WASP Software Engineering Course Module Assignment

Jiarong Gong

August 25, 2025

1 Introduction

My research is centered on compressive radiance fields, with a current emphasis on 3D Gaussian Splatting (3DGS). Radiance field methods aims to capture and reproduce the full light transport in a scene, enabling photorealistic rendering from arbitrary viewpoints. However, these models are often memory- and computation-intensive, especially when applied on high-resolution or dynamic scenes.

The ”compressive” aspect of my work targets this challenge by integrating concepts from *compressed sensing* and *dictionary learning* into radiance field representation. In *compressed sensing*, the central idea is that a signal can be reconstructed from far fewer samples than traditionally required, provided it is sparse under some basis or dictionary. Dictionary learning extends this principle by adaptively learning such a basis from data, rather than relying on predefined transforms such as the Discrete Cosine Transform (DCT) or wavelets.

In the context of 3DGS, each scene is represented as a collection of anisotropic Gaussian primitives, each carrying position, covariance, opacity, and spherical harmonics coefficients for view-dependent color. While this representation is already more compact than dense voxel grids, it still becomes heavy when representing large 3D scenes. My research investigates *dictionary learning* and *sparse representation* to further compress the Gaussian parameters – for example, by storing them as sparse coefficients in a learned dictionary and reconstructing them when rendering.

This approach brings together three traditionally separate domains:

- **Computer Graphics** — for scene representation and rendering via Gaussian Splatting.
- **Signal Processing** — for sparsity-driven compression and reconstruction.
- **Machine Learning** — for dictionary learning and parameter optimization.

Using these methods, my aim is to develop an end-to-end framework that maintains visual quality while significantly reducing storage and bandwidth requirements. This technique has many applications, such as VR/AR streaming, autonomous driving simulation, and digitization of cultural heritage.

2 Lecture Principles

My work focuses on memory- and computation-efficient radiance fields by integrating concepts from *compressed sensing*. More specifically for current research, compress GAussian parameters and reconstruct them during render time while preserving perceptual qualities. There are some ideas/concepts from Roberts’ lectures that can relate to my research.

2.1 Behavioral SE

I totally agree that people are not rational [2]. Recently I had a confuse. Why does such a good rendering have low metric values (PSNR, SSIM, Lpips)? These low metric values make me think my model is not as good. Therefore I doubted if my idea is a good way to address the heavy memory footprint problem.

After weeks I had a chance to talk to someone focusing on a similar area. I soon realized that I over-relied on these metrics and neglected the perceptual aspects. These metrics are not always good and direct measure of a good rendring. And they may lead me to the wrong way. So I adopt the idea that consider more from different point of views and evaluate models from at least perceptual effects.

2.2 Science-Engineering loop

This idea is from [3]. I guess this is very common to see in most areas: perform research and experiments to develop a ML model, deploy it in production, analyze the log information, and refine it. This forms into a Science-Engineering loop.

Relevance to my work is as follows.

Gaussian Splatting models takes up much memory during both storage and rendering. After focusing on one research direction, I often search for many materials to see how people address such problems. Then I form into my first draft of the model and perform tests on public datasets. If it does not reach state-of-the-art performance, I analyze the output information and find ways to improve the model like training a better dictionary. Locating the problem and doing research are the science side and performing the model testing is the engineering side. This is a very good practice and proved useful in many scenarios. I would similarly ideas like different dictionary sizes and kinds, to see if it results in better performance.

3 Guest-Lecture Principles

3.1 Requirements should be specified and refined

The idea is from [4]. Over the entire course of my PhD, the content will not keep completely the same with the first version. In the proposal, the Neural Radiance Filed (Nerf) was the first choice. However, with the development and advantages of Gaussian Splatting (GS), a lot of attention is caught. I think maybe it is good to start with GS.

After completing the first stage of the proposed research plan, I read the proposal again and gain some deeper insights. I find a few modifications are needed. When it comes to specific research like GS, it is required to understand the specific flaws of GS. Based on the specifications, we develop a new method. This proposal defines the whole research content of my PhD program and serves like the requirements. It needs to be specified and refined.

3.2 Goals can be further refined

Goals can be further refined into sub-goals and modified due to conflicts between goals [4]. Very similar to my case, a shift from Nerf to GS is needed since GS has several strengths over Nerf. For example, Nerf often needs dozens of hours to train a 3D model and minutes to render a 1080p image. This makes Nerf hard to promise real-time rendering and not a good choice in AR/VR applications. GS, however, often needs half hour to train and promises a real-time rendering with an FPS over 100 while has comparable performance in terms of image quality [6].

Based on what I have achieved so far, within the context of GS, new goals have been set: to improve image quality and to accelerate rendering speed. Additionally, I take some time to investigate extending the GS from static scene to dynamic scene, which is absolutely not included in previous goals.

4 Data Scientists versus Software Engineers

4.1 Agreement on Differences

In chapter 1 of [7], data scientists and software engineers usually have different education backgrounds and focuses. I totally agree with that. From my own experience, what I focus on is the model performance, e.g. the generated image quality (measured in PSNR, SSIM, and Lpips) and rendering speed. To be offered with such a position, people might need a degree in Machine Learning or Deep Learning. Software engineers would focus on the system robustness, security, and so on. And they often have a degree in Computer Science.

4.2 Evolution of Roles

I think these roles will evolve and specialise further. Each research area requires much time and effort to develop. And when a great method comes out, it would spark a surge of research interest. Let's take GS for example. Before [6], people put much attention on Nerf. Since Nerf and its variants often requires much time on both training and rendering, some focus on accelerating Nerf [10, 5]. However after [6], lots of interests are on GS. This example shows that with new breakthroughs, there would be more and more people interesting in such topic and this pushes the research to specialise further.

As mentioned in chapter 2, unicorns are rare to see, which aligns well with my work experience. I used to be a computer vision engineer and have experience working in 3 different-sized companies– startup, pre-IPO growth company, and public company. The whole system needs both software for interaction with PC and vision models for vision tasks. I have never seen people who can handle both software and vision model development.

5 Paper Analysis

5.1 First Paper

The first paper I choose is [8], titled "How Do Model Export Formats Impact the Development of ML-Enabled Systems? A Case Study on Model Integration".

5.1.1 Core Ideas and Their SE Importance

[8] conducted a holistic qualitative case study on the effects of exporting formats on the development of ML-enabled systems. They found that ONNX format is the most efficient one in most cases; for python-based systems, SavedModel and TorchScript are very convinient, but need extra work for other language-based systems; Pickle and Joblib are the most challenging to be integrated into systems.

What kind of exporting format is needed should be proposed in the requirement engineering stage in the first place. As we all know that each module would be somehow dependent on other modules. We cannot completely remove such dependencies. Thus, the export formats of AI models should be specified in the requirements. However, as mentioned in [8], little work can be found to instruct us to how to select exporting formats of AI models. This paper is exactly the one we need.

5.1.2 Relation to My Research

My research is currently focused on GS as described detailedly in the first section Introduction. One of the purposes is to accelerate rendering. Previously in Nerf research, people try to sample a trained Nerf model into an Octree format [10] to greatly accelerate Nerf rendering. This paper, combining with PlenOctree paper, reminds me of how I can do to increase the model rendering speed. I would search for some relevant papers to read as a first step. The paper is one of the not many researches that indicate how export format relates to efficiency.

5.1.3 An example of Integration into a Large AI-Intensive Project

In one of the companies I used to work with, the core business is to manufacture a intelligent machine to detect defects on the surface of wafer products. In order to automatically detect defects, they integrate a ML-enabled software system to operate on the hardwares, to which the core is a computer vision model that takes images from the camera system as input and directly outputs both the images with annotated information and the location and classification of defects.

Among the metrics of measuring how good the product is, the efficiency is one of them. Customers always prefer a machine detecting wafer products within less time. Therefore, as an AI algorithm engineer, picking the right export format is important. e.g., we often choose ONNX and engine. Of course before developing such software system, the export format is specified in the requirement stage.

My current research also takes images as input while the difference is the need of many multi-view images. In addition, our integrated system needs to real-time render images which needs much computation while the system mentioned above just needs a window to display the annotated image.

5.1.4 Adaptation of My Research

For my currently specific research on GS, the trained model actually consists of 3D point coordinates, opacity, and so on. The format is often PLY. The paper might instruct me to try other formats to more quickly read the model into memory to somehow accelerate the rendering a bit.

5.2 Second Paper

The second paper is [9], titled ”Investigating Issues that Lead to Code Technical Debt in Machine Learning Systems”.

5.2.1 Core Ideas and Their SE Importance

Some coding issues that are common to see in SE can also happen in ML workflow, such as Wrong Data Consumption, Wrong Outlier Detection, and so on (listed in Table 3). These issues contribute a lot to Technical Debt that increases the ML system maintenance costs. [9] systematically analyzes literature, interviews engineers, and refines findings. With these findings, practitioners can possibly avoid increasing long-term cost.

5.2.2 Relation to My Research

This paper is highly relevant to my research. The 4 phases (Data collection, Data pre-processing, Model creation & training, and Model evaluation) are often performed by me. Even though there are public datasets like [1] for GS, we sometimes still need to produce some datasets by ourselves, e.g. real scene datasets or synthetic datasets. When producing our own datasets, synthetic datasets for example, we need to model 3D objects,render images, and export camera poses. Then we initialize and train a 3DGS model on the processed datasets. Finally, we evaluate our model. The coding issues accross all 4 phases can happen.

5.2.3 An example of Integration into a Larger AI-Intensive Project

The startup example displayed in section **5.1.3** is a good one to elaborate again. Customers need machines to be not only accurate and efficient in defect detection, but also robust. Regarding the robustness, there is a word called repeatability (at least used in the startup company) to represent it. Repeatability measures how consistently a machine or system can produce the same result under the same conditions. In the wafer defect detection system, we are required to design and train a deep learning model with a repeatability of more than 95%. This specific number can be achieved measured in several ways. For instance, you scan the same wafer multiple times (say 10) and the system outputs the detection results. This results in 10 different numbers of defects. We compute the **std** and **mean** of the 10 numbers. If $(\mathbf{mean} - \mathbf{std})/\mathbf{mean} > 95\%$, we think the system or machine meets the required standards in terms of robustness. Of course the machine itself also has influence on repeatability like the movement of cameras.

For the accuracy and robustness purpose, there are many coding issues we need to consider and address. Finally Table 3 of [9] lists 30 issues. We can first avoid the insufficient data consumption issue. Otherwise the model can overfit to insufficient data and cannot learn good features. This could lead to an unclear classification hyperplane. With the camera shake, each time you scan the wafer and you get slightly different images. The model outputs could be greatly different. Wrong data consumption can also have an effect on the robustness of the detection system. Imagine you confused A with B and you randomly assign labels to samples A and B, how can you expect the model correctly classify those defects? The last issue I would like to discuss is insufficient evaluation metric selection. In order to save the best model to disk, we often set some evaluation metric. If a model performs best on evaluation dataset, then we think it is the best model. However, if we set only PSNR to measure how good a model is, the evaluation metric is often considered insufficient.

In standard 3DGS, PSNR, SSIM, and Lpips are often used to evaluate models' performance. In addition to these metrics, I also need to consider compression ratio. A model that has comparable performance with the sota methods in terms of PSNR, SSIM, Lpips, and compression ratio is what I want most. It does not have to outperform sota methods in all aspects, which is very difficult to achieve.

5.2.4 Adaptation of My Research

All in all, [9] is a good instruction book to avoid TD in my research. I would take some time to compare what I often do in projects with the content of the paper to refine the process of the whole workflow. One modification would be the dependency version management.

6 Search and Screening Process

Firstly I look into the list of accepted papers. Look at the titles. If there are some familiar words like Machine Learning, Deep Learning, Gaussian Splatting, and so on, I downkload those papers. Then I read their abstracts to get a rough understanding of what problems they address so that I can decide if it is useful or relevant to my research topic. Finally I pick the 2 most relevant and useful papers to read and analyze.

6.1 Pitfalls and Mitigations

I met some unfamiliar research areas like LLM and Reinforcement Learning. I simply skipped them if there are some more easy-to-understand options. And a way I often use is to read those citing a paper that I'm familiar with.

6.2 Ethical Considerations

Steps taken to ensure originality is to cite references at appropriate places. Read the references and rephrase in my own words.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022.

[2] Robert Feldt. Behavioral software (ai!?) engineering. PowerPoint presentation, Software Engineering and Cloud Computing, Chalmers University of Technology, 2025. Presented in class, June 2025.

[3] Robert Feldt. Science “vs” engineering, and ml engineering (se4ml). PowerPoint presentation, Software Engineering and Cloud Computing, Chalmers University of Technology, 2025. Presented in class, June 2025.

[4] Julian Frattini. Understand the problem before you build the solution. PowerPoint presentation, Software Engineering and Cloud Computing, Chalmers University of Technology, 2025. Presented in class, June 2025.

[5] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5501–5510, 2022.

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

[7] Christian Kästner et al. Machine learning in production: From models to products. `https://mlip-cmu.github.io/book/`, 2022.

[8] Shreyas Kumar Parida, Ilias Gerostathopoulos, and Justus Bogner. How do model export formats impact the development of ml-enabled systems? a case study on model integration. In *2025 IEEE/ACM 4th International Conference on AI Engineering–Software Engineering for AI (CAIN)*, pages 48–59. IEEE, 2025.

[9] Rodrigo Ximenes, Antonio Pedro Santos Alves, Tatiana Escovedo, Rodrigo Spinola, and Marcos Kalinowski. Investigating issues that lead to code technical debt in machine learning systems. In *2025 IEEE/ACM 4th International Conference on AI Engineering–Software Engineering for AI (CAIN)*, pages 173–183. IEEE, 2025.

[10] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5752–5761, 2021.