CS6640 – Project 3 Image Transformations
Assigned Oct. 29, 2014
Due Mon. Nov. 12 (Just before midnight)
Instructor: Guido Gerig
TAs: Avantika Vardhan, Padmashree Shrinivasa Shetty Teeka,
TA office hours Mo 3-5pm,Wed 11am - 1pm, office WEB 2626

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 slides and the documents "warp-with RBFs-GG-Oct2014.pdf" as references for this project, in particular for a solution to image warping by radial basis functions. The documents are uploaded on the class home page.

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 via backward transform (target to source):

1. Separate translation, rotation, scaling, and shear transformations, and apply NN and bilinear interpolation.*
2. An affine transformation with 6 parameters (you can cascade some transformations in the previous item).
3. For the affine transformation, demonstrate the differences between nn and bilinear, best by large zooming of an image subregion). Provide a short description on what you did and what you see.

*Hint: By transforming an image, the transformation may show parts outside of the original image size. You can either select a larger target image size or just paint pixels that have no partner pixel in the source image as black.

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 or whatever you like). 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 and Matlab help for solving overconstrained systems. E.g, for solving the system $A.x=b$ in Matlab, there are the choices of *x=A\(fastest), or x=pinv(A)*b, or x=inv(A'*A)*A'*b).*

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).

- *Choose a source and target image, or one image where you want to warp landmark points to a new location.*
- *Determine a set of corresponding landmarks (source/target, or old/new locations).*
- *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 (see comments below on forward versus backward transformation).*
- *Experiment with different sets of landmarks (e.g. one only up to a few) and with a few Gaussian widths.*
- *Also do an experiment with a relatively large landmark movement and describe/explain what you see.*
- *Show resulting images with annotations of landmarks and provide short descriptions on what you did for each experiment.*
- *Results can be best interpreted by using a checkerboard image for testing. However, additional deformation of real images of your choice can be more fun.*

- *Would you use different source and target images (rather than deforming just one image), it would be nice to overlay the resulting images in order to visualize the differences. This can be simply obtained by adding the two images (be cautious about the 8bit range, you may scale the sum back to 8bit).*

IMPLEMENTAION HINT: The true inverse to warping by radial basis functions does not exist. Whereas the exact inverse does not exist, you may implement an alternative solution 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 target 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 should be 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 $\Phi(0)$ gets to 1 and $\bar{k}_0 = (\bar{Y}_0 - \bar{X}_0)$ as before.

4 Instructions, Requirements, and Restrictions

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

2. *Your report should summarize your approach to solve the problems, show graphs, images of the results and include a critical evaluation/discussion of the results. Please do not copy the project description into your report, just the title is sufficient, but provide a short description what you did for each experiment and how you did it. The report can be written with any text program of choice, and the handin needs to be in pdf format.*

3. *You should have in your report a short description of each algorithm you used and documentation on how your code is organized. A short summary on most essential core information is sufficient, don't expand too much.*

4. *We* **do not allow to use Matlab toolbox functions** *(e.g. Imaging Toolbox) or other existing image processing libraries or solutions found elsewhere. We want you to experience challenges of implementation and give all students the same conditions for code development* [1].

5. *Your project report will be in the form of a pdf file.*

6. *Write your project code in a single directory, called* `project1-NAME` *and the best is to create a compressed tar file of code and images used to run it. Submit a separate pdf report and the compressed code/images file to canvas.*

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

8. **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.