CPSC 4040/6040
Computer Graphics Images

Assigned: Oct. 24, 2014
Due: 11:59pm, Fri., Nov. 6, 2014

Lab 6 — Projective Image Warp *(Grading: 0–10 points)*

Project Description

This week you will build a warping tool based on projective warps, which includes all of the affine and perspective warps. The tool will provide for input of a sequence of operations, will compose a single transform from the sequence, and finally will apply the transform to the input image via inverse mapping.

You will have to write procedures for constructing each of the projective transformations: translate, rotate, scale, flip, shear, and perspective. You will also have to write routines to warp an input image into the new shape, and then do the inverse mapping of the input image to the output image via the inverse of the projective transformation.

You have been provided routines to do 3×3 matrix multiplication, transform of a 2D point by a 3×3 homogeneous matrix, and inverse of a 3×3 matrix, and a few others. You may use our routines or write your own if you wish. I have also provided routines for processing the sequence of input commands for the transform specification. This is a sample routine, your job will be to fill in the code that accumulates the transformation for each of the user's inputs (it shows one examples with rotate operations). Likely, the best option will be to remove this code from the example and merge it with the code that uses OIIO and OpenGL.

Basic Requirements

The program you write should be called `warper`, and will consist of 2 parts:

Part 1:

- read an input image from an image file
- input a sequence of matrix specifications
- multiply these matrices as you go to get a final transform matrix from all of the specified matrices
- transform just the corners of the image with the forward transform
- make space for a new image exactly big enough to contain the four transformed/warped corners.

Part 2:

- calculate the inverse transform by inverting the final transformation matrix
- transform the image with the inverse matrix, truncating non-integer pixel locations to integers and accounting for if the origin has moved.
- optionally (if a second filename was specified) save the transformed image in an image file.
- display the transformed image

The command line must be of this form:

```
warper inimage.png [outimage.png]
```

If the output filename is missing, the program simply skips saving the output image. If no filenames are present, the program prints an error message and stops.

The specifications for each transform follow. Your program should loop, reading these until the `d` command is input). *Italics* indicate floating point arguments, except for the `f` command, which takes integer arguments:

`r`	θ	counter clockwise rotation about image origin, θ in degrees
`s`	*sx sy*	scale (watch out for scale by 0!)
`t`	*dx dy*	translate
`f`	*xf yf*	flip - if xf = 1 flip x coordinates, yf = 1 flip y coordinates
`h`	*hx hy*	shear
`p`	*px py*	perspective
`d`		done

For grading, your program will be tested using a script, so please adhere to the standard interface. If your program has any optional switches or features: 1) make sure that they do not interfere with the standard operation of the program (i.e. if the switches are not present the program should run exactly as specified above and below). If you want to give your program a graphical user interface, please turn in two versions of the program, one with the standard interface, and one with the graphical interface, and make sure you include a README file explaining what you have done.

Code Documentation Please see Lab 01 for the expected requirements. Your code will be evaluated based on code structure, commenting, and the inclusion of a README and build script.

Advanced Extension for 6040 students (worth 3 points)

You may do either of the following:

1. Create an executable `bilinear` which runs your program using bilinear interpolation of position in the output quadrilateral instead of the standard inverse projective matrix. To do so, you will still use the given matrix to forward transform the input image vertex positions, but the inverse map based on the matrix will be replaced with the inverse bilinear map discussed in class. Note, you may want to make use of operations in the Vector class to improve your interpolation techniques.

2. Create an executable `interactive` which runs your warper in interactive mode. Instead of allowing the user to specify an input matrix, make use of mouse input to allow the user interactively position the corners of the output warped quadrilateral and then compute the projective warp using those positions (note, some helpful code is in the provided matrix package, you may want to make use of the n-dimensional Matrix class and using an 8×8 inverse)

Submission

(Please read all of these instructions carefully.)

Please write the program in C or C++, and I recommend using OpenGL and GLUT graphics routines for the display. Use OIIO for image input and output. Please take extra care to make sure that your homework compiles on the School of Computing's Ubuntu Linux cluster—testing on your own home machine, even if it runs Ubuntu, may not be sufficient. Remember:

<u>both a working build script and README must be provided</u>

Consequently, to receive *any credit whatsoever* this code must compile on the cluster, even if it does not accomplish all of the tasks of the assignment. The grader will give a zero to any code that does not compile.

Submit using the handin procedure outline at `https://handin.cs.clemson.edu/`. You are welcome to use the commandline interface, but the web interface is sufficient. The assignment number is `lab06`.

Finally, since we are using multiple files, please only submit a single file which has aggregated everything. This file should be named `[username]_lab06.tgz` where `[username]` is your Clemson id (please remove the brackets []). For example, mine would be `levinej_lab06.tgz`. Please make sure your build script is at the top level directory.

Rubric for Grading

4040 students will be graded for the following requirements:

I. +2 Reading, Writing, and Display

Code correctly reads in input image, displays the warped image after 'd' is input signalling the warp is fully specified, and then writes out the warped image if an second filename is specified.

II. +3 Warping Basics

Code correctly supports the user interface for entering commands. The warped image dimensions are correctly computed. Code correctly computes the aggregated matrix and makes use of the inverse map to fill in color values for the warped image.

III. +3 Specific Warping Types

Rotation, scaling, translation, flipping, shearing, and perspective warps are correct.

IV. +2 Clarity

Does the code have good structure? Is the code commented and is a README provided? Is a working build script provided?

6040 students will receive 7 points for completing the above requirements, scaling the above by 70%. To achieve 10 points they must also complete:

I. Advanced Extensions (if `bilinear` is chosen)

Code correctly solves for the inverse equations, producing the appropriate a_i, b_i, and c_i coefficients. Code correctly checks the u and v values to ensure they are in the range $0 < u, v < 1$. Using these equations, the code correctly replaces the inverse matrix multiplication with an inverse map lookup based on bilinear interpolation.

II. Advanced Extensions (if `interactive` is chosen)

Code provides an interface for the user to select new positions for the four corners of the input image. Code correctly solves the system of 8 equations associated with the new coordinates. Using the solved matrix, the code correctly applies the inverse warp using the inverse matrix.

Going above and beyond these requirements may result in extra credit at the discretion of the instructor and grader. Please note any extra features you implement in the README.