HOMEWORK 2
(Early deadline October 15, Deadline October 24)

(1) Geometric transforms (100 pts)

A geometric transform is a vector function T that maps the pixel (x, y) to a new position (x0, y0). T is defined by its two components

$x_0 = T_x(x, y)$, $y_0 = T_y(x, y)$

Geometric transformations are implemented in two steps. First for a point (x_0, y_0) we find from the inverse transform T^{-1} the corresponding point (x, y). Second, since x and y are not integers, we need to estimate the brightness value at (x, y) by interpolation (from neighboring integer points).

Develop programs for the following geometric transforms:

(a) Rotation

(b) Change of scale

(c) Skewing. Skewing by an angle **a** is defined as

$x_0 = x + y \tan(\mathbf{a}),\ y_0 = y$

(d) Affine transform calculated from three pairs of corresponding points. An affine transform is defined as

$x_0 = a_0 + a_1 x + a_2 y$, $y0 = b_0 + b_1 x + b_2 y$

For each of the above transforms, implement the following two brightness interpolation approaches:

• Nearest-neighbor interpolation

• Bi-linear interpolation (from four neighboring points), which can be implemented as convolution, as discussed in class.

Run your algorithms on a picture of your choice. Print the code. Print your results for (a) Scaling by a factor 3 (b) An affine transform with points A = (1, 0),B = (−1, 0),C = (0,3) mapping to points A0 = (1.9, 0),B0 = (−0.5, 0),C0 = (0, 1) using bi-linear interpolation.

(2) Filtering (50 pts)

Consider a 5X5 image patch:

9	10	9	4	3
5	7	8	9	3
4	5	**6**	8	5
3	4	5	6	8
2	3	4	5	6

Figure 1: Image patch

(a) For the image patch in Figure 1 at the pixel at the center (that is the pixel marked in bold face) apply the following filters and round to the nearest integer value:

i. a 3 × 3 Gaussian filter:

$$\frac{1}{16}\begin{matrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{matrix}$$

ii. a 3×3 Box filter (that is averaging in a 3×3 neighborhood).

(b) Compute the edge direction and strength (that is the direction and absolute value of the image gradient) at the center pixel using the masks of the Sobel edge detector.

$$S1 = \frac{1}{8}\begin{matrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{matrix}$$

$$S2 = \frac{1}{8}\begin{matrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{matrix}$$

(c) Apply a median filter to the center pixel. Explain for what kind of noise median filtering will work best.
(d) Why is the Gaussian filter a good smoothing filter? (How can it be implemented fast? How can we implement repeated Gaussian filtering in one operation?)
(e) What happens to the two edges at the boundaries of a dark line on a white background if the image is smoothed with a Gaussian with kernel size larger than the width of the line?
(f) Explain why Box filtering (that is averaging) attenuates the noise.

3: **Edges (50 pts)**:Suppose you have a 1D image described by the equation I = cos(x/100). You run a 1D edge detector on this image, in which you smooth with a Gaussian with a sigma of 1.5, and then find points in which the magnitude of the derivative is greater than a threshold, T, and which are locally maximal. Where would you find the edges? What are the set of possible answers that you could get for all possible values of sigma and T?

4. Image Gradients - 25 Pts: a. Consider the following, small image:

1	2	3	4	5
3	4	5	6	7
5	6	**7**	8	9
7	8	9	10	11
9	10	11	12	13

a. Compute the magnitude and the direction of the gradient for the central point (**7**). Suppose *x* values increase as you go to the right, and *y* values increase as you go down.

b. Now consider a different image. Suppose the intensity of an image at location (3,7) is 17, and the gradient is equal to (3,-2). Use this to estimate the intensity you would expect to see at location (3.1, 7.3) if you were able to measure intensities with subpixel accuracy.

c. Consider an image in which the intensities can be described as $I(x,y) = (x-5)^2 + (y-1)^2$. What is the gradient at the pixel (3,2)?

5. (Filters) (25pts) Create a 2D filter that computes the derivative of an image in the diagonal direction (ie., in the direction of the vector (1,1)). That is, this filter estimates the amount the image would change if you move one pixel in a diagonal direction. Apply this filter to the "image"

1	2	3	4	5	6	7	8
2	4	6	8	10	12	14	16
3	6	9	12	15	18	21	24

Show the results that you would get for the middle row of the image.

6. (**100 Pts**): **Histogram comparison of color images**

In this problem, you will write code to generate a 3-D histogram of a color image, and then to compare two histograms to judge their similarity. To create the histogram, you will divide the R, G and B channels into 8 intervals of equal size. This will produce a total of 512 buckets.

- Do this by creating the function:

h = myColorHist(I)

which takes an rgb image as input, and returns a histogram. h can be a vector of 512 values (or it could be a 3-D, 8x8x8 array if you prefer).

- Next, create a function

d = histDist(h1, h2).

This function takes two histograms (as created by myColorHist) as input and returns a distance, which indicates the SSD between the two histograms. Keep in mind that before computing the sum of square distances between two histograms, they must be normalized; otherwise large images will always be considered to have histograms that are very different from small images.

- Finally, use these two functions to compute a little 6x6 table, showing the distance between all pairs of the images desert1, desert2, desert3, forest1, forest2, forest3. These are six, more or less random images that you will download after googling "forest" and "desert". Turn in this 6x6 table, your code, and a one paragraph write-up explaining what you think these results indicate about the potential for using color histograms to classify images by the type of scene they depict.vide the semantic similarity

Extra Credit: Develop your own function that will provide the semantic similarity between two images

7. **Masks (25 pts)**

Design a single mask that detects edge elements making an angle of 30 degrees with the x-axis. The mask should not respond strongly to edge elements of other directions or to other patterns.

8. Corner detection (50 pts)

Design a set of four 5X5 masks to detect the corners of any rectangle aligned with the image axes. The rectangle can be brighter or darker than the background. Are your masks orthogonal? Specify a decision procedure to detect a corner and justify why it would work.

9. Convolution (25 points)

Suppose that function h(x) takes value 1 for -1/2<= x <= ½ and is zero elsewhere and function f(x) takes value 1 for 10<=x<=20 and zero elsewhere. (a) Sketch the two functions f and h. (b) Compute and sketch the function f(x)*h(x) (c) compute and sketch the function h(x)*h(x)

10. (***This is about filtering)*** **(50 points):** Contours of maximum rate of change in an image can be found by locating directional maxima in the gradient magnitude of the image (or equivalently by finding zero crossings in the Laplacian of the image.

(a) Why it is considered important to convolve the image with a Gaussian before computing the Laplacian?

(b) Convolving the image with a Gaussian and then taking the Laplacian, i.e. $\nabla^2 (G * I)$, is equivalent to taking the Laplacian of the same Gaussian and then convolving that with the image., i.e. $(\nabla^2 G) * I$. Why does the second formulation lead to a more efficient implementation?

(c) The Laplacian of a Gaussian can be approximated by the difference of two Gaussian kernels with appropriately chosen standard deviations. As a result, show that the computation can be implemented as $(\nabla^2 G) * I = G_1 * I - G_2 * I$.

(d) Why is the difference of Gaussians implementation even more efficient than the $(\nabla^2 G) * I$ Laplacian of the Gaussian implementation?

11. Unsharp masking (100 pts)

(From Wikipedia) Unsharp masking produces an edge image $g(x,y)$ from an input image $f(x,y)$ via

$$g(x,y) = f(x,y) - f_{smooth}(x,y)$$

where $f_{smooth}(x,y)$ is a smoothed version of $f(x,y)$.

We can better understand the operation of the unsharp sharpening filter by examining its frequency response characteristics. If we have a signal as shown in Figure 1(a), subtracting away the low pass component of that signal (as in Figure 1(b)), yields the highpass, or `edge', representation shown in Figure 1(c).

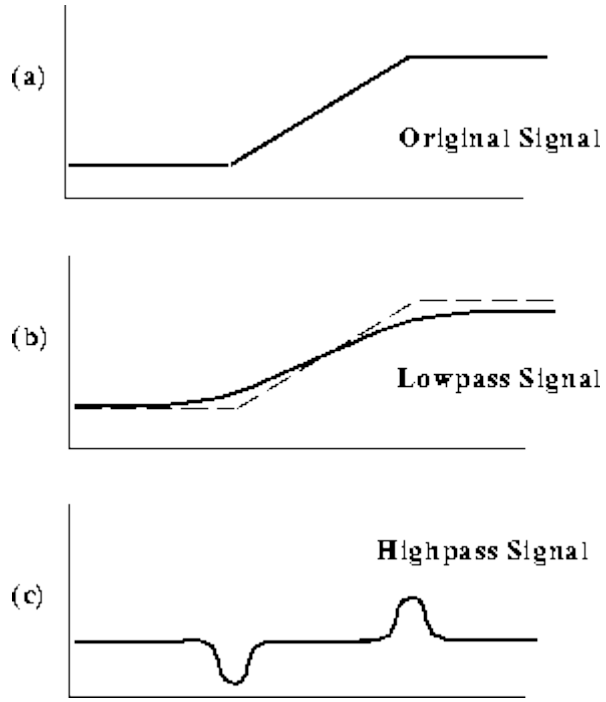

Figure 1 Calculating an edge image for unsharp filtering.

This edge image can be used for sharpening if we add it back into the original signal, as shown in Figure 2.

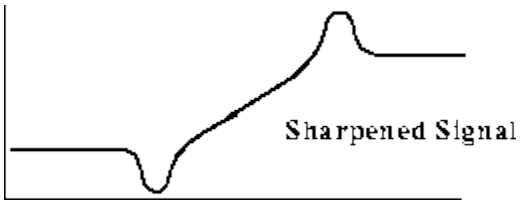

Figure 2 Sharpening the original signal using the edge image.

We can now combine all of this into the equation:

$$f_{sharp}(x, y) = f(x, y) + k * g(x, y)$$

where *k* is a scaling constant. Reasonable values for *k* vary between 0.2 and 0.7, with the larger values providing increasing amounts of sharpening.

Consider an image of your choice.

a) Perform unsharp sharpening on the raw image using a mean filter and a Gaussian filter (with the same kernel size). How do the sharpened images produced by the two different smoothing functions compare? **b)** Try re-sharpening this image using a filter with larger kernel sizes (*e.g.* 5×5, 7×7 and 9×9). How does increasing the kernel size affect the result? **c)** What would you expect to see if the kernel size were allowed to approach the image size?