**Documentation for Image Retargeting Matlab Code**

**General Flow of algorithm:**

1. Calculate gradient image.(only once for an image)
2. Calculate energy map (Forward energy or backward energy). This has to be after every seam removal.
3. Find the seam.
4. Remove the seam.
5. Go to step 2 and repeat the process till we reach desired resolution.

**Main Function:**

function [resizedImage] = ModifiedSeamCarving(ImgName, hor, ver)

Inputs: image name, number of horizontal seams to be removed, number of vertical seams to be removed.

Output: Resized Image.

Steps Followed:

1. Read image
2. Convert it to Gray Image.(Comments: its kind of optimization but not sure if its optimizing or adding more complexity. )
3. Calculate Gradient image (Comments: I guess we can use inbuilt functions also, but we have written a function to do this. Function is defined in the later section.)
4. Initialize the variables reducedinputimage, reducedgrayimage and reducedgradMean to input image, gray image and gradient mean. (comments: This is not necessary because we are not using input Image, gray Image and GradMean anywhere else. Helpful in debugging to cross check original and reduced image values)
5. Remove horizontal and vertical seams alternatively till minimum of horizontal and vertical seams (Comments: We just thought that it will be nicer if we remove alternatively, but we need to experiment on different ways to do it)
6. Change double reduced images to int. (Comments: This is not needed in C++ because you can define your variables as int, where as default double in matlab)
7. Plot images.

**Calculate Gradient Function:**

function [GradMean] = CalcGradient(inputImage)

Input: Input Image

Output: Gradient Image

Steps Followed:

1. Define sobel operators for horizontal and vertical gradients. One is transpose of another.
2. For all dimensions calculate horizontal and vertical gradient and then find magnitude of the gradient.
3. Take the mean of 3 dimensions to gradient mean image.
4. Comments: This is calculated only once.

**Remove Vertical Seam Function:**

function [ReducedInputImage,ReducedGradMean, ReducedGrayImage] = RemoveVer(ReducedGradMean, ReducedInputImage, ReducedGrayImage)

Inputs, Outputs: Reduced gray and input images and reduced gradient images.

Steps Followed:

1. Calculate forward energy.
2. Find Seam
3. Remove seams from all the 3 images and output reduced images.

**Remove Horizontal Seam Function:**

function [ReducedInputImage,ReducedGradMean, ReducedGrayImage] = RemoveHor(ReducedGradMean, ReducedInputImage, ReducedGrayImage)

Inputs, Outputs: Reduced gray and input images and reduced gradient image.

Steps Followed:

1. Horizontal is same as vertical seam removal if we transpose the image before calling the below mentioned functions.
2. permute(ReducedInputImage,[2,1,3]) gives transpose for 3D image and ReducedGradMean’ gives transpose for 2D image.
3. Calculate forward energy.
4. Find Seam
5. Remove seams from all the 3 images and output reduced images.
6. Transpose all the images again to get the original order.

**Calculate forward Energy Map Function: Algorithm in implemented in the similar way as explained below with checks for j=1 and j=cols. Costleft CostRight and CostVertical are the cost functions to calculate forward energy and explained clearly in the link mentioned.**

Calculating Energy Map : Backward Energy map is the sum of the current value at (i,j) from the gradient image and the minimum of the three neighboring pixels in the previous row, i.e. *min*((i-1,j-1),(i-1,j),(i-1,j+1)), from the energymap. For i=1 (the initial row), the values in the energymap image are set to those in the gradient image, and for when the pixel (i,j) is along the edge of an image, only (i-1,j) and either (i-1,j-1) or (i-1,j+1) are used depending on if (i,j) is on the right or left edges, respectively.

Forward Energy Map is the sum of these values plus some cost function. This cost function in the difference in gradient in left, right and vertical directions if we remove the pixel.

Reference : <http://people.csail.mit.edu/mrub/talks/SeamCarving_6.865.pdf>

Backward energy map : pages 20 to 28.

Forward enegy map : pages 69 to 76.

**Find Seam Function:**

Once the energymap is calculated, the method to find the optimal seam is to first find the minimum value in the last row (which becomes the (i,j)’th pixel), saving the pixel location for use in removal, then working backwards by finding the minimum of the 3 neighboring pixels of (i,j) in the (i-1)’th row and saving that pixel to the seam path. This process is repeated until the first row is reached, and results in the optimal seam.

**Remove Seam Function:**

After the optimal seam is found, the path of pixels that make up the seam are removed from gradient image, gray image and the original RGB image, and the remaining pixels are shifted left to form a continuous image.

I guess, this is the most costly function. We can try to optimize this as much as we can.

**Remove Seam from gradient image Function:**

This is same as remove seam function, expect we add importance diffusion when we remove seam. Importance diffusion is to add some cost to adjacent pixels when we remove a pixel. As of now we implemented it as addition of (removed pixel gradient/4) to adjacent pixels.

Comments: I am guessing that there is some mistake in doing this, because the results are so impressive. Have to check about this once again.