Stable Remaster: Bridging the Gap Between Old Content and New Displays

Nathan Paull
napaull@utexas.edu

Shuvam Keshari
skeshari@utexas.edu

Yian Wong
yian@utexas.edu

Abstract

The invention of modern displays has enhanced the viewer experience for any kind of content: ranging from sports to movies in 8K high-definition resolution. However, older content developed for CRT or early Plasma screen TVs has become outdated quickly and no longer meets current aspect ratio and resolution standards. In this paper, we explore whether we can solve this problem with the use of diffusion models to adapt old content to meet contemporary expectations. We explore the ability to combine multiple independent computer vision tasks to attempt to solve the problem of expanding aspect ratios of old animated content such that the new content would be indistinguishable from the source material to a brand-new viewer. These existing capabilities include Stable Diffusion, Content-Aware Scene Detection, Object Detection, and Key Point Matching. We were able to successfully chain these tasks together in a way that generated reasonable outputs, however, future work needs to be done to improve and expand the application to non-animated content as well.

1. Introduction

The way we perceive content has been revolutionized by the rapid progress of modern displays. From stunning 4K nature documentaries to sports broadcasts that provide such clarity that anyone can act as a referee, modern displays have enhanced the viewing experience. However, older content developed for CRT or early Plasma screen TVs has become outdated quickly and no longer meets current aspect ratio and resolution standards, resulting in a less enjoyable re-watching experience of beloved shows. Fortunately, we can solve this problem with the use of diffusion models to adapt old content to meet contemporary expectations.

Although Stable Diffusion [11] has gained popularity for image generation, its application to video often faces a challenge of temporal continuity. This poses a significant issue for aspect ratio expansion as the expanded content typically comprises static backgrounds. Our proposed project aims to address this limitation by utilizing the static spatial bias to govern the generation of new content and ensure that multiple instances of the same background region are not produced. By doing so, we can overcome the issue of temporal continuity in video generated using Stable Diffusion, thereby enhancing the quality and coherence of the output.

To achieve this goal, we will utilize a novel approach that combines Stable Diffusion with machine learning techniques. Specifically, we will use a machine learning model to identify and extract the static background regions from the input video. These regions will then be used as a reference to generate new content that preserves the temporal coherence of the video. Additionally, we will explore the use of other techniques such as motion estimation to further improve the quality of the output.

2. Related Work

2.1. Modernizing Video Techniques

In recent years, there has been a growing interest in modernizing video through techniques such as super-resolution [8, 10, 13], colorization [18, 19], and changing aspect ratio via outpainting [5, 14].

Super-resolution techniques aim to increase the resolution of videos beyond their original quality. Liu et al. [10] proposed a Bayesian approach to adaptive video super-resolution, which estimates motion, blur kernel, and noise level while reconstructing high-resolution frames. This approach achieved promising results that can adapt to various conditions. Shi et al. [13] used an efficient sub-pixel convolution layer for real-time super-resolution of 1080p videos on a single GPU, improving performance and reducing computational complexity compared to previous CNN-based methods. Kappeler et al. [8] proposed a CNN for video super-resolution that combines both spatial and temporal information, achieving state-of-the-art results with a relatively small video database for training.

Colorization is the task of coloring a grayscale video. Yatziv et al. [18] proposed a computationally efficient method for colorizing grayscale images and videos using luminance-weighted chrominance blending and fast intrinsic distance computations, resulting in high-quality outputs with reduced computational cost and user interaction. Zhang et al. [19] introduced an end-to-end network for video colorization that addresses the challenge of achieving

temporal consistency while remaining faithful to the reference style. Their approach uses a recurrent framework that unifies semantic correspondence and color propagation steps, producing superior results compared to state-of-the-art methods.

Aspect ratio conversion is an evolving task, where older videos are adapted to fit on more modern devices with different aspect ratios. Guo et al. [5] proposed a method for converting video aspect ratios using a saliency model to determine regions of interest and applying a novel cropping and expanding mode to maintain visual quality and avoid distortion. Soe et al. [14] presented an idiom-based tool for video retargeting that allows users to control cropping and panning with selected cinematic idioms to achieve an optimal viewing experience on different platforms. However, these methods focus on cropping rather than generating new regions of the video to account for the aspect ratio change, and to our knowledge, this is one of the first works to use generative machine learning for this task.

2.2. Background Collapsing and Stitching

Background collapsing and stitching techniques are essential in image and video processing for tasks such as background removal, scene extension, panorama creation, and video retargeting. These techniques provide visually consistent and seamless results while maintaining the integrity of the foreground objects.

2.2.1 Background Collapsing

Background collapsing involves identifying and reducing redundant background regions in images or videos, allowing for the preservation of important foreground elements while resizing or retargeting. This technique often employs saliency maps or object detection algorithms to determine the importance of different regions in an image or video frame.

One prominent method for background collapsing is seam carving [1], which involves removing or inserting pixels along optimal seams to resize images while maintaining the essential content. Seam carving has been further extended to videos by Rubinstein et al. [12], who introduced a method for video retargeting that reduces or expands background regions while preserving the overall content and temporal coherence.

Another approach for background collapsing is patch-based image quilting [4], which synthesizes textures by sampling patches from the input image and stitching them together in a visually consistent manner. This method has been extended to videos by Kwatra et al. [9], who introduced an algorithm for video texture synthesis using a graph-based approach to synthesize temporally coherent video textures by stitching together small spatiotemporal patches from the input video.

2.2.2 Background Stitching

Background stitching techniques are used to combine parts of images or video frames to create a seamless and visually consistent output. These methods are essential for tasks such as panorama creation, video compositing, and background extension.

One common approach for background stitching in images is feature-based alignment [2], which matches key points between overlapping regions of images and computes the transformation matrix to align and stitch the images together. This method has been further extended to videos for creating panoramic video sequences by Szeliski [15].

Another technique for background stitching in videos is the content-aware video retargeting method proposed by Wang et al. [17]. This method employs a patch-based optimization approach to generate an output video with the desired target aspect ratio by stitching together patches from the input video while maintaining foreground object proportions and minimizing distortion.

Background collapsing and stitching techniques play a crucial role in image and video processing tasks. These techniques allow for the adaptation and enhancement of visual content while preserving the integrity of foreground objects and maintaining overall visual consistency. Advances in these techniques continue to improve the quality and versatility of image and video content for display on various devices and platforms.

2.3. Stable Diffusion and Video Generation

Image synthesis is a rapidly evolving field in computer vision, but it also has significant computational demands. Diffusion models (DMs) achieve state-of-the-art synthesis results on image data and beyond by decomposing the image formation process into a sequential application of denoising autoencoders. Robin et al. [11] presented latent diffusion models, which significantly improve the training and sampling efficiency of denoising diffusion models without degrading their quality.

Dehan et al. [3] presented a method for video outpainting by converting a portrait (9:16) to landscape (16:9) video using background estimation, segmentation, inpainting, optical flow for temporal consistency, and image shifting to improve individual frame completions. They evaluated their method on the DAVIS and YouTube-VOS datasets. Ho et al. [6] introduced a method to generate high-definition videos using a base video generation model and a sequence of interleaved spatial and temporal video super-resolution models. Jin et al. [7] proposed a framework that can retarget the old video screen ratio to a wider target aspect ratio hor-

izontally while preserving the quality of the objects using segmentation and inpainting networks. Relocating objects, especially those at the edges of the frames in the video, can be challenging.

These recent advancements in modernizing video techniques, stable diffusion, and video generation show promise in improving the visual quality and compatibility of older videos for display on modern devices.

3. Methods

3.1. Dataset

Due to the nature of our project being the orchestration of several computer vision capabilities, we did not require labeled data, instead, we needed an animated video with an old aspect ratio. For our primary dataset, we have gathered videos from the animated television show 'Avatar the Last Airbender'. We have chosen this show as it has an aspect ratio of 4:3 and is popular enough to be recognized by many individuals. The image below shows how a single frame in the 4:3 aspect ratio would be displayed on many modern devices. These large black bars on the sides of the content obviously detract from the user's watching experience.

Figure 1. Sample frame in 4:3 aspect ratio showing large sidebars.

These large black bars are the areas we seek to fill in with content that does not distract from the existing frame but instead creates a more immersive experience. We plan to verify the accuracy/quality of the final video file generated by manual inspection as our goal is to assess human experience and immersion. As such if anything seems out of place or violates environmental rules set by the animator it would be deemed incorrect.

3.2. Overview

For this project, we have created a pipeline for expanding the aspect ratio of a given video which we have divided into 5 tasks which are shown in Figure 2.

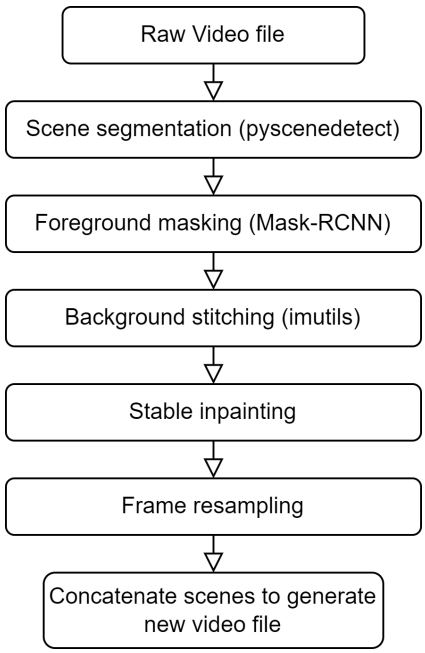

Figure 2. Flowchart for the process showing the used Python libraries in brackets

As we can see in the image the pipeline stages are Scene Segmentation, Foreground Masking, Background Stitching, Background Outpainting, and Frame Resampling. We have further broken down Background Outpainting into two sections below (Outpaint Region Selection and Background Outpainting) in spite of them being treated as one task in our code. This is in part to discuss our methods in comparison to related work and due to the fact that these two tasks are semantically different even if it doesn't make sense to separate them in the pipeline itself.

We felt that the tasks in the pipeline were key tasks due to the following assumptions about animated content: Background within a scene is constant and maintains object permanence, camera motions generally follow affine transformations, and content at the edges of a frame will be background content. These assumptions allow us to streamline computation of this task, letting us process scenes independently, only try to generate background pixels, and stitch backgrounds together simply. There are likely some flaws in these assumptions or a set of assumptions that should be included with these above which we will discuss below in the results section as we identify shortcomings with our methodology.

3.3. Scene Identification and Segmentation

For scene identification and segmentation, we chose to use PySceneDetect's Python API as it could perform the scene segmentation and save output scenes to mp4 rather than just returning frame indexes within the parent mp4 file.

We felt that this was key in any future work around parallelization as all tasks after this one in the pipeline can be run on scenes in parallel, sharply decreasing the overall runtime for expanding an episode.

When PySceneDetect is compared to alternatives such as SceneCutExtractor, and MatLabSceneDetection, and writing our own scene detection using functions within the OpenCV python API, our selection was quickly narrowed down to PySceneDetect and SceneCutExtractor. First, we felt the Python API offered by both of these libraries was crucial to making the pipeline easy to use and edit. Additionally, the ability to use a prepackaged library gave us much more time to work on the pipeline itself rather than focusing on just a single task. In our research, PySceneDetect was preferable to SceneCutExtractor because of its ability to save scenes to mp4, while SceneCutExtractor would save JSON or CSV files with frame indexes and evaluations. While this may provide more flexibility, we found the ease of use for PySceneDetect to be much more attractive.

Within PySceneDetect we used the content-aware scene detection which detects changes in the HSV color space to determine scene changes. Additionally, PySceneDetect uses ffmpeg to perform the scene cuts.

3.4. Foreground Masking

In this section of the pipeline, we make one further assumption, that objects in the foreground will first be rendered in full by the animator in addition the the assumption that objects in the background are permanent throughout the duration of a scene. This assumption motivates the idea of foreground masking so that we solely analyze the background when generating pixels.

To perform this masking we sought out a network that could recognize objects in the foreground, find bounding boxes for these objects (or masks if possible), and perform these two tasks at a relatively quick speed. We decided that bounding boxes was a minimum requirement as anything less than that would not allow us to run an algorithm such as GrabCut, however, if the method was able to generate masks on its own the need for GrabCut would be removed. These requirements led us to select Mask-RCNN, an object detection DNN built on a base ResNet Structure. Mask-RCNN not only detects a vast array of objects found in the COCO dataset but additionally generates masks for these objects and can be run at a speed of at least 5 frames per second on most GPUs. In our deployment on an Nvidia 2070 Super, we were able to get a speed of 7 frames per second.

With the selection of Mask-RCNN, all we had to do was to combine the masks of found foreground objects into a total mask that would separate the background from the foreground for the next stage in the pipeline, background stitching.

3.5. Background Stitching

The goal of background stitching was to create a total background for the scene. This is motivated by a common technique in the film used for creating semi-transparent characters. To accomplish this, two shots will be filmed sequentially, one with the character in the frame and one without the character in the frame. This allows editors to fill in the missing information with real information instead of generated information. Our goal was similar, to avoid using generated pixels wherever possible. This is because generating coherent pixels is computationally expensive and because we want to maintain pixels that exist in the original animation. If we generate the legs of a table on the boundary of a scene only for the camera to pan towards that table and have it disappear behind the original frame, we would immediately break immersion for the viewer.

Figure 3. Keypoint matching to extend the aspect ratio (generating a panorama: bottom) from two slightly different images as can be expected from consecutive frames (top 2 images) in a scene

Through this motivation for background stitching, we decided to use keypoint matching and affine transformation estimation. This method of keypoint matching and transformation estimation is common when generating panoramic photos from a set of distinct images. As such we chose to follow this method choosing SIFT as our method for keypoint detection and description. With these SIFT key points, we could then find a set of good matches to determine any affine transformations necessary for matching the set of images together.

Once we achieved a complete background we could then accurately determine which pixels can be sampled from information generated by the original animators and which pixels would need to be generated.

3.6. Outpaint Region Selection

Alongside the total background that we have constructed, we similarly construct a total mask. This mask

will contain information regarding which parts of the background have been filled with information derived from frames within the scene and which parts of the background lack any information. This total mask allows us to determine which regions of the total background will require generated pixels.

Pixels will only be generated if they fall within the bounds of the new frames, this means that each time we will sample this total mask within the region of the new frame to determine if new pixels need to be generated. If no new pixels are needed then the answer is to simply sample the total background and return. If new pixels are needed however we will use this sampled mask to inform any outpainting and then we will add these generated pixels to the total background and update the total mask so that these pixels will not be generated again. In doing this we decrease the number of pixels that must be generated.

3.7. Background Outpainting

For the task of outpainting, we will use the practice of Stable Diffusion [11] as implemented by the diffusers python library [16]. This library provides pre-trained models that can be used through a simple Python API. Specifically, we chose the Stable Diffusion Inpainting Pipeline offered by this library as it allows the user to input a mask where pixels should be generated and allows for the user to input a prompt that will describe the generated pixels. While optimization of the prompt would likely improve results, we decided to use the generic prompt of 'animated background' for all generated pixels in the hope that it would create reasonable generated pixels. These generated pixels are then added to the total background generated from background stitching which will adjust the Outpaint Region selected for the next frame within the same scene.

This step of the pipeline we found to be our most time-consuming taking up to 40 seconds per frame to generate pixels. This in part is why some of the previous steps are necessary. Without any of the previous steps, a 20-minute episode at 30 frames per second would take nearly 400 hours to process. While we cannot provide a tighter upper bound than this, the lower bound is much lower as there is no generation of duplicate pixels. This means that longer scenes create shorter runtimes as no duplicate pixels are generated. This could likely be further optimized by choosing the frames that experience large translation transformations relative to the first frame along a set of key directions.

3.8. Frame Resampling

After we have generated all necessary pixels to fill in gaps within the background the task of frame resampling is rather simple. We begin by using the affine transformation found in the background stitching step to transform our sampling region. We then simply select all pixels in the total background that are within this region. Following this we calculate the inverse of this affine transformation and use this inverse to transform our sampled pixels back into a viewable frame. We continue this for each frame until the scene has been completely reconstructed. We can now save these frames in a new scene and then concatenate all scenes together to create the reconstructed episode.

3.9. Experiments and Results

Our primary stated goal was to create a pipeline that could expand the aspect ratio of old animated content without violating the temporal coherency of the content. We believe that overall we were successful in this endeavor with the results shown below.

Figure 4. Comparison of the original frame (top) with resampled frame (bottom)

Figure 5. Comparison of the original frame (top) with resampled frame (bottom)

In both of the figures above we can see some similar results. The easiest to see is the shortcomings with Figure 4 showing black spots on the left side of the frame and Figure 5 adding Christmas trees and distorted penguins. Additionally, both images seem to have some color distortion and both possess a vertical gray line on the right-hand side of the image. We also must admit that the color distortion is not constant throughout the scene either and this can be seen in the gif results linked in the Demo section below. All of these shortcomings do bring us short of the goal of perfectly adapting old animated content for modern screens with modern aspect ratios.

However, only mentioning these shortcomings would be ignoring the parts of the expansion that the pipeline gets correct. If you specifically look at the floor in both figures you can see how the scene is properly expanded. In the second figure, the mounds of snow are continued to the boundaries of the frame quite well and show that while this method was not completely successful that it is quite close. In the Future Work section below we continue to discuss what we believe can be done to solve these shortcomings and solve the stated goal.

While we can see the effects of Stable Diffusion on the figures above, we would like to specifically discuss some results from individual steps earlier in the pipeline so that we can analyze which steps are creating the error we see in the final product. We will first discuss the Foreground Masking section of the pipeline.

Figure 6. False objects masked

As we can see in the figure above, Mask-RCNN seems to detect some false objects in the animated domain. Additionally, we can see in the figure below how Mask-RCNN can also fail to detect objects when animators break body continuity to better display character motion.

Figure 7. Foreground masking

Lastly, we can see how some of the object boundaries are very slightly off. While our method for background concatenation can help to limit some of this noise if several frames occur at similar camera positions, it is still possible for this noise to persist and affect certain frames.Overall these three results show that Mask-RCNN does not perform perfectly and adjustment or replacement of this model could lead to improved results at the output of the pipeline.

Finally, we would like to cover the results of the Background Stitching module. This module is what we believe to be the source of most of the shortcomings, especially the shortcomings related to color change around the edges of the image.

Figure 8. Sample Frame displaying edge effects on final output.

We believe that these off-color effects are due to the affine transformations performed in this step. In addition to these color effects, we also experience issues with key point matching on repetitive backgrounds where key points are harder to uniquely match. We believe that this and all other issues mentioned above are what lead to most lackluster results, but we are still excited by what all was in fact accomplished by this pipeline.

3.10. Demo

Our code is available as a GitHub repository named Stable Remaster. This Github provides instructions on how to set up the environment and run the demonstration code. The environment can be set up by creating an anaconda environment from the environment.yml file or by manually installing the libraries listed in that file. There is also the need to install ffmpeg for the full pipeline demonstration but this is not necessary for the scene-based demonstration. To run the full demonstration use the pipeline_demo.py file. To run the scene-based demonstration use the scene_demo.py file.

To view some example output from the scene_demo.py demonstration please visit the following link. Here we have two gifs showing expanded scenes as well as images of the original scenes.

4. Discussion

4.1. Future Work

There is a large amount of work that can be done to improve upon what we have accomplished in this paper. To start this pipeline has been designed in a modular manner so that future implementations can switch out components for more optimized versions. A primary example of this would be to remove Mask-RCNN, which performs object detection, with a model that performs the similar task of foreground detection. As discussed above Stable Diffusion composes a majority of our compute time, taking up to 40 seconds per frame. This is dramatically slower than all other stages in the pipeline and could be swapped out in favor of a faster algorithm. However, there is also room for fine-tuning of models like Mask-RCNN and Stable Diffusion for use in the animated domain. This would likely see much better results as animators often break the continuity of a characters body in order to accurately display motion.

As far as further replacement of pipeline steps we believe that there is a need for replacement of our implementation of background stitching as it relies of SIFT key point matching which fails on repetitive backgrounds like brickwork. This specifically led to huge issues with scenes where the camera panned across a semi-repetitive background. Similarly there is work to be done to limit the noisy affects of affine transformations on the output of the pipeline. This could even include the addition of a de-noising step at the end of the pipeline to reduce the affects of the many affine transformations. Some similar tasks to this would include

Additional extra steps in this pipeline could include tasks such as resolution scaling. We believe resolution scaling could provide two valuable capabilities. The first capability is to update resolutions to match modern displays in the same way we are updating the aspect ratio. Many displays work best with HD resolutions which many old animations do not have. Secondly, we believe down sampling could provide simpler tasks to models like Stable Diffusion, requiring the generation of fewer pixels which can then be scaled up to match a desired resolution. This would do a lot to help the computation bottleneck that we experience when running stable diffusion.

Another way to relax this computation constraint would be implementation of task parallelization. As each scene is treated independently it is not required that the computation of any two scenes be done sequentially. By running certain stages of the pipeline on scenes in parallel one could drastically cut down on the runtime of the entire pipeline. This however does make an assumption that certain scenes do not share a background, an assumption that we believe could be explored to determine which scenes would benefit from combined computation, allowing for temporal coherence across scenes and not just within them.

Lastly we believe that there is future work to be done on items that lie between the categories of background and object such as fire, lightning, and rain. These are items that should move with each frame, violating our assumption of a static background. Another assumption that is violated is the assumption of affine camera transformations. While neither of these assumptions are violated often, it is a valuable area of study to make this pipeline more robust.

4.2. Conclusion

In this paper we explore the ability to combine multiple independent computer vision tasks to attempt to solve the problem of expanding aspect ratios of old animated content such that the new content would be indistinguishable

from the source material to a brand new viewer. These existing capabilities include Stable Diffusion, Content Aware Scene Detection, Object Detection, and Key Point Matching. While we did successfully chain these tasks together in a way that generated reasonable output, we were not successful in this task. However, we still feel that the pipeline we have constructed pipeline serves as a great foundation for future work. Allowing for the introduction of new stages in the process or the replacement of old stages.

References

[1] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. *ACM Transactions on Graphics (TOG)*, 26(3):10–es, 2007. 2

[2] Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. In *International Journal of Computer Vision*, volume 74, pages 59–73. Springer, 2007. 2

[3] Loïc Dehan, Wiebe Van Ranst, Patrick Vandewalle, and Toon Goedemé. Complete and temporally consistent video outpainting. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 686–694, 2022. 2

[4] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001. 2

[5] Yandong Guo, Zhongliang Deng, Xiaodong Gu, Zhibo Chen, Quqing Chen, and Charles Wang. Aspect ratio conversion based on saliency model. In *2008 Congress on Image and Signal Processing*, volume 4, pages 92–96. IEEE, 2008. 1, 2

[6] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 2

[7] Jun-Gyu Jin, Jaehyun Bae, Han-Gyul Baek, and Sang-Hyo Park. Object-ratio-preserving video retargeting framework based on segmentation and inpainting. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, pages 497–503, 2023. 2

[8] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE transactions on computational imaging*, 2(2):109–122, 2016. 1

[9] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 277–286. ACM, 2003. 2

[10] Ce Liu and Deqing Sun. A bayesian approach to adaptive video super resolution. In *CVPR 2011*, pages 209–216. IEEE, 2011. 1

[11] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 1, 2, 5

[12] Michael Rubinstein, Ariel Shamir, and Shai Avidan. Improved seam carving for video retargeting. In *ACM Transactions on Graphics (TOG)*, volume 27, pages 1–9. ACM, 2008. 2

[13] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 1

[14] Than Htut Soe and Marija Slavkovik. A content-aware tool for converting videos to narrower aspect ratios. In *ACM International Conference on Interactive Media Experiences*, pages 109–120, 2022. 1, 2

[15] Richard Szeliski. Image alignment and stitching: A tutorial. In *Foundations and Trends® in Computer Graphics and Vision*, volume 2, pages 1–104. Now Publishers, 2006. 2

[16] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`, 2022. 5

[17] Yu-Shuen Wang, Chiew-Lan Tai, Olga Sorkine, and Tong-Yee Lee. Optimized scale-and-stretch for image resizing. In *ACM Transactions on Graphics (TOG)*, volume 27, pages 1–8. ACM, 2008. 2

[18] Liron Yatziv and Guillermo Sapiro. Fast image and video colorization using chrominance blending. *IEEE transactions on image processing*, 15(5):1120–1129, 2006. 1

[19] Bo Zhang, Mingming He, Jing Liao, Pedro V Sander, Lu Yuan, Amine Bermak, and Dong Chen. Deep exemplar-based video colorization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8052–8061, 2019. 1