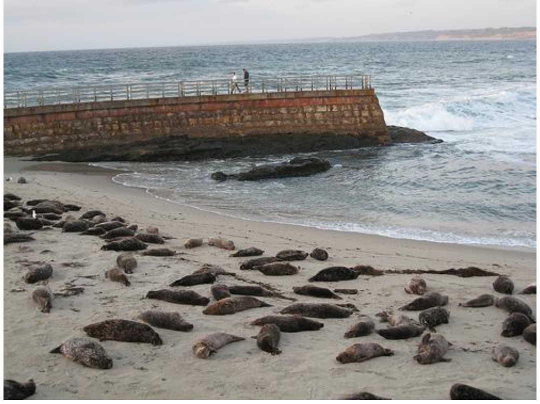
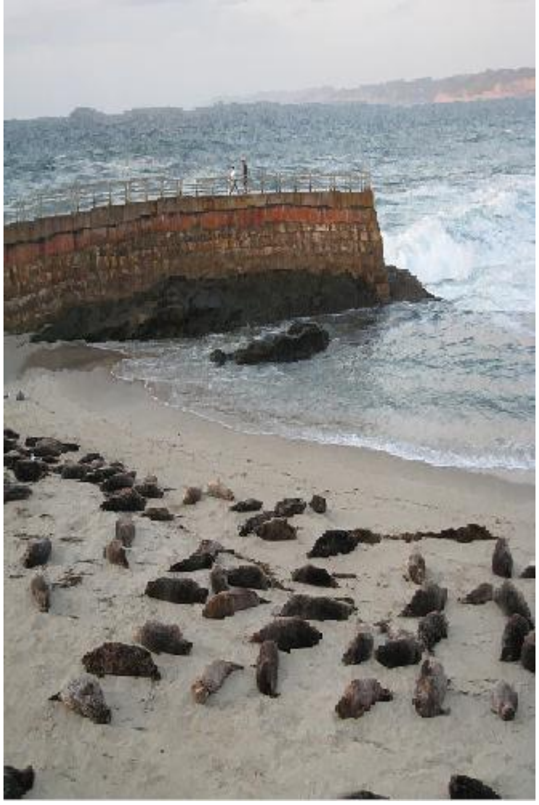
CS 378 Computer Vision Pset#1

CHI-MING YEH Unix Account: ychm

Part II :

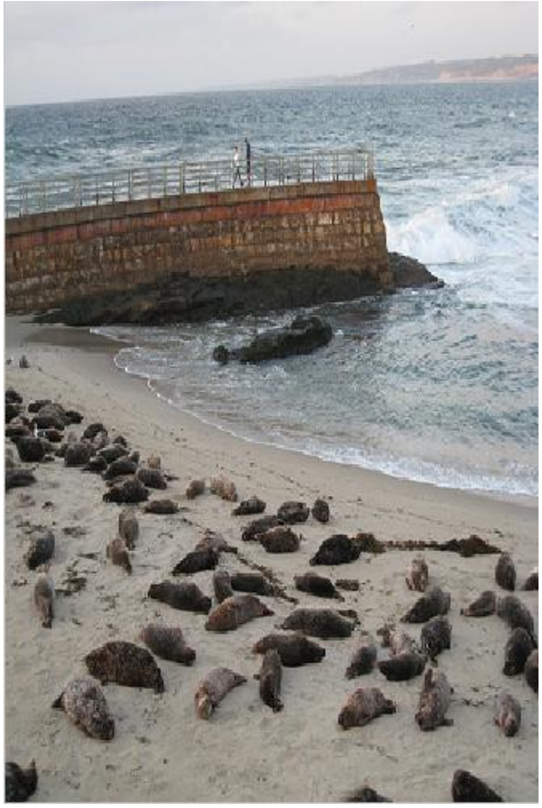


original input ‘seals.jpg’ 375x500



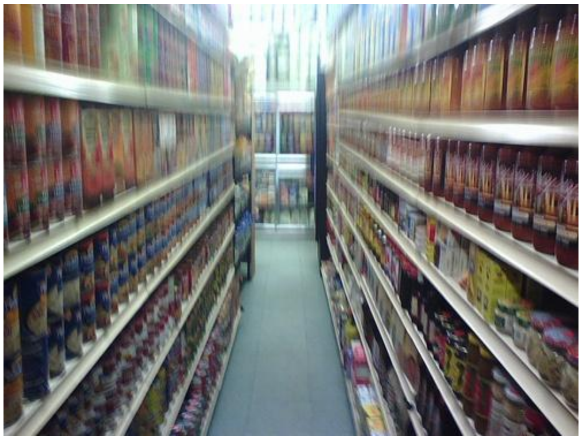
my system’s resizeIm 750x500

First doing the enlargement and then doing the removals. However, in this picture, there is no removals, only enlargement in height.

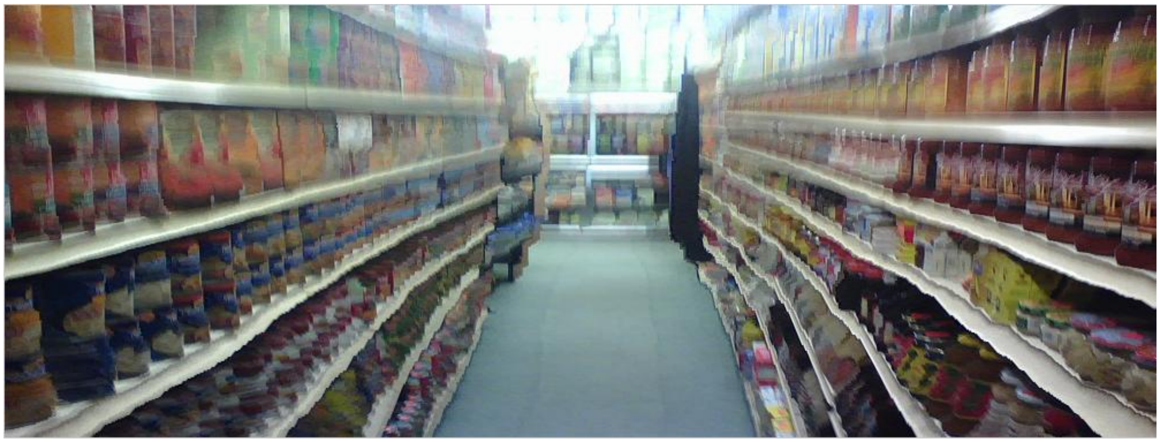


matlab’s imresize 750x500

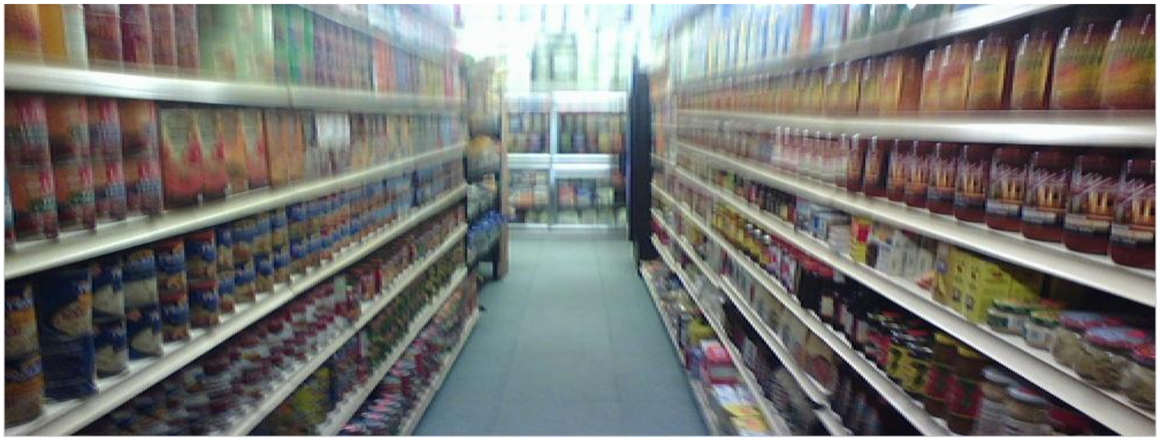
We can see the differences between my system and matlab’s imresize are in the two people. In my system’s result, because people will have edges with high energy, they won’t be picked to enlarge. Therefore, in the upper levee and the two people, they are still almost the same in the original picture. However, in matlab’s imresize, we can find out that the two people have been stretched and looks unrealistic.



original input ‘groceries.jpg’ 375x500

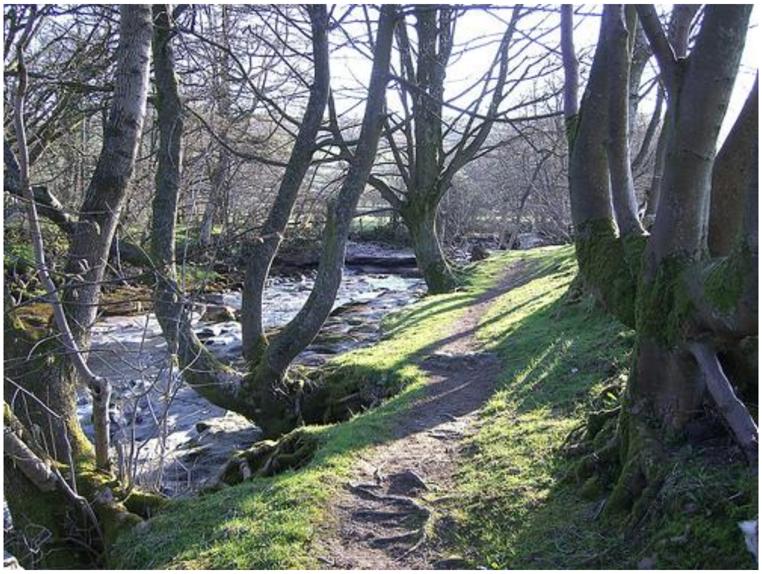


My system’s resizeIm 375x1000

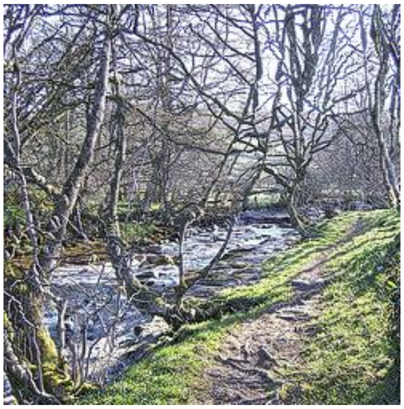


Matlab’s imresize 375x1000

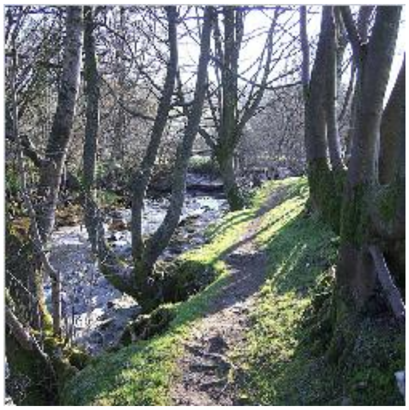
When my system tries to enlarge in width, it finds the most important parts with high edges. However, there are blurring areas in the middle of the original picture, which means they have low energy. Compared to matlab's imresize, my system's resizeIm will duplicate more pixels in the middle and cause distortion in the shelves.



original input 'trees.jpg' 375x500

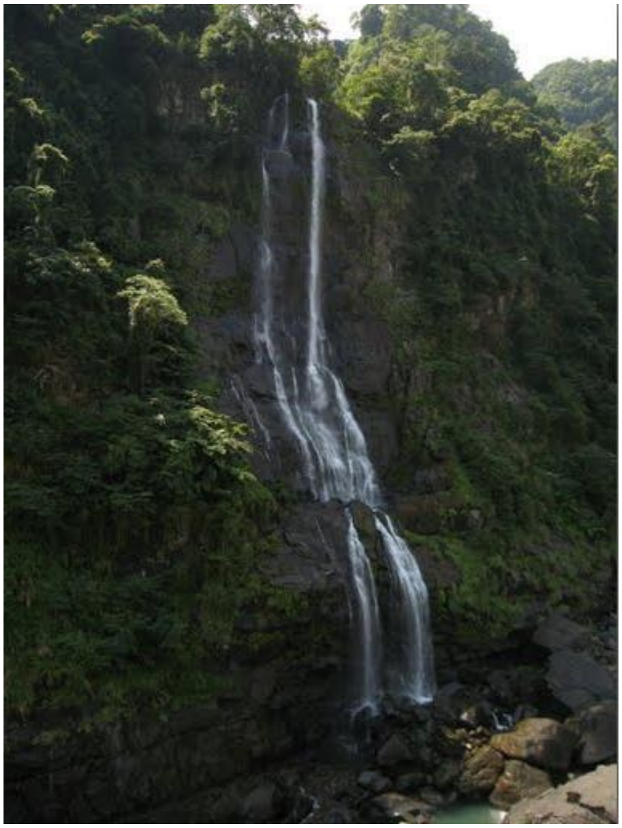


My system's resizeIm 250x250

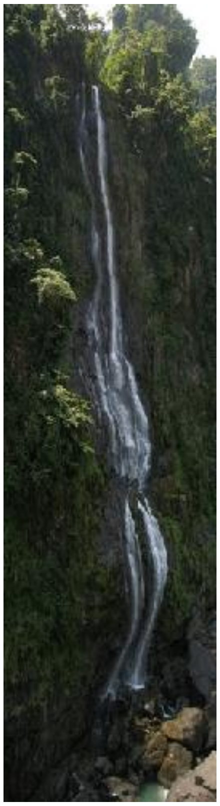


matlab's imresize 250x250

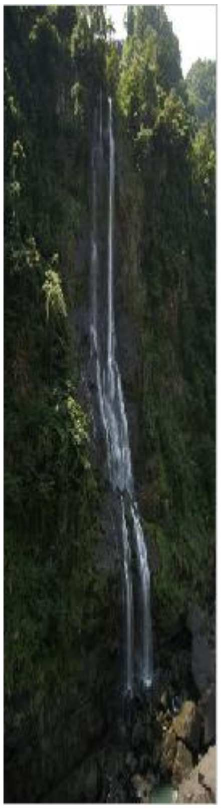
We can see that in my system's picture, it preserves more branches of the trees because they are edges with high energy. Thus, the trunks are thinner than the resized image of matlab. It depends on which kind of information you want to protect!



original input ‘waterfall.jpg’ 384x512



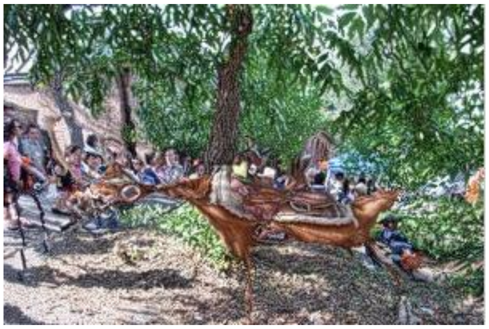
my system’s 750x200



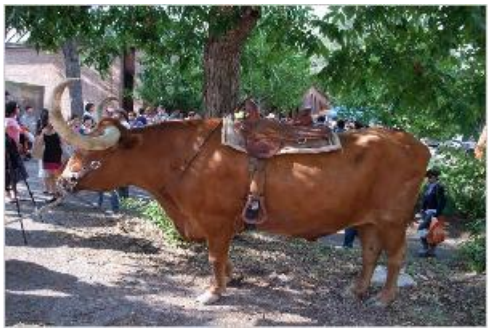
matlab’s 750x200

In this resizing, we first enlarge the height and then remove the width. We can see that in my system's resizing picture, it preserves most of the curves in the waterfall because they have more edges with high energy. However, in matlab's resizing picture, it just squeezes the whole picture and makes the waterfall looks straight.

original input 'longhorn'   480x640



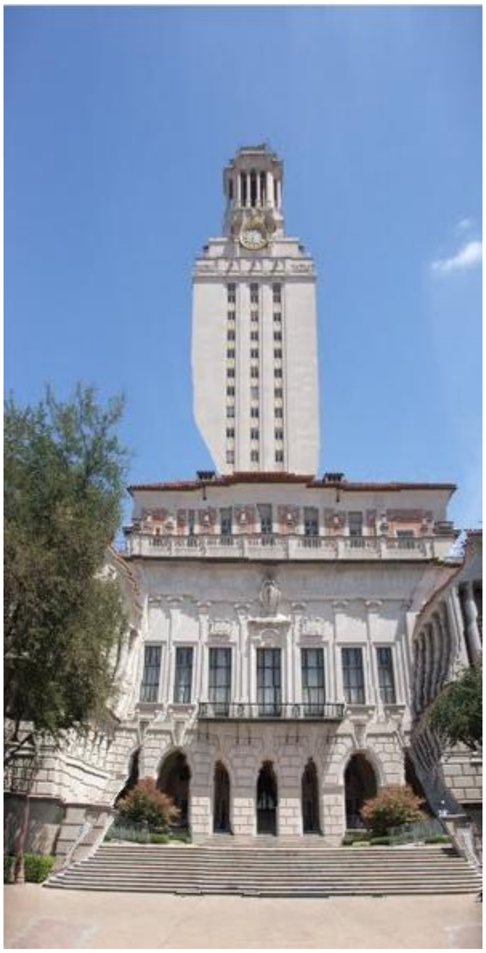
my system's resizeIm   200x300



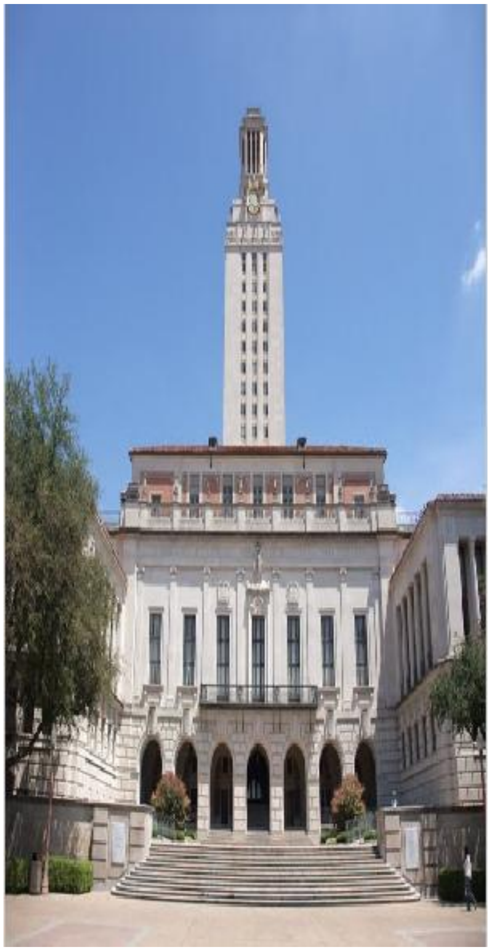
matlab’s imresize 200x300

This is a bad example for my system because it is plane and colorless inside the body of the longhorn, it disappears after we remove the pixels. We preserves the leaves of the trees because they have more edges. On the other hand, it still preserves most of the shape of the longhorn in the matlab’s imresize.

original input ‘tower’ 480x640



My system’s resizeIm 600x300



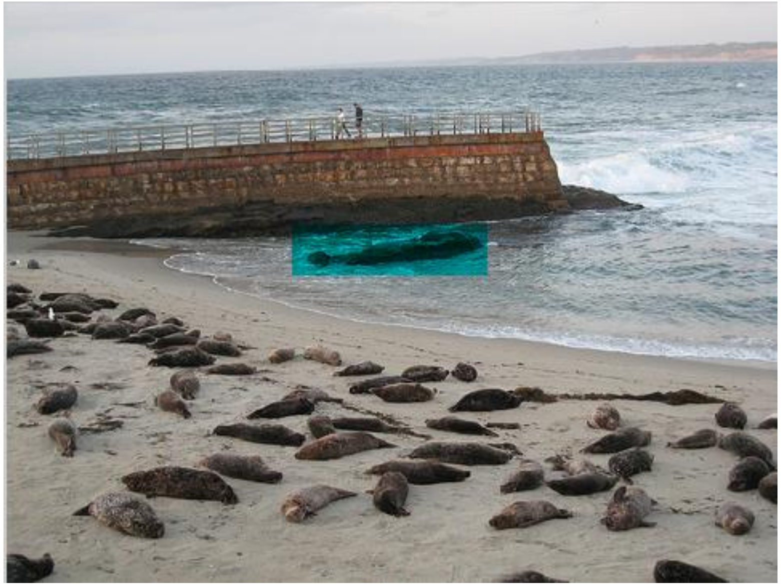
matlab’s imresize 600x300

In my system’s resizing image, it tries to preserves the original pixels of the tower. However, at some boundaries of the tower, they are mixed with some background pixels in the columns of the image matrix. When we accumulate the vertical path energy, some of the boundaries will have low energy and be cut by the program. This is the reason why there are some holes in the tower. We can find out that, by using this algorithm, the resized image will be sensitive to the shape if it is not very rectangular in their boundaries and then be distorted by the program.

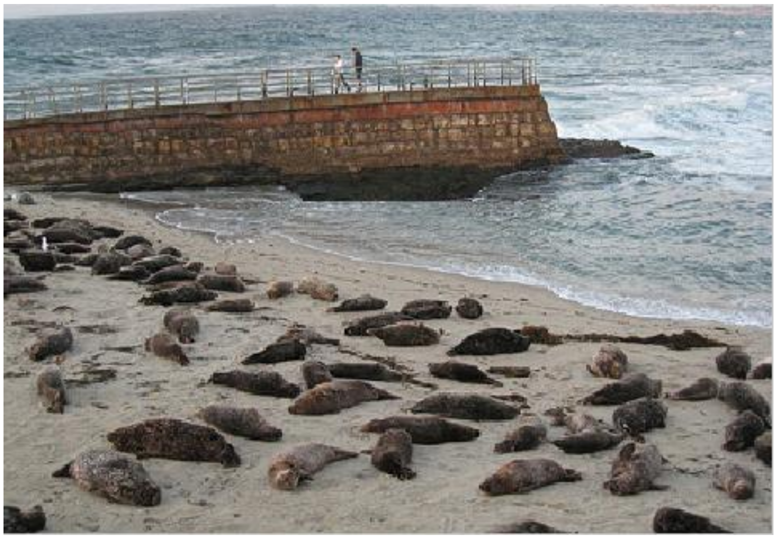
Furthermore, in my resizeIm function, it will first enlarge the image and then remove the pixels. When doing enlargement, instead of just picking one optimal seam and duplicating it, I find several seams at the same time. However, we should not pick too many seams in one step; otherwise, it will be the same as matlab’s resizeim function. We will check if we enlarge the image more than 50%, the process will be divided in several steps. This will protect the high energy pixels from be distorted.

OPTIONAL :

1. An user use its cursor to click four points in the original image, we will cut he rectangle area and show the new image compared with original one.



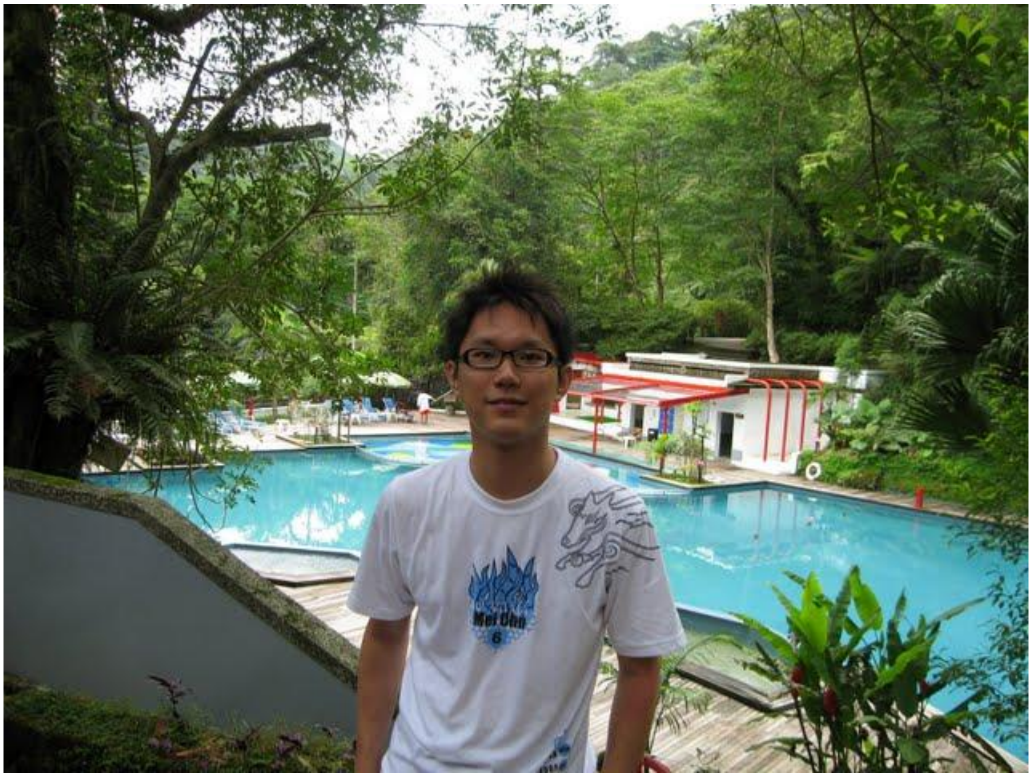
original image 375x500



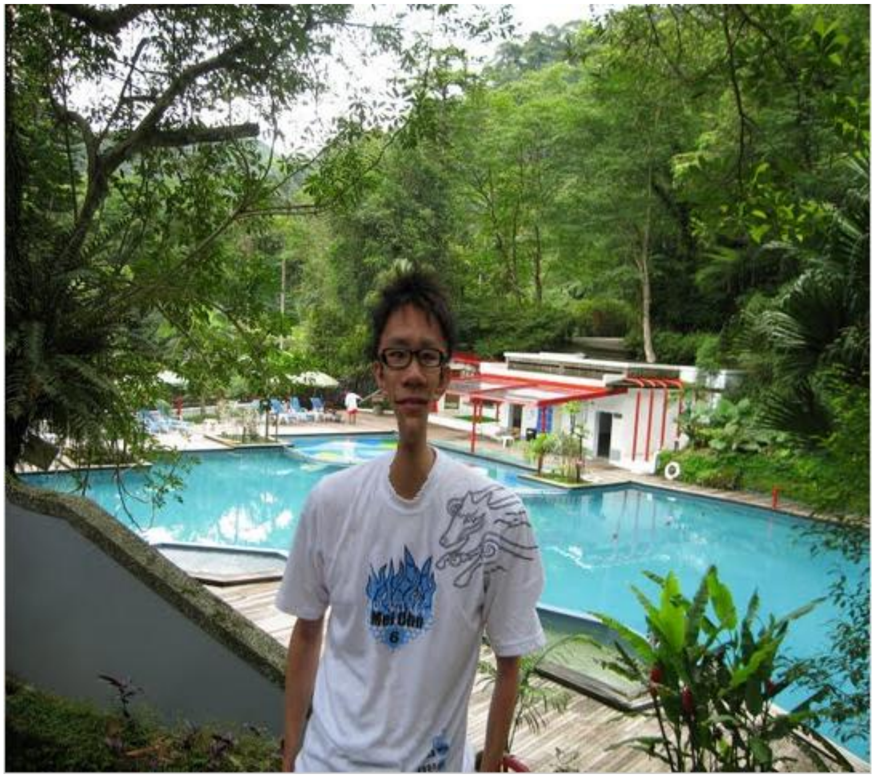
cut out image   304x429

We first use the ‘ginput’ to get four points clicked by the user. Then, inside the rectangle, we set all their energy to -100. Therefore, the selected area will be easily chosen by the program as the seam to cut. By using a while loop to check if there are still pixels with energy value as -100. If not, we know that we are done.

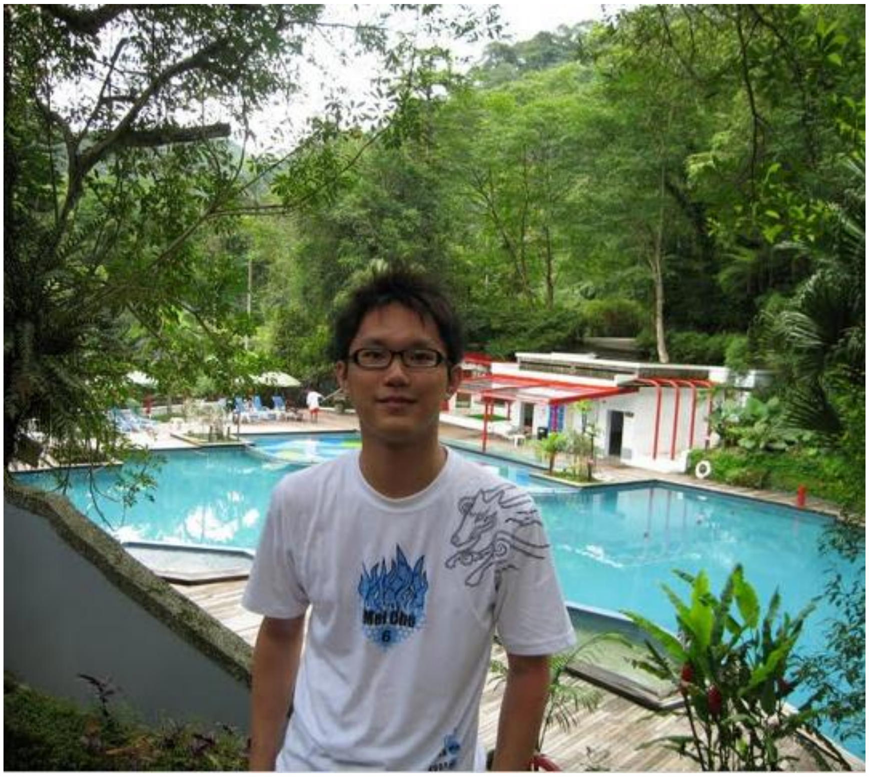
2. Using the HSV color space to protect human’s face.



original input ‘face.jpg’ 480z640



my system’s reduceWidth 480x540



Using hue enhancement to reduceWidth 480x540

We can see that in the original image, because of the colorless white t-shirt and the colorful background, my systm’s reduceWidth function will cut out the face. By using the command rgb2hsv, we can change to the HSV space. Also, we can find out that most of the human skin color is around 0 < hue < 0.13xxx. Besides, most of the background will be zero if it is plane and around 0.6 if it is colorful. Therefore, we just need to find the pixels with hue that is less than 0.13 and bigger than zero. Then, add its energy function to big enough as 1000. Finally, we will see that pixels with a skin color will be preserved by the reduce Width function.

Image acknowledgements: Thanks to the following Flickr users for sharing their photos under the Creative Commons license:
Seals.jpg is provided by allotrope
Trees.jpg is provided by russelljsmith
Groceries.jpg is provided by The Consumerist