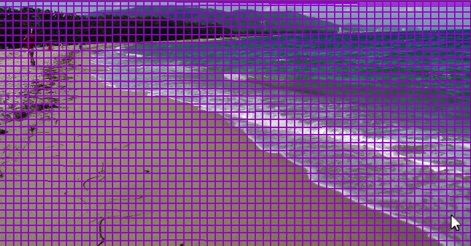
**IMAGE RETARGETING**

Involves resizing, recompositing, reshuffling and inpainting.

While resizing an image, maintaining the aspect ratio is of key importance. But in many cases it is impossible to do so. In such cases, the portions of image having lots of gradient action tends to be preserved with the same size, while not-so-perceptible regions are squished so that the image is resized to desired size.

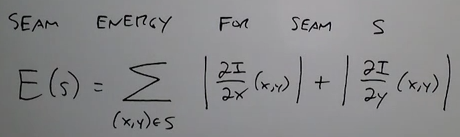




Another method is:

**Seam carving for resizing**

Consider reducing the width of the image by a pixel width. First we have to draw a seam, such that if that if al the pixels along that seam are removed, it should not be perceptible.

The idea is to have a cost function involving the gradients of the pixels along that seam. 

Example:

The same approach can be used to resize an image to bigger scale.

* Reducing image sizes -> removes seams
* Increasing image sizes -> adds seams

So we can add pixels of near average to the neighbouring pixels on both the sides of the seam.

Example:

**Seam Carving for Inpainting**

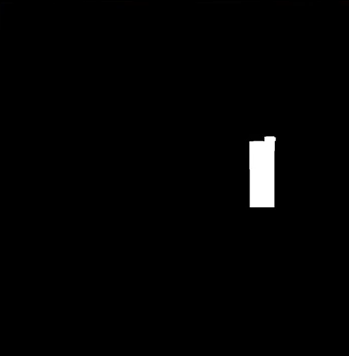
Consider an object in an image that has to be inpainted. Using the usual technique, sometimes it is impossible to fill it with the neighboring pixels values alone.

So in that case we remove ‘n’ number of seams that are **forced** to go through the object. It can be forced by using a cost function which is zero when passing through the object of interest.

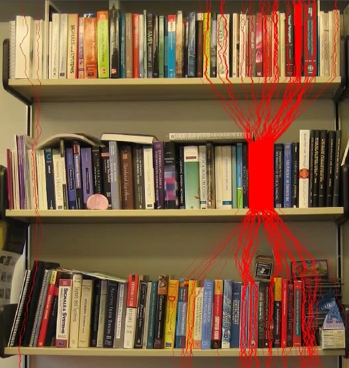
Then after removing, add ‘n’ number of seams to get back the original size.

Example:

Original image: Mask:

Seams: After removing the books:

To get original image add ‘n’ number of seams (these are different ones):



Final result of resized to original size:



Now conversely, in order to protect an object, make the cost infinite so that seams never pass through the object at all.

Now the original seam carving method looked at the backward energy. It is better to look at the forward energy: what new edges would be introduced should we remove this seam?

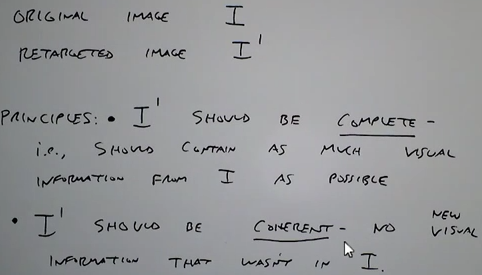
Original image: Resized image:

Notice that it is not resized as a whole, but patches of the image have been removed owing to repetitions.

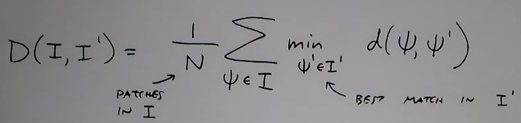
Another example: consider resizing an image having a building with 80 windows

The above can be done using ‘Bidirectional similarity’.



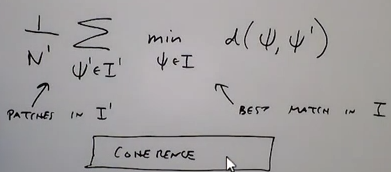
The cost function must pertain to both the above mentioned principles.

Cost function:



Looking at a bunch of patches in the original image, find the best corresponding patches in the new image and compute that average distance. That distance has to be small.

This is satisfying the ***completeness*** in the first principle.



This is the second principle namely ***coherence.***

So collectively:

