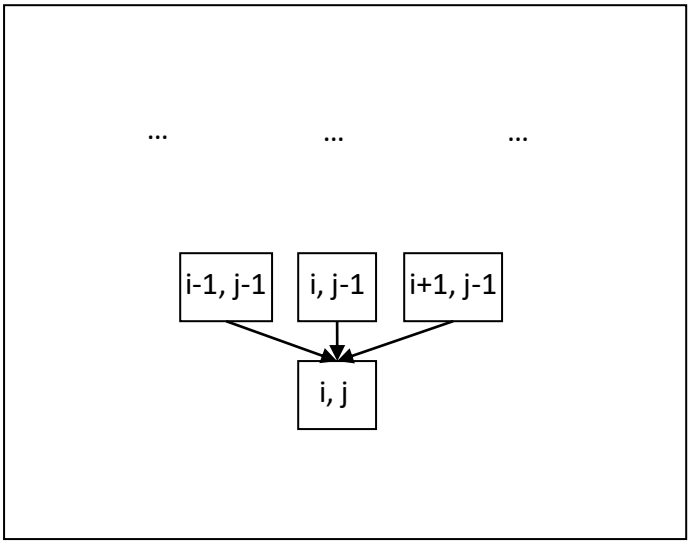
Content-Aware Image Resizing

Description

Given an image in form of 2D array of values representing its pixels, we need to reduce its size by removing some pixels. Each pixel has a value that indicates its importance based on its color called **ENERGY**. To avoid disturbing visual effects, we require to obtain the **vertical seam with minimum total energy.**

A vertical seam is a vertical **connected** path of pixels, one pixel in each row as in Figure 1**.** We call two pixels **connected** if they are vertically (in the same column) or diagonally adjacent as in Figure 2.

Figure 1: First smallest vertical seam in the Dolphin image



Energy

Figure 2: To ensure connectivity of single vertical path, each location in energy should be connected (i.e. added) to one of the three above locations… and so on

Required

Your task is to calculate the best vertical seam with the min total energy (min distortion) to be removed from the original image.

Task Details

For one "Seam" to be removed

1. Calculate energy for the image pixels [**DONE**]
2. Find the vertical path with minimum total energy [**YOUR TASK to implement**]
   `In class "ContentAwareResize.cs": public void CalculateSeamsCost(..)`
3. Remove the selected seam pixels from the original image to be cropped to smaller size [**DONE**]

Input

1. `int[,] energyMatrix`: 2D matrix filled with the calculated energy for each pixel in the image
2. `int Width, int Height`: Width and height of the image (# of columns & # of rows of image)

Output

1. `ref int minSeamValue: The min total value (energy) of the selected seam.`
2. `List<coord> seamPathCoord: List of points of the selected min vertical seam (In ANY order: Up to down or down to up).`
   `"coord": is a struct of 2 integers representing the row and the column of each value of the seam you will select.`

Testcases

- You will find sample images to test with them here. You can browse for them to use.

Template

- Code template

Appendix (The benefit of this resizing method in the real-life applications) [Optional to read]

Image resizing is a standard tool in many image processing applications. It works by uniformly resizing the image to a target size. However, this standard resizing does not consider the image content, leading to stretch/elongate the objects inside the image as shown in figure 3.

Original image (239×200) Standard resizing (159×200)

Figure 3: Elongation/Stretching effect of standard image resizing

On the other hand, Content-Aware Resizing can be used to effectively resizing the image while considering its content to avoid the stretching/elongation effects on the image objects, see figure 4 and compare with the result of figure 3.

Original image (239×200) Content-Aware resizing (159×200)

Figure 4: effect of content-aware image resizing, the dolphin is not elongated

Content Aware Resizing Idea

Content-aware resizing mainly uses an **energy** function to define the importance of pixels. The pixels with low energy are less important and can be eliminated when resizing the image. Figure 5 show the energy function of the Dolphin image, the pixels with large energy are white while those of small energy are

black. Observe that the Dolphin pixels are the most important pixels with the largest energy, so it should not be removed by the Content-aware resizing.

Figure 5: Energy function of the Dolphin image, white colors are high energies while black are low energies.

The content-aware resizing search for a **connected path** of **minimum total energy** pixels crossing the image from top to bottom, or from left to right, this path is called a **"Seam".** Figure 1 shows the minimum seam in the Dolphin image. By successively removing seams we can reduce the size of an image in one or both directions. Seam selection ensures that while preserving the image structure, we remove more of the low energy pixels and fewer of the high energy ones.

To ensure the connectivity of the pixels within a single path "Seam", each location should be connected to one of the three preceding locations as shown in figure 2.