Construction-Site Digital Twins via Gaussian Splatting

Sean Brynjólfsson
smb459@cornell.edu
Cornell University

Dyllan Hofflich
drh253@cornell.edu
Cornell University

Natalie Leung
nl455@cornell.edu
Cornell University

Danish Qureshi
daq8@cornell.edu
Cornell University

Yiwen Zhang
yz864@cornell.edu
Cornell University

1. ABSTRACT

In this paper, we investigate a method to reduce data redundancy and enhance construction site monitoring with an as-built digital twin. We propose a pipeline that combines camera trajectory prediction and Gaussian splatting to accurately reconstruct a construction scene from data collected by an autonomous robot dog. Currently, data generated by construction site monitoring is vast and inefficient to process for digital twin creation. Our results indicate that our method significantly reduces the data volume compared to traditional methods, especially across multiple rollouts. This reduction suggests a promising avenue in creating digital twins and for more effective construction inspections.

2. INTRODUCTION

Digital twins of construction sites serve as an important tool to monitor construction progress. A digital twin in construction ideally contains all up-to-date information of the physical site. Digital twins ensure that all project elements remain consistently updated with what decisions were made on-site. This allows inspectors to easily confirm the alignment of physical work with the design plans, while real-time updates enhance resource allocation and enable more frequent assessments of building conditions. Moreover, digital twins offer the capability to simulate environmental conditions, leveraging weather data to refine construction planning and identify potential errors [7]. Minimizing these errors in turn leads to the prevention of catastrophes and deaths related to uncaught construction flaws. Moreover, we received vital information on monitoring construction progress from Adam Heisler, the FCS Group construction manager for Cornell Ann S. Bowers College of Computing and Information Sciences' new building. Heisler explained that photo documentation is important for proving to contractors that work was done by the correct personnel. Additionally, this documentation is critical when making future changes to a building, like during renovation. Heisler expressed his desire for a tool that could easily allow him to see previous stages of construction without needing to tear down walls for renovations. We believe that a digital twin would be able to accomplish these documentation and temporal aspects.

When autonomous robot dogs traverse construction sites, they gather data in the form of RGB videos and depth readings. However, this amounts to an enormous amount of data collected, even just on a single traversal. From our experience, capturing only 20 seconds of 1280x720 RGB-D images taken at 30 FPS amounts to about 15 GB, which scales to an astronomical amount of data when scanning an entire construction site. Storing this data can be costly, and the sheer size makes it unwieldy. Additionally, according to a study conducted by Autodesk, they "found that 55% of U.S. organizations report that they are creating over 50% more data than they were just 3 years prior" which shows a need to have a reduction in data size [12]. One method for reducing the large file size is building off of previous scans of construction sites, only keeping the changes since the previous scan. For example, when building a wall in a room, scanning the room for each stage of the wall's construction would include plenty of redundant information as only the wall would change. In our paper, we employ the SplaTAM model to convert videos of a wall being constructed over multiple scans into a single 3D model, significantly cutting down the storage required to hold the scene. Saving a construction scene as a 3D model also has the benefit of being easier to use, which is important as according to the same survey by Autodesk: "more than 80% of all respondents describe at least 25% of their project data as unusable" [12]. When asked about the challenges of using their data, the third and second most difficult aspects were that "data is not easily accessible, either due to its structure or the

unwillingness of others to share" and "there is too much data to know how to use it efficiently/effectively" [12]. Once again, creating a system that can transform construction site data into a much smaller size while converting it into a more user-friendly format would solve both of these pressing problems.

3. GAUSSIAN SPLATTING

3.0 3D Gaussian Splatting

3D gaussian splatting is a novel method of rendering radiance fields in real-time, focusing on creating an unstructured representation of radiance fields using 3-dimensional gaussians that are accumulated to form a reconstructed image [8]. Given a set of images, the method calibrates cameras for each of them via Structure-from-Motion, which produces a sparse point cloud as part of the process of calibration. 3D gaussians are then initialized based on this point cloud. Next, properties of these gaussians are gradually optimized, along with steps for density control such as combining or splitting gaussians, until a compact and unstructured representation of the scene is reached. The final product preserves the properties of continuous and differentiable volumetric rendering (avoiding the discontinuities of point sample rendering) while allowing for a fast visibility-aware sorting algorithm of splats to ensure high-quality real-time rendering.

3.1 Usefulness For Digital Twins

In the context of creating and maintaining a digital twin, gaussian splatting is more promising for accomplishing this task than traditional approaches such as mesh-based rendering. This is because gaussian splats are interpretable, editable, composable, and easy to evaluate qualitatively. Splat data is stored akin to point information, with each gaussian listed out with properties such as position, scale, color, orientation, etc., making it very easy to single out a specific spot in a render. This interpretability has the side effect of making splats easy to edit, as specific splats can simply be removed or updated as needed without having to focus on interactions with neighboring gaussians. Extending from this, the process of composing two gaussian splat scenes is also simple since one can simply overlap the two scenes in space, remove intersecting gaussians, or even remove a general region of gaussians. With it also being easy to assess the accuracy of gaussian splats compared to a ground truth, this method checks off many of our boxes for constructing a digital twin that will change dynamically over time.

3.2 Neural Radiance Fields

Neural Radiance Fields (NeRFs) are another method of creating a continuous volumetric rendering of radiance fields. Although NeRFs boast high-quality representations of scenes, this comes at the cost of extremely costly training and rendering times. On the same scene, whereas the Mip-NeRF360 method [11] achieved a PSNR score of 24.3 with a training time of up to 48 hours and a rendering speed of 0.071 fps, 3D gaussian splatting achieved a PSNR score of 25.2 (slightly better than the NeRF) with a training time of 51 minutes and a rendering speed of 93 fps [8]. The slow rendering time of NeRFs can be partly chalked up to their lack of interpretability; frequent random sampling is required in order to determine the colors of pixels, inflating rendering times while still leaving room for inaccurate computations. With gaussian splatting having a better PSNR score with less training and with real-time rendering capabilities, it's clear they will work well for our pipeline.

4. LITERATURE REVIEW: Gaussian Splatting and SLAM

4.0 Introduction of Papers

Here we present a brief literature review of six papers related to Gaussian Splatting and/or SLAM.

[1] SplaTAM (N. Keetha, et al.)
[2] SGS-SLAM (M. Li, S. Liu, et al.)
[3] UVOS[1]*: Unsupervised vdeo object segmentation for enhanced SLAM-based localization in dynamic construction environments (*L. Yang and H. Cai)
[4] *Spacetime Gaussians* (Z. Li, Z. Chen, et al.)
[5] *Dynamic 3D Gaussians*[2] (J. Luiten, et al.)
[6] *Kimera* (A. Rosinol, M. Abate, et al.)

Of these six papers, three [1–3] focus primarily on the construction of static scenes by filtering out (or assuming the absence of) dynamic objects while the other two, [4,5], incorporate motion directly into the final representation. Nevertheless, methods that specifically incorporate dynamic information may prove useful for omitting dynamic information.

[1,2] The authors of this paper did not coin this abbreviation or term for their paper; we shorthand for sake of discussion.

To begin, we first acknowledge that SGS-SLAM [2] (*Semantic Gaussian Splatting SLAM)* is the single best performing method for gaussian-splatting/SLAM in the discipline as of when we started our research. SGS-SLAM is also the most recent and responds to the other papers [1,4,5] while also establishing the most convincing direction for future developments. To provide context for how SLAM is currently being used in general, since SLAM is a widespread technology which long precedes gaussian splatting, we also include UVOS as a comparison of how others are approaching SLAM in ways prior to/other than gaussian splatting on construction sites.

In our experimentation, we used SplaTAM because it was the most compatible open-source repository. For reasons discussed later, SplaTAM is in some ways a good choice for our experimentation and in other ways quite suboptimal.

We discovered that SplaTAM does not work as expected on a sparse dataset (dataset with smaller number of pictures). The predicted camera trajectory isn't accurate enough to splat the gaussians on the right place, leading to distorted artifacts. Therefore, without ground truth for camera pose, we decided to use Kimera [6] to perform camera trajectory and pose prediction. Kimera predicts camera trajectory in a real-time manner using semantic localization and mapping without requiring a large amount of stereo images. It makes predictions with the help of an IMU (Inertial Measurement Unit) that keeps track of the camera linear acceleration and orientation.

4.1 The Representation of Gaussians

There seem to be two main approaches when it comes to representing gaussians: anisotropic or isotropic—meaning that the gaussians can either be asymmetrically stretched or symmetrical in all directions from the origin respectively. In the anisotropic camp, we have [4,5] while [1,2] both use isotropy as a simplifying assumption.

4.1.1 Anisotropic Gaussians

Spacetime Gaussians [4] assumes not only anisotropy of the gaussians, but for each gaussian they also store a feature vector which is responsible for keeping track of both opacity and radiosity (in addition to base color). For each of these features, they also use a small MLP to learn view-dependencies for directed radiosity; they compare this approach to other papers which keep track of spherical harmonics to accomplish a similar feat. This approach is significantly more involved than the paper from *Dynamic 3D Gaussians* [5], which simply stores the color and transformation of each gaussian. This development appears to be a response to the former paper, because the authors of *Spacetime Gaussians* state "their method demonstrates appealing results for 3D tracking, but its rendering quality is less favorable due to flickering artifacts" [4].

4.1.2 Isotropic Gaussians

For isotropic gaussians, not much is to be said about the sophistication of the choice because they are simple. The main motivation for using an isotropic gaussian is the speedup and simplification of the optimization. SGS-SLAM [2] does not seem to consider the specific downsides/upsides and instead appeals to SplaTAM [1] for the decision to use isotropic gaussians. SplaTAM explains the incredible computational upsides to using isotropic gaussians (allows similar representations to run around 10–100x faster) and also that it helps with convergence because the isotropy allows for dense photometric loss. As we will also see soon, using isotropic gaussians lends itself more naturally to dense SLAM.

4.2 Dynamic Objects

Dynamic objects pose a significant problem for SLAM because most (if not all) SLAM approaches make the assumption that you can figure out where you are if you can identify some previously identified keypoints. There are generally two kinds of SLAM, dense and sparse. The sparse version aims to identify a small number of easily identifiable, distinct features to triangulate the current position, dense SLAM tries to identify as many features as possible and then figure out what position explains the current location the best (maximize agreement). SplaTAM and SGS-SLAM [1,2] both assume a static scene, which is a significant limitation because many use cases for mapping cannot be vacated or made still for the mapping run.

UVOS [3] (although not a gaussian splatting method), uses motion saliency masking along with semantic segmentation of dynamic entities like people and shows promising results for masking out keypoints associated with dynamic objects. SplaTAM [1] has a similar masking feature which could be modified to perform the same kind of masking—for SplaTAM this is its silhouette mask. The silhouette mask predicts motion between frames to figure out which parts of a scene would be previously hidden from previous vantages to know what information to focus on.

By contrast, *Spacetime Gaussians* [4] and *Dynamic 3D Gaussians* [5] fully account for dynamic objects, incorporating motion into the scene's representation.

Admittedly, ‘motion’ in the context of these papers is more of the form of a spatial video rather than the construction of a full model representation—time is not ‘factored’ out like in the previous methods [1–3].

Dynamic 3D Gaussians [5] is one of the first (and still SoTA) examples of gaussian splatting being used to localize objects as they move around in a scene. They accomplish this by requiring that gaussian splats satisfy local rigidity constraints and keeping them persistent between timesteps. One downside to this approach is that there is not enough information to guide each gaussian at every timeframe, and this leads to the flickering artifacts.

Spacetime Gaussians, builds off this critique and implements a continuous path for each gaussian so that interpolation is much more smooth. Since gaussians are also permitted to come in and out of existence in this implementation, deformation and changes of geometry are also easier to represent—this is perhaps shown most clearly in the flame-throwing example, where *Spacetime* [4] demonstrates how well their approach captures flames.

4.3 Real-Time Applications

Papers [1,2,4] all present real-time applications of gaussian splats for novel view synthesis. At present, it appears that SGS-SLAM [2] does not publish any information about runtime; since their paper is so new and the code is not released, it is likely they are still in the refactoring process and optimizing their code. Nevertheless, they claim that their approach still works for real-time applications. SplaTAM [1], however, with code available, establishes the fastest times of all the other papers, able to perform SLAM at rates of around 15 Hz and offline create novel views at over 400 FPS. Spacetime Gaussians [4] with its additional overhead is able to perform novel view synthesis at 66 FPS at an 8K resolution; it also has the highest rendering fidelity scores of all the methods.

5. EXPERIMENTATION

5.0 SplaTAM

We use SplaTAM to track and map the path of an ANYmal-D robot roving around the scene that it captures. As stated before, we use gaussian splatting as our means of rendering because it is a novel method that is easily interpretable, editable, composable, and is easy to evaluate qualitatively.

5.1 Simulating Construction Sites

To test our method, we modeled the process of constructing an interior wall with pipes. Starting with a simple room model sourced online [9], we used Blender to model the wall construction using an online frame library [10]. The construction process consists of six stages: (1) inserting the top and floor wooden plates, (2) constructing two studs on either end of the room, (3) adding the remaining vertical studs, (4) inserting cross studs, (5) installing pipes, and finally, (6) putting up sheets of drywall. Then, we added texturing and lighting in Omniverse, which stay constant across each stage. To enhance realism, we placed various construction-related and non-construction objects to reflect the messiness of a typical construction site.

Figure 1: Simulated Stages of Construction

Six stages of constructing an interior wall with pipes were rendered using Blender and Omniverse. The images above depict each stage from left to right. Various objects were placed to mimic the typical untidiness of construction sites.

5.2 Data Collection

To export our data from the simulated scenes, we employed Nvidia's Isaac Sim to simulate an Anybotics ANYmal-D walking around the room using the Anybotics NLP-4-Actuator policy, which is trained for the robot. The weights that were graciously given to us by ETH Zürich and verified on real-life rollouts were slightly modified to have increased velocity commands. We took 1280x720 RGBD images at 30 FPS for the last two stages of construction in Figure 1, amounting to under a minute of footage traversing the room. Due to time constraints, we could not run the simulation on the other stages, so we used the stages with the most drastic change that would allow us to examine if our method modifies occluded objects when incorporating successive scans.

One challenge we initially came across in our data collection was the highly specular nature of the texture of the wooden floors in the construction scenes. The specular texture introduced noise into our results, so we removed the specular component entirely before proceeding with data collection. Consequently, due to the simulated nature of our data, the images gathered in our runs generally do not contain artifacts such as blur, overexposure, underexposure, smudging, or other forms of noise. While this lack of noise is unrealistic, we believe that these conditions are sufficient for showing the data reduction achieved by our method.

5.3 Predicting Camera Trajectory

Having accurate camera trajectory is crucial for SplaTAM. SplaTAM does not perform as well as expected on the camera feed from a walking robot which is often tremulous. Inaccuracies in the camera trajectory prediction cause scene doubling that results in catastrophic failure. Here we present how we envision using Kimera [6] to prepare the camera trajectory for SplaTAM or other gaussian splatters which could take its place.

5.3.1 Kimera

Kimera is an open-source C++ library for real-time metric-semantic localization and mapping using visual-inertial data. It enhances existing SLAM systems by integrating rapid mesh reconstruction and 3D semantic labeling. Kimera is modular, comprising a Visual-Inertial Odometry (VIO) module, a pose graph optimizer, a 3D mesher, and a dense metric-semantic reconstruction module.

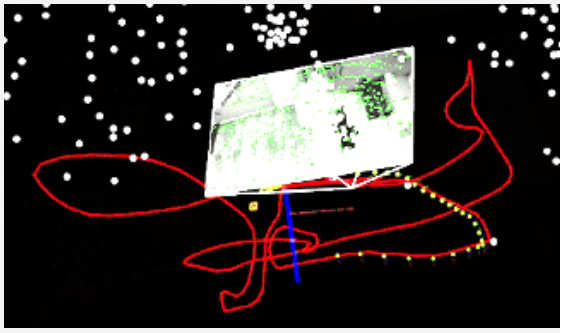


Figure 2: Kimera camera trajectory prediction visualization (red path).

Kimera-VIO: Visual-Inertial Odometry
This module is our current main focus. It takes stereo image sequence and the IMU (Inertial Measurement Unit) data as input and outputs the predicted camera trajectory that SplaTAM needs. It contains a VIO front-end which does Shi-Tomasi corner detection, stereo matching, and RANSAC[15-16] geometric verification. The pipeline then goes into the VIO back-end that integrates the results from the front-end and estimates the 3D positions of observed features using Direct Linear Transform. Finally, states that fall outside the smoothing horizon are marginalized using GTSAM[14]. Visualization of this process is shown in Figure 2.

Kimera-Semantics: Metric-Semantic Segmentation
This module is vital for our future work. It generates a semantic annotated global mesh that labels the detected objects in the reconstructed scene. Since the focus of this project is to eliminate transient data on construction sites, our goal is to detect the transient data during real-time scanning and mask off the objects.

We simulated a quadruped robot walking in the scene that we constructed (shown in Figure 3) in ROS and recorded the construction process in a ROS bag file. The IMU and the camera are both recording at 60 frames per second. The RGB-D images are at 1920x1080p resolution, which is higher for more precise feature detection. The images and data are extracted from the ROS bag file and passed into Kimera with proper intrinsics.

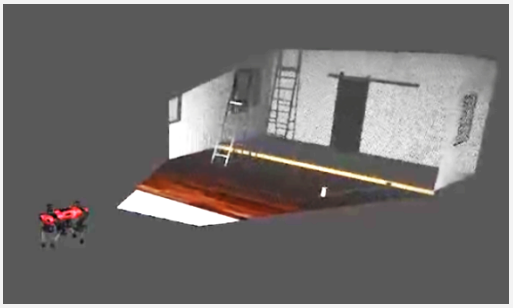


Figure 3: Simulated Quadruped Robot Recording the Constructed Scene

5.3.2 Current Implementation (Dense Video)

Because we could not get Kimera running, in the current implementation we rely heavily on using a large number of photographs, potentially supplemented by robot odometry data (our final results are presented without g.t. odom.), to achieve an accurate mapping. This is because SplaTAM performs silhouette masking, which requires small changes between images to track correctly.

Although SplaTAM requires images which are close together to perform accurate silhouette masking, we noticed given ground truth positions, we could reduce the number of input images dramatically. This suggests a future implementation where ground-truth labels could be provided by a sparse SLAM technique while SplaTAM computes the splatting. Admittedly, this makes SplaTAM itself somewhat redundant, as it means we are using the Spla– without the –TAM. We intended to use Kimera to this effect.

To be clear, we are not concerned with the number of images in the context of SplaTAM; this observation is not an *optimization*. SplaTAM is an algorithm which runs online, only needing *one* image at any given time, even if *many* total images are processed. Rather, we point out that it is possible to implement our results with any sparse SLAM technique and a splatter, rather than combining both (i.e., SplaTAM).

5.4 Data Efficiency

SplaTAM also seems to have regions where it is inefficient with its placement of gaussians. As shown in Figure 4, walls which were shear to the robot's camera developed uncharacteristically dense regions of gaussians (despite being a plain white wall), we believe this is because in the depth image there is a large *depth color* gradient along surfaces which tend to parallel with the camera rays, which encourages the placement of many gaussians. We expect these issues to be resolved in newer variants of SplaTAM.

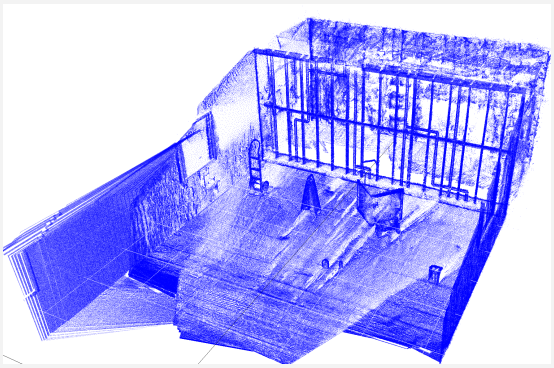


Figure 4: 3D Graph of Gaussian Centers Centers are marked in blue. An abnormally dense region of gaussians is on the left wall.

6. RESULTS

6.1 Our Implementation

Representing a series of images as a 3D Gaussian splatted renderings results in a significant reduction in stored data, especially when tracked over several runs. We found that a 2.4 gigabyte set of RGB-D images capturing our simulated construction site can be represented as a ~650 megabyte set of Gaussian splats. At a different point in time, with only a few new changes to the environment, a second scan will effectively double RGB-D data. For gaussian splats, however, the only added data will be splats representing the new changes, thanks to gaussian splatting's property of composability. Not only is initial data stored much less for gaussian splatting, the rate at which new data is accumulated is also slower. Thus, on top of reducing data required to store a scene, data redundancy is also mitigated.

Our current implementation is naive but demonstrates what future, more sophisticated implementation could pull off. There are some key assumptions which we make that future implementations would need to take into account:

1) We assume that between rollouts, the robot starts at exactly the same position and orientation relative to the first scan. Future work will need to accommodate different starting locations and estimate the relative offset.
2) Our implementation does not cull geometry which is no longer present, this leads to artefacts in the shape of old geometry with the texture of new geometry. Generally the effect is a "paint-over" in space. If geometry is only being covered-up over time, our method works.

3) People and moving/transient objects we intended to filter out as discussed in *§5.3.1* are currently incorporated as-is into the scene. This is especially problematic while we are not limiting what gets splatted because of 2) above.

6.2 On Consecutive Runs, "Paint-over" Effect

We successfully implement multiple stages of construction being composited with our method, though observe some interesting effects caused by the presence of geometry which does not agree with the images being fed in.

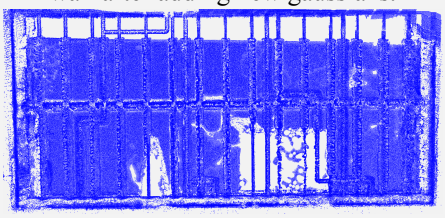
The wall appears well-reconstructed (although we have not collected any pixel-wise metrics to confirm this) and has a texture consistent with the images that were taken by the robot. When looking at the actual structure of the gaussian splat however, it is clear that the underlying geometry impacted the density of the new geometry, even if the final result is hard to refute visually. We expected the wood which covers up the pipes and framing to have a constant density but instead see clear rarefactions in these regions, demonstrated by the clear blue columns in Fig. 5.4.

We also expected geometry which was *removed* (not just covered up) to get removed by the splatter during the next run. This is not the case, and instead the old geometry gets painted over with the texture from the new image. This is quite surprising, because SplaTAM tries to optimize the *depth image* of the splats also, which means that we would have expected the *depth color* of the geometry to be updated and the splats to get deleted or pushed back and be repurposed into a part of the wall.

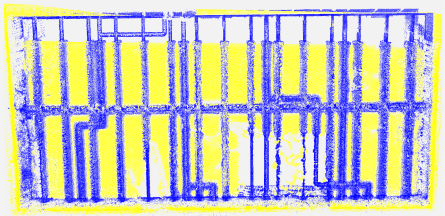
What we see instead is typified by Figure 6. The hand truck, wet-floor sign, and crate from the previous stage of construction have been painted over with the wooden texture, even when the corresponding geometry is no longer present.

(Fig. 5.1) The gaussian splat before.

(Fig. 5.2) The rendered splat of the boarded-up wall after adding new gaussians.



(Fig. 5.3) Plot of the centers of the gaussians comprising the wall. The two large holes near the base are result of geometry which is out-of-plane. *(c.f. Fig 6.)*



(Fig. 5.4) A highlighted hyperplane placed between the pipes and the generated wall.

Figure 5: New Splatted on Old

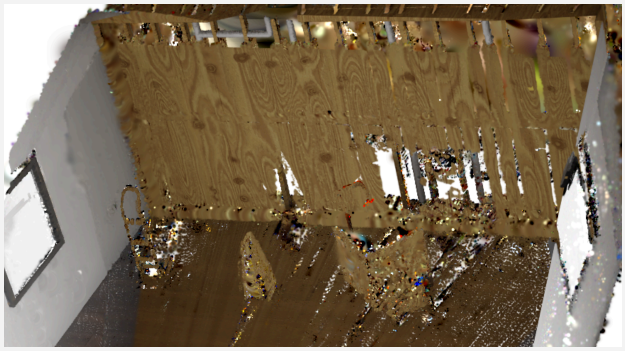
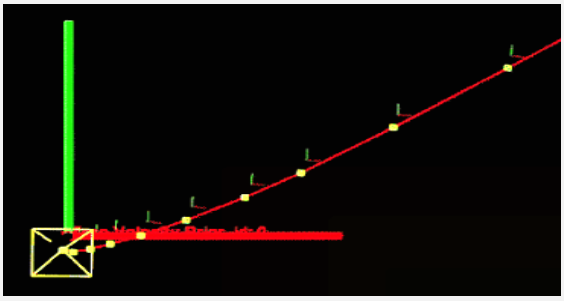
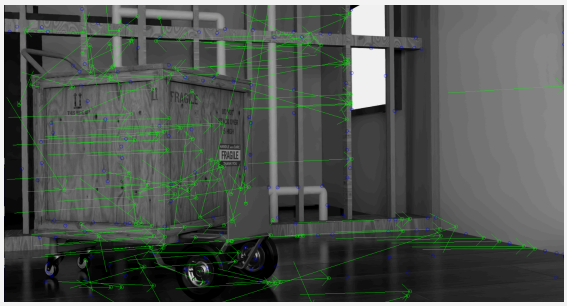


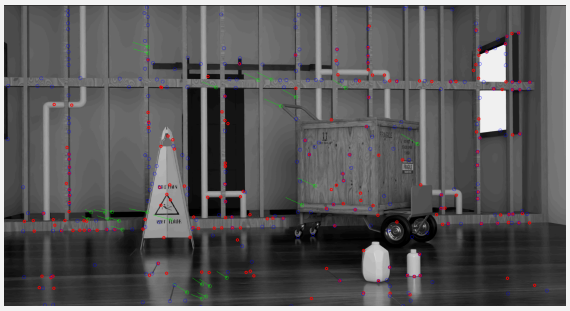
Figure 6: The "Paint-Over" Effect
The same scene in Fig. 3 from a different perspective to show the geometric errors.



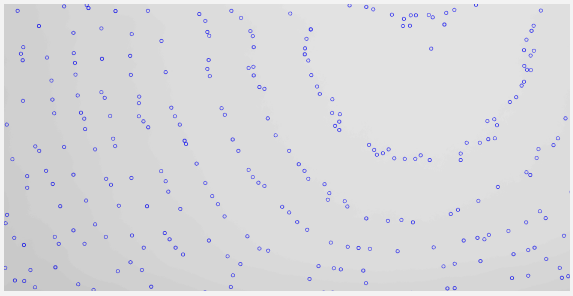
(Fig. 7.1) Camera trajectory drifts.



(Fig. 7.2) Large correspondence vectors



(Fig. 7.3) Correspondence failure (red circles)



(Fig. 7.3) "Features" (blue points) detected for a blank wall due to lack of texture.

Figure 7: Kimera camera trajectory prediction

6.3 Kimera: Camera Trajectory Drifting

Kimera's monocular pipe isn't robust enough to work on our constructed scene. The predicted camera trajectory will shoot out as a straight line after the first few frames (as shown in Fig. 7.1). We faced three main problems:

1) Abrupt camera movement: the simulated quadruped robot turns the camera too quickly so that the detected pixel correspondences generate vectors with huge magnitude (Fig. 7.2).
2) Across-frame correspondence failure due to overly reflective textures: the texture for the wooden floor is mirror-like, reflecting every object in the entire scene. The same pixels appear different across frames, confusing the model to find correspondence (Fig. 7.3).
3) Inaccurate feature detection due to overly smooth textures: there's no significant textural variation on the walls. So when facing the wall, the detected features are inconsistent across frames (Fig. 7.4).

7. FUTURE WORK

We believe that gaussian splats will become a widespread graphics technology in various fields because of their ability to create photorealistic scenes and the many benefits of their representation. For construction, this could be the optimal format for a digital twin. Figure 8 demonstrates what a more thoughtful implementation of our method could look like. Each of the four stages of construction are preserved as updates are made around and/or cover it up. All of the stages are seamlessly combined and stored as a single gaussian splat. The end result of the process is an as-built digital twin which can be spliced later to look behind the walls and see how it was all done—like an MRI for a building.

By tagging each gaussian with a timestamp, we envision an application which lets the user scroll through time to interactively traverse the construction site both spatially and temporally, allowing them to peel away walls to see behind them as it was built. This has many potential benefits to contractors who would like to see through walls before they might have to knock them down or even as a way of certifying proof of work.

Finally, we would like to point to Khronos, a framework for reconstructing spatio-temporal representations for robot environments, as a tool for our future work [13]. Khronos is able to work in dynamic environments, which is critical for monitoring a construction site over time. Specifically, its ability to discern between short-term dynamics and long-term changes can lead to more accurate reconstructions of the scene by eliminating transient data effectively.

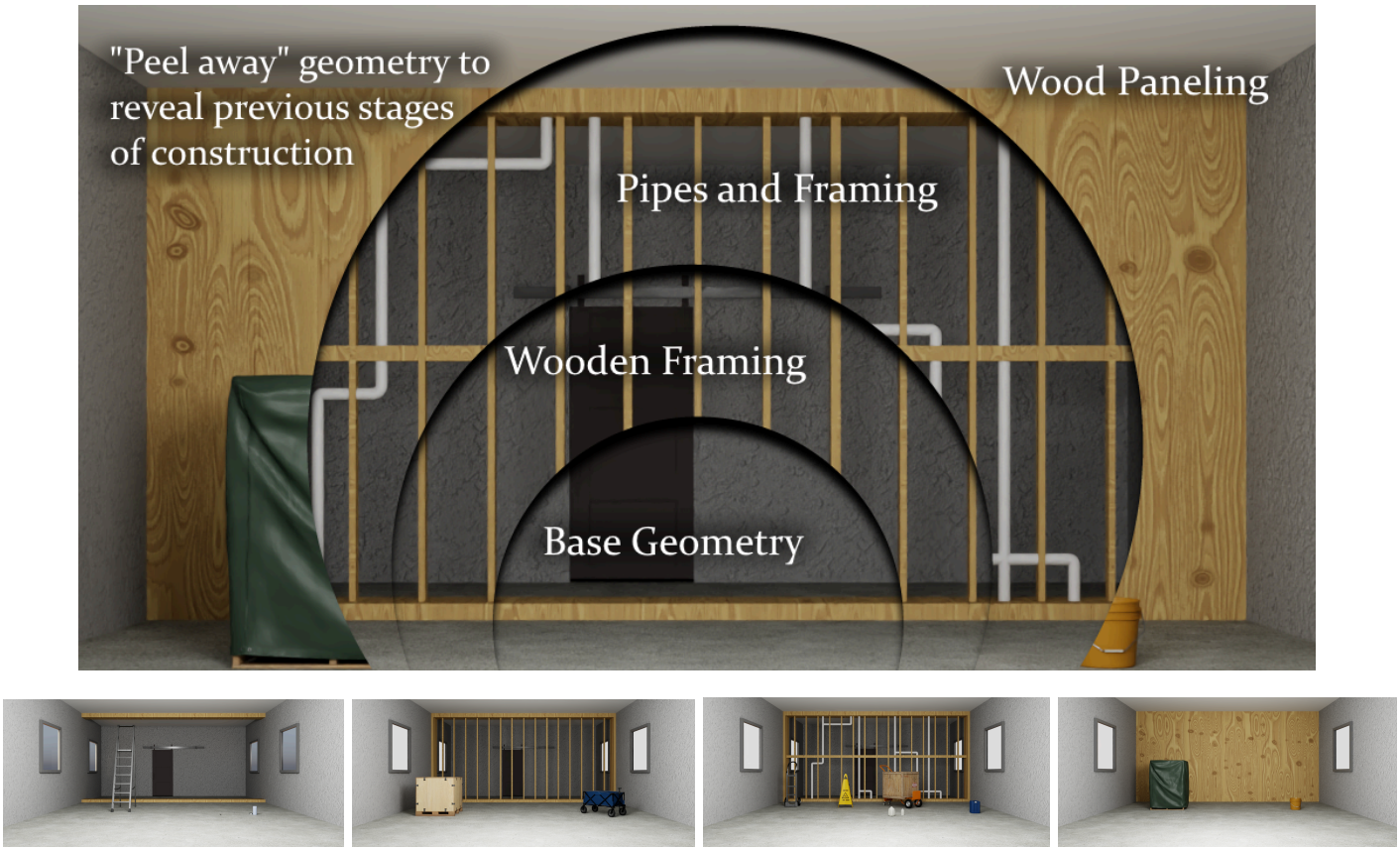


Figure 8: Rendering of Scene Composition

A mock-up of what a more sophisticated method is capable of. The four images below show examples of stages of construction, including transient data which was filtered out automatically.

8. CONCLUSIONS

We demonstrate a preliminary proof of concept that reduces data redundancy in construction scene reconstruction by camera trajectory prediction and Gaussian splatting. Specifically, we showed that if one can localize themselves within an existing Gaussian splat, it is easy to add new geometry, such as a wall, while preserving features inside that wall. For a single rollout, utilizing this method results in a 72.5% data reduction rate compared to taking and storing RGB-D images of the scene; we demonstrated that a 2.4 GB set of RGB-D images can be condensed into about 650 MB worth of Gaussian splats. Importantly, we argue that our method saves data across multiple rollouts. With just a few new changes to the environment, a second scan effectively doubles RGB-D data, while Gaussian splatting only adds splats representing the new changes. There are still drawbacks to our method as our Gaussian splatting model cannot cull geometries correctly, leading to a “painting-over” effect as shown in Figure 6. For robots which jolt as they walk, changes between frames may be too significant for SplaTAM to account for, leading to catastrophic scene doubling. We attempt to remedy this problem via Kimera-VIO to predict the camera trajectory. This was not implemented into the 3D reconstruction model within our timeframe. While our method has limitations, it does show promise for being able to downsize the amount of data which is required to keep track of a construction site and creating a digital twin, as well as opening the door for newer technologies that interface with gaussian splats.

REFERENCES

[1] N. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten, "SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM," arXiv, 2023.

[2] M. Li, S. Liu, H. Zhou, G. Zhu, N. Cheng, and H. Wang, "SGS-SLAM: Semantic Gaussian Splatting For Neural Dense SLAM," 2024. arXiv:2402.03246 [cs.CV].

[3] L. Yang and H. Cai, "Unsupervised video object segmentation for enhanced SLAM-based localization in dynamic construction environments," *Automation in Construction*, vol. 158, 2024, Art. no. 105235, ISSN 0926-5805.

[4] Z. Li, Z. Chen, Z. Li, and Y. Xu, "Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis," 2023. arXiv:2312.16812 [cs.CV].

[5] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan, "Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis," in 3DV, 2024.'

[6] Rosinol, A., Abate, M., Chang, Y., & Carlone, L. (2019, October 6). Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping. arXiv.org. https://arxiv.org/abs/1910.02490

[7] Zhang Jiaying, Cheng Jack C. P., Chen Weiwei, and Chen Keyu. "Digital Twins for Construction Sites: Concepts, LoD Definition, and Applications." Journal of Management in Engineering, vol. 38, no. 2, 2022, 04021094. American Society of Civil Engineers. https://doi.org/10.1061/(ASCE)ME.1943-5479.0000948.

[8] Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics, 42*(4). Retrieved from https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[9] Natcrot. 2021. WhiteRoom1[3D model]. Retrieved from https://sketchfab.com/3d-models/white-room1-26a61fba0d5d41dcbe01478de9831218#download

[10] BigRyan. 2017. Sweet Home 3D Online Frame Library [Online frame library]. Retrieved from https://sourceforge.net/p/sweethome3d/d-models/421/

[11] Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., & Hedman, P. (2022). Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. arXiv.org. https://arxiv.org/abs/2111.12077.

[12] *Harnessing The Data Advantage In Construction*. Autodesk. (2022, October 18). https://construction.autodesk.com/resources/guides/harnessing-data-advantage-in-construction/

[13] Schmid, L., Abate, M., Chang, Y., & Carlone, L. (2024, February 21). KhRonos: A Unified Approach for Spatio-Temporal Metric-Semantic SLAM in Dynamic Environments. arXiv.org.https://arxiv.org/abs/2402.13817

[14] F. Dellaert et al., "Georgia Tech Smoothing And Mapping (GTSAM),". https://gtsam.org/

[15] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," IEEE Transactions on Robotics, vol. 34, no. 4, pp. 1004–1020, 2018.

[16] T. Qin, J. Pan, S. Cao, and S. Shen, "A general optimization-based framework for local odometry estimation with multiple sensors," arXiv preprint: 1901.03638, 2019.