Mixed Reality Synthetic Data Generation

Grace Su[1], Khiem Vuong[2], N. Dinesh Reddy[2], Srinivasa Narasimhan[2]

***Abstract*— Synthetic data generation augments existing vision datasets and consequently helps train more robust computer vision models. However, synthetic image generation techniques proposed by prior works still face limitations in generating photorealistic data, maintaining low computation costs, and granting fine control over scene generation parameters. In particular, synthetic data generation would be especially useful for training deep learning models for traffic analysis tasks. Therefore, we propose a photorealistic synthetic road scene generation method that inserts rendered 3D objects into a real 2D photo. We first estimate the ground plane equation, camera parameters, possible vehicle trajectories, and environment illumination map from the road scene photo. Then, these scene parameters are used to render the 3D objects in a physically-based renderer. Finally, we compose the rendered object smoothly into the road scene. Simultaneously, the renderer can generate precise depth maps. Our "mixed reality" approach's results are higher resolution and more photorealistic compared to similar previous works while addressing their limitations. Thus, our approach can generate high quality synthetic images and ground truth labels for a variety of computer vision tasks. In future work, we plan to evaluate whether our synthetic data and ground truth labels can improve deep neural network performance on challenging tasks like amodal segmentation. Code for the road scene generation method is available at https://github.com/graceduansu/mixed_reality_synthetic_data_generation.**

***Index Terms*— Deep Learning, Visual Perception, Object Detection, Segmentation and Categorization, Computer Vision for Automation, Synthetic Data Generation**

I. INTRODUCTION

One of the most ubiquitous challenges in developing computer vision models is obtaining realistic, accurately-labeled, diverse, and large computer vision datasets. Firstly, it is difficult, time-consuming, and expensive to acquire and label real-world data. Secondly, real-world image data is often characterized by a long tail distribution, where only a minority of different scenarios comprise the majority of collected images. This means the dataset may not encompass the full range of possible nuances and variations in each image. Synthetic data generation addresses these issues by automatically computing new images that imitate the data distributions found in existing images of the real world. The process also allows users to configure and quickly generate more diverse scenarios that are difficult to obtain from the real world while obtaining high-accuracy ground truth labels. Then, the synthetic images can augment existing datasets and consequently train more robust computer vision models. For instance, creating object segmentation datasets often requires a time-consuming process where human annotators must select the regions of pixels for each segmentation mask by hand. However, if a synthetic image is generated by a computer, the ground-truth, pixel-accurate segmentation masks for each object can also be easily accessed during the generation process. Thus, a method that generates image data that is highly faithful to real image data could be useful for improving the training of many different computer vision tasks.

Additionally, when curating a computer vision dataset, we cannot rely on fully real image data because the images must be labeled by humans and humans cannot consistently produce pixel-accurate annotations. On the other hand, we cannot use fully synthetic images because machine learning models trained on such images will encounter significant domain adaptation problems (the sim-to-real gap) when they are tested on real-world images. Therefore, a "mixed reality" dataset that combines elements of real-world and synthetic image data would balance the advantages and drawbacks of both sources.

In the current literature, a number of works have generated synthetic object segmentation datasets by inserting 3D rendered objects into real-world scenes, but few attempt to take advantage of the 3D ground-truth information to generate labels for other vision tasks. In particular, synthetic data and 3D ground truth label generation would be especially valuable for vision tasks that predict 3D world information from 2D image data. One such application area is training deep learning models for traffic analysis tasks since it is important to obtain accurate data annotations and predict rare traffic patterns.

In this paper, we begin to investigate whether a mixed reality method can generate synthetic, realistic road scenes and facilitate the training of computer vision models. We chose to focus on road scene generation in order to assist training of traffic analysis-related computer vision tasks like object segmentation, 3D pose estimation, object tracking, anomaly detection, etc. Section II gives background information and reviews previous work related to synthetic image data generation, including 2D image composition, neural rendering techniques, and 3D object insertion. Section III discusses the proposed 3D object insertion and composition automated pipeline. Section IV presents and evaluates our object insertion results. Section V concludes the paper, describes potential applications, and outlines future work.

[1] Grace Su is with the Computer Science Department, Columbia University, New York, NY., USA. g.su@columbia.edu

[2] Khiem Vuong, N. Dinesh Reddy, and Srinivasa Narasimhan are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA., USA. {kvuong,dnarapur,srinivas}@andrew.cmu.edu

II. RELATED WORK

A. 2D Image Composition

2D image composition methods typically "cut and paste" new objects onto desired background images, then blend the new objects into the background to make the resulting composite image look more realistic. Because this image generation method cuts and pastes images of real objects, it is closer to using fully real image data. This technique is also relatively simple and thus easier to scale. But as [1] describes, the resulting composite images are frequently unrealistic because of "appearance inconsistency (e.g., incompatible illumination), geometry inconsistency (e.g., unreasonable size), and semantic inconsistency (e.g., mismatched semantic context)." Many works address these issues by training neural networks, especially GANs (generative adversarial networks), to adjust for these inconsistencies. For instance, [2] enables object and texture editing by training a GAN to replace and blend objects. [3] also uses GAN-based models to remove and insert objects and their shadows. However, neural networks trained to blend "pasted" objects still struggle to produce consistent, photorealistic results. The networks do not always learn and apply all physical rules of photos like how perspective affects perceived object size, how occluding objects cast shadows, etc. Additionally, 2D image composition methods have no way to access 3D ground-truth information or generate such data annotations.

B. Neural Rendering Techniques

Neural rendering techniques train neural networks to learn a scene's neural radiance field (NeRF) representation and therefore produce novel views of the scene. These approaches can achieve photorealistic results by representing scenes using implicit fields of volume density and view-dependent color. Many neural rendering techniques also encode the entire scene as a whole. To ensure that the NeRF representation allows for object-level editing, [4] designs an architecture that encodes individual object information. [5] also learns object-level representations by proposing a neural rendering approach that observes a scene video, then decomposes the scene into scene graphs. These works show that NeRFs can be extended to learn object-level representations and enable object manipulation. However, there still exist several cons of using NeRFs for novel view synthesis:

- Low-interpretability and editability for dataset parameter control
- An inherent lack of 3D ground-truth information
- Higher computation costs when learning the NeRF

C. 3D Object Insertion

To generate a high-fidelity image dataset, it is advantageous to render and insert 3D, physically accurate models of objects into existing real-world backgrounds. Then, the rendering process computes the correct scene geometry, reflections, based on user-selected scene parameters, and outputs realistic images. Thus, 3D object insertion is a mixed reality approach. For example, [6] uses Blender's Cycles renderer and post-processing workflow to photorealistically render and insert 3D cars. Their results suggest that object detection and instance segmentation models trained on augmented imagery generalize better than those only trained on synthetic data or those trained on limited amounts of annotated real data. Realism of the background image also significantly affects performance. However, while their pipeline estimates the road plane and camera pose for each background image, they only use background images captured by driving cars. This means that they do not use other viewpoints like those from traffic cameras. They use a fixed set of 3D car models, locations, and environment maps to augment real street scene datasets. Their synthetic dataset and code are also not released. In addition, they only evaluate their dataset on 2D tasks (segmentation and detection) and lack benchmarks for 3D vision tasks.

Recently, Chen et al. [7] further leverages available real world data by inserting objects that were viewed with similar viewpoints and distance to the camera in its original footage. Then, they reconstruct observed objects as 3D assets and warp them to the novel target view using a differentiable neural renderer. Finally, they train a generative image inpainting synthesis network to do post-composition. When using their method for data augmentation, there are small improvements on semantic segementation performance. However, their augmentation method only uses a single cloudy environment map and does not perform lighting estimation. Their synthetic dataset and code are also not released.

On the other hand, inverse rendering and lighting estimation allows one to obtain a desired background image's scene appearance parameters that can then be used to render inserted objects with the correct lighting and geometry. For example, [8] uses a deep neural network to achieve single-image inverse rendering of indoor scenes. The network simultaneously estimates the scene's depths, normals, spatially-varying albedo, roughness and lighting, thus enabling photorealistic material editing, object insertion, and rendering. However, the estimations for depths, normals, albedo, etc. are not as accurate for outdoor scenes because there is a lack of ground-truth lighting data for outdoor viewpoints.

Overall, current works in 3D object insertion for data augmentation demonstrate small improvements for traditional object segmentation and detection tasks, but do not evaluate 3D object insertion for other vision tasks that requires accurate 3D ground-truth information.

III. METHODS

After reviewing related work in synthetic image data generation, we chose to design a 3D object insertion-based approach for road scene generation. We first obtain the desired background image's scene appearance parameters: the road plane equation, vehicle trajectories, intrinsic and extrinsic camera parameters, and sun direction. Next, we use physics-based rendering to render 3D vehicle models with the obtained scene parameters. We also produce the corresponding, unoccluded depth maps for each vehicle.

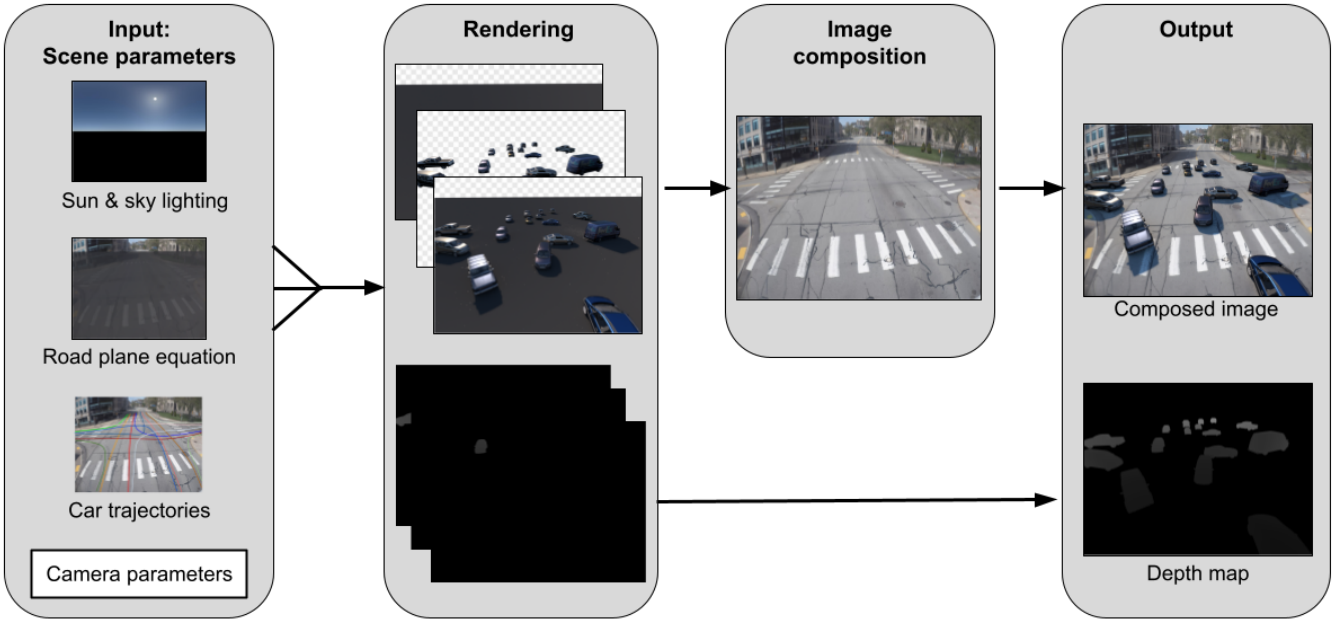

Fig. 1: Our proposed synthetic scene generation method

Finally, we composite the rendered cars onto the desired background using pixel-wise computations as described in [8]. Our synthetic scene generation method is illustrated in Fig. 1.

A. Incorporating Scene Geometry

To incorporate the desired background image's scene geometry into our vehicle renderings, we obtain the road plane equation, possible vehicle trajectories (Fig. 2), and intrinsic and extrinsic parameters of the camera that captured the desired background image. Note that our estimated scene geometry is in metric scale, thus allowing physically accurate renderings of objects.

Given the camera's GPS location, we leverage Google Street View (GSV) [9] to build the scene's geometry at that location. GSV is a street-level imagery database and a rich source of millions of panorama images with wide coverage all over the world. Every panorama image is geo-tagged with accurate GPS coordinates, capturing 360° horizontal and 180° vertical field-of-view with high resolution. We sample multiple panoramas around the desired camera's location inside a radius of 40 meters and use *structure-from-motion* (SfM) [10] to reconstruct the scene. Note that we also geo-registered the *up-to-scale* SfM reconstruction using the provided GPS coordinates of the GSV panoramas. Thus, our final 3D reconstruction of the scene is in *metric scale*.

To obtain the camera's intrinsic and extrinsic parameters, we follow the typical visual localization pipeline by localizing the desired background image (i.e., query image) w.r.t. the 3D reconstruction built with GSV images (i.e., database images). To establish robust 2D-3D correspondences, we follow hloc [11] by using learned feature matching method SuperGlue [12] with SuperPoint [13] features descriptors to match the query image with the database images. Given the 2D-3D correspondences, we perform a bundle adjustment step to retrieve the camera intrinsic and its 6DoF extrinsic parameters. Note that the large number of accurate matches between the query image and the rich GSV database images, produced by SuperPoint and SuperGlue, allows us to robustly recover both intrinsic and extrinsics parameters of the camera.

The road plane equation is estimated by fitting a plane to the set of 3D points whose 2D pixel locations are lying on the *road* obtained from off-the-shelf semantic segmentation method [14]. The possible vehicle trajectories are estimated from the real data by tracking multiple vehicles during a long period of time in 2D, which is then lifted to 3D using the road plane estimated above. We then perform spline-fitting followed by hierarchical clustering [15], where the average direction of each cluster is considered a possible vehicle trajectory. Additionally, when placing each 3D vehicle model into the scene along the vehicle trajectories, we employ collision checking between all models' 3D bounding boxes to ensure no models intersect each other in an unrealistic manner.

By incorporating the physically accurate scene geometry, as long as the inserted objects have correct metric scale, we are able to render geometrically accurate, scale-consistent road scenes (Fig. 3) and avoid inconsistencies such as unreasonable object sizes, incorrect distortions, or occlusions in our generated road scenes.

B. Lighting Estimation

We also estimate the environment map of the desired background image. After obtaining the time, date, and GPS coordinates for when and where the road scene photo was captured, our rendering software, Mitsuba [16], computes the sun's direction and generates environment map using sun

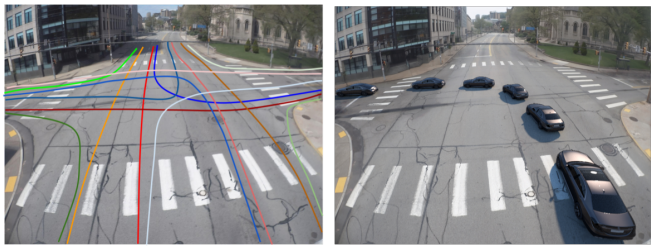

Fig. 2: All possible vehicle trajectories for this example road scene photo are visualized on the left. An illustration of one possible vehicle trajectory is generated on the right.

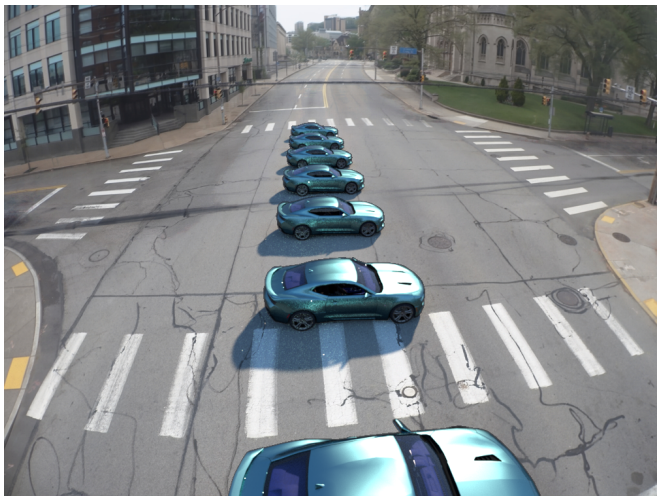

Fig. 3: Demonstration of our method's geometry, perspective, and size consistency as a result of using an estimated road plane equation and camera parameters. The same car model has been rendered and inserted at constantly increasing distances from the camera.

and sky illumination models. To avoid modeling the sun as a point light source and ensure that specular reflections are appropriately sized, we set Mitsuba's sun radius parameter to 5.

C. Physically-Based Rendering

We chose to render 3D objects using Mitsuba 0.5.0 [16] because it accurately models the physics of light scattering and can easily provide the corresponding, high resolution, ground truth rendering data such as depth maps, albedo maps, surface normals, and 3D coordinates in the world space. Other physically-based renderers also exist [8], [17], [18]. While these renderers are optimized and GPU-accelerated to be much faster than Mitsuba 0.5.0, their drawbacks include not being open-source, requiring RTX GPUs, and/or lacking crucial options that Mitsuba 0.5.0 provides. The specific options we require for our method's current implementation are the sun and sky illumination modeling plugin and the option to hide directly visible emitters.

Another vital part of physically-based rendering is incorporating the appropriate surface-scattering models for each type of material present in the scene. To achieve this, we first curate a set of 11 high quality 3D vehicle models covering 5 categories (SUV, sedan, mini-van, van, pickup truck). Each 3D models must have a high polygon count and meaningful material names in its material file. Then, based on the material names, we can map each of the material definitions to the appropriate Mitsuba BRDF (bidirectional reflectance distribution function) surface scattering model. Our material mapping rules are summarized in Fig. 4.

Material Names	Mitsuba Surface Scattering Models
"glass", "windshield"	Thin dielectric material
"plastic", "headlight", "indicator"	Smooth plastic material
"car body", "chrome", "metal", "silver"	Rough conductor material with Smooth dielectric coating
"tire", "rubber", "wheel"	Rough diffuse material
"interior"	Modified phong BRDF

Fig. 4: A summary of our material mapping rules. For each material name, we search the name for keywords and related substrings (left column), then match it to the appropriate Mitsuba surface scattering model (right column).

D. Image Composition

We use an image composition method described in [8] to insert the 3D objects while blending their shadows into the background image. We render the following images for each road scene to obtain the necessary images for final image composition and depth maps:

- I_{all}: Road plane and 3D car models
- I_{obj}: 3D car models only
- I_{pl}: Road plane only
- Individual depth map for each 3D car model

All images are rendered with Mitsuba's options to hide directly visible emitters (in our case, the environment map) and enable the image's alpha channel. Then, the masks M_{all} and M_{obj} for I_{all} and M_{obj}, respectively, are easily obtained.

To remove potential pixel artifacts on object and plane edges, we erode the boundaries of regions of foreground pixels. Next, we calculate the edges, or contours, of the object mask M_{obj} and apply Gaussian blurring. Then, we alpha blend the blurred contours with the original contours.

Finally, to compute the new, composited image I_{new}, we calculate the pixel values in the object region by pixel-wise multiplication (indicated by $\odot$):

$$I_{new} \odot M_{obj} = I_{all} \odot M_{obj} \tag{1}$$

For the pixel values in the plane region:

$$I_{new} \odot (M_{all} - M_{obj}) = I \odot \frac{I_{all}}{I_{pl}} \odot (M_{all} - M_{obj}) \tag{2}$$

IV. RESULTS

We present image examples of results produced from our synthetic road scene generation method (Fig. 5, Fig. 6).

Note in each generated result, the rendered cars respect the scene lighting and geometry while exhibiting physically accurate reflections. Additionally, there are very few visible differences between the synthetic images and real world photos.

Fig. 5: An image example produced using our augmented reality-based data generation method. All cars in the image were rendered and inserted.

Fig. 6: An example of a real traffic camera photo, with one rendered vehicle inserted for comparison (the black SUV second from the bottom of the image)

For our rendering settings, we chose a volumetric path tracer in order to handle the relatively glossy car materials. We chose to render 1000×750 images at a sample count of 32 with a maximum path depth of 4 to balance the tradeoff between higher quality results and longer rendering times. In addition, we decided to render a random number between 10 and 20 vehicles, inclusive, to provide many vehicle examples in one image while maintaining reasonable rendering times.

Using these settings, we found that on one machine with 16 CPU cores, the rendering times for a single scene roughly vary between 200 and 400 seconds, with times largely being determined by the number of vehicles in the image. However, because one Mitsuba process will use at most 2GB of RAM for our method, multiple dataset generation and rendering processes can be launched simultaneously to produce a large synthetic image dataset.

V. CONCLUSION AND FUTURE WORK

The results demonstrate that our augmented reality-based method for synthetic road scene generation produces more photorealistic results compared to previous 3D object insertion works. Additionally, the results suggest that our approach can be readily used to generate road scene images and precise ground truth labels for computer vision tasks like object segmentation, construction zone detection, object tracking, 3D pose estimation, and more.

In future work, we plan to evaluate whether our synthetic data and precise depth maps (Fig. 7) improve training and performance of amodal segmentation models, which aim to predict the object's full segmentation mask despite visual occlusions [19]. We also plan to generate sets of traffic scenes that contain rare objects (Fig. 8) and object configurations (Fig. 9). We will evaluate whether adding these scenes helps improve object detection training and robustness.

Overall, these steps will allow us to discover how synthetic data generation can potentially augment existing vision datasets and train more robust computer vision models.

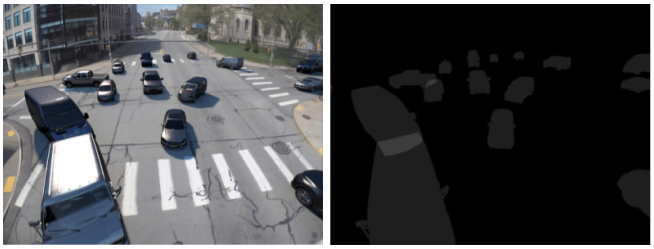

Fig. 7: We can compute the unoccluded segmentation masks from our output depth map and use them to train amodal segmentation models.

Fig. 8: Examples of rare objects: ambulance, firetruck, construction vehicle

Fig. 9: Example of a rare object configuration: Anomalous traffic pattern

ACKNOWLEDGMENT

This material is based upon work supported in parts by the National Science Foundation under Grants CCF-1730147 and CNS-2038612, and the Columbia University Egleston Scholars Program Stipend. Special thanks to the CMU Illumination and Imaging Lab and the Robotics Institute Summer Scholars program for their mentorship and support.

REFERENCES

[1] L. Niu, W. Cong, L. Liu, Y. Hong, B. Zhang, J. Liang, and L. Zhang, "Making images real again: A comprehensive survey on deep image composition," 2021. [Online]. Available: https://arxiv.org/abs/2106.14490

[2] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[3] Y. Wang, A. Liu, R. Tucker, J. Wu, B. L. Curless, S. M. Seitz, and N. Snavely, "Repopulating street scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[4] B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, and Z. Cui, "Learning object-compositional neural radiance field for editable scene rendering," in *International Conference on Computer Vision (ICCV)*, October 2021.

[5] J. Ost, F. Mannan, N. Thuerey, J. Knodt, and F. Heide, "Neural scene graphs for dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 2856–2865.

[6] H. Abu Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *Int. J. Comput. Vision*, vol. 126, no. 9, p. 961–972, September 2018. [Online]. Available: https://doi.org/10.1007/s11263-018-1070-x

[7] Y. Chen, F. Rong, S. Duggal, S. Wang, X. Yan, S. Manivasagam, S. Xue, E. Yumer, and R. Urtasun, "Geosim: Realistic video simulation via geometry-aware composition for self-driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 7230–7240.

[8] Z. Li, M. Shafiei, R. Ramamoorthi, K. Sunkavalli, and M. Chandraker, "Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2475–2484.

[9] Google, "Google Street View," https://www.google.com/streetview/.

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

[11] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *CVPR*, 2019.

[12] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.

[13] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.

[14] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," 2022.

[15] F. Li, N. D. Reddy, X. Chen, and S. G. Narasimhan, "Traffic4d: Single view reconstruction of repetitious activity using longitudinal self-supervision," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 1385–1392.

[16] W. Jakob, "Mitsuba renderer," 2010, http://www.mitsuba-renderer.org.

[17] "Marmoset Toolbag 4 - 3D Rendering, Texturing, & Baking Tools." [Online]. Available: https://marmoset.co/toolbag/

[18] "Omniverse Platform for Virtual Collaboration." [Online]. Available: https://www.nvidia.com/en-us/omniverse/

[19] N. D. Reddy, R. Tamburo, and S. G. Narasimhan, "Walt: Watch and learn 2d amodal representation from time-lapse imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 9356–9366.