Abstract of "Semantic motion segmentation for scene reconstruction with Gaussian Splatting and Time-of-Flight" by Anh Duong, Sc.B., Brown University, May 2025.

There has been significant progress in dynamic scene reconstruction, particularly in improving quality, efficiency, training time among others, yet achieving an interactable system is a significant challenge. However, as state-of-the-art (SOTA) scene reconstruction methods start to enter commercial applications, improving interactibity should be the main focus. This thesis aims to explore this space by discussing how to improve editability through using semantic motion segmentation, and additionally whether it can further improve quality. We investigate multiple different segmentation methods and incorporate their results into a SOTA Time-of-Flight Gaussian Splatting reconstruction pipeline. Through our experiments, we received positive results that led us to believe semantic motion segmentation has a future in scene reconstruction.

Semantic motion segmentation for scene reconstruction with Gaussian Splatting and Time-of-Flight

by
Anh Duong

A Thesis submitted in partial fulfillment of the requirements for Honors
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2025

This thesis by Anh Duong is accepted in its present form by
the Department of Computer Science as satisfying the research requirement
for the awardment of Honors.

Date _____ _____

James Tompkin, Advisor

Date _____ _____

Daniel Ritchie, Reader

Acknowledgements

I would like to thank my advisor Professor James Tompkin, Mikhail Okunev and Runfeng Li for their guidance and support throughout the process. I would like to thank Professor Daniel Ritchie for providing feedback on my thesis writing. Finally, I am grateful for the resources provided to me by Brown University's Computer Science Department and Brown Visual Computing Lab.

Contents

Chapter 1

Introduction

1.1 Introduction

Monocular dynamic reconstruction aims to reconstruct the scene through space and time from a single camera view. This problem is crucial for downstream applications on consumer devices with one or few collocated cameras. The hardware constraint makes this a highly under-determined problem due to lack of geometry data. To make up for this, a popular approach is to use an additional Time-of-Flight (ToF) depth sensor, already existing on many phone models.

Recent scene reconstruction efforts led to two popular approaches – neural radiance fields (NeRF [15]) or using 3D primitives (Gaussian splatting [10]). NeRF is smaller in storage size but much slower to train and render; meanwhile, Gaussian rendering is much faster despite its size. A recent method combining Gaussians and ToF signals significantly improving training and rendering time, making it even more attractive to popular consumer devices. As a result we chose this method to build our research upon.

Dynamic reconstruction methods on Gaussians can be split into two categories: using a deformation MLP to learn the movement of Gaussians across time or directly optimizing motion as a component of each Gaussians. Either way, these methods assume that all Gaussians are dynamic, which might lead to unwanted artifacts in dynamic regions [13]. Some works have explored the removal of these artifacts through the separation of static and dynamic parts of the scene. GauFre [13] attempts to indirectly segment at the pixel level by relying on Structure-from-Motion to initialize static Gaussians. On the other hand, other works looked into semantic segmentation of dynamic objects. Chen [2] manually create dynamic object masks, but only to reconstruct occluded areas. Liang [14] semantically segments dynamic objects but focuses on separating foreground from background. While needing to rely on semantic cues, object-level (semantic) segmentation might serve the same benefits as pixel-level (non-semantic) segmentation while holding more meaningful information (semantic information) in its masks.

As we are focused on designing a system for consumer devices, maximizing user interactivity is

crucial. Therefore, it is natural to choose object-level segmentation over pixel-level segmentation in this specific setting.

This led to our research questions:

- Does semantic motion segmentation help to improve the quality of monocular dynamic RGB and ToF reconstruction?
- Does semantic motion segmentation improve editability? Specifically, how easy it is to separate Gaussians representing dynamic objects?

To assess quality improvement, we compare reconstruction results with segmentation against results without. To assess editability, we separately render Gaussians belonging to the dynamic object and visually check against inputs.

Our key contribution in this thesis report is to provide insight on whether it is truly possible with current methods to derive reliable motion masks, and sequentially apply these masks meaningfully to scene reconstruction. The results of this thesis will be helpful in answering our research questions.

1.2 Background

We discuss the methods we are building upon (Sec. 1.3.1 & Sec. 1.3.2) and current segmentation approaches (Sec. 1.3.3).

1.2.1 Neural Rendering for Scene Reconstruction

Neural rendering has become increasingly popular in scene reconstruction methods. NeRF was well-received because its neural field scene representation allows for low storage cost, high scene complexity and scalability, and accurate novel view synthesis. Recently, 3D Gaussian Splatting (3DGS) outperformed NeRF in training and rendering speed. 3DGS represents a scene using 3D Gaussians as primitives, each defined as:

$$G(x) = e^{-1/2(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{1.1}$$

where μ is the center of the Gaussian in world space, and Σ is the 3D covariance matrix defined in world space. To represent view-dependent colors, each Gaussian has a set of spherical harmonics coefficients.

Covariance matrix Σ can be replaced with rotation matrix R and scale matrix S:

$$\Sigma = RSS^T R^T \tag{1.2}$$

Given a ground truth frame (camera view) F_i, Gaussians are splatted to the screen through rasterization to create a render F_i'. Training Gaussians is then defined as minimizing $L(F_i, F_i')$ through optimizing R, S, μ, spherical harmonic coefficients, and an additional α parameter (for density) for each Gaussian.

1.2.2 Time-of-Flight in scene reconstruction

Time-of-flight (ToF) data is crucial for monocular reconstruction as it provides geometry information that captured RGB does not. ToF cameras calculates depth by measuring the time a signal travels from the emitter, into the scene, then back to the sensor. Continuous-wave ToF cameras capture four raw images, modeled as $\mathbb{Q}_\phi = \{A\sin(\psi + \phi) + B\}$, where A is the amplitude (light received at each pixel), B is the bias (ambient illumination), and $\phi \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. Depth d_{ToF} is then calculated through the phase shift ψ, speed of light c and the camera modulated frequency f [6] as presented below:

$$d_{\text{ToF}} = \frac{\text{c}}{4\pi f}\psi \quad \text{where} \quad \psi = \arctan\left(\frac{Q_0 - Q_\pi}{Q_{\frac{\pi}{2}} - Q_{\frac{3\pi}{2}}}\right), \tag{1.3}$$

$$\text{because} \quad \arctan\left(\frac{Q_0 - Q_\pi}{Q_{\frac{\pi}{2}} - Q_{\frac{3\pi}{2}}}\right) =$$

$$\arctan\left(\frac{A\sin(\psi) + B - (-A\sin(\psi) + B)}{A\cos(\psi) + B - (-A\cos(\psi) + B)}\right) =$$

$$= \arctan\left(\tan(\psi)\right) = \psi.$$

There have been methods that attempted to reconstruct neural radiance fields using ToF data. TöRF [1] takes RGB and ToF data as inputs to the neural radiance field, but due to its assumption that raw images (quads) are captured simultaneously, the model has high errors for fast-moving objects. F-TöRF [17] fixes this problem by considering each of these raw images at separate timesteps. However, both approaches were computationally expensive due to neural fields' unavoidable raytracing and occasionally produced erroneous depth maps.

TotFotG [12] replaced neural radiance fields in F-TöRF with 3D Gaussian Splatting to improve training and rendering time. Though integrating ToF into 3D Gaussian Splatting resulted in brittle optimization, the authors introduced two heuristics to stabilize training and achieved comparable or even better geometry reconstructions.

TotFotG's inputs are 4 ToF raw quads for every frame and optional RGB frames.

They initialize 3D Gaussians randomly, and optimize their positions, covariance matrices, and amplitudes. Unlike neural radiance fields where time is treated as an input, they explicitly transform Gaussians across timesteps through a simultaneously trained MLP. The MLP predicts offsets between a learned canonical frame and each time step: $\delta x = \text{MLP}(x, t)$, where x is the Gaussian's canonical frame position and t is the time.

While this approach provides a solid starting point for dynamic ToF reconstruction, their assumption that all Gaussians are dynamic could impede optimization and limit editability.

1.2.3 Semantic & motion segmentation

According to Hou et al. [7], popular deep learning based semantic segmentation methods on monocular video can be divided into four types: moving object segmentation (MOS), video object segmentation (VOS), video salient object detection (VSOD), and moving object detection (MOD). These

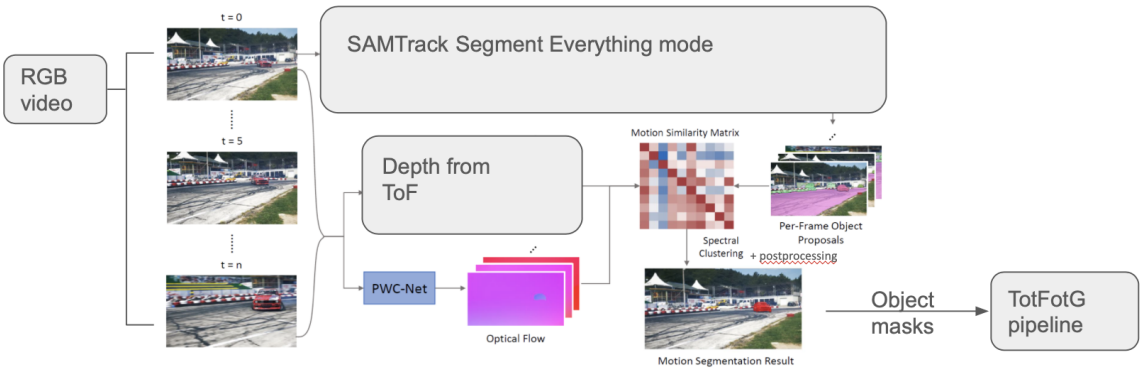

Figure 1.1: **Initialization pipeline overview.** Instead of initializing from random initial points like in TotFotG, we replace the initialization process with this pipeline, adapted from Huang et al [8]. Taking in the input RGB video, SAMTrack in 'segment everything' mode segments all objects in the first frame and tracks them across all frames. RAFT takes the RGB video and produces optical flow for each frame. Then, masks, optical flow, and depth from ToF are used to create a motion similarity matrix. We perform spectral clustering and post processing on the matrix to segment moving objects. The masks are then to be used in conjunction with other inputs in TotFotG.

methods unfortunately do not segment all moving objects, but only a selected subset. MOS segments only objects in the foreground, VSOD and VOS segment only moving objects that are the most visually important, and MOD does not perform any segmentation. In contrast, we need to segment all moving objects. Several works [3] [19] propose some CNN architectures, yet we are constrained by data availability and compute resources, meaning methods that utilize pretrained models are preferred.

Due to this constraint, Jiang et al. [9] researched the use of pretrained models, but the models are still limited by a small set of training data . With SAM's [11] arrival, generalizability became somewhat of a solved problem. This inspired Huang et al. [8], which fulfilled our requirements mentioned above. While the paper uses Recognize Anything Model [20] to choose objects to segment thus is similar to a VOS method, we modify the pipeline to instead segment all objects in the scene.

Huang et al. utilizes the SAMTrack [4] framework which wraps around SAM and some additional works. SAMTrack can be prompted to segment an object, multiple objects, or everything from a user-defined starting frame, then it tracks these objects across all frames. The result of the tracking is a collection of masks consistent across frames.

Chapter 2

Methodology

2.1 Proposed Method

We discuss multiple segmentation methods along with one approach for incorporating segmentation into TotFotG's pipeline. The purpose of this investigation of multiple methods is to detail all experimented approaches to this problem, all of which have yet to succeed. We aim to learn from these failures and identify a more effective method for generating satisfactory segmentation masks.

The first segmentation method is described below, with the remaining methods detailed in appendix A.

2.1.1 Segmentation method 1

Our first method is illustrated in fig. 1.1. Extending from TotFotG, we apply segmentation at the initialization stage. This segmentation pipeline is adapted from the methodology proposed by Huang et al. [8]. Given an RGB input video, all frames are processed using SAMTrack [4] 'Segment Everything' mode, which returns consistent object masks across frames. At the same time, these frames are fed into RAFT [18] to obtain optical flow between consecutive frames. The resulting optical flow, object masks, and depth (derived from ToF signals) are then used to compute a motion similarity matrix. As its name suggests, this matrix quantifies the pairwise motion similarity between objects ('object' is a segmented group of pixels). Spectral clustering is applied to this matrix to group objects with similar motion patterns.

To determine which of these groups are dynamic, an additional post-processing step is required. This module, which we introduce (unlike Huang et al.), is essential for explicitly segmenting motion. Finally, static and dynamic object groups are passed into separate reconstruction pipelines.

The creation of the matrix and spectral clustering are described in detail in Huang et al. We now describe the post-processing module in detail.

For each frame pair (F_1, F_2), we sample the set of all pixels P in frame F_1. At pixel x_1 in frame F_1, its movement is encoded in the estimated forward optical flow as $\vec{v}$. The corresponding pixel in

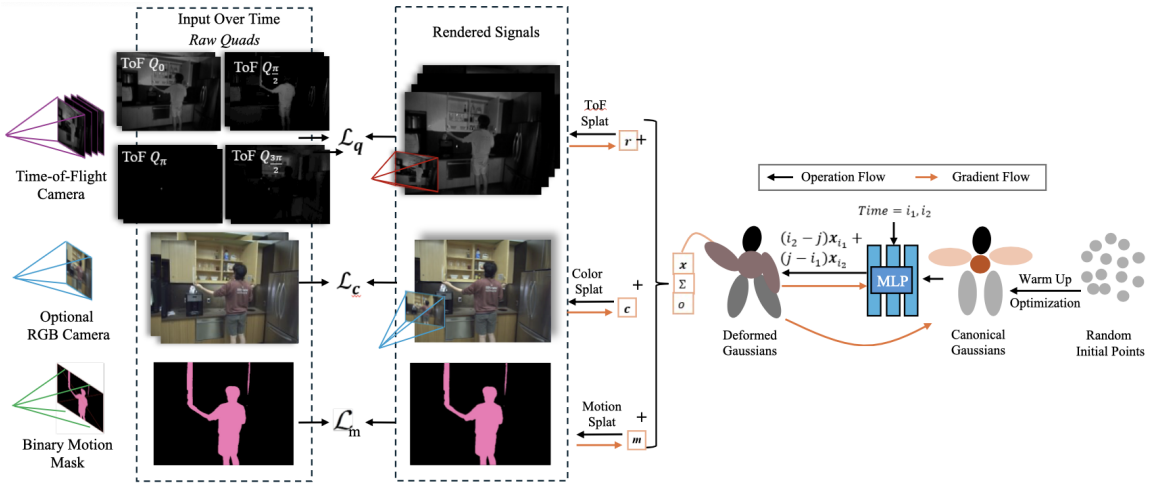

Figure 2.1: **Modified TotFotG pipeline overview.** We have motion masks as an additional source of input. Gaussians' 'motion' property gets rendered, compared with ground truth masks, loss is calculated, back propagation is executed.

frame F_2 is then

$$x_2 = x_1 + \vec{v} \tag{2.1}$$

x_2 can be converted from F_2's space to F_1's space with

$$x_2' = K_1 M_1 M_2^{-1} K_2^{-1} x_2 \tag{2.2}$$

where K_1, K_2 are corresponding known intrinsic matrices, M_1, M_2 are estimated extrinsic matrices, x_2' is the actual position of x_2 in F_1's space. The actual movement of x_1 is then

$$\vec{v'} = x_2' - x_1 \tag{2.3}$$

If $|\vec{v'}|$ is larger than a certain threshold δ, x_1 is considered to be dynamic. If an object group has at least a certain number of dynamic pixels Δ then the object group is considered dynamic in this frame pair. An object is considered dynamic across frame pairs if at least in one frame pair it belongs to a dynamic object group.

2.1.2 Segmentation in scene reconstruction

To support motion segmentation, we made changes to the original TotFotG's pipeline, as shown in fig. 2.1.

After the separation of static and dynamic objects, for each frame F_i, a render M_i is created where a pixel is set to blue (RGB(0,0,1)) if it corresponds to a static object and red ((RGB(1,0,0)) if dynamic.

Each Gaussian is now assigned an additional trainable 'motion' property $m_j = RGB(m_{jr}, m_{jg}, m_{jb})$ where $0 \leq m_{jr}, m_{jg}, m_{jb} \leq 1$. For each frame F_i, m_j is optimized to minimize the loss L_M between

the rendered 'motion' property of all Gaussians M_i' and the ground truth M_i:

$$L_M = abs(M_i' - M_i) \tag{2.4}$$

Gaussians that are static should converge to a 'motion' value of $vector3(0, 0, 1)$, and dynamic ones should converge to $vector3(1, 0, 0)$. The choice to utilize vector values instead of binary values is to allow for easier visualization.

Upon initialization, an equal number of static and dynamic Gaussians are randomly created. Through experiments, we found that the best results are achieved by training Gaussians without motion supervision in a warm-up phase, before introducing L_M minimization.

Chapter 3

Results

3.1 Data and testing environments

We use the synthetic and real-life data provided by TöRF and F-TöRF.

Our hardware for most of the pipeline is the CCV cluster, and Brown Visual Computing Lab's Alchemist machine for running SAMTrack.

3.2 Results

3.2.1 Segmentation method 1 qualitative results

Due to the lack of quality in our motion grouping results, we chose not to implement the post-processing module. We evaluated our pipeline on TöRF real-world scenes and F-TöRF synthetic and real-world scenes. While the pipeline performed well in the F-TöRF synthetic setting, the results on real-world scenes were suboptimal.

fig. 3.1 segmentation results generated by SAMTrack, while fig. 3.2 displays the optical flow outputs from RAFT.

fig. 3.3 illustrates the motion grouping outcomes across four scenes. For consistency, we constrained the number of motion groups to five. Each group is represented by a distinct color. Note that the color assignments may vary between frames, even when the underlying object groupings remain unchanged. This inconsistency likely arises from the algorithm misinterpreting camera motion as object motion.

Cupboard scene: In this scene, the arm and door consistently belong to the same motion group. However, because SAMTrack frequently segments the upper arm and torso as a single object, the motion grouping becomes inaccurate — the body's movement is misgrouped with unrelated background elements.

Deskbox scene: The only dynamic objects are the person and the box. However, the grouping algorithm erroneously associates the person with foreground objects and the box with background

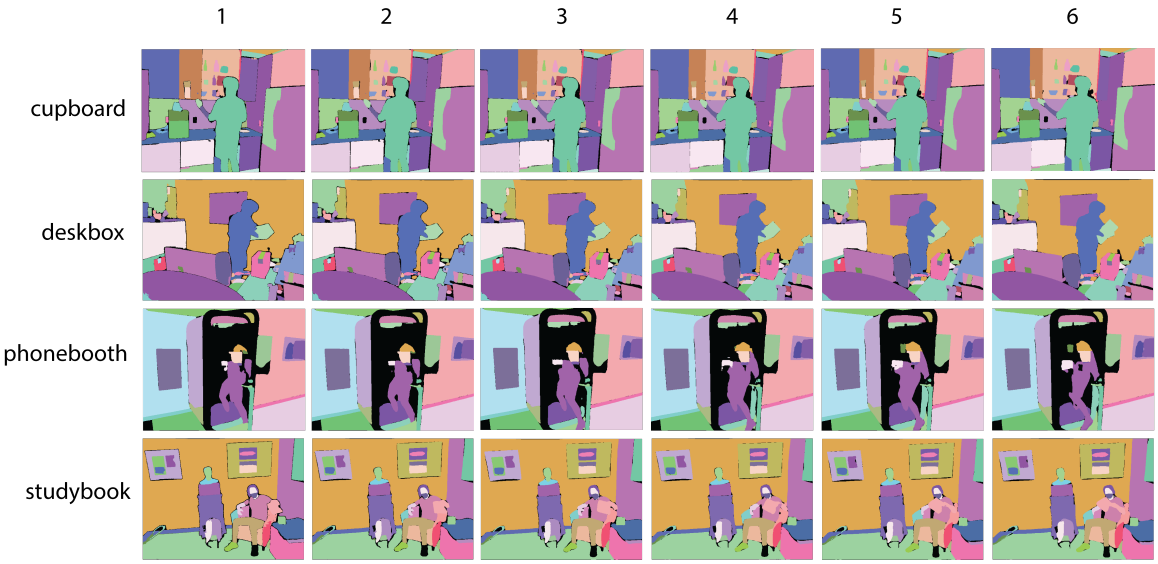

Figure 3.1: **SAMTrack masks extracted with 'Segment Everything' mode on the first six frames of four scenes *Cupboard*, *Deskbox*, *Phonebooth*, *Studybook***; input to segmentation method 1

elements. This misclassification is particularly apparent when objects are grouped into two motion categories, as shown in fig. 3.4. This issue likely stems from camera movement, which causes static foreground objects to seem dynamic. This hypothesis is supported by the optical flow inputs (fig. 3.2), which show similar motion vectors for the person and surrounding foreground objects. On the other hand, the optical flow shows the book's different motion trajectory from the rest of the scene. This distinction caused the book to be grouped with background objects.

Phonebooth scene: Although the door is stationary, its optical flow in fig. 3.2 exhibits apparent motion due to the reflections on the door's material. This may account for its frequent classification as a separate moving object. The person, who is actually in motion, is typically grouped either independently or with the door.

Studybook scene: Motion grouping results for this scene are inconsistent across frames. In many instances, moving body parts are grouped with static elements such as the floor or the bust statue.

In contrast, the algorithm demonstrates significantly better performance on synthetic scenes, including *Sliding Cube*, *Arcing Cube*, and *Occlusion* from the F-TöRF dataset. Because SAMTrack's "segment everything" mode produces unusable masks in these cases (see fig. 3.5), we manually selected object masks using interactive inputs (e.g., clicks or text prompts). In our implementation, any unsegmented (shown as black) region is treated as part of an independent background object.

For synthetic scenes, we reduced the grouping to two motion categories because of the smaller number of objects in the scene. fig. 3.6 compares the original SAMTrack masks (second row) to the solved motion groups (third row). Notably, in the *Occlusion* scene, the algorithm successfully isolates the moving cube into a distinct motion group. We observed similar success in the simpler *Sliding Cube* and *Arcing Cube* scenarios fig. 3.7.

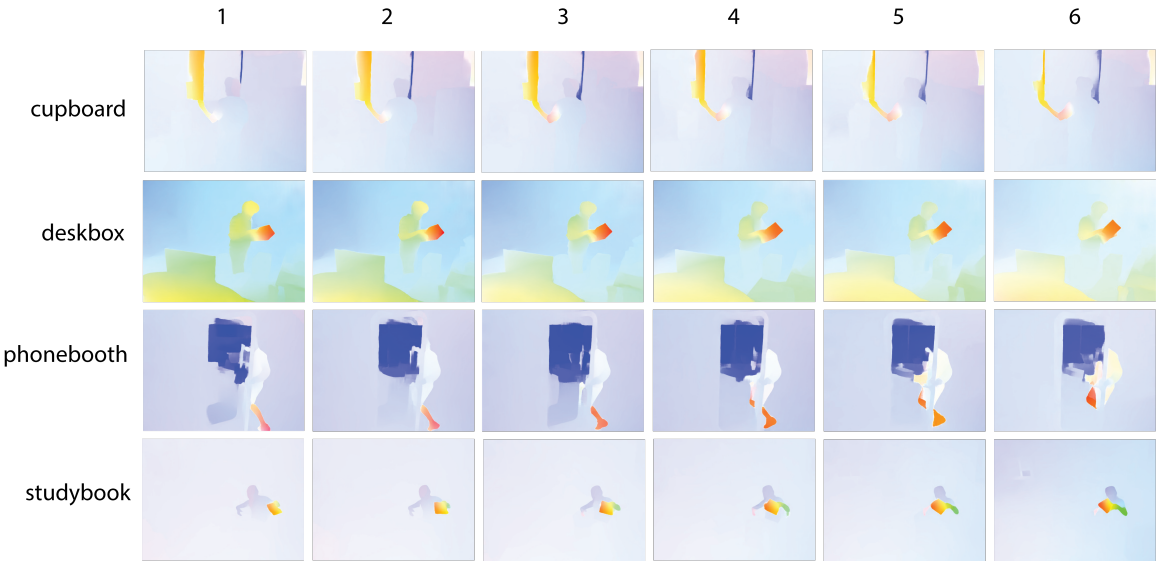

Figure 3.2: **Optical flow on the first six frames of four scenes *Cupboard*, *Deskbox*, *Phonebooth*, *Studybook***; input to segmentation method 1

One possible explanation for the disparity in our results between real and synthetic scenes, as well as between our results on real scenes and those reported by Huang et al. [8], is the differences in motion complexity and object segmentation granularity. In the TöRF real-world scenes, a large number of objects are segmented, many of which exhibit complex and varied motion patterns. This significantly increases the difficulty of computing an accurate motion similarity matrix. Meanwhile, Huang et al. segments only prominent objects in the scene, seemingly at coarser granularity. Similarly, our synthetic scenes involve far fewer segmented objects, simplifying the motion grouping process.

3.2.2 Segmentation in scene reconstruction qualitative results

Despite our attempts, we were unable to achieve fully automatic segmentation using SAMTrack. As a workaround, we manually employed SAMTrack's click-and-track functionality to obtain segmentation masks for the moving objects. These masks were then processed to become segmentation inputs for the TotFotG pipeline.

fig. 3.6 visualizes the rendered 'motion' attribute of the Gaussians, where blue indicates static Gaussians and red indicates dynamic ones. The model successfully reconstructed the motion attributes with reasonable accuracy, though some noise is present. Notably, we consistently observed the existence of a cluster of static ghost Gaussians across various scenes. For instance, in *Occlusion*, a cluster appears at the bottom left corner of the rendered depth frames.

We hypothesize that ghost Gaussians are the results of our pipeline design. In TotFotG, the canonical frame is trained with all frames for a number of warm-up iterations, thus the canonical frame is effectively the average of all frames. Consequently, Gaussians corresponding to moving

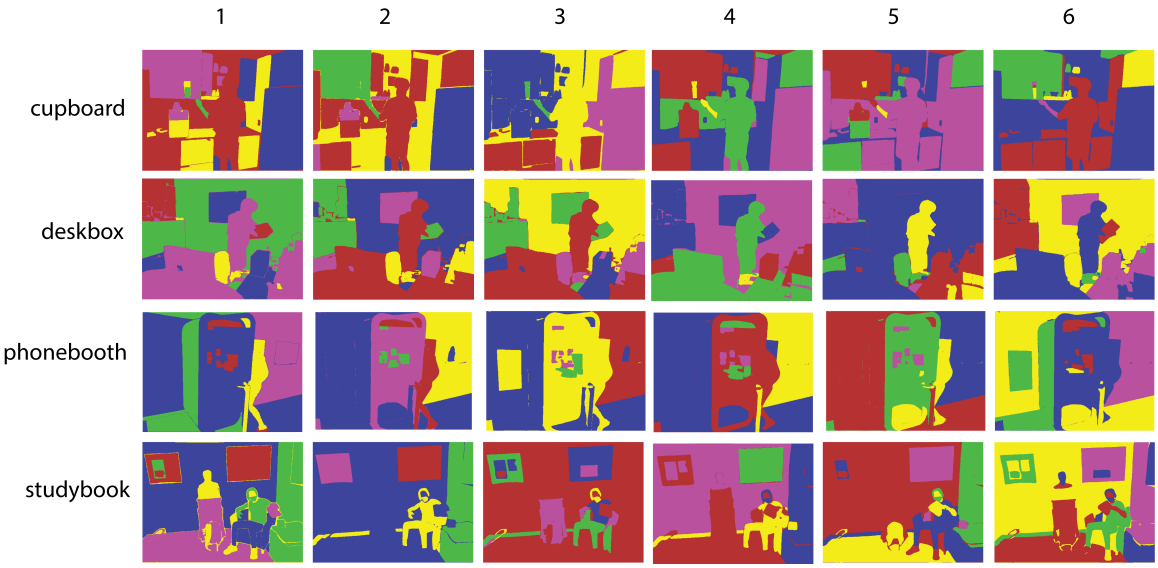

Figure 3.3: **Grouping results on the first six frames of four scenes *Cupboard*, *Deskbox*, *Phonebooth*, *Studybook*.** In this figure, objects are grouped into five different groups, represented by five different colors. Grouping results are largely inconsistent. Objects with similar motions are expected to be in the same group, but groupings change every frame.

Figure 3.4: **Grouping results on *Deskbox* with two motion groups**. Grouping results are consistent, but the two dynamic objects (person and box) are grouped with other static objects. The algorithm might have misinterpreted camera movement as object movement and misclassified foreground objects as dynamic.

objects in the canonical frame are given the average motion mask value, which is static (blue). These Gaussians then become ghost Gaussians. An alternative explanation is that the added complexity by introducing motion masks negatively affects the model's ability to find a local minima during optimization.

Nevertheless, the model demonstrates robustness to poor segmentation inputs. For example, in frame 4 of fig. 3.6, the rendered 'motion' shows the existence of part of the cube to the right of the stick, which is present in ground truth depth but not the motion mask.

fig. 3.8 presents additional results for the *Arcing Cube* and *Sliding Cube* synthetic scenes. As with *Occlusion*, ghost static Gaussians exist in both scenes as highlighted by yellow bounding boxes.

Despite the shortcomings, our method improved reconstruction quality in scene *Fan*, which poses a significant challenge for the standard TotFotG model. As shown in fig. 3.9, TotFotG was unable to reconstruct the non-linear fan motion correctly due to the limitations of the deformation MLP.

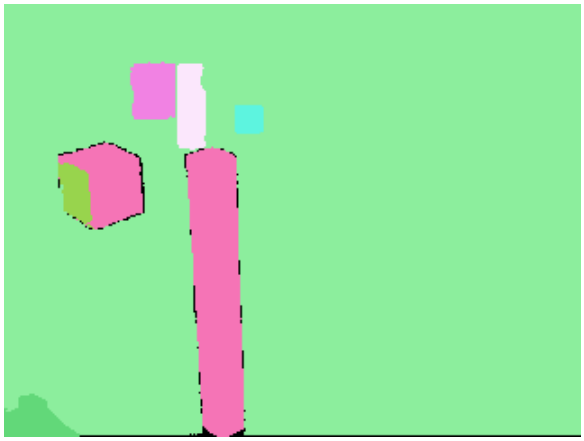

Figure 3.5: **SAMTrack segmenting everything for the first frame of scene *Occlusion*.** SAMTrack falsely segments the background into smaller objects, and the cube is segmented into 2 distinct objects. There are also edges around the objects that are not considered as part of any objects.

Table 3.1: **Metrics on synthetic scenes.** Average Dice Similarity Coefficient (DSC) and Mean Absolute Error (MAE) metrics between ground truth motion mask and rendered motion mask across all frames. Data used to calculate these metrics are results achieved after 30,000 iterations.

Metric	Occlusion	Arcing Cube	Sliding Cube
DSC*	0.965257	0.944291	0.972872
DSC**	0.533207	0.501043	0.555408
MAE	0.007117	0.005317	0.008959

However, with our approach, the motion mask helps guide the MLP to better approximate the fan's motion. Specifically, the fan's wings and its motion trajectory are more distinguishable. Some artifacts still remain, such as the black stroke in our motion mask render, or misshaped fan wings in some frames that are potentially caused by aliasing in the input data.

In addition, our method's use of static/dynamic separation and the existing semantic grouping of Gaussians should allow for straightforward isolation of moving objects. Specifically, dynamic elements can be extracted by selecting Gaussians with a 'motion' vector value of $vector3(1, 0, 0)$. However, in practice, extracting the moving object requires additional work. fig. 3.10 showcases results for 3D segmentation of the moving fan and the static scene. For the purpose of this extraction, a Gaussian is considered dynamic if its 'motion' value $RGB(m_{jr}, m_{jg}, m_{jb})$ satisfies $m_{jb} > 0.5$. While the static background depth and 'motion' are as expected except for some artifacts, results for the dynamic fan shows unwanted background Gaussians that are normally occluded by static Gaussians. To achieve a clean set of Gaussians for the fan, the scene's maximum rendered depth is truncated.

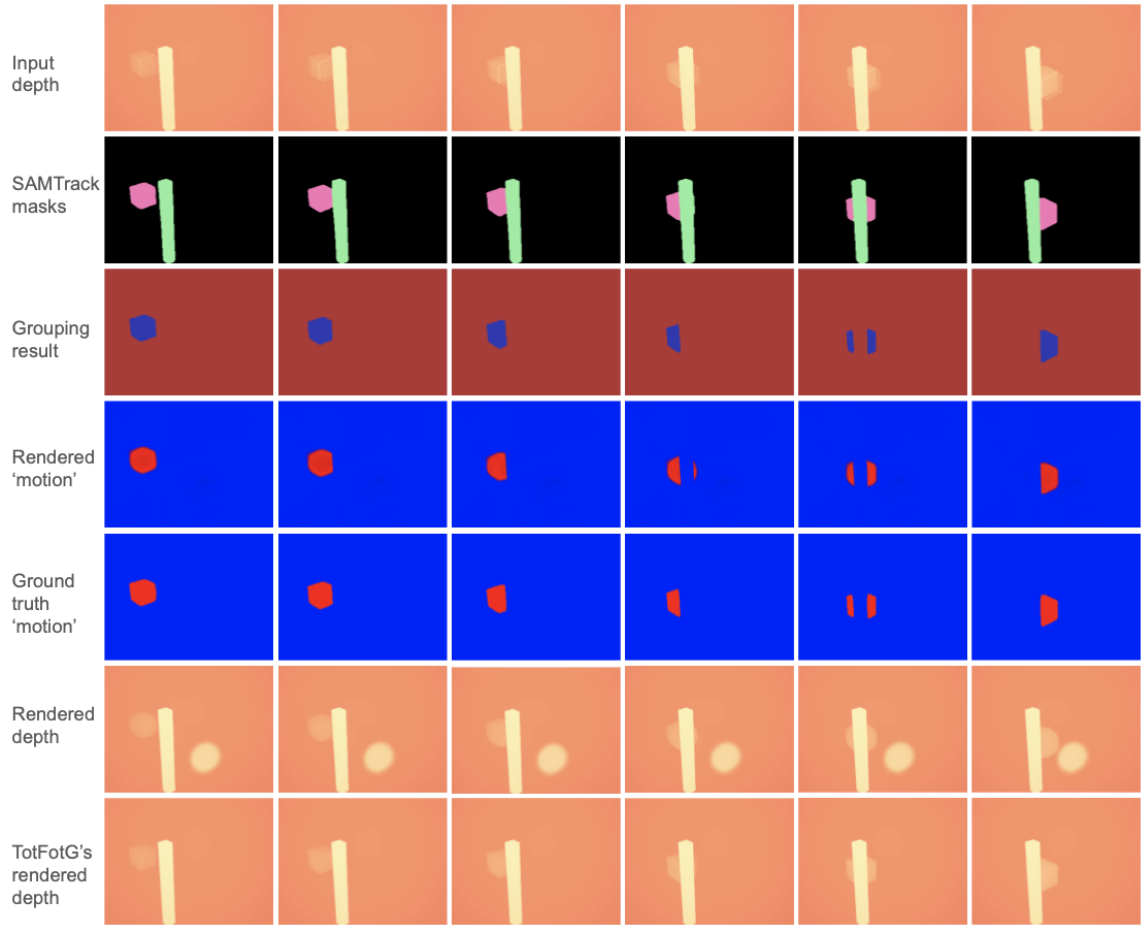

Figure 3.6: **Ground truth 'motion', depth, and pipeline results for scene *Occlusion*, along with TotFotG's rendered depth. These are results after 30000 iterations.** *Note: frames are all raw quads type 0.* Our method results in ghost Gaussians, despite overall good reconstruction. This is probably a relic of the canonical frame.

3.2.3 Quantitative results

Quantitative comparisons between our rendered motion masks and the ground truth are reported in table 3.1. Even though the conventional values for segmentation masks are binary, our predictions are continuous values ranging from 0 to 1. We therefore evaluated similarity using two variants of the Dice Similarity Coefficient (DSC):

$$\text{DSC} = \frac{\text{rendered} \cup \text{ground truth}}{\text{image size}} \tag{3.1}$$

We define DSC* to consider two values to be equivalent if $|\text{value}_1 - \text{value}_2| \leq 0.01$, while DSC** has a stricter threshold of $|\text{value}_1 - \text{value}_2| \leq 0.001$.

Overall, our predicted motion masks achieve high DSC values, indicating generally accurate pixel-wise predictions. However, the non-zero Mean Absolute Error (MAE) values reveals the existence

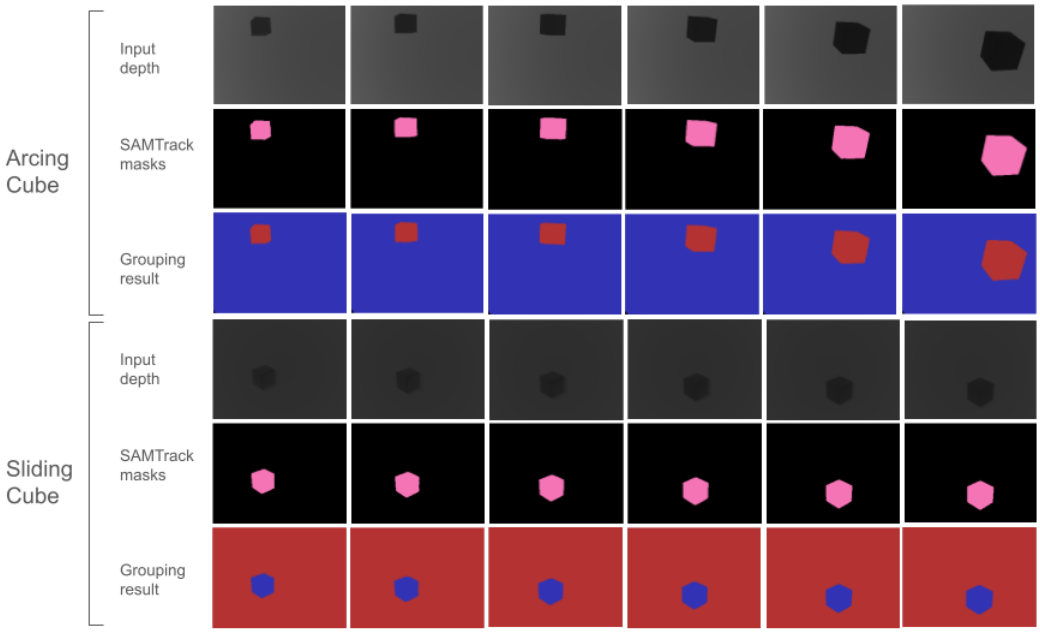

Figure 3.7: **Input depth, SAMTrack masks, and grouping results on scenes *Arcing Cube* and *Sliding Cube***. *Note: frames are all raw quads type 0.* The algorithm is able to produce accurate results with these simple scenes.

of high-error regions — likely corresponding to the ghost Gaussians identified in the qualitative analysis.

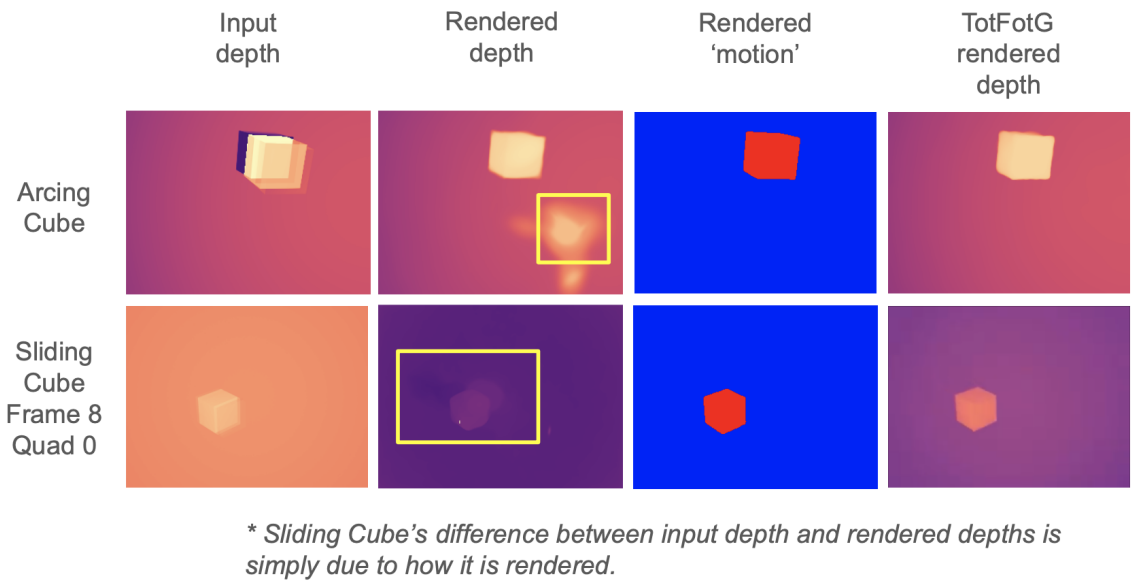

Figure 3.8: **Ground truth depth and rendered results for scenes *Arcing Cube* and *Sliding Cube*, with TöRF's rendered depth. These are results after 60000 iterations**. Boxes show ghost Gaussians. Surprisingly, our method outperforms TotFotG in Sliding Cube, potentially due to high depth values.

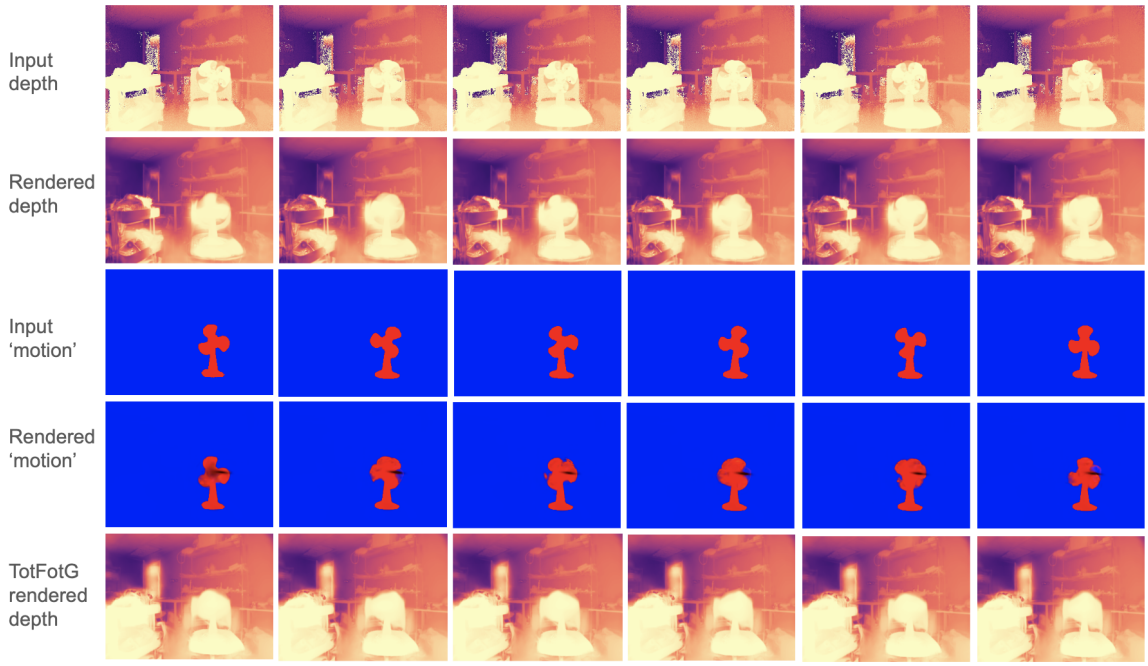

Figure 3.9: **Ground truth 'motion', depth, and reconstruction results for real-life scene *Fan*, along with TotFotG's rendered depth**. *Note: frames are all raw quads type 0.* Our method improves reconstruction of the fan's motion

Figure 3.10: **Dynamic and static Gaussians separately rendered in the *Fan* scene.** To segment moving 3D Gaussians from the static Gaussians, some additional work is needed to remove background Gaussians.

Chapter 4

Conclusion

4.1 Conclusion

We revisit our research questions:

- Does semantic motion segmentation help to improve the quality of monocular dynamic RGB and ToF reconstruction?
- Does semantic motion segmentation improve editability? Specifically, how easy it is to separate Gaussians representing dynamic objects?

We conclude that while our current approach to use semantic motion segmentation in scene reconstruction might add noise, it can be useful in certain settings as demonstrated in our experiments on the *Fan* scene. Our work is therefore a proof of concept for future research. Regarding the segmentation, the current pipeline is not viable due to two reasons: firstly, its inability to deal with a big, complex group of objects; and secondly, the unreliability of SAMTrack masks.

We also conclude that successful semantic motion segmentation can improve editability. Our method design makes it possible to segment the moving object from the scene despite the need for post-processing. While we only focused on segmenting a single dynamic object when training our Gaussians, this same philosophy can be applied to individual object segmentation (as demonstrated in this section) and to the segmentation multiple moving objects.

4.2 Future work

Further research includes more understanding and experimentation to remove noises from our reconstruction, and additional exploration of better methods for segmentation masks.

Appendix A

More Segmentation Methods

In addition to the segmentation method described above, two other methods were also explored. These two methods unfortunately did not have any results due to technical reasons like environment issues or code base age, but they are included in this thesis purely as suggestions for future research.

A.1 Segmentation method 2

Our method 2 uses DELTA [16], which takes in a sequence of RGBD frames and outputs dense motion vectors for every pixel in the first input frames. We directly run DELTA on every RGBD frame pair F_i & F_{i+1} (forward flow) and F_{i+1} & F_i (backward flow) to extract a 3D movement vector $\overrightarrow{v}$ for every pixel in frame F_i (using forward flow) and frame F_{i+1} (using backward flow). Whether a pixel in frame F_i is static or dynamic is dependent on its $|\overrightarrow{v}|$.

Because this static/dynamic separation is not semantic, we can add semantic understanding through the use of SAMTrack. A segmented object is considered dynamic when at least a percentage of its pixels are dynamic in any frame. A potential weakness of this method is the number of tunable hyperparameters.

A.2 Segmentation method 3

Our method 3 uses Towards Segmenting Anything that Moves (TSAtM) [5], which takes in a sequence of RGB frames and their optical flow as inputs and outputs moving objects masks, even when data is captured with a moving camera. This method is the most straightforward as it involves minimal pre-processing and post-processing.

However, segmentation results by TSAtM are highly unstable across frames, which means the resulting masks might have a bad impact on our reconstruction pipeline.

Bibliography

[1] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole. Törf: Time-of-flight radiance fields for dynamic scene view synthesis. *Advances in neural information processing systems*, 34:26289–26301, 2021.

[2] Minyu Chen, Weixing Xie, Zhiyang Deng, Chenglong Shi, Ruoxuan Gou, Yingxuan Chen, and Xiao Dong. Real-time gaussian splatting for dynamic reconstruction in stationary monocular cameras. *2024 IEEE 4th International Conference on Software Engineering and Artificial Intelligence (SEAI)*, pages 33–37, 2024.

[3] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss. Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data. *IEEE Robotics and Automation Letters (RA-L)*, 6:6529–6536, 2021.

[4] Yangming Cheng, Liulei Li, Yuanyou Xu, Xiaodi Li, Zongxin Yang, Wenguan Wang, and Yi Yang. Segment and track anything. *arXiv preprint arXiv:2305.06558*, 2023.

[5] Achal Dave, Pavel Tokmakov, and Deva Ramanan. Towards segmenting everything that moves. *CoRR*, abs/1902.03715, 2019.

[6] Miles Hansard, Seungkyu Lee, Ouk Choi, and Radu Horaud. *Time-of-Flight Cameras: Principles, Methods and Applications*. Springer Publishing Company, Incorporated, 2012.

[7] Bingxin Hou, Ying Liu, Nam Ling, Yongxiong Ren, and Lingzhi Liu. A survey of efficient deep learning models for moving object segmentation. *APSIPA Transactions on Signal and Information Processing*, 12, 01 2023.

[8] Yuxiang Huang, Yuhao Chen, and John S. Zelek. Dense Monocular Motion Segmentation Using Optical Flow and Pseudo Depth Map: A Zero-Shot Approach. *Proceedings of the Conference on Robots and Vision*, may 28 2024. https://crv.pubpub.org/pub/iunjzl55.

[9] Feng JIANG, Jiao LIU, and Jiya TIAN. Segmentation of motion objects in video frames using deep learning. *International Journal of Advanced Computer Science and Applications*, 14(9), 2023.

[10] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

[11] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[12] Runfeng Li, Mikhail Okunev, Zixuan Guo, Anh Duong, Christian Richardt, Matthew O'Toole, and James Tompkin. Time of the flight of the gaussians: Fast and accurate dynamic time-of-flight radiance fields. *preprint*, 2024.

[13] Yiqing Liang, Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. Gaufre: Gaussian deformation fields for real-time dynamic novel view synthesis, 2024.

[14] Yiqing Liang, Eliot Laidlaw, Alexander Meyerowitz, Srinath Sridhar, and James Tompkin. Semantic attention flow fields for monocular dynamic scene decomposition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023.

[15] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[16] Tuan Duc Ngo, Peiye Zhuang, Chuang Gan, Evangelos Kalogerakis, Sergey Tulyakov, Hsin-Ying Lee, and Chaoyang Wang. Delta: Dense efficient long-range 3d tracking for any video. *arXiv preprint arXiv:2410.24211*, 2024.

[17] Mikhail Okunev, Marc Mapeke, Benjamin Attal, Christian Richardt, Matthew O'Toole, and James Tompkin. Flowed time of flight radiance fields. In *ECCV*, 2024.

[18] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

[19] Johan Vertens, Abhinav Valada, and Wolfram Burgard. Smsnet: Semantic motion segmentation using deep convolutional neural networks. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, 2017.

[20] Youcai Zhang, Xinyu Huang, Jinyu Ma, Zhaoyang Li, Zhaochuan Luo, Yanchun Xie, Yuzhuo Qin, Tong Luo, Yaqian Li, Shilong Liu, et al. Recognize anything: A strong image tagging model. *arXiv preprint arXiv:2306.03514*, 2023.