Jay Hennig
jah3728
PS1, CS 378

cats.jpg: 599x799

scaled:

seamed: -200w -> -200h = 399x599

In these photos, there are obvious flaws in both the seam-carving output and the scaled output. The scaled version seems to smush in the cats' butts, so they look unnecessarily stout. The seam-carved version, however, maintains the integrity of all information of interest *except* the poor white cat (his head has been severely compacted). The white cat's coat is simply too uniform in comparison with the noisy grain of the wood background, so the pixels in the cat are chosen to be removed over those in the background.

books.jpg: 510x416

scaled:

seamed: -50h -> -150w = 460x266

In this example, the seam-carving is a complete failure, even before comparison with the scaled version. The books are no longer straight (for some reason unaligned vertical seams were chosen within a book, leaving crooked book bindings), and the bottom shelf becomes bowed. The scaled version of this image works a lot better. This image shows the weaknesses of seam-carving when an image is saturated with important information—there is no room to remove pixels without it being obvious.

birds.jpg: 334x500

scaled:
seamed: -25w -> -25h -> -50w = 309x425

In this image I would say that the seam-carved output wins by preserving the size of the birds and making the image more dense. However, if one looks close enough there are a few places where the bird silhouettes are a bit uneven. Since the background is an overcast sky (as opposed to a smooth, clear sky) a few too many pixels might have been taken from the regular luminance of the bird silhouettes. This is a good example of the contrast between a scaled image and an intelligently compressed image: the scaled image is left to decrease the size of objects, which might change the overall feel of the image (the birds now look farther away), while the seam-carved image simply brings the birds closer together without losing much detail (except the distance between birds).

seals.jpg: 375x500

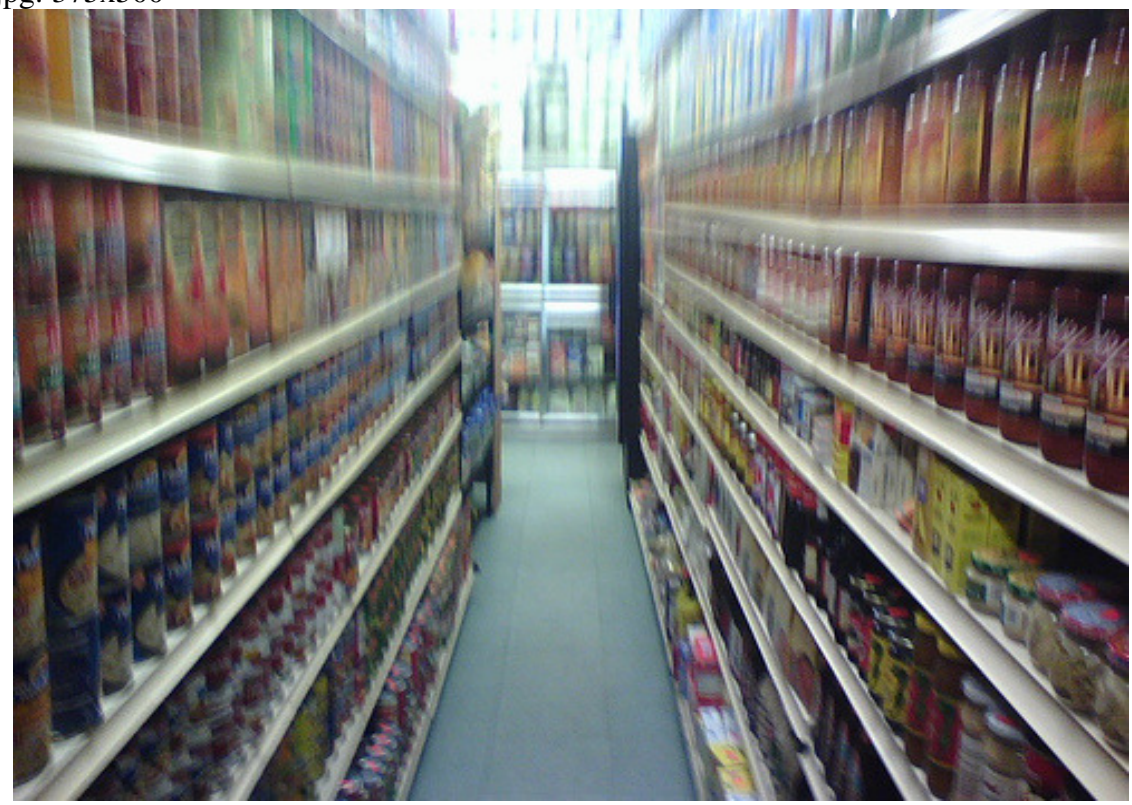
scaled:

seamed: -50w -> -30h -> -30w = 345x420

This image is another (subtle) win for seam-carving. Since the sky has a constant luminance, and is not necessarily important, seam-carving can remove this area almost entirely while scaling must downsize everything by the same amount. However, this may be a downside for images that are artistically composed (i.e. the 1/3rd rule of composition is destroyed). Also, since the seals are (to us) randomly scattered, seam-carving can move the seals almost unnoticeably closer to each other, while again scaling must downsize

everything.

groceries.jpg: 375x500



scaled:

seamed: -40h -> -60w -> -20h -> -40w = 315x400

This image is similar to the books image in that seam-carving has trouble with images saturated with important information and organized shapes. The aisles have crept unevenly close to each other, while the shelves become bowed. The scaled image, fortunately in this case, simply downsizes everything.

trees.jpg: 375x500

scaled:

seamed: -30h -> -50w -> -30h = 315x450

The effect of seam-carving in this case is not at first recognizable, though if this scene were familiar if would be immediately obvious. The trees on the right have lost a lot of density, and the shadows of some of the trees have been thinned as well.

Overall, the benefits of seam-carving are high as long as the scene is not recognizable or if the objects are relatively unorganized or sparse. Scaling must blindly scale everything down, which results in skewed objects if the downsize is not consistent in both dimensions, but has the benefit of working over a wider range of images (unless the seam-carving idea is supplemented by a bit more user input, such as regions of

importance, or skin-tones).

EXTRA CREDIT: #2
Removal of selected area of image: output = removeObject(im, a1, a2, b1, b2) where a1, a2 are the row, column coordinates of the top left corner of the desired rectangle to be removed; b1, b2 are the bottom right corner of the same rectangle; im is a uint8 image. To display the output image, simply use imshow(output).

The user specifies a rectangle in an image that he wishes to be removed. The first step is to calculate the shorter dimension of this rectangle, and to remove seams in that dimension so less seams will be removed total. After that, the energy function is altered to force seams to be chosen from the rectangle of interest. Assuming we are removing horizontal seams, if we change the energy function to “NaN”s for the pixels whose y-values are *not* in the rectangle of the, a seam will not be chosen going through this area and will thus be chosen from the rectangle. If we are removing vertical seams, the pixels whose x-values are not in the rectangle will be changed to “NaN.” In this way, we can enforce seams being chosen in our rectangle, and then it is simply a matter of removing seams until the rectangle is gone entirely.

In the photo above, the two birds in the upper left hand corner have been removed. However, the output is not exactly pretty (perhaps because of the noisy background?): the two birds closest to them have become skewed and unrecognizable. But the birds are gone!