Assignment 4
Image Enhancement

CS 473/673: Medical Image Processing

Due 4:00pm Monday 29 February 2016

1. Effect of Resampling

[6 marks] In this question, you will write a Matlab script called `resample_demo.m` to observe the effect of repeated resampling of the image `pd.jpg`. Store the image in an array called `f`. Use the linear interpolation feature of `MyAffine`, as well as the `'centred'` feature, to apply the two following sequences of rotations about the centre of the image:

(a) Apply a sequence of **two** rotations, 30° followed by −30°, to `f`. That is, rotate `f` by 30°. Then rotate the resulting image by −30°, so that your final image is in its original orientation. Call the final image `fa`.

(b) Apply a sequence of **six** rotations, each of 5°, followed by **another six** rotations, each of −5°. Again, the final image should be in its original orientation. Call the final image `fb`.

(c) Add lines to your script to compare `fa` and `fb` to `f` by displaying two plots of the absolute difference images (one for $|\texttt{fa}_n - \texttt{f}_n|$, and one for $|\texttt{fb}_n - \texttt{f}_n|$). Use the same intensity scale for the two difference images, so that 0 maps to black, and 50 maps to white.

(d) Compute the sum of the absolute differences by adding up $|\texttt{fa}_n - \texttt{f}_n|$ over all the pixels (likewise for `fb`). Based on the sum-of-absolute-differences (SAD) value, which method is more accurate, (a) or (b)? Explain why. Place your answer, including the computed SAD values, in a comment at the end of your script.

You may (should) use the solution versions of `MyAffine`, `p2m`, etc. The supplied version of `MyAffine` has the same calling signature as the assigned version from A2, but has extended functionality. See its help file for more information.

2. Image Derivatives

[6 marks] Implement your own derivative function in Matlab. Your function should be called `MyDerivative`, and should work on both 2D and 3D datasets. The function must have the calling prototype,

```
dfdx = MyDerivative(f, ax)
```

and return a finite-difference approximation to the derivative with respect to the motion parameter specified by the number `ax`. The parameter `ax` is a number from 1 to 6, referring to the motion parameters as they are ordered in the input for the function `p2m`:

(1) rotation about the r-axis (axis that enumerates rows, positive down)
(2) rotation about the c-axis (axis that enumerates columns)
(3) rotation about the s-axis (axis that enumerates slices)
(4) shift in the positive r direction
(5) shift in the positive c direction
(6) shift in the positive s direction

The return value, `dfdx`, is an array the same size as `f`. The rotational parameters should be interpreted as rotations about the centre pixel (see the function `GetCentre`), as opposed to rotations about the Matlab index coordinate system origin. The (r, c, s) axes form a right-handed coordinate system, and the direction of positive rotation is defined accordingly.

For $\frac{\partial f}{\partial r}$, $\frac{\partial f}{\partial c}$, and $\frac{\partial f}{\partial s}$ (i.e. derivatives with respect to translation parameters), your function should use central differencing for all internal pixels, and either forward or backward differencing for the pixels on the borders of the image where the central-differencing template would extend outside the image. You should use a Δ step of 1 pixel.

For $\frac{\partial f}{\partial \theta_r}$, $\frac{\partial f}{\partial \theta_c}$, and $\frac{\partial f}{\partial \theta_s}$ (i.e. derivatives with respect to rotation parameters), your function should use a central-differencing approximation with a θ-step of 1°, and employ either linear or cubic interpolation. Your function should make use of your previous functions (eg. `MyAffine`).

For the sake of clarity, we will define the derivative of a pixel's intensity with respect to motion parameter m as

$$\frac{\partial f_n}{\partial m} = \lim_{\Delta m \to 0} \frac{[T(m + \Delta m)f]_n - [T(m)f]_n}{\Delta m}$$

where the notation $[T(m)f]_n$ refers to the intensity of pixel n of the image f after it has been moved by the operator $T(m)$. That is, how does the intensity of pixel n change as you apply the motion m to the image? Note that this is opposite to a partial derivative; the difference is whether you move your location, or move the image... the effect is opposite. I include this so that you can tell when your derivatives should be positive and when they should be negative.

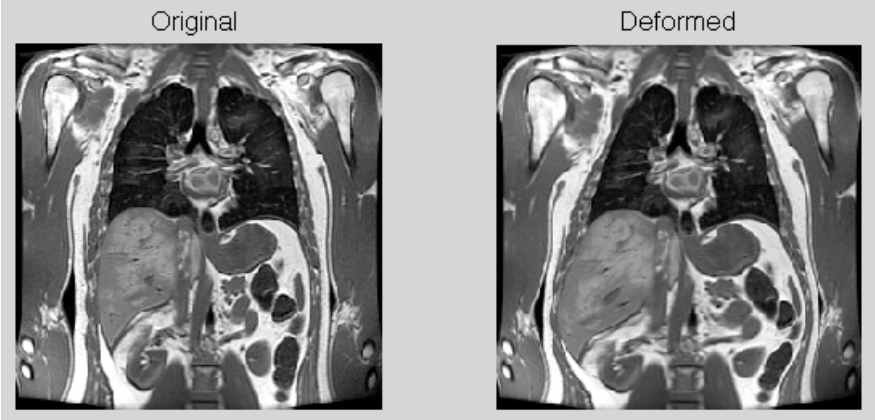
3. **Graduate Question**

[5 marks] Implement a non-affine image deformation function that has the calling signature

```
g = MyDeform(f, params, method)
```

where `f` is the input image (2D only), `method` is one of `'nearest'`, `'linear'` or `'cubic'`, and `params` is an array of parameters that specifies the deformation. You may use the starter code supplied online; however, you must replace the code where indicated. It is up to you how you parameterize your deformation. In class, I mentioned a few examples: piecewise-affine, polynomial, basis functions (Gaussian, Fourier, etc.). You can choose whichever you like, or come up with a different parameterization.

Along with your function, write a script that reads in an image and applies an interesting deformation using a random set of parameters (include your sample image with your submission). By *interesting*, I mean that the deformed image should be easily distinguishable from the original (usually). Your script should also display the original and deformed images, like the figure below. Your `MyDeform` function should be designed so that the parameters make sense no matter what the input image size is (although you can assume that the input image is 2D, and is at least 2×2 in size). This script should be saved in a file named `random_deform.m`.



What do I submit?
Your submission should consist of the following files:

1. `resample_demo.m`, including comments at the end
2. `MyDerivative.m`
3. (for CS 673 only) `MyDeform.m`, `random_deform.m`, your image file

Place your code in a directory, **zip up the directory**, and submit the zip file to the "Drop Box" on Desire2Learn. It is important that you submit a compressed file, like a zip or tar-gnuzip, etc., instead of submitting a bunch of individual files.