Assignment 2. Segmentation via Deformable Models Methods
Passed out on 27 September 2018. Due on Tuesday 16 October 2018 Pizer

In this assignment you will be segmenting a collection of 2D slices of MR images of a brain structure called the corpus callosum. It appears in the image as with a bright intensity relative to its background. 32 of these images can be found in the files provided with this assignment. You are to apply two different deformable model segmentation methods to these images and answer questions about how application of a third would work.

Method 1: Geodesic Snakes. Find the ITK-SNAP program for doing geodesic snakes and the associated paper by P. Yushkevich. The input images for this part are provided in the file *corpus_callosum.zip*. SNAP takes as input only 3D images. For each of our input 2D images, we have created a 3D version consisting of 7 slices, with the middle three slices being identical copies of our 2D image, and the first two and the last two slices being a uniform image with intensity equal to the mean of the intensities of the given 2D image. SNAP shows the middle slice (which is a copy of our original image) by default. Try to do the segmentation on this slice.

Besides the input image SNAP requires an initial object contour, which it then deforms. For each image make your input contour be a small circle entirely within the corpus callosum. See how the method works by displaying the resulting boundary superimposed on the target image. Pay attention to how far on average the result is from the boundary you see in the image.

Method 2: Active Shape Models (ASM)

In the data set you are provided each image will have an associated list of 64 (x,y) values for corpus callosum boundary points for that image. These points are in correspondence across the cases. The point tuple for each object will be referred to as its PDM.

You will need to find a shape space for these PDMs. To do that, you will first have to translate and rotate each object's point set so that the point sets (tuples) are normalized in orientation with respect to each other. Then you will need to use PCA to generate the shape space. The following will explain how to do each of these.

2a, Normalization in translation and orientation

Do the following for each object's point set: Compute the mean of its points, i.e., their center of mass, and subtract the mean from each point. Then using the resulting points (x_i, y_i), compute the 2 ×2 2nd-moment matrix (with values of $\Sigma_{\text{points } i}$ of $x_i\, x_i$, in the upper left element, of $x_i\, y_i$, in the lower left and upper right elements, and of $y_i\, y_i$, in the lower right element), and compute its 2 eigenvectors and 2 eigenvalues. Create a matrix whose first column is the eigenvector with the larger eigenvalue and whose second column is the eigenvector with the smaller eigenvector. This matrix is a rotation matrix; you will need its inverse, which is its transpose, to apply to each boundary point in the point set. The result is now normalized.

2b. Computing the shape space

Pick a random subset of ¾ of the cases; these will form 24 training PDMs. The remaining 8 will form 8 test images (none of which have one of the training PDMs as their segmentation). Compute the mean, over the cases (not over the points), of each point in the sets. The resulting point set will be referred to as the mean object PDM. Subtract the mean object PDM from each object's point set. Then do PCA on the result. This will yield 64 eigenvectors $\mathbf{v}_i$ with corresponding eigenvalues σ_i^2.

Plot the eigenvectors in decreasing order, and choose a number, K, no greater than 6, of eigenvectors that have the dominant eigenvalues. These eigenvectors will be a basis for your shape space. That is, every point tuple, after mean object PDM subtraction, can be written as $\Sigma_{i=1}^{K}\, a_i\, \sigma_i \mathbf{v}_i$, and zeroing the a_i for i>K will be a vector in the shape space basis. Of course, to compute the coefficient $a_i\, \sigma_i$, you will dot product the point tuple with the ith eigenvector.

Finally, the shape space will consist only of coefficient factors a_i that are in [-2.5, +2.5], i.e., no more than 2.5 standard deviations from the mean in that dimension.

2c. Projecting a PDM into the shape space

In the active shape method, at each stage of the iteration you will have a PDM that is not in the shape space and you will need to project that PDM onto the shape space. To do so, you will need to first subtract the mean object PDM and then compute its a_i coefficient factors for i=1, 2, .., K. For every a_i which has a negative value less than -2.5, you will set that a_i value to -2.5 (put it in the shape space). For every a_i which has a positive value greater than +2.5, you will set that a_i value to +2.5 (put it in the shape space). You will leave those of the K coefficients that are within [-2.5, +2.5] unchanged. The projected PDM will be
the mean object tuple + $\sum_{i=1}^{K} a_i \,\sigma \mathbf{v}_i$.

2d. Shifting the point tuples in an iteration:

The active shape model method for segmentation iterates a 2-step process in which at the first step each point in a boundary PDM is moved along its normal to a position optimized for geometry-to-image match and in the second step the result is first shifted by the change in the PDM center of mass and rotated by using the 2nd moment matrix of the new PDM. Then the tuple of differences from that center of mass is projected into its shape space.

The point shifting will require you first to compute a normal direction at each boundary point. For the ith point consider the line segments from that point to the (i+1)th mod the number of points and from that point to the (i-1)th mod the number of points. For each line the coordinate difference duple is the form (a, b), so its unit normal is of the form $(-b, a)/(a^2+b^2)^{½}$. Average the unit normals for the two line segments to obtain the normal you should use for your point i.

The segmentation geometry-to-image match value should be the directional derivative of intensity in the reverse-normal direction (reverse because the target object is light on a darker background). That is, if the normal is $\vec{u}$, the geometry-to-image match (which is being maximized) is $-\vec{u} \cdot \nabla I$. with the gradient taken according to an appropriate scale using the code you used in assignment 1.

Along each normal you should be taking the arg max of this directional derivative over a sequence of 11 points, namely the point at the present point position, 5 equally-spaced points along the normal in the direction going outward from the boundary and 5 points with the same spacing along the normal in the direction going inward from the boundary. Use a spacing that is 1 pixel-width in length (but because it is normally not in a horizontal or vertical direction, it the positions will not normally be at pixel centers).

Computing the intensity gradients at locations that are not pixel centers will require a special computation. What would be best is to evaluate the analytic derivative of Gaussian applied to the image data that would come from the approach used in Derivative.m from the previous assignment. However, that would require rewriting the Derivative.m code. So instead, for each point at which you need to evaluate the intensity gradient, find the 4 surrounding pixel centers, compute the gradients $\underline{g}$ at each, and use bilinear interpolation to your point from them to compute the gradient you need. Bilinear interpolation from four pixel centers (j,k), (j,k+1), (j+1,k), and (j+1,k+1) [all of these in normal Cartesian coordinates, not matlab indices) to your point $(j+\Delta x, k+\Delta y)$ is given by $\Delta y\,(\Delta x\, \underline{g}(j+1,k+1)+(1-\Delta x)\, \underline{g}(j,k+1)) + (1-\Delta y)(\Delta x\, \underline{g}(j+1,k)+(1-\Delta x)\, \underline{g}(j,k))$. You are welcome to use the bilinear interpolation function in matlab.

Evaluation

For each method you will have a result for each test case. Visually compare these results, and report your conclusion.

The data locations and formats

The data, which is on the sakai site for this assignment, consists of four arrays, named *greyimages*, *binaryimages*, *pdms,* and *correctpdms*. *greyimages* is a $256^2 \times 32$-element array containing the 32 test and training images. For every value between 1 and 32 of the second index you get a 256×256 image stored in row-major order as a single column in the *greyimages* array.

Similarly, *binaryimages* is a $256^2 \times 32$-element array containing the 32 correct segmentations of 32 test and training images. Each is a 256×256 image stored in row major order as a single column in the *binaryimages* array. In that array, the value zero indicates a pixel that is not in the corpus callosum, and the value 1 indicates a pixel that is in the corpus callosum.

pdms is a 128×32-element array containing the translation and rotation (but not scale)- normalized pdms that you will use for training. You will not use the *correctpdms* is a 128×32-element array containing pdms in image coordinates..

Each array has the 32 cases in the same order.

To display the ith image in one of the image arrays, you can use the matlab command:

```
figure,
greyimage =reshape(greyimages(:,i),256, 256); /*or binaryimage  and binaryimages, respectively
imagesc(greyimage), colormap('gray');
```

To display a pdm (without care for reflections and correct scaling of the coordinates) use

```
pdm=pdms(:,i);
plot( pdm(1:2:end), pdm(2:2:end), 'b.');
```

3. Active Appearance Models Segmentation (AAM)

Assume each training image has been provided with a point tuple of points $\underline{\mathbf{w}} = (\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n)$ scattered in the image and in corresponding locations across the cases. After location and rotation normalization, these points provide the geometric representation. Answer the following questions about how AAM would be trained and applied.

3a. The geometry training will involve mean computation, yielding $\underline{\mu}_g$, and PCA (over training cases) on the mean-normalized $\underline{\mathbf{w}}$ values from the training images. This PCA will yield eigenvalues γ_i and eigenvectors u^i. A particular $\underline{\mathbf{w}}$ will be represented by a tuple $\underline{a}$; say that tuple has m_a entries. Given $\underline{a}$, what value of $\underline{\mathbf{w}}$ will be implied?

3b. The intensity to geometry match training will involve mean computation on the N intensity-pixel tuples $\underline{I}$, yielding $\underline{\mu}_I$ and PCA (over training cases) on the mean-normalized intensity pixel tuples $\underline{J}$. This PCA will yield eigenvalues ι_i and eigenvectors $\underline{v}^i$. A particular $\underline{I}$ will be represented by a tuple $\underline{b}$; say that tuple has m_b entries Given $\underline{b}$, what value of $\underline{I}$ will be implied?

3c. Analysis of the relation among scores $\underline{a}$ and $\underline{b}$ from each training case will involve PCA on its $\underline{a}$ value concatenated with its $\underline{b}$ value. This PCA will yield eigenvalues β_i and eigenvectors $\underline{h}^i$. A particular a,b pair will be represented by a tuple $\underline{c}$; say that tuple has m_c entries. Given $\underline{c}$, what value of $\underline{a}$ will be implied?

3d. The regression will map the intensity residue $\Delta\underline{I}$ (a column vector) into $\Delta\underline{c}$ (one column vector with each row representing a feature) by a matrix multiplication. Thus the matrix applying the regression will be computed from the two matrices. How many entries will there be in each the columns of the regression matrix?

3e. In each iteration the present tuple $\underline{c}$ will be displaced to a new tuple $\underline{c} + \underline{\Delta c}$. What will be the new value of $\underline{b}$?

3f. The displacement from $\underline{\mathbf{w}}$ to $\underline{\mathbf{w}} + \underline{\Delta \mathbf{w}}$ will be computed from the modified values of which eigenmode coefficients? Will that be applied to the reference image or to the target image?

3f. The intensity change from $\underline{I}$ to $\underline{I} + \underline{\Delta I}$ will be computed from the modified values of which eigenmode coefficients? Will that be applied to the reference image or to the target image?