Comp 766 Assignment 1

Jonathan Pearce 260672004

July 23, 2020

Problem 1b. To draw a circle we need the coordinates of the center and the radius of the circle. These are the parameters required to draw a circle in MATLAB as well for example. Using this information we can exploit the radial symmetry of circles in order to simplify the equations from part (a). In part (a) the equations involved all the pixels/voxels of each image, in this simplified case we should be able to solve the problem with only three pixels/voxels.

Assume that the center of the circles do not sit on the origin of our coordinate axes and that we do not already have the coordinates of the circle centers. This is a reasonable assumption as most image processing tools will have the origin of the coordinate axis in your image corner so that indices are always positive.

Without loss of generality take binary image X (denote the other image Y). In image X we can find the left and right most pixel in the circle, this can be done by looking at every pixel in the image (brute force), however there may be faster techniques, for example a modified binary search. let $X_l = (i_l, j_l)$ be the left most pixel and $X_r = (i_r, j_r)$ be the right most pixel. Let $X_c = (i_c, j_c)$ be the circle center, we have

$$i_c = \frac{i_l + i_r}{2}$$

$$j_c = \frac{j_l + j_r}{2}$$

Therefore because the problem stated that X and Y had circles that were aligned the circle center coordinates in the average image A will be the same as the coordinates from the X and Y ($A_c = X_c = Y_c$)

We will compute the halfway transformation of the right most pixel X_r. From the equation(s) in part (a) we have,

$$A_r = h(X_r, 0.5) = X_r + \int_0^{0.5} v(h(X_r, \tau), \tau) d\tau$$

We now have the coordinates of $A_c = (i_c, j_c)$ and $A_r = (i_r^A, j_r^A)$. The radius of the average circle r_A follows directly,

$$r_A = i_r^A - i_c$$

We now have the radius and center coordinates for the average binary circle and are therefore capable of producing the average binary image A.

An interesting note is that in this solution we do not ever leverage any information from image Y, we only deal with image X and the diffeomorphic transformations that map X to Y. If we were to make use of Image X and Y to solve this problem we would not require the diffeomorphic transformation instead we would compute A_r as follows,

$$A_r = \frac{X_r + Y_r}{2}$$

Problem 1c. This problem is a more generalized version of part b. First we solve for $X_c = (i_c, j_c)$ in the exact same way as part (b). Now we will compute the halfway transformation of X_r and X_c. From the

equation(s) in part (a) we have,

$$A_r = h(X_r, 0.5) = X_r + \int_0^{0.5} v(h(X_r, \tau), \tau) d\tau$$

$$A_c = h(X_c, 0.5) = X_c + \int_0^{0.5} v(h(X_c, \tau), \tau) d\tau$$

We now have the coordinates of $A_c = (i_c^A, j_c^A)$ and $A_r = (i_r^A, j_r^A)$. The radius of the average circle r_A follows directly,

$$r_A = i_r^A - i_c^A$$

We now have the radius and center coordinates for the average binary circle and are therefore capable of producing the average binary image A.

Problem 2. L is a differential operator that ensures smoothness. As shown in lecture 3 Slide 65, L should ensure that Green's function is continuous in the coordinate system axes and that the matrix K function is positive definite as an operator. A diagonal matrix of laplacian operators with identities satisfies these requirements. The use of Equation (1) is more generally justified in [1], it is a smoothness term that guarantees sufficient differentiability to insure the existence and uniqueness of the solution of the ODE associated with the diffeomorphism.

The laplacian of a function $f : \mathbf{R^3} \to \mathbf{R}$ is expressed as follows:

$$\Delta^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$$

In other words the laplacian computes the sum of second derivatives. Therefore L computes the laplacian of each entry of the 3x1 vector $v(t)$ [2] (Page 145) and we get,

$$Lv(t) = [\Delta^2 v_1(t), \Delta^2 v_2(t), \Delta^2 v_3(t)]^T$$

It follows that as $v(t)$ varies from a smooth vector field to a perturbed one (1) increases. The idea here is that we want our diffeomorphic transformations to be smooth and optimal as opposed to erratic and/or unnecessarily long.

Problem 3a.
Example 1 : This is for image segmentation. Let z_i be the observed pixel intensity of pixel i.

Unary Potential:

$$\phi(x_i, z_i) = (z_i - \mu(x_i))^2$$

Where $\mu(x_i)$ is the current average pixel intensity for class x_i. This term encourages pixels of similar intensity to cluster together into the same class.

Interaction Potential:

$$\psi(x_i, x_j) = e^{n(x_i)}$$

Where n is a function that counts the number of pixels j that are tangent to i with $x_j \neq x_i$. This function measures whether a pixel is currently estimated to be in a region interior (0 tangent pixels with different classes), on an edge (1 pixel tangent with different class), at a corner (2), end of line (3) or simply a single pixel inside another classes region (4). This formula encourages regions of the same class, and severely penalizes outlier pixels.

Example 2 : This is for image de-noising, assuming the scheme used for applying noise to pixel intensities was a Gaussian with mean zero and variance σ^2. Let x_i be the pixel intensity of our current image estimate. Let z_i be the observed pixel intensity in the original noisy image. Let σ^2 be the variance of pixel intensity.

Unary Potential:

$$\phi(x_i, z_i) = \frac{(x_i - z_i)^2}{2\sigma^2}$$

This potential tries to keep the difference in pixel intensities at pixel i between the original noisy image and the de-noised image close together. Since Gaussian noise was added, a small change is more likely than a large change and this unary potential reflects that. The $2\sigma^2$ term ensures that we make use of that that we know the underlying noise transformation was Gaussian.

Interaction Potential:

$$\psi(x_i, x_j) = \mid x_i - x_j \mid$$

This measures the absolute difference in pixel intensity between pixel i and j. This potential encourages pixel that are close to each other in the image to have similar intensity values and penalizes according to the variation of intensity values in the image.

Example 3 : From Boykov and Jolly [3]. z_i is the observed pixel intensity for pixel i. x_i is the annotation of the pixel i as either object or background. The goal of this paper was to develop a Markov random field model that could separate objects and background in an image.

Unary Potential:

$$\phi(x_i, z_i) = R(x_i, z_i)$$

Where R is a function that takes the annotation of a pixel and applies the pixel intensity value to a custom penalization function for that class of annotations. In the paper they give an example of a histogram as a known intensity model for background and object pixels.

$$\phi(x_i, z_i) = Hist_{x_i}(z_i)$$

A pixel with intensity that fits normally into its annotation class' histogram will have low energy, where as a pixel with intensity that is an outlier in its annotation class' histogram will have a high energy.

Interaction Potential:

$$\psi(x_i, x_j) = \frac{1}{d(i,j)}\delta(x_i, x_j)$$

Where $\delta = 1$ if $x_i \neq x_j$ and $\delta = 0$ otherwise. Immediately if the annotation of two nearby pixels is the same then the interaction energy is 0. Otherwise, the energy is $\frac{1}{d(i,j)}$, where $d(i, j)$ is the distance between pixel i and j. This term is inversely proportional to distance, therefore pixels with different annotations close together will incur a larger penalty than pixels that are far away from each other. The purpose of this potential is too encourage smooth annotation regions.

Problem 3b. My answer below was assisted by the work of Glocker, particularly [4] [5].

It is easy to see the similarity between equation (2) and the energy minimization that we deal with in markov random fields, therefore it seems likely that an appropriate markov random field scheme could accurately model equation (2).

The obvious first idea is to simply have a dense markov random field with a random variable for every pixel i in the source image, however it quickly becomes evident that producing a solution with this setup is not computationally feasible. Glocker proposed a dimensionality reduction technique in which a set of k control points are selected, where k is much smaller than the number of pixels in the source image. Each control point is given a random variable in the markov random field, and we aim to solve for the optimal

displacements of these controls points for this registration task. Glocker's proposition was that the displacements of the pixels that are not selected as control points can simply be expressed as a linear combination of the control point displacements,

$$D(i) = \sum_{j=1}^{k} \hat{w}_j(i)\gamma_j$$

Where $D(i)$ is the displacement of pixel i. $\hat{w}_j(i)$ is the weight given to the jth control point displacement as a function of the location of pixel i, distance is a natural fit for this weight function, closer control points should receive more weight. γ_j is the total displacement of the jth control point. With this simplified problem setup we simply need to find the optimal displacements for the small set of k control points and the remaining pixel displacements will follow directly. Next we will define our potential energy terms. With the observed variable being pixel intensity, denoted $I(p)$ for control point p of image I. The hidden random variable being the displacement update x_p. Also the model parameter(s) is the current displacement estimate for all pixels in the image.

Unary Potential:

$$\phi(x_i) = \sum_{p \in \Omega} \hat{w}_i(p) \mid I(p + \delta_p + x_i) - J(p) \mid$$

Where Ω is the neighbourhood of pixels around control point i. An important implementation note is that we should utilize domain knowledge to make Ω as small as possible while still being large enough to capture the largest displacements between images I and J. $\hat{w}_i(p)$ is the weight function for control point i relative to pixel p. Finally we have the displacement evaluation, where $I(p + \delta_p + x_i)$ is the observed pixel intensity in image I at pixel $p + \delta_p + x_i$ and $J(p)$ is the observed pixel intensity in image J at pixel p. Where δ_p is the current estimate for the displacement of pixel p and x_i is the displacement update for control point i. δ_p is a function of every control point displacement however in the unary term we only have access to the displacement of control point i and therefore this displacement evaluation is our best approximation.

Interaction Potential:

$$\psi(x_i, x_j) = \frac{\mid x_i - x_j \mid}{\mid i - j \mid}$$

Where x_i and x_j are the displacements updates for control points i and j. This term penalizes variations between the displacement updates of neighboring control points i and j. It is a discrete approximation of a gradient penalty. This interaction potential favors smooth updates for the displacement field.

Finally we must address how to ensure this markov random field will guarantee not only successful registration but in fact successful diffeomorphic registration. Glocker addresses this in [5], where he states "we can simply restrict the maximum displacement to be 0.4 times the control point spacing. Thus, every morphing will fulfill the diffeomorphic properties, and since we compose the single morphings on the image level and the composition of two diffeomorphisms produces a diffeomorphism, our final solutions are diffeomorphisms as well." The key point being that the neighbourhood Ω has to respect the control point spacing to a factor of 0.4. Therefore using our domain knowledge of the approximate maximum displacement between our two image I and J, we know how large the neighbourhood of pixels Ω must be in order to capture all control points displacements. Then using this neighbourhood size and the condition provided by Glocker above that relates maximum displacement of control points to control points spacing. In theory the choice of k and the selection of the k control points should be easier then before and will ensure diffeomorphic registration is satisfied.

Problem 4. Overall the implementation process was challenging and eventually successful. In my code I have implemented Equation (9) from the paper, which accounts for Equations (1) to (8) in a joint probability distribution. I have implemented Equations (10) and (11) to complete the E-step of the algorithm. Finally I implemented Equations (12) - (17) to get the parameter updates as part of the M-step. Overall the implementation of these equations went quite well. These equations are implemented in 'HMRF˙EM.m'.

I did not implement biological constraint 1 ('Conditional Dependencies on Tumor Occurrence') since we

only had one channel (C=1) of tumor image and therefore the exponential factor did not create a computational issue like it would in the case where (C=3). I did not implement biological constraint 2 ('Hyper- and Hypo-Intense Tumor Structures') since I was not sure whether a tumor imaged in a CT scan should be considered Hyper or Hypo intense. I implemented biological constraint 3 ('Spatial Regularity of the Tumor Prior'), with $\beta = 0.1$, I did not compare model performance across different β values.

The more difficult part of the implementation process was deciding how to initialize different parameters, specifically π_{ki}(Equation (1)), α as well as μ and σ. The initialization of α was the first one I solved and the only one of these that I was sure was correct. The authors discuss different initial values for α at the very end of page 10 of their paper. I choose $\alpha = 0.01$ for initialization, I did not compare model performance across different α initialization values. In order to initialize μ and σ there were two choices that I tried, the first being utilizing the kmeans estimates from the tumor image, these estimates were usually pretty close to the final values and provided quick convergence. However in the paper I did not see any notes on how to initialize μ and σ so I also tested out a uniform initialization scheme to help evaluate the correctness of my code, here I choose $\mu_i = 120$ and $\sigma_i = 15$ for all i, 120 is roughly in the middle of the intensity spectrum $[0, 255]$ and 15 seemed to be roughly what the standard deviations were converging to in other tests. The initialization of π_{ki} for the case where $k = 3$ was simple, we had 2 healthy tissue classes and therefore each pixel i either had a distribution $[0, 1]$ or $[1, 0]$ depending on how kmeans classified the pixel in the atlas prior. An important note is that with these 'binary' distributions a pixel classified as healthy tissue in the prior atlas could only ever be classified as healthy tissue or as a tumor (latent class, not involved in the prior) by the EM algorithm, since π_{ki} was 0 for every other class in the prior and therefore equation (9) would be 0 in each iteration. This detail did not affect results for $k = 3$ however for $k = 4$ it had a significant effect. In the case $k = 4$, the kmeans algorithm on the prior image does not work well since my code asks for $k - 1 = 3$ prior class estimations however the prior atlas image really only contains 2 classes (healthy tissue and background), therefore we get a very skewed and inaccurate estimation of tumor border pixels (Figure 3). Therefore if we kept the binary prior distributions then it was impossible for the tumor border pixels to be properly classified. There were 2 possible solutions, smooth the prior distribution π_{ki} or add a second latent variable $\hat{\alpha}$ that would consider the probability of tumor border occurrences. Changing the equations from the paper to account for a second latent variable class seemed too complicated and therefore I decided to smooth the prior atlas distribution π_{ki}. The procedure I used was as follows the class estimated by kmeans in the prior image received weight $\frac{1}{2}$ and the other two classes received weight $\frac{1}{4}$. For example a binary distribution such as $[0, 1, 0]$ would be smoothed to $[\frac{1}{4}, \frac{1}{2}, \frac{1}{4}]$. The key to this smoothing was to allow any pixel in the tumor to be classified as any of the k classes no matter what the prior distribution said about that pixel, this allowed for the poor prior estimate of the tumor border class to be accounted for in the EM algorithm.

Important implementation note: In my code, in 'demo.m' you need to run the code until the class indexes match between the prior distribution and the kmeans estimate of the tumor image. Because kmeans does not use a deterministic procedure the ordering of classes is random. When running my code if the indexes do not match the EM algorithm will not run and the code will ask you to run the program again.

Results:

Figure 1: k=3, μ and σ initialized by kmeans, π_{ki} non-smoothed binary distribution for each pixel i, EM iterations = 10.

Figure 2: k=3, μ and σ initialized uniformly, π_{ki} non-smoothed binary distribution for each pixel i, EM iterations = 20.

Figure 3: Region of pixels between healthy tissue and background is what kmeans estimated as the tumor border class in the prior

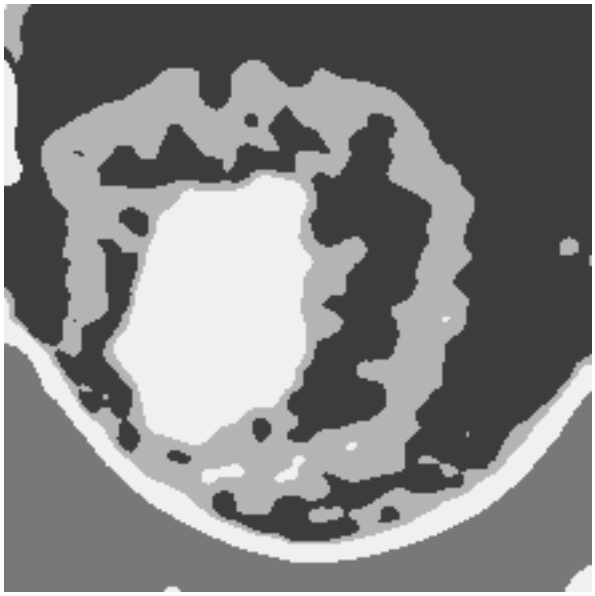

Figure 4: k=4, μ and σ initialized by kmeans, π_{ki} smoothed distribution for each pixel i, EM iterations = 10.

Figure 5: k=4, μ and σ initialized uniformly, π_{ki} smoothed distribution for each pixel i, EM iterations = 30.

From these figure we can see that the model struggles at the edge of the healthy tissue and the background, usually miss classifying it as a tumor region. This problem may be caused by the gaussian blurring of the image. For k=3 the different initialization of μ and σ resulted in similar segmentation, with the uniform initialization requiring more iterations before convergence as we would expected. However, for k=4 the difference in initialization created very different answers. The kmeans initialization seems to have lead to a more accurate answer, therefore we could argue that as the number of segmentation classes increases, intelligent initialization of μ and σ becomes more critical in reaching a better answer. Overall I was pleased that the smoothing idea for π_{ki} seemed to work for $k = 4$ case.

References

[1] S. C. Joshi and M. I. Miller. Landmark matching via large deformation diffeomorphisms. *IEEE Transactions on Image Processing*, 9(8):1357–1370, Aug 2000.

[2] Bradley C. Davis. *Medical Image Analysis via Frechet Means of Diffeomorphisms*. PhD thesis, Chapel Hill, NC, USA, 2008. AAI3304360.

[3] Yuri Boykov and Marie-pierre Jolly. Interactive graph cuts for optimal boundary region segmentation of objects in n-d images. *Proceedings. Eighth IEEE International Conference on Computer Vision: 2001. Vol.1*, 1:105–112, 07 2001.

[4] Ben Glocker. Random fields for image registration. 2011.

[5] Ben Glocker, Nikos Komodakis, Georgios Tziritas, Nassir Navab, and Nikos Paragios. Dense image registration through mrfs and efficient linear programming. *Medical Image Analysis*, 12(6):731 – 741, 2008. Special issue on information processing in medical imaging 2007.