Differential Equations in Image Processing and Computer Vision 2008

Example Solutions for Programming Assignments 2 (P2)

Problem 1

The complete file `isonondiff.c` with supplemented code reads

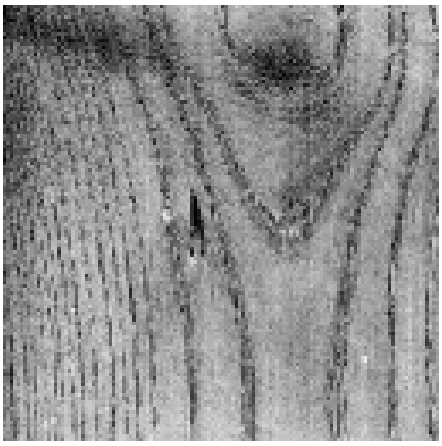
```
/* ---------------------------------------------------------- */
void isonondiff

     (float    ht,          /* time step size, 0 < ht <= 0.25 */
      long     nx,          /* image dimension in x direction */
      long     ny,          /* image dimension in y direction */
      float    hx,          /* pixel size in x direction */
      float    hy,          /* pixel size in y direction */
      float    lambda,      /* contrast parameter */
      float    sigma,       /* noise scale */
      float    **u)         /* input: original image;  output: smoothed */


/*
 Isotropic nonlinear diffusion.
 Explicit discretization.
*/

{
long    i, j;             /* loop variables */
float   rxx, ryy;         /* time savers */
float   **f;              /* work copy of u */
float   **dc;             /* diffusion coefficient */
float   df_dx, df_dy;     /* derivatives */
float   two_hx, two_hy;   /* time savers */
float   grad;             /* |grad(v)| */


/* ---- allocate storage ---- */

alloc_matrix (&f,  nx+2, ny+2);
alloc_matrix (&dc, nx+2, ny+2);


/* ---- copy u into f ---- */
```

```
for (i=1; i<=nx; i++)
 for (j=1; j<=ny; j++)
     f[i][j] = u[i][j];


/* ---- regularize f ---- */

if (sigma > 0.0)
   gauss_conv (sigma, nx, ny, hx, hy, 3.0, 1, f);


/* ---- calculate diffusivity ---- */

two_hx = 2.0 * hx;
two_hy = 2.0 * hy;
dummies (f, nx, ny);

for (i=1; i<=nx; i++)
 for (j=1; j<=ny; j++)
     {
        /* calculate grad(f) */

        df_dx = (f[i+1][j]-f[i-1][j])/two_hx;
        df_dy = (f[i][j+1]-f[i][j-1])/two_hy;
        grad = sqrt(df_dx*df_dx+df_dy*df_dy);

        /* calculate diffusivity dc */

        // Weickert diffusivity
        if ((grad*grad)==0)
             dc[i][j] = 1;
        else
             dc[i][j] = 1-exp(-3.31488/pow((grad/lambda),8));

// Perona-Malik diffusivity
        // dc[i][j] = 1.0 / (1.0 + (grad*grad) / (lambda*lambda) );

        // Charbonnier diffusivity
        // dc[i][j] = 1.0 / sqrt(1.0 + (grad*grad) / pow(lambda,2.0) ) ;
     }
```

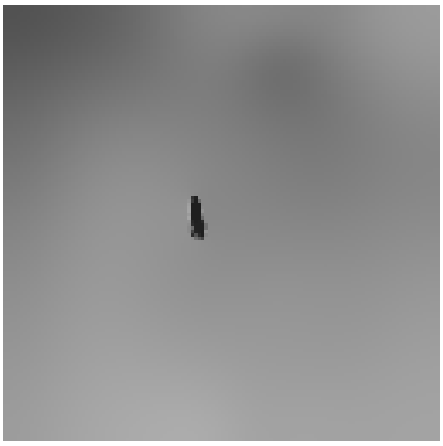
```
/* ---- calculate explicit nonlinear diffusion of u ---- */

/* copy u into f */
for (i=1; i<=nx; i++)
 for (j=1; j<=ny; j++)
     f[i][j] = u[i][j];

/* dummy boundaries */
dummies (f, nx, ny);
dummies (dc, nx, ny);

/* diffuse */
rxx = ht / (2.0 * hx * hx);
ryy = ht / (2.0 * hy * hy);
for (i=1; i<=nx; i++)
 for (j=1; j<=ny; j++)
     u[i][j] = f[i][j]
        + rxx * ( (dc[i+1][j] + dc[i][j]) * (f[i+1][j] - f[i][j])
                + (dc[i-1][j] + dc[i][j]) * (f[i-1][j] - f[i][j]) )
        + ryy * ( (dc[i][j+1] + dc[i][j]) * (f[i][j+1] - f[i][j])
                + (dc[i][j-1] + dc[i][j]) * (f[i][j-1] - f[i][j]) );


/* ---- disallocate storage ---- */

disalloc_matrix (f,  nx+2, ny+2);
disalloc_matrix (dc, nx+2, ny+2);

return;

} /* isonondiff */

/* ------------------------------------------------------- */
```

Problem 2

In order to isolate the defect in the wooden surface we apply 600 iterations with $\lambda = 7$, $\sigma = 1$ and $\tau = 0.2$. The obtained results look as follows:



original



$t = 1$ (τ=0.2, n_{iter}=600)

Problem 3

In order to denoise the image we first apply the Perona-Malik diffusivity and try out several combinations of parameters. We have used a time step size of $\tau = 0.25$ here. Some examples for the corresponding results can be seen here:

original

$t = 25$, $\lambda = 1$, $\sigma = 1.5$

$t = 25$, $\lambda = 3$, $\sigma = 1.5$

$t = 25$, $\lambda = 10$, $\sigma = 1.5$

$t = 25$, $\lambda = 3$, $\sigma = 6$

$t = 25$, $\lambda = 3$, $\sigma = 0.5$

As one can see, a smaller value for σ leads to an adaptation of the contrast enhancement to the noise (the computation of the gradient that serves as argument of the diffusivity function g is very sensitive to noise).

As a consequence the image is not denoised properly. But also a too large value for σ gives non-optimal results. Important edges are not respected any longer by the contrast enhancement any longer, sincer they are reduced significantly by the Gaussian smoothing via a kernel of standard deviation σ.

The change of the contrast parameter λ has a similar effect. If we decrease λ even very small structures (and thus even a small amount of noise) lead to backward diffusion and thus to contrast enhancement. Thus also parts of the noise are enhacned. If one choses λ too large forward diffusion is applied to noise but also to many important edges in the image.

As a good choice of parameters, we would propose $t = 25$, $\lambda = 5$, and $\sigma = 1.5$.

Problem 4

We evalute the usefulness of the Charbonnier diffusivity for denoising and contrast enhancement. To this end, we perform 100 iterations ($\tau = 0.25$) with $\lambda = 1$, $\sigma = 1.5$.

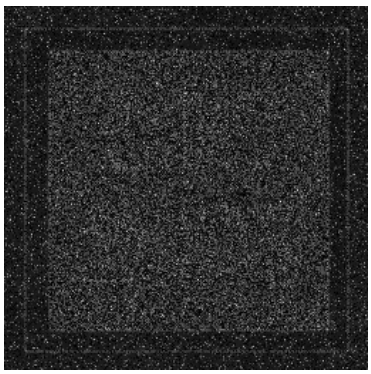
original

$t = 25$, $\lambda = 1$, $\sigma = 1.5$

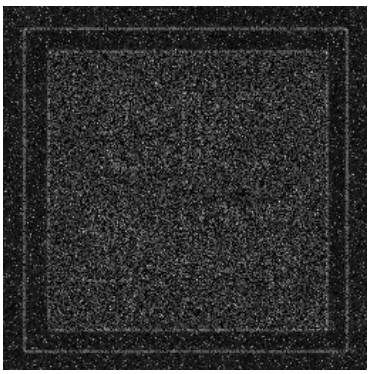
As one can see, this diffusivity removes the noise quite well, but blurs discontinuities slightly. This is not surprising, since the Charbonnier diffuisivity performs only forward diffusion. Thus it adapts less to noise than the Perona-Malik diffusivity from Problem 3. However, as a consequence, it also does not offer contrast enhancement which requires backward diffusion.

Problem 5

We display the method noise for the two best results shown above:



Method noise, Perona-Malik
$t = 25$, $\lambda = 5$, $\sigma = 1.5$



Method nosie, Charbonnier
$t = 25$, $\lambda = 1$, $\sigma = 1.5$

It is clearly visible that the Charbonnier process removes more information at the image edges than the Perona-Malik filter.