CS6640 – Project 3 Image Transformations
Assigned Oct. 24, 2012
Due Mon. Nov. 11 (Just before midnight)
Instructor: Guido Gerig
TA: Lingbing Jiang, Office MEB 3115, office hours Tue,Thu 3 - 5pm

Goals

The purpose of this assignment is to learn about linear and nonlinear image transformations (warping). Please read following instructions carefully.

Materials: Please use the updated documents "warp-with RBFs-GG-Oct2010.pdf" and also "Lecture_10_18_Transformations_Durrleman.pdf" as references for this project, in particular for a solution to image warping by radial basis functions. The documents are uploaded under slides and also in the Project 3 directory.

1 Affine Image Transformation

Implement a program that transforms an image given an affine transformation (6 parameters), which includes rotation, translation, scaling and shear. Please note that the transformation is generally applied from the target image backwards to the source image, i.e. you step through each pixel of the new image, determine the position in the source image, and take/calculate the intensity at this pixel to be used for the target image. Two types of interpolations need to be implemented:

1. Nearest neighbor (NN): Take the intensity from the pixel which is closest to the non-grid position of the transformation. Please note that this can be easiest achieved by rounding a non-grid (x,y)-ccordinate to the next integer coordinates.

2. Implement bilinear interpolation from the 4 neighbors of the non-grid coordinate, folling the discussion in the course.

Take an input image of your choice and apply:

1. Separate translation, rotation, scaling, and shear.

2. An affine transformation with 6 parameters.

3. For the affine transformation, apply NN and bilinear interpolation and show the differences (you show a part that is strongly zoomed (which can be calculated with your scaling transform).

2 Calculation of affine transform from landmarks

As discussed in the course, a transformation can be determined based on a set of corresponding landmarks in a source and a target image. A minimum of 3 points with (x,y) coordinates is required, but more landmarks result in a more stable solution by solving an overconstrained linear equation system. Please note that you can use Matlab help and Mathworks web-based help for hints on solutions.

- Implement a module that determines a pixel position with the mouse.

- Use this module to create sets of corresponding pixel positions in a source and a target image.

- Set up the linear equation system and implement a solution to solve for the affine transformation between the source and target image(*).

- Apply the transformation and and check for the correctness of the result by displaying source, target and resulting images side by side. You can even create another result by blending the result and source together (e.g. adding the images) to have some visual check of geometry differences.

As test images, you can use own pictures (e.g. of frontal view of faces of humans or animals). Would such pictures not be available, you can search the web (e.g. http://www.face-rec.org/databases/). You can be creative about the choice of images, applications as shown in the course slides are for example lip reading in video sequences or normative atlas building in medicine.

(*) see additional materials in regard to solving a linear equation system.

3 Calculation of nonlinear warping using radial basis functions

Note: Only start this section after successful completion of the two sections above!

As discussed in class and explained on the course slides, you can warp images based on sets of corresponding landmarks via a nonlinear transformation that uses radial basis functions (RBFs). You can see that elements of code developed in the previous sections will be the same, such as determining sets of corresponding landmarks and solving a linear equation system.

Following the course notes, setup a linear equation system based on RBFs for image warping and choose a strategy for solving the system (e.g. Matlab functions).

- Determine a set of corresponding landmarks.

- Choose a Gaussian kernel for the function $\Phi(\bar{x})$ where the Gaussian width σ is your free parameter.

- Setup the equation system, solve for the parameters.

- Given the solution, transform the image.

- Experiment with different sets of landmarks (e.g. one only up to a few) and with a given set of landmarks, a few Gaussian widths.

- Show resulting images and discuss.

IMPLEMENTAION HINT: The true inverse to warping by radial basis functions does not exist. It is suggested that you first implement the transformation from your source image $\mathbf{f(\bar{X})}$ to the new target image $\mathbf{g(\bar{Y})}$, knowing that by stepping through every pixel in your source image X you get a new location Y, but that this can create holes as this is a forward process:

$$Y = X + \bar{k}_o * \Phi(||X - X_0||).$$

By replacing X by X_0, you can verify that the landmark X_0 is moved to its new location Y_0, with $= 1$ and $\bar{k}_0 = (\bar{Y}_0 - \bar{X}_0$ as shown in our class lecture.

Whereas the exact inverse does not exist, you may implement an approximation where you put the radial deformation kernel not at the source landmarks X_0 but at the target landmark Y_0.
The equation becomes

$$X = Y - \bar{k}_o * \Phi(||Y - Y_0||).$$

This allows you to step through your new, empty image Y pixel by pixel, calculate the pixel location X to be transformed, take the intensity at this X (by nearest neighbor or bilinear interpolation), and paint this intensity at your location Y. Since this is a backwards process, there are no holes created since every new pixel Y finds an original value in your source image. By replacing Y by Y_0, you can verify that the landmark Y_0 gets its intensity from the source landmark X_0 since the kernel Φ gets to 1 and $\bar{k}_0 = (\bar{Y}_0 - \bar{X}_0$ as before. Implementation of this process does only require minimal changes to the forward process, and you should try this process and compare.

Hint 2: Results can be best interpreted by using a checkerboard image for testing. Howeve, deforming real images of your choice can be more fun.

4   Instructions, Requirements, and Restrictions

1. Please use your name **"NAME"** in all reportts and submitted materials to uniquely identify your projects.

2. Write your project code in a single directory, called `project1-NAME`.

3. For Matlab each individual function (including functions you define) should be a ".m" file, and your functions should call one another as necessary.

4. We **do not allow to use Matlab toolbox functions** (e.g. Imaging Toolbox) or other existing image processing libraries in order to give all students the same conditions for code development [1].

5. You should have in your report a short description of each algorithm you used and documentation on how your code is organized. Failure to do this will result in a loss of points. Please remmember to **add your name** to the report title.

6. **You can use submit your report as html or pdf document** (created from Word, LaTeX or other word processing system of your choice.

   - Using html: Your project report will be in the form of an html file called **index.html**, contained in that directory. All links from that file must be relative and all other files necessary to read your report must be in that directory (or subdirectories).

7. You should use examples of images in your report. They should be viewable in the browser when we open your html file.

8. You will submit a single tar file created from from your project directory with the unix command *tar -czf project1-NAME.tgz ./project1-NAME*.

9. Please remember or look-up the honor code and requirement to provide your own solution as discussed in the syllabus.

10. **Please look up the late policy as defined in the syllabus**

---

[1] The core MATLAB package comes with several rudimentary functions that can be used to load, save, and perform custom functions on images. Taken from wikibooks