UNIVERSITY OF CAMBRIDGE
Department of Computer Science and Technology

Machine Visual Perception
Course Project Report

Text-Guided Dynamic Gaussian Splatting Editing via Diffusion

Authors:
Samuel Mashil (sm2712)
Nicolas Seyedzadeh (ns896)
Robert Woodland (rbw32)

Group number: 5

Chapter 1: Introduction and Motivation

Section 1.1: Introduction to the problem

[Provide a thorough introduction to the problem and why it is important. Briefly explain what general techniques there are and how your project fits.]

This project uses Gaussian Splatting (Kerbl, et al., 2023), a novel technique for efficient 3D scene rendering, to create a tool inspired by Instruct-NeRF2NeRF (Haque, et al., 2023). By reconstructing a video as a Gaussian splatting model, we can apply Stable Diffusion on images rendered from the reconstruction and reconstruct these new images to get a new edited 4D scene. Frames can then be rendered to provide new perspectives on the edited scene, or to reconstruct the path of the original video.

Whilst Instruct-NeRF2NeRF achieves high-quality results in this field using NeRFs to reconstruct the scene, this comes as a double-edged sword, where rendering new images is much slower and is still far away from being used in real-time applications, even as the technology matures.

Gaussian splatting offers a promising alternative due to its faster rendering capabilities, whilst still preserving image quality. This project explores whether this substitution can improve the performance and usability of such systems.

Section 1.2: Background and related work

[Include a few very relevant related works and how your work relates to those, expanding on the previous section. We do not expect you to cover all previous works.]

Gaussian Splatting (Kerbl, et al., 2023) - Gaussian splatting is a technique by which volume data for a scene (data representing a scene as a 3D point cloud) can be rendered as a collection of 3D Gaussian primitives, rather than the usual surface or line primitives used for typical rendering. This rendering technique, being much faster than traditional NeRF (Neural Radiance Field) models, allows for real-time rendering and reconstruction, whilst still maintaining a comparable level of photorealism to NeRFs.

Instruct-NeRF2NeRF (Haque, et al., 2023) - This pipeline combines a text-guided image-to-image translation model (InstructPix2Pix) with NeRFs, enabling text-guided editing of 3D scenes. Whilst effective, the computational overhead for training NeRFs limits its practical use in scenarios requiring real-time feedback or processing of large datasets.

InstructPix2Pix (Brooks, et al., 2023) - A critical component of both the Instruct-NeRF2NeRF pipeline and our own project, this model enables text-based manipulation of input images using a pre-trained diffusion model. It serves as the interface for converting textual instructions into pixel-level modifications.

Section 1.3: Overview of the idea

[Provide an overview stating why the idea of the project makes sense and what the main motivation is.]

Given the rising interest in using AI for making labour-intensive tasks easier and the introduction of technologies such as Stable Diffusion and LLMs, there has been lots of research into the applications of such technologies across various fields. In the field of video editing, you may find that once a video has been made, you want to keep adding changes post-production. For humans, this is highly intensive and requires specialised knowledge, so the opportunity to apply modern machine learning methods to trivialise this task is something that has interested researchers in recent years.

The current state-of-the-art approach to this problem is Instruct-NeRF2NeRF, which continues training on the NeRF using InstructPix2Pix on rendered images. The details of this algorithm will be explored further in later sections, but we attempt to adapt the process to use Gaussian splatting in place of NeRFs, leveraging the impressive results obtained by Gaussian splatting in recent years to enable improved-performance, real-time, high-quality, text-guided video editing.

Chapter 2: Method

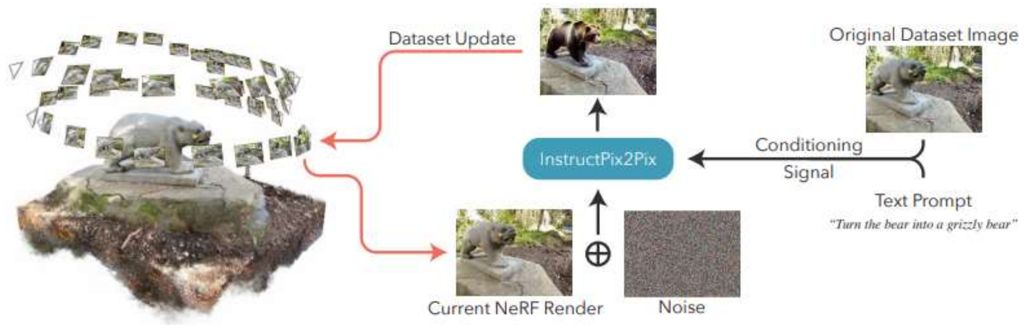
Section 2.1: Baseline algorithm

[Explain the baseline architecture you used to build your algorithm on. You may reproduce figures from the original papers.]

The baseline algorithm we are using is that of Instruct-NeRF2NeRF, which operates as follows:

- Split an input video into frames
- Apply COLMAP, a structure-from-motion pipeline, to the image frame dataset to obtain relative camera positions for each image
- Train a NeRF model on the original image dataset, producing a NeRF representation of the original scene
- Render images of the scene from given camera positions using the model
- Use InstructPix2Pix to apply text-guided modifications to these images, using the original image from said perspective as a conditioning signal (to keep the new image closer to the original even after several iterations)
- Add the edited image(s) back into the original dataset, and continue training the NeRF model on it
- Render new images from the NeRF model and repeat the process

The following figure, used in their paper, illustrates this pipeline:



Section 2.2: Algorithm improvements

[Explain what you implemented to improve over the baseline. You may include figures to explain the idea and logic. Focus on the ideas and not the implementation.]

We used Instruct-NeRF2NeRF as more of a guide than a foundation for our algorithm by carefully considering the pipeline that they used to enable editing of a reconstructed scene using text instructions.

To improve on the existing framework, Instruct-NeRF2NeRF, we perform the following:

- **Replacing NeRF with Gaussian Splatting:** Adapting the rendering pipeline to accept Gaussian primitives.
- **Increasing the fluidity of the pipeline:** By incorporating COLMAP binaries and FFmpeg into the pipeline, we can create a much more streamlined and flexible pipeline. This allows us to simply upload a single video file with a prompt, unlike the more sophisticated pre-processing required for Instruct-NeRF2NeRF.
- **Batch editing images:** Instead of iteratively updating each image, we run them in batches, thus speeding up overall processing time.

Section 2.3: Implementation details

[Explain how you implemented the improvements. You may include code snippets with the corresponding explanations.]

Frameworks and tools:

- **Gaussian Splatting:** Adapting the code from the original paper on Gaussian splatting (Kerbl, et al., 2023), we added additional processing to perform the text-guided edits.
- **Stable Diffusion:** We tried using both RealDream12 and InstructPix2Pix Stable Diffusion models. We found that RealDream12 is better at manipulating individual objects in the scene (e.g., adding/removing objects), whilst InstructPix2Pix is better at applying more general prompts to the whole scene (e.g., changing the weather or a texture in the scene). RealDream12 was, however, quite sensitive to hyperparameter tuning, with small changes to parameters such as the strength value of the original image in the edited image affecting the output image greatly. We decided to stick with InstructPix2Pix as it seemed more reliable and consistent, so long as the limitations are understood.
- **COLMAP:** We used COLMAP, a widely-used structure-from-motion pipeline, in order to deduce depth information from our images, such that camera positions for each picture relative to each other could be determined.
- **FFmpeg:** We used FFmpeg, a common tool for video manipulation, to extract frames efficiently from our video input files.

Challenges addressed:

- Compatibility and dependency issues
    - We had several issues regarding COLMAP versioning and prerequisite libraries. Conda was an attempted solution, but between the notebook and the libraries being unable to be found by Conda, it was decided that it was best to just install the dependencies directly in the Colab environment.
- Updating the dataset when running diffusion
    - In the Instruct-NeRF2NeRF pipeline, it is made clear that they use the new edited images to update the NeRF, however there is no obvious way to add images to a Gaussian splatting model, and so retraining the model with the new images was decided to be the best option.
- Choosing the size of the batch of images to update with InstructPix2Pix at a time

    - This was a trade-off between the advantages of the larger iteration numbers, which enforced consistency and made convergence more careful, and having concern for the efficiency of the model, given that the bulk of the work was in model training, which takes longer with further iterations.
- Parallelisation of editing
    - By editing several images concurrently, we were able to see noticeable improvements in the time required for our pipeline - improvements that became all the more noticeable with higher iteration counts.

Section 2.4: Data pipelines

[Explain your data format, how you consume the data in your algorithms, and data augmentation.]

1. The inputs to the pipeline are the video file and the editing prompt.
2. The data pipeline begins with taking the video sequence and preprocessing it to extract frames. The chosen frame rate must be high enough to keep temporal coherence between the images, but low enough that it doesn't produce too many frames to slow down reconstruction and editing.
3. This dataset of images is fed as an input to COLMAP. The COLMAP structure-from-motion pipeline determines and adds information about camera position and depth.
4. Next, we run diffusion on a subset of our images, based on our edit prompt and desired iteration count, replacing the originals with these new edited images.
5. Then, we train and produce a Gaussian splatting model on these edited images and any remaining original images.
6. We render the output of the Gaussian models, giving us the new reconstructed images at the same positions as our original extracted frames.
7. Assuming a desired iteration count > 1, we repeat from Step 4 until each image has been edited at least once.
8. Output the final edited Gaussian splatting model.

Section 2.5: Training procedures

[Explain which framework and optimizers you use, how you implemented the training logic.]

The training for the Gaussian splatting model was performed through the Gaussian splatting module (Kerbl, et al., 2023). This meant that the optimisers used were the default optimisers from the Gaussian splatting paper. Regarding InstructPix2Pix, we used the pre-trained model by Tim Brooks that was used in the original paper (Brooks, et al., 2023). Finally, considering the whole pipeline as its own single model, the training logic consisted of retraining Gaussian splatting models based on the edits of the InstructPix2Pix model and repeating this process on the renders generated by the model.

We get the batch of images that we are going to edit using Pix2Pix using the following:

$$\bigcup_{i=1}^{\frac{length(dataset)}{n}} dataset[i \times n] \qquad \text{for n iterations}$$

Effectively, by taking every ‘nth’ image, we can edit all parts of the image so as to not leave, for example, one side unedited. Then, in the following iteration, we repeat the same, but counting from the next image in the data (i.e., starting from the 2nd image for the 2nd iteration, and taking every nth image),, such that the new batch of images consists of images we haven’t yet edited. By the end of the training process we have a diffusion model in which each frame was at one point edited and has been integrated into the model.

Section 2.6: Testing and validation procedures

[Explain which framework you use, how you implemented the testing/validation logic.]

Given that there is no ground truth for this model, since we are editing features with diffusion that were never originally there, there is no test set with which we can evaluate the model against. This means that our testing and validation comes mainly from our subjective view of the photorealism of the final rendered output, given the original video and the input prompt.

Chapter 3: Experiments and Evaluation

Section 3.1: Datasets

[Explain the datasets utilized: what they contain, why they are utilized, assumptions, limitations, possible extensions.]

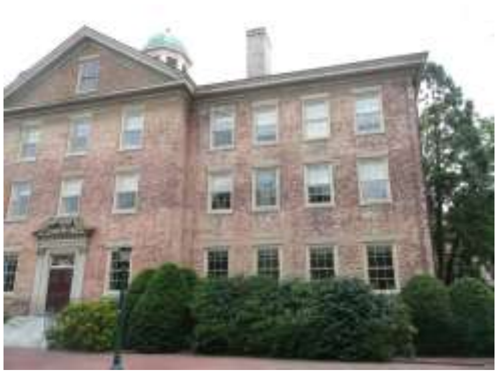
As part of testing our method, we have used various sources of videos. These have included a selection of free-to-use videos from sites such as Pexels, as well as datasets provided by COLMAP, hosted at demuc.de (Schönberger, n.d.). Specifically, we used a video of a shipwreck from Pexels for a lot of the initial COLMAP testing, given the smooth camera movement and visibility of all angles, before moving onto images of the “South Building” from the COLMAP datasets, as it wasn’t too large, worked well with COLMAP, and gave ample room for possible editing.

An example original image from the South Building dataset

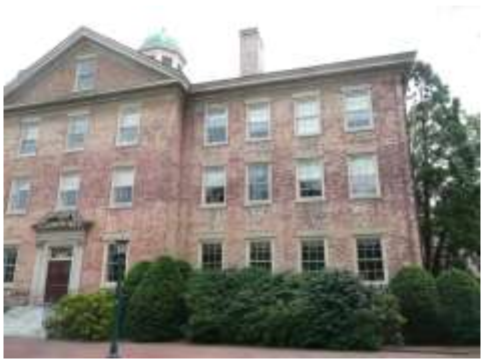
Section 3.2: Training and testing results

[Explain the training and testing results with graphs and elaborating on why they make sense, what could be improved.]

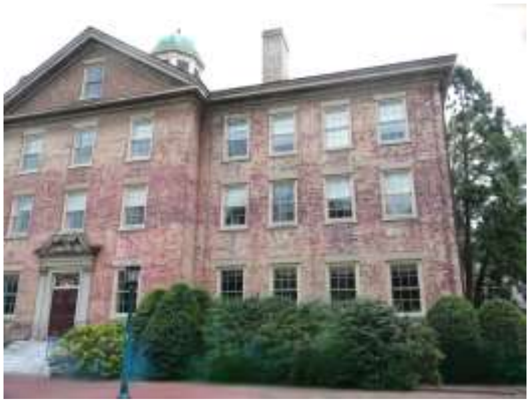
Result after the first iteration:



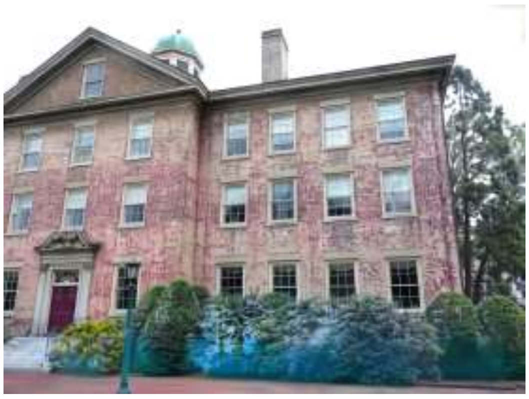
Result after the second iteration:



Result after the third iteration:



Result after the fourth iteration:



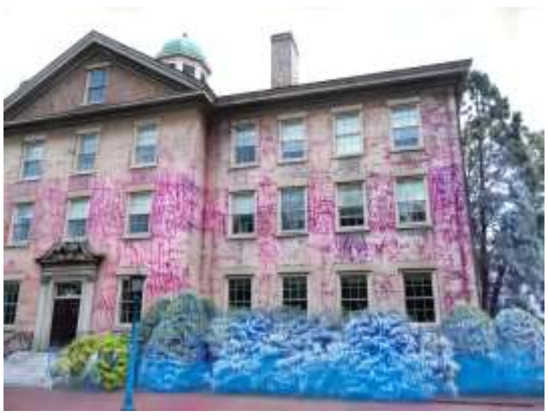
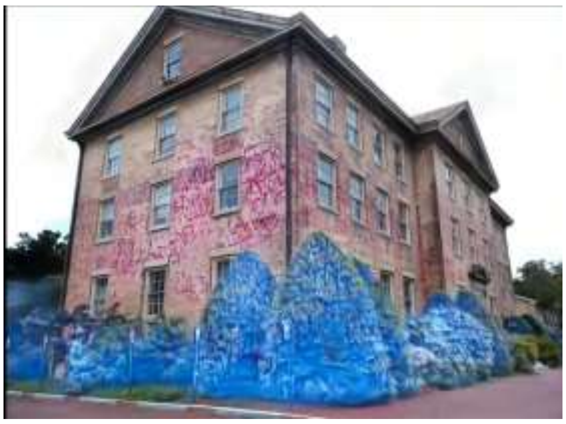
Here, we see the result of each iteration from a single viewpoint after running the algorithm with the prompt “add graffiti to the house”. We see that at the beginning very minor changes are made to the wall, it's slightly more red. On the following two images, we see the red develop and some more graffiti added. In the final image, the graffiti is much clearer. We also see some limitations of InstructPix2Pix here as it starts to edit the bushes as well as the house, displaying its inability to distinguish parts of the image.

We also saw a speedup on the image diffusion when we edited our images in parallel on each iteration. Editing each image individually, as is done by Instruct-NeRF2NeRF, came out at 24 seconds per image, but with a batch of 16, this process took 6 minutes and 6 seconds, or 22.875 seconds per image. It must be noted that this method is significantly more GPU intensive although we believe this to be reasonable given the rest of the pipeline. We also found a logarithmic decrease in the editing time with lower resolutions - a resolution of 963 x 720, for example, took just 1 minute and 54 seconds for 16 images (7.125 seconds per image), with less than ten times the pixel count of our original images (at resolution 3072 x 2304).

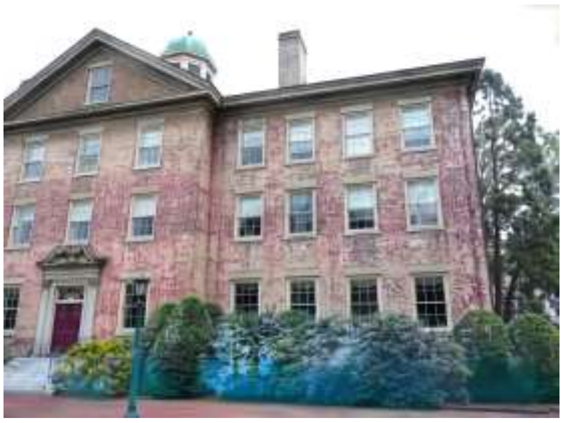
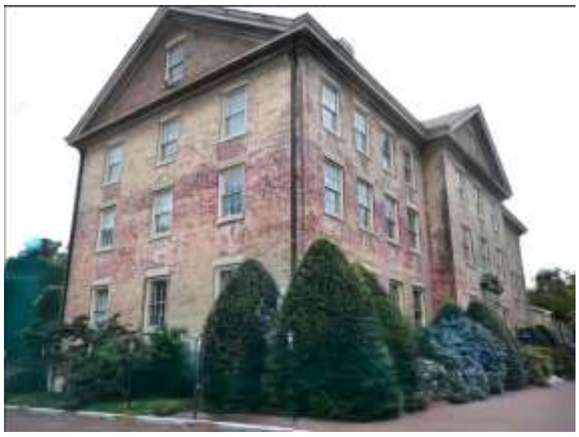
Section 3.3: Qualitative results

[Show in figures and explain visual results. Include different interesting cases covering different aspects/ limitations/ dataset diversity. If not converged, explain what we can expect once converged. Include any other didactic examples here.]

Non-Iterative:

Iterative:

Running our pipeline iteratively produces a less visually different, yet more accurate, result. This is because edits performed to images are spread across the set of output renders of the previous iteration's Gaussian model, and so the changes are built up slowly rather than being applied at full intensity at once across the whole model.

Here, for the iterative result, we ran InstructPix2Pix on every fourth image in the dataset, and resampled our images from a new Gaussian splat each time. Were we to have more time on this project, we would try altering the amount of iterations and the "batch size" (number of images we change between iterations). We could run InstructPix2Pix on each image twice, for example, so as to achieve the accuracy of the iterative approach with the bigger visual difference of the non-iterative approach. As such, we can also vary the 'batch size' in order to make the iterations converge slower and, potentially, with more regularity.

Chapter 4: Conclusions and Future Directions

Section 4.1: Conclusions

[Summarize what the project was about and the main conclusions.]

This project demonstrated the potential of Gaussian Splatting as an alternative to NeRF for text-guided 4D editing. The results validated that our approach achieved faster rendering times, without compromising visual fidelity, in turn validating the idea that Gaussian Splatting could be a higher quality reconstruction model for this task compared to the use of NeRF in the current state-of-the-art model, Instruct-NeRF2NeRF. With more time to spend on optimizations and design choices, this pipeline also has significantly more room to grow than Instruct-NeRF2NeRF.

Section 4.2: Discussion of limitations

[Explain the limitations of your technique. You may want to refer to previous sections or show figures on the limitations.]

The first limitation was the very high GPU memory requirements - with a high-quality paid Colab GPU with 40 GB of VRAM, we were still not free from memory issues when trying to diffuse images in batches and were required to make quality sacrifices, like scaling down the resolution, especially during testing. The code was also simply not usable with a smaller GPU like the one we started with, which had 15 GB of VRAM. Regarding computational time, even with large quality sacrifices, the pipeline is still not suited to rapid development, even if it is an improvement over Instruct-NeRF2NeRF, which is likely a prerequisite for large-scale adoption of the technology.

We also have limitations from our camera parameter registration library, COLMAP. This choice was made mainly as a product of the viable inputs into our Gaussian splatting model and wasn't a huge focus of the pipeline, so it wasn't explored in depth in this report, despite being a vital piece of the puzzle when starting with just a set of images. COLMAP has issues regarding the resilience of its registration. Specifically, videos are limited to those where a camera is slowly circling a stationary object, and deviations from this tend to just be thrown out by the registration program. This means that arbitrary scene editing is not always possible using our method. Additionally, the limited efficiency of COLMAP meant that we couldn't reasonably re-run it after each iteration, leading to ghostly artifacts related to the new, changed depth information of edited images.

Finally, we inherit the limitations of the InstructPix2Pix model, which includes the difficulties of translating a 2D edit into a 3D scene. This issue was faced by Instruct-NeRF2NeRF, where editing textures, replacing objects and changing global properties work well in the model, but adding or removing entire objects is a struggle, translating into non-realistic artifacts in the Gaussian-splatted or NeRF models. This leads to particular difficulties when trying to change isolated parts of the scene. Similarly, we also have a fundamental problem that was also faced by Instruct-Nerf2NeRF when trying to use InstructPix2Pix, which is that inconsistencies between edited images lead to larger problems, as the Gaussian splatting model sometimes failed, or was poor at consolidating these images into a 3D scene.

Section 4.3: Future directions

[State a few future directions for research and development. These typically follow from the discussion on limitations.]

The main improvement that we can see will come from performance improvements of the algorithm. A lot of the bottlenecks during the process of our research came from memory or time constraints. This meant that we couldn't run our implementations at higher resolutions, with longer videos, or with more iterations, without training times ballooning to unreasonable orders of magnitude for most applications. In order to make the pipeline viable for real applications, the speed of the process will have to start approaching minutes instead of hours. Ways of approaching this performance improvement are currently being researched, for example, approaching Gaussian splatting with Levenberg-Marquardt optimisation (Höllein, et al., 2024). Work in the future should, as such, be conscious of the current landscape regarding the most modern techniques for approaching the problem, whether hardware or software.

Secondly, in future work regarding our camera parameter registration, COLMAP, different models can be considered for this process based on their resilience to camera movement, as well as time and space efficiency, and compatibility with current Gaussian splatting models. A better camera parameter registration algorithm would be very valuable, as by default, users will have an input in a video file format, likely taken themselves, and so the conversion of the file into a format valid for the rest of the code is as much a part of the pipeline as everything else.

Finally, the InstructPix2Pix limitation can be addressed by looking into alternate models, such as the aforementioned RealDream12, that also perform this text-guided image-to-image translation, but potentially without the limited ability to add and remove parts of the image. Similarly, this limitation can also be overcome by looking into current methods that will isolate editing to only certain parts of the image by incorporating semantic and motion-based masks with an inpainting model to hopefully provide a higher fidelity image to the reconstruction based on the prompt. Further work could also look into the conditioning images used by Instruct-NeRF2NeRF in their extension of the InstructPix2Pix algorithm. This may add a grounding to the images to stop them becoming too different to the originals especially when experimenting with higher iterations.

References

Brooks, T., Holynski, A. & Efros, A. A., 2023. *InstructPix2Pix: Learning to Follow Image Editing Instructions.* [Online]
Available at: https://arxiv.org/abs/2211.09800
[Accessed 06 December 2024].

Haque, A. et al., 2023. Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.* s.l.:s.n.

Höllein, L., Božič, A., Zollhöfer, M. & Nießner, M., 2024. *3DGS-LM: Faster Gaussian-Splatting Optimization with Levenberg-Marquardt.* [Online]
Available at: https://arxiv.org/abs/2409.12892
[Accessed 06 December 2024].

Kerbl, B., Kopanas, G., Leimkuhler, T. & Drettakis, G., 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*.

Schönberger, J., n.d. *COLMAP Datasets.* [Online]
Available at: https://demuc.de/colmap/datasets/
[Accessed 06 December 2024].