MLMI17 Advanced Computer Vision Coursework 2

Instructor: Elliott Wu
Due: Tuesday, 24 March 2026, 12:00 PM (noon)

Submission Guidelines

Your answers should contain an explanation of what you do. You must also give an interpretation of what the results and visualizations you provide mean – why are the results the way they are? Each question should be labelled and answered separately and distinctly. Total combined length of answers must not exceed 5 sides of A4 (excluding the cover page and references), minimum 11pt font, 1 inch margins. Please submit the PDF and all your code in a ZIP file. Please use the code base provided in this GitHub repository to run the experiments: https://github.com/CambridgeCVCourses/CW2. It can take *a few hours* to complete this coursework and you will need to use the HPC, so do not leave it to the last minute.

1 Task 1 – Camera Pose Estimation and Sparse Reconstruction

1.1 Using `COLMAP` [20%]

Capture a video of a scene of your choice and run `COLMAP` [3, 4] using `ns-process-data` provided in `nerfstudio` [5] to estimate camera poses and reconstruct a sparse point cloud.

- You must identify one successful capture and one failure capture. A failure case is defined as either unsuccessful registration or fewer than 5% of frames being registered.
- For the successful case, visualise the estimated camera poses and sparse point cloud using `Viser`, and include at least one visualisation in your report.
- For the failure case, provide a explanation of why you think the reconstruction failed.

Detailed instructions for running `COLMAP` and visualising results are available in the GitHub repository. Two example videos are provided for reference, and `COLMAP` should succeed on both. See Fig. 1 for an example of the visualisation tool.

1.2 Using π^3 [20%]

Next, use π^3 [6] to estimate camera poses and sparse point clouds for the same scenes you captured. An incomplete script is provided that runs π^3 inference and exports the results in the same format as the outputs of `ns-process-data`.

- You are required to implement two small parts of the `write_transforms_json` function inside `pi3_inference.py` that exports π^3 predictions into the coordinate system convention and format used by `nerfstudio`.
- Test whether π^3 works on your videos. Compare the results produced by π^3 with those from `COLMAP`, and summarise the advantages and limitations of the two methods.

2 Task 2 – Novel View Synthesis

2.1 Comparing NeRF and 3DGS [20%]

- Train a NeRF [2] model on the provided `static_scene` using the default `nerfacto` model in `nerfstudio` and the camera poses computed from `COLMAP`.

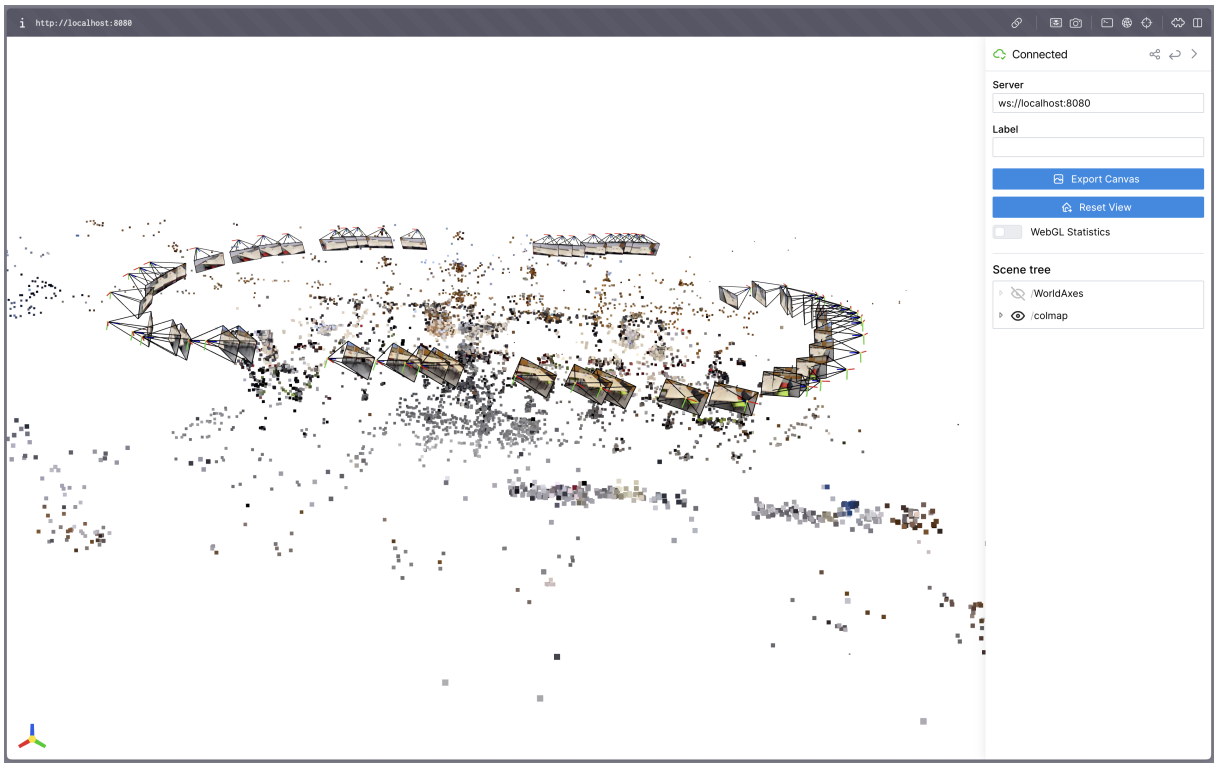

Figure 1: Visualising `COLMAP` results on the provided `static_scene` example using `Viser` [7]. Each pyramid shape indicates the estimated camera pose of a frame extracted from the input video, and the point cloud shows the sparse 3D reconstruction of the scene.

- Train a 3D Gaussian Splatting (3DGS) [1] model on the same `static_scene` using the default `splatfacto` model in `nerfstudio` and the camera poses computed from `COLMAP`.
- Compare the rendered 3D scenes and explain the differences you observe, such as visual quality, rendering characteristics, and training behaviour. Include visualisations and interpretation of your findings in the report. See Fig. 2 for an example of the 3DGS rendering on the provided `dynamic_scene`.

2.2 Comparing Static and Dynamic Scenes [10%]

- Train a 3DGS model on the provided `dynamic_scene` using the same default `splatfacto` model in `nerfstudio` and the camera poses computed from `COLMAP`.
- Compare the results obtained for the static and dynamic scenes. Describe and interpret the differences you observe. Include visualisations and interpretation of your findings in the report.

2.3 Comparing Rendering Results with `COLMAP` and π^3 Poses [10%]

- Train a 3DGS model on the provided `dynamic_scene` using the same default `splatfacto` model in `nerfstudio`, but now with the camera poses computed from π^3.
- Compare the rendering results against those produced with `COLMAP` poses. Describe and interpret the differences you observe. Include visualisations and interpretation of your findings in the report.

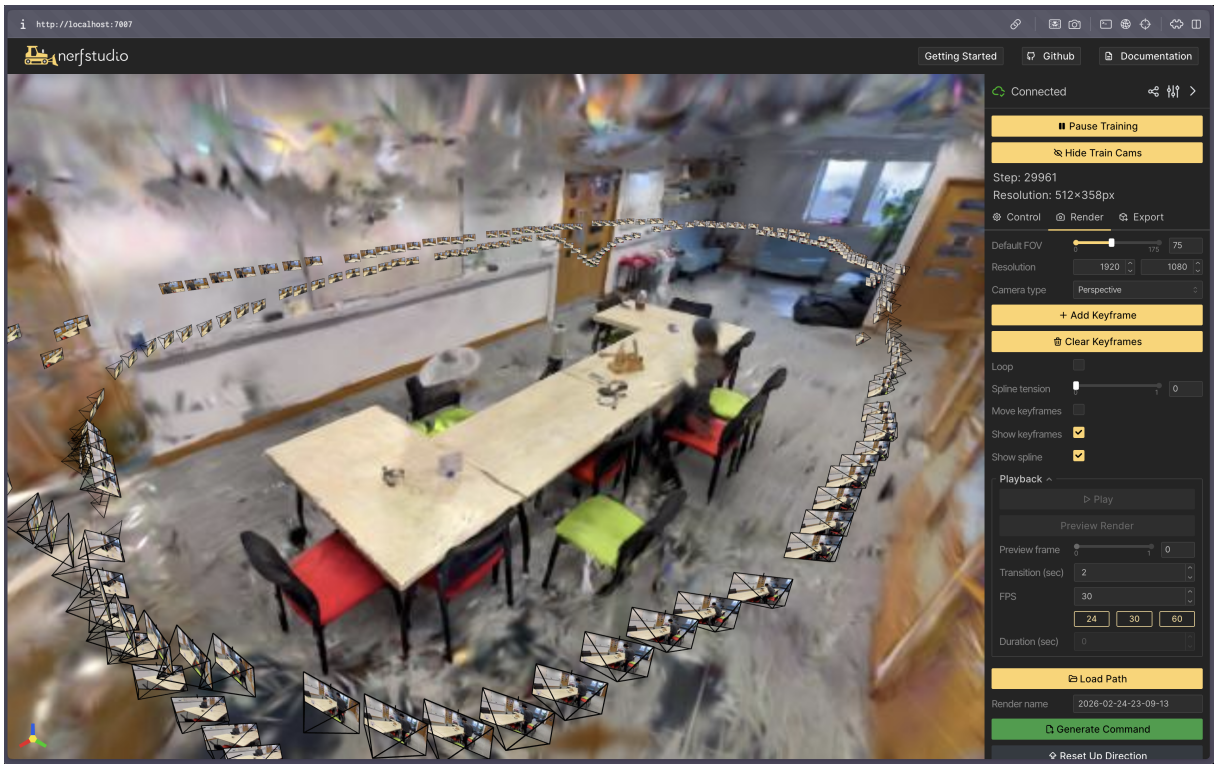

Figure 2: Visualising 3DGS rendering results on the provided `dynamic_scene` example using `nerfstudio` [5].

2.4 Improving the Dynamic Scene [20%]

Propose methods to improve the reconstruction and rendering quality of the dynamic scene. You may describe potential solutions even if you are unable to validate them through experiments. Clearly justify why your proposed approaches could address the observed limitations.

References

[1] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *TOG*, 42(4), 2023. 2

[2] 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. In *ECCV*, 2020. 1

[3] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1

[4] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 1

[5] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *SIGGRAPH*, 2023. 1, 3

[6] Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *ICLR*, 2026. 1

[7] Brent Yi, Chung Min Kim, Justin Kerr, Gina Wu, Rebecca Feng, Anthony Zhang, Jonas Kulhanek, Hongsuk Choi, Yi Ma, Matthew Tancik, and Angjoo Kanazawa. Viser: Imperative, web-based 3d visualization in python. *arXiv preprint arXiv:2507.22885*, 2025. 2