Image Processing and Computer Vision 2007/08
Example Solutions for Practical Assignments 1 (P1)

Problem 1 (Noise Generation and Convolution)

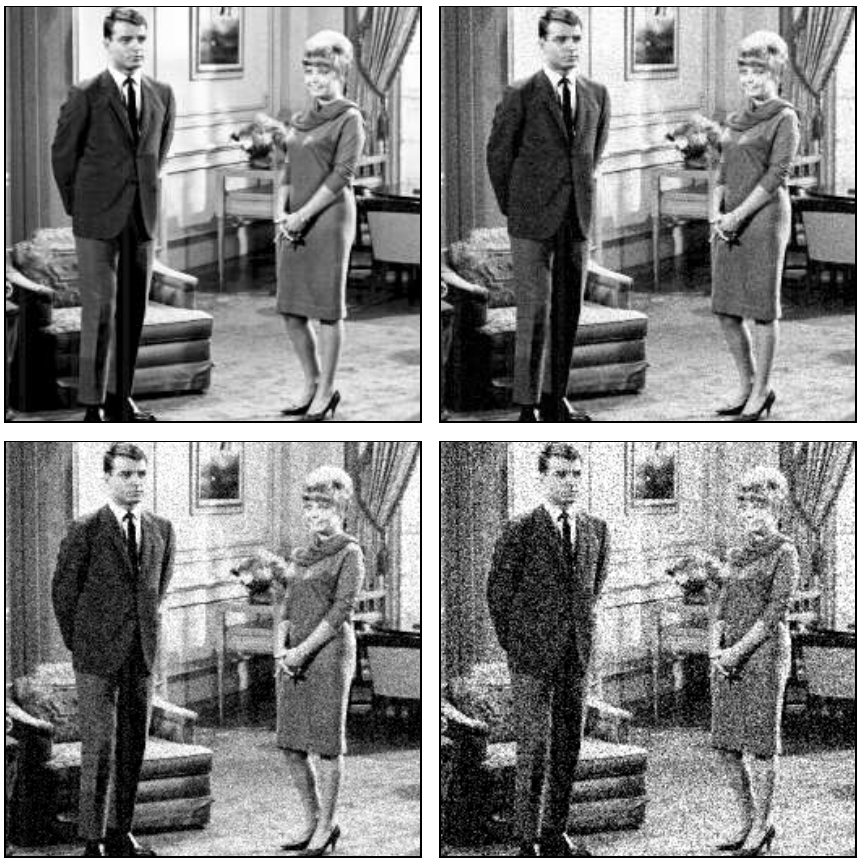
(a) The Box-Muller methods requires implementing the following code:

```
/* ---------------------------------------------------------- */
void gauss_noise

     (float  **f,   /* image, changed */
      long   nx,    /* pixel number in x direction */
      long   ny,    /* pixel number in y direction */
      float  sigma) /* standard deviation of Gaussian noise */
/*
 adds Gaussian noise with standard deviation sigma and mean 0 to
 the image f by means of the Box-Muller method.
*/

{
long    i, j;        /* loop variables */
float   pi;
float   x, y, z;     /* random numbers */
pi = 3.1415927;

for (i=1; i<=nx; i++)
 for (j=1; j<=ny; j++)
     {
     /* draw two [0,1]-uniformly distributed numbers x, y */
     x = (float)rand() / RAND_MAX;
     y = (float)rand() / RAND_MAX;
     /* assemble a N(0,1) number z according to Box-Muller */
     if (x > 0.0)
        z = sqrt (- 2.0 * log(x)) * cos (2.0 * pi * y);
     else
        z = 0.0;
     /* add Gaussian noise to data */
     f[i][j] = f[i][j] + sigma * z;
     }

return;
}
/* ---------------------------------------------------------- */
```

The noisy variants of the image `couple.pgm` for Gaussian noise of zero mean and standard deviation $\sigma = 10$, 20 and 40 are depicted below. The corresponding standard deviations of the images are given e.g. by

Noise Level	Std. Dev. of Image
0	75.27
10	75.94
20	77.91
40	85.30

As one can see, the variance of the images increases with the standard deviation of Gaussian noise that has previously been added.



Noisy variants of the image `couple.pgm`. *(a) Top left:* Original image. *(b) Top right:* Gaussian noise $\sigma = 10$. *(c) Bottom left:* Ditto for $\sigma = 20$. *(d) Bottom right:* Ditto for $\sigma = 40$.

(b) Adjusting the amount of Gaussian presmoothing yields the following denoising results:

Noisy variants of the image `couple.pgm`. *(a) Top left:* Original image. *(b) Top right:* Denoised image for $\sigma = 10$. *(c) Bottom left:* Ditto for $\sigma = 20$. *(d) Bottom right:* Ditto for $\sigma = 40$.

The corresponding values for the standard dev. of the Gaussian read:

Noise Level	Std. Dev. of Gaussian
10	0.80
20	1.00
40	1.50

Please note that these values are only adjusted visually (other solutions are possible as well). In order to justify the choice of such values, one could optimise them with respect to well-known quality measures such as the signal-to-noise ratio (SNR).

Problem 2 (Interpretation of the Fourier Spectrum)

(a) The image `pattern.pgm` contains a texture consisting of parallel sine waves.

The middle of the three points is located exactly in the centre. It corresponds to a frequency 0, i.e. to a constant component in the image (rescaled average grey value).

The other two points, located at positions symmetric to the centre, correspond to one single frequency namely that of the "wave" that fills the image.

A frequency here is a vector, having a direction and a norm. The direction corresponds to the "front propagation" direction of the wave (and is therefore perpendicular to the direction of the stripes) while the norm (distance to the centre) expresses the density of the periodic stripes.

There are two explanations why the points come in pairs. First, one verifies using the formula of the DFT that a real-valued image must have a point-symmetric Fourier spectrum (Fourier coefficients at symmetric positions are complex conjugates of each other). Second, it is impossible to distinguish frequencies which are equal up to the sign.

(b) **Cut-off errors.** Each of the three Gaussians is cut at the image boundaries, and treated as periodically repeated by the Fourier transform. While for the small kernels `gauss1.pgm` and `gauss2.pgm` the values at the image boundaries are close enough to 0 to leave no visible effects in the Fourier spectrum, the cutting off of fairly large values in `gauss3.pgm` spoils the rotational invariance of the Gaussian itself and therefore of its Fourier spectrum.

(c) The lines in `tile.pgm` induce in the Fourier spectrum visible beams starting off in the centre and directed perpendicular to the corresponding lines.

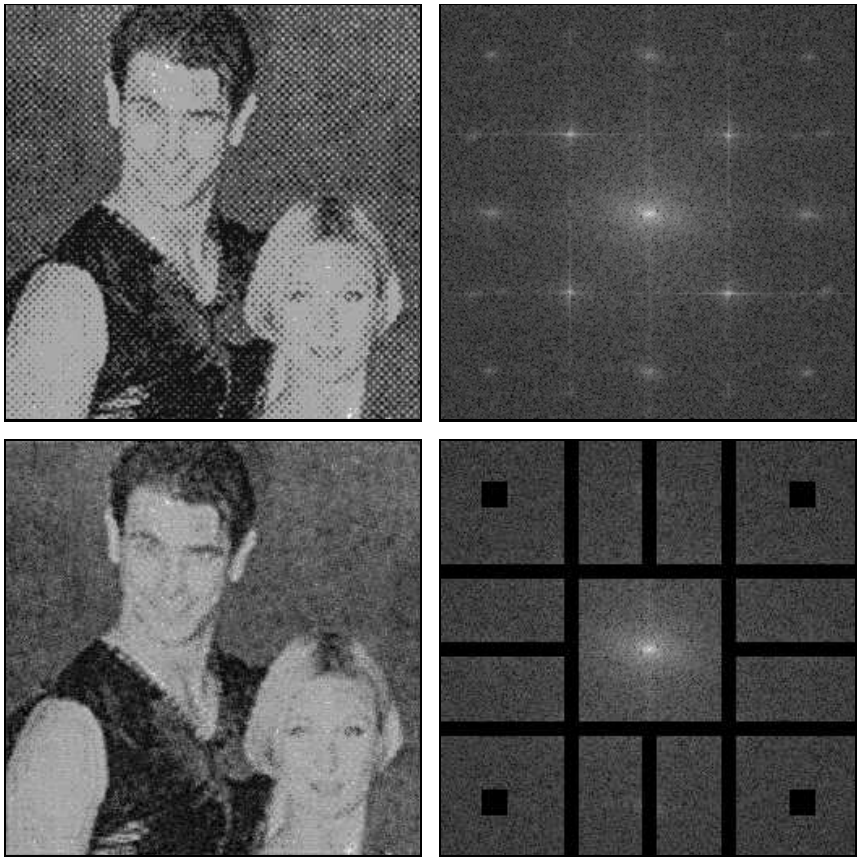
The reason is that a single edge in the spatial domain is represented by a superposition of wave-like patterns of different frequencies but equal direction. The phases and amplitudes of these waves are adjusted such that their slopes add up to give the edge at the specified location but cancel elsewhere.

A close look at the spectrum of `tile.pgm` reveals that there do not go out radially from the centre but from the image boundary. These are traces of aliasing effects.

Remembering that images and also Fourier spectra are treated periodically by the DFT we see that some of the radial lines extending from the centre do not end at the image boundaries but are prolongated beyond that boundary, wrapping around to the opposite image boundaries. Translated into frequencies: These lines depict high frequencies which don't fit in our Fourier spectrum but are represented in it by lower frequencies. This is aliasing.

Problem 3 (Filtering in the Fourier Domain)

(a) The fourier spectra of `dancing.pgm` show noticeable contributions in the high frequency range. Obviously these coefficients are related to the undesired pattern that has a relatively high frequency itself. By setting these coefficients zero the pattern can be removed successfully.

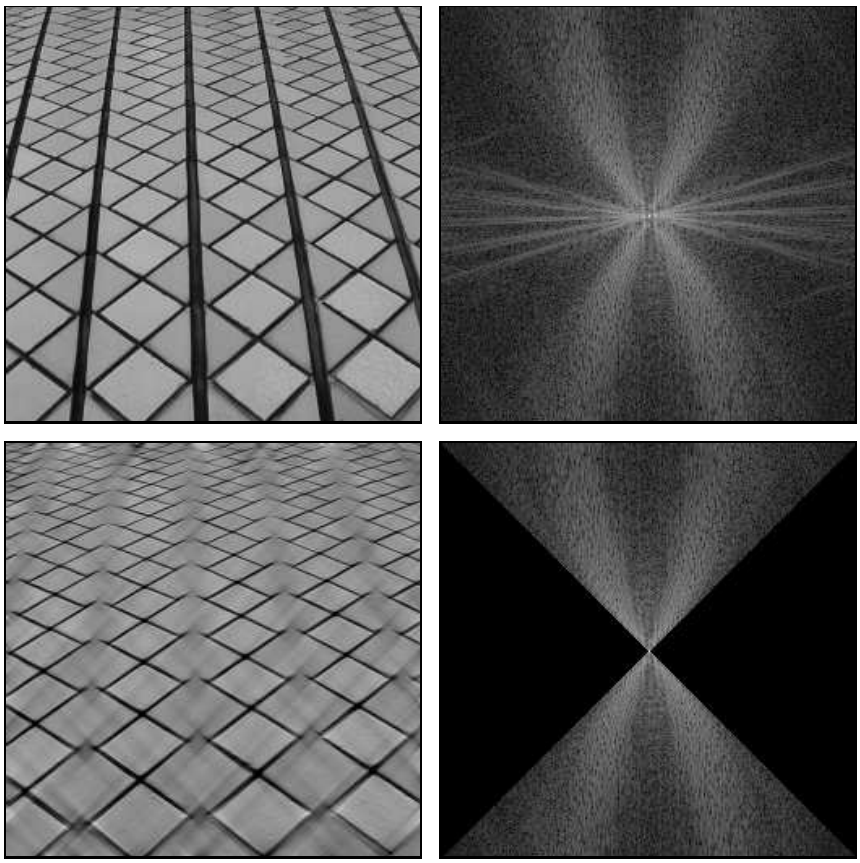


(a) Top left: Image `dancing.pgm`. *(b) Top right:* Fourier spectrum. *(c) Bottom left:* Filtered version. *(d) Bottom right:* Fourier spectrum of the filtered version.

```
/* ---------------------------------------------------------- */
 void filter_dancing

     (float    **ur,  /* real part of Fourier coeffs, changed */
      float    **ui,  /* imag. part of Fourier coeffs, changed */
      long     nx,    /* pixel number in x direction */
      long     ny)    /* pixel number in y direction */
/*
 allows to modify the Fourier coefficients
*/

{
long    i, j;         /* loop variables */

for (i=0; i<=nx-1; i++)
 for (j=0; j<=ny-1; j++)
     {
       /* long left vertical line */
       if ((i >= 76)  && (i <= 84))
       {
                 ur[i][j] = 0.0;
                 ui[i][j] = 0.0;
       }
       /* long right vertical line */
       if ((i >= 173) && (i <= 181))
       {
                 ur[i][j] = 0.0;
                 ui[i][j] = 0.0;
       }
       /* long upper horizontal line */
       if ((j >= 76)  && (j <= 84))
       {
                 ur[i][j] = 0.0;
                 ui[i][j] = 0.0;
       }
       /* long lower horizontal line */
       if ((j >= 173) && (j <= 181))
       {
                 ur[i][j] = 0.0;
                 ui[i][j] = 0.0;
       }
       /* two short vertical lines */
       if ((i >= 124) && (i <= 132) && ((j <= 76)  || (j >= 181)))
```

```
        {
                ur[i][j] = 0.0;
                ui[i][j] = 0.0;
        }
        /* two short horizontal ines */
        if ((j >= 124) && (j <= 132) && ((i <= 76)  || (i >= 181)))
        {
                ur[i][j] = 0.0;
                ui[i][j] = 0.0;
        }
        /* upper left square */
        if ((j >= 25) && (j <= 40) && ((i >= 25)  && (i <= 40)))
        {
                ur[i][j] = 0.0;
                ui[i][j] = 0.0;
        }
        /* upper right square */
        if ((j >= 25) && (j <= 40) && ((i >= 215)  && (i <= 230)))
        {
                ur[i][j] = 0.0;
                ui[i][j] = 0.0;
        }
        /* lower left square */
        if ((j >= 215) && (j <= 230) && ((i >= 25)  && (i <= 40)))
        {
                ur[i][j] = 0.0;
                ui[i][j] = 0.0;
        }
        /* lower right square */
        if ((j >= 215) && (j <= 230) && ((i >= 215)  && (i <= 230)))
        {
                ur[i][j] = 0.0;
                ui[i][j] = 0.0;
        }
      }
}
/* ------------------------------------------------------------ */
```

(b) The dominant black lines (vertical) are represented by the five horizontal lines in the Fourier spectrum. By setting the corresponding coefficients in the Fourier domain zero, the dominant lines in the original image disappear.



(a) Top left: Image `tile.pgm`. *(b) Top right:* Fourier spectrum. *(c) Bottom left:* Filtered version. *(d) Bottom right:* Fourier spectrum of the filtered version.

```
/* ------------------------------------------------------------ */
void filter_tile

     (float    **ur,  /* real part of Fourier coeffs, changed */
      float    **ui,  /* imag. part of Fourier coeffs, changed */
      long     nx,    /* pixel number in x direction */
      long     ny)    /* pixel number in y direction */
/*
 allows to modify the Fourier coefficients
*/

{
long    i, j;         /* loop variables */

for (i=0; i<=nx-1; i++)
```

```
 for (j=0; j<=ny-1; j++)
     {
       /* set left triangle zero */
       if ((i < 128) && (i-128 < j-128) && (i-128 < -j+128))
       {
                 ur[i][j] = 0.0;
                 ui[i][j] = 0.0;
       }

       /* set right triangle zero */
       if ((i > 128) && (128-i < 128-j) && (128-i < j-128))
       {
                 ur[i][j] = 0.0;
                 ui[i][j] = 0.0;
       }
     }
}
/* ------------------------------------------------------------ */
```
