Simulated Gold-Standard for Quantitative Evaluation of Monocular Vision Algorithms

Jack Akers[a], Andrew Buck[a], Raub Camaioni[b], Derek T. Anderson[a], Robert H. Luke III[b], James M. Keller[a], Matthew Deardorff[b], and Brendan Alvey[a]

[a]Department of Electrical Engineering and Computer Science, University of Missouri, Columbia MO, USA
[b]U.S. Army DEVCOM C5ISR Center, Fort Belvoir, VA, USA

ABSTRACT

In the physical universe, truth for computer vision (CV) is impractical if not impossible to obtain. As a result, the CV community has resorted to qualitative practices and sub-optimal quantitative measures. This is problematic because it limits our ability to train, evaluate, and ultimately understand algorithms such as single image depth estimation (SIDE) and structure from motion (SfM). How good are these algorithms, individually and relatively, and where do they break? Herein, we discuss that while truth evades both the real and simulated (SIM) universes, a SIM CV gold-standard can be achieved. We outline an extensible SIM framework and data collection workflow using Unreal Engine with the Robot Operating System (ROS) for three dimensional mapping on low altitude aerial vehicles. Furthermore, voxel-based mapping measures from algorithm output to a SIM gold-standard are discussed. The proposed metrics are demonstrated by analyzing performance across changes in platform context. Ultimately, the current article is a step towards an improved process for comparing algorithms, evaluating their strengths and weaknesses, and automating algorithm design.

Keywords: point cloud, voxel, structure from motion, monocular depth estimation, single image depth estimation, unreal engine, simulation, ground-truth, gold-standard, XAI, evaluation

1. INTRODUCTION

The field of 3D reconstruction (3DR) is decades old with countless groundbreaking innovations from academic and commercial parties. 3DR enables many applications like self driving cars, unmanned aerial vehicles (UAV), film, augmented reality, earth observation, security and defense, agriculture, robotics, computer vision (CV), archaeology, and beyond. However, the physical universe is a 3DR bottleneck. Accurate and dense truth at scale is impractical if not impossible to obtain. How do we perform objective, quantitative analysis for tasks like fine-tuning, training, and/or evaluating 3DR algorithms? Where does an algorithm work? Where does it fail? Is one algorithm better than another, by how much and in what context? A lack of real world truth significantly hinders our ability to address these fundamental and essential questions.

Herein, we use simulation (SIM) to create datasets in a controlled environment with dense and highly accurate ground-truth. Different metrics are discussed for quantitative evaluation of a 3DR algorithm relative to truth in a static scene. Figure 1 illustrates how current article contributions fit into our longer-term research goals. These operations enable a human-in/on/over-the-loop (HITL/HOTL) to iteratively explore and fine-tune 3DR algorithms in different platform (e.g., drone) contexts. These operations are also needed for fully automated data-driven tasks like closed-loop optimization.

The remainder of the article is structured as follows. In Section 2, related work is summarized. Next, in Section 3 we discuss our SIM framework and workflow, Section 4 outlines truth, and Section 5 presents different voxel-based evaluation metrics. Last, Section 6 has two examples for a real-time 3DR algorithm that show the proposed ideas in action. The reader can refer to Table 1 for notation and acronyms.

Send correspondence to Jack Akers
E-mail: jdapm8@missouri.edu

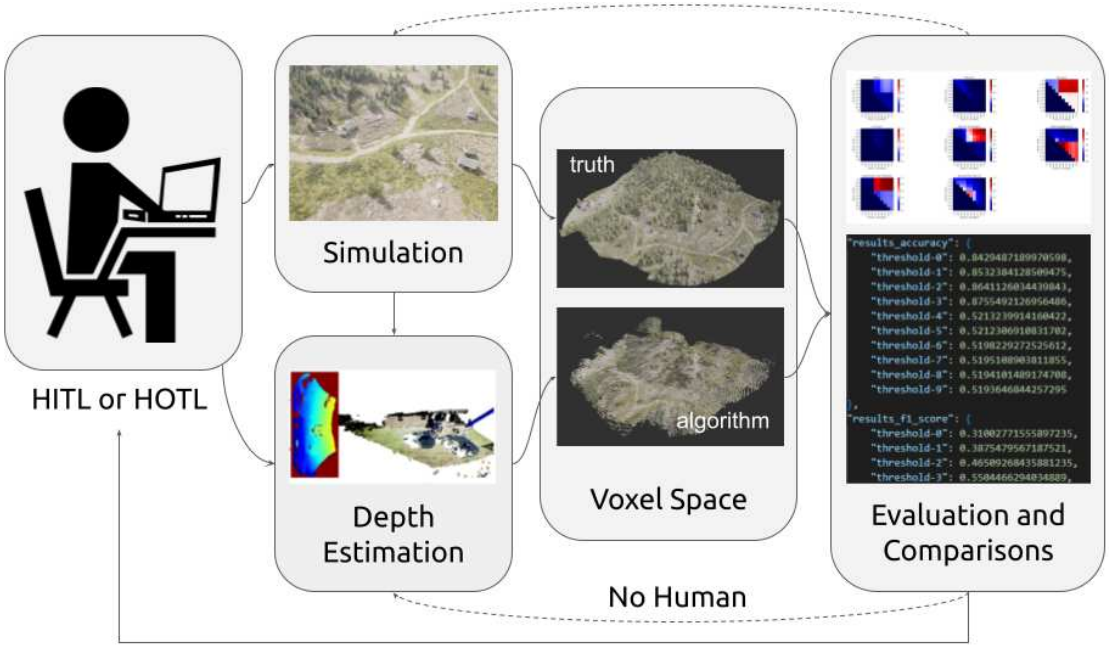


Figure 1: Iterative process for depth estimation algorithm and unmanned aerial vehicle flight context design and evaluation. See Section 1 for more details.

Table 1: Notations and Acronyms

Acronym	Description	Notation
3DR	3D Reconstruction	
SfM	Structure from Motion	
MVS	Multi View Stereo	
SIM	Simulation	
PC	Point Cloud	
Truth	True state of nature	
Gold-standard	Not the truth, best that can be achieved	
SIDE	Single Image Depth Estimation	
	Image at time t, pixel (x, y)	$I_t(x, y) \in \{0, 255\}$
	Depth at time t, pixel (x, y)	$D_t(x, y) \in \Re^+$
VS	Voxel Set	$V(i, j, k) \in [0, 1]$
	Free voxel threshold	$\lambda_F \in [0, 1]$
	Occupied voxel threshold	$\lambda_O \in [0, 1]$
$\tilde{V}_O, \tilde{V}_F, \tilde{V}_U$	Estimated Occupied, Free, and Unknown VS	$\tilde{V}_O(i, j, k), \tilde{V}_F(i, j, k), \tilde{V}_U(i, j, k) \in \{0, 1\}$
V_O, V_F, V_U	Gold-Standard Occupied, Free, and Unknown VS	$V_O(i, j, k), V_F(i, j, k), V_U(i, j, k) \in \{0, 1\}$

2. RELATED WORK

2.1 Monocular Depth Estimation

Many small UAVs are equipped with a monocular image sensor and a GPS/IMU/magnetometer module that provide a color image and camera extrinsics. This is sufficient for techniques to estimate 3D depth. While the current article focuses on a 3D mapping algorithm called EpiDepth,[1] related state-of-the-art methods include multi-view stereo (MVS) and structure from motion (SfM),[2] simultaneous localization and mapping (SLAM),[3–5] etc.), deep learning (DL) for single image depth estimation (SIDE)[6–8]), and DL-based optical flow.[9]

The EpiDepth algorithm[1] takes a stream of world-aligned image frames as input and produces (for acceptable image pairs) a dense, per-pixel depth estimation. Figure 2 shows example EpiDepth image pairs on SIM data collected using AirSim and Unreal Engine (UE). EpiDepth can run in real-time on embedded hardware with

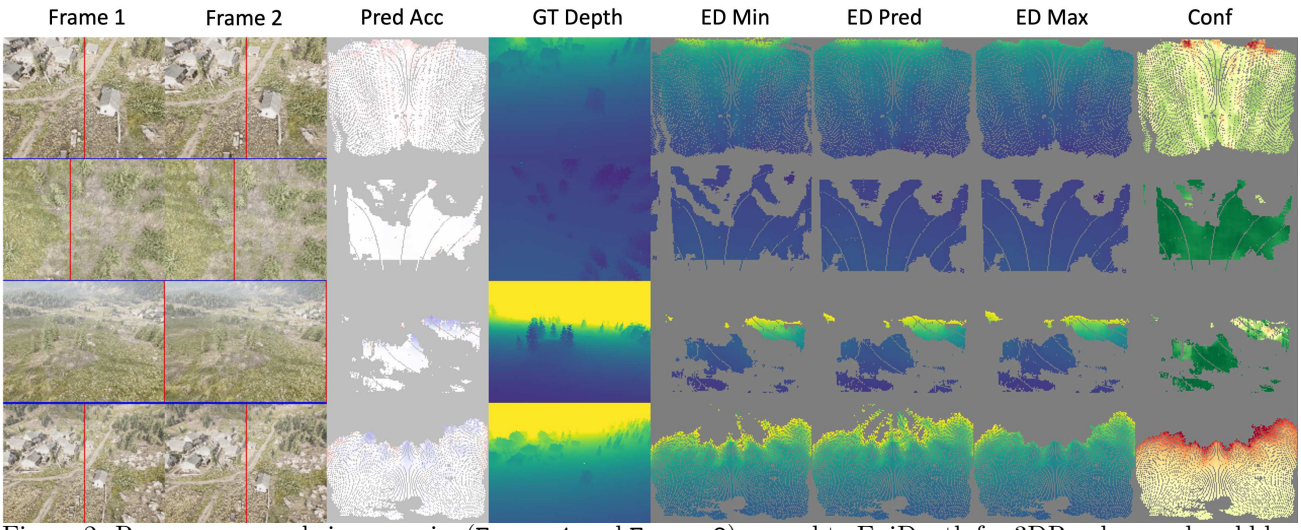


Figure 2: Rows are example image pairs (`Frame 1` and `Frame 2`) passed to EpiDepth for 3DR, where red and blue lines indicate the location of the epipole. `Pred Acc` is a seismic color coded error map, i.e., ground-truth depth minus EpiDepth predicted depth. `GT Depth` is a viridis color coded truth image, and `GT Min`, `GT Pred`, and `GT Max` are EpiDepth's minimum, average, and maximum possible depth assignments per-pixel. Last, `Conf` is a per-pixel confidence that we use to commit depths to voxel space; see our prior article on capturing uncertainty in monocular depth estimation for further details.[10]

parameters to adjust the scale and quality. Frames are selected sequentially based solely on their relative camera extrinsics. In order to be considered, a pair of frames must have a temporal and spatial displacement within a defined range, and also be oriented in the same general direction. From a valid frame pair, the color images are warped and aligned so that optimized block matching techniques can be applied to produce disparity and depth images. The camera-space pixels are then projected into a 3D point cloud (PC) and mapped into world coordinates from the known camera extrinsics. In this study, we do not perform any refinement of the camera position (i.e. using SLAM methods), and rely on the accuracy of the UAV position and orientation sensors to place the points into a common world-space coordinate frame. For the sake of this article, Epidepth generates a depth map, $D_t(x, y) \in \Re^+$, for a selected image pair. While any units could be used, e.g., cm, feet, meters, in whatever space, e.g., UTM, depth maps in this article are in meters in UEs global coordinate system.

2.2 World Space Representation

Next, a world *model* can be addressed in many ways. For example, the majority of 3D mapping algorithms yield unstructured PCs in a local camera coordinate system. Methods like SfM and SLAM combine observations across looks (images or image pairs) to produce a relative, or when external position information is present, global 3D PC. Frequently, such data is quantized into a more efficient data structure like an octree. In areas like robotics, many simplify the world further into an occupancy grid map (OGM). In an OGM, each discrete unit typically has a number (e.g., probability) that is ultimately used to determine its state, e.g., unknown, free, or occupied (UFO). In recent years, more advanced probabilistic structures such as OctoMap[11] and UFOMap[12] have been proposed for ground robotics and drones. In Ref. 13, O'Meadhra et al. introduced a compressed, generative, and efficient variable resolution structure based on probabilistic OGM via Gaussian mixture models for autonomous cave surveying[14] and reactive collision avoidance.[15] Last, for sake of completness, another common practice is to convert PCs into a mesh by the use of methods like the ball-pivoting algorithm (BPA) or Poisson reconstruction.

In this article, an Epidepth depth map at time step t is provided to UFOMap to produce a probabilistic voxel set $V(i, j, k) \in [0, 1]$. Given user specified parameters λ_F and λ_O, $V(i, j, k)$ can be partitioned into binary sets $\tilde{V}_O, \tilde{V}_F, \tilde{V}_U$, corresponding to occupied, free, and unknown. Figure 3 shows a UFO map.

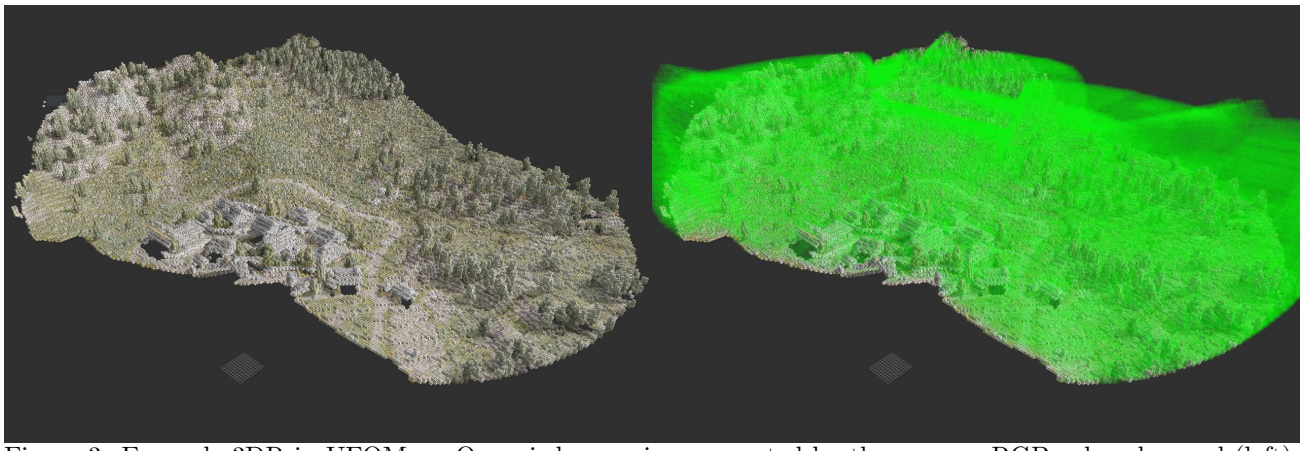


Figure 3: Example 3DR in UFOMap. Occupied space is represented by the average RGB color observed (left), while free space may also be (optionally) visualized as a green translucent fog (right). Note that while not displayed, the remaing voxels are considered to be explicitly unknown space.

2.3 Simulation Environment

A classical SIM environment, driven largely by robotics, is Gazebo.[16] While Gazebo is not state-of-the-art with respect to photorealism, it has advantages such as extensive integration with the robotic operating system (ROS)[17] and it supports a wide range of physics and sensors (e.g., RGB, RGBD, GPS, IMU, LiDAR, sonar, etc.). With respect to improved realism, Apple's HYPERSIM, Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding,[18] does not support physics but it can produce nearly indistinguishable (to a human) photorealistic imagery with information about scene geometry, material information, lighting, per-pixel instance segmentations with respect to diffuse reflectance, diffuse illumination, and non-diffuse residual term that captures view-dependent lighting effects. Another current approach is NVIDIA's Isaac Gym,[19] an end-to-end simulation platform, built from the ground up to run large-scale, physically-accurate multi-sensor simulation for applications like autonomous vehicles and virtual worlds. Epic Games' Unreal Engine (UE) is another solution which has an extremely generous licence that varies based on terms of use;[20] ranging from free to percentage of profits. The UE (see Figure 4) started as a game engine, but it has recently settled into new applications like architecture, film, computer graphics, and beyond. A somewhat new trend is extensions to these core tools, like the AirSim[21] plugin for UE4; which allows for external python support and simulation of different low altitude drones. Last, a more recent competitor to UE is Unity.[22]

Figure 4: Example urban and rural images showing UE5 photorealism.

The above examples are from robotics, computer graphics, gaming, and the hobbyist communities. SIM dates back decades and its scope is beyond the current article. SIM frequently involves approaches from finite element modeling, wave simulation, ray casting, and beyond to accurately model topics like electromagnetics

and physics. Examples include COMSOL for multiphysics SIM and MEEP, a free and open-source software package for electromagnetics simulation via the finite-difference time-domain (FDTD). These SIMs are often based on simulating sensors such as the visual spectrum (RGB) to multispectral, hyperspectral, radar, LiDAR, acoustics, and beyond. A recent physics-based example from the literature is the virtual autonomous navigation environment (VANE) and ANVEL platforms.[23] The reader can also refer to the internet for more expensive and custom business solutions, such as Scale AI. While the number of SIM possibilities is endless, these tools are only a part of the big SIM picture. Most of these SIM solutions are provided "as is". That is, they have some level of trust associated with them, typically based on their design, but most take shortcuts or are discrete approximations to continuous processes and only a few have undergone a more rigorous process of verification and validation (V&V). The reader can refer our article[24] on a recent historical review of the development of verification and validation theories for simulation models.

The ideas expressed in this article can be executed on a variety of *platforms*. To date, we have a single implementation that runs on Windows, Ubuntu, and a Jetson for real-time UAV processing and autonomy. Additionally, while we focus on UE, another SIM environment such as Unity could be used with minimal effort. The reader can refer to our article on "How Should Simulated Data Be Collected for AI/ML and Unmanned Aerial Vehicles" for full details about how we produce RGB, depth, object IDs, and other data layers.[25]

3. SIM ENVIRONMENT AND WORKFLOW

This section documents our process for generating and evaluating a 3DR relative to a SIM gold-standard. First, datasets are collected through AirSim and UE. These collections are carefully controlled with all movements pre-scripted and deterministic to ensure repeatability and accuracy. The collect process generates several assets, including precise camera extrinsics, RGB image data, and a gold-standard per-pixel depth. Figure 5 illustrates our pipeline relative to a single flight (data collection).

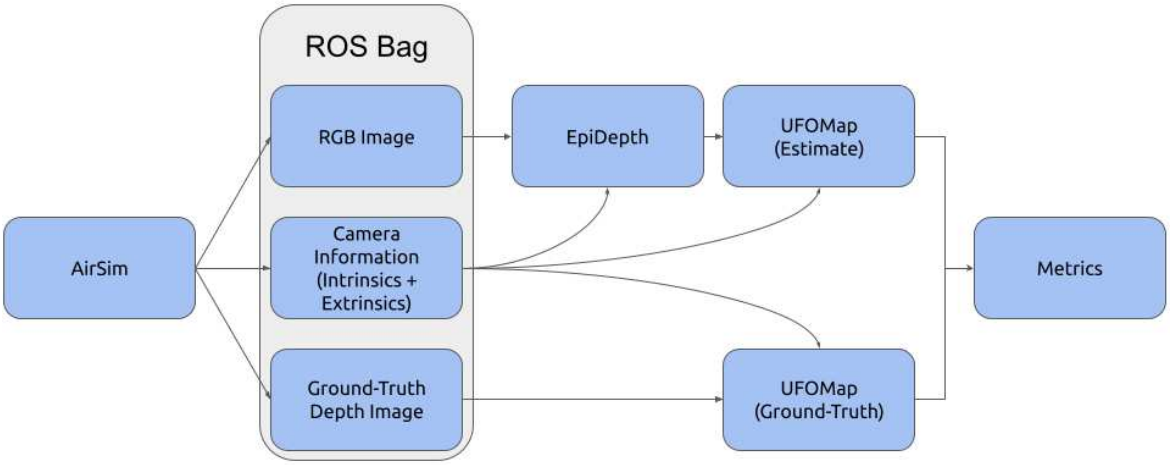


Figure 5: High-level illustration of the flow of data during the course of a single 3D reconstruction and evaluation. First, AirSim is used to fly a drone around and obtain data in UE, resulting in bagged imagery with truth. Next, Epidepth is run offline on the ROS bags to produce an aggregate UFOMap. This 3DR is then subjected to evaluation relative to a ground-truth UFOMap.

Our pipeline uses ROS as a backbone for inter-process communication and modular compatibility. More specifically, we are using ROS Noetic Ninjemys, released on May 23rd, 2020 for Ubuntu 20.04 (Focal). While limited support for the Windows operating system is available, we found that using an Ubuntu 20.04-based Docker container for computing metrics was necessary for ensuring compatibility with the popular "catkin_tools" CLI required to build the UFOMap node.

The result of our first step, data and truth generation, is a ROS .bag file. Our codebase is flexible and it has been run in real-time on an embedded device (Jetson) using real drone imagery and metadata. However, in our

SIM workflow, and when it comes to our real-time platform, there are experimental advantages to producing ROS bags versus online processing. First, due to a multitude of reasons (code, random numbers, computer processing and ROS queues, etc.), subtle differences could arise from run-to-run and lead to slightly different data. If a goal is to explore 3DR algorithm parameters, this adds an uncontrolled confounding factor to our experiment. Instead, if data is collected and stored once to a ROS .bag, then offline we can ensure that algorithms downstream receive exactly the same data. While subtle, this is an important detail.

Next, we replay the contents of a stored .bag file containing the collected SIM data. Note that, since all of the input data is already in a standard ROS format at this point, the entire pipeline is fully compatible with any SIM platform, not just our current choice of AirSim, so long as it is capable of producing the required information. This process is relatively straightforward using the "rosbag" command-line tool included with ROS. However, since downstream nodes may use ROS' TF2 transform system to look up camera extrinsics at an arbitrary point at time, a critical part of this phase is that the Header types contained within several of the messages need to be modified so that their timestamps reflect the time of replay rather than the time. We accomplish this using our own "ros_replay" tool written in Python that interacts directly with ROS' included "rosbag" library. While we found it best to entirely avoid using the TF2 library wherever possible for this application, the "ros_replay" tool is still incredibly useful for manual control and logging of message timing and metadata.

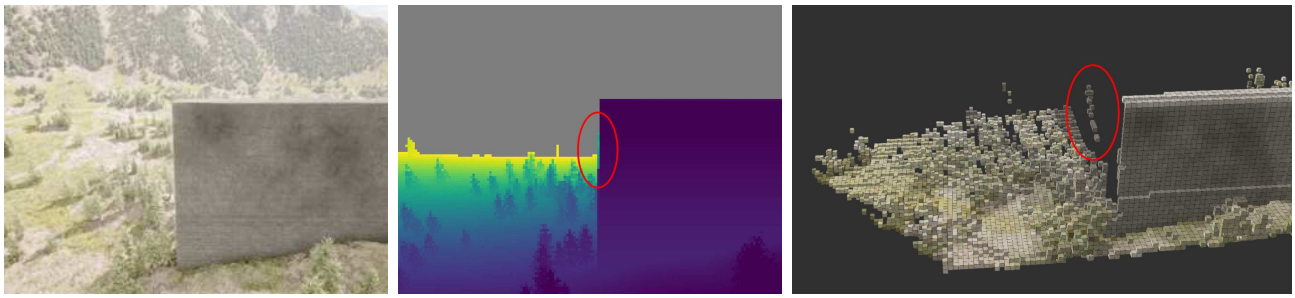
The next step is to run the 3DR algorithm. In this paper, we explore EpiDepth. Our implementation contains native support for ROS and performs all of its I/O through it. The output of this process is a 2D estimated depth map and RGB PC, calculated with respect to the recorded platform extrinsics. Since EpiDepth is processing a fixed ROS bag, its parameters can be varied to better understand our algorithm.

The final processing phase is our custom fork of UFOMap. It uses the output PCs along with camera pose information to form an aggregate probabilistic model of the world in voxel space. We run two instances of UFOMap concurrently: one for aggregating the estimated output, and another for aggregating the gold-standard output. Our custom fork introduces several new ROS services, including the ability to export the map as a CSV file that can be easily processed externally in Python. Thus, we can evaluate the final state of the 3D representation directly using voxel-space metrics.

4. GOLD-STANDARD VS GROUND-TRUTH

In the physical universe, the true state of nature for many phenomena is fundamentally not known, or at least not known to a certain degree with respect to limitations in observation. Thus, we instead use the term *gold-standard* to denote the "benchmark that is the best available under reasonable conditions".[26] This is the case for CV. What is an image? What is a pixel? A pixel (picture element) is a discrete unit of space and time denoting a sensor's response to an incoming collection of information that traveled to the lens, was recorded by the film back, and eventually converted into an electrical signal. A pixel is a "bundle" of data, not a single sample. Thus, questions like "what object is at" or "what is the depth of" a pixel are fundamentally ill-posed questions. A sensor, e.g., RGB camera, only records a single value per pixel. SIM only records a single value as well. It is inefficient for SIM to capture and record all incident information, and most researchers will never make use of such data regardless. However, in its absence, there is no real truth. Instead, we are subject to living with a gold-standard. While this sounds sad, it is simply reality. The good news is, this gold-standard is orders of magnitude more accurate and dense than data collected in the real world.

The aim of the current section is to simply raise awareness of this conundrum, as it cuts at the heart of how to define evaluation metrics relative to SIM. Specifically, not all SIM gold-standard data can be taken with full legitimacy. The reader can refer to our article on "Ignorance is bliss: flawed assumptions in simulated ground-truth"[27] for further details on why, when, and where this occurs, and ultimately methods for remediation. Figure 6 is a common example that demonstrates depth artifacts that arise from mixed pixels on the edge of objects. In our prior article,[27] we discuss the bias and error associated with computing a single truth aggregate per pixel. This frequently occurs in the context of multi-sample (spatial and/or temporal) anti-aliasing when generating an image and its associated truth. Furthermore, we discuss problems with the common misconception that it is possible to simply fix this problem by forcing a single truth, e.g., assign a single object ID or depth in the case of UEs movie render queue. This action is not correct and it results in uni-modal truth that effects AI and/or



(a) (b) (c)

Figure 6: Example illustrating a scenario that can arise in the generation of SIM truth. Consider the wall in 6a. If a single depth is stored per pixel, then object edges are mixed (part wall, part terrain), what should SIM store? In 6b and 6c, AirSim computed an anti-aliased value of the foreground (wall) and background (terrain). The result is floating depth observations that are wrong, as they are neither wall nor terrain. Instances like this are why we refer to the SIM data as a "gold-standard" rather than ground-truth.

our evaluation metrics. Instead, in[27] we argued for a distribution-based approximation to the truth. Similar flaws can occur in SIM contexts like billboard-based foliage, translucent materials such as glass or water, and other areas where SIM takes shortcuts for the sake of efficiency. In summary, SIM limitations exist and impact our ability to obtain 100% accurate truth. The question is how do we minimize such effects, and how should we design metrics with such knowledge in mind?

5. EVALUATION MEASURES

In this section, we explore a few measures to compare 3DR algorithm outputs to a gold-standard acquired via SIM[*]. Herein, we focus on multi-frame voxel set measures[†]. This was decided because our applications only require voxel level resolution, not a PC or mesh. However, there are consequences associated with this decision. First, a voxel space has lost all connection to what images/pixels a given voxel's data is derived from. This level of information would be impractical to store, and its not clear exactly how one would store it (e.g., in UFOMap) or even if it's worth storing. As a result, UFO space can only be compared at an overall aggregate level, not at an individual correspondence resolved observation level. Second, we have followed UFOMap's procedure for converting depth maps and 3D point clouds into a voxel space. Specifically, individual observations are converted to voxel space ray queries, versus view frustums, and probabilistic updates to voxel space are made. By design, the algorithm voxel space is what it is. However, what is the gold-standard voxel space? Herein, we take the gold-standard depth, per frame, and use it to build a UFOMap. The rationale is as follows. If the 3DR algorithm was perfect, these are the exact depths. If we were to propagate those values through UFOMap, the result would be a corresponding gold-standard. Thus, we are comparing what we did get to what we should have obtained. This is different from a procedure that takes the geometry of a SIM scene and directly converts it into UFO voxel sets. The reason why we did not pursue that route is because those sets would contain elements that were not observed during simulated flight and thus also not observable by our algorithm. This complicates scoring. Thus, our outlined procedure is the fairest approach we could imagine. In the remainder of this section, our measures can largely be divided into two categories, set-based (Algorithms 1 and 2) and surface-2-surface based (Algorithm 3).

[*]The reader can refer to the CV literature for more common real-world quantitative measures. Individual frame measures for complex scenes are often based on a sparse, secondary, and often questionable (error-bound) sensors like LiDAR. Typical multi-frame measures include feature track statistics, e.g., how many features, track strengths, track lengths, etc. There even exists preliminary SIM work for synthetic feature track analysis.[28] The literature also possesses many PC-to-PC, PC-to-surface, surface-to-PC, and related measures.

[†]In future work, we plan to address per-frame measures. An advantage of per-frame is pixel correspondence. That is, each estimated pixel depth/location can be compared to its exact corresponding depth/location.

Algorithm 1 : p =Prec(V, $\tilde{V}$) $\in [0,1]$, a =Acc(V, $\tilde{V}$) $\in [0,1]$, r =Recall(V, $\tilde{V}$) $\in [0,1]$, F1 =F1(V, $\tilde{V}$) $\in [0,1]$

INPUT: $V = \{V_O, V_F, V_U\}$, $\tilde{V}$, $\lambda_F \in [0,1]$, $\lambda_O \in [0,1]$

1. Set $\tilde{V}_F(i,j,k) = 1$, $\forall (i,j,k)$ s.t. $\tilde{V}(i,j,k) < \lambda_F$, 0 otherwise ▷ Make binary sets
2. Set $\tilde{V}_O(i,j,k) = 1$, $\forall (i,j,k)$ s.t. $\tilde{V}(i,j,k) > \lambda_O$, 0 otherwise ▷ Make binary sets
3. Calculate confusion matrix M with respect to $\{\tilde{V}_O, \tilde{V}_F\}$ and $\{V_O, V_F\}$ ▷ Conf(Truth, Algorithm)
4. $p = \frac{M(1,1)}{M(1,1)+M(2,1)}$ ▷ Precision
5. $a = \frac{M(1,1)+M(2,2)}{M(1,1)+M(1,2)+M(2,1)+M(2,2)}$ ▷ Accuracy
6. $r = \frac{M(1,1)}{M(1,1)+M(1,2)}$ ▷ Recall
7. $\text{F1} = 2\frac{p*r}{p+r}$ ▷ F1-score

RETURN: $p, a, r, \text{F1}$

In Algorithm 1, the gold-standard and 3DR algorithm UFOMaps are first partitioned into their corresponding free, occupied, and unknown binary sets. From there, a number of standard crisp metrics can be computed, e.g., precision, accuracy, recall, and F1-score. Thus, these measures are based on the logic, "how similar is a 3DR algorithm estimate to the gold-standard on a per voxel basis?" A problem is, this metric is harsh and unforgiving. There is a level of acceptable error, based on the size of a voxel. However, if a 3D mapped point is even a fraction of a percentage outside of the answer voxel, it is counted against the algorithm. It is well-known that error in our depth estimation algorithm increases with range and with respect to factors like baseline. The point being, this class of metric does not exhibit graceful degradation with respect to this nature of inaccuracy.

Algorithm 1 can be carried out a number of ways. We use a confusion matrix based on combining occupied and free space. However, Algorithm 1 can clearly be computed on any combination of crisp sets, e.g., free-to-free, occupied-to-occupied, free and occupied to free and occupied, and etc. It is up to the user to decide what their question is. Occupied-to-occupied informs us about how well two methods agree on mapping the visible regions on observed surfaces. On the other hand, free-to-free informs the reader about how much free space the algorithm should have seen versus what it did. This can be useful in contexts like drone/robot exploration. Note that metrics on unknown space are meaningless, as the default state of the map is an infinite grid of unknown space, and applying a bounding box for calculations in the same manner as free/occupied space would be arbitrary. Thus, only free and occupied space are evaluated.

Algorithm 2 : Raw counts and statistics of VS A in B

INPUT: $A = \{A_O, A_F, A_U\}$, $B = \{B_O, B_F, B_U\}$, distance threshold ϵ

1. l_{AB} is the number of locations in A such that $A_O(i,j,k) == 1$
2. $\hat{l}_{AB}$ is the number of locations in A that have a distance less than ϵ to B
3. $c_{AB} = \left(\frac{\hat{l}_{AB}}{l_{AB}} \in [0,1]\right)$ ▷ Percentage of points in A matched to B with respect to ϵ

RETURN: $\hat{l}_{AB}$, c_{AB} ▷ Raw counts and percentage of matched points

Algorithm 2, is a simpler and more tolerant variant of Algorithm 1. Let the 3DR algorithm voxel set be A and the gold-standard be B. As already stated, the gold-standard (B) may contain questionable observations (it is not actual truth). Furthermore, it is common practice to filter a PC and/or voxel set. We have not considered any of these factors in Algorithm 1. Algorithm 2 begins by counting the total number of occupied points in A[‡]. Next, the distance of each occupied voxel in A is calculated to B. Voxels with distance greater than user defined value ϵ are removed. Thus, only points that have a reasonable match to B are considered. Next, a ratio of points that match to B over all possible mapped points is computed. This $[0,1]$ ratio informs the user how well A maps to B, relative to tolerance ϵ. A high value indicates that the algorithm is very accurate in its mapping. However, the challenge with Algorithm 2 is, consider an algorithm that only assigns a few points. Let that algorithm be extremely good in its mapping. This ratio can be high, but overall only a small portion of the scene is mapped. One way to identify this scenario is to run Algorithm 2 with the truth as A and algorithm as B. Thus, the reader

[‡]The reader can clearly swap out occupied points for free points, but semantics need be considered.

needs to probably consider measures in both direction to understand accuracy and completeness of mapping. It should be noted, this algorithm is slightly improved, in tolerance, over Algorithm 1. However, it is still geared to return a relation in $[0, 1]$, versus helping us understand something like, "what is the average error between our algorithm and truth?"

Algorithm 3 : Surface-2-Surface MSE of VS A to B, $m_{AB} = \text{Surf2Surf}(A, B) \in [0, \infty]$

INPUT: $A = \{A_O, A_F, A_U\}$, $B = \{B_O, B_F, B_U\}$, distance threshold ϵ
1. Initialize $G(i, j, k) = 1, \forall i, j, k$ such that $B_O(i, j, k) == 0$, $G(i, j, k) = 0$ otherwise.
2. $D = \text{DistTransform}(G)$. ▷ Distance to occupied cells (set of distances)
3. Let I be the set of all indices in A such that $A_O(i, j, k) == 1$. ▷ All occupied cells in other set
4. Let Z be the set of all D distances for the set of indices I. ▷ Corresponding distances
5. Remove all values in Z greater than ϵ. ▷ Remove extreme distances and error points in gold-standard
5. Set m_{AB} to the average of Z. ▷ Final MSE value

RETURN: m_{AB}

The last measure considered herein is Algorithm 3. This algorithm is based on surface-2-surface comparison between occupied sets. Since we are considering 3DR CV applications, e.g., a drone mapping an environment, it is assumed that occupied locations are visible surfaces on objects, free space is empty visible volumes, and unknown are locations not seen and solid non-visible volumes. Algorithm 3 begins by computing the distance transform from set A to B (two sets of mapped surfaces). As outlined in the last paragraph, A and B can be our algorithms voxel space and truth, or vice versa. Next, extreme differences (user threshold ϵ) are removed and the average error is computed on the remainder. Unlike Algorithms 1 and 2, this measure considers extreme non-matches, it more gracefully degrades, and it returns an average error vs an overlap amount. However, this error is clearly subject to the underlying voxel resolution. It cannot resolve tolerances less than that resolution and the result is clearly still impacted by the discrete grid that the voxel space resides on.

Before we conclude this section, we discuss the elephant in the room. Which measure is best? This is complicated. Each measure captures a different aspect of our result. Furthermore, say the application is exploring a large area and exploration is the goal. The user might be concerned with the amount of free space observed, its rate of expansion, and accuracy of observed locations might be relatively unimportant. On the other hand, another application might not care about clearing a lot of space. Instead, the user might desire to know how much occupied space was mapped and to what error level. The point is, different applications and users will likely require using different combination logic with respect to these measures. Last, its worth mentioning that these are not mutually exclusive measures. For example, a user might run Algorithm 3 first to get a feel for what the average voxel mapping error is in some unit, e.g., meters. Next, the reader might run Algorithm 2 from algorithm-to-truth and truth-to-algorithm to better understand what percentage of the truth was observed, at what error rate, and how many locations in the algorithms voxel space did not reasonably map to truth. The point is, our initial paper explores different measures and how to connect these calculations to high level user desires. In future work, we will go the next step and explore attributing a task and connecting what measures and combination logic are required to rank order experiments.

6. DATASETS AND EXAMPLES

The ideas outlined in this paper can be used for a number of feats. For example, our workflows can be used to understand and pick an operating context, e.g., relative camera pose, flight altitude, motion, etc. Our process can also be used to iteratively explore and fine-tune a 3DR algorithm. Our process can even be used to compare 3DR algorithms to each other instead of an algorithm to the truth. There is simply too much ground to cover in this initial article. As a result, we focus on a single use case from our research. We assume EpiDepth is operating ideally, thus parameters have already been selected, and we try to use the outlined measures to understand and identify an ideal platform operating context in a specified environment.

Figure 7 shows our UE environment. We used the Mountain Village Environment[29] from the UE Marketplace.[30] This environment was selected for a number of reasons, specifically: 3D object, material, and visual

Figure 7: Example large 2x2 km map with man made structures (houses, roads, etc.) and nature (hills, grass, trees, etc.) in Mountain Village Environment[29] from the UE Marketplace.[30]

quality; cost (79.99 USD); use of man made structures (buildings); nature (trees, rocks, etc.); environment size (2x2 km); and bounded horizon (enclosed by close mountains). Overall, this scene is nice because it allows us to collect data over a region with buildings with common edge and unique features. On the other hand, another portion of this scene can be used that has just rocks, grass, trees, and other nature with perhaps harder to match and map features. The point being, this scene is a nice balanced 3DM sandbox.

In Table 2, we discuss our eight datasets. This data can be described according to the following underlying variables: region, altitude, and look angle. Two regions were explored: *region one*, which has man made structures (roads, buildings, etc.), and *region two*, which is just nature (rocks, grass, trees). Two altitudes were explored, "low" and "high". For this article, this level of granularity is sufficient. We are only interested comparing these two contexts. In our article on training and evaluating passive ranging in SIM,[31] we explored distance as a continuous variable and semantic IDs were used to subdivide performance relative to specific object categories. Thus, we could ask questions like, "what is performance for vehicles as a function of range?" The last variable we consider herein is look angle, which is either nadir (straight down) or slant angle (45°).

Table 2: Dataset Summary

DS1	region 1 (man made), height high, nadir look angle
DS2	region 1 (man made), height high, 45 degree look angle
DS3	region 1 (man made), height low, nadir look angle
DS4	region 1 (man made), height low, 45 degree look angle
DS5	region 2 (nature), height high, nadir look angle
DS6	region 2 (nature), height high, 45 degree look angle
DS7	region 2 (nature), height low, nadir look angle
DS8	region 2 (nature), height low, 45 degree look angle

All together, this results in $2 \times 2 \times 2 = 8$ datasets. The simplest task we could perform is to analyze performance with respect to a single dataset. However, when combined, these datasets help us address more complex questions like the following. A specific question would be Q1="how high should we fly if we are looking nadir in a region with man made structures?" Mid level question is Q2="do we do better in a structured man made region versus nature" or Q3="do we do better flying high or low?" Of course, the highest, thus most generic question we could ask is, Q4="what situation do we do best in?" Clearly, each of these questions require us to compare across different datasets. Q1 requires us to compare performance on DS1 to DS3. Q2 requires comparing {DS1, DS2, DS3, DS4} to {DS5, DS6, DS7, DS8}, and Q4 is analysis over all datasets.

Figure 8 shows the proposed measures applied to DS1. The x-axis in these figures shows variation in the occupied threshold and y-axis free threshold variation. Also, the matrix diagonal and upper diagonal are the only values reported. The lower diagonal values do not correspond to a legitimate scenario, as the free threshold needs to be less than the occupied threshold. These plots are interesting to look at on a number of levels. First, consider `Recall`. The reader can see that there is a performance drop off when free is greater than 0.43 and when occupied is less than 0.51. This makes sense, as 0.5 models unknown. `Recall`'s highest value is when occupied is highest, 0.9, and free is lowest, 0.2. This make sense from the stance of `Recall`, as it represents probably the simplest points to map and therefore the algorithm does best in that context. However, if you look at the `F1 Score`, a combination of `Recall` and `Precision`, you see a different story. When multiple objectives are combined, the best overall performance is a set of thresholds slightly around 0.5. The point is, the user can use these plots to decide, at a quantitative vs qualitative level, what thresholds to select. Furthermore, the user can see what performance, and associated sensitivity with respect to thresholds, can be achieved. Additionally, the `F1 Score` for DS1 is low. Why? It is hard to gather more from these plots. It is also equally hard to look at a single `F1 Score` and know how good that value is. This is the point where a user needs to look into the data at a more qualitative level to help understand if that is perhaps acceptable. Definitely, these values are much easier to use on a comparative basis, e.g., is DS1 better than DS2?

Figure 8 also shows surface-to-surface and intersection ratio results. The reader can see that these are different criteria and there is no obvious winning combination of thresholds across measures. Based on the `F1 Score`, `MSE`, and `Intersection Over` results, a threshold of 0.51 for occupied and 0.43 is likely a good tradeoff, and it leads to to an `Intersection Over Estimate` overlap score of 0.6, `Intersection Over GT` of 0.3, `MSE GT to EpiDepth` of 0.8, and `MSE EpiDepth to GT` of 0.2. This says, "60% of EpiDepth's points match the gold-standard truth, 30% of the truth locations were discovered by EpiDepth, and EpiDepth's error (relative to a 1m cube voxel size) is approximately 0.2m."

Figure 9 is a qualitative comparison of EpiDepth on DS1 relative to the gold-standard. This figure gives the reader a feel for what the EpiDepth reconstruction looks like relative to the numeric measure data just presented. Quickly, the reader can see that Epidepth is having problems with identifying trees. This could not be determined alone just using the provided metrics. Further analysis needs to go into understanding why, e.g., the trees are small structures, EpiDepth's search window was too large, simulation had repetitive textures that made matching features hard, etc. The point is, the metrics give one level of understanding, but its not currently a substitute for qualitative analysis.

Next, Figure 10 shows our measures for DS2. The reason for showing a second set of measures is to explore cross experiment comparison. If we stick with the same set of UFOMap thresholds, then Figure 10 shows a slightly lower `F1 Score` and slightly elevated `MSE Epidepth to GT`. Things got worse for off-Nadir. Why?

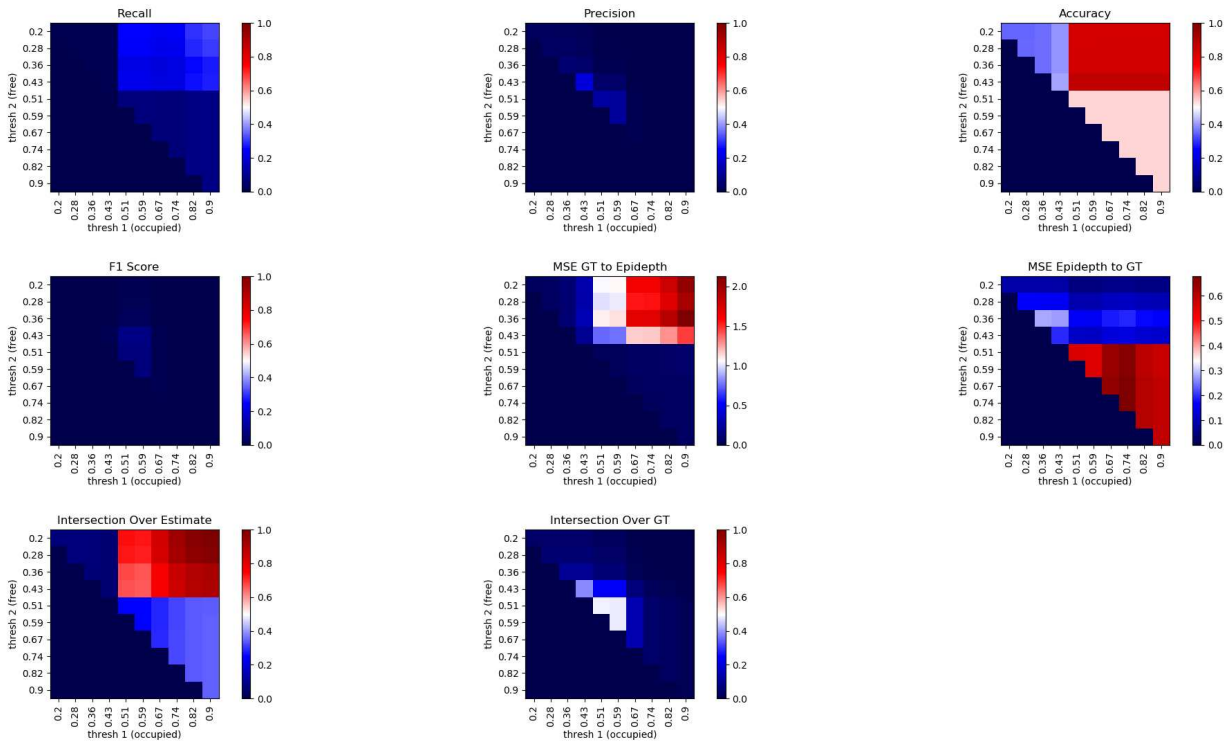


Figure 8: Example measures for DS1. In each image, the x-axis represents the occupied threshold and the y-axis represents the free threshold.

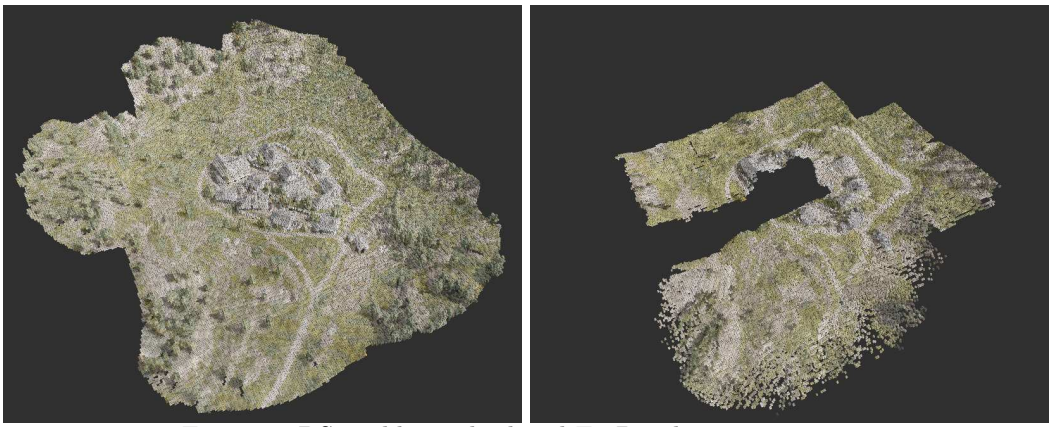


Figure 9: DS1 gold-standard and EpiDepth reconstruction.

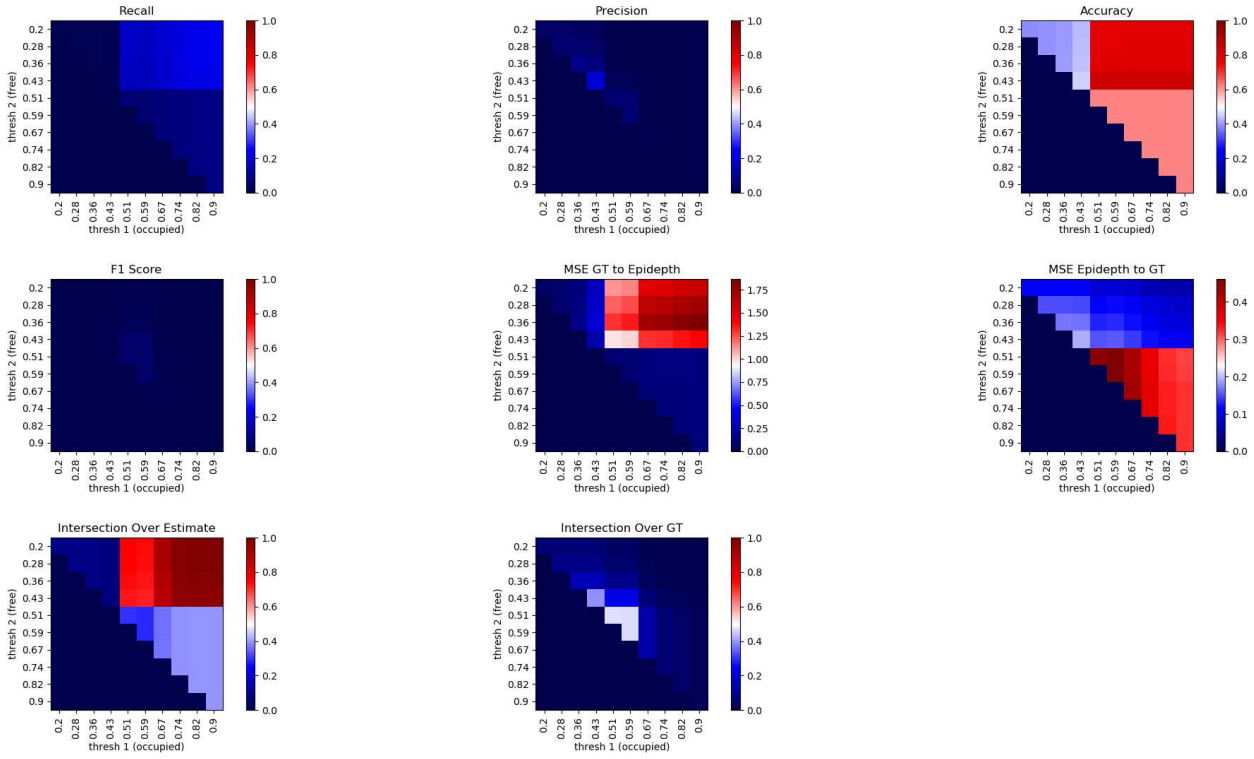


Figure 10: Example measures for DS2.

The point of showing two results is to inform the reader that care should be taken with respect to drawing conclusions across experiments. For example, consider Figure DS1 (Nadir) versus DS2 (slant look angle). These were collected over the same region on the map. This is where the details come in. First, there are two gold-standards, one for each experiment. Second, the slant angle naturally sees more of our map. This makes it challenging to directly compare the measure results. Is it OK if we have a slightly worse F1 Score if we are mapping more space? Should these results be normalized or an intersecting set of co-observed locations be identified and compared? The point is, comparing results across experiments might not be an apples-to-apples experiment.

The point is, we intentionally picked different platform operating conditions because it's a large factor in many areas of research. For example, what is the ideal flight conditions for mapping some area of interest? We are able to, today, do analysis and draw conclusions with respect to a single experiment. However, cross experiment with respect to variation in platform changes presents a harder and open research question. What do we need to do in order to provide a rank-ordered preference across flights? While we focused on platform variation, a simpler and more straight-forward experiment that is less fuzzy is to explore parameters in our algorithm. For example, we could collect one flight, with fixed pose and motion, and change EpiDepth parameters (search window size, frame pair selection, etc.). The platform is not changing, just the 3DR algorithm. This type of multi-experiment exploration is more well-suited for the ideas directly expressed herein.

7. CONCLUSIONS AND FUTURE WORK

In this article, we presented a framework and workflow for collecting SIM data to evaluate the impact of 3DR algorithm parameters and platform motion. Different set and surface-to-surface voxel-based measures were presented. These metrics were then used to individually analyze different datasets.

This article is a preliminary investigation. More research is needed in all of the areas identified above. For example, the proposed measures are focused on aggregated multi-look voxel spaces. We would like to also

explore per-frame measures with improved pixel-level correspondence and aggregation of this information across looks. Our article is also focused on a static scene. It remains a question about how to best address dynamic environments. Furthermore, SIM truth results in a gold-standard. The very nature of mixed truth pixels should be incorporated into our measures, and it's not currently understood to what degree errors in the gold-standard impact and skew our measures. We would also like to explore evaluation of associated metadata, e.g., RGB data or surface normal per voxel. At the moment, we have only considered the task of can the algorithm determine the location of surfaces in the world.

In addition, we only demonstrated preliminary results for one nature scene across eight contexts. A further analysis on multiple scenes with varying conditions, e.g., time of day, weather, etc. should be considered. We also highlighted that a more fair starting point, with respect to comparing measures across experiments, would be algorithm parameter search. We are currently working on a separate article just focused on how to use the human-in-the-loop to achieve this, i.e., 3DR algorithm fine-tuning. Another interesting avenue of exploration is how take this multitude of different measures and to attribute applications. What is the right set of measures and method of combining their logic to make a rank ordering of experiments?

There are also a number of minor, but important, details we would like to address. Herein, AirSim was used to control our agent and get information from UE. AirSim is technically not supported for UE5, but it has been extended by the community. We are working on our own direct C++ plugin that does control and improved photorealism collection via UE's Movie Render Queue. It could be interesting to study what impact these rendering features have on a 3DR algorithm. Furthermore, we would like to improve the generation of data layers, like semantic IDs, as bundles (due to limitations like mixed pixel truth). The measures used herein did not consider class identifier context. Inclusion of this information in evaluation can lead to a much deeper understanding of what "features" drive and limit our CV algorithms.

REFERENCES

[1] Camaioni, R., Luke, R. H., Buck, A., and Anderson, D. T., "EpiDepth: A real-time monocular dense-depth estimation pipeline using generic image rectification," in [*Geospatial Informatics XII*], **12099**, 101–114, SPIE (May 2022).

[2] Schönberger, J. L. and Frahm, J.-M., "Structure-from-motion revisited," in [*2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 4104–4113 (2016).

[3] Kiss-Illés, D., Barrado, C., and Salamí, E., "GPS-SLAM: An augmentation of the ORB-SLAM algorithm," *Sensors* **19**(22) (2019).

[4] Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D., "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics* **31**, 1147–1163 (Oct. 2015).

[5] Mur-Artal, R. and Tardos, J. D., "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics* **33**, 1255–1262 (Oct. 2017).

[6] Gordon, A., Li, H., Jonschkowski, R., and Angelova, A., "Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras," in [*Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*], (Oct. 2019).

[7] Li, H., Gordon, A., Zhao, H., Casser, V., and Angelova, A., "Unsupervised monocular depth learning in dynamic scenes," (2020).

[8] Goldman, M., Hassner, T., and Avidan, S., "Learn stereo, infer mono: Siamese networks for self-supervised, monocular, depth estimation," in [*Computer Vision and Pattern Recognition Workshops (CVPRW)*], (2019).

[9] Hui, T.-W., Tang, X., and Loy, C. C., "LiteFlowNet: A lightweight convolutional neural network for optical flow estimation," in [*Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 8981–8989 (June 2018).

[10] Buck, A., Anderson, D., Camaioni, R., Akers, J., Luke, R., and Keller, J., "Capturing uncertainty in monocular depth estimation: Towards fuzzy voxel maps," in [*FUZZ-IEEE*], (2023).

[11] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W., "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots* **34**, 189–206 (Apr. 2013).

[12] Duberg, D. and Jensfelt, P., "UFOMap: An Efficient Probabilistic 3D Mapping Framework That Embraces the Unknown," *arXiv:2003.04749 [cs]* (Mar. 2020).

[13] O'Meadhra, C., Tabib, W., and Michael, N., "Variable resolution occupancy mapping using gaussian mixture models," *IEEE Robotics and Automation Letters* **4**(2), 2015–2022 (2019).
[14] Tabib, W., Goel, K., Yao, J., Boirum, C., and Michael, N., "Autonomous cave surveying with an aerial robot," *IEEE Transactions on Robotics* **38**(2), 1016–1032 (2022).
[15] Dhawale, A., Yang, X., and Michael, N., "Reactive collision avoidance using real-time local gaussian mixture model maps," in [*2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*], 3545–3550 (2018).
[16] "Gazebo." http://gazebosim.org/. (Accessed: 1 March 2021).
[17] "Robot Operating System (ROS)." https://ros.org. (Accessed: 8 March 2022).
[18] Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., and Susskind, J. M., "Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding," in [*ICCV*], (2021).
[19] Liang, J., Makoviychuk, V., Handa, A., Chentanez, N., Macklin, M., and Fox, D., "Gpu-accelerated robotic simulation for distributed reinforcement learning," (2018).
[20] "Unreal Engine." https://www.unrealengine.com/. (Accessed: 8 March 2022).
[21] "AirSim." https://github.com/microsoft/AirSim. (Accessed: 1 March 2021).
[22] "Unity." https://unity.com/. (Accessed: 1 March 2021).
[23] Rohde, M. M., Crawford, J., Toschlog, M. A., Iagnemma, K., Kewlani, G., Cummins, C. L., Jones, R. A., and Horner, D. A., "An interactive physics-based unmanned ground vehicle simulator leveraging open source gaming technology: progress in the development and application of the virtual autonomous navigation environment (vane) desktop," in [*Defense + Commercial Sensing*], (2009).
[24] Durst, P., Bethel, C., and Anderson, D., "A historical review of the development of verification and validation theories for simulation models," *International Journal of Modeling, Simulation, and Scientific Computing* **08** (01 2017).
[25] Kerley, J., Anderson, D., Alvey, B., and Buck, A., "How should simulated data be collected for ai/ml and unmanned aerial vehicles?," in [*SPIE*], (2023).
[26] Versi, E., ""gold standard" is an appropriate term.," *BMJ* **305**(6846), 187–187 (1992).
[27] Buck, A., Anderson, D., and Fraser, J., "Ignorance is bliss: flawed assumptions in simulated ground truth," in [*FUZZ-IEEE*], (2023).
[28] Degol, J., Lee, J. Y., Kataria, R., Yuan, D., Bretl, T., and Hoiem, D., "Feats: Synthetic feature tracks for structure from motion evaluation," in [*2018 International Conference on 3D Vision (3DV)*], 352–361 (2018).
[29] "Mountain Village Environment." https://www.unrealengine.com/marketplace/en-US/product/mountain-village-environment. (Accessed: 8 March 2022).
[30] "Unreal Marketplace." https://www.unrealengine.com/marketplace/en-US/store. (Accessed: 1 March 2021).
[31] Buck, A., Deardorff, M., Murray, B., Anderson, D., Keller, J., Popescu, M., Ho, D., and Scott, G., "Estimating depth from a single infrared image," in [*MSS*], (2023).

APPENDIX A. FULL DATASET MEASURE RESULTS

The remaining measures for datasets 3 through 8 are reported in Figures 11 through 16.

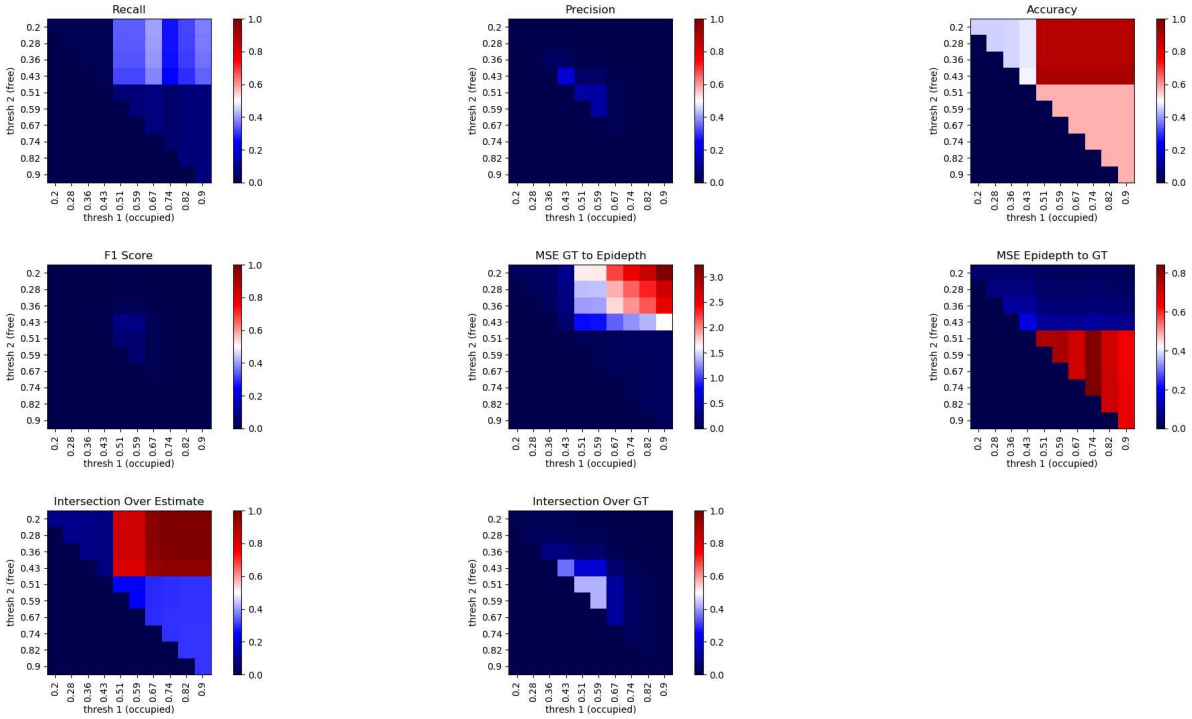


Figure 11: Measures for DS3.

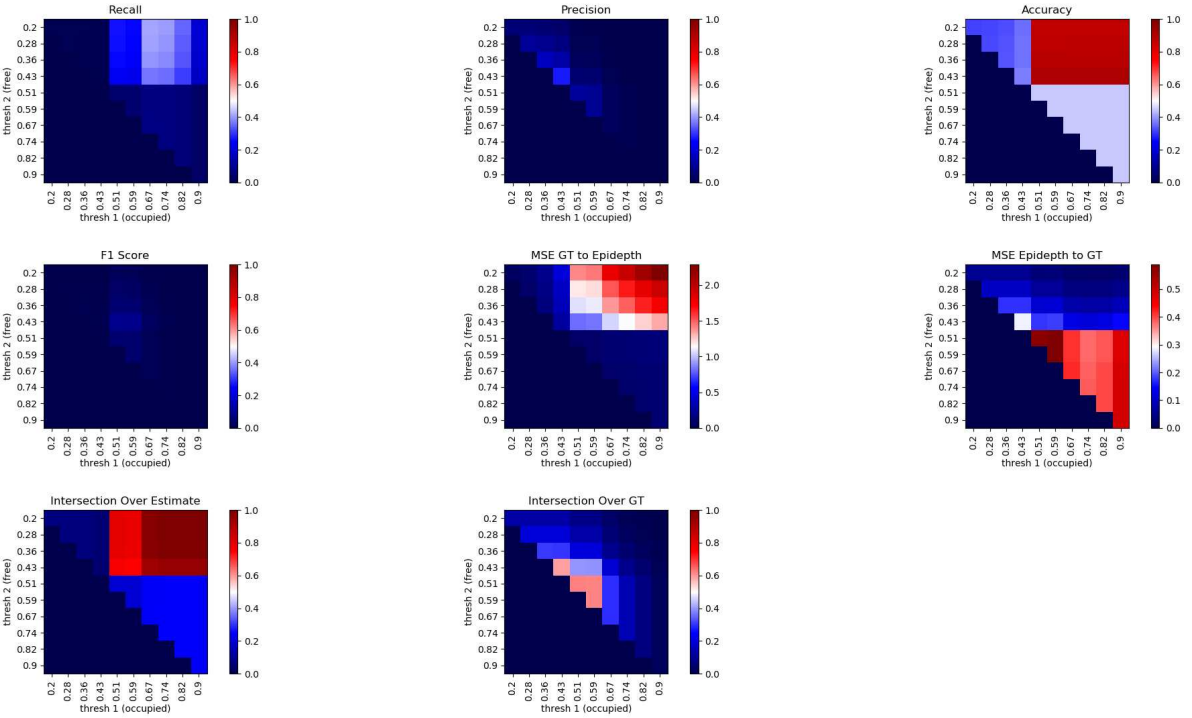


Figure 12: Measures for DS4.

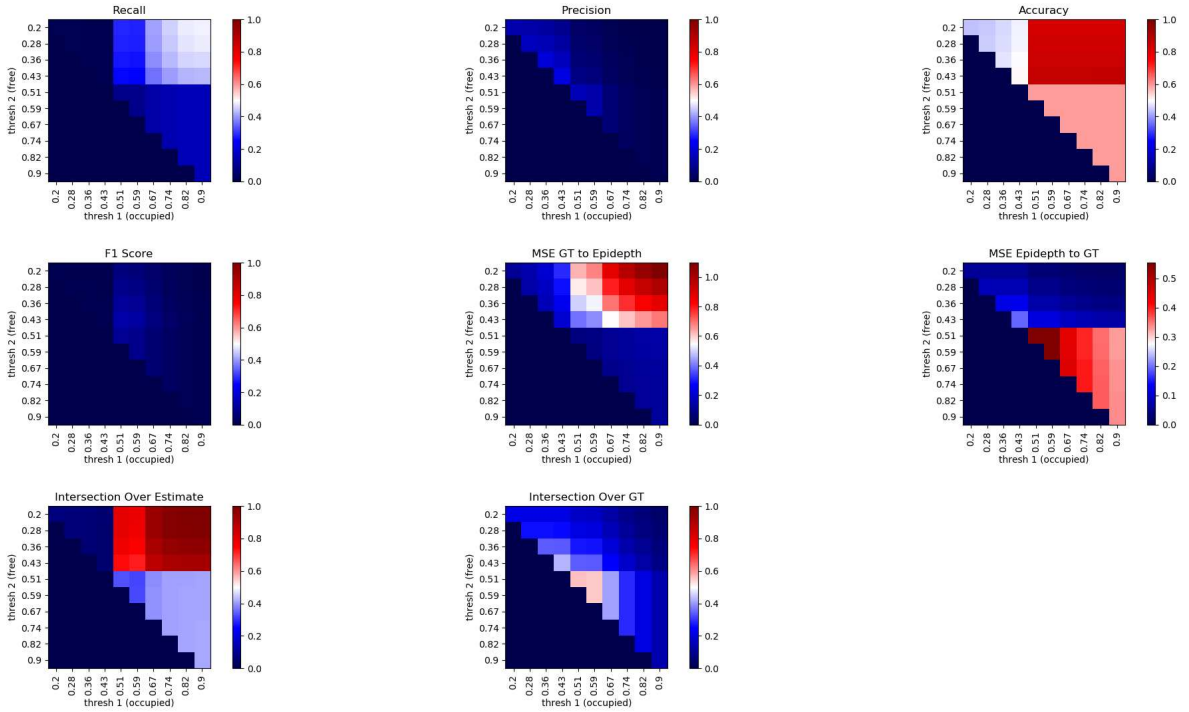


Figure 13: Measures for DS5.

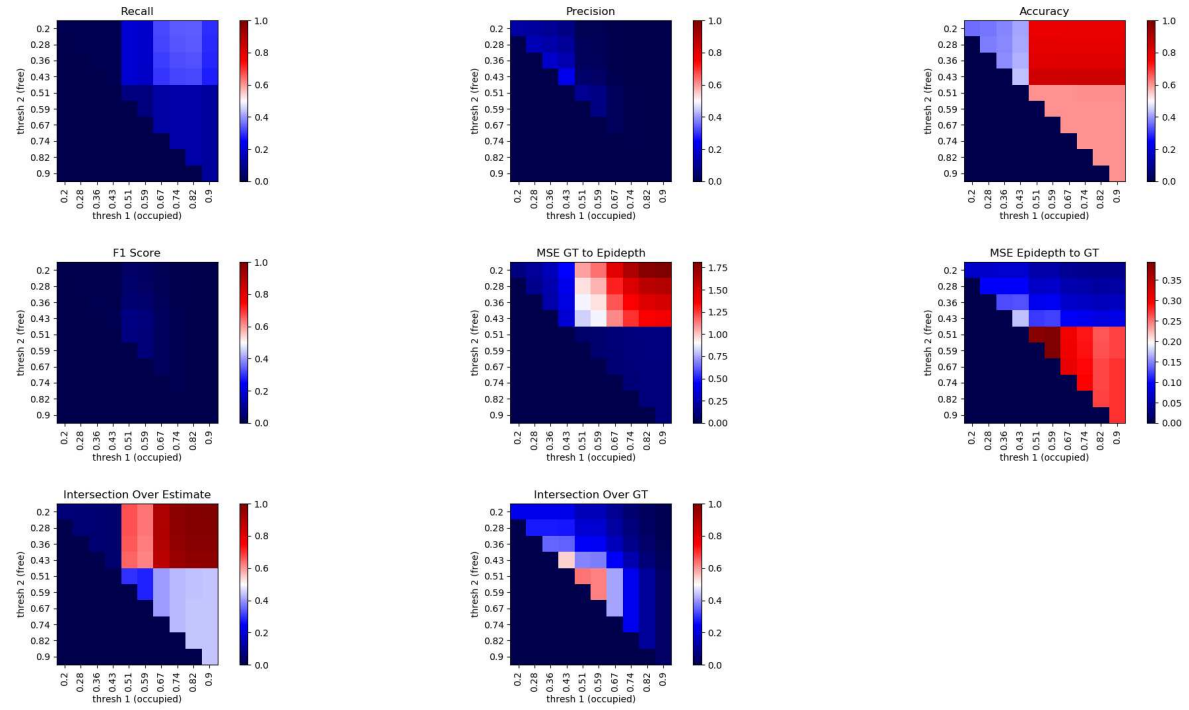


Figure 14: Measures for DS6.

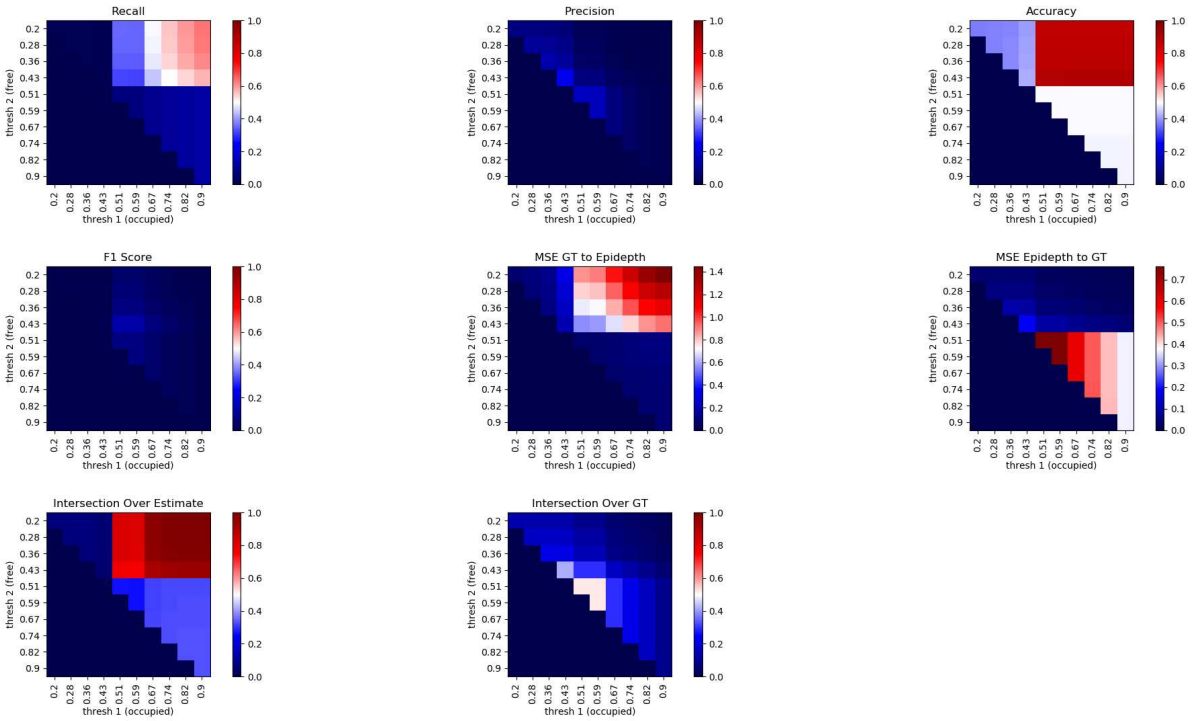


Figure 15: Measures for DS7.

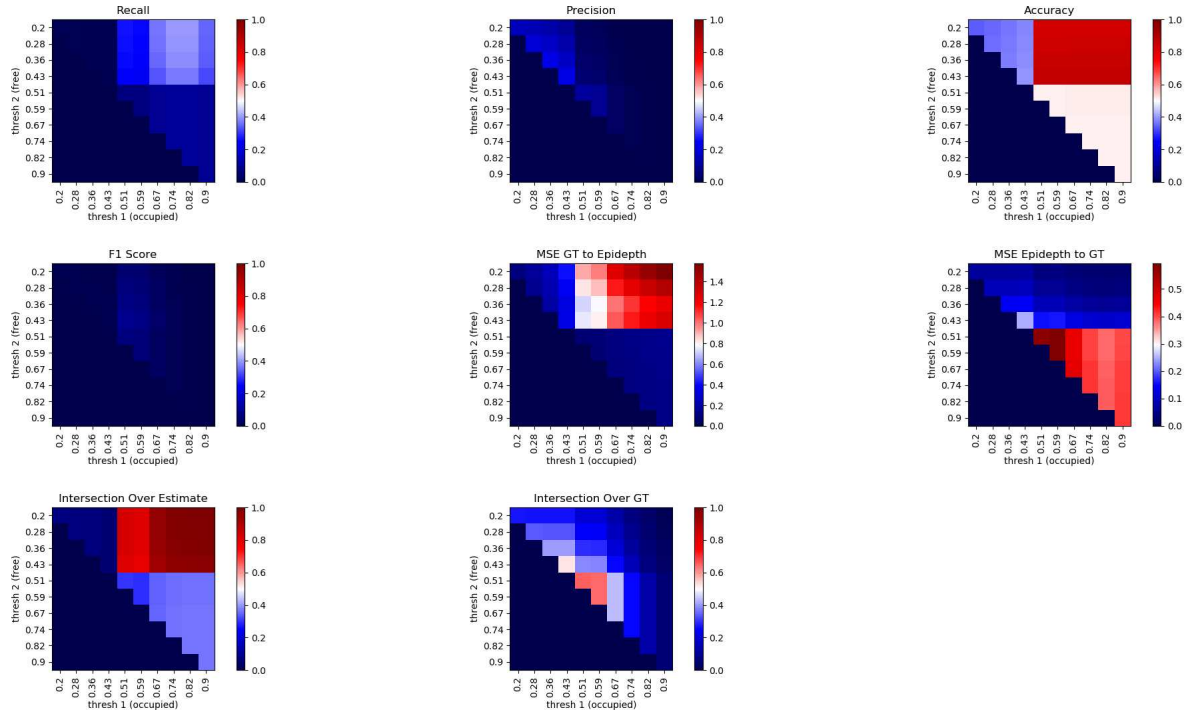


Figure 16: Measures for DS8.