Lab 3: Geometric Processing

due TBD

Goal:

In this lab you will implement some geometric transformations to an image in MATLAB.

Introduction:

You have a handout of Chapter 5 from the *Digital Image Processing Using MATLAB* book to read about spatial transformations. You should also have a section of another chapter from a digital image processing book by Baxes.

"**Warping**" loosely refers to a **spatial coordinate transformation**. Mathematically this can be expressed as a **mapping**, T. This operation distorts an input image with coordinates (x,y) to produce an output image with coordinates (w,z). Note that T can be written as a matrix that performs the operation (via **matrix multiplication**):

$$(x,y) = T\{(w,z)\} \qquad (1)$$

Page 182 of the Chapter 5 handout has an example (Figure 5.12), and page 183 of the Chapter 5 handout has a table showing some common transformation matrices (Table 5.3).

The three general categories of spatial coordinate transformation include **RST** (a **linear combination** of **rotation**, **scaling**, and **translation**, also known as an **affine transformation**), **polynomial warping**, and **triangulation**. Several affine transformations are given in Table 5.3. Polynomial warping applies a **polynomial function** to a set of known points (**GCPs**, or **ground control points**) defined on the image and **non-linearly** warps the rest of the image to fit that model. Triangulation fits a set of triangles around a set of GCPs and performs an independent warping (either affine or polynomial) for each triangle.

Note that the above discussion *only* refers to transformation of **image coordinates**. It does *nothing* about the *values* (e.g. DN or BV) the remapped pixels should have, only *where* they are mapped to. So how do we determine the value stored at some pixel location (w,z) in the output image that was mapped from some location (x,y) in the input image? That is where "**interpolation**", or **pixel value assignment**, comes into play.

It is first important to understand the two strategies for warping and subsequent interpolation (see Chapter 5 of the handout): (a) forward mapping and (b) inverse mapping. For each input pixel (x,y), **forward mapping** computes (using RST, polynomials, or triangulation) where it will be in the output image (w,z) and simply copies the input pixel value to the new location. With this simple strategy, a serious interpolation problem arises when *multiple pixels* from the input image map to the *same*

location in the output image (yes, this is possible!), or when output pixels cannot be assigned a value since they are not mapped to (this is also possible!). For these reasons, forward mapping is seldom used in practice.

The way around these awkward problems is **inverse mapping**. This strategy *starts with the output image* and computes (using RST, polynomials, or triangulation) where each location (w,z) in the output image originates from in the input image. Mathematically, whereas the warping step of forward mapping uses equation (1), inverse mapping uses:

$$(w,z) = T^{-1}\{(x,y)\} \qquad (2),$$

where T^{-1} is the **matrix inverse** of T. In this way, those locations in the output image that do originate from the same location (or no location) in the input image can be identified, and all locations (w,z) can be both mapped *and* then easily assigned a value via interpolation. Values can be assigned using nearest neighbor interpolation, bilinear convolution, or cubic convolution. The **nearest neighbor** method simply identifies the input image pixel closest to the location (x,y) mapped *from* (w,z) (see equation (2)) and uses its value in the output image. **Bilinear interpolation** uses a weighted average of the values stored in the four nearest pixels. **Cubic convolution** uses the nearest 16 pixels and the *sinc* function. Further details on these algorithms are given in the reading.

Instructions

Your task is to take the `test.img` image from the first lab and write a MATLAB program to apply the following affine spatial coordinate transformation: scale by 50% and rotate counterclockwise by 30°. You will need to build the transformation matrix (or matrices, if you like) that performs these operations. Then, while applying the transform using inverse mapping, your program should apply either nearest neighbor resampling or bilinear interpolation (it's up to you which one – extra credit for bilinear) to assign the correct pixel values to the output image.

What to Turn In and How:

All I want are the `.m` file(s) for your MATLAB program(s), nothing else. Be sure to put your name in a comment at the top of all your files. Also be sure to adequately comment your code so that we know what your programs do and how they work. Turn in your code via the `DROPBOX` on `//yowa.geo.utep.edu/GEO5336`.