

Simplifying NeRF: Creating an Intuitive Web-Based 3D Scene Interface

Eduard von Briesen

March 31, 2023
Version: My First Draft



Department of Mathematics,
Informatics and Statistics,
Institute of Informatics



Munich Film School,
Chair for AI

Masters Thesis

Simplifying NeRF: Creating an Intuitive Web-Based 3D Scene Interface

Eduard von Briesen

Supervisors Prof. Dr. Sylvia Rothe and Cristoph Weber

March 31, 2023



Eduard von Briesen

Simplifying NeRF: Creating an Intuitive Web-Based 3D Scene Interface

Masters Thesis, March 31, 2023

Supervisors: Prof. Dr. Sylvia Rothe and Cristoph Weber

LMU Munich

Department of Mathematics, Informatics and Statistics

Institute of Informatics

Artificial Intelligence and Machine Learning (AIML)

Akademiestraße 7

80799 Munich

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Abstract (German)

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Acknowledgement

Contents

1. Introduction	1
1.1. Background	1
1.2. Overview of NeRF	2
1.3. Research Objectives	2
1.4. Research Question	3
2. Related Work	5
2.1. Existing Methods for creating NeRFs	5
2.2. Review of tools and technologies in film and VFX	5
2.3. User Research	5
2.4. Conclusion	5
3. Methodology	7
3.1. Research Design	7
3.2. Initial User Research	7
3.2.1. Participant Selection Criteria	7
3.2.2. Interview Methodology	8
3.2.3. Key Findings	8
3.3. Prototype Development	10
3.3.1. Technical Specifications	10
3.3.2. Design Considerations	10
3.4. User Study and Evaluation	10
3.4.1. Tasks Based Usability Test	10
3.4.2. User Experience Questionnaire	11
3.4.3. Follow-up Interview	12
3.4.4. Qualitative Analysis	12
3.4.5. Quantitative Analysis	13
4. User Interface Design	15
4.1. User Flow and Navigation	15
4.2. User Interface	16
4.2.1. Dashboard	16
4.2.2. Project Section	17

5. Technical Implementation	21
5.1. System Architecture	21
5.2. Frontend Development	21
5.3. Backend Development	23
5.4. Challenges and Solutions	24
5.4.1. Limitations of tRPC	24
5.4.2. Working with the nerfstudio CLI	25
5.5. Lessons Learned	26
5.6. Future Directions	26
5.6.1. Improved CLI Integration	26
5.6.2. Improved Viewer Integration	26
6. Results	27
6.1. Analysis of User Experience Questionnaire	27
6.2. Findings from Qualitative User Testing	27
6.3. Integration and Findings	27
7. Conclusion	29
7.1. Key Findings	29
7.2. Contributions to the Field	29
7.3. Future Work	29
A. Appendix	31
A.1. Interview Questions	31
A.2. User Study Questionnaire	33
A.3. User Testing Results	34
A.4. Prototype Documentation	34
Bibliography	35
List of Figures	37
List of Tables	39

Introduction

1.1 Background

The present state of NeRF research presents significant advancements that have greatly influenced the field of 3D scene modeling and rendering. This section provides an overview of the current research landscape, highlighting key NeRF frameworks and relevant projects, and identifying the challenges and opportunities that inform our research objectives.

Two prominent NeRF frameworks with user interfaces, namely Instant NGP [5] and Nerfstudio [9], have emerged as leaders in enabling users to explore and manipulate 3D scenes. These frameworks offer features such as real-time scene rendering, adjustable training parameters, and the creation of camera trajectories for video rendering.

However, the utilization of these interfaces often demands a high level of technical expertise, as they are designed to complement, rather than replace, command-line interfaces. Users must engage with terminal-based processes for tasks such as video data preprocessing, model training, and rendering output.

Additionally, several innovative projects have expanded the NeRF landscape. Notably, CLIP-NeRF [12], Instruct-NeRF2NeRF [4], Text2LIVE [2], and SINE [1] have introduced text-based editing approaches, broadening the possibilities for manipulating NeRF models. PaletteNeRF [13] focuses on color editing, while NeRF-Editing [15] enables mesh editing.

This research aims to identify key challenges and opportunities in NeRF frameworks and interfaces, as demonstrated by these significant contributions. This knowledge will guide the development of a user-friendly, web-based interface and integrated editing plugins, with the ultimate goal of enhancing the accessibility and usability of NeRF frameworks for a broader user base.

1.2 Overview of NeRF

1.3 Research Objectives

The research objectives of this study are as follows:

1. **Exploration of NeRF Interaction Capabilities:** This study aims to explore the existing interaction capabilities within NeRF frameworks comprehensively. It involves an analysis of the current state of NeRF interfaces and an investigation into user engagement, visualizations, and manipulation of NeRF scenes.
2. **Development of a Web-Based User Interface:** Building on insights gained from the exploration phase, the primary objective is to design and implement a user-friendly web-based interface for NeRF.
3. **Streamlined NeRF Creation and Manipulation:** The central goal is to simplify the process of NeRF creation and manipulation, eliminating the need for users to deal with complex command-line interfaces or extensive local setup. The web-based interface will provide an intuitive and efficient user experience.
4. **Integration of Diverse Editing Plugins:** To enhance the creative potential of NeRF, various editing plugins will be integrated into the web-based interface. The objective is to expand the functionality and versatility of the NeRF framework.

The research aims to advance NeRF frameworks' capabilities and accessibility, making them accessible to a broader audience and fostering innovation in 3D scene modeling and rendering.

1.4 Research Question

1. **Enhancing NeRF Frameworks:** How can a web-based interface improve the user experience and accessibility of NeRF frameworks, and what impact will these enhancements have on user-friendly NeRF creation and manipulation?
2. **Overcoming Technical Challenges:** What technical challenges and limitations are associated with current NeRF frameworks and interfaces, and how can innovative design and technology choices in a web-based interface overcome these challenges?
3. **Innovative Editing Integration:** How can novel editing approaches be seamlessly integrated into a web-based NeRF interface to enhance creativity and usability, and how do these methods compare with traditional NeRF editing techniques?

Related Work

2.1 Existing Methods for creating NeRFs

Luma AI

Nerfstudio

2.2 Review of tools and technologies in film and VFX

2.3 User Research

2.4 Conclusion

Methodology

3.1 Research Design

The research was planned in three distinct phases, initial user research, development of the prototype and user testing. Each of these phases informs the one building on it.

3.2 Initial User Research

The initial round of user interviews was conducted to gather insights for the development of a user-friendly NeRF interface. The focus was on understanding the needs, challenges and preferences of users with varying levels of experience with the creation of NeRF models.

3.2.1 Participant Selection Criteria

Participants were selected based on their pre-existing experience with NeRF and some association to the film industry. They represent a wide range of experiences and technical knowledge of NeRF, but also applied the technology in different contexts.

This diversity in users helped to identify common needs and challenges, as well as to understand the specific requirements of different user groups.

3.2.2 Interview Methodology

The interviews themselves were semi-structured, following a set of predefined questions (A.1), while also allowing for open-ended discussions and follow-up questions. The interviews were conducted in a one-on-one setting, either in person or via video call, and lasted between 30 and 60 minutes. Although the interview template was prepared in english, all interviews were conducted in german.

Questions followed a logical flow, starting with general background information and moving on to more specific needs and preferences. After being introduced to the purposes of the interview and the goals of this research, participants were questioned about their experience with NeRF and the tasks they typically aim to achieve when working with NeRF. They were asked what challenges or pain points they encounter when using current NeRF frameworks or interfaces, and what features or functionalities would make a NeRF interface most useful for their work or research. Interviewees were encouraged to share there own thoughts and suggestions on the development of a NeRF interface.

With the agreement of participants, the interviews were recorded and transcribed for further analysis. The data was then coded and categorized to identify common themes and patterns across the interviews.

3.2.3 Key Findings

NeRF in the Film Industry

There exist different applications of NeRF in the film industry, ranging from traditional visual effects to virtual production and location scouting in the pre-production phase. Tradition creation and animation of 3D scenes is costly and time-consuming, and NeRF is seen as a potential solution to this problem.

Some participants expressed their concerns about the quality of current NeRF models, that is often not sufficient for professional use. Lack of detail and visual artifacts in the scene were mentioned as common issues. Additionally it lacks the ability to edit and manipulate the models when compared to traditional 3D models,

which presents a crucial limitation for the application in a creative industry like film, where models are iteratively developed and refined.

A more realistic use-case of NeRF in the film industry, with the technologies current limitations, is the use in pre-visualization and location scouting. The ability to quickly capture and render 3D scenes from real-world locations is seen as a potential time-saver in the pre-production phase of film-making. Models can be used to plan camera movements and lighting, and to get a better understanding of the spatial relations between different elements in the scene. One concern raised in this context was the lack of accurate scale of the models, when exporting them into other 3D software.

Optimizing Parameters and Workflow

A typical workflow when creating NeRFs follows three steps, first the input data has to be pre-processed, then the actual model training is done and finally the some data is exported. Parameterization in the pre-processing and training phases can have a profound impact on the quality of the final NeRF. This was noted by participants with a technical background, that were working with NeRF in a research context. For them the optimization of parameters is an integral part of there workflow, often re-training models multiple times, in order to analyze the variation in the results. Results can be quantified with tooling to analyze runs, such as TensorBoards.

User Interface and User Experience

Participants

3.3 Prototype Development

3.3.1 Technical Specifications

3.3.2 Design Considerations

3.4 User Study and Evaluation

To evaluated the usability and usefulness of the developed prototype, a user study was conducted. The study was designed to gather feedback on the user experience of the prototype, and to identify any usability issues or challenges that users encountered. It followed a mixed-methods approach, combining quantitative and qualitative data collection and analysis. Participants were given a set of tasks to complete with the prototype, after which they fill out a questionnaire and participate in a short follow-up interview.

3.4.1 Tasks Based Usability Test

The usability test was conducted in a controlled environment, with participants being asked to complete a two tasks with the prototype. The tasks were designed to cover a range of functionalities and features of the prototype, and represent a typical workflow when creating NeRF models.

1. Task: Create a new project.
2. Task: Upload a prepared video file.
3. Task: Pre-process the uploaded file to prepare it for training.
4. Task: Switch to an existing project train a NeRF that already pre-processed data.
5. Task: Start a NeRF training.

6. Task: Create a camera path in the viewer.

7. Task: Export a video.

To keep an appropriate time frame, none of the tasks required completion of a training process, and pre-processed data and pre-trained models were provided to the participants. On average, participants took 30 minutes to complete the tasks.

Participants were passively observed while working on their tasks, to identify any problems or operation errors they encountered and to determine their overall performance. In addition the screen was recorded to capture the participants interactions with the prototype, and to allow for a more detailed analysis of their behavior later on.

3.4.2 User Experience Questionnaire

After completing their tasks, users were asked to fill out the User Experience Questionnaire (UEQ) [ueq], a standardized questionnaire for the assessment of user experience. It measures user experience in six dimensions:

- **Attractiveness** - the overall impression of the product
- **Perspicuity** - the clarity and understandability of the product
- **Efficiency** - the perceived effort required to use the product
- **Dependability** - the perceived reliability and trustworthiness of the product
- **Novelty** - the perceived originality and innovation of the product
- **Stimulation** - the perceived level of excitement and engagement with the product

This covers both classical usability goals (Efficiency, Perspicuity, Dependability) and user experience qualities (Novelty, Stimulation). Attractiveness is purely a valence dimension, and is not directly related to usability or user experience.

In total the questionnaire consists of 26 items, each represented by two terms of opposite meaning. The order of the terms is randomized for each item, to avoid bias. Participants are asked to rate each item on a 7-point scale, from -3 to +3, with 0 representing a neutral response. An example of the scale can be seen below:

boring o o o o o o exciting

The format of the questionnaire gives participants a clear and simple way to quickly express their feelings and thoughts about the prototype, without much effort.

In this study the questionnaire was filled out by participants in digital form, using a web-based survey tool. The survey included additional questions to gather demographic information and to capture prior experience with NeRF and other 3D modeling tools. This allowed for a more efficient data collection and analysis, across in-person and remote participants.

3.4.3 Follow-up Interview

After completing the usability test, participants were engaged in a short follow-up interview, to gather more detailed feedback on their experience with the prototype. Similar to the initial user interviews, these interviews were semi-structured, following a pre-defined set of question, with room for participants to share their own thoughts and suggestions. The questions can be categorized into general usability, tasks specific feedback and suggestions for improvement. The interview template can be found in the appendix A.2.

3.4.4 Qualitative Analysis

Both the recordings of the usability test and the follow-up interviews were analyzed to identify common themes and patterns in the feedback of participants. The video recordings were coded to highlight any usability issues or challenges that participants encountered during the tasks. The audio recordings of the interviews were transcribed and coded.

3.4.5 Quantitative Analysis

User Interface Design

This chapter outlines the design of the prototype, focusing in the user interface and the user experience. The design process was informed by the initial user research.

4.1 User Flow and Navigation

As an initial step in the design process, a user flow diagram was created to visualize required user interaction and their relations. In a first rough sketch, key interactions were arranged in a linear sequence, representing the typical workflow when creating NeRF models (4.1).

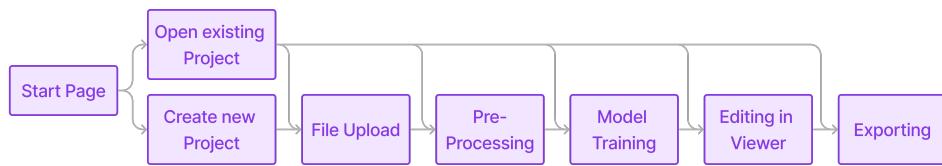


Fig. 4.1.: Flow Diagram

Building on this outline, complex interaction were broken down into smaller steps to scope out what user actions were required to complete a task. Interactions could then be group into views, and the navigation between these views could defined. The views were also enriched with more detailed information on the exact user interactions (4.2).

These diagrams were used as a reference throughout the design and development process, to ensure a structure that followed the user's mental model and to keep the user interface consistent.

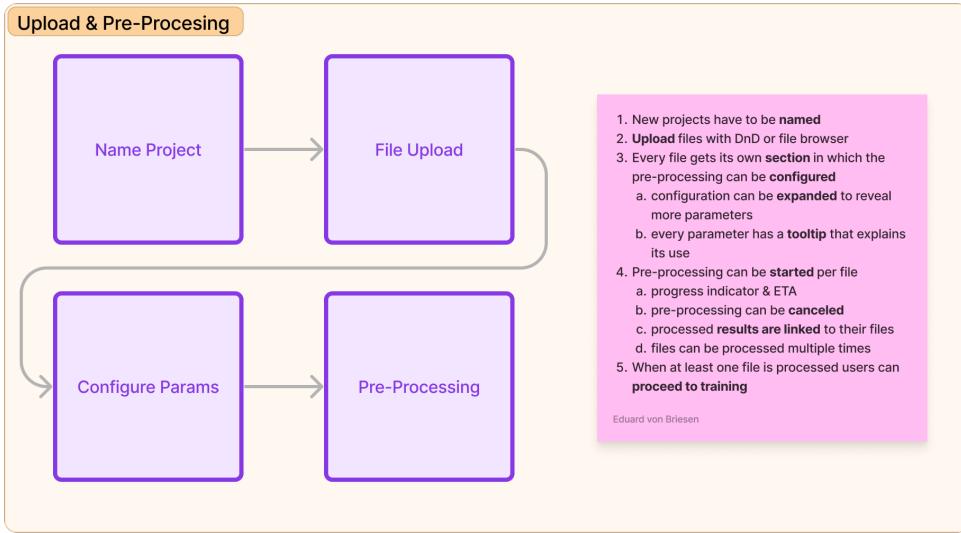


Fig. 4.2.: Excerpt of a View from the Flow Diagram with detailed interactions

4.2 User Interface

The user interface was designed to be as simple as possible, while still providing all necessary functionality. The design of the prototype can be broken down into two main parts: a dashboard that gives an overview of all projects and a project section that provides users with the tools to create and edit NeRF models.

4.2.1 Dashboard

The dashboard is the first view that users see when opening the application. It shows all previously created projects and allows users to create new ones. Projects are represented as cards, showing the project name, a preview of on the provided input images (if present), and tags the indicate the current status of the project. An additional card is present through which users can create a new project, by providing a name. Projects can be opened by clicking a button on the respective card, when creating a new project, users are redirected to the project section.

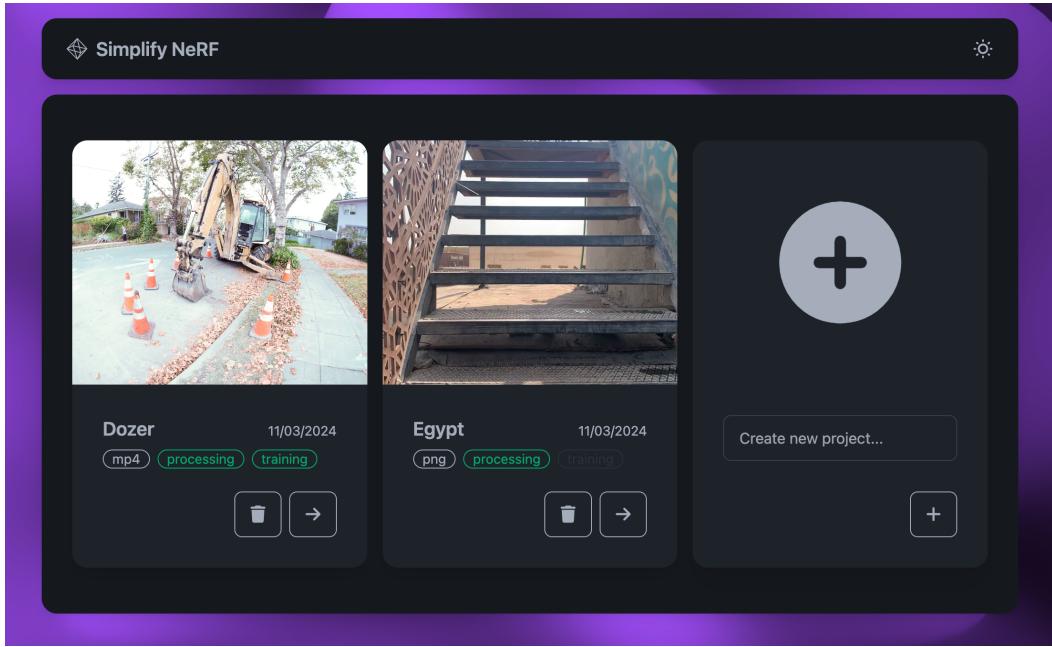


Fig. 4.3.: Dashboard

4.2.2 Project Section

The project section is the core of the application, through which users can create and edit NeRF models. The section is divided into three parts: the input section, the training section, and the rendering section. Across all sections, users can track their progress through a progress bar at the top of the screen, that also enables easy navigation between the different sections.

Input Section

The input section combines the first few interactions, as mapped out in the user flow diagram. First users are prompted to upload their input data, which can be done by dragging and dropping files into the browser window or by clicking a button to open a file dialog. Files can be either a set of images or a video, and there are some guardrails in place to ensure that the input data is valid. Once the input data is uploaded, it has to be processed before it can be used for training. The pre-processing can be configured by the user, this includes parameters such as the lens-type, or matching method. Parameter input fields vary based on the type of input data, and are only shown when relevant. Once the user is satisfied

with the settings, they can start the pre-processing. Feedback on the progress of the pre-processing is given through a console that shows the output of the process running on the server. When the pre-processing is finished, the user can move on to the training section.

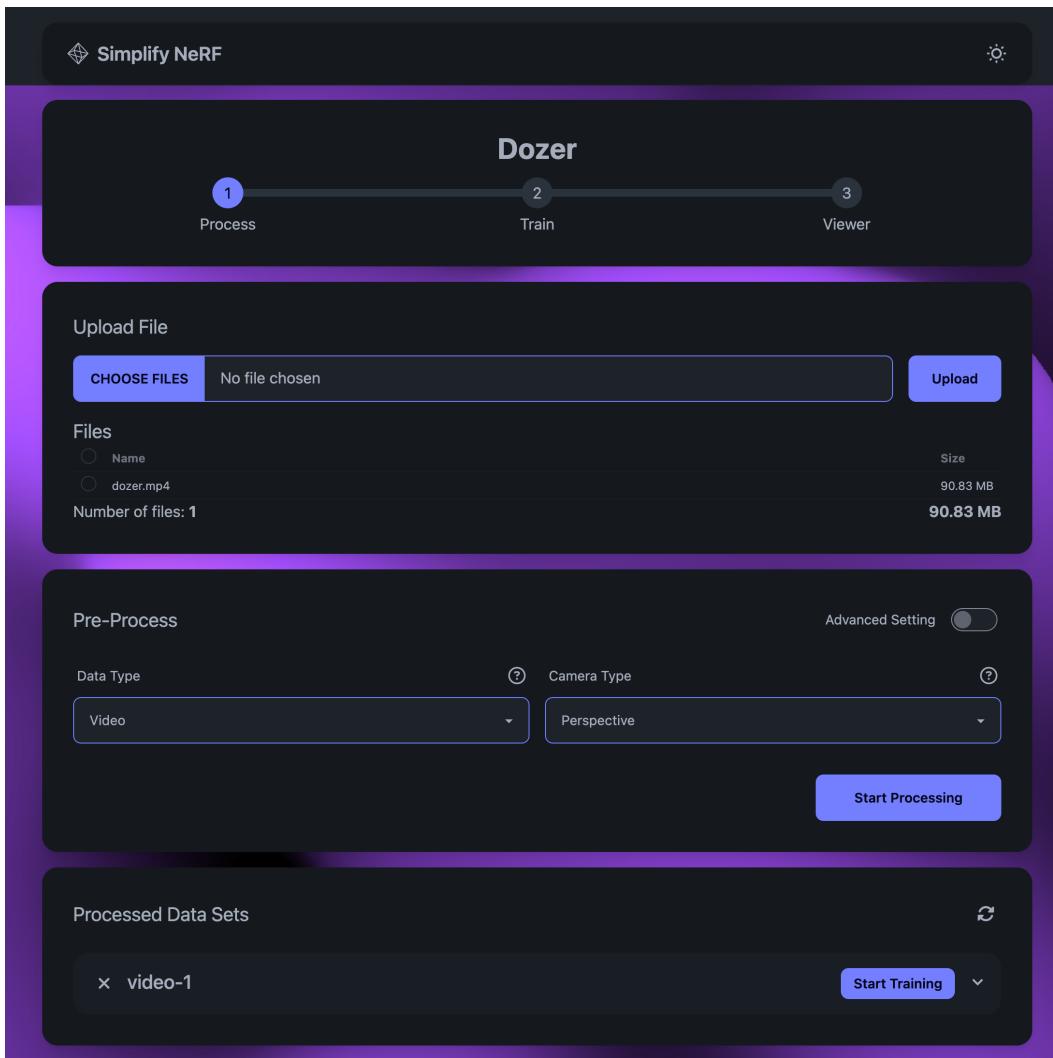


Fig. 4.4.: Processing Input Data

In case the data was already pre-processed, a list is visible that shows all available pre-processed data, and the user can select one to use for training. Users can also inspect the configuration with which the data was pre-processed, and delete it if necessary.

Training Section

The training section is structured similarly to the input section, with a form that allows users to configure the training process, and a console that shows the output of the training process running on the server.

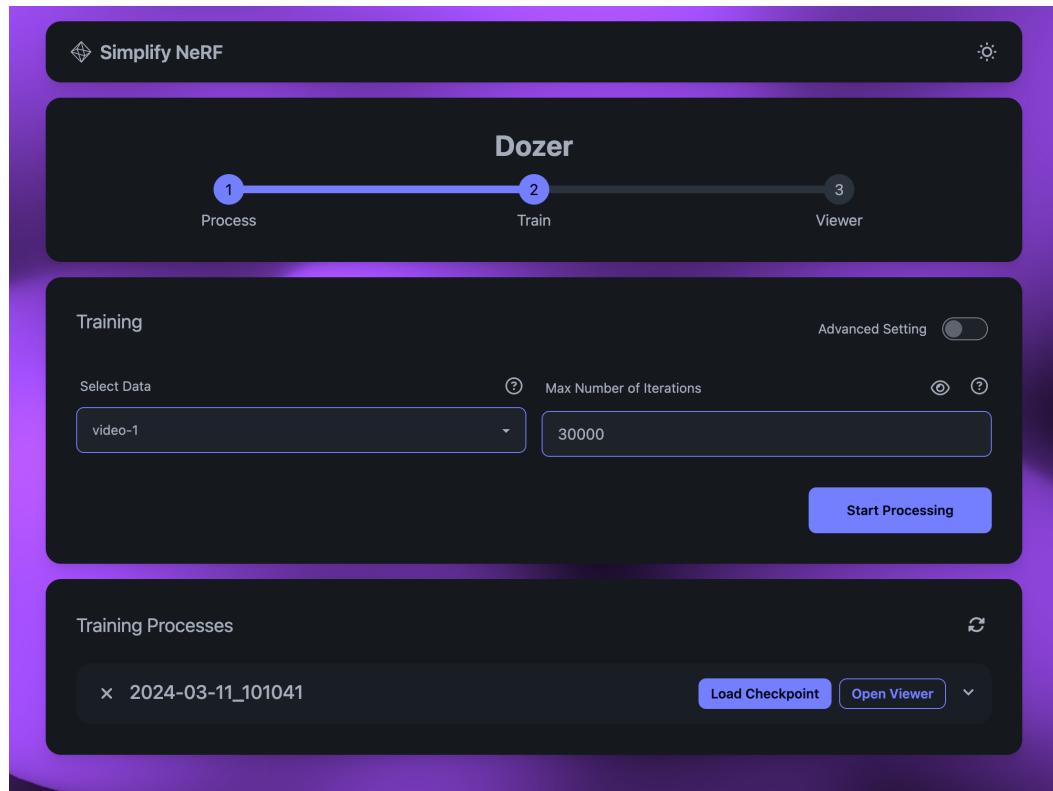


Fig. 4.5.: Training Section

Rendering Section

The rendering section is the final step in the process, and allows users to render the final NeRF model. At its core is the Nerfstudio Viewer that is integrated into the application. It provides users with all the functionality available in the standalone version, with a few integration that simplify the render process. Instead of providing commands that have to be executed in the terminal, the rendering is started by clicking a button.

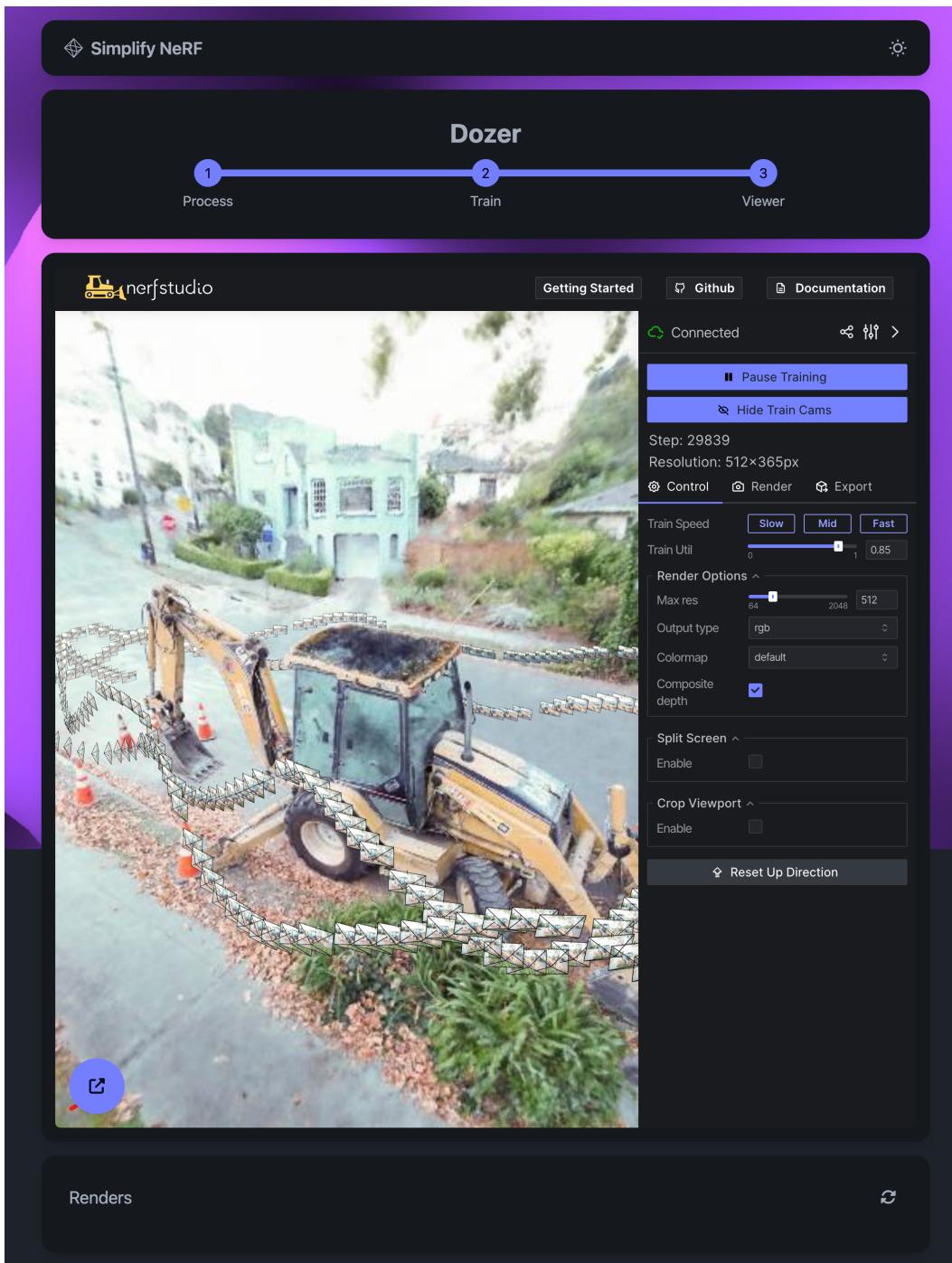


Fig. 4.6.: Viewer Section

Technical Implementation

5.1 System Architecture

The architecture followed a standard client server pattern, with the server functioning as a wrapper for the nerfstudio CLI, and the client as a web application. The server is responsible for handling incoming requests from the client, and translating them into commands that the nerfstudio CLI could understand. Requests from the client were sent to the server using HTTP requests, and in case of an asynchronous operation, the server would update the client on using websockets.

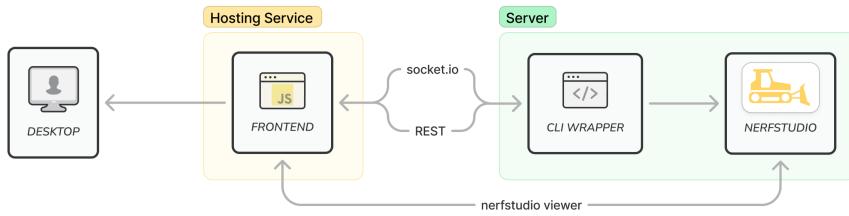


Fig. 5.1.: System Architecture

For the prototype, all the components were composed into a single docker container and deployed as a single unit. This leveraged the pre-configured container provided by nerfstudio, and allowed for a quick and easy deployment of the prototype.

5.2 Frontend Development

The frontend was built with React [@7], as it is a popular and well supported framework for building web applications. Vite [@11] was used as a build tool, as it provides a fast and efficient development experience. To accelerate the styling pro-

cess, Tailwind CSS [@8] was used, it is a utility-first CSS framework that provides a set of pre-defined classes that can be used to style components. In addition the daisyUI [@3] component library provided a set of pre-styled components that could be used to quickly build the UI.

Extensibility

Extensibility was a key consideration during the development of the frontend, as the underlying nerfstudio CLI is in itself extendable. All parameters for processing and training a NeRF model are configurable using JSON or strongly typed Type-Script objects.

```
1 const stepsPerSave: NumberInput = {  
2     name: "stepsPerSave",  
3     label: "Steps Per Save",  
4     tooltip: "Number of steps between each save of the model.",  
5     inputType: "number",  
6     defaultValue: 1000,  
7 };
```

Listing 5.1: Parameter Option configuration

Currently the supported input types are: number, select and boolean, but it is easy to add new types by extending the configuration object. It is also possible to define dependent parameters, that are only shown when a certain condition is met. Images to illustrate the effect of a parameter can be added as well, to provide additional context to the user.

Filters and presets are also configurable using arrays of the names of the parameters that should be included in the filter or preset. The full configuration can be found in the `frontend/src/config` folder in the codebase.

Nerfstudio Viewer Integration

The nerfstudio viewer is built using Viser [6], a library for building 3D visualizations using python. This posed some limitations in integrating it into the frontend, as it is not easily possible to directly embed a python application into a web application.

To work around this, the viewer was hosted by nerfstudio and embedded into the frontend using an iframe.

Some modifications could be made to the viewer to make it more suitable for embedding. This included some simple styling changes to make the viewer fit better into the frontend. More complex changes were also made to improve the user experience. The viewer contained several interactions where it was necessary for the user to copy console commands, to be used in the CLI. These interactions were replaced with buttons that send a request to the server to execute the command instead.

This main shortcoming of this approach is that the frontend is unaware of the state of the viewer, and cannot update the UI based on action triggered in the viewer. It has to rely on polling the server to get the current state of rendering processes to give feedback to the user.

These modifications were done at build-time of the container by applying a patch to the nerfstudio source code of the base image.

5.3 Backend Development

The backend was built using tRPC [@10], a framework for building type-safe APIs in TypeScript. This type-safety was useful in building the API, as nerfstudio endpoints require a specific set of parameters of various types, that could be easily defined using TypeScript and reused in the frontend.

A major concern for the backend was the handling of asynchronous operations. Processes can often take several minutes to complete, and the user should be able to see the progress of these operations. TRPC implements a feature called subscriptions, which allows the client to subscribe to a certain event, and receive updates when that event occurs. This is used for any long running operations, such as pre-processing or training a NeRF model.

Some additional endpoints were implemented using Express. This includes the file upload endpoint, which is used to upload images and videos to the server. As well

as the render endpoint, called through the viewer, as there does not exist a tRPC client for Python.

```
1  export const nerfstudioRouter = router({
2    process: publicProcedure
3      .input(
4        z.object({
5          project: z.string(),
6          dataType: z.enum(["images", "video"]),
7          ...
8        }),
9      )
10     .subscription(({ input }) => {
11       return observable<{message: string}>((emit) => {
12         const args = ...
13         const process = spawn("ns-process-data", args, {
14           cwd: path.join(WORKSPACE, input.project),
15         })
16         process.stdout.on("data", (data: any) => {
17           emit.next({message: data.toString()});
18         });
19       });
20     })
21   });

```

Listing 5.2: Example tRPC endpoint for Pre-Processing

All project related data is stored in a workspace directory, which is mounted as a volume in the docker container. This allows for the data to persist between container restarts, and for the user to access the data outside of the container.

5.4 Challenges and Solutions

5.4.1 Limitations of tRPC

Although tRPC is a powerful tool for building APIs, with tight integration into the frontend, it is not without its limitations.

It lacks support for multipart/form-data file uploads, which are necessary for uploading images and videos. This was solved by implementing a separate endpoint using Express, which is used to upload files to the server.

The way tRPC handles subscriptions is also not ideal for long running processes. If the client disconnects, the subscription is lost, and the client will lose the context for incoming events. A better solution could use a standard websocket connection, with the events containing the state necessary for the client to properly update the UI.

As mentioned above, tRPC does not have a client for Python, which is necessary for the viewer. This required the implementation of another separate endpoint using Express, which is called by the viewer to render the NeRF model.

All these limitations could be solved by using a more general purpose framework, such as Express.

5.4.2 Working with the nerfstudio CLI

Building the prototype against the nerfstudio CLI offered a useful layer of abstraction. This enabled rapid development, without having to worry about the underlying implementation details of a huge codebase. However, this abstraction comes at the cost of flexibility and overall complexity of the system.

The CLI is built using tyro [14], a tool for building command line interfaces in Python using configuration objects. Efforts were made to translate the configuration objects into TypeScript objects, that could be used throughout the project. This would allow for a more seamless integration of the CLI. Sadly this was not possible, as the configuration objects are not easily serializable, and would require a lot of additional work to implement.

Thus the implemented solution has to rely on manually constructing the commands based on documentation, which is error prone and not very maintainable.

5.5 Lessons Learned

5.6 Future Directions

5.6.1 Improved CLI Integration

As outlined above the approach of wrapping the nerfstudio CLI with a custom API has some limitations. A more integrated and robust solution would be built within the nerfstudio codebase itself.

A possible solution might be able to re-use the existing configuration object, used for CLI, to build a REST API. This would allow for a more seamless integration of the CLI into the frontend, and would allow for more flexibility in the future. It would also reduce the overall complexity of the system, as it would not require a separate server to handle requests.

5.6.2 Improved Viewer Integration

The current approach of embedding the viewer in an iframe is not ideal. The nerfstudio viewer in version 0.3.4 and earlier, was built using React with a direct integration of Viser. With the first major release of nerfstudio, the viewer was rewritten in Python, and the React integration was removed.

Re-building this projects frontend with a framework such as Viser, would be very limiting in terms of features and flexibility. A better solution would be to integrate Viser into the frontend, akin to the original nerfstudio viewer. This integration would allow for a more seamless interaction between the frontend and the viewer, and improve the overall user experience. UI elements could be re-used to ensure a consistent look and feel, and the frontend could react to events triggered in the viewer.

6

Results

6.1 Analysis of User Experience Questionnaire

6.2 Findings from Qualitative User Testing

6.3 Integration and Findings

Conclusion

7.1 Key Findings

7.2 Contributions to the Field

7.3 Future Work

Appendix

A.1 Interview Questions

Introduction

- Thank the interviewee for their participation.
- Briefly explain the purpose of the interview, which is to gather insights for the development of a NeRF interface.
- Assure the interviewee that their responses will be kept confidential.

Background

- Can you briefly describe your experience with NeRF or 3D modeling in general?

Needs and Challenges

- What specific tasks or objectives do you typically aim to achieve when working with NeRF models or 3D scenes?
- What are the main challenges or pain points you encounter when using current NeRF frameworks or interfaces?

Usability and Features

- In your opinion, what features or functionalities would make a NeRF interface most useful for your work or research?
- Are there any specific editing or manipulation tools you find lacking in current NeRF interfaces?

- How important is real-time visualization and interactivity in a NeRF interface for your needs?

Ease of Use

- How do you envision the ideal user interface for NeRF? What elements would make it easy to use, even for those with limited technical expertise?
- What level of technical knowledge or familiarity with 3D modeling should the ideal NeRF interface require from its users?

Integration and Compatibility

- Are there any other software tools or workflows you typically use alongside NeRF, and how important is it for a NeRF interface to integrate with these tools?
- Do you have any preferences regarding the file formats or data compatibility that the NeRF interface should support?

Feedback and Suggestions

- Are there any additional thoughts, suggestions, or requirements you would like to share regarding the development of a NeRF interface?
- Is there anything else you believe is essential for us to understand about your needs and expectations?

Closing

- Thank the interviewee for their time and valuable input.
- Provide contact information for any follow-up questions or clarifications.

A.2 User Study Questionnaire

Usability Experience

- Can you share your overall impressions of using the Simplify NeRF application?
- Were there any specific features or functionalities of the application that stood out to you positively? If so, why?
- On the other hand, were there any aspects of the application that you found challenging or frustrating? Please elaborate.

Task-Specific Feedback

- Were there any particular steps or actions within the tasks that you found confusing or unclear? If yes, could you describe them?
- Were there any features or functionalities you expected to find in the application that were missing? If so, what were they?

Suggestions for Improvement

- Based on your experience using the Simplify NeRF application, do you have any suggestions for improving its usability or functionality?
- Are there any specific changes or enhancements you would like to see in future versions of the application?
- How do you think the application could better meet your needs or expectations as a user?

Closing

- Is there anything else you would like to add or share about your experience with the Simplify NeRF application?

- Thank the participant for their time and valuable feedback. Offer contact information for any follow-up questions or clarifications.

A.3 User Testing Results

A.4 Prototype Documentation

Bibliography

- [1]Chong Bao, Yinda Zhang, Bangbang Yang, et al. “SINE: Semantic-driven Image-based NeRF Editing with Prior-guided Editing Field”. In: *The IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*. 2023 (cit. on p. 1).
- [2]Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. *Text2LIVE: Text-Driven Layered Image and Video Editing*. _eprint: 2204.02491. 2022 (cit. on p. 1).
- [4]Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. “Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023 (cit. on p. 1).
- [5]Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *ACM Trans. Graph.* 41.4 (July 2022). Place: New York, NY, USA Publisher: ACM, 102:1–102:15 (cit. on p. 1).
- [6]*nerfstudio-project/viser*. original-date: 2022-11-16T00:13:51Z. Mar. 25, 2024 (cit. on p. 22).
- [9]Matthew Tancik, Ethan Weber, Evonne Ng, et al. “Nerfstudio: A Modular Framework for Neural Radiance Field Development”. In: *ACM SIGGRAPH 2023 Conference Proceedings*. SIGGRAPH ’23. 2023 (cit. on p. 1).
- [12]Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. “CLIP-NeRF: Text-and-Image Driven Manipulation of Neural Radiance Fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 3835–3844 (cit. on p. 1).
- [13]Qiling Wu, Jianchao Tan, and Kun Xu. *PaletteNeRF: Palette-based Color Editing for NeRFs*. _eprint: 2212.12871. 2022 (cit. on p. 1).
- [14]Brent Yi. *brentyi/tyro*. original-date: 2021-10-05T08:54:08Z. Mar. 26, 2024 (cit. on p. 25).
- [15]Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, et al. “NeRF-Editing: Geometry Editing of Neural Radiance Fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 18353–18364 (cit. on p. 1).

Webpages

[@3] *daisyUI Tailwind CSS Components (version 4 update is here). URL: <https://daisyui.com/> (visited on Mar. 25, 2024) (cit. on p. 22).*

[@7] *React. URL: <https://react.dev/> (visited on Mar. 25, 2024) (cit. on p. 21).*

[@8] *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. Nov. 15, 2020. URL: <https://tailwindcss.com/> (visited on Mar. 25, 2024) (cit. on p. 22).*

[@10] *tRPC - Move Fast and Break Nothing. End-to-end typesafe APIs made easy. | tRPC. URL: <https://trpc.io/> (visited on Mar. 25, 2024) (cit. on p. 23).*

[@11] *Vite. URL: <https://vitejs.dev> (visited on Mar. 25, 2024) (cit. on p. 21).*

List of Figures

4.1. Flow Diagram	15
4.2. Excerpt of a View from the Flow Diagram with detailed interactions	16
4.3. Dashboard	17
4.4. Processing Input Data	18
4.5. Training Section	19
4.6. Viewer Section	20
5.1. System Architecture	21

List of Tables

Colophon

This thesis was typeset with $\text{\LaTeX}2\epsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at <http://cleanthesis.der-ric.de/>.

