Assignment 1
Affine Transformations

CS473/CS673: Medical Image Processing

Due 4:00pm Monday January 25, 2016

Goal: to lay the foundation for representing the affine transformations that you will apply to images later.

Questions

1. [4 marks] Write a Matlab function called `p2m` (which stands for "parameters-to-matrix") that creates a 4×4 matrix representing a 3D rigid-body transformation. It takes as input a 6-vector, $[\theta_1, \theta_2, \theta_3, t_1, t_2, t_3]$, and returns the composite transformation matrix resulting from applying the rotations and translations in the order they are listed (from left to right). For example, the first operation is a rotation of the (Euler angle) θ_1 degrees about the axis 1 (using a right-handed system), and the last operation is a translation of t_3 pixels along axis 3. Your function should output the transformation matrix as a single composite 4×4 matrix using homogeneous coordinates.

2. [4 marks] Write a Matlab function called `m2p` that does the exact opposite of question 1 above. That is, given a 4×4 matrix representing a 3D rigid-body transformation in homogeneous coordinates, extract the rotation angles and translations used to compose the transformation matrix. Keep in mind that the order in which component operations are applied matters. *Note:* You may assume that all the angles are less than $90°$ in magnitude.

3. [4 marks] Write a Matlab script called `make_matrix` to compute the transformation matrix that performs the following composite transformation:

   (a) translates to the left by 9 pixels, and down by 21 pixels, then

   (b) rotates $10°$ counter-clockwise about the pixel (50, 60) (specified here in Matlab index coordinates).

   Express your answer as a single 4×4 transformation matrix with respect to Matlab's index coordinates, and save it in a variable named `M`. You may use any of the functions from questions 1 and 2. In your code, show your work by expressing the composite transform in its component parts before multiplying them together (part marks for this). Remember, your script should save your matrix in the variable `M`.

What do I hand in?

1. `p2m.m`
2. `m2p.m`
3. `make_matrix.m`

Place your code in a directory, zip up the directory, and submit the zip file through the "Drop Box" on Desire2Learn. Your Matlab code will be tested by running it. However, we will also be looking at the listings of your scripts and functions. It is essential that you document your code. A well-designed and properly implemented script or function will NOT earn full marks without comments in the code.