UNIVERSIDADE ESTADUAL DO RIO GRANDE DO SUL

GABRIEL HENRIQUES KUNZ

Implementação do método de Seam Carving com diferentes algoritmos de mapeamento de energia

Guaíba - RS

Janeiro/2021

IMPLEMENTAÇÃO DO MÉTODO DE SEAM CARVING COM DIFERENTES ALGORITMOS DE MAPEAMENTO DE ENERGIA

Trabalho de conclusão de curso apresentado como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação

Orientador: Dra. Adriane Parraga

Guaíba - RS

Janeiro/2021

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO

K96i Kunz, Gabriel Henriques

Implementação do método de Seam Carvin com diferentes algoritmos de mapeamento de energia / Gabriel Henriques Kunz. – Guaíba/RS, 2021.

69 p. : il.

Orientadora: Prof.ª Dr.ª Adriane Parraga.

Trabalho de conclusão (Graduação) - Universidade Estadual do Rio Grande do Sul, Curso de Bacharelado em Engenharia da Computação, Unidade Universitária em Guaíba, 2021.

1. Seam Carving. 2. Redimensionamento. 3. Processamento de imagem. I. Parraga, Adriane. II. Título.

Daniella Vieira Magnus - Bibliotecária – CRB 10/2233

GABRIEL HENRIQUES KUNZ

IMPLEMENTAÇÃO DO MÉTODO DE SEAM CARVING COM DIFERENTES ALGORITMOS DE MAPEAMENTO DE ENERGIA

Monografia sob o título "Implementação do método de Seam Carving com diferentes algoritmos de mapeamento de energia", defendida por Gabriel Henriques Kunz e aprovada em XX de XX de XXXX, em Guaíba, estado do Rio Grande do Sul, pela banca examinadora constituída pelos professores:

Guaíba - RS, 28 de Janeiro de 2021:

Dra. Adriane Parraga
Orientadora
Universidade Estadual do Rio Grande do Sul

Dra. Leticia Vieira Guimarães
Universidade Estadual do Rio Grande do Sul

MSc. Raphael Ruschel
University of California, Santa Barbara

Aos meus pais, por sempre fazerem de tudo
para que o estudo se mantivesse presente na minha vida.

AGRADECIMENTOS

Em primeiro lugar agradeço a minha família por todo o apoio e incentivo dado a mim durante o curso e por todo esforço em me garantir condições de estudo e ensino de qualidade, sempre priorizando meu desenvolvimento e meu futuro. Gostaria de agradecer também aos excelentes professores que compõem o corpo docente da UERGS e compartilharam comigo o seu conhecimento nas suas respectivas áreas durante todos esses anos de curso. Um agradecimento especial para a Professora Dra. Adriane Parraga pela ajuda como orientadora nesse trabalho e pela sua extrema dedicação ao ensino e desenvolvimento de seus alunos, seja como professora ou como coordenadora de curso.

RESUMO

O redimensionamento de imagens sempre foi realizado sacrificando algum elemento essencial, seja a proporção ou forma. Desde 2007, o método de Seam Carving vêm sendo utilizado e aprimorado para que o conteúdo subjetivo das imagens seja levado em consideração durante o redimensionamento. Por ser um método baseado na identificação de objetos relevantes em uma imagem, os algoritmos por trás dessa identificação são essenciais para o sucesso da implementação do método. Esse trabalho realiza a implementação do método de Seam Carving utilizando diferentes algoritmos de detecção de borda para identificar zonas de alta energia que devem ser preservadas na imagem durante o redimensionamento. Os resultados são então comparados a fim de identificar a existência de um ou mais algoritmos que proporcionem melhores resultados. O uso de métricas escalares e visuais também é feito com o objetivo de encontrar um padrão que possa ser usado para suportar e definir a eficácia de cada implementação do método. Entre os algoritmos testados, os baseados em primeira derivada se mostraram mais eficientes na preservação de conteúdos na imagem. As métricas escalares de energia média e entropia de Shannon foram posteriormente substituídas por uma métrica baseada no algoritmo SIFT que se mostrou adequado para mensurar a qualidade do método *Seam Carving*.

Palavras-chave: Redimensionamento de imagem. Seam Carving. Mapeamento de energia. Redimensionamento com base em conteúdo.

ABSTRACT

Image resizing was always performed by sacrificing essential elements, such as proportion or shape. Since 2007, the Seam Carving method has been used and enhanced so that the subjective content of images is taken into consideration during its resizing. Being a method based on the identification of relevant objects in an image, the algorithms behid that identification are essential for the implementation success. This paper perfoms the implementation of the Seam Carving method using different edge detection algorithms to identify high-energy regions that must be preserved during the resizing. The results are then compared to find out the existence of one or more algorithms that can provide better overall results. The use of scalar and visual metrics is also performed with the objective to find a pattern that can be used to support and define the efficiency of each method implementation. Among the tested algorithms, those based on the first derivative are found to be more efficient in preserving content in the image. The scalar metrics of mean energy and Shannon's entropy were later replaced by a metric based on the SIFT algorithm which is suitable to measure the quality of the method.

Keywords: Image resizing. Seam Carving. Energy mapping. Content-aware scaling.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Com a evolução da tecnologia em geral e o aumento da capacidade de transmissão de dados, o uso de imagens digitais se tornou cada vez mais comum no nosso dia a dia. Dispositivos dos mais diversos tamanhos são utilizados para o consumo dessas imagens, como por exemplo laptops, tablets, celulares e até mesmo smartwatches. Para que uma imagem digital alcance toda essa gama de aparelhos, é necessário muitas vezes redimensioná-la.

Ao redimensionar uma imagem pelos métodos convencionais existentes, recorte e alteração de proporção de tela, o tamanho e a forma daquilo que é representado na imagem acaba sendo alterado, causando distorções no conteúdo e perda de informações relevantes. Por esse motivo, o redimensionamento é um processo ainda evitado nas mídias atuais. Exemplos disso são plataformas de streaming de vídeo e páginas web com o ajuste de conteúdo para diferentes tipos de tela. Com o objetivo de corrigir esse problema, Avidan e Shamir (2007) propuseram um algoritmo de redimensionamento que leva em consideração também o conteúdo da imagem e não somente suas propriedades geométricas, chamado de *Seam Carving*.

O método *Seam Carving*, cujo nome pode ser traduzido de forma literal para *Escultura de Costuras*, consiste na retirada ou inserção de segmentos de pixels conectados (carvings ou linhas de costura) de uma imagem a fim de esculpir a mesma para um tamanho reduzido ou ampliado.

Quando o algoritmo foi inicialmente desenvolvido em 2007, a escolha dos *carvings* alterados era feita com base na *backward energy*, capaz de exibir quais segmentos eram os que continham menos informação relevante na imagem (AVIDAN; SHAMIR, 2007). No ano seguinte, o conceito de *forward energy* foi introduzido para que o *seam carving* pudesse ser aplicado de forma eficaz também em vídeos, considerando o impacto que a alteração do *carving* iria causar na energia total da imagem ou frame (RUBINSTEIN; SHAMIR; AVIDAN, 2008). Para a obtenção de melhores resultados, é possível aplicar o método *Seam Carving* em conjunto com outros processos de redimensionamento, como o de *scaling* (alteração de escala/proporção de tela) (DONG et al., 2009). A técnica se mostrou tão relevante que hoje em dia se encontra implementada em softwares profissionais de edição de imagem como o Adobe Photoshop e o GIMP.

Em 2020, foi publicado um artigo de estudo demonstrando a utilização do método na área médica. Nesse artigo, o redimensionamento de imagens médicas como ressonâncias foi realizado a fim eliminar a distorção causada quando as imagens eram exibidas em aparelhos móveis a bordo de ambulâncias (QINGFANG et al., 2020).

Por mais que o foco do *Seam Carving* seja o redimensionamento de imagens, o mesmo pode ser usado em outras aplicações na área de processamento de imagens, como por exemplo de auxílio na segmentação de linhas de texto de manuscritos. Sobre esse tópico, em 2018 (DALDALI; SOUHAR, 2018) e em 2020 (BERRICHE; AL-MUTAIRY, 2020) o método foi implementado de forma que a remoção dos *seams* fosse feita em grandes áreas sem texto escrito de forma a padronizar o espaçamento entre as linhas dos textos permitindo a digitalização e arquivamento de documentos históricos e importantes.

1.1 Objetivo

Por se tratar de um método que utiliza algoritmos de detecção de borda para o cálculo da energia dos pixels e relevância de objetos na imagem, existe uma grande variedade de formas de se implementar o

Seam Carving. Este trabalho irá apresentar a implementação do método e a comparação dos resultados de seis algoritmos diferentes utilizados para o mapeamento de energia.

Os algoritmos utilizados são: Sobel, Prewitt, Laplaciano, Roberts, Canny e Forward Energy. Para a comparação dos resultados, uma análise visual suportada por três métricas será utilizada. As métricas são: Energia média, Entropia de Shannon e Correspondência de Características (*Feature Matching*).

1.2 Organização do trabalho

O trabalho está organizado da seguinte forma: No Capítulo 2 serão apresentados conceitos de processamento de imagem e *Seam Carving*. No Capítulo 3 a metodologia do trabalho é descrita. No Capítulo 4 serão listados os resultados e discussões. E por fim, no Capítulo 5 serão mostradas as conclusões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Para uma melhor compreensão do processo de *Seam Carving*, este capítulo apresenta grande parte dos conceitos essenciais referentes a imagens, tais como suas definições matemáticas e computacionais e também como elas são analisadas e modificadas digitalmente.

2.1 Imagens digitais

2.1.1 Imagens em escala de cinza

Uma imagem digital em escala de cinza é definida matematicamente através de uma função bidimensional $f(x, y)$ onde a amplitude de cada ponto da função, chamado de *pixel*, corresponde a intensidade monocromática daquele mesmo ponto da imagem. Se mapeada utilizando valores inteiros, a função define a imagem como uma matriz de M linhas e N colunas. Dessa forma, as variáveis x e y constituem as coordenadas dos pixels presentes na imagem, onde $x = 0, 1, 2, 3, ..., M-1$ e $y = 0, 1, 2, 3, ..., N-1$.

$$f(x,y) = \begin{pmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{pmatrix} \tag{2.1}$$

O valor de energia ou intensidade de cada pixel depende da quantidade de bytes utilizados para amostrar a imagem digitalmente. Usualmente é utilizada a taxa de 1 byte por pixel ($8bits/pixel$) para fotografias e imagens, proporcionando um total $2^8 = 256$ níveis de intensidade para cada pixel. A Tabela 1 apresenta outros valores de taxa de amostragem usados para imagens digitais.

Tabela 1 – Taxas de amostragem para os tipos mais comuns de imagens digitais.

Escala de cinza (Imagens de intensidade):

Canais	*Bits/Pix.*	*Níveis de intensidade*	*Uso*
1	1	[0,1]	Imagens binárias: documentos, ilustrações, fax
1	8	[0,255]	Universal: fotos, scan, impressões
1	12	[0,4095]	Alta qualidade: fotos, scan, impressões
1	14	[0,16383]	Profisional: fotos, scan, impressões
1	16	[0,65535]	Altíssima qualidade: medicina, astronomia

Fonte: (BURGER; BURGE, 2016)

2.1.2 Imagens coloridas

Para a representação digital de imagens coloridas é amplamente utilizado o esquema de componentes *Red*, *Green* & *Blue* (RGB), onde cada pixel possui uma combinação de 3 intensidades de vermelho, verde e azul. Conforme apresentado na Figura 1 e na equação 2.4, essa combinação cria um vetor coluna para um pixel z onde z_1, z_2 e z_3 representam as intensidades de vermelho, verde e azul respectivamente:

$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \tag{2.2}$$

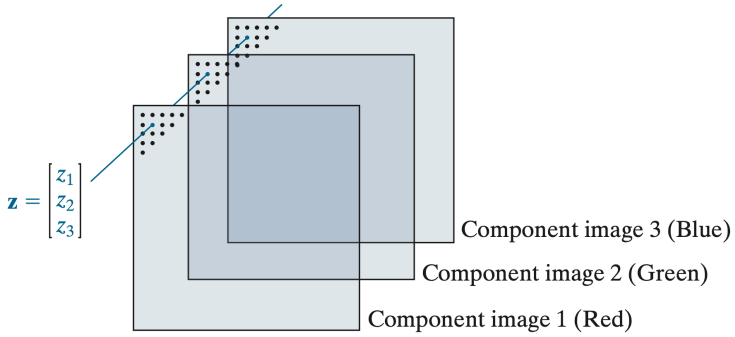


Figura 1 – Componentes de uma imagem colorida RGB. Fonte: (GONZALEZ; WOODS, 2018)

Imagens compostas de mais de um componente de intensidade são também chamadas de imagens multiespectrais e cada componente pode também ser referenciada como canal. Com a utilização de 8 bits de taxa de amostragem (2^8 níveis de intensidade) para cada canal em uma imagem RGB, é possível representar mais de 16 milhões de cores como mostra a equação 2.5.

$$\text{Combinações de intensidades RGB} = 2^8 \cdot 2^8 \cdot 2^8 = 2^{24} = 16.777.216 \tag{2.3}$$

O esquema de cores RGB é baseado em um conceito de cores aditivas. De forma oposta, existe também o esquema de cores *Cyan*, *Magenta*, *Yellow* & *Black* (*Key*) (CMYK). Esse modelo utiliza as cores subtrativas ciano, magenta e amarelo misturadas sobre um fundo usualmente branco, sendo por esse motivo encontrado em aplicações de pré-impressão. A Figura 2 mostra ambos esquemas de cores lado a lado e a Tabela 2 apresenta as taxas de amostragem e aplicações comuns de imagens coloridas.

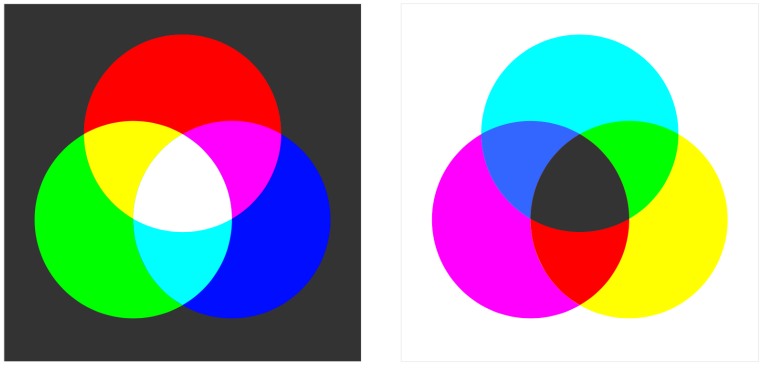


Figura 2 – Esquemas de cores RGB e CMYK. No lado esquerdo: (RGB) cores aditivas vermelho, verde e azul misturadas para produzir ciano, magenta, amarelo e branco. No lado direito: (CMYK) cores subtrativas ciano, magenta e amarelo misturadas para produzir vermelho, verde, azul e preto. Fonte: (SZELISKI, 2011)

Tabela 2 – Taxas de amostragem para os tipos mais comuns de imagens digitais coloridas.

Imagens coloridas:

Canais	*Bits/Pix.*	*Níveis de intensidade*	*Uso*
3	24	$[0,255]^3$	RGB, universal: fotos, scan, impressões
3	36	$[0,4095]^3$	RGB, alta qualidade: fotos, scan, impressões
3	42	$[0,16383]^3$	RGB, profissional: fotos, scan, impressões
4	32	$[0,255]^4$	CMYK, pré-impressão digital

Fonte: (BURGER; BURGE, 2016)

2.2 Filtragem

Para modificar uma imagem, existem basicamente dois tipos de processos: as operações pontuais e a filtragem. Com operações pontuais, cada pixel da imagem é alterado levando em consideração um critério que analisa somente o valor do próprio pixel a ser modificado, independente do restante da imagem. Esse tipo de modificação é utilizado para operações mais simples como alteração de brilho e contraste, inversão de cor, transformação de cores, entre outros.

Para processos mais complexos, como redução de ruídos, aumento de nitidez e detecção de bordas, é necessário a análise de pixels vizinhos para uma alteração proporcional de um pixel na imagem. Essa análise é realizada através da filtragem, operação que consiste na aplicação de filtros lineares em uma imagem, denominados dessa forma pois a combinação dos valores dos pixels vizinhos é feita de forma linear (BURGER; BURGE, 2016).

2.2.1 Correlação e Convolução

Dois conceitos importantes para o entendimento de filtragem em imagens digitais são os de correlação e convolução. A correlação consiste no processo de mover um kernel, também chamado de máscara ou filtro, pela imagem e calcular a soma dos produtos em cada posição. Matematicamente, a correlação de um kernel $w(x, y)$ com uma imagem $f(x, y)$ é definida através da seguinte expressão:

$$w(x, y) \oplus f(x, y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i, j) f(x + i, y + j) \tag{2.4}$$

A convolução consiste no mesmo processo, porém com o kernel rotacionado a 180^o. A diferença entre os dois conceitos pode ser melhor compreendida com o exemplo unidimensional apresentado na Figura 3 utilizando uma função f e um kernel w. Em uma imagem digital 2-D, a rotação 180^o equivale a inversão do kernel em relação a um eixo e, depois, em relação ao outro (GONZALEZ; WOODS, 2018). A expressão a seguir apresenta a definição matemática da convolução de um kernel $w(x, y)$ com uma imagem $f(x, y)$:

$$w(x, y) * f(x, y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i, j) f(x - i, y - j) \tag{2.5}$$

Para qualquer filtro linear, o tamanho e forma da região de suporte, assim como o peso individual de cada pixel, são especificados pelo kernel do filtro ou máscara $H(i, j)$. O tamanho do kernel H é igual ao tamanho da região filtrada, e cada elemento de $H(i, j)$ especifica o peso do pixel correspondente na soma (BURGER; BURGE, 2016). Para um filtro de suavização 3x3, que consiste na alteração do pixel para que o mesmo fique entre um valor médio de seus vizinhos, o seguinte kernel é utilizado:

$$H = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \mathbf{\frac{1}{9}} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.6}$$

Onde cada um dos nove pixels contribui com um nono de seu valor para o resultado final.

O processo de aplicação do filtro é ilustrado na Figura 4, onde os passos a seguir são feitos em cada posição (u, v) da imagem:

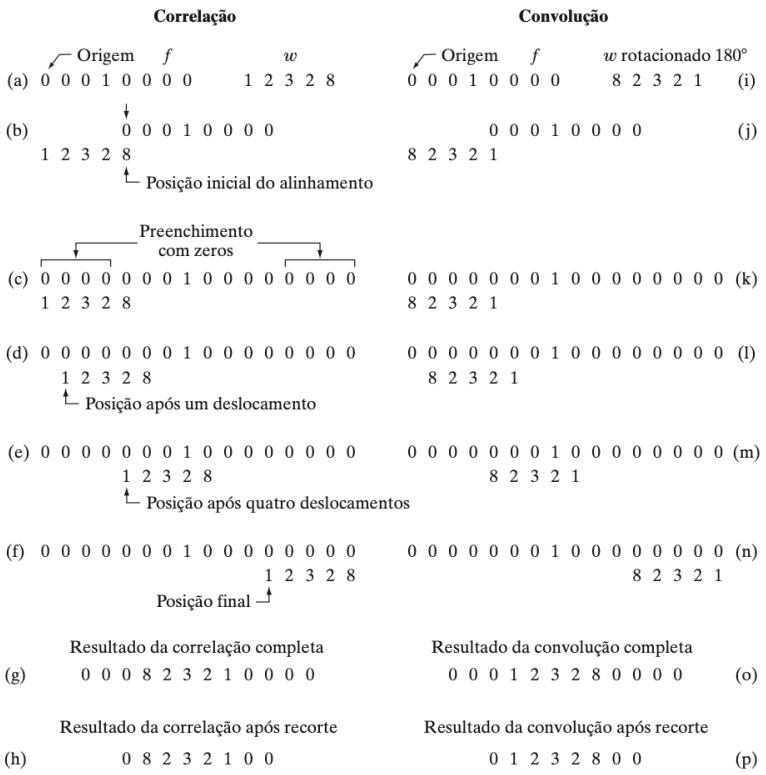


Figura 3 – Ilustração de convolução e correlação unidimensional de um filtro com impulso unitário discreto. Tanto a correlação quanto a convolução são funções de deslocamento. Fonte: (GONZALEZ; WOODS, 2010)

1. O kernel H é deslocado sobre a imagem original I de forma que sua origem $H(0,0)$ coincida com a posição atual (u,v) da imagem.
2. Todos os coeficientes do kernel $H(i,j)$ são multiplicados pelo elemento $I(u+i,v+j)$ correspondente da imagem, e os resultados são somados.
3. Por fim, a soma resultante é armazenada na posição atual da nova imagem $I'(u,v)$.

2.3 Tipos de filtros lineares

Filtro passa-baixa

Filtros passa-baixa recebem esse nome pois têm como objetivo permitir a passagem de componentes de baixa frequência durante a filtragem. Em uma imagem digital, componentes de baixa frequência representam nenhuma ou mudanças suaves de intensidades de pixels. Por isso, a filtragem passa-baixa é utilizada para suavização de imagens, removendo ruídos da mesma através da identificação e alteração de pixels que possuem grande diferença de intensidade com relação a seus vizinhos.

Um dos exemplos mais simples de filtro passa-baixa é o que utiliza o cálculo de média da região de suporte, apresentado na equação 2.8 da subseção anterior. Nesse tipo de filtro, cada pixel contribui com valor proporcional a média do tamanho do kernel. Em um kernel de tamanho 3x3, cada pixel contribui com um nono de seu valor, em um kernel 5x5, um vinte e cinco avos, e assim por diante. Quanto maior o tamanho do kernel utilizado, mais borrada se torna a imagem resultante.

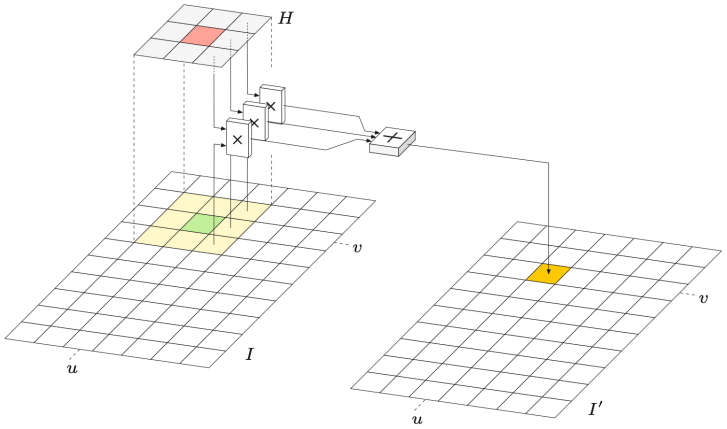


Figura 4 – Operação de filtragem linear. O kernel H é colocado com a sua origem na posição (u, v) da imagem. Cada coeficiente do filtro $H(i, j)$ é multiplicado pelo pixel correspondente da imagem $I(u + i, v + j)$, os resultados são somados e a soma final é inserida como o novo valor do pixel $I'(u, v)$. Fonte: (BURGER; BURGE, 2016)

No exemplo de filtragem passa-baixa mostrado na Figura 5, o seguinte kernel de média foi utilizado:

$$H = \begin{bmatrix} \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \end{bmatrix} = \frac{1}{25} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{1} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.7}$$

Figura 5 – Exemplo de aplicação de filtro passa-baixa utilizando o cálculo de média com kernel 5x5. Fonte: Autor

Filtro passa-alta

De forma oposta, com a finalidade de realçar as transições de intensidade através da permissão da passagem de componentes de alta frequência, existem os filtros passa-alta. Que por sua vez são amplamente utilizados com a finalidade de detectar bordas em imagens digitais.

Em uma função digital, suas derivadas são descritas em termos de diferenças e embora existam diversas maneiras de defini-las, é necessário que qualquer definição utilizada para a *primeira derivada* (1) seja zero em áreas de intensidade constante; (2) seja diferente de zero no início de um degrau ou rampa de intensidade; e (3) seja diferente de zero ao longo das rampas. De forma similar, qualquer definição de uma *segunda derivada* (1) deve ser zero em áreas constantes; (2) deve ser diferente de zero no início *e* no final de um degrau ou rampa de intensidade; e (3) deve ser zero ao longo de rampas de inclinação constante (GONZALEZ; WOODS, 2010).

Resultantes de uma expansão da série de Taylor, as seguintes definições de primeira e segunda derivada, respectivamente, são utilizadas para satisfazer as condições mencionadas:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{2.8}$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \tag{2.9}$$

O uso de derivada parcial é feito para manter a notação inalterada quando outras varáveis são adicionadas na função. Como por exemplo uma imagem bidimensional f(x,y).

Baseado nessas duas equações de diferenças, é possível separar alguns dos filtros passa-alta mais comuns em dois grupos: os de primeira e os de segunda derivada.

Filtros de primeira derivada

Roberts

Um dos detectores de bordas mais simples é o operador Roberts. Ele consiste na convolução de dois kernels 2x2 (H_1^R e H_2^R) para o cálculo do gradiente ao longo das diagonais da imagem:

$$H_1^R = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.10}$$

$$H_2^R = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{2.11}$$

É possível ver na Figura 6 que nem todas as bordas da imagem são detectadas na operação. Isso ocorre porque o sucesso da detecção de bordas do operador depende muito da angulação das mesmas na imagem, já que o kernel possui uma abordagem diagonal. A imagem mais à direita da figura consiste na sobreposição das bordas encontradas pelos kernels de Roberts, apresentados pelas imagens centrais.

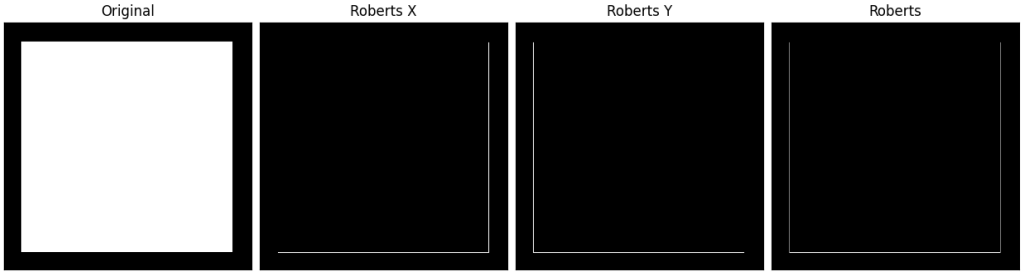


Figura 6 – Exemplo de detecção de borda com operador Roberts. Fonte: Autor

Prewitt

Com o uso de kernels 3x3, o algoritmo Prewitt calcula as derivadas parciais da imagem para a detecção de bordas verticais e horizontais:

$$H_x^P = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \tag{2.12}$$

$$H_y^P = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.13}$$

Conforme o resultado na Figura 7, algumas bordas também são perdidas devido a orientação das mesmas e a direção da varredura feita com os kernels.

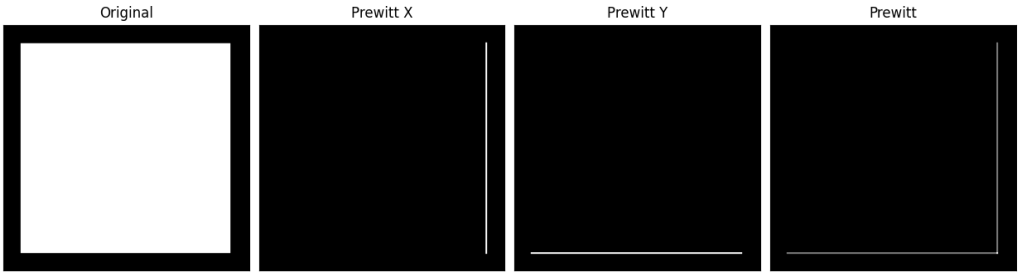


Figura 7 – Exemplo de detecção de borda com operador Prewitt. Fonte: Autor

Sobel

De forma similar ao Prewitt, o operador Sobel realiza o mesmo cálculo de gradiente na imagem, porém adicionando um peso maior aos eixos centrais da zona de suporte.

$$H_x^S = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{2.14}$$

$$H_y^Y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.15}$$

A Figura 8 apresenta um melhor resultado em relação aos filtros citados anteriormente. Detectando, no exemplo utilizado, todas as bordas existentes.

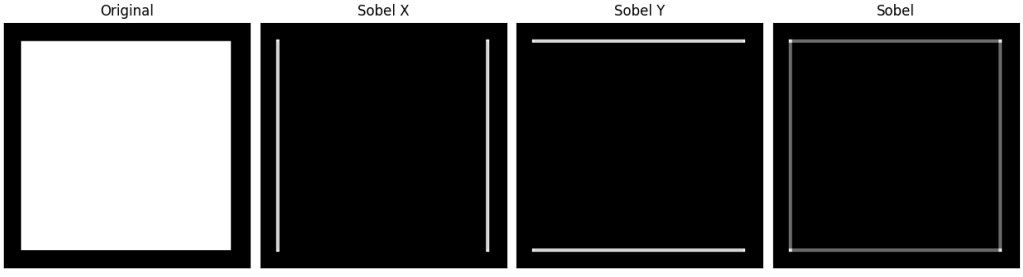


Figura 8 – Exemplo de detecção de borda com operador Sobel. Fonte: Autor

Filtros de segunda derivada

Laplaciano

O Laplaciano (∇^2) de uma função 2D $f(x, y)$ é definido como a soma das segundas derivadas parciais ao longo das direções x e y (BURGER; BURGE, 2016):

$$(\nabla^2 f)(x, y) = \frac{\partial^2 f}{\partial^2 x}(x, y) + \frac{\partial^2 f}{\partial^2 y}(x, y) \tag{2.16}$$

Cada derivada parcial pode ser estimada através de dois filtros unidimensionais H_x^L e H_y^L que por sua vez podem ser combinados a fim de obter o filtro Laplaciano 2D H^L:

$$H_x^L = [1 \quad -2 \quad 1] \tag{2.17}$$

$$H_y^L = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \tag{2.18}$$

$$H^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.19}$$

A Figura 9 mostra o resultado do laplaciano na detecção de bordas em uma imagem digital.

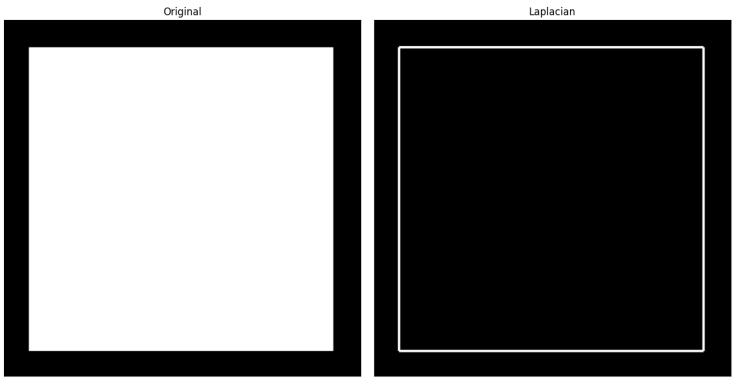


Figura 9 – Exemplo de detecção de borda com operador Laplaciano. Fonte: Autor

2.4 Seam carving

Em 2007, Shai Avidan e Ariel Shamir desenvolveram e publicaram um método de redimensionamento de imagem com base no seu conteúdo (AVIDAN; SHAMIR, 2007). Nesse método, um operador é utilizado para definir, com base na energia dos pixels, regiões que podem ser removidas ou introduzidas na imagem para diminuir ou aumentar suas dimensões sem distorcer objetos relevantes para o propósito da imagem. Essas regiões são denomidas *seams* e consitem em caminhos otimizados de 8 pixels conectados, verticalmente ou horizontalmente, onde a otimização de cada caminho é definida por uma função de energia. Dando o nome de *Seam Carving* ao processo.

A simples remoção de pixels individuais destruiria a forma retangular da imagem já que isso resultaria em linhas de tamanhos diferentes. Remover colunas completas para contornar esse problema

impactaria de forma muito significativa o conteúdo. Para então manter a forma da imagem redimensionada, a estratégia de *Seam Carving* propôs a definição de internal seams.

2.4.1 Definição de seam

Em uma imagem $\mathbf{I}$ com dimensões $i \times j$, um seam é definido por um caminho vertical de pixels conectados na imagem de cima para baixo, contendo um, e apenas um, pixel em cada linha da imagem. Essa conexão é dada através dos 8 pixels de uma zona de suporte 3x3, ou seja, cada pixel do seam deve estar em contato com o pixel da próxima linha, seja através da borda ou canto da região.

A conexão dos pixels em um seam é feita com base na energia do pixel. Para encontrar um seam em uma imagem, uma matriz auxiliar $M(i, j)$ é utilizada para armazenar o menor valor de energia encontrado até o pixel que está sendo analisado (KARANTH, 2018):

$$M(i,j) = e(i,j) + min(M(i-1,j-1), M(i-1,j), M(i-1,j+1)) \tag{2.20}$$

Essa matriz contem o menor valor de energia de cada posição, considerando todos os possíveis seams até o ponto analisado desde o topo da imagem. Portanto, a energia mínima necessária para percorrer a imagem de cima para baixo é armazenada na última linha de $M(i, j)$ (KARANTH, 2018).

2.4.2 Operador de seleção do Seam Carving

Com o objetivo de preservar o conteúdo relevante da imagem, o algoritmo deve ser capaz de decidir quais pixels podem ser removidos a fim de minimizar o impacto no resultado final. Para decidir a relevância dos pixels, a seguinte função de energia é utilizada:

$$e_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I} + \frac{\partial}{\partial y}\mathbf{I}| \tag{2.21}$$

Conforme visto na seção 2.3 e apresentado na equação 2.10 a derivada parcial de cada componente de uma imagem digital pode ser descrita através de equações de diferenças.

2.4.3 Preservação de energia

Para avaliar o desempenho do algoritmo após o redimensionamento, a média de energia (e_m) de todos os pixels da imagem pode ser usada através da seguinte equação:

$$e_m = \frac{1}{|\mathbf{I}|}\sum_{p \in I} e(p) \tag{2.22}$$

A remoção de pixels aleatórios da imagem mantém o resultado da média inalterado, porém, em um redimensionamento que leva em consideração o conteúdo como é o caso do *Seam Carving*, o resultado deve aumentar. Visto que o algoritmo remove pixels de baixa energia ao mesmo tempo que mantém aqueles com maior energia.

2.4.4 Seleção de seams para remoção ou inserção

2.4.4.1 Backward e forward energy

Originalmente, o método proposto por Shamir e Avidan focava na remoção do seams que continham a menor quantidade total de energia. Sem considerar a energia que era introduzida na imagem após a

remoção dos seams. Baseado nisso, Rubinstein em 2008 propôs um novo critério para encontrar o melhores seams para serem removidos da imagem.

Esse novo critério direciona a análise para o resultado da imagem, olhando adiante no tempo após a remoção do seam ao invés de somente olhar atrás, antes da remoção do mesmo da imagem. Para isso, a cada iteração, o seam cuja remoção introduz o menor valor de energia na imagem é buscado. Esses seams não são necessariamente os que possuem a menor energia, mas que irão deixar menos vestígios na imagem resultante após sua remoção (RUBINSTEIN; SHAMIR; AVIDAN, 2008).

Com esse novo critério, os termos backward e forward energy passaram a ser bastante utilizados quando se trabalhando com *Seam Carving*. O backward energy se refere a operação original proposta por Shamir e Avidan enquanto a forward energy se refere a proposta de Rubinstein.

2.4.5 Estado da arte e avanços no método

Com o passar dos anos o método foi sendo aperfeiçoado através do uso de novos operadores de energia na detecção dos seams e da combinação com outros algoritmos de redimensionamento já conhecidos e utilizados. Como em 2009, onde Dong e Zhou realizaram a combinação do *Seam Carving* com o ajuste de proporção para proteger o aspecto visual global da imagem e de estruturas locais na mesma (DONG et al., 2009).

Devido ao fato dos seams serem removidos unidirecionalmente, isto é, somente verticalmente ou horizontalmente, grandes distorções foram detectadas quando os objetos relevantes ocupavam em sua maioria a imagem na vertical ou horizontal. Para solucionar esse problema, Shi e Peng propuseram em 2010 uma dependência vertical e horizontal durante a aplicação do método, chamada de *Seam Carving* Bi-direcional. Dessa forma, se um objeto ocupasse a imagem verticalmente, a transferência da redução da largura para o aumento da altura era realizado. Minimizando a distorção no objeto durante o redimensionamento (SHI et al., 2010).

Por mais que a proposta inicial de Avidan e Shamir tenha sido feita com o intuito de reduzir o tamanho de uma imagem, desde suas primeiras implementações foi notado que o mesmo algoritmo poderia ser usado para ampliação. Wang e Yuan exploraram essa aplicação do método em 2013 propondo um redimensionamento de imagens de alta qualidade focado no processo de expansão mantendo a propriedade de alta resolução da imagem de entrada (WANG; YUAN, 2013).

Em 2016, Emaduldeen e Hassan destacaram a importância do pré-processamento da imagem durante a aplicação do Seam Carving para melhores resultados. Também realizaram um estudo comparativo da preservação de energia de imagens redimensionadas utilizando seis diferentes tipos de detectores de borda como operadores de energia (EMADULDEEN; HASSAN, 2016).

Saindo do escopo de redimensionamento de imagens, Daldali e Souhar demonstraram em 2018 a utilização do *Seam Carving* para a segmentação de linhas de texto em manuscritos árabes e também japoneses e em latim (DALDALI; SOUHAR, 2018).

Atualmente, em 2020, o algoritmo de *Seam Carving* tem se mostrado relevante também na área médica, onde Qingfang e seus colegas propuseram o uso do método baseado em preservação de contorno para adaptação de alta qualidade de imagens médicas. A proposta foi feita para uso em sistemas inteligentes embarcados em ambulâncias que possuem telas e monitores de diferentes tamanhos, tornando a integridade da proporção dos objetos em imagens exibidas nelas de extrema importância (QINGFANG et al., 2020).

3 METODOLOGIA

Este capítulo aborda as ferramentas e metodologias utilizadas no desenvolvimento do trabalho. O código fonte desenvolvido bem como o guia rápido de instalação e execução do programa pode ser acessado através do repositório aberto hospedado na plataforma GitHub e disponível no link <https://github.com/GabrielKunz/seam-carving>.

3.1 Python

Para a implementação do *Seam Carving* foi utilizada a linguagem de programação Python em sua última release no momento da publicação desse trabalho, 3.8.6 (PSF, 2020). Apresentada pela primeira vez em 1991, Python é uma linguagem interpretada e de alto nível desenvolvida com o propósito de melhorar a legibilidade do código ajudando seus usuários a escreverem códigos claros e assertivos tanto para projetos de pequena quanto larga escala. O núcleo da linguagem é bastante simples e a sua grande gama funcionalidades se dá através do uso de bibliotecas desenvolvidas por terceiros e importadas no início da execução do programa.

3.1.1 Bibliotecas

- **Numba** Por se tratar de uma linguagem interpretada, o Python muitas vezes pode não oferecer a melhor performance quando executada. Para contornar esse problema foi utilizada a biblioteca Numba que permite que trechos isolados de código sejam previamente compilados gerando códigos de máquina que são chamados em seguida durante a execução do programa. (NUMBA, 2020)

- **NumPy** Conforme citado na seção 2.1, imagens digitais são tratadas na computação como funções bidimensionais (quando em escala de cinza) e tridimensionais (quando possuem múltiplos canais) e importadas como grandes matrizes para manipulação a nível de código. A biblioteca NumPy provê suporte para esses tipos de estruturas computacionais permitindo que operações matemáticas de alto nível sejam realizadas com elas. (NUMPY, 2020)

- **OpenCV** A biblioteca de código aberto OpenCV oferece milhares de algoritmos de processamento de imagem, visão computacional e aprendizado de máquina. Todos os algoritmos da biblioteca são altamente otimizados e auxiliam na execução de processos custosos em questão de processamento, como por exemplo a convolução de kernels com imagens para filtragem linear. (OPENCV, 2020)

3.1.2 Ambiente de desenvolvimento

O programa foi desenvolvido no sistema operacional macOS 10.15.16 e executado através da Z shell integrada nele. Como a maioria dos sistemas baseados em Unix, o Python 3.8 já vem instalado por padrão, sendo necessário somente a instalação das bibliotecas utilizadas no programa. A fim de evitar conflitos de versões entre as bibliotecas python instaladas, é recomendável a criação e utilização de um ambiente virtual com o módulo venv disponível para a linguagem.

É possível executar o programa em outro sistema Unix, como por exemplo o Ubuntu, utilizando os mesmos comandos em Bash. No sistema operacional Windows, é recomendável a utilização da plataforma Anaconda. Com ela é possível possível criar o ambiente virtual e realizar a instalação das bibliotecas

python através de sua IDE e executar o programa utilizando a Anaconda shell integrada na plataforma ou o próprio Windows PowerShell.

3.2 Aplicação do Seam Carving

O programa desenvolvido consiste de um programa principal chamado *sc.py* que segue o processo descrito no fluxograma simplificado apresentado na Figura 10.

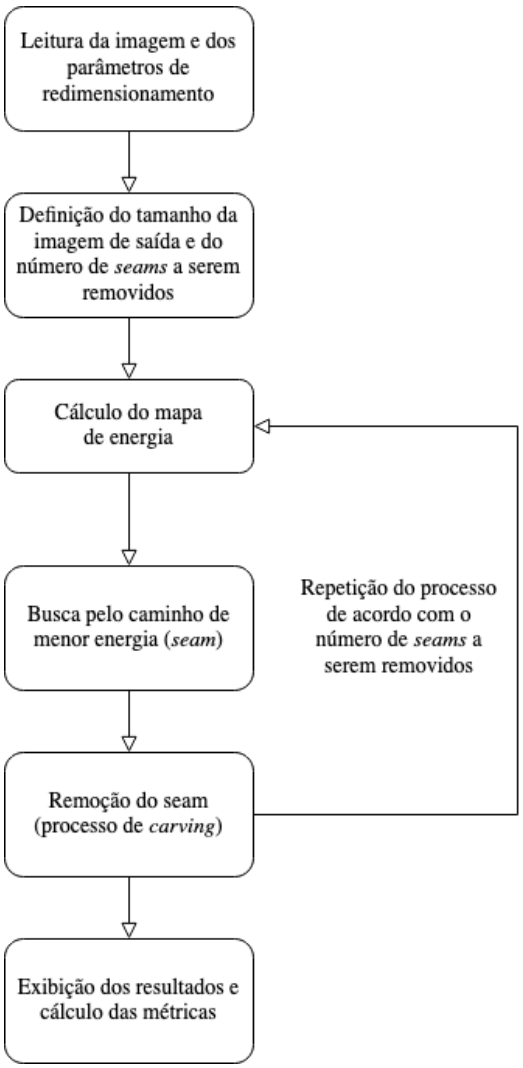


Figura 10 – Diagrama do programa principal Fonte: Autor

Durante a inicialização do programa principal, as classes *backward_energy.py* e *forward_energy.py* contendo os métodos para os diversos algoritmos de mapeamento de energia são instanciadas para serem usadas durante o processo de seam carving. Após isso, a imagem a ser redimensionada é lida e as variáveis de auxiliares e de saída são ajustadas conforme os parâmetros informados. O possíveis parâmetros de entrada estão apresentados na Figura 11.

O método de *Seam Carving* é realizado através de 3 funções no programa principal: *calculateEnergy*, *findSeam* e *seamCarving*. No diagrama apresentado na Figura 10, cada uma das funções corresponde respectivamente a um dos três blocos centrais cuja sequência é repetida de acordo com o número de seams a serem removidos.

Através do cálculo do mapa de energia M da imagem, é definida a relevância de cada pixel. Com

```
foo@bar src % python3 sc.py -h
usage: sc.py [-h] -in IN -scale SCALE -seam SEAM [-energy ENERGY] [-plot] [-metrics]

optional arguments:
  -h, --help      show this help message and exit
  -in IN          Path to input image
  -scale SCALE    Downsizing scale. e.g. 0.5
  -seam SEAM      Seam orientation (h = horizontal seam, v = vertical seam)
  -energy ENERGY  Energy mapping algorithm (s = Sobel, p = Prewitt, l =
                  Laplacian, r = Roberts, c = Canny, f = Forward energy)
  -plot           Plot result after resizing
  -metrics        Save metrics in a .csv file
```

Figura 11 – Parâmetros de entrada do programa principal Fonte: Autor

base nesse mapa, o caminho de menor energia (seam) é buscado em M e salvo em uma matriz auxiliar *backtrack* que contém as mesmas dimensões que a imagem original *img* e o mapa de energia M.

Durante o processo de remoção do seam (carving) a matriz *backtrack* é utilizada para percorrer o caminho de menor energia selecionado enquanto uma máscara booleana *mask* é construída para deletar os pixels do seam. Após a remoção dos pixels, a imagem é reajustada em suas dimensões para que a execução das três funções mencionadas possa ser realizada novamente.

3.2.1 Métodos utilizados

Como a remoção dos seams é feita baseada na energia do pixel, o sucesso do redimensionamento depende do algoritmo utilizado para traduzir a informação relevante na imagem para esses valores de energia em cada pixel. Nesse trabalho, diferentes algoritmos de detecção de borda são utilizados para a obtenção de diferentes resultados de energia.

Redimensionamento padrão

O primeiro tipo de redimensionamento a ser feito é um redimensionamento padrão sem qualquer mapeamento de energia. Essa implementação é a utilizada atualmente para rapidamente alterar o tamanho de uma imagem através da remoção de pixels de maneira igualmente distribuída através de toda a imagem. O Código 3.1 apresenta a função utilizada para o redimensionamento padrão.

Código 3.1 – Redimensionamento padrão

```
1  def standardResize(img, scale, seam_orientation):
2      """
3      Resize the image without considering its content
4      """
5      if seam_orientation == 'h':
6          width = img.shape[1]
7          height = int(img.shape[0] * scale)
8          dsize = (width, height)
9          std_resize_image = cv2.resize(img, dsize)
10     elif seam_orientation == 'v':
11         width = int(img.shape[1]*scale)
12         height = img.shape[0]
13         dsize = (width, height)
14         std_resize_image = cv2.resize(img, dsize)
15
16     return std_resize_image
```

Backward energy

O segundo tipo de redimensionamento usado é o de backward energy conforme descrito originalmente por Avidan e Shamir (AVIDAN; SHAMIR, 2007) com 5 algoritmos diferentes de detecção de borda, gerando 5 resultados diferentes. Os algoritmos utilizados foram o Sobel, Prewitt, Laplaciano, Roberts e Canny. Para os 4 primeiros algoritmos, a convolução das máscaras mencionadas na seção 2.3 foi realizada nos três canais da imagem redimensionada. Para a implementação do algoritmo Canny, a função *canny* disponibilizada pela biblioteca OpenCV foi utilizada.

Os métodos implementados para cada algoritmo de backward energy são apresentados a seguir:

Código 3.2 – Sobel

```
17 def sobel(self, img):
18     """
19     Sobel edge detection
20     """
21     kernel_sy = np.array([
22         [1.0, 2.0, 1.0],
23         [0.0, 0.0, 0.0],
24         [-1.0, -2.0, -1.0],
25     ])
26     kernel_sy = np.stack([kernel_sy] * 3, axis=2)
27     kernel_sx = np.array([
28         [1.0, 0.0, -1.0],
29         [2.0, 0.0, -2.0],
30         [1.0, 0.0, -1.0],
31     ])
32     kernel_sx = np.stack([kernel_sx] * 3, axis=2)
33
34     img = img.astype('float32')
35     sobel = np.absolute(convolve(img, kernel_sx)) + \
36         np.absolute(convolve(img, kernel_sy))
37
38     self.energy_map = sobel.sum(axis=2)
39
40     if img.shape[1] == self.input_image.shape[1]:
41         cv2.imwrite(SOBEL_EDGE_PATH, np.rot90(sobel, 3, (0, 1)))
42         cv2.imwrite(SOBEL_ENERGY_PATH, np.rot90(
43             self.energy_map, 3, (0, 1)))
44
45     return self.energy_map
```

Código 3.3 – Prewitt

```
46 def prewitt(self, img):
47     """
48     Prewitt edge detection
49     """
50     kernel_px = np.array([
51         [1.0, 0.0, -1.0],
52         [1.0, 0.0, -1.0],
53         [1.0, 0.0, -1.0],
54     ])
55     kernel_px = np.stack([kernel_px] * 3, axis=2)
56     kernel_py = np.array([
57         [1.0, 1.0, 1.0],
58         [0.0, 0.0, 0.0],
59         [-1.0, -1.0, -1.0],
60     ])
```

```
61      kernel_py = np.stack([kernel_py] * 3, axis=2)
62
63      img = img.astype('float32')
64      prewitt = np.absolute(convolve(img, kernel_px)) + \
65          np.absolute(convolve(img, kernel_py))
66
67      self.energy_map = prewitt.sum(axis=2)
68
69      if img.shape[1] == self.input_image.shape[1]:
70          cv2.imwrite(PREWITT_EDGE_PATH, np.rot90(prewitt, 3, (0, 1)))
71          cv2.imwrite(PREWITT_ENERGY_PATH, np.rot90(
72              self.energy_map, 3, (0, 1)))
73
74      return self.energy_map
```

Código 3.4 – Laplaciano

```
75  def laplacian(self, img):
76      """
77      Laplacian edge detection
78      """
79      kernel_l = np.array([
80          [0.0, 1.0, 0.0],
81          [1.0, -4.0, 1.0],
82          [0.0, 1.0, 0.0],
83      ])
84      kernel_l = np.stack([kernel_l] * 3, axis=2)
85      img = img.astype('float32')
86      laplacian = np.absolute(convolve(img, kernel_l))
87
88      self.energy_map = laplacian.sum(axis=2)
89
90      if img.shape[1] == self.input_image.shape[1]:
91          cv2.imwrite(LAPLACIAN_EDGE_PATH, np.rot90(laplacian, 3, (0, 1)))
92          cv2.imwrite(LAPLACIAN_ENERGY_PATH, np.rot90(
93              self.energy_map, 3, (0, 1)))
94
95      return self.energy_map
```

Código 3.5 – Roberts

```
96  def roberts(self, img):
97      """
98      Roberts edge detection
99      """
100     kernel_r1 = np.array([
101         [1.0, 0.0],
102         [0.0, -1.0],
103     ])
104     kernel_r1 = np.stack([kernel_r1] * 3, axis=2)
105     kernel_r2 = np.array([
106         [0.0, 1.0],
107         [-1.0, 0.0],
108     ])
109     kernel_r2 = np.stack([kernel_r2] * 3, axis=2)
110
111     img = img.astype('float32')
112     roberts = np.absolute(convolve(img, kernel_r1)) + \
```

```
113         np.absolute(convolve(img, kernel_r2))
114
115     self.energy_map = roberts.sum(axis=2)
116
117     if img.shape[1] == self.input_image.shape[1]:
118         cv2.imwrite(ROBERTS_EDGE_PATH, np.rot90(roberts, 3, (0, 1)))
119         cv2.imwrite(ROBERTS_ENERGY_PATH, np.rot90(
120             self.energy_map, 3, (0, 1)))
121
122     return self.energy_map
```

Código 3.6 – Canny

```
123 def canny(self, img):
124     self.energy_map = cv2.Canny(img, 100, 100, L2gradient=True)
125
126     if img.shape[1] == self.input_image.shape[1]:
127         cv2.imwrite(CANNY_ENERGY_PATH, np.rot90(
128             self.energy_map, 3, (0, 1)))
129
130     return self.energy_map
```

Forward energy

O terceiro e último redimensionamento usado é o de forward energy conforme descrito por Rubinstein (RUBINSTEIN; SHAMIR; AVIDAN, 2008).

O método de forward energy avalia o custo da energia adicionada na imagem após a criação de novas bordas pela remoção de um seam. Para o cálculo desse custo, é suficiente analisar somente uma pequena região próxima ao pixel a ser removido.

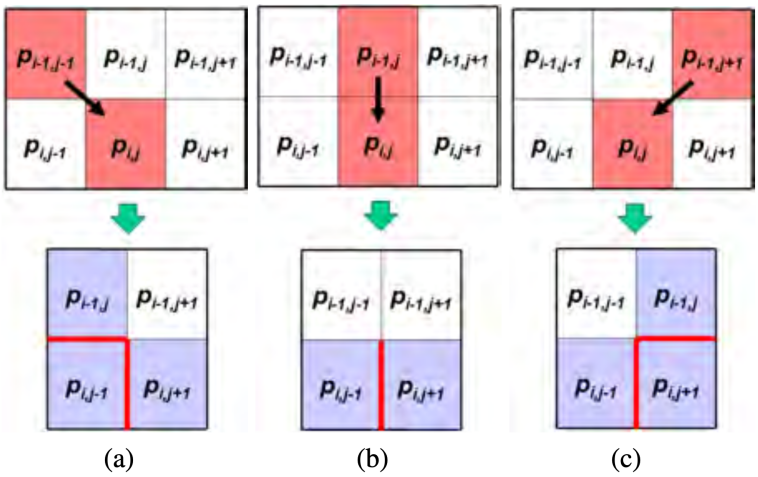


Figura 12 – Forward energy: possíveis direções para um seam e as novas bordas (em vermelho) criadas após a remoção de cada seam Fonte: (RUBINSTEIN; SHAMIR; AVIDAN, 2008)

O custo dessas novas bordas é medido através das diferenças entre os pixels que se tornam os novos vizinhos depois da remoção do seam. A Figura 12 apresenta as três possíveis direções de um seam, definindo então três custos a serem medidos: (a) o custo C_L pelo caminho à esquerda ($\boldsymbol{L}eft$) , (b) o custo C_U pelo caminho superior ($\boldsymbol{U}p$) e (c) o custo C_R pelo caminho à direita ($\boldsymbol{R}ight$). As equações matemáticas

para o cálculo de cada custo são apresentadas a seguir, onde I é a imagem:

$$C_L(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j-1)| \tag{3.1}$$

$$C_U(i,j) = |I(i,j+1) - I(i,j-1)| \tag{3.2}$$

$$C_R(i,j) = |I(i,j+1) - I(i,j-1)| + |I(i-1,j) - I(i,j+1)| \tag{3.3}$$

Os custos calculados são utilizados na matriz cumulativa M para o cálculo do seams através de programação dinâmica. Para seams verticais, $M(i,j)$ é atualizada com base no critério a seguir:

$$M(i,j) = min \begin{cases} M(i-1,j-1) + C_L(i,j) \\ M(i-1,j) + C_U(i,j) \\ M(i-1,j+1) + C_R(i,j) \end{cases} \tag{3.4}$$

O Código 3.7 apresenta o método utilizado para a criação do mapa de energia da imagem com base no método de forward energy.

Código 3.7 – Forward Energy

```
131 def fast_forward_energy(self, img):
132     h, w = img.shape[:2]
133     img = cv2.cvtColor(img.astype(np.uint8),
134                        cv2.COLOR_BGR2GRAY).astype(np.float64)
135
136     energy_map = np.zeros((h, w))
137     m = np.zeros((h, w))
138
139     U = np.roll(img, 1, axis=0)
140     L = np.roll(img, 1, axis=1)
141     R = np.roll(img, -1, axis=1)
142
143     cU = np.abs(R - L)
144     cL = np.abs(U - L) + cU
145     cR = np.abs(U - R) + cU
146
147     for i in range(1, h):
148         mU = m[i-1]
149         mL = np.roll(mU, 1)
150         mR = np.roll(mU, -1)
151
152         mULR = np.array([mU, mL, mR])
153         cULR = np.array([cU[i], cL[i], cR[i]])
154         mULR += cULR
155
156         argmins = np.argmin(mULR, axis=0)
157         m[i] = np.choose(argmins, mULR)
158         energy_map[i] = np.choose(argmins, cULR)
159
160     self.energy_map = energy_map
161
162     if img.shape[1] == self.input_image.shape[1]:
163         cv2.imwrite(FORWARD_ENERGY_PATH, np.rot90(
164             self.energy_map, 3, (0, 1)))
165
```

```
166     return self.energy_map
```

3.3 Métricas

Para definir a eficácia de cada método de mapeamento de energia utilizado no redimensionamento, uma comparação visual entre as imagens finais e os mapas de energia será realizada. Para suplementar essa análise, serão utilizados também o cálculo da energia média e entropia de Shannon das imagens originais, do resultado obtido com um redimensionamento padrão e do resultado obtido com o redimensionamento *Seam Carving* de cada algoritmo de mapeamento de energia.

3.3.1 Energia média

Conforme explicado na subseção 2.4.3, a energia média da imagem pode ser usada como parâmetro de avaliação de preservação de energia após o redimensionamento e é dada pela seguinte equação:

$$e_m = \frac{1}{|\mathbf{I}|} \sum_{p \in I} e(p) \tag{3.5}$$

Onde p são os pixels pertencentes a imagem $\mathbf{I}$ e $e(p)$ é a energia ou intensidade de cada pixel.

No programa desenvolvido, o cálculo da energia média é realizado através da função *mean* proporcionada pela biblioteca Numpy e tendo como parâmetro de entrada uma imagem RGB.

Aplicando um redimensionamento que leva em consideração o conteúdo de uma imagem, é esperado que a energia média após a operação aumente, visto que os pixels de maior energia que compõem elementos relevantes da imagem foram mantidos. Em uma operação de redimensionamento padrão, onde o conteúdo não é considerado, a energia média da imagem tende a se manter a mesma.

3.3.2 Entropia de Shannon

Outro parâmetro de avaliação utilizado foi a entropia de Shannon, definida como a aleatoriedade de informação contida nas variáveis de um sistema. Nesse caso, as variáveis avaliadas são os pixels da imagem com base no número de intensidades que cada uma pode assumir.

Matematicamente, a entropia de uma imagem $\mathbf{I}$ é definida como:

$$S(\mathbf{I}) = -\sum_{k}^{n} p_k \cdot log(p_k) \tag{3.6}$$

onde,

$$p_k = \frac{\text{Número de ocorrências com intensidade } k}{\text{Número de intensidades}} \tag{3.7}$$

No programa desenvolvido, o cálculo da entropia de Shannon é feito através da função *shannon_entropy* proporcionada pela biblioteca scikit-image e tendo como parâmetro de entrada uma imagem RGB. De forma similar a energia média, é esperado que a entropia da imagem aumente com o redimensionamento feito através do método de *Seam Carving*, pois as variáveis de informação relevante são mantidas. Em contrapartida, é esperado que a entropia se mantenha a mesma com um redimensionamento padrão.

4 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados obtidos com o programa desenvolvido ao longo do trabalho. Nele são apresentadas as imagens utilizadas para teste, suas versões redimensionadas e também os valores de Energia média (EM) e Entropia de Shannon calculados durante o processamento. Também será proposto o uso de uma nova métrica para a comparação das imagens. Essa nova métrica considera melhor o conteúdo de cada imagem por ser baseada em um mapeamento de características visuais.

Três imagens serão analisadas no decorrer do capítulo, as duas primeiras foram reduzidas verticalmente (remoção de *seams* horizontais) devido ao fato de possuírem objetos compostos em sua maioria por linhas horizontais. A terceira imagem foi reduzida horizontalmente (remoção de *seams* verticais) pelo motivo oposto das duas primeiras.

Como serão gerados sete resultados diferentes de redimensionamento, um com redimensionamento padrão e mais seis com os algoritmos de mapeamento de energia testados, apenas o redimensionamento padrão e os dois melhores resultados (um baseado na análise visual e outro na diferença de EM e entropia) serão apresentados. Os resultados restantes podem ser encontrados no Apêndice A, Apêndice B e Apêndice C.

4.1 Imagem de teste 1

A primeira imagem utilizada para o teste do programa é de autoria própria e foi escolhida por ser uma imagem rica em detalhes e com seções bem divididas e composta em sua maioria por objetos horizontais, favorecendo um redimensionamento vertical. A Figura 13 possui dimensões de 1024x768 pixels e energia média de 106.0055 e entropia de 7.8256.

Resultado com redimensionamento padrão

A Figura 14 apresenta a imagem da Figura 13 redimensionada verticalmente por um fator de 0.65, resultando na remoção de 269 linhas. A energia média resultante é de 105.8924 e entropia de 7.8300.

É possível ver que todos os aspectos originais da imagem foram mantidos, com exceção da forma. A proporção de céu, ilha e mar se manteve a mesma porém todos possuem agora um aspecto achatado para se adaptar à nova altura.

Resultado baseado em análise visual

A Figura 15 mostra o mapa de energia e a imagem redimensionada gerados pelo algoritmo Sobel. Dentre os demais, esse foi o algoritmo que apresentou melhores resultados visuais, o mapa de energia mostra que a ilha e o mar são seções da imagem que possuem alta energia, portanto são seções que o algoritmo busca ao máximo preservar. No resultado é possível ver que essas seções se mantiveram inalteradas com a maioria dos *seams* removidos do céu.

Resultado baseado nas métricas de energia e entropia

Analisando os resultados da Tabela 3 e da Tabela 4, o algoritmo Canny foi o que gerou o maior aumento em relação aos valores originais de energia média de 106.0055 e entropia de 7.8256. Portanto, conclui-se que esse foi o algoritmo que mais preservou as regiões de alta energia durante o redimensionamento. O resultado desse algoritmo é apresentado na Figura 16 junto com seu mapa de energia.

Figura 13 – Imagem de teste 1 - Dimensões de 1024x768 pixels, EM de 106.0055 e entropia de 7.8256. Fonte: Autor

Figura 14 – Imagem de teste 1 reduzida verticalmente com fator de 0.65 utilizando redimensionamento padrão. Agora com com dimensões de 1024x499 pixels, EM de 105.8924 e entropia de 7.8300. Fonte: Autor

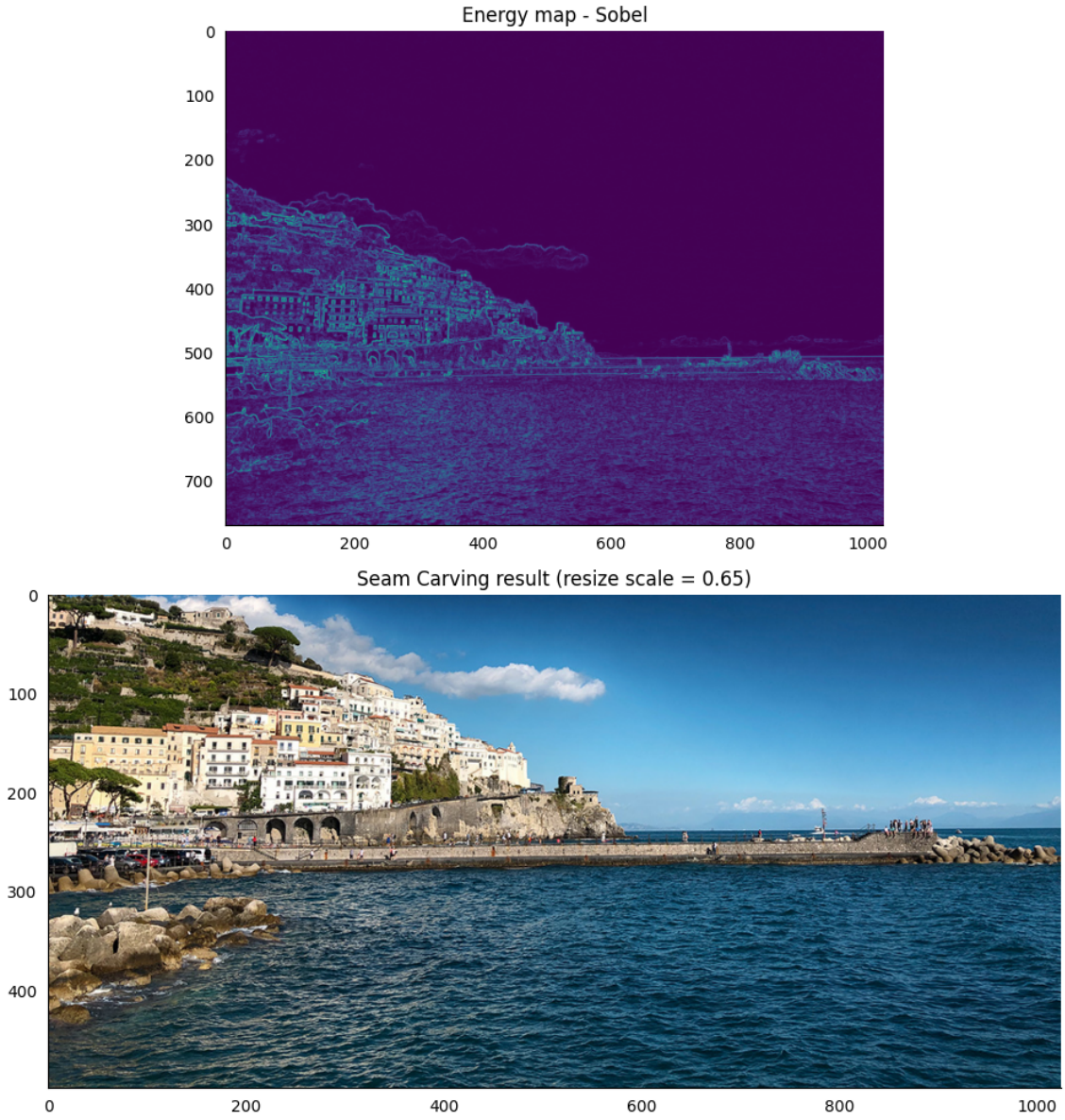


Figura 15 – Resultado do redimensionamento com o uso do Sobel para mapeamento de energia. Fonte: Autor

Por ser um algoritmo de detecção de borda com limiares discretos, o mapa de energia gerado pelo Canny é bastante agressivo. Não existe praticamente distinção alguma entre a ilha e o mar. Isso faz com que mais *seams* sejam removidos do céu em comparação com os outros algoritmos. O resultado obtido ainda é bom, mas distorce alguns dos objetos presentes no topo da ilha, pois estão mais próximos do céu e são removidos por serem considerados caminhos de menor energia com relação aos definidos na parte inferior da imagem.

4.2 Imagem de teste 2

A segunda imagem foi escolhida por ser comumente usada em trabalhos que tratam do método de *Seam Carving*. Assim como a primeira imagem, ela contém objetos horizontais e portanto foi reduzida verticalmente. A Figura 17 possui dimensões de 800x640, energia média de 56.9350 e entropia de 7.0908

Resultado com redimensionamento padrão

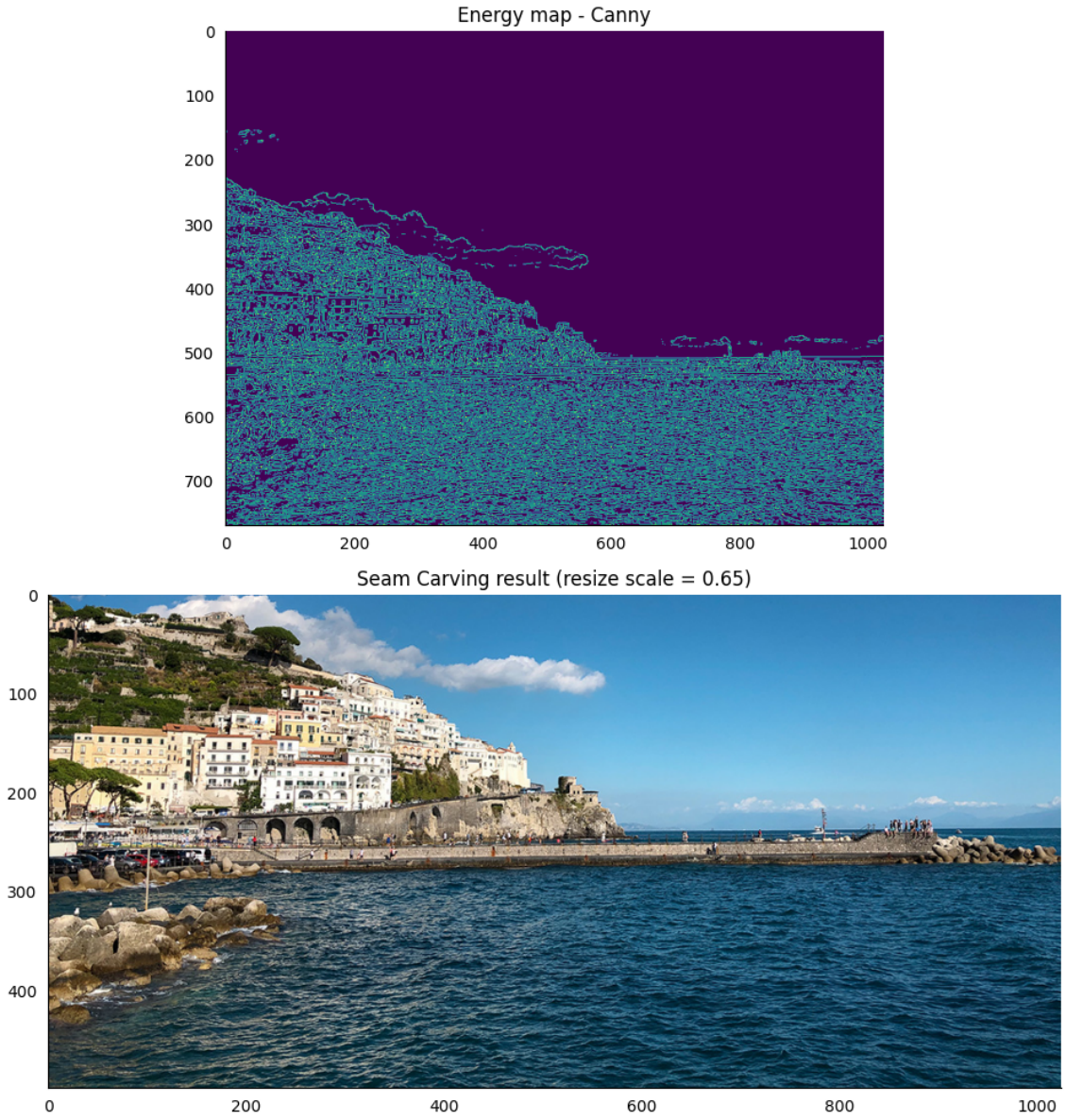


Figura 16 – Resultado do redimensionamento com o uso do Canny para mapeamento de energia. Fonte: Autor

A Figura 18 apresenta a imagem da Figura 17 redimensionada verticalmente por um fator de 0.65, removendo 224 linhas. O redimensionamento padrão alterou a energia média da imagem para 56.8632 e a entropia para 7.0818.

Nessa imagem, os objetos de relevância (veículos, pessoas e barracas) estão concentrados numa porção pequena e central da imagem. Com o redimensionamento padrão, essa seção foi fortemente afetada e distorcida.

Resultado baseado em análise visual

Dentre os seis algoritmos testados, os que apresentaram o melhor resultado visual foram o Prewitt e o Sobel. Ambos produziram resultados muito semelhantes porém o algoritmo Prewitt foi escolhido nesse critério por respeitar melhor a região de encontro do céu com a silhueta das montanhas e distorcendo menos as duas seções.

Figura 17 – Imagem de teste 2 - camping.jpg com dimensões de 800x640, EM de 56.9350 e entropia de 7.0908. Fonte: Autor

Figura 18 – Imagem de teste 2 reduzida verticalmente com fator de 0.65 utilizando redimensionamento padrão. Agora com com dimensões de 800x416, EM de 56.8632 e entropia de 7.0818. Fonte: Autor

Tabela 3 – Resultados de energia média para imagem amalfi.jpg reduzida verticalmente com fator de 0.65 utilizando o método de *Seam Carving*

Imagem: **amalfi.jpg** (1024x768) - 269 seams removidos

Mapeamento de energia	*EM Seam Carving*
Canny	111.4312
Prewitt	105.4105
Sobel	105.3924
Forward_energy	104.9483
Roberts	103.9749
Laplaciano	102.3068

Fonte: Autor

Tabela 4 – Resultados de entropia para imagem amalfi.jpg reduzida verticalmente com fator de 0.65 utilizando o método de *Seam Carving*.

Imagem: **amalfi.jpg** (1024x768) - 269 seams removidos

Mapeamento de energia	*Entropia Seam Carving*
Canny	7.8693
Sobel	7.8586
Prewitt	7.8585
Forward_energy	7.8541
Roberts	7.8484
Laplaciano	7.8338

Fonte: Autor

A Figura 19 apresenta o mapa de energia e o resultado gerado pelo algoritmo Prewitt.

Resultado baseado nas métricas de energia e entropia

Novamente, dentre todos os algoritmos, o Canny foi o que criou o resultado com maior diferença positiva de energia média e entropia quando comparado aos valores da imagem antes do redimensionamento, conforme apresentado na Tabela 5 e Tabela 6.

Tabela 5 – Resultados de energia média para imagem camping.jpg reduzida verticalmente com fator de 0.65.

Imagem: **camping.jpg** (800x640) - 224 seams removidos

Mapeamento de energia	*EM Seam Carving*
Canny	61.4703
Laplaciano	58.3795
Roberts	57.7850
Prewitt	57.1619
Sobel	57.1253
Forward_energy	56.8917

Fonte: Autor

Tabela 6 – Resultados de entropia para imagem camping.jpg reduzida verticalmente com fator de 0.65.

Imagem: **camping.jpg** (800x640) - 224 seams removidos

Mapeamento de energia	*Entropia Seam Carving*
Canny	7.2541
Laplaciano	7.1314
Sobel	7.1189
Prewitt	7.1183
Roberts	7.1155
Forward_energy	7.0993

Fonte: Autor

A imagem resultante apresentada na Figura 20 em um primeiro momento parece melhor visual-

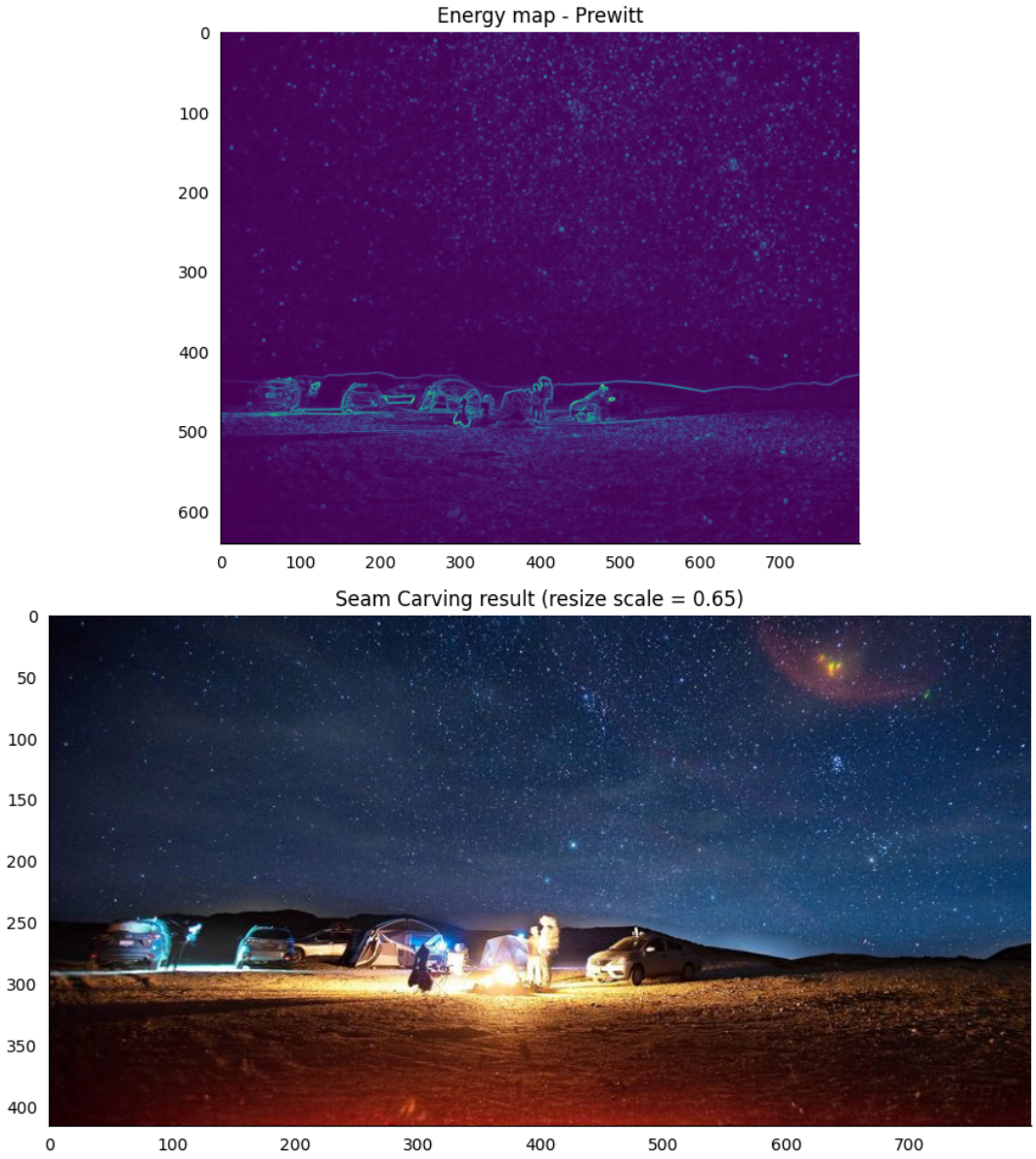


Figura 19 – Resultado do redimensionamento com o uso do Prewitt para mapeamento de energia. Fonte: Autor

mente quando comparada com o Prewitt pois a distorção da silhueta das montanhas foi praticamente nula. No entanto, ao olhar mais atentamente para o céu, é possível ver que a maioria das estrelas foi removida, inclusive a faixa de luz avermelhada no canto superior direito, que se tornou apenas um pequeno ruído nas linhas superiores do resultado.

4.3 Imagem de teste 3

Assim como a imagem anterior, a terceira imagem utilizada nesse capítulo foi selecionada por ser amplamente usada na internet para testes com *Seam Carving*. Ela também é utilizada como padrão na aplicação web desenvolvida por Aryan Naraghi para a demonstração do método (NARAGHI, 2020). A Figura 21 mostra a versão utilizada com dimensões de 1054x700, energia média de 135.4371 e entropia de 7.8492.

A imagem contém três objetos chaves: a torre, a pessoa e as nuvens. Os dois primeiros objetos

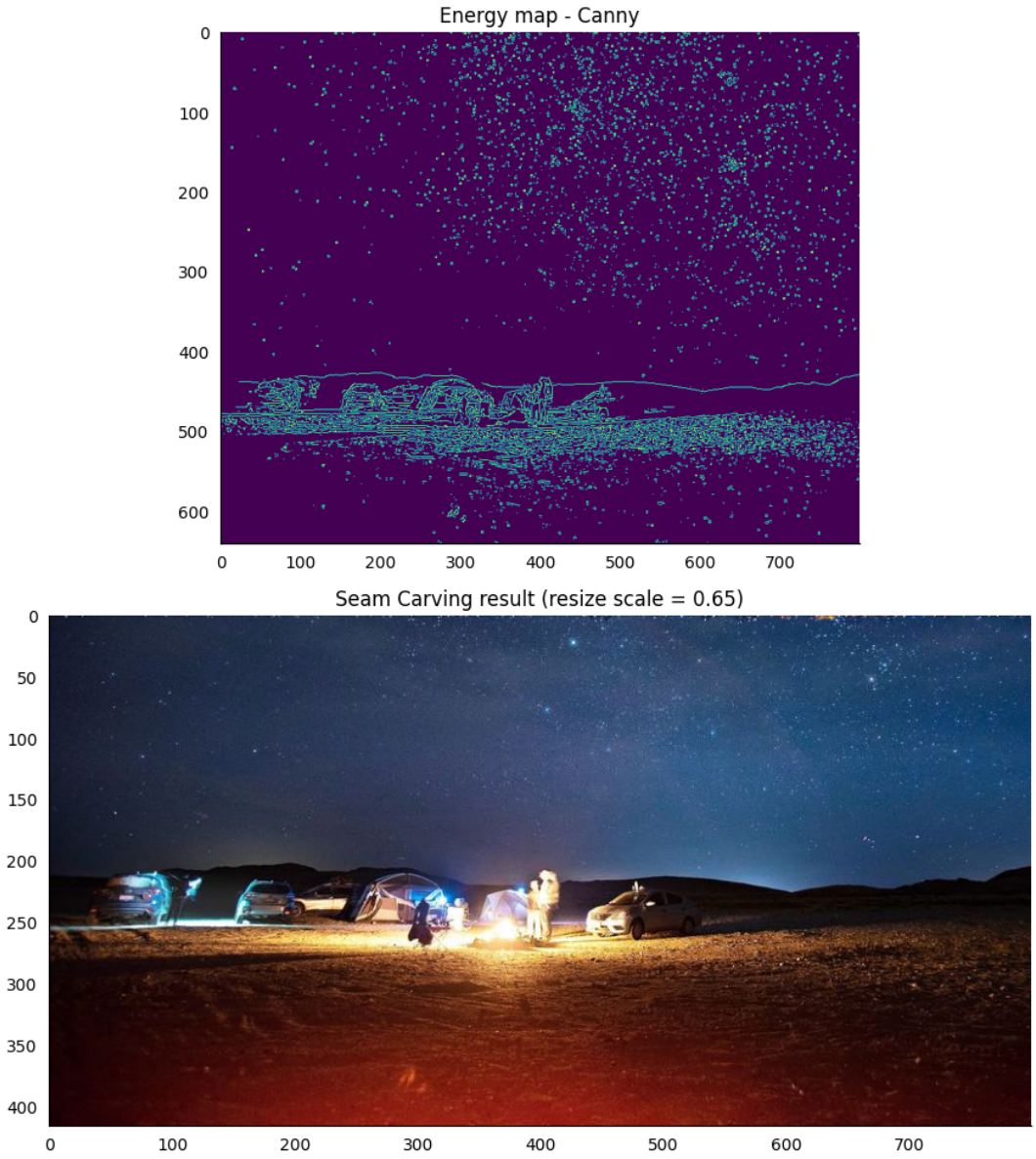


Figura 20 – Resultado do redimensionamento com o uso do Canny para mapeamento de energia. Fonte: Autor

são verticalizados, o que favorece uma redução de tamanho horizontal. Por esse motivo, a imagem foi redimensionada horizontalmente por um fator de 0.75.

Resultado com redimensionamento padrão

A Figura 22 mostra a imagem anterior redimensionada sem o uso do *Seam Carving*. Novamente temos todos os objetos presentes na imagem distribuídos proporcionalmente, porém com sua forma distorcida, como é possível observar principalmente na torre que foi comprimida horizontalmente.

Resultado baseado em análise visual

Assim como a figura anterior, o Sobel e o Prewitt se destacaram visualmente entre os demais resultados. O Prewitt foi escolhido por demostrar menos distorção nas nuvens do que o Sobel. A Figura 23 contém o plot gerado pelo algoritmo Prewitt.

Com esse algoritmo houve praticamente nenhuma alteração nos principais objetos mencionados. A

Figura 21 – Imagem de teste 3 - tower.jpg com dimensões de 1054x700, EM de 135.4371 e entropia de 7.8492. Fonte: Autor

Figura 22 – Imagem de teste 3 reduzida verticalmente com fator de 0.7 utilizando redimensionamento padrão. Agora com com dimensões de 790x700, EM de 135.4441 e entropia de 7.8538. Fonte: Autor

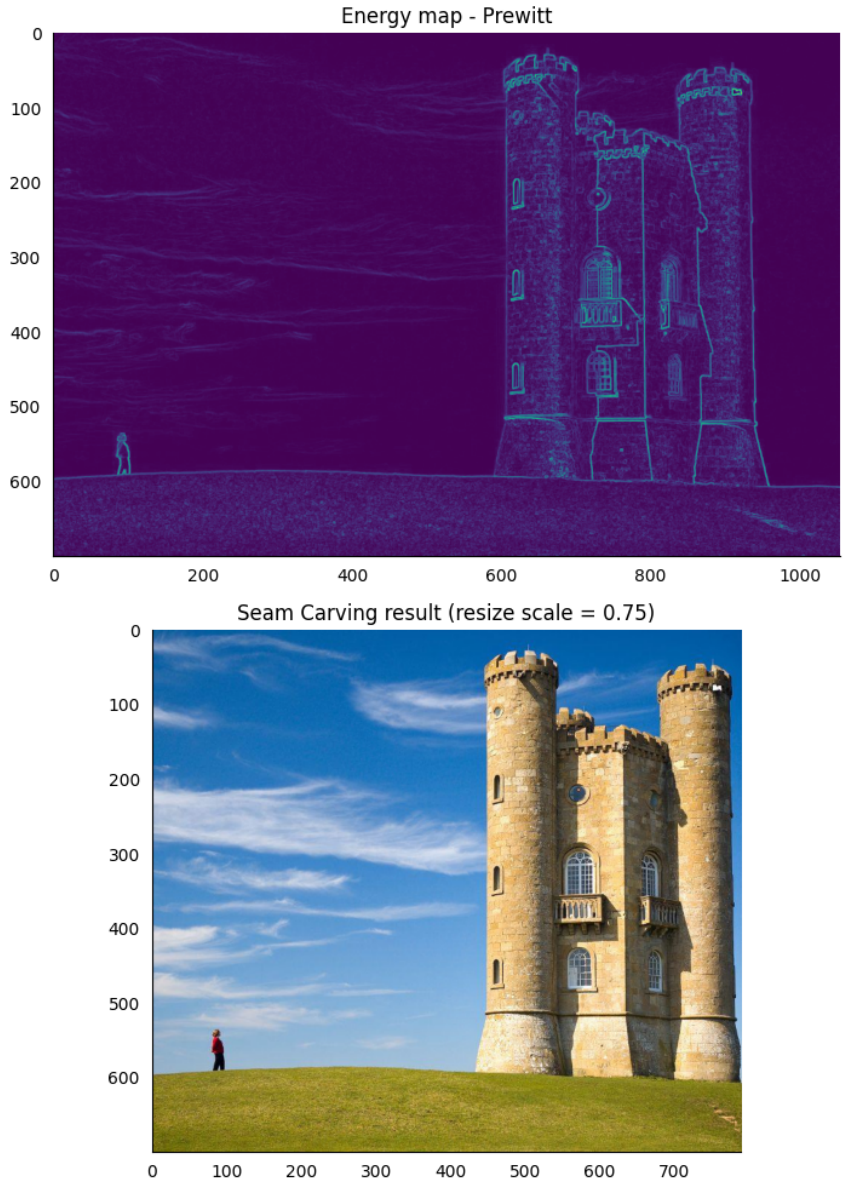


Figura 23 – Resultado do redimensionamento com o uso do Prewitt para mapeamento de energia. Fonte: Autor

área do céu à direita da torre foi completamente removida por conter menos energia. Como consequência, a mancha no gramado no canto inferior direito da imagem foi removida também em parte, mas a sua remoção tem pouco impacto na semântica da imagem.

Resultado baseado nas métricas de energia e entropia

Analisando a Tabela 7 é possível ver que o maior aumento de energia média ocorreu no resultado do redimensionamento com o Prewitt e o Sobel, estando de acordo com o resultado da análise visual. Porém em relação a entropia, a Tabela 8 tabela mostra que todos os valores aumentaram de forma similar com o redimensionamento, diferente das duas imagens anteriores onde pelo menos um algoritmo se destacou dos demais. A diferença de entropia entre eles é mínima e ocorre somente na terceira casa decimal, onde o algoritmo Laplaciano obteve maior valor.

Tabela 7 – Resultados de energia média para imagem tower.jpg reduzida horizontalmente com fator de 0.75.

Imagem: **tower.jpg** (1054x700) - 264 seams removidos	
Mapeamento de energia	*EM Seam Carving*
Prewitt	134.5863
Sobel	134.5854
Roberts	134.4033
Forward_energy	133.9841
Laplaciano	133.9415
Canny	133.1468

Fonte: Autor

Tabela 8 – Resultados de entropia para imagem tower.jpg reduzida horizontalmente com fator de 0.75.

Imagem: **tower.jpg** (1054x700) - 264 seams removidos	
Mapeamento de energia	*Entropia Seam Carving*
Laplaciano	7.8794
Forward_energy	7.8785
Canny	7.8760
Roberts	7.8755
Prewitt	7.8742
Sobel	7.8741

Fonte: Autor

4.4 Proposta de métrica SIFT para o Seam Carving

Durante a análise dos resultados gerados pelo programa, notou-se que as métricas de EM e entropia citadas na seção 3.3 não apresentaram desempenho conforme o esperado. Observa-se que o algoritmo de mapeamento de energia que gerou o resultado com maior EM e entropia não foi o que gerou melhor resultado quando analisado visualmente.

Isso ocorre porque ambas métricas não levam em consideração a semântica da imagem e o contexto dos objetos nela inseridos. Por causa disso, uma nova métrica que levasse em consideração esses tópicos foi acrescentada ao trabalho. A nova métrica escolhida é baseada no algoritmo de *Scale-invariant feature transform* (SIFT) que pode ser traduzido para Transformação de característica invariável em escala. Esse algoritmo de detecção de características é capaz de identificar objetos e regiões de relevância entre duas imagens iguais mesmos que elas possuam escala, orientação e iluminação distintas. O processo do algoritmo SIFT pode ser brevemente descrito através de 3 etapas: Detecção de extremos em espaço de escalas, Localização de pontos chaves e Atribuição de orientações (OPENCV, 2021).

Detecção de extremos em espaço de escalas

Para permitir a detecção de características em diferentes escalas, um espaço de escalas é inicialmente construído para o algoritmo. Neste espaço, funções gaussianas com diferentes valores de desvio padrão σ são convoluídos com a imagem em diferentes escalas, gerando uma representação dessa imagem em cada uma das escalas (SANTOS, 2016). A equação do filtro gaussiano é apresentado na equação 4.1:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}} \tag{4.1}$$

Para a detecção de pontos chaves no espaço de escalas construído, a técnica de Diferença de gaussianas (DoG) é utilizada. Com essa técnica, é criada uma pirâmide de gaussianas (Figura 24) e calculada a diferença entre imagens convoluídas com dois filtros gaussianos diferentes, um com σ e outro com $k\sigma$. A equação 4.2 apresenta o cálculo da DoG onde L é a imagem convoluida com o filtro gaussiano da equação anterior (SANTOS, 2016).

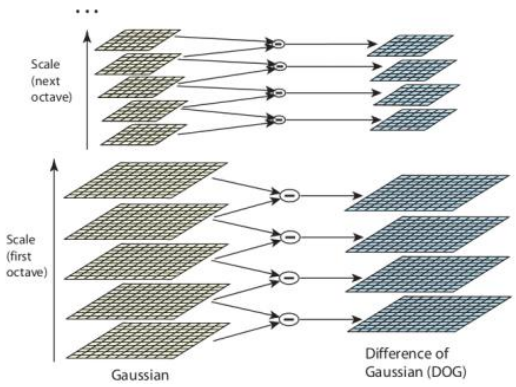


Figura 24 – Pirâmide de gaussianas com o cálculo de DoG. Fonte: (LOWE, 2004)

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{4.2}$$

Após o cálculo das DoG, a detecção de extremos é realizada através da comparação com os 8 pixels vizinhos da DoG atual e com os 9 pixels da escala anterior e da escala seguinte, conforme apresentado na Figura 25.

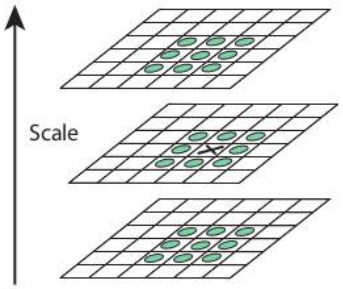


Figura 25 – Detecção de extremos em um espaço de escalas. Fonte: (LOWE, 2004)

Localização de pontos chaves

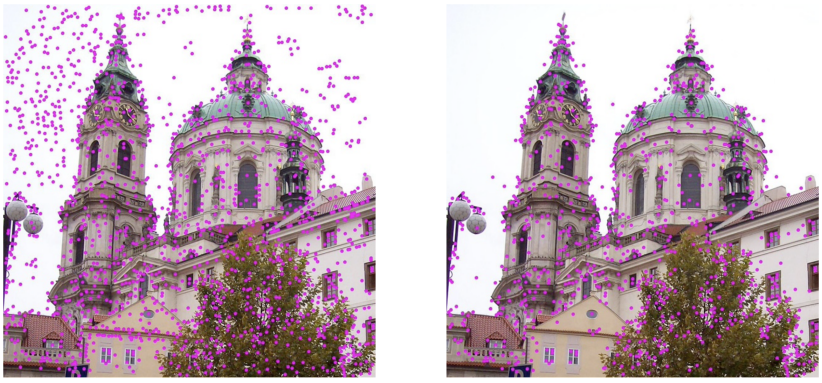
A detecção dos extremos realizada no passo anterior gera um grande número de possíveis pontos chaves na imagem que precisam ser refinados para melhores resultados. Esse processo de refinamento dos pontos de interesse é feita através da remoção de pontos de baixo contraste e de pontos localizados ao longo de bordas, ambos altamente suscetíveis a ruídos.

Para a eliminação de pontos de baixo contraste, uma expansão em série de Taylor (LOWE, 2004) ou o Laplaciano (SANTOS, 2016) do espaço de escalas $D(x, y, \sigma)$ podem ser utilizados junto com um limiar pré-determinado. Para remover pontos ao longo de bordas, é preciso encontrar as curvaturas ao longo delas através do cálculo dos autovalores para a matriz Hessiana de segunda ordem apresentada na equação 4.3.

$$H = \begin{bmatrix} D_{xx} & D_{xx} \\ D_{yx} & D_{yy} \end{bmatrix} \tag{4.3}$$

Onde D_{xx} representa $\frac{\delta^2 f}{\delta x \delta x}$, D_{xy} representa $\frac{\delta^2 f}{\delta x \delta y}$ e assim por diante (SANTOS, 2016). A Figura 26 apresenta a diferença entre os pontos de interesse em uma imagem antes e depois do refinamento.

Atribuição de orientações



(a) Resultado original (b) Resultado refinado

Figura 26 – Pontos de interesse calculados pela DoG antes (a) e depois (b) da remoção de pontos de baixo contraste e ao longo de bordas. Fonte: (SANTOS, 2016)

Nesse passo, com a finalidade de garantir a propriedade de invariância a rotação, uma orientação baseada em propriedades locais da imagem é atribuída a cada ponto de interesse. A abordagem consiste no uso da escala do ponto para selecionar a gaussiana, L, com escala mais próxima de forma a manter a propriedade de invariância de escala do algoritmo. Para cada amostra de imagem, $L(x, y)$, nessa mesma escala, a magnitude do gradiente, $m(x, y)$, e a orientação, $\theta(x, y)$ são calculados usando as equações 4.4 e 4.5, respectivamente.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{4.4}$$

$$\theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \tag{4.5}$$

Um histograma de orientação é então criado a partir das orientações de gradiente de pontos de amostra localizados ao redor do ponto de interesse. Esse histograma de orientação possui 36 *bins* para uma cobertura 360º. Picos no histograma de orientação correspondem a direções predominantes nos gradientes locais. O pico mais alto do histograma é detectado e então qualquer outro pico local que seja equivalente a pelo menos 80% do maior pico existente é também utilizado para a atribuição da orientação do ponto (LOWE, 2004).

Parametrização da métrica

A métrica foi utilizada levando em consideração dois fatores: o número de correspondências e a distribuição das correspondências através da imagem. O algoritmo de mapeamento de energia que gerar o resultado com maior número de correspondências e distribuição equilibrada delas na imagem é considerado o algoritmo com melhor desempenho.

4.4.1 Resultado SIFT para imagem de teste 1

A Tabela 9 apresenta os resultados de cada algoritmo para a correspondências de características na primeira imagem analisada.

Segundo a tabela, o algoritmo Sobel é o que possui maior número de correspondências. O resultado da métrica está de acordo com o obtido através da análise visual. A Figura 27 apresenta a distribuição das correspondências entre a imagem original e a imagem redimensionada.

Tabela 9 – Resultados do número de correspondências de características para imagem amalfi.jpg reduzida verticalmente com fator de 0.65 utilizando o método de *Seam Carving*.

Imagem: **amalfi.jpg** (1024x768) - 269 seams removidos

Mapeamento de energia	*Número de correspondências*
Sobel	4520
Prewitt	4512
Roberts	4483
Forward_energy	4457
Canny	4451
Laplaciano	4442
Sem mapeamento (redimensionamento padrão)	537

Fonte: Autor

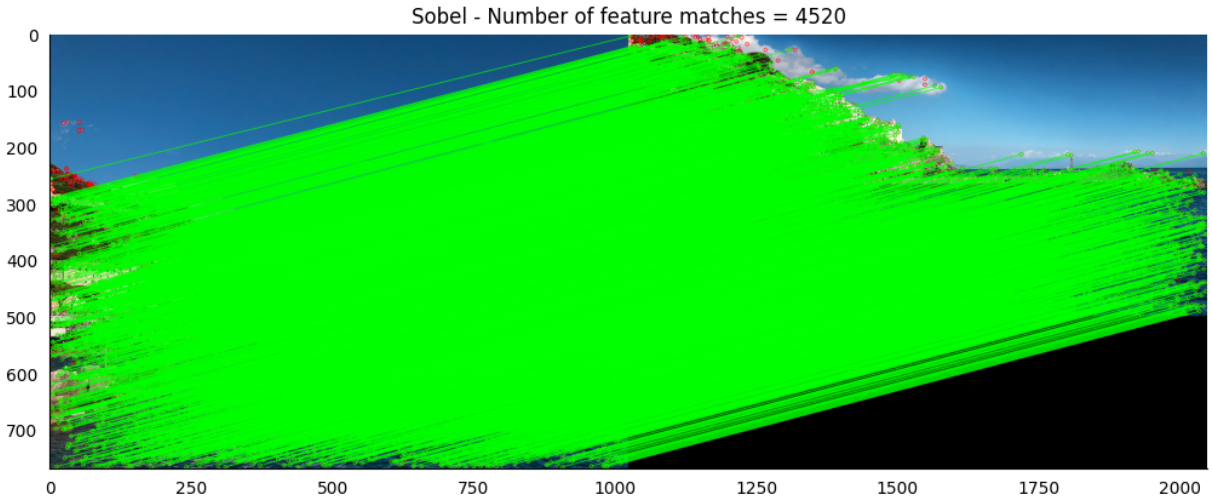


Figura 27 – Visualização da correspondência de características do redimensionamento com Sobel. À direita é exibida a imagem original e à esquerda é exibida a imagem redimensionada. Fonte: Autor

O algoritmo Canny, apontado pela EM e entropia como o de melhor desempenho, possui um número de correspondências menor porém não tão distantes do Sobel. Só que quando analisada a distribuição das correspondências apresentada na Figura 28, é possível notar que um número maior de características no topo da ilha foi perdido durante o redimensionamento.

4.4.2 Resultado SIFT para imagem de teste 2

Novamente, o uso da nova métrica se mostrou útil para uma interpretação mais contextualizada de cada resultado. Apesar da Tabela 10 apresentar o Prewitt em segundo lugar em relação as correspondências, o auxílio visual da métrica mostra que suas correspondências são melhor distribuídas do que o Canny que ficou em primeiro lugar. A distribuição visual das correspondências dos algoritmos Prewitt e Canny podem ser consultadas na Figura 29 e Figura 30 respectivamente.

4.4.3 Resultado SIFT para imagem de teste 3

Assim como as duas imagens anteriores, quando a nova métrica foi utilizada, o algoritmo Canny foi o que se destacou por ser aquele com o maior número de correspondências. Os valores encontrados são apresentados na Tabela 11.

O algoritmo Prewitt escolhido como melhor pela análise visual ficou em quarto lugar se tratando do número absoluto de correspondências. Porém, a diferença entre o Sobel e o Forward Energy que ficaram logo à frente não foi tão grande.

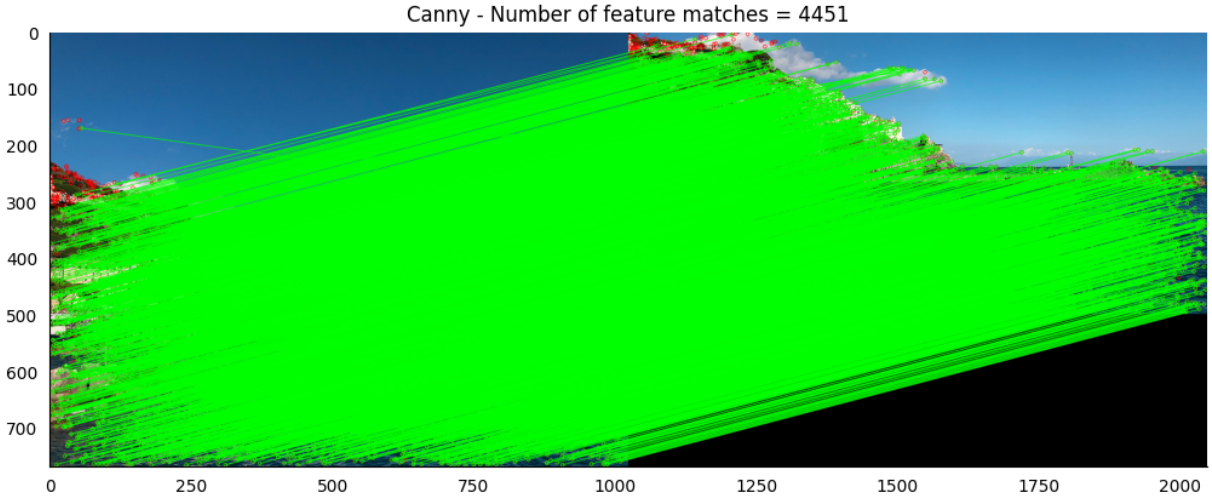


Figura 28 – Visualização da correspondência de características do redimensionamento com Canny. À direita é exibida a imagem original e à esquerda é exibida a imagem redimensionada. Fonte: Autor

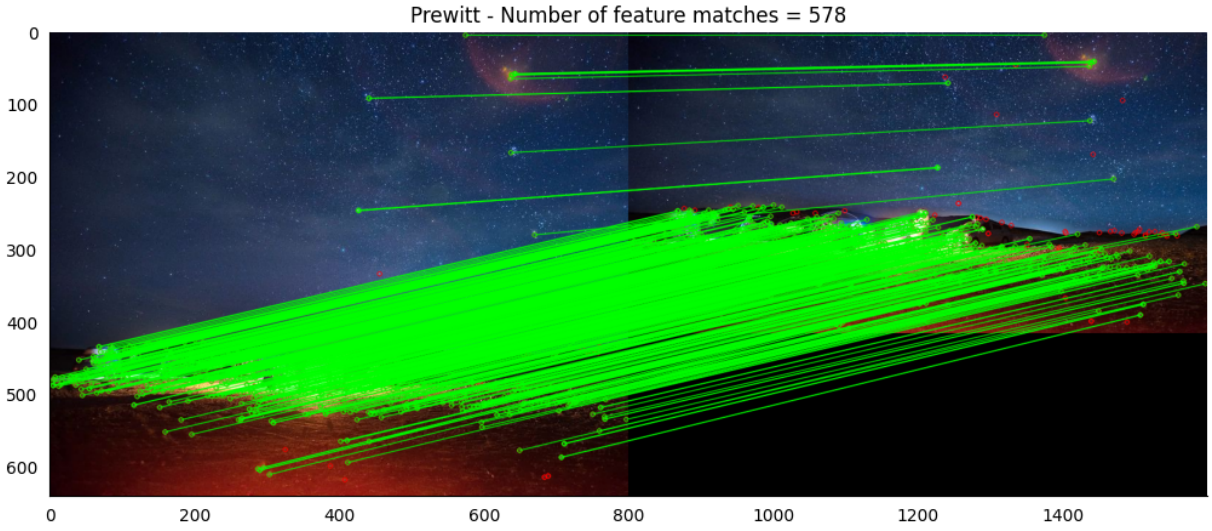


Figura 29 – Visualização da correspondência de características do redimensionamento com Prewitt. À direita é exibida a imagem original e à esquerda é exibida a imagem redimensionada. Fonte: Autor

Tabela 10 – Resultados do número de correspondências de características para imagem camping.jpg reduzida verticalmente com fator de 0.65 utilizando o método de *Seam Carving*.

Imagem: **camping.jpg** (800x640) - 224 seams removidos

Mapeamento de energia	Número de correspondências
Canny	609
Prewitt	578
Sobel	573
Forward_energy	564
Roberts	564
Laplaciano	514
Sem mapeamento (redimensionamento padrão)	109

Fonte: Autor

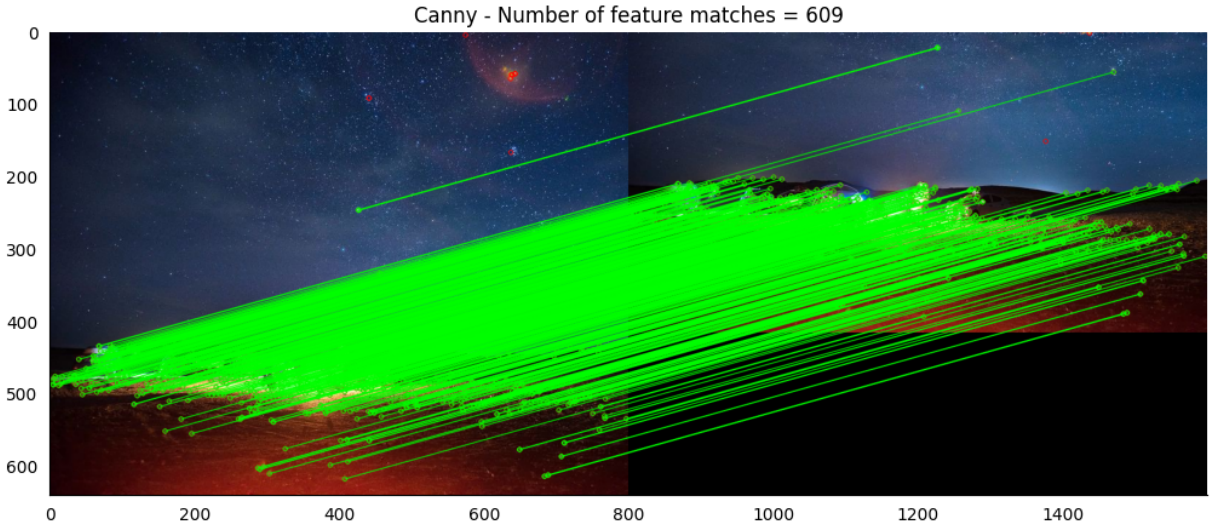


Figura 30 – Visualização da correspondência de características do redimensionamento com Canny. À direita é exibida a imagem original e à esquerda é exibida a imagem redimensionada. Fonte: Autor

Tabela 11 – Resultados do número de correspondências de características para imagem tower.jpg reduzida verticalmente com fator de 0.65 utilizando o método de *Seam Carving*.

Imagem: **tower.jpg** (1054x700) - 264 seams removidos

Mapeamento de energia	Número de correspondências
Canny	1270
Forward_energy	1226
Sobel	1210
Prewitt	1209
Roberts	1192
Laplaciano	1191
Sem mapeamento (redimensionamento padrão)	368

Fonte: Autor

Se comparada a distribuição visual dessas correspondências entre o algoritmo Canny (Figura 31) e o algoritmo Prewitt (Figura 32), é possível ver que a correspondência do Canny não está bem distribuída, apresentando uma grande perda de características na região do céu com nuvens, no gramado e na pessoa à esquerda da torre.

O resultado visual do Prewitt não apresentou as mesmas perdas que o Canny. Houve algumas perdas de características no gramado abaixo da torre, mas a distribuição das correspondências no geral teve uma melhor cobertura na imagem.

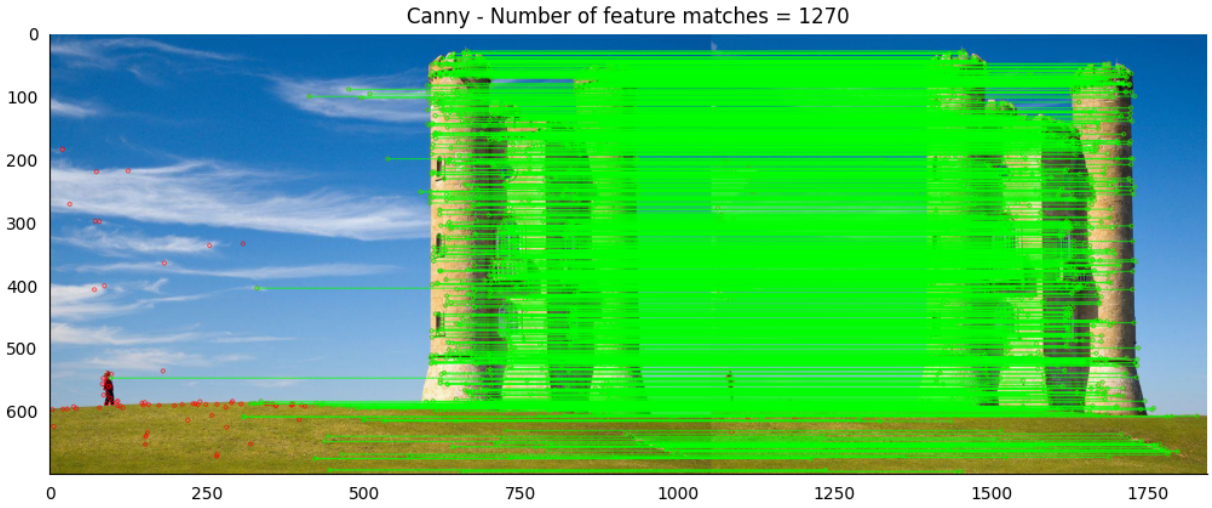


Figura 31 – Visualização da correspondência de características do redimensionamento com Canny. À direita é exibida a imagem original e à esquerda é exibida a imagem redimensionada. Fonte: Autor

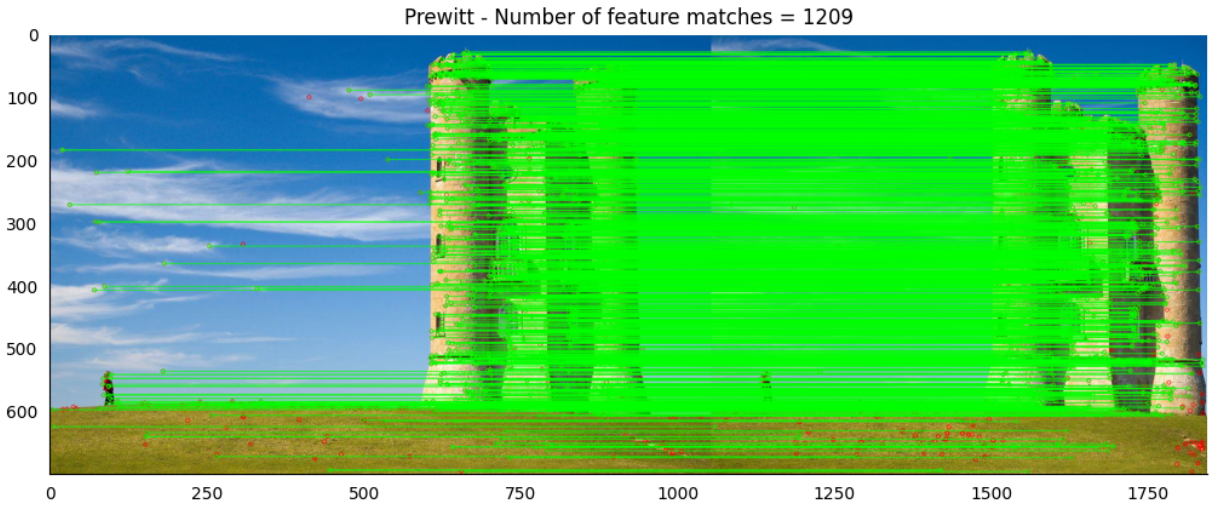


Figura 32 – Visualização da correspondência de características do redimensionamento com Prewitt. À direita é exibida a imagem original e à esquerda é exibida a imagem redimensionada. Fonte: Autor

4.5 Comparação com trabalhos relacionados

Durante o desenvolvimento deste trabalho, alguns comportamentos comuns a trabalhos relacionados foram notados. Em seu artigo original do método em 2007, Avidan e Shamir compartilharam que o método de *Seam Carving* possui dois grandes fatores limitantes: A quantidade de conteúdo na imagem e a disposição desse conteúdo na imagem (AVIDAN; SHAMIR, 2007). O primeiro fator limita o método quando a imagem é muito condensada em relação ao conteúdo, de forma que a mesma não contem áreas de menor relevância. Em uma imagem desse tipo, qualquer forma de redimensionamento que leva em consideração o conteúdo não será bem sucedido. Um bom exemplo é a imagem de teste 1 apresentada no início do capítulo. Com exceção da região do céu, o resto da imagem contém muita informação de forma que distorções no conteúdo começam a aparecer uma vez que todos os *seams* possíveis do céu já foram removidos. O segundo fator ocorre em imagens onde o conteúdo é distribuído de forma que previne que *seams* a serem removidos desviem de partes importantes. Um exemplo disso neste trabalho é a parte

superior da imagem de teste 2. A disposição das estrelas na imagem torna difícil a detecção e remoção de *seams* que não passem por elas.

Outro comportamento comum foi informado por Rubinstein em seu trabalho junto com Avidam e Shamir. Rubinstein percebe que o uso do algoritmo de *forward energy* é capaz de melhor manter a estrutura geral da imagem durante o redimensionamento, porém afetando o conteúdo. E que muitas vezes o uso de algoritmo de primeira derivada como o Sobel eram necessários para preservar o conteúdo (RUBINSTEIN; SHAMIR; AVIDAN, 2008). Neste trabalho o mesmo comportamento foi observado onde o algoritmo de *forward energy* não performou tão bem em relação a preservação de conteúdo quanto algoritmos de primeira derivada nos casos analisados.

5 CONCLUSÕES

Com a análise dos resultados apresentados no capítulo anterior, conclui-se que a performance de cada algoritmo depende da imagem de entrada. Porém, foi possível notar que algoritmos de primeira derivada como o Sobel e o Prewitt proporcionaram resultados melhores de maneira geral para as imagens utilizadas.

Os algoritmos de segunda derivada ou combinativos como o Laplaciano e Canny apresentam uma detecção de borda mais agressiva que acaba gerando mudanças perceptíveis demais nos objetos das imagens. Os resultados desses algoritmos mais agressivos podem ser melhorados através do ajuste de parâmetros e o pré-processamento das imagens redimensionadas em trabalhos futuros, o que exige um esforço maior de processamento e aumento de complexidade no algoritmo.

Pôde-se concluir também que as métricas de energia média e entropia de Shannon são altamente variáveis e não apresentam padrão aparente pois não levam em consideração todo o contexto da imagem e dos objetos inseridos nela. Devido ao fato da imagem ser uma entidade cujo objetivo depende muito do seu conceito e interpretação de quem observa, é preciso que esses fatores sejam levados em consideração quando uma alteração é feita. A métrica de correspondência de características (*Feature Matching*) se mostrou mais confiável que as anteriores, porém somente quando a distribuição das correspondências na imagem é levada em consideração junto com o número total encontrado. Para trabalhos futuros, traduzir a distribuição visual das correspondências em uma métrica escalar seria ideal pois iria permitir que a mesma fosse combinada junto com o número total de correspondências a fim de se obter um valor escalar final que definisse o nível de sucesso do redimensionamento.

Outro fator que se mostrou relevante durante a implementação do método foi a necessidade de processamento. Sem a utilização do Numba para a compilação das funções de *carving*, imagens com dimensões hoje em dia consideradas pequenas como 1024x768 levaram em torno de 6 minutos para serem redimensionadas. Com a compilação das funções, o tempo de processamento da mesma imagem caiu para em torno de 36 segundos. Portanto, o estudo e melhora do algoritmo pode ser realizado em linguagens de alto nível, porém para um melhor processamento é ideal que as funções de *carving* sejam ao menos compiladas previamente. A implementação do algoritmo em uma linguagem de nível mais baixo como C++ forneceria melhor desempenho, mas em contrapartida aumentaria a complexabilidade do código.

Por fim, como cada imagem tem um algoritmo ótimo diferente, um programa mais eficiente para a implementação do *Seam Carving*, ao invés de utilizar um único algoritmo de mapeamento de energia, deveria ser capaz selecionar um automaticamente baseado na imagem de entrada. Em um primeiro momento isso pode ser feito com através da realização de diferentes redimensionamentos simultâneos com a seleção do melhor no final do processo, mas o ideal é que isso seja feito sem a necessidade de realizar todo o redimensionamento completo, somente com base nos mapas de energia da imagem ou de outra característica auxiliar.

REFERÊNCIAS

AVIDAN, S.; SHAMIR, A. *Seam Carving for Content-Aware Image Resizing*. [S.l.: s.n.], 2007. Citado 4 vezes nas páginas 15, 24, 30 e 51.

BERRICHE, L.; AL-MUTAIRY, A. *Seam carving-based Arabic handwritten sub-word segmentation*. [S.l.: s.n.], 2020. Citado na página 15.

BURGER, W.; BURGE, M. *Digital Image Processing: An Algorithmic Introduction Using Java*. [S.l.]: Springer-Verlag London, 2016. Citado 5 vezes nas páginas 17, 18, 19, 21 e 24.

DALDALI, M.; SOUHAR, A. *Handwritten Arabic Documents Segmentation into Text Lines using Seam Carving*. [S.l.: s.n.], 2018. Citado 2 vezes nas páginas 15 e 26.

DONG, W. et al. *Optimized Image Resizing Using Seam Carving and Scaling*. [S.l.: s.n.], 2009. Citado 2 vezes nas páginas 15 e 26.

EMADULDEEN, M.; HASSAN, R. *Image Seam Carving Based on Content Aware Resizing by Gradient Method*. [S.l.: s.n.], 2016. Citado na página 26.

GONZALEZ, R. C.; WOODS, R. E. *Processamento de Imagens Digitais*. 3. ed. [S.l.]: Pearson Education do Brasil, 2010. Citado 2 vezes nas páginas 20 e 22.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 4. ed. [S.l.]: Pearson Education Limited, 2018. Citado 2 vezes nas páginas 18 e 19.

KARANTH, K. *Implementing Seam Carving with Python*. [S.l.], 2018. Disponível em: <https://karthikkaranth.me/blog/implementing-seam-carving-with-python/>. Acesso em: 13 agosto 2020. Citado na página 25.

LOWE, D. *Distinctive Image Featuresfrom Scale-Invariant Keypoints*. [S.l.: s.n.], 2004. Citado 2 vezes nas páginas 46 e 47.

NARAGHI, A. *Seam Carving Demo*. [S.l.], 2020. Disponível em: <https://www.aryan.app/seam-carving/>. Acesso em: 27 dezembro 2020. Citado na página 41.

NUMBA. *Numba*. [S.l.], 2020. Disponível em: <https://numba.pydata.org/>. Acesso em: 04 outubro 2020. Citado na página 27.

NUMPY. *NumPy*. [S.l.], 2020. Disponível em: <https://numpy.org/>. Acesso em: 04 outubro 2020. Citado na página 27.

OPENCV. *OpenCV*. [S.l.], 2020. Disponível em: <https://opencv.org/>. Acesso em: 04 outubro 2020. Citado na página 27.

OPENCV. *Introduction to SIFT (Scale-Invariant Feature Transform)*. [S.l.], 2021. Disponível em: <https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html>. Acesso em: 10 janeiro 2021. Citado na página 45.

PSF. *Python 3.8.6*. [S.l.], 2020. Disponível em: <https://www.python.org/downloads/release/python-386/>. Acesso em: 04 outubro 2020. Citado na página 27.

QINGFANG, L. et al. *Contour-Maintaining-Based Image Adaption for an Efficient Ambulance Service in Intelligent Transportation Systems*. [S.l.: s.n.], 2020. Citado 2 vezes nas páginas 15 e 26.

RUBINSTEIN, M.; SHAMIR, A.; AVIDAN, S. *Improved Seam Carving for Video Retargeting*. [S.l.: s.n.], 2008. Citado 4 vezes nas páginas 15, 26, 32 e 52.

SANTOS, R. *Sistema de autenticação por biometria acelerado por GPU*. [S.l.: s.n.], 2016. Citado 3 vezes nas páginas 45, 46 e 47.

SHI, M. et al. *Optimal Bi-directional Seam Carving for Content-Aware Image Resizing*. [S.l.: s.n.], 2010. Citado na página 26.

SZELISKI, R. *Computer Vision: Algorithms and Applications*. [S.l.]: Springer-Verlag London, 2011. Citado na página 18.

WANG, Q.; YUAN, Y. *High quality image resizing*. [S.l.: s.n.], 2013. Citado na página 26.

Apêndices

APÊNDICE A – RESULTADOS PARA A IMAGEM DE TESTE 1

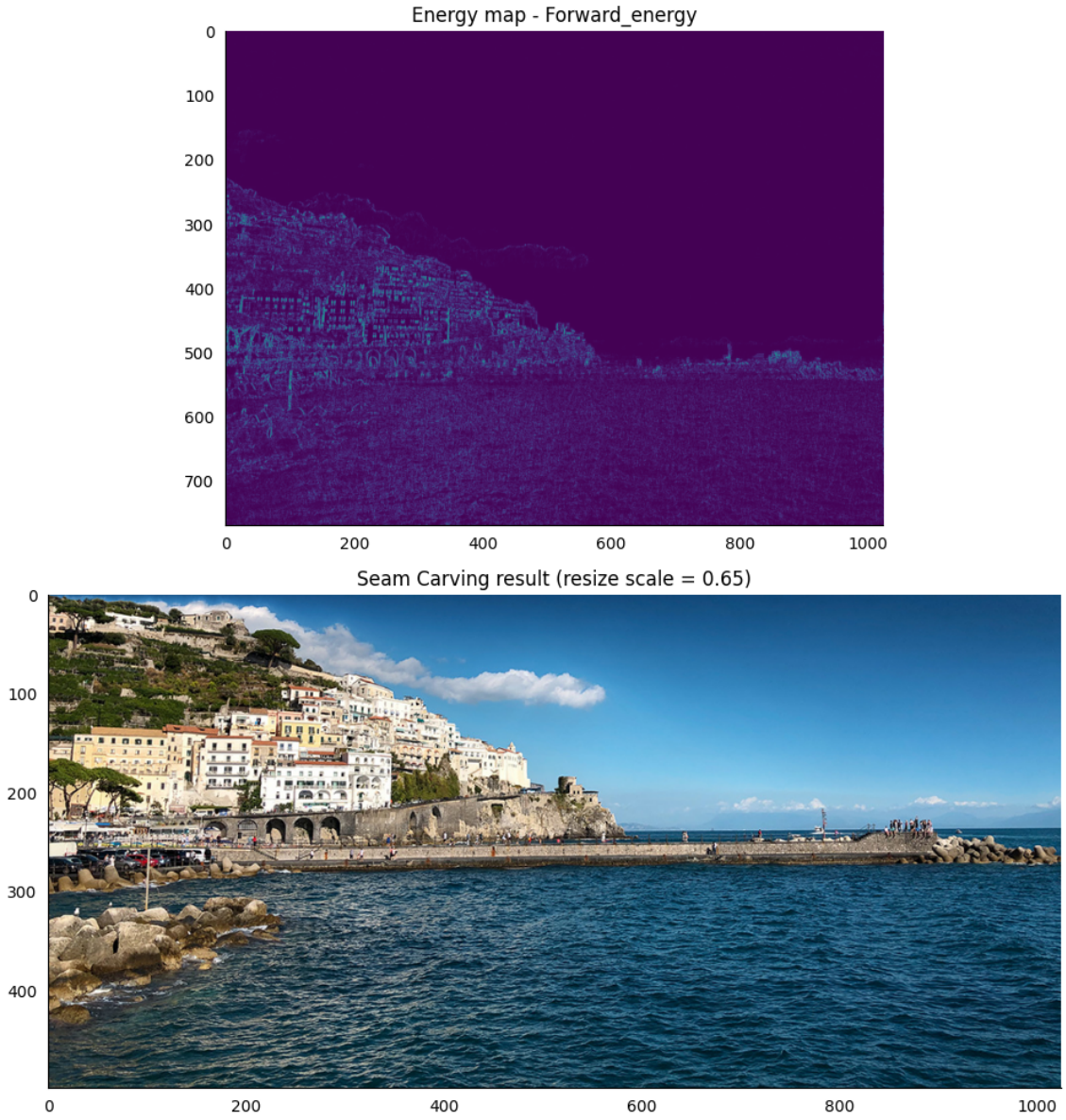


Figura 33 – Resultado do redimensionamento com o uso do Forward Energy para mapeamento de energia. Fonte: Autor

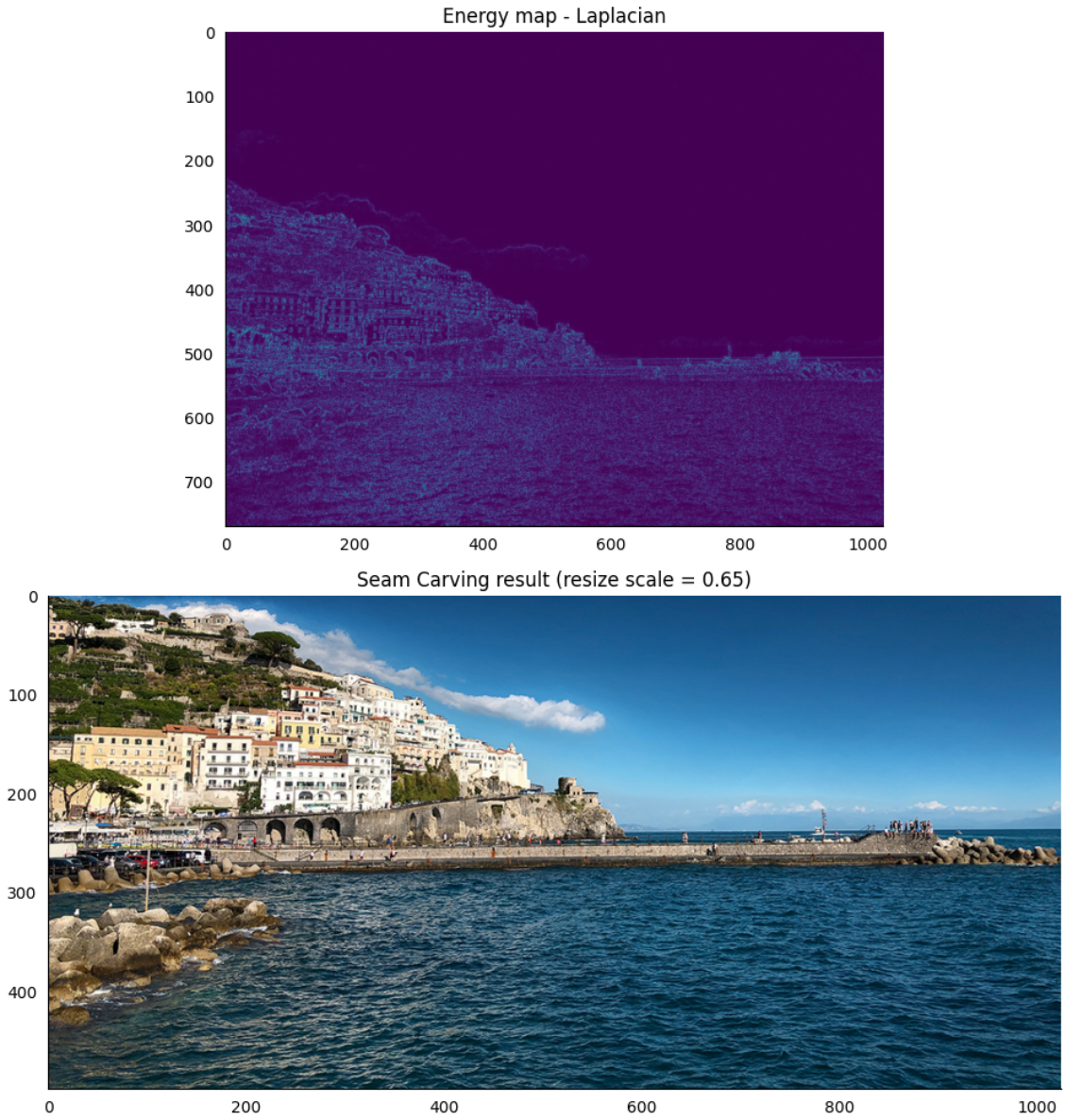


Figura 34 – Resultado do redimensionamento com o uso do Laplaciano para mapeamento de energia. Fonte: Autor

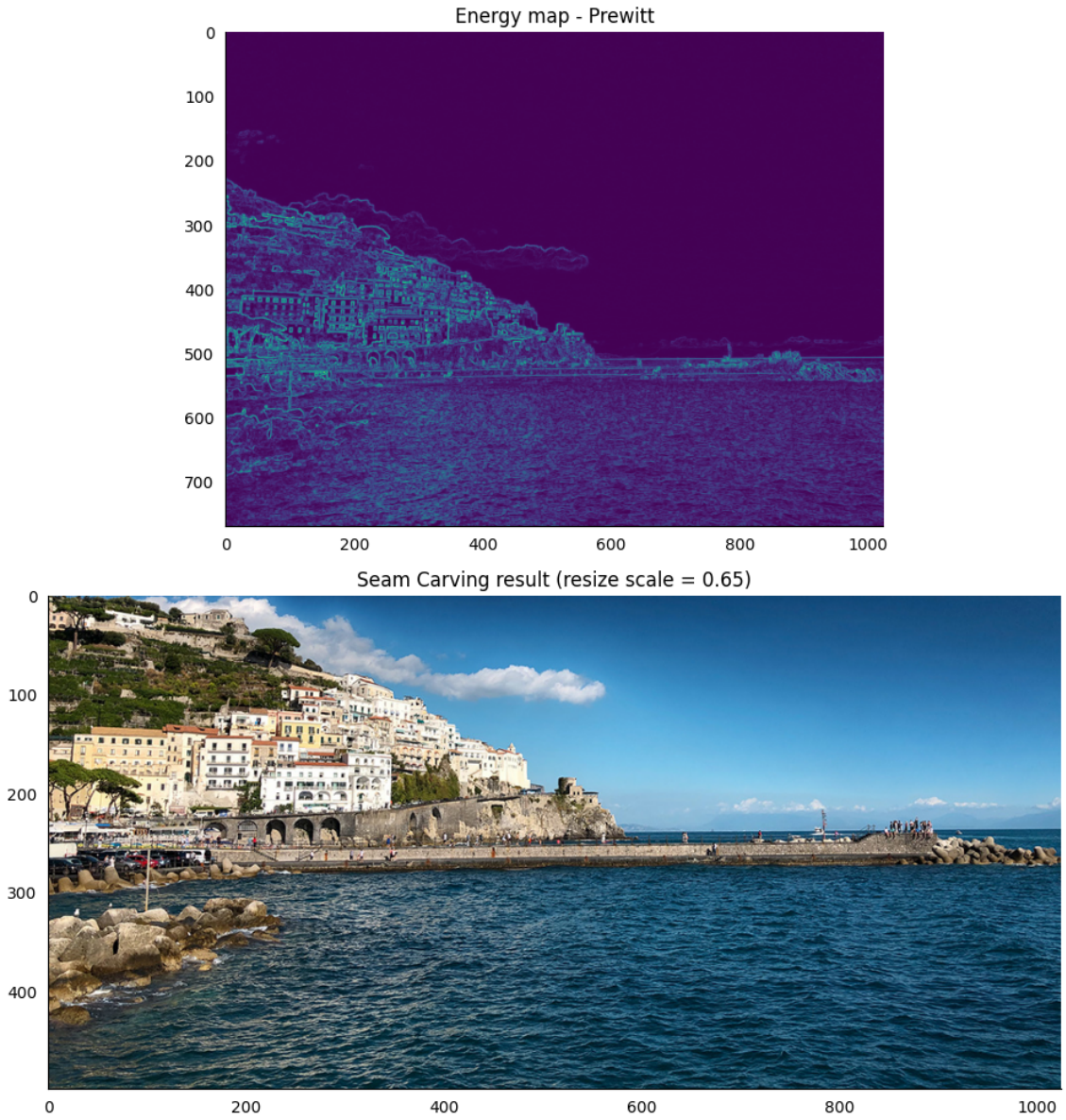


Figura 35 – Resultado do redimensionamento com o uso do Prewitt para mapeamento de energia. Fonte: Autor

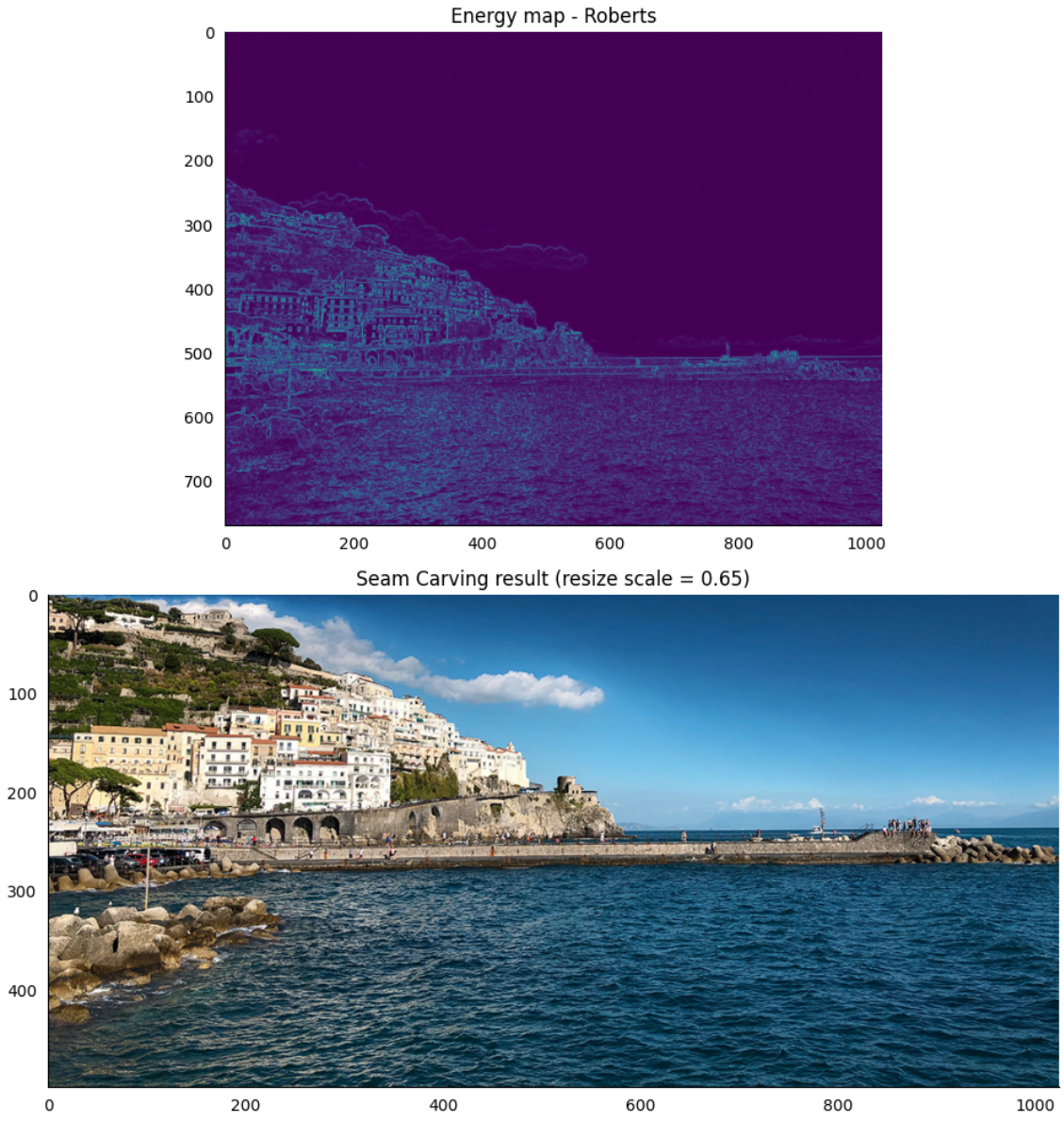


Figura 36 – Resultado do redimensionamento com o uso do Roberts para mapeamento de energia. Fonte: Autor

APÊNDICE B – RESULTADOS PARA A IMAGEM DE TESTE 2

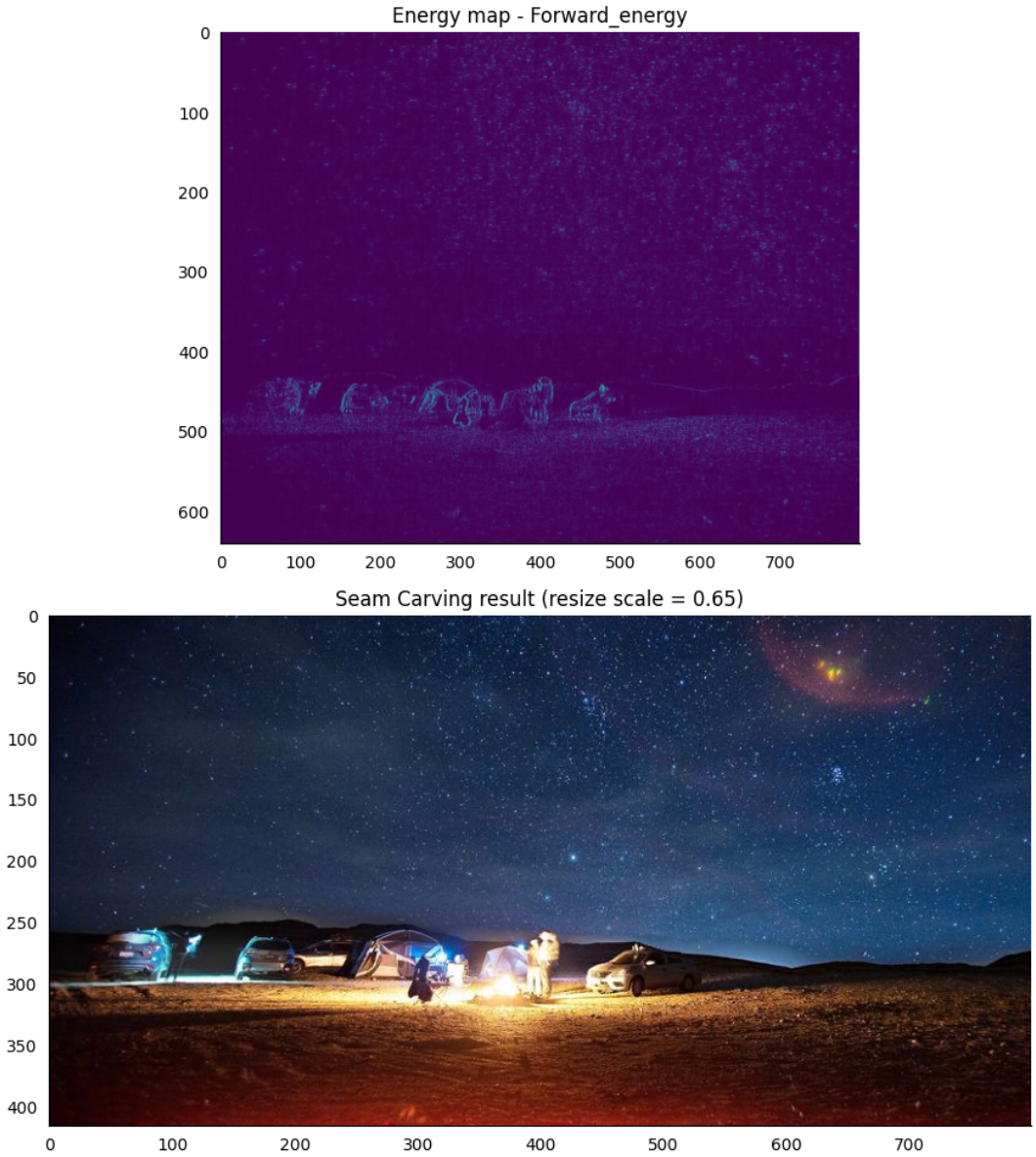


Figura 37 – Resultado do redimensionamento com o uso do Forward Energy para mapeamento de energia. Fonte: Autor

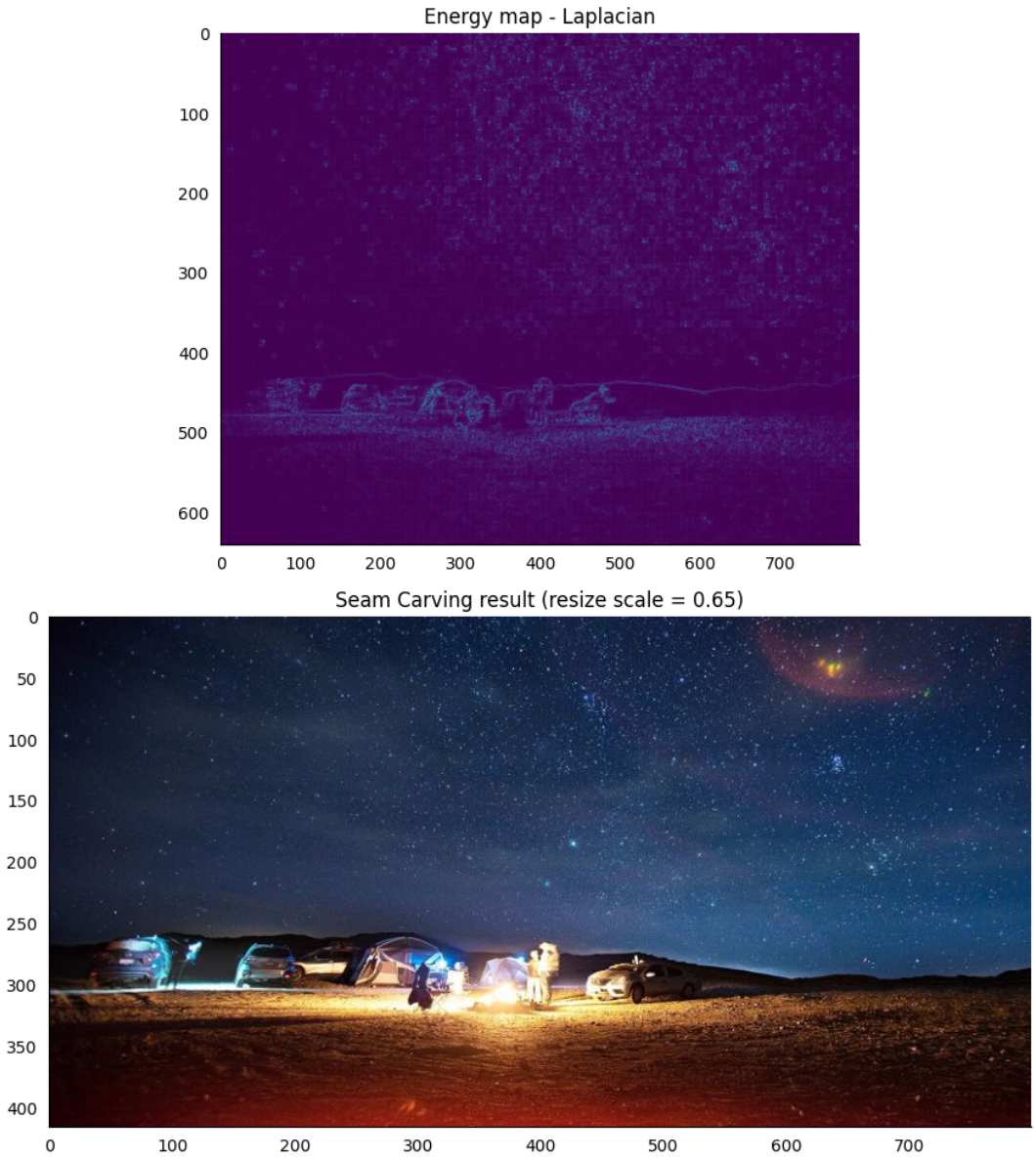


Figura 38 – Resultado do redimensionamento com o uso do Laplaciano para mapeamento de energia. Fonte: Autor

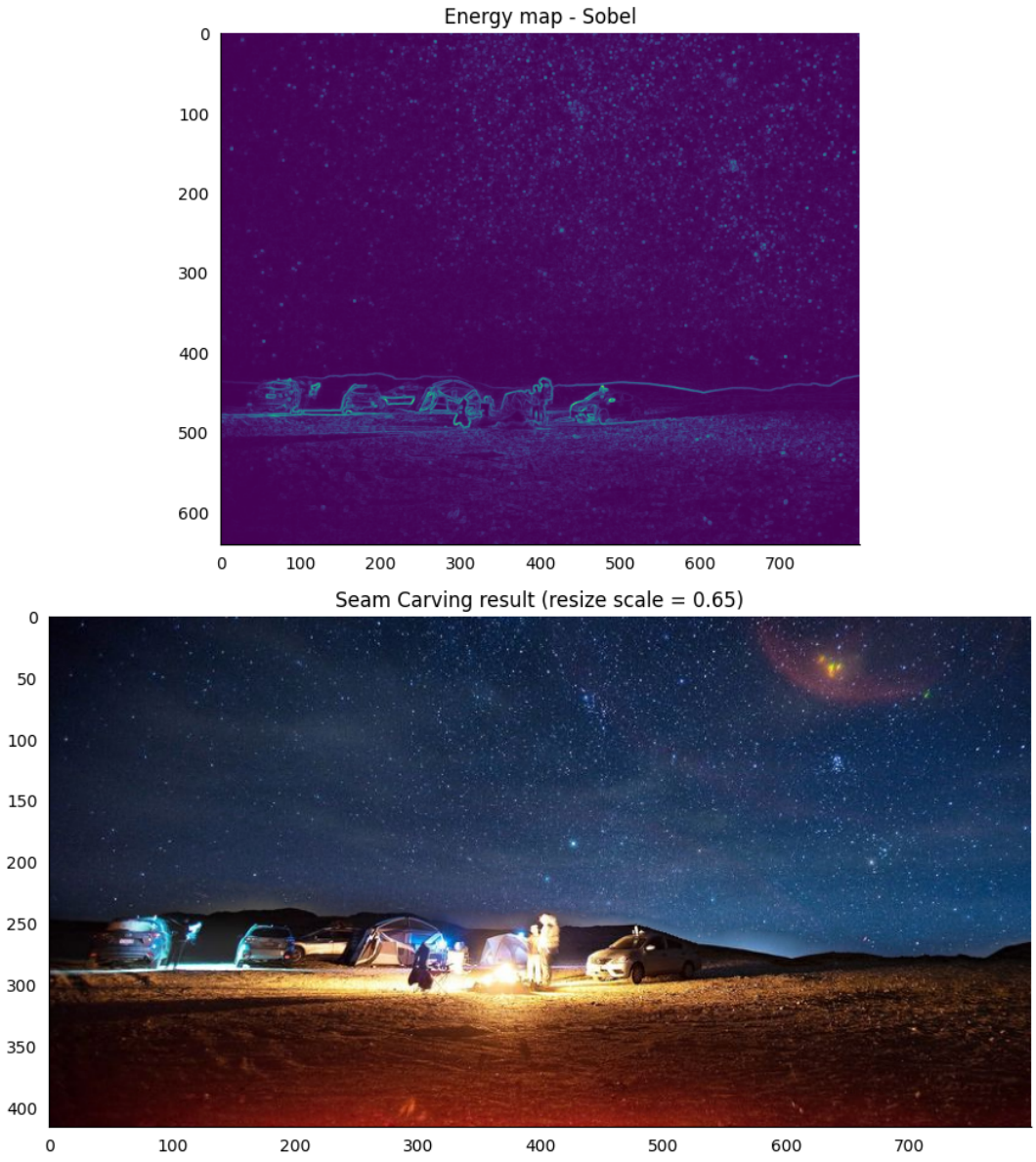


Figura 39 – Resultado do redimensionamento com o uso do Sobel para mapeamento de energia. Fonte: Autor

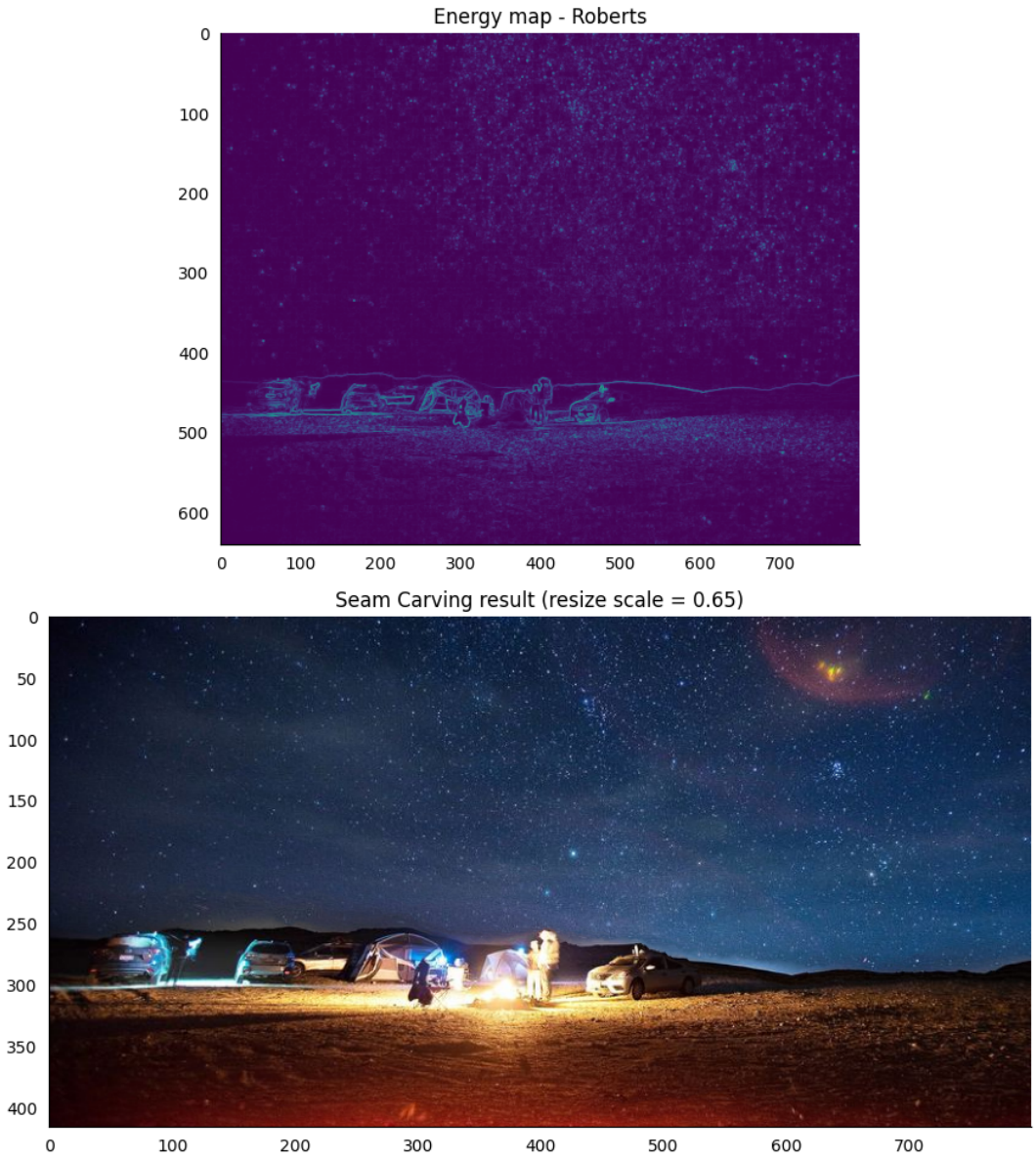


Figura 40 – Resultado do redimensionamento com o uso do Roberts para mapeamento de energia. Fonte: Autor

APÊNDICE C – RESULTADOS PARA A IMAGEM DE TESTE 3

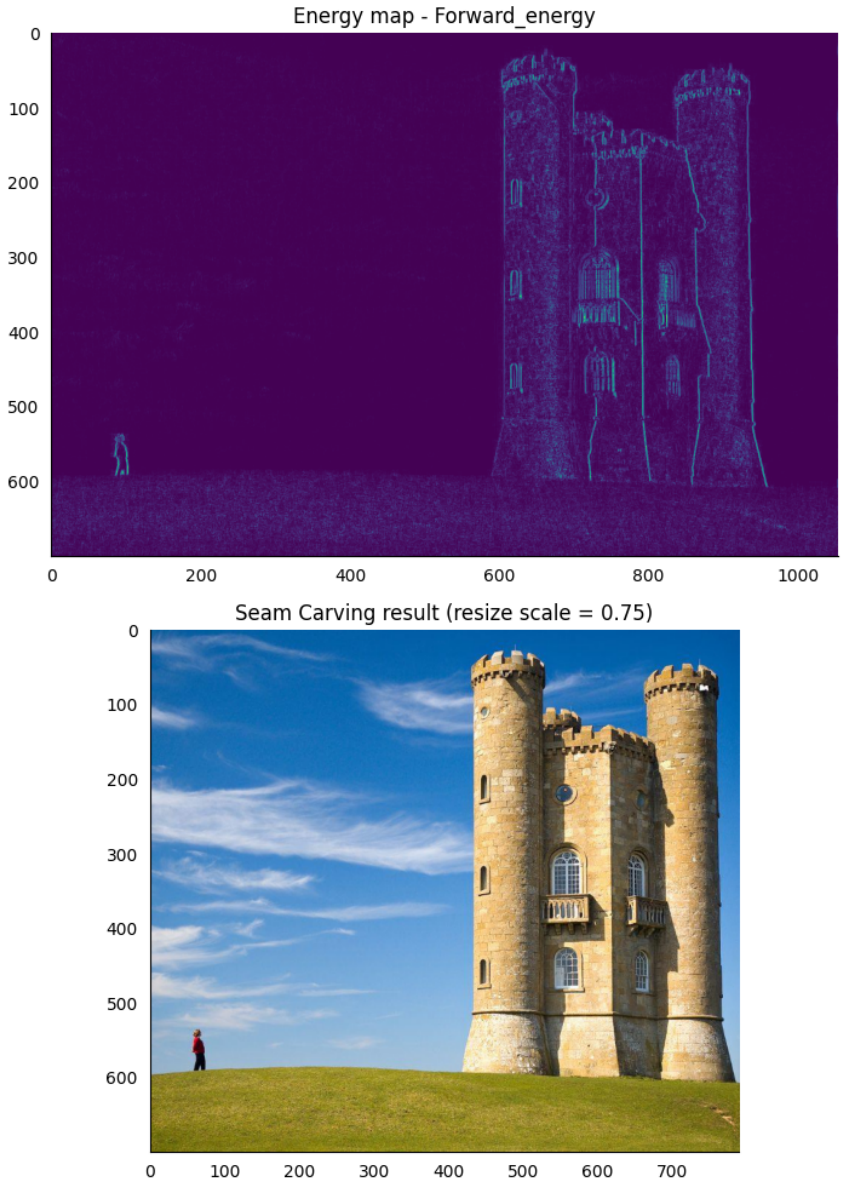


Figura 41 – Resultado do redimensionamento com o uso do Forward Energy para mapeamento de energia. Fonte: Autor

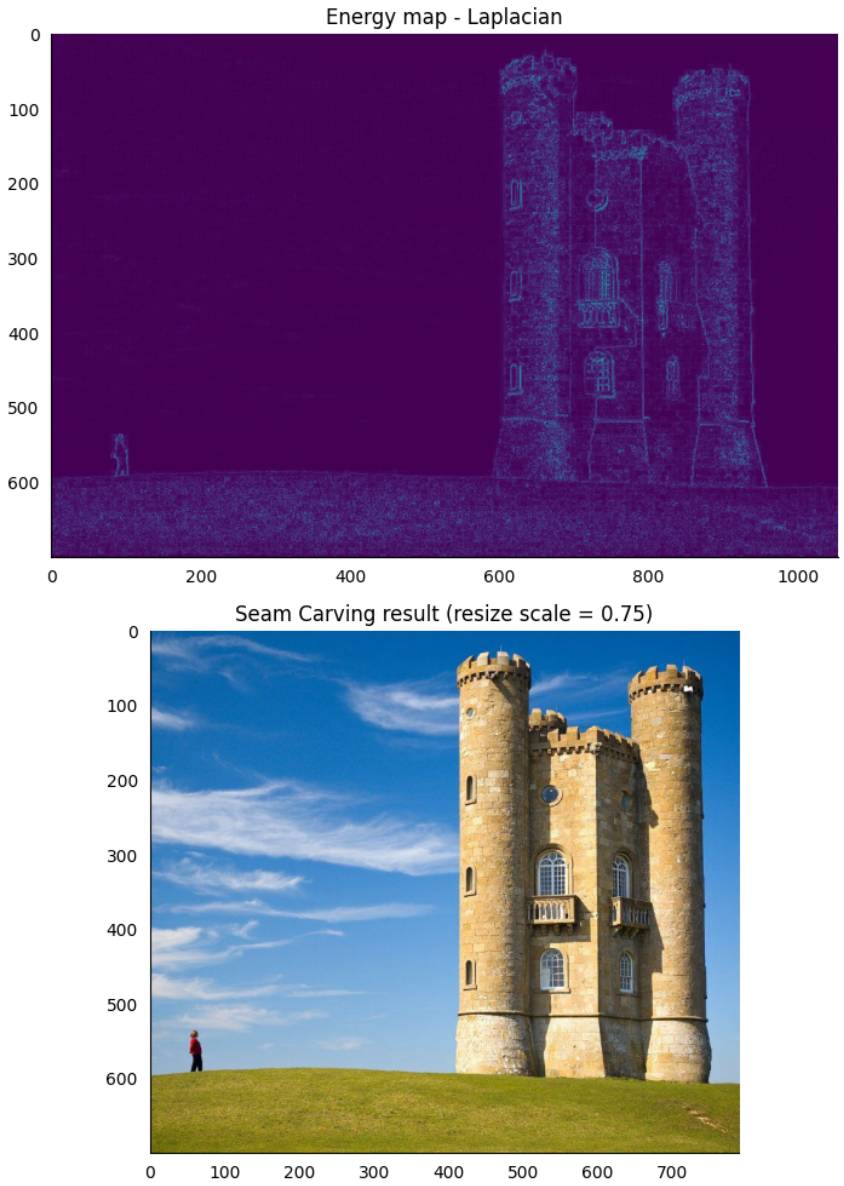


Figura 42 – Resultado do redimensionamento com o uso do Laplaciano para mapeamento de energia. Fonte: Autor

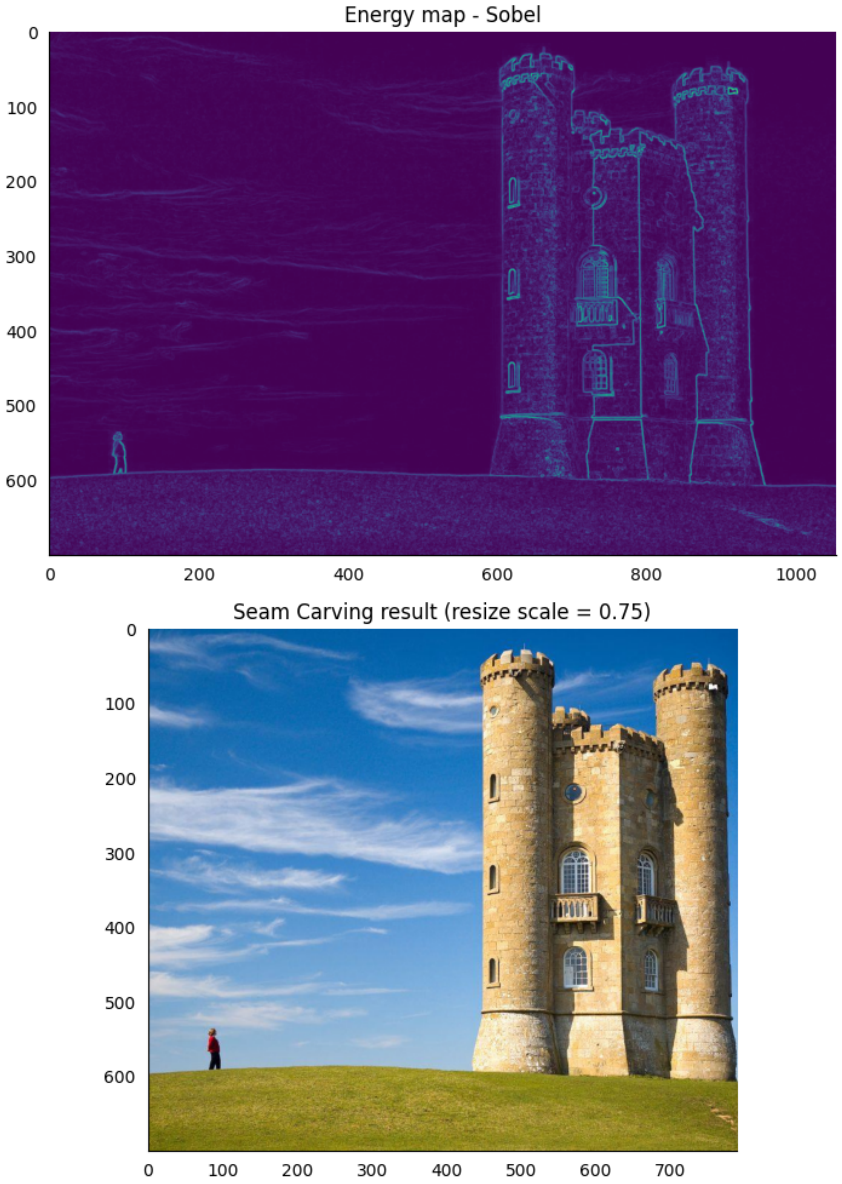


Figura 43 – Resultado do redimensionamento com o uso do Sobel para mapeamento de energia. Fonte: Autor

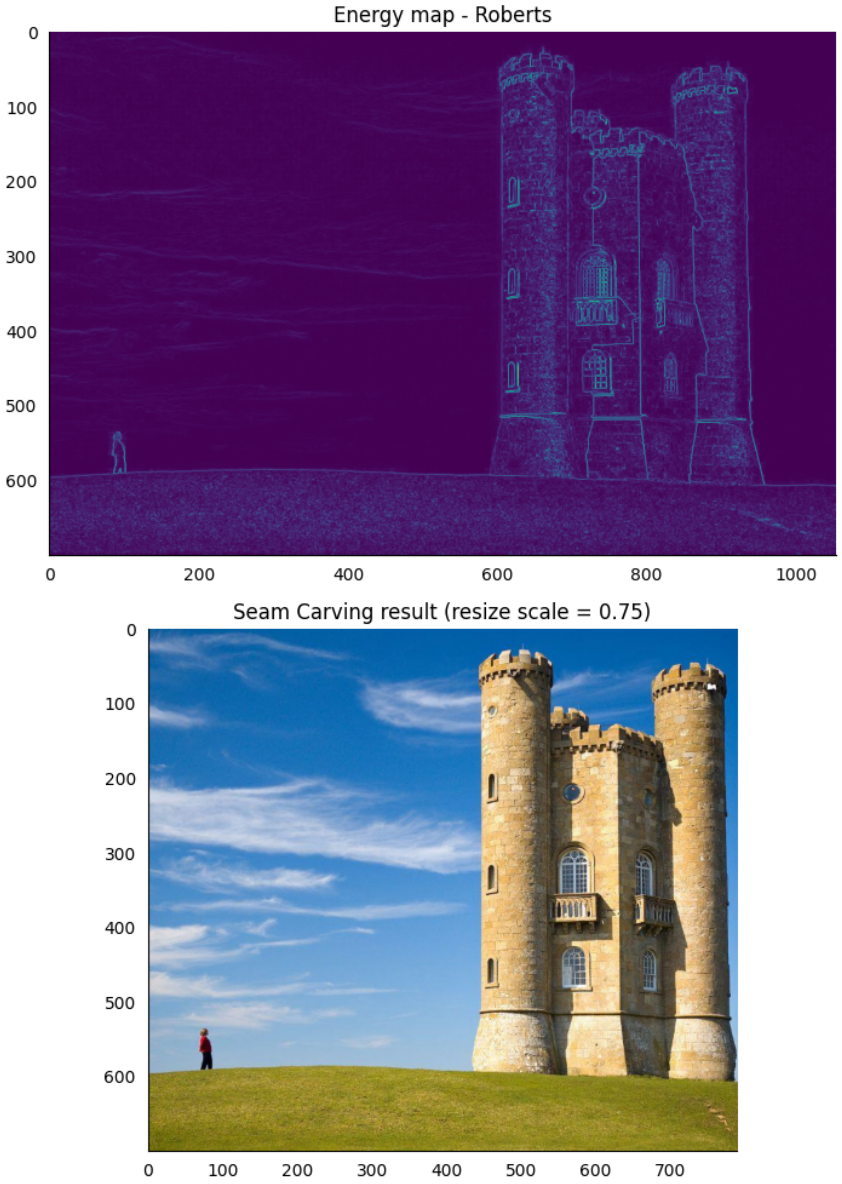


Figura 44 – Resultado do redimensionamento com o uso do Roberts para mapeamento de energia. Fonte: Autor

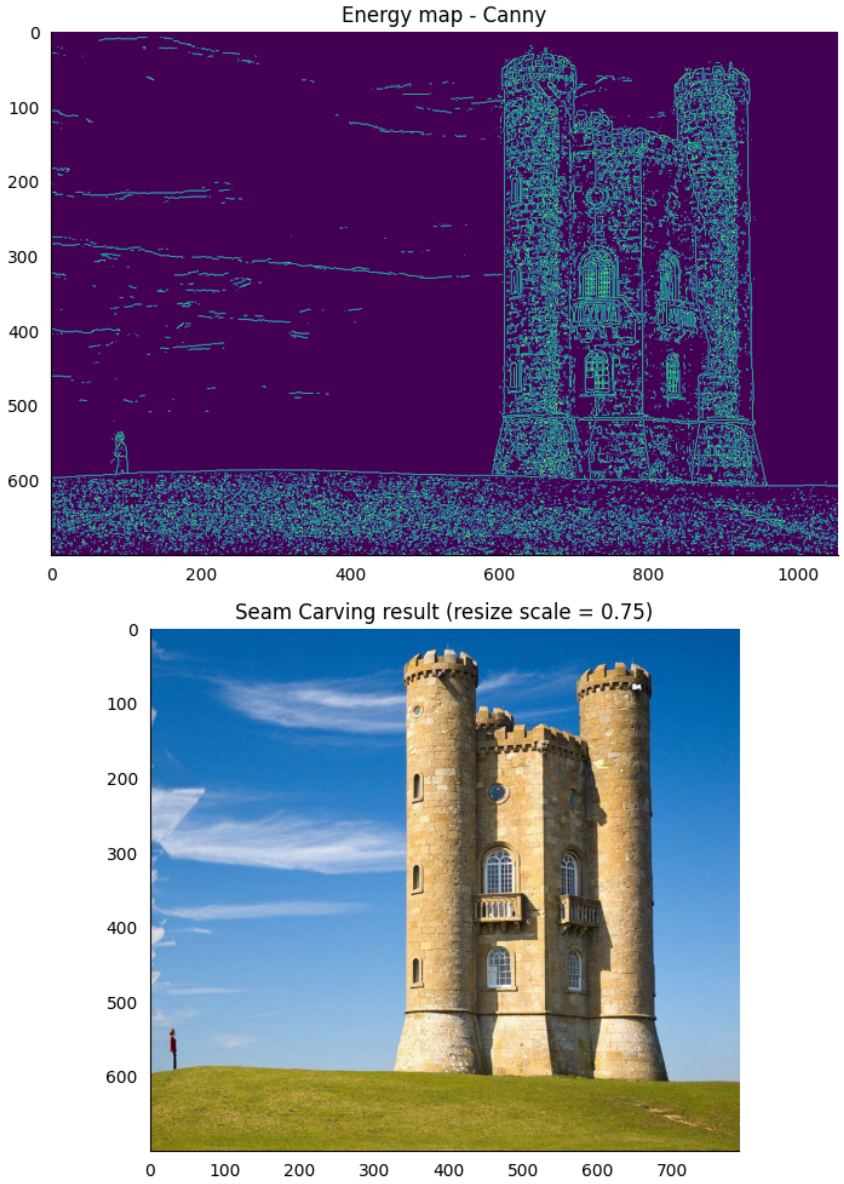


Figura 45 – Resultado do redimensionamento com o uso do Canny para mapeamento de energia. Fonte: Autor