3D Scene Reconstruction Using Instant-NGP (Instant Neural Graphics Primitives)

Jaroslav Venjarski, Michal Ivan Basoš, Matej Hromada[1]
[1] Faculty of Electrical Engineering and Information Technology, STU Bratislava, Ilkovičová 3, 812 19 Bratislava, Slovakia
Institute of Multimedia Information and Communication Technologies, FEI STU
jaroslav.venjarski@stuba.sk

Abstract - **Instant Neural Graphics Primitives (instant-ngp) technology enables the rapid reconstruction of 3D scenes from 2D images. This paper provides a practical guide to the installation, configuration, and execution of the instant-NGP tool, focusing on processing video inputs using the colmap2nerf.py script and the main instant-ngp.exe application. The primary contribution of this work is an experimental analysis of the impact of key data preparation parameters such as video frame rate and scene bounding box size, on processing speed and the final visual quality of the 3D reconstruction. The results offer recommendations for the optimal setting of these parameters depending on the nature of the input video and the desired output.**

Keywords **- NeRF; instant-NGP; 3D scene reconstruction; novel view synthesis; multi-resolution hash encoding; CUDA; GPU; COLMAP; parameter configuration.**

I. INTRODUCTION

The creation of detailed three-dimensional models of real-world scenes and objects is a crucial task in numerous fields, ranging from virtual and augmented reality, through robotics and autonomous systems, to the digital preservation of cultural heritage. Traditional 3D reconstruction methods often require a compromise between processing speed and the visual quality of the resulting model. Neural Radiance Fields (NeRF) technology [1] brought a significant advancement in the quality of novel view synthesis, albeit at the cost of extremely long training times, which limited its practical deployment.

The Instant Neural Graphics Primitives (instant-ngp) project [2], developed by NVIDIA, presents a solution to this issue. By utilizing innovative techniques, particularly multi- resolution hash encoding and a highly optimized implementation for modern graphics processing units (GPUs) with CUDA support, instant-ngp enables the training of NeRF models orders of magnitude faster, within seconds to minutes, while maintaining high visual quality.

This paper focuses on the practical aspects of using the instant-ngp tool. The aim is to provide the reader with a comprehensible guide covering the entire process, from the installation of necessary tools and dependencies, through the preparation of input data from video footage using the `colmap2nerf.py` script and the COLMAP tool [4], to the execution of training and visualization in the `instant-ngp.exe` application. A key component of this paper is an experimental analysis of the impact of two important data preparation parameters: the frame rate extracted from the video (`-video_fps`) and the defined scene size (`-aabb_scale`) on the overall processing time and, crucially, on the visual quality and detail of the resulting 3D reconstruction. Based on these experiments, we formulate practical recommendations for selecting these parameters.

II. OVERVIEW OF UTILIZED TECHNOLOGIES

The 3D scene reconstruction process using instant-ngp involves the interplay of several key technologies and tools. This chapter provides a brief overview of their roles within the entire workflow:

- **Instant-NGP (`instant-ngp.exe`):** Represents the core of the entire system. It is a software tool from NVIDIA that implements rapid training and rendering of neural scenes. It utilizes an implicit scene representation and is highly optimized for GPU execution using CUDA. It provides an interactive graphical user inter- face for visualizing and controlling the training process, as well as options for exporting results.
- **NeRF (Neural Radiance Fields) [1]:** This is the fundamental concept upon which instant-ngp is built. Instead of traditional explicit geometry (e.g., polygons), NeRF represents a scene using a neural network (MLP) that learns to map 3D coordinates and viewing directions to color and density. This enables highly realistic rendering of novel views.
- **Multi-resolution Hash Encoding & Tiny CUDA NN:** These are the key technologies responsible for the dramatic speed-up of instant-ngp compared to the original NeRF. Hash encoding efficiently encodes input 3D coordinates using multiple levels of detail and small, quickly accessible hash tables. Tiny CUDA NN is an NVIDIA library providing extremely optimized implementations of these hash tables and small MLP net- works directly for the GPU.
- **`colmap2nerf.py`:** A Python script for data preparation, which calls FFmpeg and COLMAP.
- **FFmpeg:** A tool for extracting frames from video.
- **COLMAP [4]**: An external, open-source software for

Structure from Motion (SfM). In the context of instant-ngp, its SfM functionality is primarily used to automatically estimate camera intrinsic and extrinsic parameters (camera poses) from a set of input 2D images. These poses are essential for training the NeRF model.

- **C++/CUDA, Python, CMake:** These are the languages and tools used for development and compilation.

III. Requirements and Environment Setup

Successful execution of instant-ngp requires a specific hardware and software configuration.

A. Hardware and Software Requirements

Basic requirements include:

- **NVIDIA GPU:** Turing/Ampere/Ada architecture recommended (Maxwell minimum), with at least 8GB VRAM (4GB minimum).
- **OS:** Windows 10/11 (64-bit).
- **NVIDIA CUDA Toolkit:** Compatible version (e.g., 11.6+).
- **Microsoft Visual Studio:** 2019 or newer (with C++ components).
- **CMake:** Version 3.21+.
- **Python:** Version 3.7+.
- **Git**.
- **COLMAP** (binaries).
- **FFmpeg** (binaries).
- (Recommended: **Anaconda/Miniconda**).

The project must be compiled according to the official documentation [2] before use.

IV. Data Preparation and Training Execution

This section describes the basic steps for video processing and running instant-ngp.

A. Video Processing using `colmap2nerf.py`

The `colmap2nerf.py` script automates data preparation from video: it extracts frames (via FFmpeg), estimates camera poses (via COLMAP), and generates the necessary `transforms.json` file. Key command-line parameters include:

- `-video_in`: Path to the video.
- `-video_fps`: Number of frames extracted per second (impact analyzed in Section V.B).
- `-run_colmap`: Executes COLMAP.
- `-aabb_scale`: Scene size (impact analyzed in Section V.C).
- `-colmap_matcher`: COLMAP's feature matching method (e.g., `exhaustive`, `sequential`).

Example for video processing:

```
python scripts/colmap2nerf.py --video_in
<path/to/video.mp4> --video_fps 2 --run_colmap
--aabb_scale 8 --overwrite
```

Figure 1. Video processing.

B. Launching Training and Visualization with `instant-ngp.exe`

After generating `transforms.json`, training is initiated with the command:

```
.\instant-ngp.exe --scene <path/to/project>
```

Figure 2. Starting training.

The application loads the data and displays the training progress in an interactive GUI window, allowing scene navigation and parameter adjustment.

C. Exporting the 3D Model (Mesh)

Once a satisfactory NeRF model has been trained, `instant-ngp.exe` allows for the extraction of its geometry as a 3D mesh. This is typically done via the graphical user interface, often under a section labeled "Export mesh / volume / slices" by clicking a button such as "Mesh it!". The resulting mesh, commonly in `.obj` format, can then be imported into 3D modeling software like Blender for further editing, texturing, or preparation for 3D printing. Figure 1 shows an example of such an exported mesh, visualizing the reconstructed geometry.

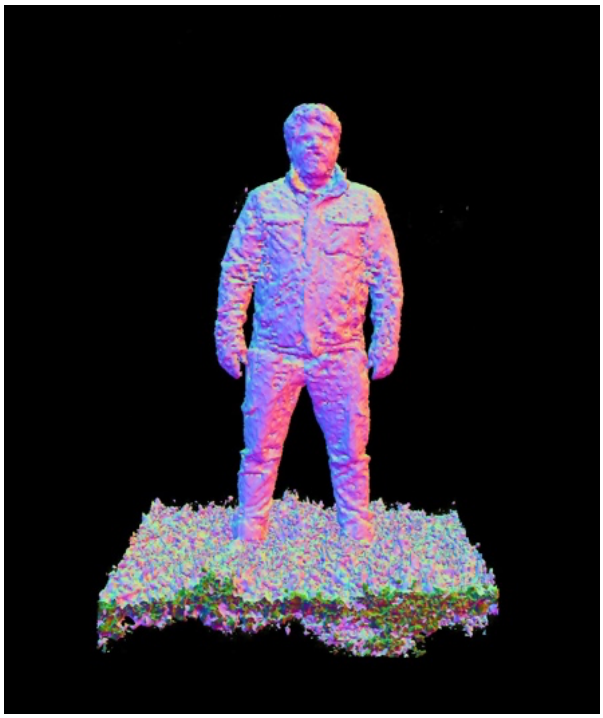

Figure 3. Example of an exported 3D mesh from a scene reconstructed with instant-ngp, visualized with normal map colors.

V. Analysis of Parameter Impact

We experimentally analyzed how the `-video_fps` and `-aabb_scale` parameters affect 3D reconstruction quality using a 4K handheld orbital shot (60fps, 46 seconds). The footage features a single subject centered in an outdoor environment, captured in one continuous 360° circuit with consistent camera distance and smooth movement throughout. All processing was performed on an NVIDIA GeForce RTX 4070 GPU.

A. Experimental Methodology

For various values of `-video_fps` (e.g., [2, 4, 8]) and `-aabb_scale` (e.g., [4, 8, 16]), we measured the data preparation time (`colmap2nerf.py`) and subjectively assessed the visual quality of the resulting reconstruction in `instant-ngp.exe`.

B. Impact of the `-video_fps` Parameter

Lower FPS (e.g., 2) significantly accelerated data preparation (COLMAP execution) but could lead to incomplete reconstruction with rapid camera movement. Higher FPS (e.g., 4 or 8) improved pose estimation robustness and slightly enhanced model quality, at the cost of considerably longer preparation time. Visual comparisons further illustrate these trade-offs.

C. Impact of the `-aabb_scale` Parameter

The choice of `-aabb_scale` affected the scene's extent and detail. Lower values (e.g., 4 or 8) were suitable for scenes with a limited extent or for focusing on a specific object, providing better detail in that area. Higher values (e.g., 16) captured a larger space but could lead to a loss of detail if the primary subject was small within that volume. A value too small resulted in scene cropping.

Figure 4. Visual result of reconstruction with `-aabb_scale=16`.

Figure 5. Visual result of reconstruction with `-aabb_scale=8`.

Figure 6. Visual result of reconstruction with `-aabb_scale=4`.

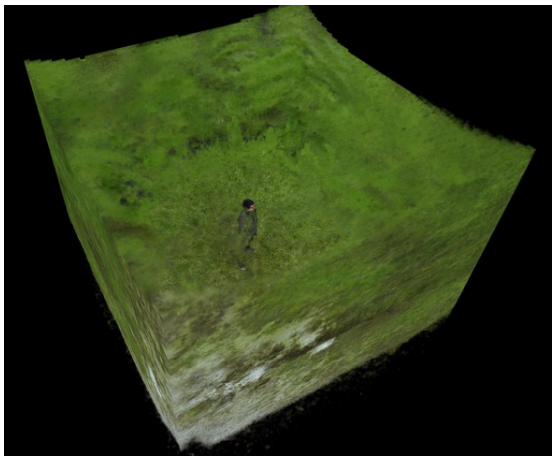

Figure 7. Visual result of reconstruction with `-aabb_scale=4` (full scale, unedited).

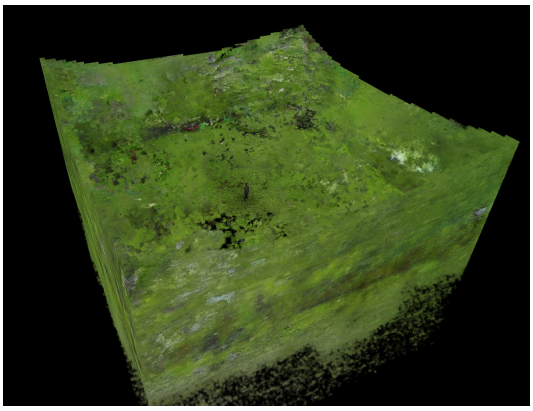

Figure 8. Visual result of reconstruction with `-aabb_scale=16` (full scale, unedited).

D. Summary of Results

Parameter selection involves a trade-off. Lower FPS is preferable for preparation speed, while higher FPS is better for robustness. `-aabb_scale` should be chosen according to the scene size – smaller for objects, larger for extensive scenes.

VI. Conclusion

This paper provided a practical overview of using the instant-ngp tool for reconstructing 3D scenes from video inputs, from installing the necessary components to running training and visualization. A key contribution was the analysis of the impact of the `-video_fps` and `-aabb_scale` pa- rameters of the `colmap2nerf.py` script on the resulting process. Our experiments confirmed that the choice of these parameters represents an important compromise. A lower `-video_fps` value (e.g., 2) significantly reduces the data preparation time (especially COLMAP execution), but with rapid camera movement, it can lead to a less robust or in-complete reconstruction. Higher FPS increases robustness but at the cost of longer processing time. The `-aabb_scale` parameter must be adapted to the size and type of the scene – too small a value will crop the scene, while too large a value can reduce the detail of the reconstruction in the area of interest.

Based on our observations, we recommend starting with `-video_fps=2` or `-video_fps=4` and

`-aabb_scale=8` or `-aabb_scale=16` for typical handheld videos. These values can then be fine-tuned according to the specifics of the particular video and the requirements for the final quality and speed.

Instant-ngp proves to be a powerful and relatively user-friendly tool for the rapid creation of photorealistic 3D models. However, understanding the impact of key data preparation parameters is essential for achieving optimal results.

Acknowledgement

Research in this paper was supported by projects DISIC (09I05-03-V02-00077), InteRViR (VEGA 1/0605/23), and NEXT (ERASMUS-EDU-2023-CBHE-STRAND-2, ID: 101129022).

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neu- ral Radiance Fields for View Synthesis," in Proc. European Conf. Comput. Vis. (ECCV), Glasgow, UK, 2020, pp. 405– 421.

[2] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," ACM Trans. Graph., vol. 41, no. 4, Art. no. 102, July 2022.

[3] J. Venjarski, Ľ. Likó, Š. Tibenský, M. Vančo, and G. Rozinaj, "Keypoint-Based Foreground-Background Image Segmentation," in Proc. 66th International Symposium ELMAR-2024, Zadar, Croatia, 2024, pp. 113–116.

[4] J. L. Schönberger and J. M. Frahm, "Structure-from-Motion Revisited," in Proc. IEEE Conf. Comput. Vis. Pattern Recog- nit. (CVPR), Las Vegas, NV, USA, 2016, pp. 4104–4113.

[5] NVIDIA Developer Blog, "Getting Started with NVIDIA Instant NeRFs," [Online]. Available: https://developer.nvidia.com/blog/getting-started-with-nvidia-instant-nerfs/

[6] NVlabs, "instant-ngp," GitHub Repository, [Online]. Avail- able: https://github.com/NVlabs/instant-ngp

[7] bycloudai, "instant-ngp-Windows," GitHub Repository, [On- line]. Available: https://github.com/bycloudai/ instant-ngp-Windows?tab=readme-ov-file

[8] NVIDIA, "CUDA Toolkit," [Online]. Available: https:// developer.nvidia.com/cuda-downloads

[9] Microsoft, "Visual Studio Downloads," [Online]. Avail- able: https://visualstudio.microsoft.com/ downloads/

[10] Anaconda, "Anaconda Distribution," [Online]. Available: https://www.anaconda.com/download

[11] CMake, "Download CMake," [Online]. Available: https://cmake.org/download/

[12] COLMAP, "Release 3.7," GitHub Release, [Online]. Available: https://github.com/colmap/colmap/ releases/tag/3.7

[13] javieryu, "nerf_bridge," GitHub Repository, [Online]. Available: https://github.com/javieryu/nerf_ bridge