the Ministry of Education and Science of Ukraine

United Nations Educational, Scientific and Cultural Organization

Junior Academy of Science of Ukraine

CATEGORY: Computer & Mathematics

QUICK CONTENT-AWARE IMAGE RESIZING METHOD

Author
Heorhii Ambartsumov
Scientific advisor
Tetiana Vlasenko

National Center "Junior Academy of Sciences of Ukraine" under the auspices of UNESCO

Contacts:
Phone: +380 68 777 5898
Email: gosha20052811@gmail.com

Kyiv, Ukraine, 2022

ABSTRACT

In modern world of social media, one may often need to quickly edit a specific image. And image resizing is undoubtedly the most popular image editing operation. It can be performed in a number of ways, the most usual of which are cropping, stretching, and pinching. These operations are very fast and simple; however, they lead to loss either of content or proportions of the image.

In 2007, a content-aware resizing algorithm called seam carving method was presented by Shai Avidan and Ariel Shamir. This method allows changing image size by modifying only those parts of picture which are the least important for image perception. Although this algorithm is effective, it is quite complex and requires a decent computing power for quick image processing.

The goal of this research was to create more optimized method of content-aware image resizing, which will make possible its use on mobile devices and therefore enable much more users to resize images more effectively.

Key objectives of research included:

1. Analysis of original seam carving method
2. Determining the most resource-consumptive components of original algorithm
3. Development of alternative components
4. Research of ways of image preprocessing that may increase quality of image processing
5. Implementation of original and improved methods in Python programming language
6. Testing and comparison of both methods in terms of speed and resource consumption

In the result of the research, we created a content-aware image resizing method, which is much faster than the original one and comparable in terms of processing quality. Therefore, this algorithm is perfect for low-performance devices, such as smartphones or tablets and indeed enables more users to use content-aware image resizing.

LIST OF ILLUSTRATIONS

GLOSSARY

Pixel A color dot, of which an image is composed. Pixel color is encoded using RGB model.

Image A rectangular matrix, which consists of pixels. By default, it is assumed that image has n × m resolution, where n is its height and m is its width; origin is located in the point (1,1) and corresponds to upper left pixel.

Seam A vertical or horizontal stripe of pixels. A seam consists only of vertical and diagonal or horizontal and diagonal components.

Optimal seam or minimal seam A seam which has minimum total energy of pixels in it.

Energy of a pixel Importance of pixel for perception of image.

Energy map of an image A matrix, each cell of which denotes energy of the corresponding pixel of the image.

Representativeness of a pixel Ability of a pixel to convey color of some group of nearby pixels.

Nodal pixel A pixel, which lies on intersection of lines of rectangular grid placed on image.

Nodal image An image, which consists of nodal pixels of the original image and approximately conveys its contents.

Nodal seam A low-detailed seam on image, pixels of which correspond to those of optimal seam of nodal image.

TABLE OF CONTENTS

INTRODUCTION

Our research work "Quick Content-Aware Image Resizing Method" is dedicated to the development and implementation of a high-speed method of image retargeting that would meet the tight complexity requirements of mobile devices. The previous studies in this area, such as (Rubinstein et al., 2008) focused rather on improving quality of the resulting images, we, however, emphasized runtime and complexity.

In course of work, we:

- Analyzed content-aware image resizing method, known as a seam carving algorithm
- Developed a set of solutions to the problem of quick lossless image resizing, described an algorithm used for this
- Implemented both original and improved methods in Python
- Tested both programs on a set of various images
- Analyzed aspects of the performance of classic and optimized methods, compared runtime and quality of results
- Offered practical application for the new image resizing method

The concepts of nodal elements introduced in the research may be reapplied and used beyond this work for further improvement of speed of lossless image resizing. The results of our research will be accessible and open to anyone.

Chapter 1

MAIN ASPECTS OF THE ORIGINAL SEAM CARVING METHOD

The seam carving method, which further might be called original method, works by selecting and modifying a number of vertical and horizontal stripes of pixels, seams, which have the least value for overall image perception. The process of image modification consists of several steps.

First, the algorithm determines the importance of every pixel of the image in way of assigning a numeric value to each of them, the so-called pixel energy. The energies of pixels are stored in a matrix – so-called energy map of the image.

After that, the algorithm calculates and selects the seam, which should be removed or doubled. Algorithm always selects the seam, which has the least total energy of the pixels in it. Then the algorithm removes or doubles the seam, depending on the mode of operation.

The detailed analysis of the seam carving algorithm is presented further.

1.1 The hypothesis of the original algorithm

The principle of the original algorithm is based on assumption that every pixel of the image has a specific value for image perception, the so-called *pixel energy*. The energy of the key parts of the image should be higher, while the energy of non-important parts, such as background, should be lower. Thus, the operations on image should minimize the change in the overall energies of pixels of the image. From here the notation $E_{i,j}$ will be used to indicate energy of a pixel with coordinates (i, j). The first and second coordinates of the pixel indicate its vertical and horizontal position respectively, with the origin $(1, 1)$ located in the leftmost pixel of the first row, if not stated otherwise.

1.2 Objects and concepts, which the original algorithm uses

The original seam carving algorithm operates in the way of deleting of inserting horizontal or vertical stripes of pixels from or into the image. A stripe, or a *seam*, which is being deleted or inserted, should have the minimum possible sum of the energies of pixels which it contains. The algorithm assigns a specific *energy* to each

pixel of the image and creates an *energy map* of the image in order to calculate the *minimal* or the *optimal* seam.

1.3 Calculating the energy. Backward energy method.

Backward energy method of energy calculation was first presented in the original seam carving paper (Avidan & Shamir, 2007). This method is based on an assumption that the most valuable parts of the image should be the least monotonous.

Therefore, the energy of the pixel is directly based on the color difference between the pixel and its neighbors.

It implies that for determining the pixel energy it would be sufficient to calculate the sum of absolute values of the differences between the pixel and its neighbors along the three color channels, implying here and after that the Red-Green-Blue (RGB) color model is used.

Let's denote color of pixel at (i, j) as $I_{i,j}$. Then the simplified formula of the pixel energy, which is denoted here and after as $E_{i,j}$ takes such look:

$$E_{i,j} = \left|I_{i,j} - I_{i-1,j}\right| + \left|I_{i,j} - I_{i,j-1}\right|$$

As an example, Figures 1.1 and 1.2 are provided, which depict an image and its energy map respectively. Pixels with higher energy are marked with a lighter color.

Figure 1.1: Example image

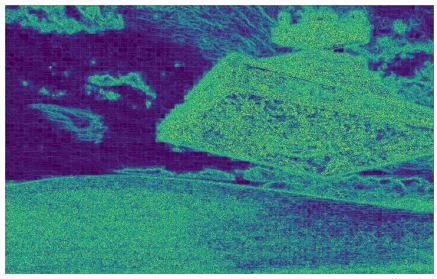


Figure 1.2: Energy map of an image

Source: https://www.wallpaperflare.com/star-warsf-windows-xp-1680x1050-technology-windows-hd-art-wallpaper-sxsct

1.4 Calculating the minimal seam

Below a way of calculating the minimal seam is provided, which was proposed by the authors of the original paper. The process of finding the minimal *vertical* seam is shown as an example.

Let's notice a fact: the minimal seam which passes through coordinates (i, j) also passes through coordinates $(i-1, j-1)$, $(i-1, j)$, or $(i-1, j+1)$. Therefore, we can define the formula for sum of energies of pixels of the seam which ends in the point (i, j) recursively:

$$
\begin{aligned}
S_{i,j} &= E_{i,j} + \min(S_{i-1,j-1}, S_{i-1,j}, S_{i-1,j+1}) \text{ for } i > 1,\ 1 < j < m \\
S_{i,j} &= E_{i,j} + \min(S_{i-1,j}, S_{i-1,j+1}) \text{ for } i > 1, j = 1 \\
S_{i,j} &= E_{i,j} + \min(S_{i-1,j-1}, S_{i-1,j}) \text{ for } i > 1, j = m \\
S_{i,j} &= E_{i,j} \text{ for } i = 1
\end{aligned}
$$

The notation $S_{i,j}$ will be used further in the text.

Thus, one can assert that a vertical seam with minimal sum of energies ends in point (n, k), where n is the height of the image, and $k \in [1, m]$, where m is the width of the image. The point of the last row with minimal S will be the endpoint of the minimal seam.

Having determined the endpoint, the algorithm needs to reconstruct the seam. Remember that when it calculates the cumulative energy, it always selects previous point with minimal S. Therefore, it should just keep picking the minimum among the three options for previous pixel, and in such way recreate the seam.

Dynamic programming may be used for implementation of this method. An example of an optimal seam is shown on Figure 1.3.

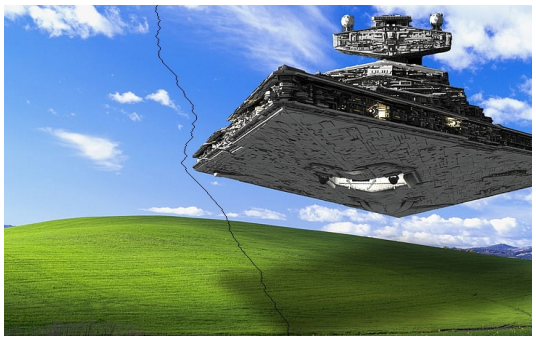


Figure 1.3: Optimal seam on the image

1.5 Alternative option for calculating energies and the seam. Forward energy method.

This method was proposed in 2008 in the paper on extension of the seam carving algorithm (Rubinstein et al., 2008). It relies on an idea that the optimal seam should not have the minimal energy, but *add* to the image as little energy as possible after

its deletion. Added energy is the energy which is created in the result of pixels becoming neighbors after deletion of some seam.

Here is shown the principle of work of this method on an example of finding a vertical seam. For each pixel of the image there are three options of seam's trajectory, which are shown on Figure 1.4. In the process of calculation, the option with the least added energy should be selected.

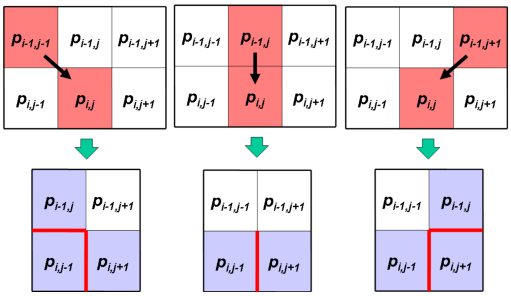


Figure 1.4: Three options of seam's trajectory

Source: (Avidan & Shamir, 2007)

Let's again denote the pixel's color as I. Let's also denote the cost of each of three options as $C_L(i,j)$ (Fig. 1.3a), $C_U(i,j)$ (Fig. 1.3b), $C_R(i,j)$ (Fig. 1.3c) respectively. Then:

$$
\begin{aligned}
C_L(i,j) &= \left|I_{i,j+1} - I_{i,j-1}\right| + \left|I_{i-1,j} - I_{i,j-1}\right| \\
C_U(i,j) &= \left|I_{i,j+1} - I_{i,j-1}\right| \\
C_R(i,j) &= \left|I_{i,j+1} - I_{i,j-1}\right| + \left|I_{i-1,j} - I_{i,j+1}\right|
\end{aligned}
$$

These values are used to fill a cumulative cost matrix, M, in order to calculate the seams using dynamic programming. For vertical seams, the matrix will be filled in such way:

$$
M_{i,j} = \min \begin{cases} M_{i-1,j-1} + C_L(i,j) \\ M_{i-1,j} + C_U(i,j) \\ M_{i-1,j+1} + C_R(i,j) \end{cases}
$$

Just like with backward energy method, the seam can be restored while going up and greedily selecting the least predecessor.

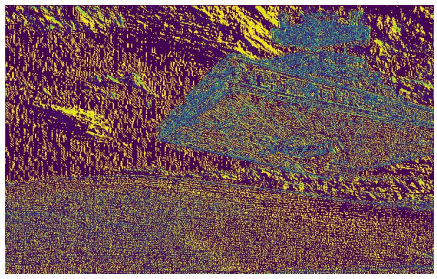


Figure 1.5: Energy map

Figure 1.6: Optimal seam

Figures 1.5 and 1.6 show a forward-energy map and the optimal seam of an image respectively.

1.6 Cutting the image

To make the image's width or height k pixels smaller, the algorithm should just delete an optimal vertical or horizontal seam respectively k times. Figure 1.7 shows image from Figure 1.1 with width decreased by 20% using (a) backward (b) forward energy method.

a) b)

Figure 1.7: Image cut using two energy methods

1.7 Increasing image size

Stretching the image is done by finding an optimal seam in the image and doubling it several times. The process of search remains usual, as described above. To avoid double seam selection and seam intersection, after each iteration the seam which has been selected from the image is deleted and added an array. After repeating this operation required number of times, all the stored seams are doubled on the original image.

1.8 Time complexity of the seam carving algorithm

The algorithm is quite computationally complex, because on each iteration it:

1. Calculates the energy map in $O(n \times m)$
2. Finds the optimal seam in $O(n \times m)$
3. Deletes it in $O(n \times m)$

This process is repeated k times, as usually modification of multiple seams is required. In such way the complexity results in $O(k \times n \times m)$. As usually $300 \leq$

$k, n, m \leq 4000$, the time of execution will be quite big. Because of this the seam carving algorithm is not usually used on portable devices. Speedup of the energy calculation and seam finding processes, however, would solve this problem.

Chapter 2

QUICK ALGORITHM FOR CONTENT-AWARE IMAGE RESIZING

While the original seam carving algorithm calculates energies of all the pixels of the image, the optimized variant utilizes simplification of the original image and solving the task of finding the seam on a matrix of a smaller size. After calculating the seam, the algorithm translates the pixels of the simplified seam onto the original image, which result into an approximate seam. The last step consists in detailing the approximate seam and modifying it just like in usual seam carving. The detailed description of the optimized method is provided further.

2.1 Factors, which slow down the original algorithm

The most resource-consumptive stages of processing an image are calculation of the energy map and finding the optimal seam. Therefore, creation of a quicker way of finding the minimal seam would allow a significant speed-up.

2.2 Representativeness of a pixel. Nodal pixel. Nodal image.

Usually, on each image that is perceptible by a human, the pixels may be nominally assigned to specific color groups. Let's define *representativeness* of a pixel as ability of a pixel to convey a color of some group of nearby pixels. The representativeness of a pixel depends on area of field, which it represents, and on coloring of that area. Then, to reach more achieve more monotonous representativeness of pixels of the image, it is beneficial to take groups of equal size.

Representing the image as a grid with rectangular cells is one of the ways to break an image into such groups. Let's call the pixels which lay on the nodes of a grid *nodal*. Let's assume that a nodal pixel represents a cell, in upper left corner of which it lies. If all the nodal pixels are assembled into a matrix, the result would be an image with lower resolution, which will approximately convey the contents of the original image. Let's call such a low-resolution image a *nodal image*.

On Figure 2.1, an image with grid (a) and its nodal image (b) are depicted. If the grid has cell size $v \times h$, the pixel of the nodal image with the coordinates (i, j) will have coordinates $(i \times v, j \times h)$ on the original image. In this case, it is assumed that

the coordinates of origin point are $(0, 0)$ for simplification of the formulas.

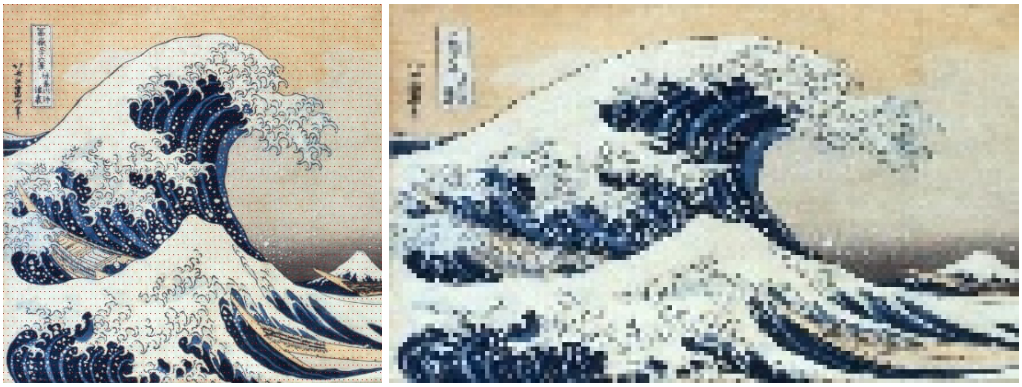


Figure 2.1: An image with 8×5 grid (left), nodal image generated with given image and grid (right).

Source: https://artincontext.org/wp-content/uploads/2021/11/The-Great-Wave-off-Kanagawa-Katsushika-Hokusai.jpg

2.3 Nodal seam. Calculating the nodal seam.

Let's assume that each of the pixels of nodal image *perfectly* represents the cell to which it is assigned. Let's calculate the optimal seam on a nodal image, generated by image I and a grid with cell size $v \times h$. Then pixel with coordinates (i, j) of the nodal image which lies on the calculated seam has the following properties:

- It corresponds to pixel of the original image with coordinates $(i \times v, j \times h)$
- It lies on the optimal seam of the *original* image

Let's define a *nodal seam* as a seam that consists of nodal pixels of the original image, which correspond to pixels of the optimal seam of the nodal image. Then every pixel of the nodal seam will lie on the optimal seam of the image.

Figure 2.2 shows a nodal seam of an image.

2.4 Improving the representativeness of the pixels

The method of finding the nodal seam will be precise only in case of ideal representativeness of nodal pixels, which lie on the seam. In reality, this property is very unlikely to be completely true. However, it may be made closer to perfect, using image preprocessing.

One of the ways to increase representativeness of nodal pixels of the image is to decrease the number of colors of the image. This operation will make the cell more

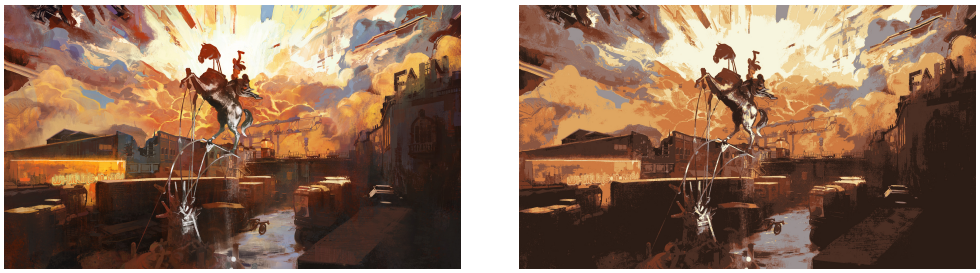
Figure 2.2: The nodal seam on the right is marked with red dots

Source: https://media-cdn.tripadvisor.com/media/photo-s/18/45/47/1b/amazing-views-of-the.jpg

monotonous, and, in such way, will increase the representativeness of some nodal pixels.

In this paper, the k-means clustering algorithm was used to narrow the color palette of the image. It was chosen because of manual selection of number of colors, to which the palette will be brought.

The k-means method works in way of breaking the set of pixels of the image into a number of groups, called clusters, and optimizing the partition depending on color and distance among pixels. After that each pixel of the cluster is assigned a specific color, which represents the general coloring of pixels of the cluster. In such way the required decrease of colors of the image is achieved, and the representativeness of nodal pixels is increased.



a) b)

Figure 2.3: Decreasing color palette of image (a) to 10 colors (b)

Source: https://discoelysium.com/media?tab=Wallpapers

2.5 Detailing the nodal seam

Once the algorithm has found the nodal seam, it needs to *detail* it – add more intermediate points between the nodal pixels. In order to do that, the optimal seam between each pair of consecutive pixels of the nodal seam should be calculated. This process is shown on example of finding the optimal vertical seam on the image.

As finding the optimal seam between two points may be quite a computationally complex task, the search can be restricted to a specific rectangular part of a plane, without losing possible options for the seam. If the first point has the coordinates (i_1, j_1), and the second point – $(i_2, j_2), i_2 > i_1$, and the height of the cell is h, then the coordinates of the fragment will be such:

$n_1 = i_1$ – the first coordinate of the upper-left point

$n_2 = i_2$ – the first coordinate of the lower-right point

$m_1 = j_1 - \frac{h}{2} - \frac{j_1 - j_2}{2}$ – the first coordinate of the lower-right point

$m_2 = j_1 + \frac{h}{2} - \frac{j_1 - j_2}{2}$ – the second coordinate of the lower-right point

One can easily prove that the seam will never leave these coordinates. Of course, some cells of the region will be never accessible by a valid seam, but it is faster to work with fragment of rectangular shape due to specifics of memory allocation.

Now, the task is to find the optimal seam between endpoints, which lies inside the calculated region. The first step would be to create energy map of this area. It is done in the usual way, using backward of forward energy method. Then a matrix of cumulative energies is created, similar to that used in the original algorithm. It is calculated in the following way:

$$
\begin{aligned}
&S_{i_1,j_1} = 0,\\
&S_{i_1,k} = +\infty \text{ for } k \neq j_1,\\
&S_{i,j} = \min \begin{cases} S_{i-1,j-1} + E_{i,j} \\ S_{i-1,j} + E_{i,j} \\ S_{i-1,j+1} + E_{i,j} \end{cases} \qquad \text{for } i \neq i_1
\end{aligned}
$$

Having calculated the matrix, it is easy to recreate the seam using greedy ascent, similar to that used in seam carving method.

To get the full optimal seam, the optimal seams between each pair of the points of the nodal seam should be found first, and then concatenated. This way of finding the seam will be further called *the nodal seam detailing method*.

As an additional means of increasing precision of image processing, image pre-clustering was used. Figure 2.4 shows the optimal seam found using the usual seam carving algorithm (a), and the seam found using nodal seam detailing method with preprocessing (b). In both cases forward energy method was used.

a) b)

Figure 2.4: Optimal seams using usual (a) and nodal (b) methods

Source: https://www.peakpx.com/en/hd-wallpaper-desktop-omtdm

Although in the upper part of the image the seams differ, the difference is insubstantial, considering the monotonous coloring of this region.

2.6 Cutting the image using the fast seam finding method

Decrease of the image size may be achieved by finding the optimal seam with the quick method on the clustered image and deleting it from the original and clustered image simultaneously. If the size is decreased by more than one pixel, this process is repeated the needed number of times. Figure 2.5 illustrates an example of such operation, where in both cases forward energy is used.

2.7 Increasing image size using the fast seam finding method

This is done analogically to the original method. The only difference is in the principle of seam calculation and pre-clustering, which is present only in fast method. Figure 2.6 is an example of such increase, both cases use forward energy method.

2.8 Complexity of the new method

The pre-clustering is not included into the complexity analysis as it occurs only once, and, according to the results of the tests, usually takes a negligibly small fraction of

a) b) c)

Figure 2.5: Image (a) cut by 20% using usual seam carving (b), fast method (c)

Source: https://www.airbus.com/en/products-services/commercial-aircraft/passenger-aircraft

a) b) c)

Figure 2.6: Image (a) with size increased by 30% using usual method (b), fast method (c)

Source: https://www.leicestermercury.co.uk/news/local-news/its-bit-mystery-car-left-566707

execution time.

Let's consider one iteration of the algorithm. During it, the algorithm:

- Creates the nodal image in $O((\frac{n}{v}) \times (\frac{m}{h}))$, where n, m are dimensions of image, v, h are dimensions of a cell
- Calculates energies and the nodal seam using the same number of operations
- Details the nodal seam in $O(h^2 \times \frac{m}{h} = h \times m)$
- Deletes it in $O(n \times m)$ in worst case

One may see that formally the time complexity still remains $O(n \times m)$ for one modification operation, but in reality, these optimizations give a huge speedup. Also, the time complexity of the deletion may be improved to $O(n)$ using 2D linked lists, but in that case the time complexity of nodal image creation would be raised to $O(\frac{n}{v} \times \frac{m}{h} \times \log_2 m)$ (because then heavy additional data structures would have to be added). It is quite a controversial decision. In following chapter, more data about practical testing is provided.

Chapter 3

COMPARISON OF ORIGINAL AND OPTIMIZED METHODS

3.1 The test data

In order to get an accurate representation of the algorithms' capabilities in terms of speed and accuracy, a set of 99 various images was used. The resolutions of images varied from 200×200 to 1000×1000 pixels. The parameters of the algorithms were set randomly according to the following rules:

- Type of operation (cut/stretch) – completely randomly,
- Energy calculation function (forward/backward) – completely randomly
- $P \in [10, 80] \cap \mathbb{Z}$ where P is the percent by which image was cut or stretched
- $C \in [5, 25] \cap \mathbb{Z}$ where C is the number of colors to which the color palette is reduced, only for optimized method
- $V, H \in [4, 20] \cap \mathbb{Z}$, $H \leq V$ where V and H are vertical and horizontal dimensions of a cell of the grid, only for optimized method

All the operations changed only the image's width. For each image and corresponding parameter of settings, both algorithms were tested, and their results were compared.

All data provided is the output of realizations written in Python language (Ambartsumov, 2022a).

3.2 Methods of quality and speed comparison

Each of the 99 pairs of the images from test set was thoroughly reviewed and evaluated. The main quality criteria were preservation of small details and preservation of large and medium objects. Considering these factors, the verdict about supremacy of one of the methods was made, another option was tie.

Besides this, the execution time in each of the test cases was compared. Not absolute, but processor time was measured, which allowed to minimize influence of side factors on the final result.

a) b) c)

Figure 3.1: Image (a) cut by 32% using seam carving (b), optimized method (c)

Source: https://unsplash.com/photos/rf-0DQu5M6Y

Figure 3.1 shows an example of a test case. The original image (a) has dimensions 352×488 pixels. For the optimized method, the following parameters were used: 11 colors (clusters), cell size 20×20. As one may see, the quality of processing of both small and big objects is similar both in result of processing with original (b) and optimized (c) method. Nonetheless, it took 74.5 seconds in the first case and 7.8 seconds in the second case to process the image.

Proving that in case of relatively *small* cell size the optimized method processes images as quality as and much faster than the original method was the main purpose of testing. In result of full analysis of test data, the next statistics arose:

- In 60% of cases the optimized method works as good as the original one
- In 10% of cases the optimized method works better than the original one
- In 30% of cases the optimized method works worse than the original one

The mean increase in speed while using the optimized method was approximately 9.87 times.

However, let's not forget that the parameters of the algorithms were determined randomly. Because of this the statistics provided above contained test cases with non-optimal parameters, such as large cell size, as well.

Considering only those test cases in which the cell size was less than or equal to 11×11 pixels, the following numbers appear:

- In 62% of cases the optimized method works as good as the original one
- In 24% of cases the optimized method works better than the original one

- In 14% of cases the optimized method works worse than the original one

In this selection the average speedup was approximately 9.63 times.

Full test data can be found at (Ambartsumov, 2022b), and its shortened version in Appendix A.

CONCLUSION

The research has resulted in a promising method of image processing. It is not inferior in quality of image resizing to the seam carving method that is considered a standard in the image reformatting. At the same time, the optimized method is usually 2 – 10 times faster than the seam carving algorithm, being much less complex

The resulting algorithm is ideal for usage on mobile devices such as smartphones, tablets, or laptops. With the constantly growing number of images shared via portable devices and, consequently, the need for fast image editing, our method corresponds to the very present market demand.

The algorithm present in this paper may be further extended. The concepts of representativeness and nodal elements can be used for further improvement of processing speed and quality of resulting images.

The algorithm was implemented in Python programming language and published on GitHub (Ambartsumov, 2022a) under open MIT license. It is intended to be a sample for those, who want to recreate our quick image resizing method.

REFERENCES

Ambartsumov, H. (2022a). Agamendon/seam-carving. https://github.com/Agamendon/seam-carving

Ambartsumov, H. (2022b). Results. https://drive.google.com/drive/folders/1TSi0UU6e-oSlHnX4erMcIfFcHUTThHQk?usp=sharing

Avidan, S., & Shamir, A. (2007). Seam carving for content-aware image resizing. *ACM Trans. Graph.*, *26*(3), 10.

Rubinstein, M., Shamir, A., & Avidan, S. (2008). Improved seam carving for video retargeting. *ACM Transactions on Graphics (SIGGRAPH)*, *27*(3), 1–9.

Appendix A

TABLE OF TEST RESULTS

Name	Image height	Image width	Mode	Perc	Clust	Cell height	Cell width	Efunc	Or. meth. time	New meth. time	Diff. (New vs Old)
0.jpg	661	427	rem	12	18	16	9	forw	44.1	8.4	<
1.jpg	675	750	add	68	8	20	14	forw	557.8	39.9	=
2.jpg	543	403	rem	23	18	16	15	forw	57.0	8.9	>
3.jpg	923	623	rem	80	10	20	18	forw	566.0	48.9	=
4.jpg	605	963	rem	21	15	11	7	backw	333.3	24.2	=
5.jpg	988	830	rem	63	7	20	20	forw	936.3	62.0	=
6.jpg	830	459	add	79	12	14	14	forw	277.6	32.4	=
7.jpg	698	815	add	69	20	11	4	backw	664.5	62.1	=
8.jpg	263	485	rem	65	21	13	12	backw	83.5	10.7	<
9.jpg	809	523	add	49	15	13	13	forw	266.8	26.9	<
10.jpg	243	552	add	67	19	10	6	backw	99.5	12.4	<
11.jpg	281	689	rem	25	25	20	20	backw	89.1	9.7	<
12.jpg	923	663	rem	60	8	6	5	backw	535.7	56.4	=
13.jpg	207	308	rem	57	23	13	10	backw	24.2	5.0	<
14.jpg	561	352	rem	78	22	18	17	backw	104.7	18.4	<
15.jpg	395	879	add	75	15	9	6	backw	444.0	38.8	<
16.jpg	683	540	rem	17	8	10	9	forw	102.0	9.8	=
17.jpg	907	909	rem	12	12	20	19	forw	274.3	19.9	=
18.jpg	790	765	add	34	23	11	7	backw	409.8	39.1	=
19.jpg	340	209	rem	72	22	19	8	backw	20.6	5.9	<
20.jpg	566	520	add	15	18	20	4	backw	66.7	10.6	=
21.jpg	392	504	add	14	13	20	18	backw	37.8	5.9	=
22.jpg	348	934	rem	72	8	18	8	backw	438.2	26.4	<
23.jpg	730	224	rem	32	8	17	17	forw	32.6	6.3	=
24.jpg	334	413	add	17	17	16	15	forw	28.4	6.0	=
25.jpg	343	834	add	72	7	20	18	forw	361.7	24.7	=

26.jpg	841	983	rem	66	12	4	4	forw	1169.0	141.6	>
27.jpg	941	930	add	31	9	15	9	backw	678.5	40.0	=
28.jpg	427	548	add	29	13	19	19	backw	97.8	10.7	<
29.jpg	716	586	add	21	14	19	17	backw	147.6	15.7	=
30.jpg	818	751	add	14	20	20	9	backw	189.6	22.7	=
31.jpg	603	478	rem	39	12	19	17	backw	133.5	14.4	=
32.jpg	685	591	add	41	25	18	6	forw	262.2	26.2	>
33.jpg	784	986	add	29	6	19	18	backw	598.3	32.0	<
34.jpg	206	487	rem	80	19	17	15	backw	71.6	9.2	=
35.jpg	724	452	add	15	16	17	13	backw	62.7	10.6	=
36.jpg	880	998	rem	30	17	20	10	forw	728.8	43.4	=
37.jpg	394	538	rem	16	23	14	13	forw	55.3	8.4	=
38.jpg	883	969	add	66	24	5	5	forw	1195.1	129.2	>
39.jpg	892	307	rem	58	16	19	15	backw	107.5	18.4	<
40.jpg	598	764	add	33	6	14	13	backw	307.9	20.9	=
41.jpg	378	769	add	54	20	19	18	backw	283.1	24.1	=
42.jpg	490	483	rem	24	16	20	20	forw	80.9	9.6	<
43.jpg	274	806	add	40	21	20	14	forw	182.6	15.5	=
44.jpg	400	263	rem	40	13	14	9	forw	29.1	5.2	=
45.jpg	352	488	rem	32	11	20	20	forw	74.5	7.8	=
46.jpg	645	787	add	44	22	20	20	forw	439.7	35.4	<
47.jpg	252	764	add	11	11	18	16	forw	48.1	5.6	=
48.jpg	522	459	add	69	22	15	4	forw	165.1	21.7	=
49.jpg	439	470	rem	38	18	13	5	backw	93.2	11.9	=
50.jpg	764	605	rem	51	22	20	19	backw	329.7	32.0	=
51.jpg	634	562	rem	38	14	20	20	backw	192.3	18.6	<
52.jpg	684	523	add	51	16	19	18	forw	232.3	24.9	<
53.jpg	705	364	rem	18	20	19	4	forw	49.5	9.2	=
54.jpg	494	976	rem	33	9	20	18	forw	418.9	21.9	=
55.jpg	258	632	rem	60	12	7	7	backw	130.4	14.0	<
56.jpg	313	207	rem	75	8	13	7	backw	19.3	4.7	=
57.jpg	470	458	add	29	6	15	14	backw	76.4	8.4	=
58.jpg	734	953	add	57	18	10	4	backw	842.4	72.9	>
59.jpg	266	233	rem	14	8	19	14	forw	6.0	1.5	<

60.jpg	409	747	add	33	12	17	11	forw	204.2	14.7	<
61.jpg	778	573	add	51	18	18	8	backw	311.2	30.5	=
62.jpg	787	721	add	11	12	17	17	forw	139.0	14.5	=
63.jpg	838	908	rem	78	23	12	12	backw	1039.2	78.8	<
64.jpg	952	985	rem	29	12	19	17	forw	730.4	37.8	<
65.jpg	570	707	add	40	18	19	16	forw	296.5	25.5	<
66.jpg	430	444	rem	33	13	11	9	forw	76.1	8.7	=
67.jpg	659	388	add	15	5	15	13	backw	42.3	6.1	=
68.jpg	401	487	add	69	15	19	18	forw	141.3	16.4	=
69.jpg	892	562	rem	71	23	10	10	backw	406.7	45.6	=
70.jpg	771	638	add	13	6	19	17	forw	123.0	11.3	=
71.jpg	575	780	rem	41	5	20	19	backw	364.2	22.0	<
72.jpg	847	688	add	28	25	20	20	backw	310.1	32.7	>
73.jpg	582	505	add	55	5	15	7	backw	182.5	17.8	<
74.jpg	910	310	rem	11	6	15	10	forw	29.8	5.1	=
75.jpg	569	425	rem	56	13	14	5	forw	136.8	15.6	=
77.jpg	266	522	add	67	9	11	8	backw	98.8	11.0	=
78.jpg	964	612	add	46	6	14	7	forw	415.2	32.0	=
79.jpg	471	396	add	32	22	10	7	forw	65.2	10.7	>
80.jpg	468	368	add	11	15	16	6	backw	20.9	4.9	=
81.jpg	227	374	rem	37	10	10	10	backw	28.6	4.3	=
82.jpg	696	512	rem	34	18	8	5	backw	163.4	20.9	=
83.jpg	660	927	rem	54	19	18	9	forw	727.3	44.9	<
84.jpg	327	780	add	15	18	10	6	forw	89.9	10.6	=
85.jpg	964	466	add	79	18	19	4	backw	313.9	40.7	=
86.jpg	463	928	add	32	10	19	17	backw	333.1	20.5	<
87.jpg	895	999	rem	75	19	9	7	backw	1295.6	103.5	=
88.jpg	474	512	rem	33	11	9	9	backw	102.0	11.3	>
89.jpg	593	540	add	36	17	15	13	forw	164.7	18.1	<
90.jpg	876	237	add	18	6	16	4	backw	25.2	5.3	>
91.jpg	564	899	add	10	17	18	15	forw	136.5	16.0	=
92.jpg	287	614	add	23	24	19	12	backw	67.2	8.7	=
93.jpg	239	911	add	54	25	11	8	backw	246.8	21.1	=
94.jpg	664	776	rem	80	10	19	17	backw	600.3	45.2	<

95.jpg	353	690	add	50	25	20	13	backw	193.9	19.2	>
96.jpg	649	680	add	71	22	16	14	backw	431.0	41.3	<
97.jpg	986	903	rem	56	11	20	19	backw	1040.2	58.8	=
98.jpg	223	506	add	50	20	13	6	backw	65.2	8.2	=
99.jpg	531	871	add	14	10	18	15	backw	166.5	13.1	=