The report should be done as a Jupyter Notebook. The report should be a complete description of the objectives of the work, the methods used to solve the problem, experimental evidence of a working system, the code, and clear delineation of what you have done vs. what you are leveraging that others have done. If you have used the work of others YOU MUST INCLUDE ATTRIBUTION by citing this work inline and as part of a "bibliography" at the end. You should describe what worked, what did not, and why. If you are working in a group you need to submit your own report and this report should be clear about what your individual contribution was. It is ok to include your collaborators work in your report, but you must be clear about your section and write this yourself. This project report constitutes a large fraction of your final grade–take it seriously and include enough material and details for us to give you a good grade. If you are having trouble imagining the structure of the report, refer to published research papers in CV as a possible model.

try citing something (Park et al., 2020)

```
<center><b>Automatic Remastering Of Classic Franchise Video Games</b></center>
```

```
<center><b>Zijie Zhu (zz2973)</b></center>
```

Introduction

The 21st century has seen many great video games born. Although some of them are standalone works published by boutique studios, such as Braid, most of the masterpieces are parts of their corresponding franchise series from established game companies, such as Halo, Call Of Duty, and Hitman, etc. As many gamers noted, however, making an excellent sequel often seems even harder than creating an exceptional initiation. Multiple series have introduced games with groundbreaking graphics yet mediocre storylines or play styles, so disappointed players would rather go back to previous installments which lured them into the franchise and brought them immense joy. However, only then would they realize that their previously experienced graphics already seem unacceptable by current standards, and they end up in a dilemma between good graphics and good content.

From time to time, video game companies does release a remastered / remade / rebooted version of their previous works to bring back players into the franchise, such as Halo: The Master Chief Collections (MCC). But such work is not always cost-benefit efficient from the perspective of the game companies, so they make fewer of these than new games or simply charge the same price as a new game. One good example is the uproar among gamers in early 2022 caused by Naughty Dog charging a full price of $70 for a remake of The Last Of Us Part I. Therefore, it would be great if there is a procedure to automatically enhance the graphics of previous installments to current-generation levels. Such a procedure would be highly useful for both the game companies which can cheaply publish remastered versions and the gamers who can create graphics mods on their own. In this

project, I will work on prototyping parts of such a procedure using Halo: Combat Evolved and Halo 2, two highly popular and classic first-person shooting games.

Related Work

The problem at hand concerns video-to-video translation, where we want to translate the original game to an updated game or a totally different game. Naturally, video-to-video translation is based on image-to-image translation with potential improvements on scene coherence and an emphasis on fast prediction speed so the video can preserve high frames-per-second (FPS). One might think this particular translation is paired, since the original game and the remastered game have highly similar content. However, it is naturally hard to play the two games in the same precise way, and even the non-play-controlled cutscenes may take different camera angles. Therefore, the approaches we visit here focus on unpaired image-to-image translation.

One important dataset for benchmarking studies in this area is GTA→CityScapes, where in-game rendered driving data from GTA V are compared with real-world street recordings. Notable previous works utilizing this dataset include Cycle-Consistent Adversarial Networks (CycleGAN), Contrastive Unpaired Image-to-Image Translation (CUT), and Enhancing Photorealism Enhancement (EPE). There are also previous important works on unpaired translation and style transfer, such as Style Transfer by Relaxed Optimal Transport and Self-Similarity (STROTSS) and Whitening and Coloring Transforms (WCT), but they are older approaches with higher runtime that would result in low FPS, thus not suitable for our discussion. Since CUT is a sequel of CycleGAN with better performance and lighter architecture (thus faster prediction and higher FPS), we will ignore CycleGAN and focus on CUT and EPE in the Methods section later.

Data Curation

Getting and Cleaning Videos

I get all data from YouTube where gamers publish their campaign walkthrough videos. Such walkthrough videos are usually long (i.e. more than four hours) and have high resolution (i.e. at least 720p) and some have high frame rate (i.e. 60 FPS). Notice that our game usually consists of the single-player campaign and the multiplayer

There are several considerations in the data cleaning part, most of which are achieved with `ffmpeg`: 1. Initially, I aimed for the highest resolution of each video, such as 1080p or 4K. However, I could only get up to 7.5GB of GPU memory size on Google Cloud VM, and high-resolution dataset always quit with CUDA out-of-memory error. Sicne 480p is the highest resolution that the model could train, I had to settle with this resolution for all videos. Naturally, frame rate did not face the same problem and I preferred 60-FPS videos if available.

1. Certain gamers focus more on audience entertainment and lingered at the same location in game for too long, so I also picked walkthrough videos with no commentary when possible, as such videos usually focus more on gameplay. 1. To reduce noise in game scenery data, I also avoid videos with subtitles and gamer webcam. If all available videos have subtitles, I crop out the bottom area of the video containing the subtitles. 1. The opening and closing credits are all trimmed out to focus on gameplay. 1. The campaign gameplay data may still consist of cutscenes, which are argueably different from regular gameplay scenes. However, they are much harder to detect and clean out, and are relatively consistent with in-game data, so I decided to keep them.

Sampling Videos to Images

Since the videos are generally more than five hours long, we have ample data to train the model, but we need to find a balance between dataset size and training speed. The default parameters for the original CUT model are suited for training its `grumpy cat` dataset with around 300 images in both the original and the transferred domain. To assemble a dataset of similar sizes, I started with sampling videos once every minute. As I will elaborate below, however, I later moved on to sampled videos once every second.

Methods

- CUT / FastCUT
-

CUT

Enhancing Photorealism Enhancement (EPE)

Enhancing Photorealism Enhancement by Intel Labs where the researchers used deep learning techniques to make GTA V look even more photorealistic. The researchers achieved this by training their neural networks on real, annotated cityscape data like KITTI, CityScapes, and Mapillary Vistas. The neural networks consist of a G-buffer encoder, an image enhancement network, and a perceptual discriminator

Why not: 1. G-buffer hard to record and collect. Also doubtful on its in-game performance, though technically it is generated by in-game deferred rendering. 2. Robust segmentation does not work well on Halo game scenes, such as the Arbiter being recognized as various different objects.

my workflow would be the following, using Halo as an example: 1. Treat the remastered Anniversary version of Halo 2 as the “reality” dataset and use the G-buffer encoder or other open source / commercial solutions to annotate it 2. In comparison to the problems faced by the Intel Lab researchers, I have the advantage of the fact that the original Halo 2 and the remastered Halo 2

have exactly the same content and only different graphics rendering. Therefore, I do not need video annotations that are too accurate, since the content are already lined up. With that being said, I am also looking for other papers and methodologies that can actually leverage this advantage of content mapping. 3. I will use the paper's methodology to remaster the original Halo 2, and compare the results with the actual remastered Halo 2. Of course, the expectation is to have reasonable improvements rather than beat the actual remastered version. 4. More interestingly, I can then try to remaster Halo 1 using the networks trained on Halo 2 comparisons, and compare the results with the actual remastered Halo 1.

Experiments

Metrics and Evaluations

Other Applications

SinCUT for pre-game styling

Future Improvements

Since my time and budget are limited and I do not have a teammate, I may not complete all steps above or have impressive results, but I will try my best to accomplish at least something that shows visually interesting work. As I continually search for better references and methodologies, I may also have revisions on my final project that are different from this proposal.

Conclusion

Bibliography

```
!jupyter nbconvert --to markdown report.ipynb
!pandoc -s report.md -t pdf -o report.pdf --citeproc --bibliography="list.bib" --csl="apa.cs
```

```
[NbConvertApp] Converting notebook report.ipynb to markdown
[NbConvertApp] Writing 9873 bytes to report.md
```

```{bibliography}
```


Park, T., Efros, A. A., Zhang, R., & Zhu, J.-Y. (2020). Contrastive learning for unpaired image-to-image translation. *European Conference on Computer Vision*.