About the Fascinating World of Seam Carving: Implementing and Optimizing for Video

Alexi Canesse and Clément Yvernes
Computer Science Master's degree, École Normale Supérieure de Lyon

Project designed for the Computational Geometry and Digital Images course

Computer Science
École Normale Supérieure de Lyon
France
May 1, 2023

Contents

1 Introduction

Seam carving is an algorithm which aims to resize an image while preserving its important features. Unlike traditional approaches, seam carving seeks to remove the parts of the image that are of least interest. This algorithm was originally developed by Shai Avidan and Ariel Shamir in 2007 [AS07]. Since a video is a sequence of images, it is possible to extend the algorithm to video resizing. Nonetheless, going from image to videos is not straightforward and induce many new problems such as temporal consistency and performance.

Therefore, the main motivation for this project was to implement an optimized version of seam carving for videos. First, an optimized version of the original algorithm has been developed. The rest of the work focused on the application of optimizations to extend the algorithm to videos. A special effort has been made to allow the code to run on multiple threads.

This document provides a brief overview of the basics of seam carving and its applications, including the differences between seam carving for images and videos. This section also examines existing methods for optimizing seam carving for videos. Then, this document dives into the technical details of our implementation of seam carving for video. It also presents a comprehensive analysis of our implementation including its results and discusses the limitations and trade-offs we had to make.

2 Background

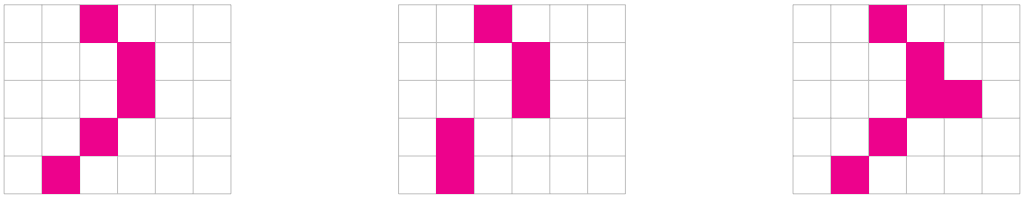
Seam carving is a content-aware image resizing algorithm. Scaling an image can result in distorted or disproportionate content. To address this issue, seam carving identifies and removes seams of pixels that have the least energy in the image, while preserving the important content. This technique can be used to reduce the size of an image while retaining its most significant features, or to add extra pixels to an image to increase its size.

Original image | Scaled image | Seam carved image

Figure 1: Comparison of resizing methods.

2.1 The Basics of Seam Carving and Its Applications

A seam is a vertical or horizontal connected path of pixels. For instance, a vertical seam is a path of pixels connected from top to bottom in an image with one pixel in each row. Similarly, on a horizontal seam, there is only one pixel in each column. In order to reach the desired format, the algorithm will successively remove horizontal and vertical seams. Removing a vertical (resp. horizontal) seam reduce the width (resp. height) by one.



A Seam | Not a seam | Not a seam

Figure 2: Comparison of a seam and non-seams.

To find which is the most appropriate seam to remove, the algorithm computes an energy for each pixel of the picture. This energy can be a gradient magnitude, entropy, visual saliency or any quantity that quantifies the interest of a pixel in the image. Then, the algorithm finds the seam with less energy. Dynamic programming can be use to compute this quantity efficiently. This seam is then removed from the image.

Removing a seam could create a high energy zone. Forward energy [RSA08] is here to address this issue. We wont present it in details here. This method is not flawless nonetheless it improves the results in many case and seams to have an even greater impact on video (confirmed by our tests).

Figure 3: Example of an energy map.

Seam carving has a variety of applications, including image resizing, objects removal, image retargeting, and video retargeting. When used for image resizing, seam carving preserves the essential content of the image, making it suitable for different screen sizes without distorting the important features. In object removal, unwanted objects can be removed from an image by identifying and removing the corresponding seams. Seam carving can also be used for retargeting both images and videos to fit different viewing platforms while retaining the important content.

2.2 Seam Carving for Images vs. Videos

Applying seam carving to video introduces several new challenges due to the temporal dimension of video. One of the main challenges is maintaining the consistency of the content across frames. When a seam is removed from one frame, the corresponding pixels in subsequent frames can represent very important information. For instance, removing a seam can change the apparent distance between objects in the background. If seams are removed between these objects only during a part of the video, then the apparent distance between these objects may change during

the video. In the worst case, this can make it appear as if background elements teleport from one frame to the next.

Another challenge is the computational complexity of processing video. Seam carving involves analyzing each frame of the video, and the computational cost can be significant, especially for high-resolution videos. For example, if the seams are recalculated on each frame then the computation time to resize one second of a video can be multiplied by one or two orders of magnitude compared to the time taken to resize a single frame.

Finally, there is also the challenge of handling dynamic content in the video. For example, if an object is moving across the scene, it may not be appropriate to remove or add seams that intersect with the object. Thus, the seam carving algorithm needs to be able to detect and adapt to changes in the content of the video.

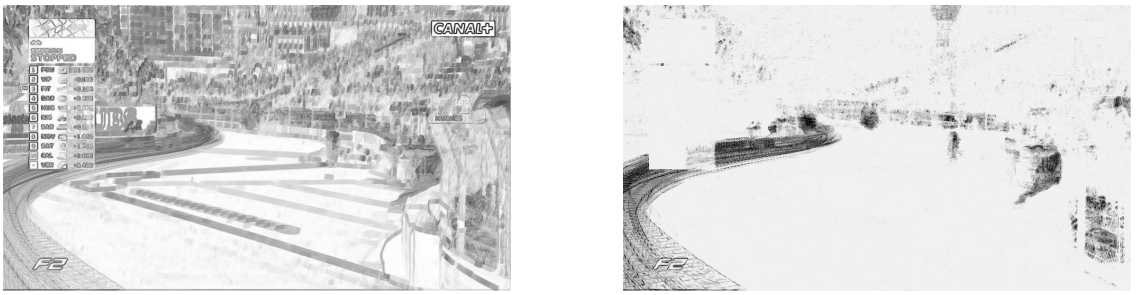
2.3 Seam Carving for Videos

Various methods exist for adapting seam carving techniques to videos, each with their own strengths and weaknesses. We decided to use a "static" algorithm. A unique energy map is created for the entire video and running the classic seam carving algorithm on each frame, removing the same seams throughout the video.

One advantage of this approach is its speed. By avoiding the need for extensive computations, it is one of the fastest methods available. Additionally, it is relatively easy to implement. However, this method also produces superior results in comparison to other approaches in several ways. Specifically, when the camera is stationary, fixed objects remain stationary, whereas dynamic methods can produce a jittery effect. Moreover, while this algorithm may generate more visual artifacts, it does not produce any temporal artifacts.

This method is not without drawbacks. As soon as the camera moves or the scene changes, the result does not make sense. If a video is a composed of fixed plans linked by moving scenes, using a combination of "static" seam carving and other dynamic approaches could address this issue. Another issue (that we will address later) is that this method does not work inline.

In order to create an energy map for the video, we need to know which pixels get important "sometime" and where there is a lot of movement. To capture both those information, we compute an energy map for each frame and combine them using a combination of the maximum and the temporal derivative.



Maximum Temporal derivative

Figure 4: Comparison of video energies for `tests/f2.m4v`.

The maximum helps to catch fixed important items such as watermarks or walls. The temporal derivatives helps to catch moving objects and to effectively choose how much weight we want to give them. This slows down the computations and adds an hyper-parameter that needs to be tuned but it greatly improve the result.

3 Implementation

The algorithm is implemented in C++ which offer a good balance between performance and ease of use. We are using the library openCV. It offers efficient image processing and is easy to use. It is also widely used which could help a lot when troubleshooting issues.

3.1 Using the code

The project has been intensively tested on MacOS Ventura on an Intel chip and has been tested on a Ubuntu x64 computer. We do not know how well it works on other systems. Notice that the project has not been tested on any Windows computer.

First, get a copy of the code. One copy has been provided with the submission of our project but you can also clone our repository.

```
git clone https://github.com/alexicanesse/FastVideoSeamCarving
```

The project may work with C++11 but has only been tested with `C++17` so we do not offer any guarantee if you decide to use an older standard. Same goes for `CMake 3.20`. The libraries `OpenCV` and `Boost` are used and must been installed in order to compile. To install those libraries run the following command in a terminal

```
# For MacOS users
brew install opencv boost

# For Ubuntu users
sudo apt install libopencv-dev libboost-all-dev
```

Once you're all set, you can finally compile. Run the following commands at the project's root.

```
# Create a build directory
mkdir build
cmake -B ./build -S .
cmake --build ./build
```

You can finally run the program. Everything is specified using command line options. Here are the options

```
-h [ --help ]             print the help message
-i [ --input ] arg        set the input file
-o [ --output ] arg       set the output file
--h_scale arg             set the horizontal scaling factor
--v_scale arg             set the vertical   scaling factor
-r [ --show_result ]      show the output file
-e [ --show_energy_map ]  show the energy map
```

3.2 Speed Optimization Techniques

Seam carving is a computationally expensive algorithm and video increase this cost by a some orders of magnitude. Hence we had to careful about optimisation.

3.2.1 Optimization through lower resolution processing

The algorithm only need a gray-scale version of the video and works on it. Hence, modifying only the gray-scale video during the computation and removing all seams only at the end is enough. When we load the input, we make a gray-scale version of it and cut its resolution by two. Instead of remove one pixel wide seams, we remove two pixel wide seams. By reducing the resolution, we were able to significantly increase the speed of our processing while still maintaining acceptable levels of accuracy. It is important to notice that the original video is not affected by this downscale.

3.2.2 Optimization through multi-threading

Many tasks are made on each frame independently. Those tasks could be done concurrently without any issue. Hence we applied multi-threading everywhere it make sense. We have to be careful with parallel computing : it is not easy to do and can lead to unexpected bugs that are among the hardest to troubleshoot. It is also important to remember that multi-threading is not always faster. Hence, we tested every parallel applications to ensure that they all make computing faster. It improved our performance linearly.

4 Results

All the results present in this report are also available on our git and have been submitted with it. The original content and the parameters are also sharred. Hence, all the results presented are easily reproducible.

4.1 Visual Examples of Seam Carving on Images

Our first tests are done on images. The first image [5] is a picture in a mountainous environment. The sky corresponds to a large part of the image. We expect the algorithm to cut a large part of the sky because there is little energy in the sky: there are not many information in the sky.

Original image | 0.9 horizontal and 0.8 vertical | 0.6 horizontal and 0.5 vertical

Figure 5: An exemple with `tests/montagne.jpg`. [Clément Yvernes]

The cloud and the shape of the plateau are well preserved. When the resizing is very important the overall composition of the photo is lost. But the overall shape is preserved. The algorithm works well on simple images.

Original image | 0.9 horizontal and 0.8 vertical | 0.9 horizontal and 0.6 vertical

Figure 6: An exemple with `tests/alexi.jpg`. [Alexi Canesse]

The second test is performed on a picture [6] less suitable for seam carving. There is no low energy or low interest area of the image. Nevertheless, the results are quite satisfactory and the overall composition of the image is well preserved. This picture does not look altered without context even though it has been seam carved with a small factor.

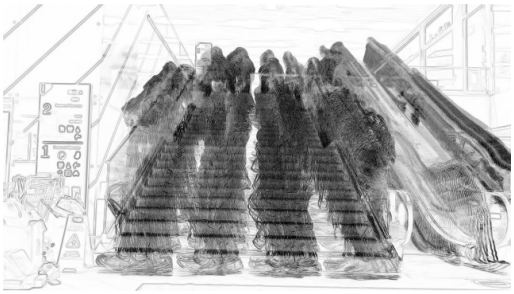
Overall, the algorithm works extremely well on images.

4.2 Visual Examples of Seam Carving on Videos

Our results on fixed images are excellent even with complicated images. Nonetheless, video is the quintessential challenge and we were not able to shine as bright here. But are results are still interesting

The first video test is a dance video [7] with a fixed camera. The interesting part of the video that we want to keep is the choreography.

Result with 0.8 horizontal and 0.9 vertical



Energy map

Figure 7: An exemple with `tests/dance_720p.m4v` [tuz21]

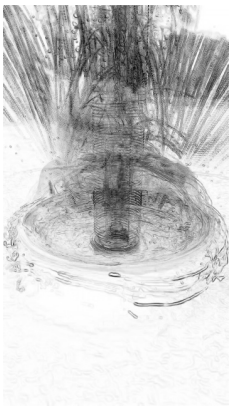
The algorithm preserves the choreography well. This example shows very well the interest of taking into account the time derivative to calculate the energy map. This derivative allows to detect moving objects and thus avoid that a seam comes to cut these objects. If we pay attention to the left part of the image, we see deformations that are due to the seams. It is likely that a person not paying attention to the details would not notice the resizing of the video.

The second video tested is a slow motion splash [8]. This video is interesting because interesting objects pass through all the points of the image.

Original

0.8 vertical

0.8 horizontal and 0.6 vertical



Energy map

Figure 8: An exemple with `tests/slowmo.mp4` [Tec22]

The results are satisfactory, the overall evolution of the splash is convincing. It is still possible

to find some artifacts by taking the time to study the precise trajectory of the drops.

Our implementation is rather efficient on videos taken with fixed cameras. When the resizing ratios are reasonable the results are satisfactory.

4.3 Performance Analysis of the Implementation

We did not have enough time to make a real performance analysis of our implementation and do not want to present a naive approach. Hence, instead of concrete results, we are going to present the methodology we designed to evaluate our performances which is what really matters.

To give some results, all the content provided has been computed using a 2019 Intel (4GHz) Macbook Pro with 16 GBs of RAM and each computation took less than two hundred seconds (usually a lot less). We reach real-time performance on 480p videos that are at least 10 seconds long. And the scaling factor are not too small.

There are two important factors to measure : the impact of the resolution and of the number of frames.

In order to measure the importance of the impact of the resolution, we must fix the number of frames and modify the resolution. We must be careful with this : the way we made our implementation is such that the position of the seams will affect the running time. This is not negligible. Hence, we need to make the measures for many images that are as representative as possible. This measure should be done for each frequent resolution at some number of frames (including one). To measure the impact of the number of frames the methodology is similar.

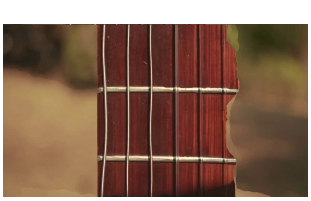
All of this can be automated with some scripting that generates a `*.dot` file that we could directly open using Ti*k*Z. The hardest part is getting the content. Having enough is not an easy task. Nonetheless, the challenge is making sure that our sample is representative.

We made some optimisations such as subs-ampling and spent some time making sure our code is optimized. We regret not being able to give a proper evaluation of our performance optimisations.

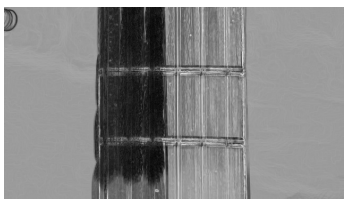
4.4 Limitations and Trade-offs of the Implementation

We have seen that our implementation is efficient for videos taken with a fixed camera. However, it is no longer efficient for videos that are not in fixed camera. Indeed, the time derivative of the background becomes important, which disturb the energy map.

This effect can be seen very well on the following video. This is a video that shows the vibrations of guitar strings [10]. If you pay attention to the video, you will notice that a shock wave and the movement of the camera disturb the background. The background is also blurred. For a human eye, these perturbations are very negligible but they make our implementation inefficient.



0.9 vertical and 0.9 horizontal



Energy map

Temporal derivative

Figure 9: An exemple with `tests/guitar.mp4`. [Ruy18]

The use of the time derivative is the source of this effect. The following example [10] shows that by not using the time derivative, the result becomes satisfactory again.

0.5 × maximum + 0.5 × temporal

Maximum only

Figure 10: An exemple with `tests/guitar.mp4`. [Ruy18]

5 Conclusion

The project successfully implemented a static seam carving algorithm for videos and evaluated its performance. The algorithm's performance was found to be excellent on images, but it faced challenges on videos, particularly when the camera was moving relative to the background. The hyperparameter tuning could improve the algorithm's performance further. Nonetheless, the algorithm's performance was satisfactory, and it can be used as a starting point for developing a more robust seam carving algorithm for videos.

5.1 Summary of the Project and Findings

The project aimed to implement a static seam carving algorithm for videos and evaluate its performance. The algorithm used maximum and temporal derivative to create a single energy map. Although the algorithm works marvellously on images, its performance varies on videos. It works great on some videos when the camera is fixed and the video is easy enough, but it fails hard if the camera is moving relative to the background. The hyperparameter needs to be tuned to improve the performance, which was not explored in the project. Overall, the performance of the algorithm was good, but further improvements can be made.

5.2 Suggestions for Further Research and Improvement

5.2.1 Live seam carving

In order to implement a real-time static seam carving solution for video, we propose the use of an optimization technique under constraints. The solution involves keeping a buffer of a few seconds of video segments, working on each segment independently, and leaving a short gap (approximately 5 to 10 percent of the segment size) between each segment. Using optimization under constraints, we then create manifolds that smoothly connect the adjacent segments. By applying this technique, we may achieve a continuous and high-quality video output, while reducing the size of the video in real-time without significant loss of content.

5.2.2 Using computer vision to get better results

It is surely possible to improve the results of seam carving by using recent advances in computer vision. Recent advances in tracking [Mar+22] [Jav+23], and foreground object detection [Zha+22] provide effective means to detect areas of interest during the video. These methods could allow to detect the areas where it is preferable not to remove seams. For instance, it would be possible to look for an energy that allows to preserve the coherence of the background without having to worry about the foreground elements. Moreover, as ne of the main costs of seam carving is the calculation of the pixel energy and finding the right seam. These methods may also accelerate the algorithm by detecting the areas where it is interesting to remove a seam. The calculations would then be done only on these areas which would reduce the calculation time.

However, deep learning models are often slow. Using them directly on the video could cost too much computation time to hope to obtain a time saving. However, these models may work on videos of lower quality. All the proposals made before could be realized on lower resolution videos. It would be enough to extend the detected areas to the original video and to apply seam carving only on the areas that require a careful calculation.

References

[AS07] Shai Avidan and Ariel Shamir. "Seam carving for content-aware image resizing". In: *ACM SIGGRAPH 2007 papers*. SIGGRAPH '07. New York, NY, USA: Association for Computing Machinery, July 2007, 10–es. ISBN: 978-1-4503-7836-9. DOI: 10.1145/1275808.1276390. URL: https://doi.org/10.1145/1275808.1276390 (visited on 04/23/2023).

[Jav+23] Sajid Javed et al. "Visual Object Tracking With Discriminative Filters and Siamese Networks: A Survey and Outlook". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.5 (2023), pp. 6552–6574. DOI: 10.1109/TPAMI.2022.3212594.

[Mar+22] Seyed Mojtaba Marvasti-Zadeh et al. "Deep Learning for Visual Tracking: A Comprehensive Survey". In: *IEEE Transactions on Intelligent Transportation Systems* 23.5 (2022), pp. 3943–3968. DOI: 10.1109/TITS.2020.3046478.

[RSA08] Michael Rubinstein, Ariel Shamir, and Shai Avidan. "Improved seam carving for video retargeting". In: *ACM transactions on graphics (TOG)* 27.3 (2008), pp. 1–9.

[Ruy18] Ruy Mascarúa. *Guitar strings vibrating*. Mar. 2018. URL: https://www.youtube.com/watch?app=desktop&v=XOCGb5ZGEV8 (visited on 04/29/2023).

[Tec22] Technical Kingdom. *Satisfying slow motion video #shorts #satisfying #slowmotion - YouTube*. 2022. URL: https://www.youtube.com/shorts/M8R9IwPX-ng (visited on 04/29/2023).

[tuz21] tuzelity SHUFFLE. *World's biggest Running man Challenge TUZELITY SHUFFLE*. Mar. 2021. URL: https://www.youtube.com/watch?v=IYm-n-vwhnw (visited on 04/29/2023).

[Zha+22] Xinyue Zhao et al. "A survey of moving object detection methods: A practical perspective". In: *Neurocomputing* 503 (2022), pp. 28–48. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.06.104. URL: https://www.sciencedirect.com/science/article/pii/S0925231222008359.