Introduction

In this assignment, we'll be looking at 2-D signals in the form of images. The first task will look at some time-varying medical data for an MR angiogram (a type of medical image used to map the arteries). The second task will use image registration to stitch together images taken from nearby viewpoints.

Completing the assignment

This assignment can be done individually or in pairs (though we strongly encourage you to work in pairs). **Note:** work with a **different partner** from the previous assignments, unless you get permission from the instructor.

As before, Python is the preferred language, and all the supporting code to handle I/O and display that we're providing is in Python. For handling images, we'll be using the `opencv` library, available at `opencv.org`. They have a fairly comprehensive set of installation tutorials under their documentation page. We'll also be using a Python wrapper for the maxflow code commonly used in vision applications. It's available at `https://github.com/letterx/PyMaxflow.git`, to install on Linux or OSX, just

```
git clone https://github.com/letterx/PyMaxflow.git
cd PyMaxflow
python setup.py build
sudo python setup.py install
```

For submission, package up your code as a zip or tar file. Include your written answers as a pdf or doc file named `writeup.[pdf|doc]`. There's a bunch of graphs and images we want you to output, include these in your writeup, and please label them so we know which figures go with which subproblem. Please submit your code via CMS, so that we have it all in one place.

If you have any questions about this assignment, please contact the TA (`afix@cs.cornell.edu`).

Checkpoint

We'd like to make sure you get started on time (well before the end of classes) so please submit the following **on November 25th**.

1. Come up with an english-language description of your unary costs for part 1 below.
2. Describe and justify your choice of binning scheme for the mutual-information section, and the dilation distance for Hausdorff.

1 Segmentation

We've provided you with a sequence of images from an MR angiogram. These images are already properly registered, meaning that each pixel (say, the one at location (i, j)) always represents the same point in the subject, just at different times. An angiogram works by injecting a highly magnetic substance, called a contrast agent, which shows up very brighly on the MRI. As the contrast agent flows through the circulatory system, dfferent locations will show up more brighly at different times, mostly concentrated in the arteries.

We want to find a segmentation of the subject, so that we can map the arteries. A segmentation gives every pixel (i, j) in the 2-D grid a label $\{0, 1\}$, where 1 indicates an artery. There are many possible segmentations given the data, so we find the best one by solving a minimization problem. We have a variable x_k for each (i, j) in the grid, and x_k is 0 or 1. Each variable has a *unary cost* for being 0 or 1, denoted

$u_k(x_k)$. It will be up to you to decide what u_k looks like — we'll describe how to do this below. In order to get a smooth-looking segmentation, we also add costs for neighboring pixels being different, where in the grid, (i, j) and $(i + 1, j)$ are neighbors, as are (i, j) and $(i, j + 1)$.

$$\min_x \sum_k u_k(x_k) + \sum_{k,k' \text{ neighbors}} \lambda\{x_k \neq x_{k'}\} \tag{1}$$

where $\{x_k \neq x_{k'}\}$ is 1 if $x_k \neq x_{k'}$ and 0 otherwise.

For setting up the unary terms, we need to come up with a measure of how much we think a given location in the subject appears to be part of an artery. Because the contrast-agent hits different points at different times, a single image can't be used to construct the right unary term. You'll need to construct your own unary term: the unary term u_k should look at the entire time profile for the pixel (i, j) across all times t, and should give two costs $u_k(0)$ and $u_k(1)$ for being labeled 0 or 1 respectively. Some operations that you might consider in designing u: the maximum intensity over all times, the sum of intensities, the average intensity, the variation of the intensity over time.

You've been provided with code to compute the minimum of functions that look like (1). Your task for this assignment is (1) design a unary term to categorize the pixels, (2) using the provided code, compute and output the segmentation.

Include both the segmented image and a short description of how you designed u_k in your writeup. Note that you'll probably have to experiment with your choice of u to get good looking results.

2 Image registration

Our second task is image registration, which takes two images taken from nearby vantage points, and lining up the parts where they overlap. Our general method for doing this will be to look at all possible small translations between the two images, and try to compute some score judging the similarlity of the overlapping parts. We pick the best-scoring translation, and use that one to line up the two images.

To be a bit more precise: let k, l be integers in $[-20, 20]$. The overlap between images I_1 and I_2 if we translate I_1 by (k, l) are $I_1[k : n, l : m]$ and $I_2[0 : n - k, 0 : m - l]$, where n, m are the height and width of I_1, I_2.

We'll search over all pairs $k, l \in [-20, 20]^2$, and for each (k, l) compute the similarity of the overlap. We'll use 2 methods for computing similarity in the next two sections.

2.1 Mutual information

Mutual information is a measure of how frequently colors occur together in two images. We're trying to compare the two distributions of colors in each image. We will actually be looking at the *joint distribution* for the two images, and giving a measure of how simple this distribution is, called the *joint entropy*.

The joint distribution tells how frequently pairs of colors occur at corresponding pixels in the two images. We write it $p(c_1, c_2)$, meaning this is the number of times a pixel in I_1 has color c_1 and the correponding pixel in I_2 has color c_2, divided by the total number of pixels.

Joint entropy is a measure of how much information the two distributions share. For our purposes, we can compute it by

$$-\sum_{c_1} \sum_{c_2} p(c_1, c_2) \log p(c_1, c_2) \tag{2}$$

Joint entropy is low when the two distributions have similar information content. So, to find the best translation to register the images, we search over all small translations and pick the one with the lowest joint entropy.

One final detail, because we have to compute a probability $p(c_1, c_2)$ for all pairs of colors, if we use full-color images with 32 bits of colors, we'll have far too many pairs to work with. So, you should 'bin' the colors into (say) 16 bins first. You'll need to design a scheme for doing this as part of the checkpoint.

2.2 Hausdorff fraction

Another way to register images is to try to line up the edges present in both. You can think of this as trying to match shapes from one image to the other. There's a standard edge-detector used in many computer vision applications called the Canny edge detector. There is a function in opencv which implements this for you, which you should use.

The edge detector will return a 0/1 image, where pixels with value 1 have an edge present, and 0 pixels do not. We want to find a translation where the edge maps are as similar as possible. The natural measure for comparing 0/1 images is to sum up the number of times the values in the two edge maps agree. That is, if E_1, E_2 are the two edge maps, then we sum up the number of pixels where $E_1[p] = E_2[p]$.

One issue with this approach is that the edges in the two images may not quite match, because Canny does not always find every edge. So, to fix this, we will "dilate" the edge map by a few pixels. A dilation of an edge map means a pixel has value 1 whenever it is an edge, or it's within distance d of an edge. You'll need to choose a good dilation factor as part of the checkpoint.

The Hausdorff fraction is the percentage of the 1 pixels in the undilated edge map (from the 2nd image) that lie over a 1 pixel in the dilated edge map (from the 1st image).

Once you've computed the dilated edge maps, you can compute the Hausdorff fraction between these for corresponding image pairs. Note that you don't have to recompute these for every translation, you can just appropriately trim the results of the edge map for the untranslated pairs.

2.3 Your task

Implement both the joint-entropy measure, and Hausdorff fraction. Use these to find the best translation for each image pair, and each method. There are 3 image pairs: `reg-1-x.jpg`, `reg-2-x.jpg` and `reg-3-x.jpg`. We've given you a function to overlay two images on top of each other, so include the resulting images, and the values of the translations you found in your writeup.