Computer Graphics – Exercise 1 – Image Processing

Objective

The objective of this exercise is to implement several image processing algorithms and demonstrate their behavior. You will implement all of the following algorithms:

- Sampling (40 pts)
    - Nearest neighbor
    - Bilinear interpolation
    - Gaussian sampling
- Warping/Transforming (25 pts)
    - Scale by a real factor (i.e. 0.5, 2.0)
    - Rotate by a real factor
    - Non-linear mapping (fish-eye, sine, bulge, swirl…)
- Convolution (35 pts)
    - Uniform Blur
    - Gaussian Blur
    - Edge Detect
    - **Custom kernel**
- Dithering (bonus)
    - Floyd-steinberg

Definitions:

- Source image – the input image of the algorithm, the image supplied by the user.
- Destination image – the output image of the algorithm.

Sampling

These algorithms will be used and tested in the warping algorithms which appear below

Nearest neighbor

Nearest neighbor is the simplest method of sampling an image. Using nearest neighbor, every pixel in the destination image corresponds to exactly one pixel of the source image and receives its color.

Notice that if during sampling, you enlarge the image, several pixels of the destination are going to take their color from a single pixel of the source image. This effect is sometimes referred to as **pixelization**. Enlarging an image using nearest neighbor can be thought of as simply stretching its pixels. Also notice that if you shrink the image, some of the source pixels are going to get skipped entirely and their color is not going to end up in the destination.

Bilinear interpolation

Bilinear interpolation is a slightly better sampling scheme than **nearest-neighbor**. Instead of directly copying pixel values from the source image it uses linear interpolation to set the destination's pixels values. For example, in the case of enlarging an image, bilinear interpolation works as follow:

For every pixel **d** in the destination image, find its originating coordinates in the source image, set **d's** value using bi-linear interpolation of the four nearest neighbors (in the source image), as seen in class.

In the case of shrinking an image the process is slightly different and is left for you as part of the exercise. Another explanation of bilinear interpolation can be found in the following locations:

- http://www.gamedev.net/reference/articles/article2007.asp
- http://en.wikipedia.org/wiki/Bilinear_interpolation

Gaussian sampling

Each destination pixel's color is calculated as a weighted average of its source's neighbors. It is up to you to determine the parameters of the Gaussian.

Warping/Transforming

In all of these assignments, use all three of the sampling methods you implemented above

Scale

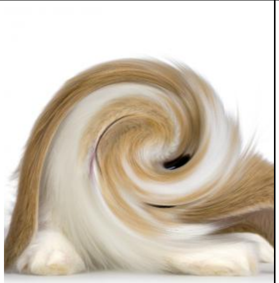
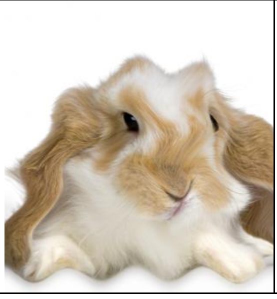
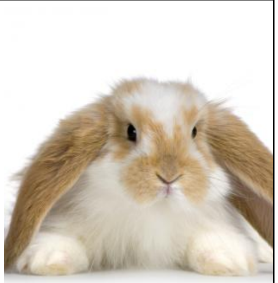
Given a source image and a scaling factor, output a destination image who matches the desired size. The scaling factor is uniform (i.e. same scale in X and Y axis).

Rotate

Rotate a source image by a given number of degrees. Note that you need to calculate the size of the axis-aligned rectangle in which the rotated image can fit.

Non-linear mapping

Implement at least one non-linear mapping to transform a source image. Some examples:

Original	Twirl	Sine	Pinch
			

Convolution

Given a source image, apply a convolution filter on it to produce the destination image. The built in kernels should be blur (uniform and Gaussian) and an edge-detection operator. Also you should support a custom convolution kernel supplied by the user (size can be fixed – 3x3 or 5x5). Output image should be at the same size as the input. *Note that you need to address what happens at the edges of the image.*

Dithering (bonus)

As a bonus assignment you may implement an error-diffusion dithering algorithm such as **Floyd-Steinberg** (which we learn in lesson #3, see also here - http://en.wikipedia.org/wiki/Floyd-Steinberg_dithering).

Input to the algorithm should include the number of desired colors in the output image.

How to write and submit the assignment?

The assignment should be written in a high level programming language (we highly recommend Java but C/C++ is accepted, please ask before using other languages).

The program should be named **cg10a_ex1** and accept as command line arguments the name of a configuration file. The configuration file will be of the following format:

```
<name of source image>

<name of destination image>

<name of algorithm to run>

<list of parameters separated by coma, can be empty if no parameters>
```

Just to be extra clear, four lines, text format. Matrix information (e.g. kernel) should be represented column by column.

For example:

Example1.config

```
Parrot.jpg
Small_parrot.jpg
Scale_nearestneighbor
0.5
```

Example2.config

```
Seahorse.jpg
Rotated_seahorse.jpg
Rotate_gaussian
4
```

Example3.config

```
Building.png
Blurred_building.jpg
Convolution_custom
0 0.125 0 0.125 0.5 0.125 0 0.125 0
```

Running your program should look something like this:

```
Java –classpath . cg10a_ex1 Example1.config
```

Additional notes

- Algorithm names in the config file should be:
  - Warping
    - Scale_nearestneighbor
    - Scale_bilinear
    - Scale_gaussian
    - Rotate_<nearestneighbor, bilinear,Gaussian>
    - Warp_<nearestneighbor, bilinear,Gaussian>
  - Convolution
    - Blur_gaussian, Blur_uniform, Edge_detect, Convolution_custom
  - Dithering
    - Dithering
- We will publish a few sample images on the website for you to try out
- If using Java use Java 6(!)
- You are expected to implement everything yourself, without use of image processing utilities of the Java library
- We recommend using *IntBuffer*, *ByteBuffer* or *BufferedImage* to store an image in Java

Grading

- You will be graded based on the **quality** of your implementation's output on our inputs (compared to the state of the art and to other submissions)
- Although optimization is not required your code must run at a reasonable amount of time (e.g. in comparison to others)
- Copying code from anywhere/anyone is strictly prohibited and will result in disqualification

Submission

- Submission must be in pairs (unless approved by Lior)
- On submission date (by midnight) you must send the following items to the address listed below
  - Source code should be placed in a zip file
  - Executable (if its not java) should be packaged in a zip file and its extension changed to **zi_** so it isn't filtered by the mail server
  - Documentation should be a 1-3 page document (doc, pdf) explaining the structure of your code, how to operate the application and anything else needed to make it work
- The email subject should be of the following format: **ex1 <id1> <id2>**, for example "ex1 03444555 012333333"
- Submission and questions should be directed to:
  chen.goldberg+TAU_CG@gmail.com