	1.5	0.5	100	0.5	12
Trial 3	3	0.5	100	0.5	12
Trial 4	0.5	0.75	100	0.5	12
Trial 5	0.5	1	100	0.5	12
Trial 6	0.5	5	100	0.5	12
Trial 7	0.5	0.5	150	0.5	17
Trial 8	0.5	0.5	200	0.5	25
Trial 9	0.5	0.5	250	0.5	30
Trial 10	0.5	0.5	100	0.6	12
Trial 11	0.5	0.5	100	0.75	24
Trial 12	0.5	0.5	100	0.9	30

Additionally, multiple waypoint navigation tests were performed in a simulation environment, and results are depicted in Figures 6 and 7 [56]. In the experiments, the team of sUASs was commanded to navigate between defined waypoints. In Figure 6, the effect of the entropy threshold value can be seen during the experiment with three sUAS. As the selected entropy threshold increases, the smoothness of the sUAS trajectories increases. Increasing the entropy threshold is another means by which to configure system grouping and mission priorities. The results with a threshold of 1.0 were determined as the best due to an adequate mix of smooth trajectory and early formation cohesion. The top plots of Figures 6 and 7 depict the $x - y$ position of each sUAS, and the lower plots depict the entropy values over time.

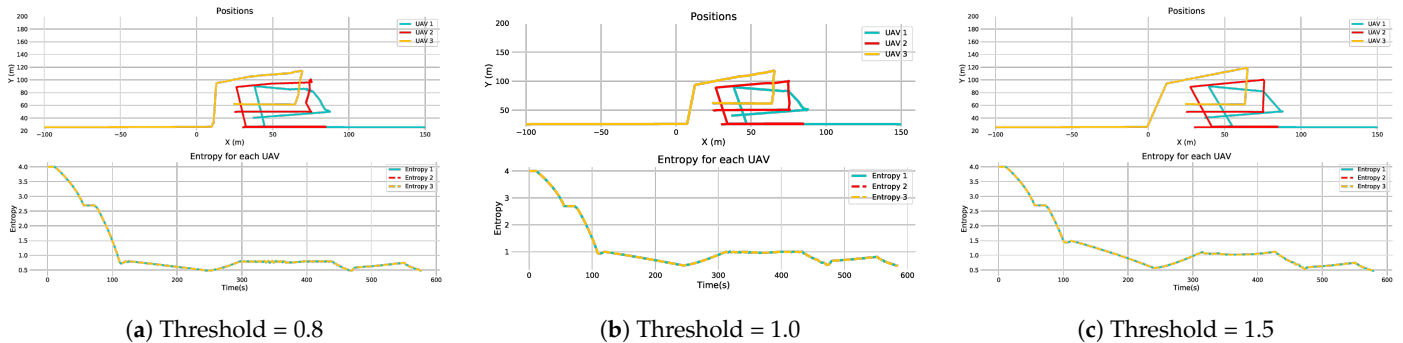


Figure 6. Parameter analysis with three sUAS.

In order to demonstrate the effect of an increased number of sUASs in the team, results from simulations with six sUAS are depicted in Figure 7. As new sUAS agents are added to the team, they populate the simulation environment in a linear pattern. As the team grows, so must the entropy threshold value. Close inspection of the entropy values for each sUAS show that as they move towards the first objective, they also reduce their entropy by decreasing the distance between each other. Initially, the team grouped together to reduce the entropy of each sUAS to below the threshold. Once this was achieved, they continued to group as they moved to the waypoint one until they reached the minimum distance limit. Once each sUAS reached its waypoint, the entropy of each sUAS increased so that it was possible to reach its destination and form the echelon formation.

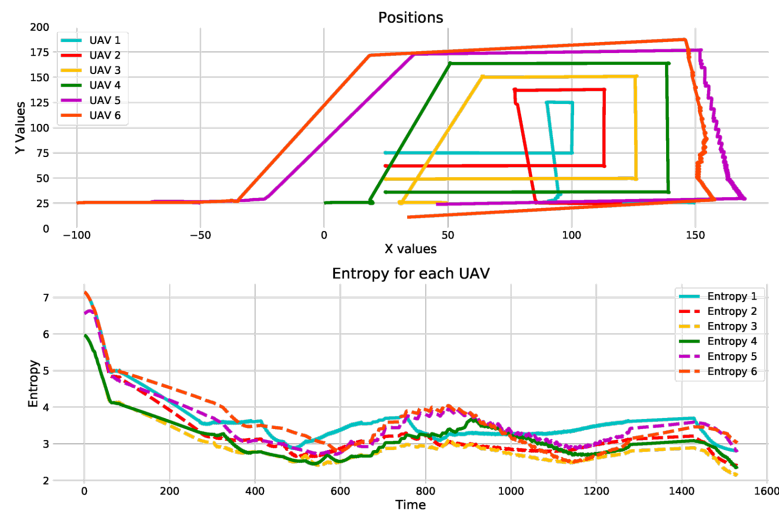


Figure 7. Parameter analysis with six sUASs.

Further, entropy threshold values for teams of 3, 6, 9, 12, 15, 18, and 20 were determined by trial and error. A bracketing method was used where the entropy threshold values were changed between low failure and high failure values until the mission was completed at the minimum entropy threshold setting. Table 3 shows the team size and entropy threshold values. The number of sUASs was plotted as a function of the entropy threshold, and Equation (16) was derived. Using all of the values, the final entropy equation was found to have an R^2 value of 0.9984. Using Equation (16), entropy threshold values for 5, 8, 11, 14, 17, and 19 sUASs were calculated, and trials were run to confirm the accuracy of the equation. In all cases, the mission was completed successfully with using the calculated entropy threshold, and stability of the system was confirmed.

$$y = 1.11x - 2.57 \tag{16}$$

Table 3. Entropy values for team by number of sUASs.

Number of sUAS in the Team	Entropy Value
3	1
6	3.9
9	7.3
12	10.6
15	14.3
18	18
20	19.7

5. 3D Reconstruction Parameter Analysis

The COLMAP software was designed to take large datasets of photos (several thousands) and reconstruct 3D models of environments from the images. For the application addressed in this study, the goal is to reconstruct a model of just an area or object of interest so that it can be inspected in some detail and a determination can be made based on that information. The first step taken in this endeavor was to use a benchmarking dataset in ideal circumstances to determine (i) the fewest required photos to build a model and (ii) the best locations for the camera to take photos. This was done using the dataset available in [55] provided by ToHoku University. A ground truth model is also provided within the dataset [55]. This set of photos includes 108 images taken from three different levels; each level has a set of 36 circumferential photos. Representative images are depicted in Figure 8. Using this set of images, 60 trials were created for analysis by manipulation of the set of input variables. The control variables were

- Quality;
- Number of photos;
- Camera angle;
- Multiview formation.

COLMAP allows for command line control of the program so that it can be written into a script and each step individually so that the user has total control over the process. It also has the ability to run an automatic re-constructor that runs the entire process from start to end using presets. The benchmarking model was run on a computer (System A) with the following specifications:

- Intel Core i7-8550U CPU @ 1.99 GHz;
- Four physical cores with eight logical processors;
- 32 Gb RAM;
- Nvidia GeForce MX130 graphics card.

Due to the limitations of the system, only the “Low” and “Medium” settings were used for the benchmark dataset. The numbers of images used for the trials were 36, 54, 72, 90, and 108. Trials were run using 18 images as well, but did not result in any successful models being constructed. The camera angle changes what position the photos are taken from. Changing this allows for photos from only one level to an assortment of photos from all three levels. This was used to generate “Low”, “Middle”, “High”, “Low and Middle”, “Low and High”, “Middle and High”, and “Low, Middle, and High” arrangements. The multiview formation is the formation the photos were used in. When using less than the maximum number of photos, it is necessary to remove some from the process. This was done using alternating photos from different levels and were labeled as “Stacked”, “Staggered”, and “Stepped”.

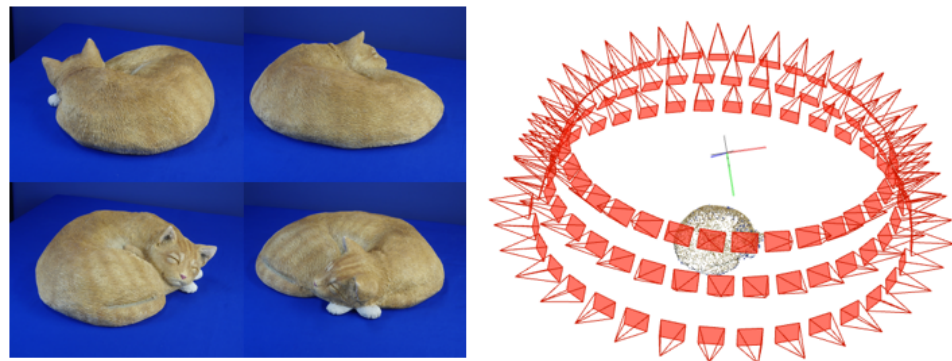


Figure 8. ToHoku University dataset [55].

By manipulating these variables, it was possible to generate 60 different trials. Once the test trials were completed, the data from each trial were recorded. The output models were compared against each other for usability and computational time. Usability requires the model have enough points to build a mesh without having too many outlying points for it to cause interference. Examples of under-meshed and over-meshed are shown in Figures 9 and 10, respectively.

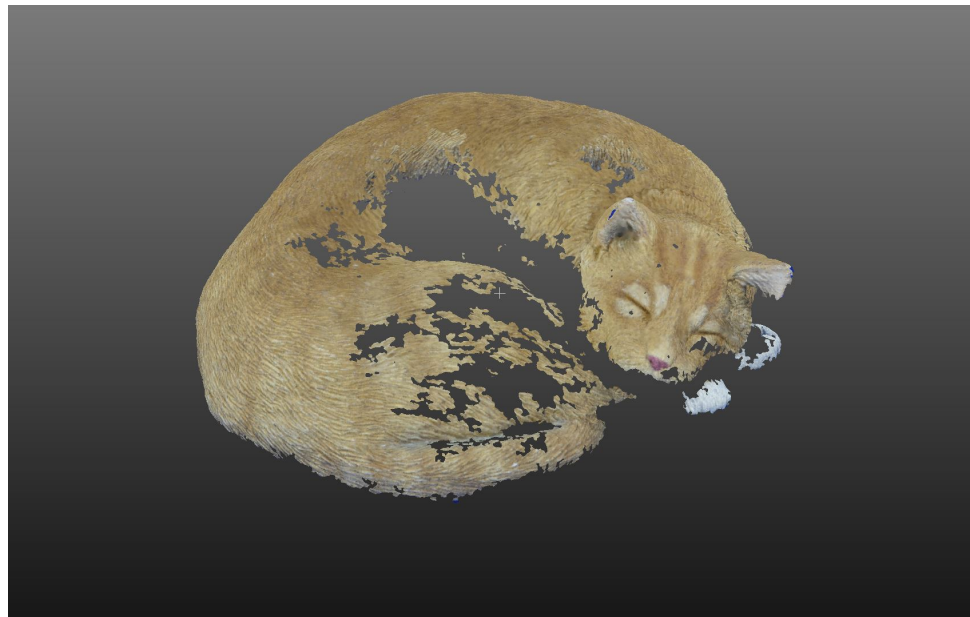


Figure 9. Trial 31 resulted in an under-meshed model which does not display all of the object.

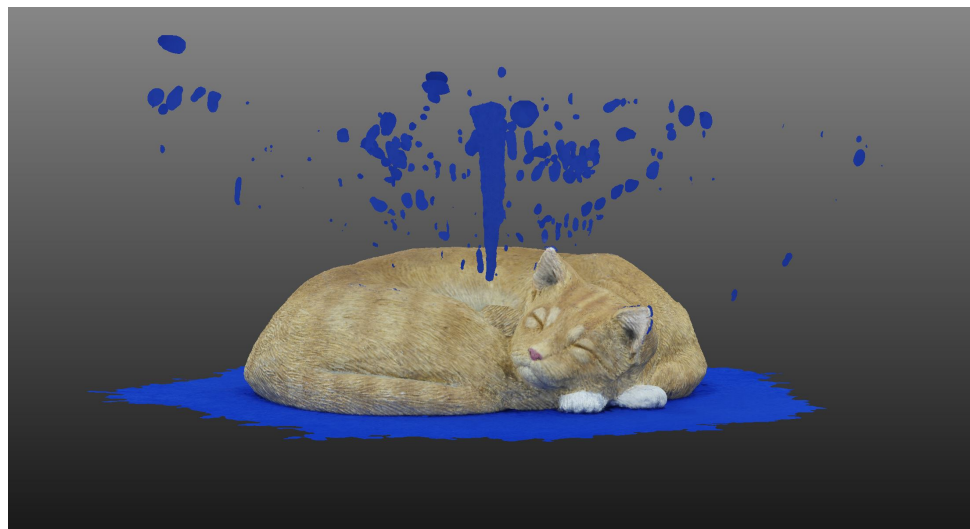


Figure 10. Trial 20 resulting in an over-meshed model with many outlying points that skewed the model and make it difficult to use.

5.1. Case 1: Results with the Benchmarking Dataset

After all of the models were run, the computational times were compared. The minimum time required to generate a model was 4.16 min, and the maximum time was 70.97 min. The trial information of most complete models is given in Table 4. The results in Figures 11 and 12 show that image count has a direct correlation with computational time. Results also revealed that with less than a certain number of photos, the model does not mesh. This was examined by trying to create a model using 18 photos and resulted in a failure to mesh. From Figure 13, it can be seen that using the “Medium” setting requires approximately the same time to compile 36 photos as the “Low” setting takes for 108. This can be effective if the limiting factor is the number of photos available. There is also a subjective solution to what makes a model the “best”. In ideal circumstances, time is not an issue, and camera placement can be effectively controlled. However, in a real world implementation, the decision on what makes a model good enough could be entirely dependent on the time available. It is also of note that using too much data has a negative effect of adding too much noise to the model (Figure 10).

Table 4. Trial information for the most complete models.

Trial #	Parameters				Time [Minutes]
	Quality	Number of Photos	Camera Angle	MultiView Formation	Total
13	Low	72	L&M	All	11.366
14	Low	72	L&M	All	11.494
17	Low	72	L&M&H	Staggered	11.957
19	Low	108	L&M&H	All	18.85
55	Low	90	L&M&H	Stack	14.084
56	Low	90	L&M&H	Staggered	14.292
25	Medium	36	L&H	Stack	18.56
26	Medium	36	L&H	Staggered	18.828
51	Medium	54	M&H	Stack	34.387
53	Medium	54	L&M&H	Stack	31.98
54	Medium	54	L&M&H	Staggered	31.595

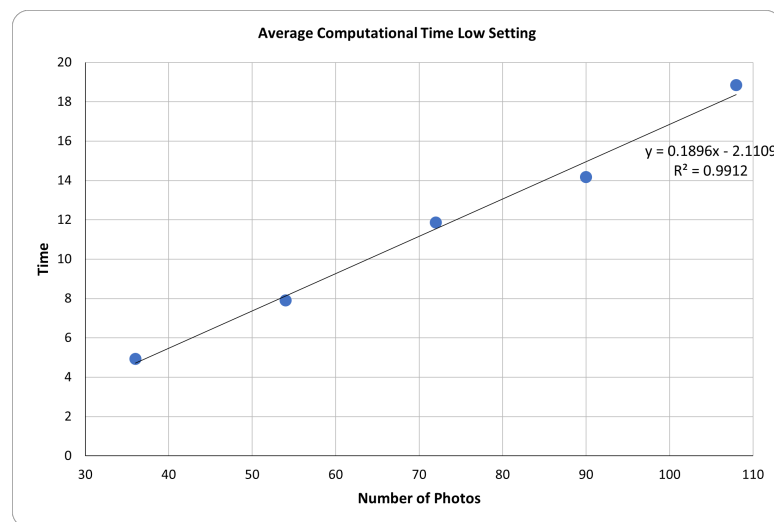


Figure 11. Average computational time in the “Low” setting.

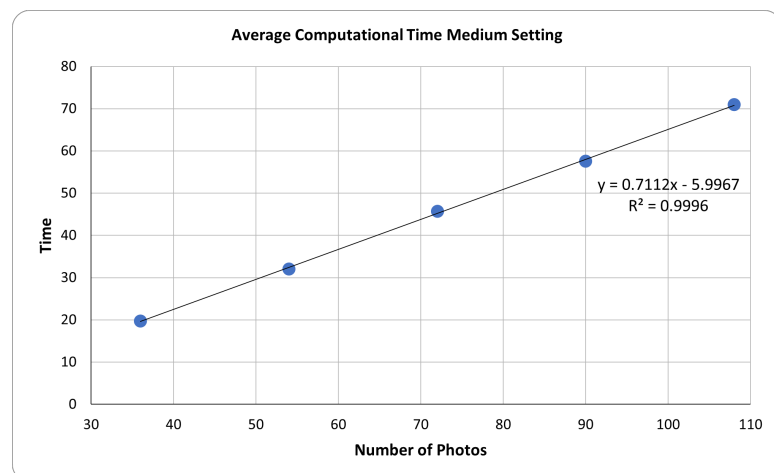


Figure 12. Average computational time for the “Medium” setting.

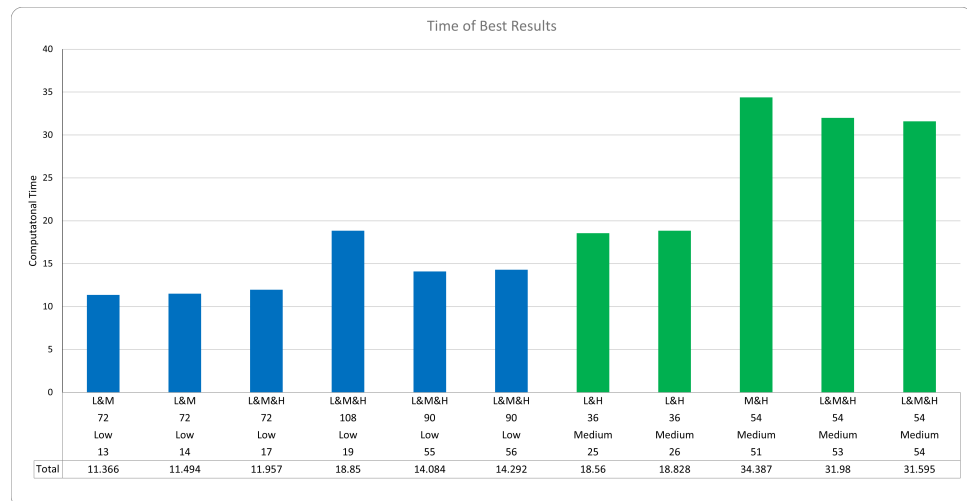


Figure 13. Time chart of the “best” models.

Once an initial assessment of all of the models was performed, they were separated into several groups for comparison. For further analysis, the “best” model was selected. This was done by collecting all of the completed models together and then visually comparing them against each other for completeness with the fewest number of outliers. Through this comparison process, trial number 25 was determined to be the best model for this further investigation. Only selecting the best model was done visually considering outliers and completeness. However, after the initial selection, the model was compared against the laser scanned ground truth data provided with the dataset. The computational time for that trial was 18.56 min. The actual ground truth data were of a model that is roughly 300 mm from end-to-end at the widest part. Trial 25 generated 1,943,933 points with 97.02% of the values within 3 mm. This investigation provided enough information for a starting point for the next dataset. It also gives a good idea of what can be done to determine optimal number of photos to be used and most effective camera orientations. Furthermore, it should be noted that the design focus was to develop a rapid framework that could be used in minutes. For more detailed models, it is possible to improve the model and add a post-processing step, or even to run at a higher processing level to refine the model as desired. However, all those will increase the computational time. For instance, the image in Figure 10 took nearly twice as long as the “best” model which was accurate within 1% and did not require post processing.

5.2. Case 2: Results with Custom Dataset

Once the benchmarking dataset was complete, a custom dataset was constructed for validation. To accomplish this, a trashcan with exactly 1-foot pieces of tape on it was used as the object to be reconstructed. The purpose of the tape on the trashcan was to create ground truth information. It was placed inside of a circle with a diameter of 10 feet and 32 equally spaced markings for camera placement. Figure 14 depicts the circle and Figure 15 depicts the placement of the trashcan.

To complete the dataset, 96 photos were taken from three different levels of 32 circumferential locations. The heights were set using a tripod and labeled *low*, *middle*, and *high* having settings of 1', 4', and 6'4", respectively. The photos were taken using a phone camera with 12MP resolution. For this dataset, a hardware improvement was made so that the trials could be run on all of the available COLMAP “Quality” settings which included: “Low”, “Medium”, “High”, and “Extreme”. As the computational power of graphics card in System A became insufficient to run custom dataset cases, a different computer (System B) was used with the following specifications:

- Intel Core i9-10850K CPU @ 3.60 GHz.
- Ten physical cores with twenty logical processors.

- 128 Gb RAM.
- Nvidia GeForce RTX3070 graphics card.



Figure 14. Circle for custom dataset construction.



Figure 15. Object placement inside the circle.

A similar process to the previous parameter analysis was used to evaluate the custom dataset. After all of the trials were completed, each model was evaluated for completeness. Having the ability to evaluate the COLMAP software with the custom dataset led to several new observations. With the “Cat” dataset, an “overmeshed” model resulted in outlier points that appeared where no object could be, as shown in the cat model (Figure 10).

For the custom dataset, an overmeshed model resulted in too much information generated far outside the area of interest; the reconstructed area ended up larger. For the custom dataset, the area of interest was the 10' diameter circle that the trashcan was situated inside. A good example of this is depicted in Figure 16.

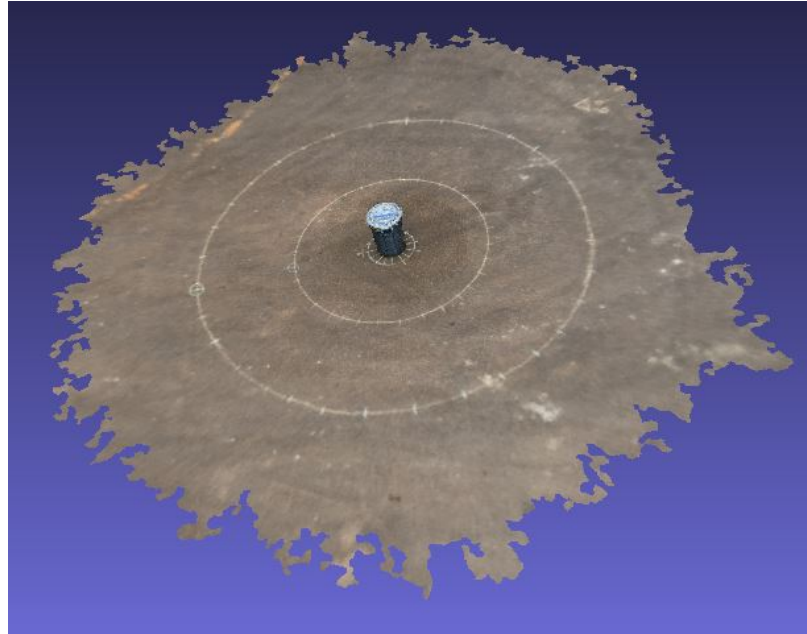


Figure 16. Desired area of interest.

Figure 17 depicts the overmeshed example where the modeled area was much too large, which resulted in the object being so overmeshed that the detail of the trashcan was diminished to just a rough cylindrical shape (Figure 18). As more images were given to the system and run at a higher resolution, it allowed the algorithm to detect and match more features. This resulted in the meshed area becoming larger. The trashcan is an object that has a monocolored, symmetrical shape with few features. This results in a poor close up mesh of the trashcan when the area is overmeshed.



Figure 17. Larger reconstruction than area of interest.



Figure 18. Close view of in larger area reconstruction.

The area that is too large shown in Figure 17 has a radius of roughly 75 ft, with the furthest objects being over 150 ft away. The observable meshed area has a direct relationship to the setting at which the software is being run. On the “Low” setting, the area is limited to immediate area regardless of how many images. This also holds true for the “Extreme” setting, where, regardless of how many images are used, the modeled area is very large. This only applies to the completed models. A list of trials that constructed a usable model are shown in Table 5.

Table 5. List of trials that created a model.

Trial #	Quality	Parameters			Time [Minutes]
		Number of Photos	Camera Angle	MultiView Formation	Total
2	Low	32	Middle	All	1.599
3	Low	32	High	All	1.615
4	Low	64	L&M	Stack	1.657
5	Low	64	L&H	Staggered	1.677
6	Low	64	M&H	Stack	3.333
7	Low	96	L&M&H	Staggered	3.418
15	Low	32	M&H	Stacked	1.567
16	Low	32	M&H	Staggered	1.588
18	Low	48	L&M&H	Staggered	1.598
28	Low	22	M&H	Stacked	0.96
29	Low	21	M&H	Staggered	0.892
30	Low	22	M&H	Stepped	0.951

Table 5. Cont.

Trial #	Quality	Parameters			Time [Minutes]
		Number of Photos	Camera Angle	MultiView Formation	Total
31	Low	32	L&M&H	Step	0.936
33	Medium	32	Middle	All	4.608
35	Medium	64	L&M	Stack	7.576
36	Medium	64	L&H	Staggered	4.901
37	Medium	64	M&H	Stack	10.12
38	Medium	96	L&M&H	Staggered	12.98
40	Medium	16	Middle	Half	1.442
42	Medium	32	L&M	Stacked	1.73
43	Medium	32	L&M	Staggered	1.55
45	Medium	32	L&H	Staggered	1.596
46	Medium	32	M&H	Stacked	4.577
47	Medium	32	M&H	Staggered	4.565
48	Medium	48	L&M&H	Stacked	4.907
49	Medium	48	L&M&H	Staggered	4.853
59	Medium	22	M&H	Stacked	2.546
61	Medium	22	M&H	Stepped	2.53
62	Medium	32	L&M&H	Step	2.364
64	High	32	Middle	All	37.063
66	High	64	L&M	Stack	75.239
68	High	64	M&H	Stack	80.84
73	High	32	L&M	Stacked	25.277
74	High	32	L&M	Staggered	25.848
77	High	32	M&H	Stacked	36.53
78	High	32	M&H	Staggered	36.441
79	High	48	L&M&H	Stacked	52.196
80	High	48	L&M&H	Staggered	52.004
90	High	22	M&H	Stacked	18.923
91	High	21	M&H	Staggered	19.619
92	High	22	M&H	Stepped	18.839
96	Extreme	32	High	All	75.788
97	Extreme	64	L&M	Stack	155.417
99	Extreme	64	M&H	Stack	183.944
100	Extreme	96	L&M&H	Staggered	245.359
107	Extreme	32	L&H	Staggered	35.039
108	Extreme	32	M&H	Stacked	11.153
109	Extreme	32	M&H	Staggered	74.884
110	Extreme	48	L&M&H	Stacked	108.258
111	Extreme	48	L&M&H	Staggered	111.694
124	Extreme	32	L&M&H	Step	54.397

A comparison of the datasets revealed the most complete models were generated when the system was set to the highest possible settings. However, using all of the images in the dataset resulted in non-optimal models that were over-meshed with many outliers, as shown in Figures 10, 17, and 18. Better models were generated when two levels of images were used, and all of the images on those levels were used. At lower settings, good models can be generated but require more images. For instance, 76 images were required to generate an “excellent” model on the “Low” setting, but only 36 images were required to generate a comparable model for the “Medium” setting for the “Cat” dataset. The best model for the custom dataset is shown in Figure 19.

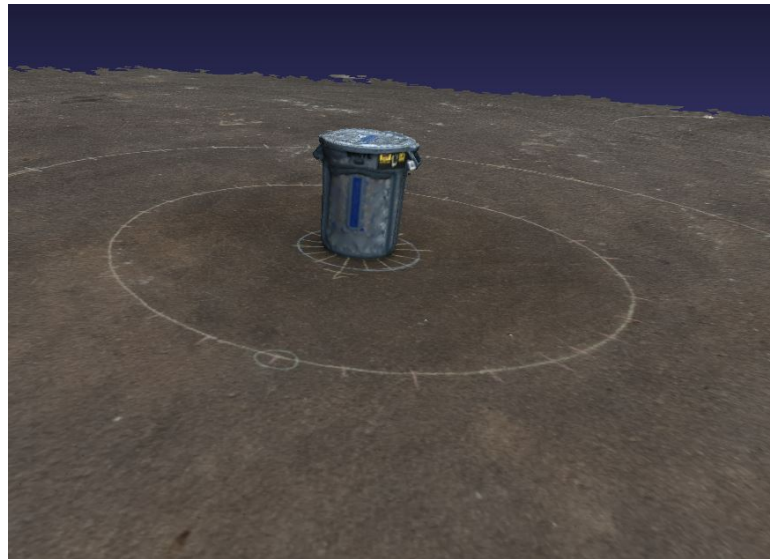


Figure 19. Best model from the custom dataset.

6. Validation Model Results

Using the components from the distributed behavior model and 3D reconstruction, the complete framework was established. A representation of the pipeline of the developed framework is depicted in Figure 20. A confirmation implementation was designed to replicate a real world application. Given that the most robust models were constructed using images from two levels, two sUASs were used in the team. The two sUASs took off and move into the grouping phase. Once the entropy of the system fell below the threshold value of 0.1, the team moved into the mission phase and started its movement to the object of interest. When the team got closer to the object, they shifted into an orbit pattern, with one sUAS moving into a higher position and the other moving into a lower position, and started their orbit pattern. As the orbit began, so does the recording. A typical orbit took approximately 2 min and 10 s, and approximately 500 images were recorded.

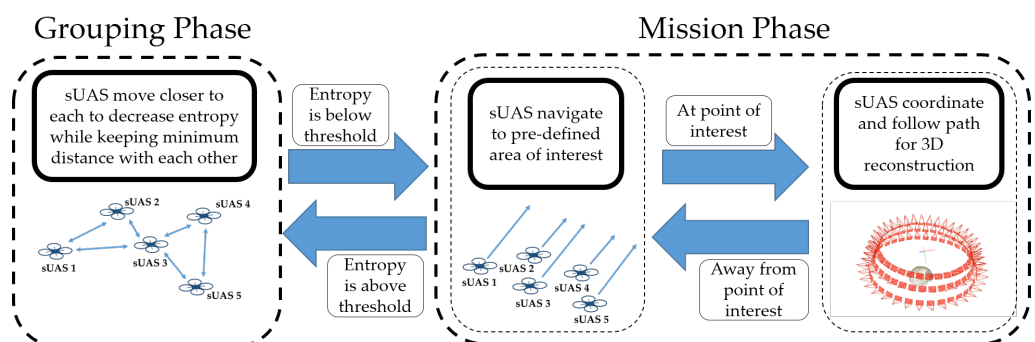


Figure 20. Complete pipeline of the developed framework.

AirSim provides the capability to take images from the perspective of the sUAS in the simulation environment. These are the images that were used in the reconstruction of the 3D model. The resolution of these images can be specified in the settings, and for the validation model experiments, they were set to 3840×2160 pixels. Images from the perspective of the sUAS are depicted in Figure 21.

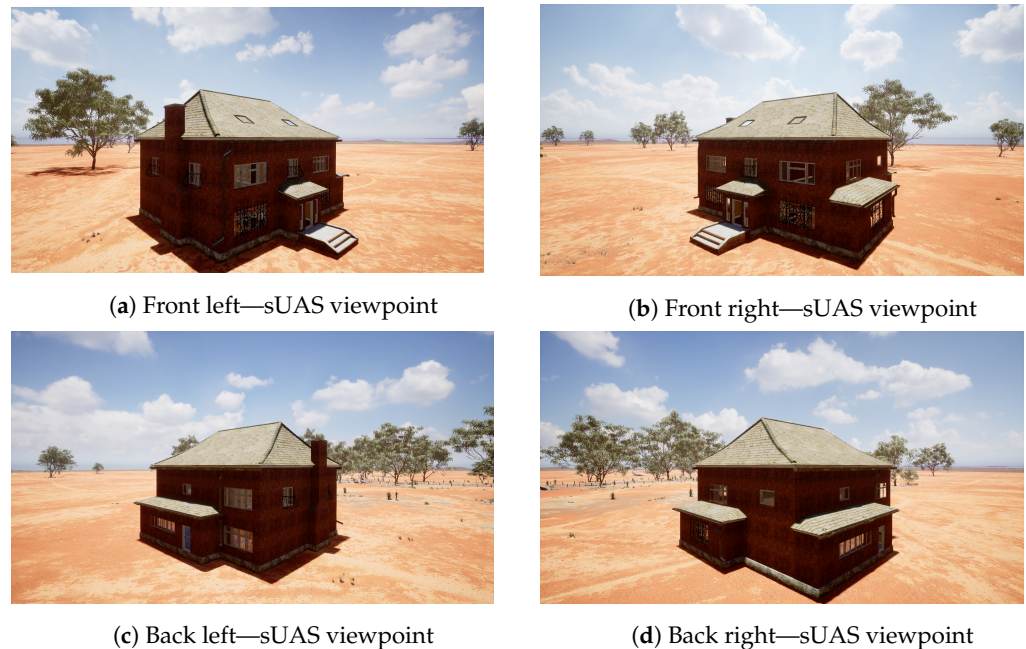


Figure 21. Images taken from the sUAS in AirSim.

The 3D model of the object in interest, a house in these validation experiments, was constructed using 71 images from the dataset (Figure 22). Figure 22a depicts that the area around the house was limited to a reasonable area without the interference of multiple outliers. By limiting the area to a small radius, it is possible to generate better models and reduce the amount of time required to complete it. The reconstructed model has enough details; the ones that can be noticed are entry and exit locations, the chimney, and windows. Upon close inspection, the gutters can also be observed. It is clear enough that if a person, car, or animals were in the example, they could also be recorded. This provides confidence that the developed framework can perform well in real world applications. As presented in 3D reconstruction parameter analysis section, in the virtual world, it is more difficult to detect features because everything is perfectly simulated. In the real world, on the other hand, there are more features to detect, as presented in Section 5.2.

The reconstruction of the model took 6 min 4 s from when the team completed their orbits. Looking at the total time spent, depending on where the sUASs were spawned, it could take up to 7 min for them to get through the grouping phase and then make their way to the objective. From there, they made an orbit, which took 2 min and 10 s. Modern off-the-shelf sUASs are capable of much faster physical operations that could accomplish this movement portion of the mission much faster. The tradeoff would be the time it takes to build the model, and it takes a little more than 6 min with the settings used in the simulations.

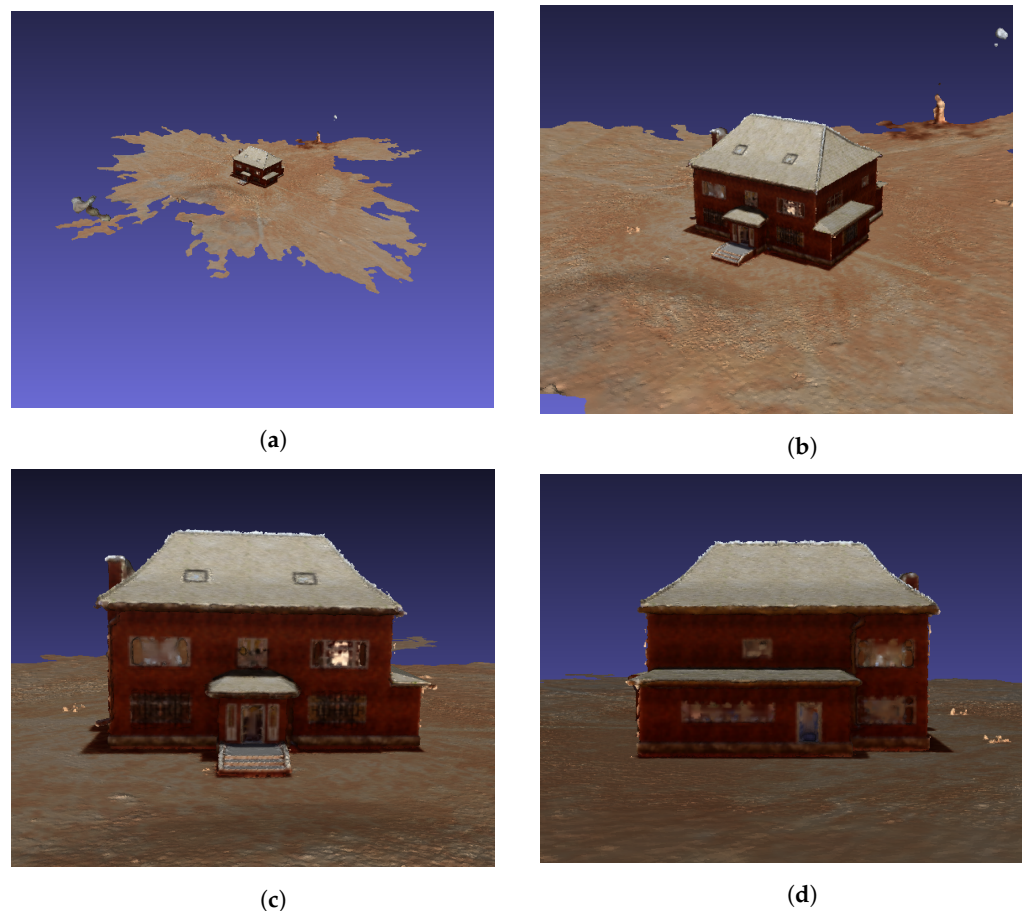


Figure 22. Images of the 3D reconstructed model. (a) 3D reconstructed model—far view; (b) 3D reconstructed model—closer view; (c) 3D reconstructed model—front view; (d) 3D reconstructed model—vack view.

7. Discussions and Future Directions

In this paper, the aim was to present proof-of-concept results of the developed framework for rapid 3D reconstruction using distributed model-based multi-agent sUASs. There are several points that need to be addressed; those are worth mentioning here. In terms of distributed behavior model, the parameters should be tuned very carefully; otherwise, there might be cases of instability. One possibility is that sUASs may fail to group, and there are two ways that can happen; either the entropy threshold value is set too low, or the entropy is too high and the sUASs bypass the grouping phase all together. Another point that should be considered is that sUASs may have “edge characteristics”. That resulted in the entropy value being very close to the threshold value but not below the threshold, so one or more of the sUASs had an erratic path that resembled a wave pattern. This possible challenge was mostly solved by introducing sectors, described in Section 3.2.

In terms of 3D reconstruction, it should be noted that the sUASs in the simulation environment each had one fixed camera. This makes it difficult to focus the cameras in the correct orientation so that it can look down at the object. When investigating the best camera poses, the best cases with just one angle were all done from the “Middle” or “High” positions where more of the object was in view in each image. In the cases where only perpendicular images were used, developed framework could not build a complete model. This limitation could be experienced in the simulation models; the “High” position also resulted in a much larger model radius. This is because when a sUAS is at the top of the roof of the house model, it needs to be in a position where the roof is in the bottom of the frame. When in this position, the rest of the image has a view out to the horizon. This results in poor models with huge model radii and many artifacts. Using a sUAS with a

camera on a gimble would allow for maximizing the area in the field of view with the object of interest.

One immediate future work for this study will be to apply the framework using actual sUAS platforms in real world. Although parameter analysis in the real world with a fixed camera was presented, it is necessary to validate the results by using sUAS platforms. The goal of using fixed camera experiments is to eliminate all factors and have a controlled experiment to focus on proof-of-concept results of the developed framework. However, it is important to also test effects of real-world issues, such as vibration, coupled movement of the sUAS and camera, and motion blur, on the presented framework's performance. That is why experiments with actual sUASs are planned for immediate future work of this study.

Additionally, the intent is to have all computations done by the onboard hardware of the sUAS. The simulation times recorded in the experiments were for a machine that is substantially better than the companion computers that could be placed on sUASs. However, given that comparative models can be established using lower resolution with more images, it is not likely that this would be much of an issue. Using twice or three times the images would not be a difficult adjustment to make given that most available off-the-shelf sUASs have high-resolution cameras mounted on them.

Another reason for plan future work with an actual platform is to see what kind of images are able to be recorded in different weather conditions. The simulation environment images were taken in a nearly motionless environment without wind, rain, platform vibration, glare, and several other uncertainties. Incorporating the framework onto a physical platform would undoubtedly bring up many challenges, some of which would likely be unforeseen. Integration onto hardware would also give a better estimate of mission completion time as well. The simulated sUASs are not designed to be high performance; as a result, the navigation time to the object in interest is the longest part of the process. This would not be the case when using an actual sUAS platform, and the mission completion time could be even shorter with relatively faster platforms.

8. Conclusions

In this study, a rapid 3D reconstruction framework was presented using structure from motion with images obtained from a team of sUAS. Details of the distributed behavior model for a team of sUASs and 3D reconstruction steps were provided. For the distributed behavior model that is based on the entropy of the system, a parameter analysis was conducted with the intent of having a robust and scalable algorithm capable of navigating the sUASs to the desired positions. Parameters used for evaluation were (i) minimum distance between sUASs and (ii) entropy threshold. Simulations were run with various numbers of sUASs to confirm scalability, robustness, and threshold values. A minimum distance setting was confirmed, and a function for scaling the threshold value was determined. For the 3D reconstruction step, COLMAP was evaluated with two separate datasets for optimization with the intent of reducing the computational time required to build a model that is usable. One dataset was a controlled environment for initial analysis, and the other dataset was of a real world environment for applicability to hardware implementation. The priorities are usability of the model and speed of reconstruction. During this evaluation, four parameters were analyzed in software for evaluation: quality, image number, camera angle, and multiview formation. As a validation experiment, a team of two sUASs was able to use the entropy-based distributed behavior model to take off from different locations and then group together in the simulation environment. Then, they made their way towards a predefined object and recorded images of the object while in orbit. Once the orbit was complete, the total of 71 images, which were separated by approximately 10 degrees and from two levels, were used to reconstruct the model, and a rapid model could be constructed in low resolution in as little as 6 min 4 s from when the team completed their orbit motion. This would be helpful in emergency situations, such as wildfires, disaster relief efforts, and search and rescue missions. In those situations, an in-depth, highly accurate model is

not necessarily required, but a rapid solution could be crucial. The planned future work included transitioning the framework onto real hardware systems.

Author Contributions: Conceptualization, methodology, investigation, writing—original draft preparation, D.S.S.J.; supervision, project administration, writing—review and editing, H.E.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Schonberger, J.L.; Frahm, J.M. Structure-from-motion revisited. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4104–4113.
- Balch, T.; Arkin, R.C. Behavior-based formation control for multirobot teams. *IEEE Trans. Robot. Autom.* **1998**, *14*, 926–939. [[CrossRef](#)]
- Lawton, J.R.; Beard, R.W.; Young, B.J. A decentralized approach to formation maneuvers. *IEEE Trans. Robot. Autom.* **2003**, *19*, 933–941. [[CrossRef](#)]
- Monteiro, S.; Bicho, E. Attractor dynamics approach to formation control: Theory and application. *Auton. Robot.* **2010**, *29*, 331–355. [[CrossRef](#)]
- Xu, D.; Zhang, X.; Zhu, Z.; Chen, C.; Yang, P. Behavior-based formation control of swarm robots. *Math. Probl. Eng.* **2014**, *2014*, 205759. [[CrossRef](#)]
- Olfati-Saber, R. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Trans. Autom. Control.* **2006**, *51*, 401–420. [[CrossRef](#)]
- Vásárhelyi, G.; Virágh, C.; Somorjai, G.; Nepusz, T.; Eiben, A.E.; Vicsek, T. Optimized flocking of autonomous drones in confined environments. *Sci. Robot.* **2018**, *3*, eaat3536. [[CrossRef](#)]
- Monteiro, S.; Bicho, E. A dynamical systems approach to behavior-based formation control. In Proceedings of the 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292), Washington DC, USA, 11–15 May 2002; IEEE: Piscataway, NJ, USA, 2002; Volume 3, pp. 2606–2611.
- Fredslund, J.; Mataric, M.J. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Trans. Robot. Autom.* **2002**, *18*, 837–846. [[CrossRef](#)]
- Zhang, J.; Yan, J.; Zhang, P. Multi-UAV Formation Control Based on a Novel Back-Stepping Approach. *IEEE Trans. Veh. Technol.* **2020**, *69*, 2437–2448. [[CrossRef](#)]
- Liu, W.; Gao, Z. A distributed flocking control strategy for UAV groups. *Comput. Commun.* **2020**, *153*, 95–101. [[CrossRef](#)]
- Lee, G.; Chwa, D. Decentralized behavior-based formation control of multiple robots considering obstacle avoidance. *Intell. Serv. Robot.* **2018**, *11*, 127–138. [[CrossRef](#)]
- Qu, X.; Wan, Y.; Zhou, P.; Li, L. Consensus-Based Formation of Second-Order Multi-Agent Systems via Linear-Transformation-Based Partial Stability Approach. *IEEE Access* **2019**, *7*, 165420–165427. [[CrossRef](#)]
- Cofta, P.; Ledziński, D.; Śmigiel, S.; Gackowska, M. Cross-Entropy as a Metric for the Robustness of Drone Swarms. *Entropy* **2020**, *22*, 597. [[CrossRef](#)]
- Albani, D.; Manoni, T.; Arik, A.; Nardi, D.; Trianni, V. Field coverage for weed mapping: Toward experiments with a UAV swarm. In Proceedings of the International Conference on Bio-Inspired Information and Communication, Pittsburgh, PA, USA, 13–14 March 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 132–146.
- Schranz, M.; Umlauf, M.; Sende, M.; Elmenreich, W. Swarm Robotic Behaviors and Current Applications. *Front. Robot.* **2020**, *7*, 36. [[CrossRef](#)]
- Arnold, R.D.; Yamaguchi, H.; Tanaka, T. Search and rescue with autonomous flying robots through behavior-based cooperative intelligence. *J. Int. Humanit. Action* **2018**, *3*, 18. [[CrossRef](#)]
- Yuheng, Z.; Liyan, Z.; Chunpeng, L. 3-d deployment optimization of uavs based on particle swarm algorithm. In Proceedings of the 2019 IEEE 19th International Conference on Communication Technology (ICCT), Xi'an, China, 16–19 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 954–957.
- Cao, H.; Zhang, H.; Liu, Z.; Zhou, Y.; Wang, Y. UAV path planning based on improved particle swarm algorithm. In Proceedings of the 2021 7th International Symposium on Mechatronics and Industrial Informatics (ISMII), Zhuhai, China, 22–24 January 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 284–287.
- Deng, Q.; Yu, J.; Wang, N. Cooperative task assignment of multiple heterogeneous unmanned aerial vehicles using a modified genetic algorithm with multi-type genes. *Chin. J. Aeronaut.* **2013**, *26*, 1238–1250. [[CrossRef](#)]

21. Wildermuth, D.; Schneider, F.E. Maintaining a common co-ordinate system for a group of robots based on vision. *Robot. Auton. Syst.* **2003**, *44*, 209–217. [[CrossRef](#)]
22. Vail, D.; Veloso, M. Multi-robot dynamic role assignment and coordination through shared potential fields. *Multi-Robot. Syst.* **2003**, *2*, 87–98.
23. Lakas, A.; Belkacem, A.N.; Al Hassani, S. An Adaptive Multi-clustered Scheme for Autonomous UAV Swarms. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2021; IEEE: Piscataway, NJ, USA, 2020; p. 1567.
24. Barnes, L.; Garcia, R.; Fields, M.; Valavanis, K. Swarm formation control utilizing ground and aerial unmanned systems. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008; IEEE: Piscataway, NJ, USA, 2008; pp. 4205–4205.
25. Bayram, Ç.; Sevil, H.E.; Özdemir, S. A distributed behavioral model for landmine detection robots. In Proceedings of the International MultiConference of Engineers and Computer Scientists 2007, IMECS 2007, Hong Kong, China, 21–23 March 2007; International Association of Engineers: Hong Kong, China, 2007.
26. MacKenzie, D.C. Collaborative tasking of tightly constrained multi-robot missions. In Proceedings of the Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings of the 2003 International Workshop on Multi-Robot Systems 2003, Washington, DC, USA, 17–19 March 2003; Volume 2, pp. 39–50.
27. Li, R.; Ma, H. Research on UAV Swarm Cooperative Reconnaissance and Combat Technology. In Proceedings of the 2020 3rd International Conference on Unmanned Systems (ICUS), Harbin, China, 27–28 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 996–999.
28. Zhang, Y.; Zhang, Z.; Zhang, J.; Wu, J. 3D building modelling with digital map, lidar data and video image sequences. *Photogramm. Rec.* **2005**, *20*, 285–302. [[CrossRef](#)]
29. Abayowa, B.O.; Yilmaz, A.; Hardie, R.C. Automatic registration of optical aerial imagery to a LiDAR point cloud for generation of city models. *ISPRS J. Photogramm. Remote Sens.* **2015**, *106*, 68–81. [[CrossRef](#)]
30. Yang, B.; Chen, C. Automatic registration of UAV-borne sequent images and LiDAR data. *ISPRS J. Photogramm. Remote Sens.* **2015**, *101*, 262–274. [[CrossRef](#)]
31. Beger, R.; Gedrange, C.; Hecht, R.; Neubert, M. Data fusion of extremely high resolution aerial imagery and LiDAR data for automated railroad centre line reconstruction. *ISPRS J. Photogramm. Remote Sens.* **2011**, *66*, S40–S51. [[CrossRef](#)]
32. Sohn, G.; Dowman, I. Data fusion of high-resolution satellite imagery and LiDAR data for automatic building extraction. *ISPRS J. Photogramm. Remote Sens.* **2007**, *62*, 43–63. [[CrossRef](#)]
33. Zhao, G.; Xiao, X.; Yuan, J.; Ng, G.W. Fusion of 3D-LIDAR and camera data for scene parsing. *J. Vis. Commun. Image Represent.* **2014**, *25*, 165–183. [[CrossRef](#)]
34. Sinsley, G.; Long, L.; Geiger, B.; Horn, J.; Niessner, A. Fusion of unmanned aerial vehicle range and vision sensors using fuzzy logic and particles. In Proceedings of the AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference, Seattle, WA, USA, 6–9 April 2009; p. 2008.
35. Korpela, I.; Dahlin, B.; Schäfer, H.; Bruun, E.; Haapaniemi, F.; Honkasalo, J.; Ilvesniemi, S.; Kuutti, V.; Linkosalmi, M.; Mustonen, J.; et al. Single-tree forest inventory using lidar and aerial images for 3D treetop positioning, species recognition, height and crown width estimation. In Proceedings of the ISPRS Workshop on Laser Scanning, Espoo, Finland, 12–14 September 2007; pp. 227–233.
36. Song, H.r.; Choi, W.s.; Lim, S.m.; Kim, H.d. Target localization using RGB-D camera and LiDAR sensor fusion for relative navigation. In Proceedings of the Automatic Control Conference (CACs), 2014 CACS International, Kaohsiung, Taiwan, 26–28 November 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 144–149.
37. Carlevaris-Bianco, N.; Mohan, A.; McBride, J.R.; Eustice, R.M. Visual localization in fused image and laser range data. In Proceedings of the Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, San Francisco, CA, USA, 25–30 September 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 4378–4385.
38. Schönberger, J.L. Robust Methods for Accurate and Efficient 3D Modeling from Unstructured Imagery. Ph.D. Thesis, ETH Zurich, Zurich, Switzerland, 2018.
39. Nikolov, I.; Madsen, C. Benchmarking close-range structure from motion 3D reconstruction software under varying capturing conditions. In Proceedings of the Euro-Mediterranean Conference, Nicosia, Cyprus, 31 October–5 November 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 15–26.
40. Bianco, S.; Ciocca, G.; Marelli, D. Evaluating the performance of structure from motion pipelines. *J. Imaging* **2018**, *4*, 98. [[CrossRef](#)]
41. Stathopoulou, E.K.; Remondino, F. Open-source image-based 3D reconstruction pipelines: Review, comparison and evaluation. In Proceedings of the 6th International Workshop LowCost 3D–Sensors, Algorithms, Applications, Strasbourg, France, 2–3 December 2019; pp. 331–338.
42. Maxence, R.; Uchiyama, H.; Kawasaki, H.; Thomas, D.; Nozick, V.; Saito, H. Mobile photometric stereo with keypoint-based SLAM for dense 3D reconstruction. In Proceedings of the 2019 International Conference on 3D Vision (3DV), Quebec City, Canada, 16–19 September 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 574–582.
43. Mentasti, S.; Pedersini, F. Controlling the flight of a drone and its camera for 3D reconstruction of large objects. *Sensors* **2019**, *19*, 2333. [[CrossRef](#)]

44. Lundberg, C.L.; Sevil, H.E.; Das, A. A VisualSfM based Rapid 3-D Modeling Framework using Swarm of UAVs. In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 22–29.
45. Daftry, S.; Hoppe, C.; Bischof, H. Building with drones: Accurate 3D facade reconstruction using MAVs. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 3487–3494.
46. Gao, K.; Aliakbarpour, H.; Fraser, J.; Nouduri, K.; Bunyak, F.; Massaro, R.; Seetharaman, G.; Palaniappan, K. Local Feature Performance Evaluation for Structure-From-Motion and Multi-View Stereo Using Simulated City-Scale Aerial Imagery. *IEEE Sens. J.* **2020**, *21*, 11615–11627. [[CrossRef](#)]
47. Shah, S.; Dey, D.; Lovett, C.; Kapoor, A. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 621–635.
48. Tsallis, C. Possible generalization of Boltzmann-Gibbs statistics. *J. Stat. Phys.* **1988**, *52*, 479–487. [[CrossRef](#)]
49. Bayram, C.; Sevil, H.E.; Ozdemir, S. Swarm and entropic modeling for landmine detection robots. In *Trends in Intelligent Systems and Computer Engineering*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 105–116.
50. Das, A.N.; Doelling, K.; Lundberg, C.; Sevil, H.E.; Lewis, F. A Mixed reality based hybrid swarm control architecture for manned-unmanned teaming (MUM-T). In Proceedings of the ASME International Mechanical Engineering Congress and Exposition. American Society of Mechanical Engineers, Tampa, FL, USA, 3–9 November 2017; Volume 58493, p. V014T07A019.
51. Das, A.; Kol, P.; Lundberg, C.; Doelling, K.; Sevil, H.E.; Lewis, F. A rapid situational awareness development framework for heterogeneous manned-unmanned teams. In Proceedings of the NAECON 2018-IEEE National Aerospace and Electronics Conference, Dayton, OH, USA, 23–26 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 417–424.
52. Sevil, H.E. Anomaly Detection using Parity Space Approach in Team of UAVs with Entropy based Distributed Behavior. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020; p. 1625.
53. Szeliski, R. *Computer Vision: Algorithms and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2010.
54. Fischler, M.A.; Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* **1981**, *24*, 381–395. [[CrossRef](#)]
55. Mönnig, J. ToHoku University Multi-View Stereo (THU-MVS) Datasets. Available online: <http://www.aoki.ecei.tohoku.ac.jp/mvs/> (accessed on 8 October 2021).
56. Smith, D.S., Jr. A Rapid Structure from Motion (SfM) Based 3-D Modeling Framework Using a Team of Autonomous Small Unmanned Aerial Systems (sUAS). Master's Thesis, The University of West Florida, Pensacola, FL, USA, 2021.