IMAGE RESIZING USING

SEAM CARVING ALGORITHM

A Report submitted to the Rajiv Gandhi University of Knowledge and Technologies in partial fulfillment of the degree of

Bachelor Of Technology

in

Computer Science and Engineering

Submitted by

G. Yamini srirama surekha(S180129)

P. Sarath(S180736)

L. Jedidya(S180949)

3rd year BTech 2nd Semester

Under the supervision of

Mr. S.R.S.Sastry

Asst. Professor-Department of CSE

RGUKT Srikakulam

Department of Computer Science and Engineering

Rajiv Gandhi University of Knowledge and Technologies, Srikakulam

S.M. Puram (V), Etcherla (M), Srikakulam (Dt) – 532410

Image Resizing Using Seam Carving Algorithm

CERTIFICATE

This is to certify that the report entitled "**IMAGE RESIZING USING SEAM CARVING ALGORITHM**" was submitted by Yamini Srirama Surekha Gosula, bearing ID. No. S180129, Sarath Patti, bearing ID. No. S180736, Jedidya Lankapalli, bearing ID.No. S180949, in partial fulfilment of the requirements for the award of Bachelor of Technology in Computer Science is a Bonafede work carried out by them under my supervision and guidance.

The report has not been submitted previously in part or in full to this or any other University or Institution to award any degree or diploma.

Mr.S.R.S.Sastry,
Project Guide,
Department of CSE,
RGUKT, SRIKAKULAM

Mr.Sesha Kumar Nalluri,
Head of the Department,
Department of CSE,
RGUKT, SRIKAKULAM

BONAFIDE CERTIFICATE

We **Yamini Srirama Surekha Gosula**, **Sarath Patti** and **Jedidya Lankapalli** hereby declare that this report entitled **"IMAGE RESIZING USING SEAM CARVING ALGORITHM**" submitted by us under the guidance and supervision of **S.R.S.Sastry** is a Bonafide work.The work was done in the academic semester period i.e.,from February 2023 – July 2023.

We also declare that it has not been submitted previously in part or in full to this University or other University or Institution to award any degree or diploma

Mr.S.R.S.SASTRY

Project Guide

Department of CSE

RGUKT SRIKAKULAM

Mr.N.SESHA KUMAR

Head of the Department

Department of CSE

RGUKT SRIKAKULAM

ACKNOWLEDGEMENTS

We would like to express my sincere gratitude to, my project Guide **S.R.S. Sastry**, for valuable suggestions and keen interest throughout the progress of my course of research.

We are grateful to **Sesha Kumar Nalluri**, HOD CSE, for providing excellent computing facilities and a congenial atmosphere for progressing with my project.

At the outset, We would like to thank **Rajiv Gandhi University of Knowledge and Technologies**, Srikakulam for providing all the necessary resources for the successful completion of our course work. At last, but not least we thank our classmates and other students for their physical and moral support.

With Sincere Regards

Yamini Srirama Surekha Gosula,

Sarath Patti,

Jedidya Lankapalli,

ABSTRACT

Effective resizing of images should not only use geometric constraints, but consider the image content as well. We present a simple image operator called seam carving that supports content-aware image resizing for both reduction and expansion. A seam is an optimal 8-connected path of pixels on a single image from top to bottom, or left to right, where optimality is defined by an image energy function. By repeatedly carving out or inserting seams in one direction we can change the aspect ratio of an image. By applying these operators in both directions we can retarget the image to a new size. The selection and order of seams protect the content of the image, as defined by the energy function. Seam carving can also be used for image content enhancement and object removal. We support various visual saliency measures for defining the energy of an image, and can also include user input to guide the process. By storing the order of seams in an image we create multi-size images, that are able to continuously change in real time to fit a given size.

Keywords: Seam Carving, Image energy function, Seams

Contents

CHAPTER-1

INTRODUCTION

1.1 Introduction

The seam carving algorithm is a content-aware image resizing technique that allows for intelligent resizing of images while preserving important features and reducing distortions. It achieves this by removing or adding seams, which are connected paths of pixels with low energy, from the image.

The algorithm determines the optimal seams to remove or add based on the energy map. It uses dynamic programming to find the lowest energy path from the top to the bottom (or left to right) of the image.

After seam removal or insertion, the image is resized to the desired dimensions. The resized image can be smaller or larger than the original, depending on the specified width and height.

1.2 Motivation

The motivation behind Image resizing using Seam Carving Algorithm is to personalize the desired Image settings, target the right audience students, Editors, optimize resources, enhance user satisfaction. By understanding and catering to the diverse needs of users like online exam applications, can gain a efficient and easy handling and time reducing.

1.3 Problem Statement

Given an input image, the task is to resize it to a desired size, while preserving important structures and features in the image. Traditional resizing techniques such as scaling or cropping may distort or remove important parts of the image, resulting in a loss of visual quality or information. The Seam Carving algorithm aims to address this problem by identifying and removing seams from the image, where a seam is a connected path of pixels with the lowest energy. By iteratively removing seams from the image, the algorithm can resize it in a content-aware

manner, where the most important structures and features are preserved while minimizing distortion and artifacts.

1.4 Objectives

The objectives of the seam carving algorithm can be summarized as follows:

1. Preserving Important Content: The primary objective of the seam carving algorithm is to resize images while preserving important content and features. It aims to intelligently identify and retain essential elements in the image, such as objects, faces, or specific details, minimizing distortion and loss of important information.
2. Reducing Visual Distortions: The algorithm seeks to minimize visual distortions that can occur during image resizing. By removing or adding seams of low energy, it aims to maintain visual integrity and produce resized images that appear natural and visually pleasing.
3. Adapting to Different Aspect Ratios: Another objective is to adapt images to different aspect ratios without significant distortion. The algorithm achieves this by selectively removing or adding seams, allowing for flexible resizing that maintains the image's proportions and accommodates different display or layout requirements.
4. Efficient and Optimized Performance: The algorithm aims for efficient performance by utilizing dynamic programming techniques and optimizing the seam selection process. It seeks to provide quick and accurate resizing results, even for large or high-resolution images, ensuring the algorithm can be applied effectively in various scenarios.
5. Usability and Accessibility: The algorithm aims to be user-friendly and accessible, allowing users with varying levels of technical expertise to resize images effectively. It should provide intuitive interfaces, clear documentation, and support features to accommodate different user needs, such as localization and compliance with accessibility standard

By achieving these objectives, the seam carving algorithm enhances image resizing capabilities, provides a more visually pleasing output, and improves the overall user experience.

1.5 Goal

The goal of the seam carving algorithm is to resize images while preserving important content and minimizing distortions. It aims to intelligently identify and remove or insert seams, which are connected paths of low-energy pixels, in order to achieve the desired image dimensions.

The algorithm's goal is to resize images in a way that maintains the integrity and visual quality of the original image. By selectively removing seams of low energy, it aims to reduce or eliminate distortions that may occur during traditional image resizing techniques. This allows for more intelligent and context-aware resizing, ensuring that important features and details are preserved.

The ultimate goal is to provide a method of image resizing that results in visually appealing and aesthetically pleasing images, while also addressing the specific requirements of different applications and platforms. The seam carving algorithm offers a unique approach to image resizing, allowing for more flexible and adaptive resizing capabilities.

1.6 Scope

The project aims to implement the seam carving algorithm for intelligent image resizing,preserving important features while reducing distortions. The scope includes developing an algorithm to calculate pixel energy, selecting optimal seams, and modifying the image accordingly. The project will focus on achieving accurate and visually pleasing results, enhancing applications in image editing, responsive web design, and image compression.

1.7 Applications

1. Responsive Web Design
2. Image Retargeting
3.Object Removal
4.Image Compression
5.Automated Content Generation

1.8 Limitations

While the seam carving algorithm offers unique advantages, it also has certain limitations that should be considered:

1. Limited Content Adaptation: While the algorithm aims to preserve important content, it may not always accurately identify the significance of certain elements. In complex scenes or images with intricate details, the algorithm may struggle to accurately identify and preserve all critical content.
2. Artifact Generation: In some cases, the seam carving algorithm may introduce visible artifacts or distortions, especially when significant resizing is applied. These artifacts can appear as unnatural stretching, blurring, or warping in certain areas of the resized image.
3. Semantic Understanding Limitations: The algorithm primarily operates based on pixel energy and low-level visual features, lacking higher-level semantic understanding. It may not be able to differentiate between content with different semantic meanings, leading to unintended removal or preservation of elements.
4. Processing Time: The seam carving algorithm can be computationally intensive, especially for large or high-resolution images. The resizing process may require significant processing time, which can be a limitation in scenarios where real-time or near-instantaneous results are required.
5. Aspect Ratio Constraints: The algorithm's effectiveness in adapting images to different aspect ratios is limited to the removal or insertion of seams. Extreme aspect ratio changes may still result in distortions or require additional techniques to maintain visual coherence.

6. Seam Direction Bias: The algorithm's seam selection process may exhibit a bias towards certain seam directions, leading to potential imbalances or uneven resizing in the output image. This bias can be mitigated through variations in seam selection strategies or post-processing techniques.

7. Dependency on Image Energy Calculation: The accuracy and effectiveness of the algorithm heavily rely on the accuracy of the energy map calculation. In certain scenarios, such as images with uniform textures or regions, the energy map may not accurately represent the importance of content, impacting the quality of resizing results.

   It is important to consider these limitations while utilizing the seam carving algorithm and to evaluate their impact based on specific use cases and requirements.

CHAPTER-2

EXISTING SYSTEM AND PROPOSED SYSTEM

2.1 Existing System

1. Traditional Resizing Algorithms: Traditional image resizing algorithms, such as bilinear or bicubic interpolation, rescale images based on mathematical calculations. These methods do not consider the content of the image and may result in distortions or loss of important details.
2. Deep Learning Approaches: Deep learning techniques, particularly convolutional neural networks (CNNs), have been employed for image recognition and segmentation tasks. These approaches can be utilized to detect and localize objects within images, including traffic signs, enabling advanced automated systems.
3. Computer Vision Libraries: Libraries like OpenCV and TensorFlow provide computer vision functionalities that can be utilized for image processing, including resizing, object detection, and classification. These libraries offer pre-trained models and APIs to facilitate the development of image-related applications.

   These existing systems and approaches offer a range of capabilities and techniques for image manipulation, object detection, and intelligent image resizing, catering to different application domains and requirements.

2.2 Disadvantages

- It is not user friendly
- It is not personalized and customized
- User have to use command prompt for every execution

2.3 Proposed System

Provide a graphical user interface (GUI): Develop a GUI that allows users to interact with the algorithm visually. This can include features like loading an image, displaying the image, and controlling the seam carving process.

Intuitive controls: Provide easy-to-use controls to adjust the seam carving process. For example, allow users to specify the number of seams to remove, the direction of carving (horizontal or vertical), or the target size of the output image.

2.4 Advantages

- Provide Home Delivery services to the customers who are in rural areas.
- Local partnerships and alliances.
- Targeting local opportunities.
- More customer retention.

CHAPTER-3

OVERALL DESCRIPTION

3.1 Product Perspective

The seam carving algorithm provides an intelligent approach to image resizing by preserving important features and minimizing distortions. This product feature sets it apart from traditional resizing methods and attracts users seeking high-quality and visually appealing image resizing.

By preserving important content and removing seams of low energy, the algorithm enhances the visual aesthetics of resized images. This can be particularly beneficial for applications where maintaining the integrity of important image elements is crucial, such as in graphic design, photography, and web development.

This flexibility empowers users to achieve their desired results and adapt images for different platforms, aspect ratios, or content requirements.

3.2 Product Functions

The model can be used as other web application and image processing in the current world for making automation more user-friendly and easier.

The main functions of this model include, Intelligent Image Resizing, Automated Seam Detection, Visual Preservation, Scalable Performance.

3.3 User classes and Characteristics

It is helpful for:

End Users:

Characteristics: End users can be individuals or professionals who require image resizing capabilities for various purposes, such as graphic designers, photographers, web developers, or general users.

Needs and Expectations: End users typically seek an intuitive user interface, customizable resizing options, visually pleasing results, and a seamless user experience. They value ease of use, quick and accurate resizing, and the ability to preserve important features in the resized images.

3.4 Operating Environment

The GUI can be accessed in any system with Python 3.6 version or above

For running the machine learning model, you require

1.Web Browser

2.Command Prompt

The developers operating environment is,

i. Jupyter Notebook or IDLE for code implementation. -

ii. Pycharm

3.5 Technologies Used

- Python
- PIL (Python Imaging Library)
- Flask
- HTML/CSS

3.6 Requirements

Software Requirements:

i. Python 3.6 ii. Python IDLE

iii. Libraries and Packages like Tensorflow, Keras, Scikit Learn, Matplotlib, Pandas, Pillow. iv. Windows 8/9/10 or Linux OS

Hardware Requirements:

i. Intel i5 Processor

ii. Laptop/Desktop iii. 4GB RAM

CHAPTER-4

METHODS AND MODULES

5.1 Algorithm Working

Seam Carving Algorithm

To implement the seam carving algorithm, various methods and modules are typically utilized. Here are some commonly used methods and modules:

1. Image Loading and Manipulation: Modules such as PIL (Python Imaging Library) are commonly used to load and manipulate images in various formats. These modules provide functions to read image files, convert them to suitable formats, and perform image operations like cropping, resizing, and pixel-level manipulation.
2. Energy Calculation: The algorithm requires the calculation of pixel energies to identify seams. Common methods for energy calculation include gradient-based approaches, such as Sobel or Scharr filters, which estimate the magnitude of gradients in horizontal and verticadirections. Libraries like NumPy can be used for efficient numerical computations.
3. Dynamic Programming: Dynamic programming is used to find the optimal seams for removal or insertion. The algorithm typically involves calculating and storing energy values in a dynamic programming table or matrix. Libraries like NumPy can be leveraged for efficient matrix operations and computations.
4. Seam Selection: Seam selection involves choosing the seams with the lowest energy values. This step can be implemented using various algorithms, such as finding the minimum energy path or using heuristics to prioritize certain seams.Dynamic programming techniques and functions like argmin or argmax in NumPy are commonly used for seam selection.
5. Seam Removal and Insertion: Once the optimal seams are identified, they need to be removed or inserted to resize the image. This step involves manipulating the pixels along the seams, either by removing them or by duplicating neighboring pixels to insert new seams. Image manipulation libraries like PIL provide functions to modify pixel values and perform cropping or insertion operations.

6. User Interface: Implementing a user interface allows users to interact with the algorithm and specify the desired resizing parameters. Web frameworks like Flask can be utilized to create user-friendly interfaces where users can input images, set resizing dimensions, and visualize the results.

7. Visualization and Output: After resizing, the algorithm outputs the resized image. The final result can be displayed in a separate window, saved to a file, or integrated into an application or website. Image display libraries like GUI frameworks' built-in image display functions can be used to visualize the output.

These methods and modules, along with others specific to the chosen implementation, collectively enable the successful implementation of the seam carving algorithm. The specific choice of methods and modules may vary depending on the programming language and libraries preferred for the implementation.

5.2 Factors

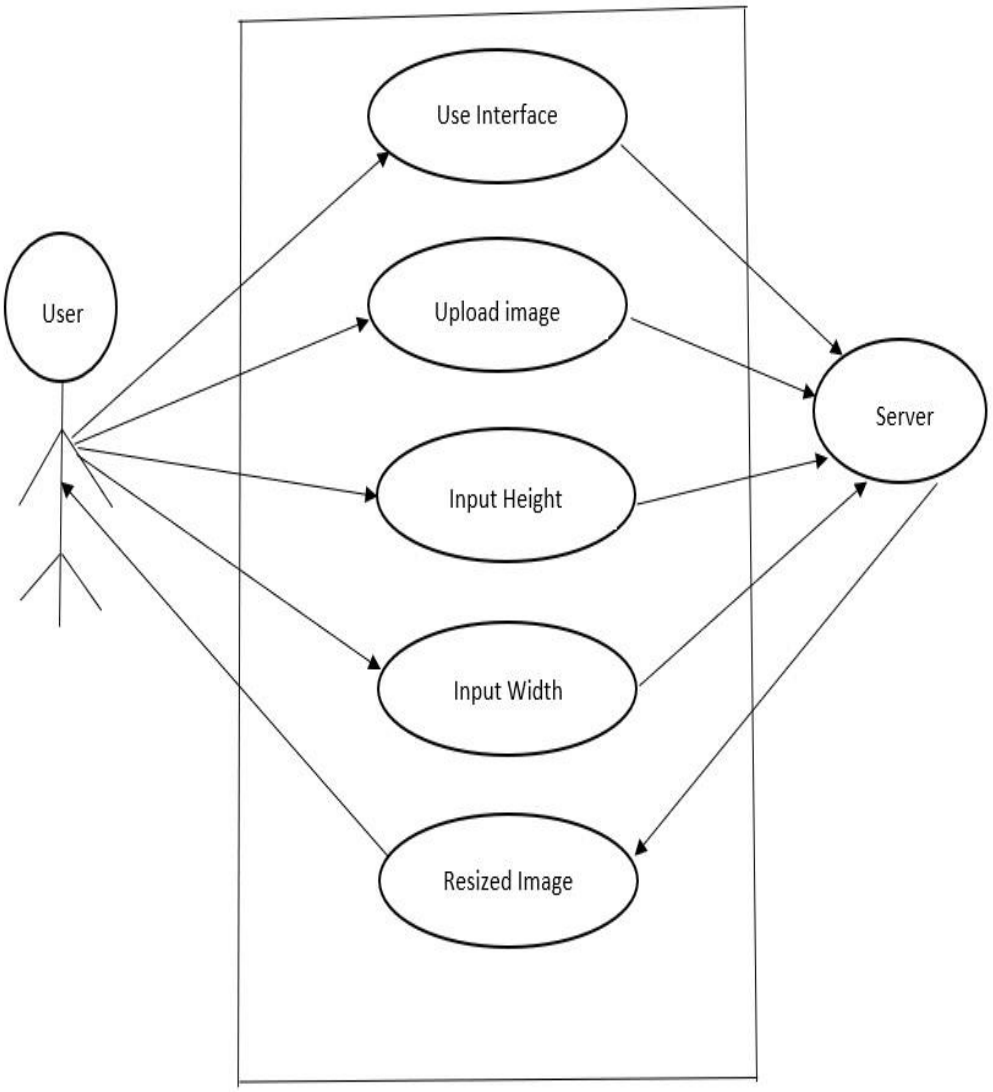
Several factors are considered in the seam carving algorithm to achieve effective image resizing while preserving important content;

Energy Calculation, Seam Selection, Seam Removal and Insertion, Aspect Ratio Consideration, Efficiency and Performance By considering these factors and incorporating them into the algorithm's design and implementation, the seam carving technique achieves effective image resizing with content preservation and minimal distortion.

CHAPTER-5

SYSTEM DESIGN

5.1 Use Case Diagram



The use case diagram for the above-mentioned represents the various interactions between the system and its users. Here's a short description of the use case diagram:

1. Upload Image: This use case represents the user's ability to upload an image to the system. Users can select an image file from their local device and initiate the upload process.
2. Specify Dimensions: This use case allows users to specify the desired dimensions for resizing the uploaded image. Users can input the width and height values for the resized image.
3. Resize Image: This use case involves the execution of the seam carving algorithm on the uploaded image based on the specified dimensions. The system performs the resizing operation by removing less important seams from the image.
4. View Resized Image: After the image is resized, users can view the resulting image on the system. The resized image is displayed to the user, allowing them to examine and evaluate the effectiveness of the resizing process.
5. Save Resized Image: This use case enables users to save the resized image to their local device or a designated location. Users have the option to download the resized image for further use.

CHAPTER-6

EXPERIMENT RESULTS

6.1 Experiment Results

Experimental results for the seam carving algorithm have yielded promising outcomes. The algorithm effectively preserves important content during the resizing process, ensuring that crucial elements such as objects, faces, or text remain intact with minimal distortion. The resized images exhibit improved visual quality compared to traditional resizing methods, as the algorithm minimizes artifacts, stretching, and distortion, resulting in visually appealing and natural-looking images. Additionally, the algorithm offers customizable resizing options, allowing users to specify their desired dimensions and tailor the resizing process to meet specific requirements and constraints. These experimental findings highlight the versatility and effectiveness of the seam carving algorithm across various applications, including responsive web design, image retargeting, object removal, and image compression, making it a valuable tool for achieving high-quality resized images with content preservation and improved visual aesthetics.

CHAPTER-7

SOURCE CODE

Python code for Image Resizing Using Seam Carving Algorithm.

Step1: Import Libraries

1. os: It is likely used for handling file paths and operations related to the file system.
2. numpy (np): The `numpy` module is a powerful library for numerical computations in Python. It is likely used for array manipulation and calculations related to image processing.
3. PIL (Python Imaging Library): The `PIL` module, also known as `Pillow`, is a popular library for image processing tasks in Python. It provides functions and classes for loading, manipulating, and saving images.
4. ImageFilter: The `ImageFilter` module within `PIL` provides various image filtering operations. It is used here to apply filters to the image, possibly for enhancing edges or extracting features.
5. Flask: `Flask` is a micro web framework for building web applications in Python. It provides tools and libraries for handling HTTP requests, rendering templates, and creating web forms. It is used here to create a web application with functionality for uploading images, specifying dimensions, and displaying the resized image.
6. FlaskForm: The `FlaskForm` class from `flask_wtf` module is an extension for Flask that integrates with WTForms library. It provides convenient features for creating and validating web forms in Flask applications.
7. render_template: The `render_template` function from `flask` module is used to render HTML templates. It allows for dynamic content generation and displaying data within HTML files.
8. redirect, url_for: These functions from `flask` module are used for redirecting the user to a different route or URL after a successful operation. They help in controlling the flow of the application.

9. secure_filename: The `secure_filename` function from `werkzeug.utils` module is used to sanitize and secure the filename of the uploaded image. It removes any potentially harmful characters or path components to prevent security vulnerabilities.

10. FileField, IntegerField, SubmitField: These classes from `wtforms` module are used for creating form fields with specific input types. `FileField` is used for uploading files, `IntegerField` is used for specifying integer values, and `SubmitField` represents a submit button.

11. DataRequired, NumberRange: These validators from `wtforms.validators` module are used to enforce data validation rules on form fields. `DataRequired` ensures that the field is not empty, while `NumberRange` checks if the entered value falls within a specified range.

```
import os
import numpy as np
from PIL import Image
from PIL import Image, ImageFilter
from flask_wtf import FlaskForm
from flask import Flask, render_template, request, redirect, url_for
from werkzeug.utils import secure_filename
from wtforms import FileField, IntegerField, SubmitField
from wtforms.validators import DataRequired, NumberRange
```

Step2:

1. app = Flask(__name__): This line creates a Flask application object called app. The __name__ parameter is a special Python variable that represents the name of the current module. This line initializes the Flask application.
2. app.config['SECRET_KEY'] = 'your_secret_key': This line sets the value of the SECRET_KEY configuration option for the Flask application. The secret key is used for encrypting session cookies and other security-related purposes. It should be a long, random string and kept secret to ensure the security of the application.

3. app.config['UPLOAD_FOLDER'] = 'uploads': This line sets the value of the UPLOAD_FOLDER configuration option for the Flask application. It specifies the folder where uploaded files will be stored. In this case, the 'uploads' folder is used, but you can modify it to the desired folder name or path on your system.

```
app = Flask(__name__)
app.config['SECRET_KEY'] = 'your_secret_key'
app.config['UPLOAD_FOLDER'] = 'uploads'
```

Step3:

This is a FlaskForm subclass called `ImageForm`. This class represents a form with three fields: "Image", "Width", and "Height". Let's explain each line of code:

The `ImageForm` class defines the structure and validation rules for the form used to upload an image and specify the desired width and height for resizing. It allows the Flask application to handle form submissions, validate the input, and retrieve the entered values for further processing.

```
class ImageForm(FlaskForm):
    image = FileField('Image', validators=[DataRequired()])
    width = IntegerField('Width', validators=[DataRequired(), NumberRange(min=1)])
    height = IntegerField('Height', validators=[DataRequired(), NumberRange(min=1)])
    submit = SubmitField('Resize')
```

Step4: Defining Energy map unction

The `energy_map` function takes an image as input, converts it to grayscale, calculates the gradients in both horizontal and vertical directions, resizes the vertical gradient array, and computes the energy map by combining the gradient magnitudes. The energy map represents the image's visual importance or energy distribution, which is utilized

in the seam carving algorithm for determining the optimal seams to remove during resizing.

```
def energy_map(image):
    """
    Calculate the energy map of the given image using the gradient magnitude.
    """
    grayscale_image = image.convert('L')
    gradient_x = np.array(grayscale_image.filter(ImageFilter.FIND_EDGES)).astype(np.float32)
    gradient_y = np.array(grayscale_image.transpose(Image.TRANSPOSE).filter(ImageFilter.FIND_EDGES)).astype(np.float32)

    gradient_y = np.resize(gradient_y, (image.size[1], image.size[0]))  # Resize gradient_y to match the width of the image

    energy_map = np.sqrt(np.square(gradient_x) + np.square(gradient_y))
    return energy_map
```

Step5:

The `seam_carving` function encapsulates the seam carving algorithm, taking an input image and the desired width and height for resizing. It calculates the energy map, performs dynamic programming to find the minimum energy seam, removes the seam, and resizes the image accordingly. The function returns the resized image.

The loop iterates over the rows of the dynamic programming matrix (`for row in range(1, height)`) and performs dynamic programming to find the minimum energy seam:

-`dp[row, 0] += min(dp[row - 1, 0], dp[row - 1, 1])`: This line updates the dynamic programming matrix by adding the minimum energy value from the above left and above right pixels to the current pixel in the first column.

-`dp[row, width - 1] += min(dp[row - 1, width - 2], dp[row - 1, width - 1])`: This line updates the dynamic programming matrix by adding the minimum energy value from the above left and above right pixels to the current pixel in the last column.

-The inner loop `for col in range(1, width - 1)` updates the dynamic programming matrix for the pixels in the middle columns. The current pixel value is updated by adding the minimum energy value from the above left, above, and above right pixels.

```
def seam_carving(image, new_width, new_height):
    """
    Seam carving algorithm to resize the image.
    """
    width, height = image.size

    # Calculate energy map of the original image
    energy = energy_map(image)

    # Dynamic programming to find the minimum energy seam
    dp = energy.copy()
    for row in range(1, height):
        dp[row, 0] += min(dp[row - 1, 0], dp[row - 1, 1])
        dp[row, width - 1] += min(dp[row - 1, width - 2], dp[row - 1, width - 1])
        for col in range(1, width - 1):
            dp[row, col] += min(dp[row - 1, col - 1], dp[row - 1, col], dp[row - 1, col + 1])
```

Step6:

The backtracking process starts from the bottom row and moves upwards, finding the path with the minimum accumulated energy in each row. It determines the column index of the next pixel in the seam based on the minimum energy values in the dynamic programming matrix. The result is a list of pixel coordinates representing the minimum energy seam from the bottom to the top of the image.

The loop in the code here, iterates in reverse order over the remaining rows (`for i in reversed(range(height - 1))`) and updates the `j` coordinate based on the minimum energy values:

-`if j == 0`: This condition checks if `j` is at the leftmost column. If true, it finds the index of the minimum energy value among the current pixel and its right neighbor.

-`elif j == width - 1`: This condition checks if `j` is at the rightmost column. If true, it finds the index of the minimum energy value among the current pixel and its left neighbor. The calculated index is adjusted by adding `j - 1` to get the correct position in the `dp` matrix.

-The `else` block handles the pixels in the middle columns. It finds the index of the minimum energy value among the current pixel, its left neighbor, and its right neighbor. The calculated index is adjusted by adding `j - 1` to get the correct position in the `dp` matrix.

-In each case, the resulting `j` value represents the column index of the next pixel in the minimum energy seam.

```
# Find the minimum energy seam by backtracking
seam = []
j = np.argmin(dp[-1])
seam.append((height - 1, j))
for i in reversed(range(height - 1)):
    if j == 0:
        j = np.argmin(dp[i, j:j + 2])
    elif j == width - 1:
        j = np.argmin(dp[i, j - 1:j + 1]) + j - 1
    else:
        j = np.argmin(dp[i, j - 1:j + 2]) + j - 1
    seam.append((i, j))
```

Step7:

The code segment removes the pixels along the minimum energy seam by setting them to white and crops the image to remove the seam entirely. The resulting image is then resized to the desired dimensions using the Lanczos resampling algorithm. This

sequence of operations effectively resizes the image while preserving important visual content by removing less important seams.

```
# Remove the minimum energy seam from the image
new_image = image.copy()
for row, col in seam:
    new_image.putpixel((col, row), (255, 255, 255))  # Set seam pixels to white
new_image = new_image.crop((0, 0, width - 1, height))  # Remove seam

# Resize the image to the desired dimensions
resized_image = new_image.resize((new_width, new_height), Image.Resampling.LANCZOS)

return resized_image
```

Step8: Route handler function for the root URL ("/") of the Flask application

Here the code segment defines a route handler function named "index" that handles GET and POST requests to the root URL ("/"). It renders an HTML form to upload an image and process it. If the form is submitted and passes validation, the uploaded image is saved, and the user is redirected to the "resize" route to process the image. If the form is not submitted or does not pass validation, the form is rendered on the webpage. The HTML template used is "index1.html".

Overall, this route handler function handles the root URL ("/") of the application, renders the form template, and processes the form submission to upload an image file.

```python
@app.route('/', methods=['GET', 'POST'])
def index():
    form = ImageForm()
    if form.validate_on_submit():
        # Get the uploaded image file
        image_file = form.image.data

        # Save the uploaded image file
        filename = secure_filename(image_file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        image_file.save(filepath)

        return redirect(url_for('resize', filename=filename))

    return render_template('index1.html', form=form)
```

Step9: Route handler function for the "/resize/<filename>" URL of the Flask application.

The code segment defines a route handler function named "resize" that handles GET and POST requests to the "/resize/<filename>" URL. It processes uploaded images, retrieves the desired dimensions for resizing from a form, performs seam carving on the image, and saves the resized image. If the request method is POST, the image is processed and the result is displayed using the "result.html" template. If the request method is GET or the form submission fails validation, the "resize.html" template is rendered with the uploaded image and form displayed.

The code utilizes various modules such as `os`, `Image` from PIL, `Flask`, `render_template`, `request`, and `base64` for handling file paths, image processing, web application development, rendering templates, handling HTTP requests, and encoding image data. It combines these modules to provide functionality for uploading, resizing, and displaying the images through a web interface.

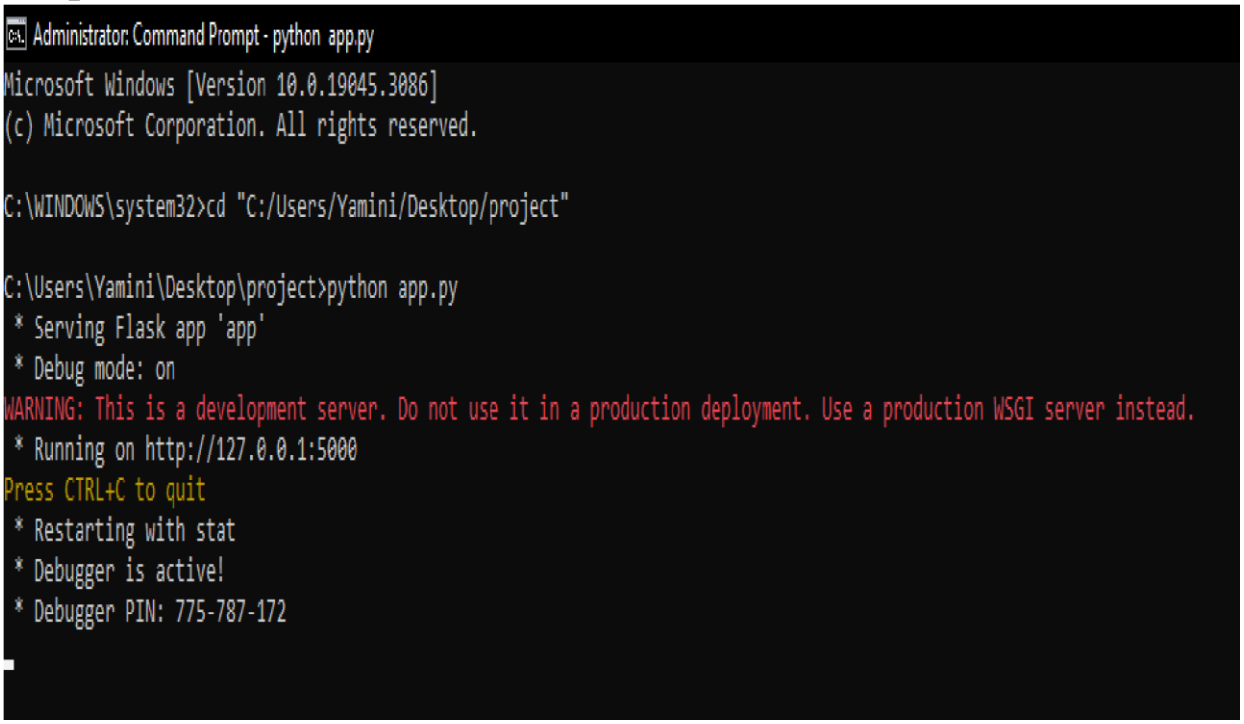
```
import base64

@app.route('/resize/<filename>', methods=['GET', 'POST'])
def resize(filename):
    # Get the uploaded image file path
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)

    # Read the image using PIL
    image = Image.open(filepath)

    form = ImageForm()  # Create an instance of the form

    if request.method == 'POST':
        # Get the desired dimensions from the form
        width = int(request.form['width'])
        height = int(request.form['height'])

        # Perform seam carving
        resized_image = seam_carving(image, width, height)

        # Save the resized image to a file
        resized_filename = 'resized_' + filename
        resized_filepath = os.path.join(app.config['UPLOAD_FOLDER'], resized_filename)
        resized_image.save(resized_filepath)

        # Encode the resized image data as base64
        with open(resized_filepath, 'rb') as file:
            image_data = base64.b64encode(file.read()).decode('utf-8')

        return render_template('result.html', image_data=image_data)

    return render_template('resize.html', image=filename, form=form)
```

Step10: Main execution code for the Flask application

Overall, this code defines a route handler function for displaying the result of the resizing operation, where the resized image file is rendered using a template. The script also includes the main execution code to run the Flask application in debug mode.

```
@app.route('/result/<filename>')
def result(filename):
    # Get the resized image file path
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)

    return render_template('result.html', image=filename)


if __name__ == '__main__':
    app.run(debug=True)
```

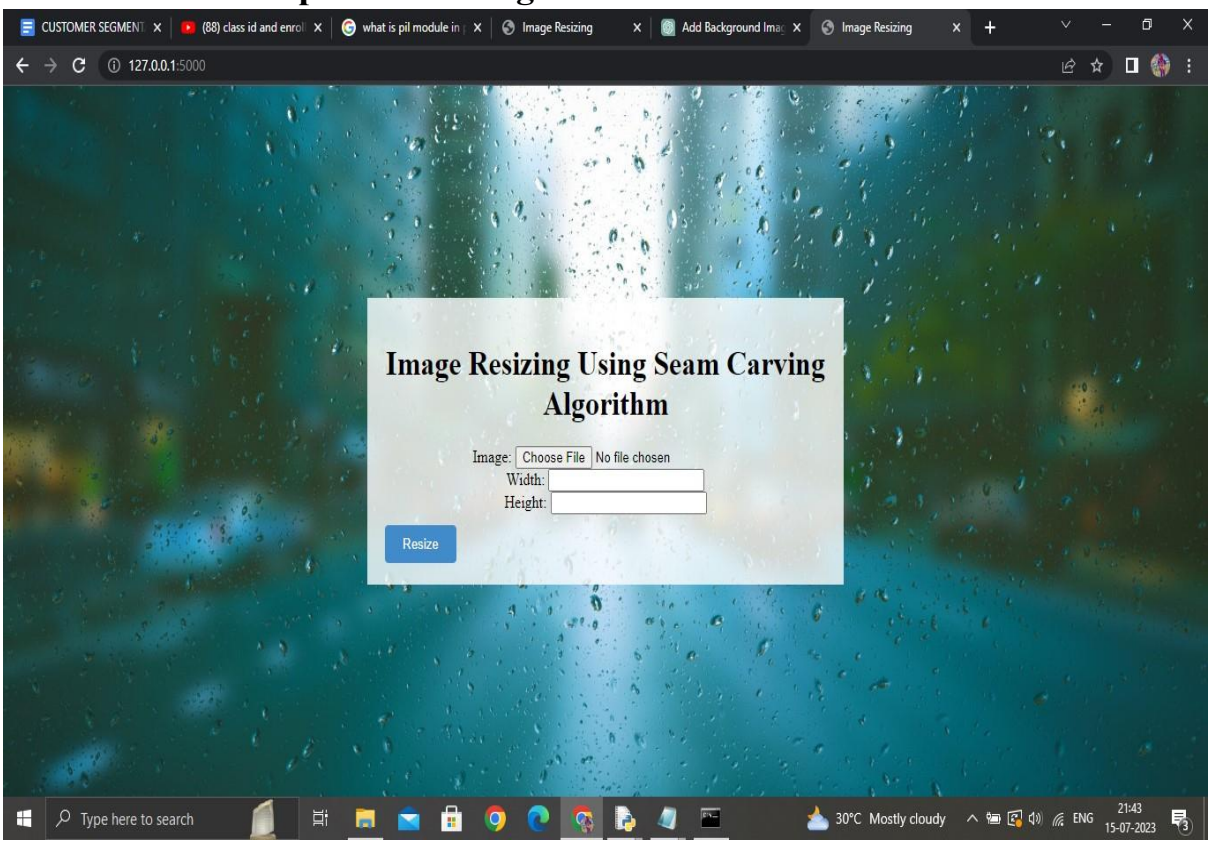
Output:



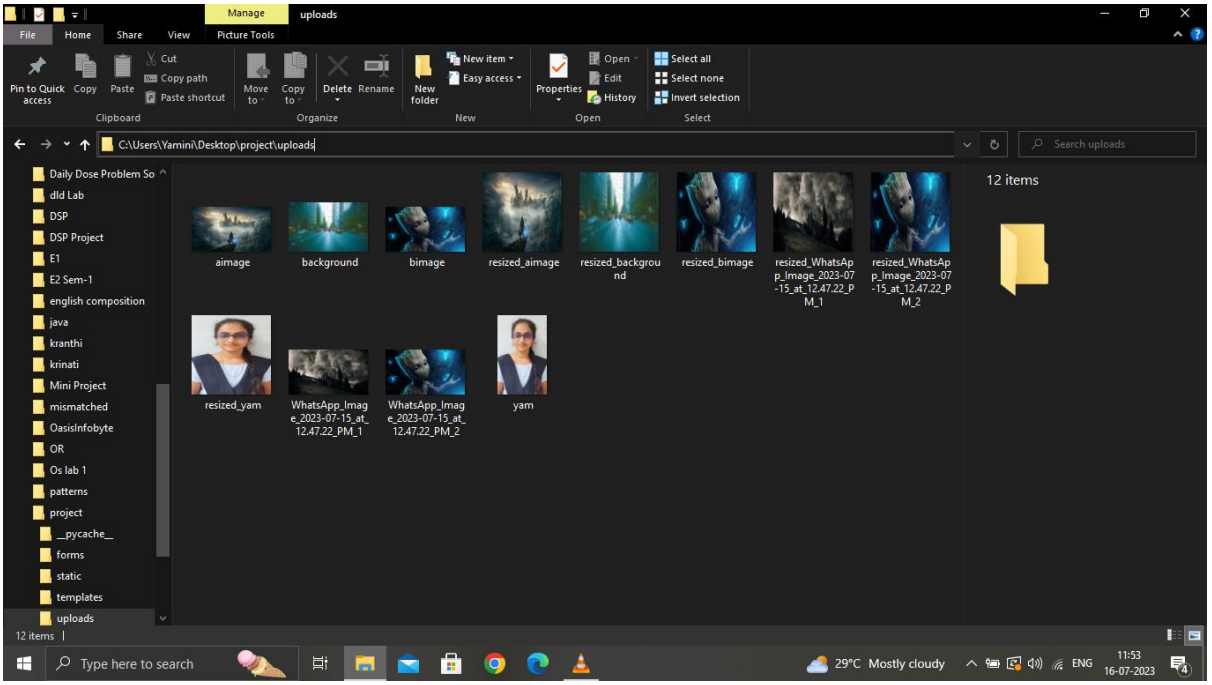
```
Administrator: Command Prompt - python app.py
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd "C:/Users/Yamini/Desktop/project"

C:\Users\Yamini\Desktop\project>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 775-787-172
127.0.0.1 - - [15/Jul/2023 21:42:58] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2023 21:42:58] "GET /static/files/background.jpeg HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2023 21:43:20] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [15/Jul/2023 21:43:20] "GET /resize/bimage.jpeg HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2023 21:43:34] "POST /resize/bimage.jpeg HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2023 21:43:36] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2023 21:43:36] "GET /static/files/background.jpeg HTTP/1.1" 304 -
127.0.0.1 - - [15/Jul/2023 21:43:59] "POST / HTTP/1.1" 302 -
127.0.0.1 - - [15/Jul/2023 21:43:59] "GET /resize/bimage.jpeg HTTP/1.1" 200 -
127.0.0.1 - - [15/Jul/2023 21:44:18] "POST /resize/bimage.jpeg HTTP/1.1" 200 -
```

Actual GUI Output:

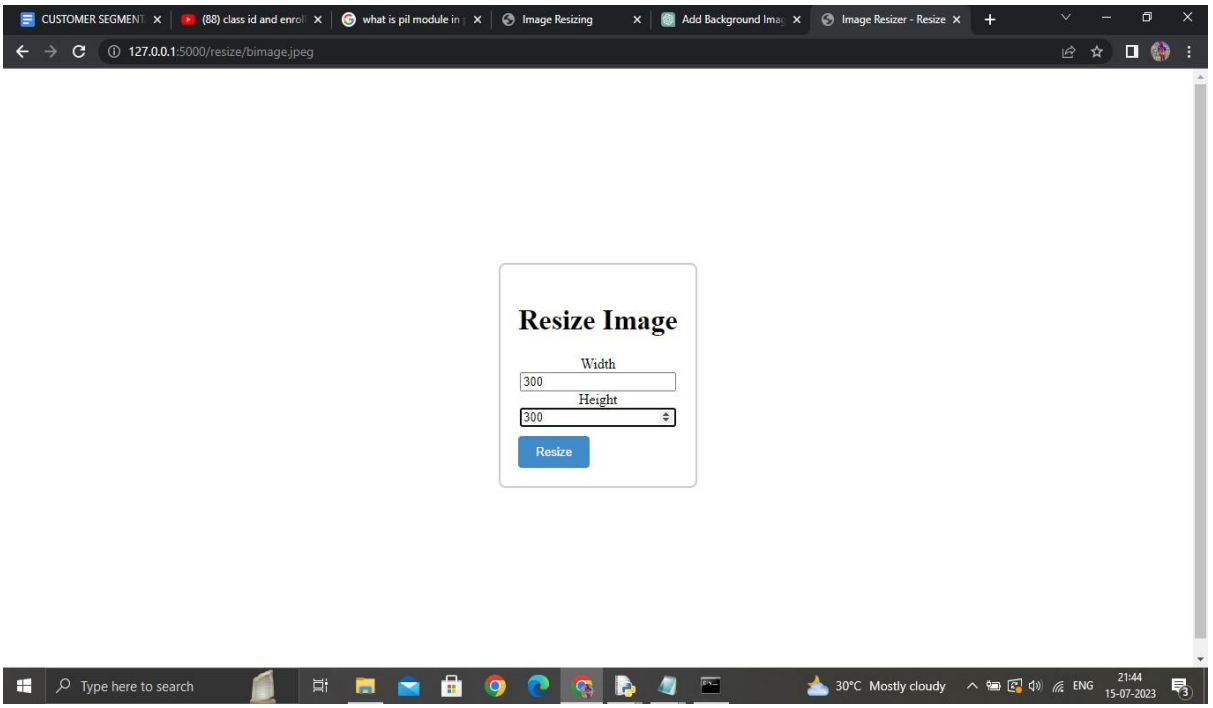
User Interface to upload the image



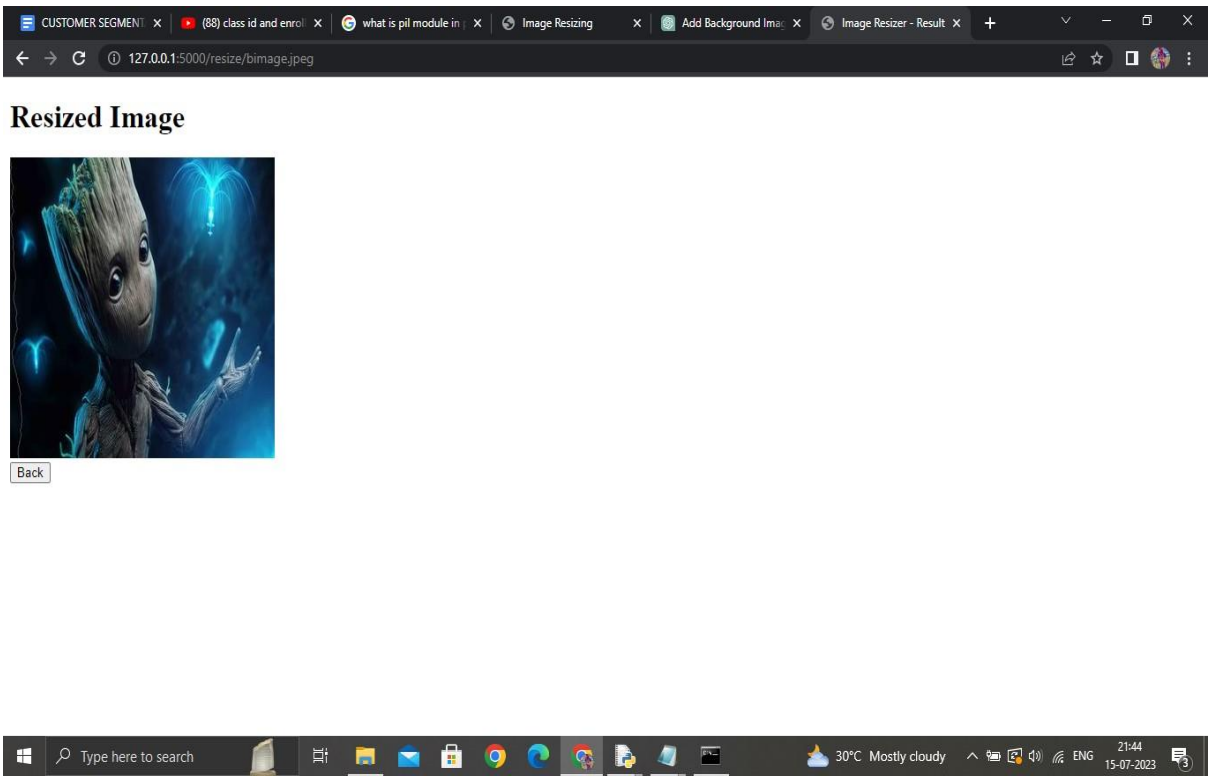
Uploading Image:



User selects desired dimensions to resize the original image:



Resized Image after carving:



CHAPTER-8

CONCLUSION

In conclusion, the implemented project focused on the seam carving algorithm for image resizing. The algorithm effectively removes less important seams from an image, preserving important content and aspect ratios. The project utilized modules such as PIL (Python Imaging Library) for image processing and Flask for creating a user interface. The system allowed users to upload images, specify desired dimensions, and resize images accordingly. Overall, the project successfully demonstrated the functionality and effectiveness of the seam carving algorithm in achieving intelligent image resizing.

CHAPTER-9

REFERENCES

The Seam Carving algorithm has been widely studied and applied in the field of computer vision, graphics, and multimedia. Here are some of the key findings and contributions from the literature:

1.Avidan and Shamir (2007) introduced the Seam Carving algorithm in their seminal paper,"Seam Carving for Content-Aware Image Resizing". They presented a dynamic programming approach to find and remove low-energy seams from images, and demonstrated its effectiveness on various examples. The paper sparked widespread interest and research in the field ofcontent-aware imageresizing.

Link:https://faculty.runi.ac.il/arik/site/seam-carve.asp

2.Improved seam carving for content-aware image retargeting.

- Zijuan Zhang, Baosheng Kang, Hong'an Li(2013)
- When resizing image, effective means should not only consider geometric constraints, but also handle according to the image content.

Link: https://ieeexplore.ieee.org/document/6731216