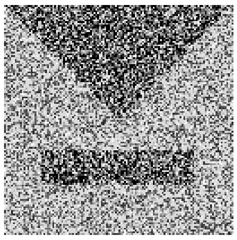
Differential Equations in Image Processing and Computer Vision 2008
Example Solutions for Programming Assignments 6 (P6)

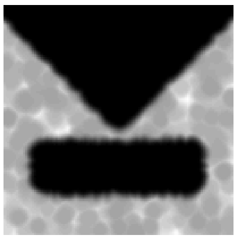
Problem 1

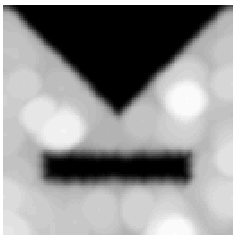
In the file `erosion.c` the routine `erosion` had to be supplemented with code for the Osher-Sethian scheme. The complete routine `erosion` reads

```
/* ------------------------------------------------------ */
void erosion

     (float    ht,        /* time step size, 0 < ht <= 0.707 */
      long     nx,        /* image dimension in x direction */
      long     ny,        /* image dimension in y direction */
      float    hx,        /* pixel width in x direction */
      float    hy,        /* pixel width in y direction */
      float    **u)       /* input: original image ;  output: smoothed */

/* Osher-Sethian 2-D erosion. */

{
long    i, j;                   /* loop variables */
float   **f;                    /* u at old time level */
float   fxp, fxm, fyp, fym;     /* one-sided differences */


/* ---- allocate storage f ---- */
alloc_matrix (&f, 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];


/* ---- create reflecting dummy boundaries for f ---- */
dummies (f, nx, ny);


/* ---- loop ---- */
for (i=1; i<=nx; i++)
 for (j=1; j<=ny; j++)
     {
     /* one-sided spatial derivatives */
     fxp = (f[i+1][j] - f[i][j]) / hx;
     fxm = (f[i][j] - f[i-1][j]) / hx;
     fyp = (f[i][j+1] - f[i][j]) / hy;
     fym = (f[i][j] - f[i][j-1]) / hy;

     /* SUPPLEMENT CODE */
     /* evolution */
     u[i][j] = f[i][j] - ht * sqrt (   sqr(max(fxm,0.0))
                                     + sqr(min(fxp,0.0))
                                     + sqr(max(fym,0.0))
                                     + sqr(min(fyp,0.0)) );
     }
/* ---- disallocate storage for f ---- */
disalloc_matrix (f, nx+2, ny+2);

return;

} /* erosion */
/* ------------------------------------------------------- */
```

Moreover, we had to apply fisrt erosion and then dilation with $t = 9$ to the image `noise.pgm`. Below the original image as well as the two results (rescaled) are shown.

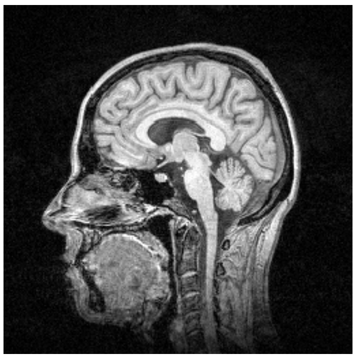

`noise.pgm`

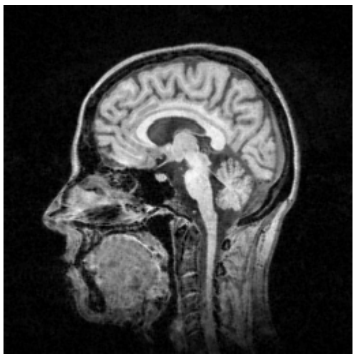

erosion

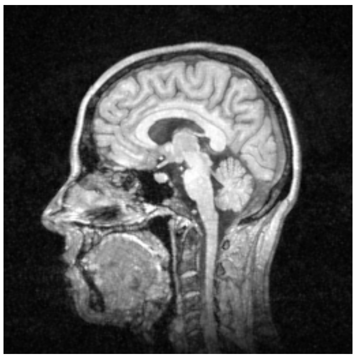

erosion + dilation

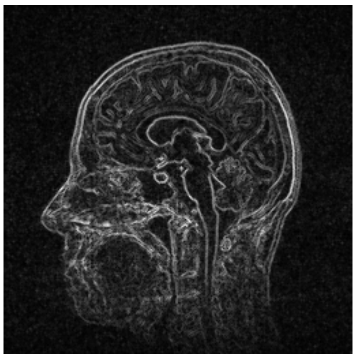
As one can, see the noise is suppressed quite well. However, the restoration is far from being perfect.

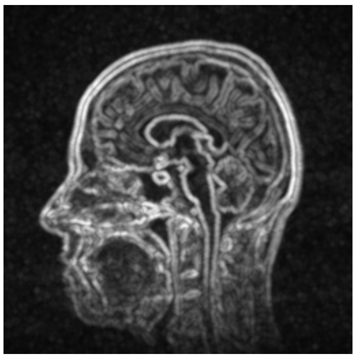
In our second experiment, we apply both erosion and dilation with different evolution times t to the image `head.pgm`. The obtained results as well as the difference between both – the so-called morphological gradient – is shown below.

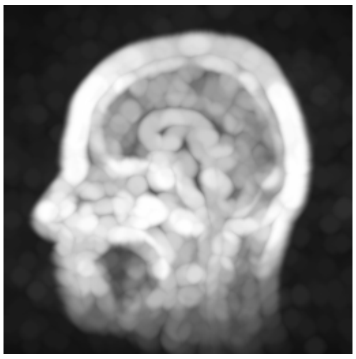

`head.pgm`


erosion


dilation


morph. gradient $t = 0.5$

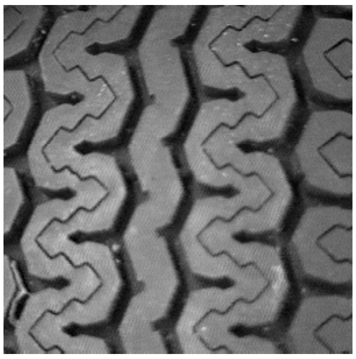

morph. gradient $t = 2$

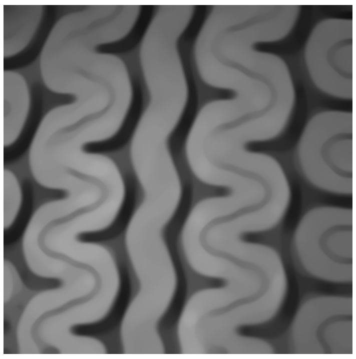

morph. gradient $t = 8$

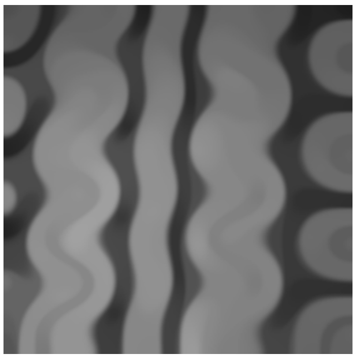
As one can clearly see, the morphological gradient is able to capture all edge information of the original image (just like the ordinary gradient). Moreover, one can observe that the morphological gradient becomes larger in its spatial extension if the evolution time is chosen larger.

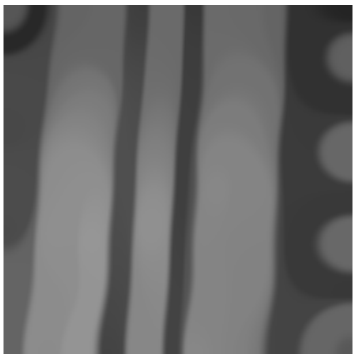
Problem 2

In our second task we had to apply mean curvature motion (MCM) with different evolution times to the image `tyre.pgm`. The obtained results are given by:


`tyre.pgm`

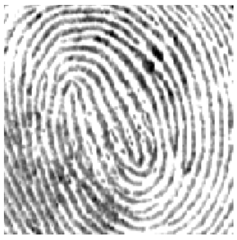

MCM $t = 20$

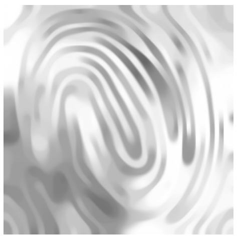

MCM $t = 100$


MCM $t = 400$

As the images show, the snake like pattern on the tyre is straightened. Moreover, small details disappear very fast.

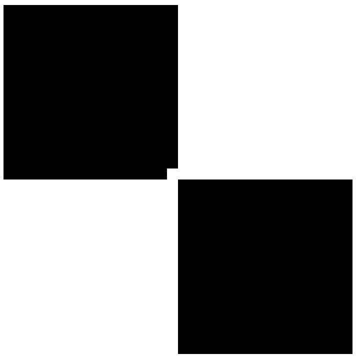
For processing fingerprints, MCM is not suitable, since it does not allow to connect isolated lines in direction of their orientation. Isolated lines will be shrunken to points instead and finally disappear. The computed results for applying MCM with different evolution times to the image `finger.pgm` is shown below.

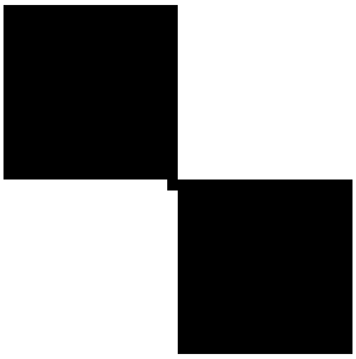

`finger.pgm`

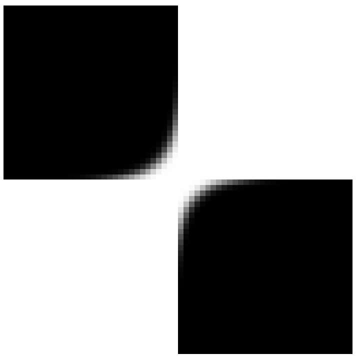

MCM $t = 50$

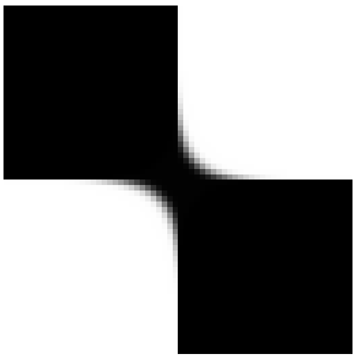
MCM $t = 200$

Finally, we had to investigate the behavior of MCM with respect to the two images `check1.pgm` and `check2.pgm`. The two evolution for $t = 20$ can be seen below.


check1.pgm


check2.pgm


MCM $t = 20$


MCM $t = 20$

The behavior is not surprising, since in the case of `check1.png` the white squares are connected, while the case of `check2.png` the black squares are connected. These component stay connected. The two squares of the other color, however, are not connected. Following the rules of MCM, they are convexified and shrink to points separately.

Problem 3

In the third problem we try to measure corner angles with the affine morphological scale space and the method by Alvarez and Mazorra.

We use the programme `corner_detect.c` and apply the explicit scheme with time step size $\tau = 0.01$. Applying this to the image `corner01.pgm` with several different stopping times yields:

Number of steps	Stopping time	Velocity	Angle
1000	10	0.876	105.0
2000	20	0.930	98.3
5000	50	0.959	94.7
10000	100	0.971	93.4

We see that the programme usually over-estimates the angle. (The correct angle is $90°$.) Increasing the number of time steps makes the least-squares fit better and allows for a better result.

Similar observations are made for the image `corner02.pgm` where the correct angle is $53.1°$:

Number of steps	Stopping time	Velocity	Angle
1000	10	1.335	58.6
2000	20	1.365	56.5
5000	50	1.379	55.5
10000	100	1.386	55.0

In a second step, we try out how robust the process is with respect to noise. The resulting angles estimates for `corner01_50.pgm` are:

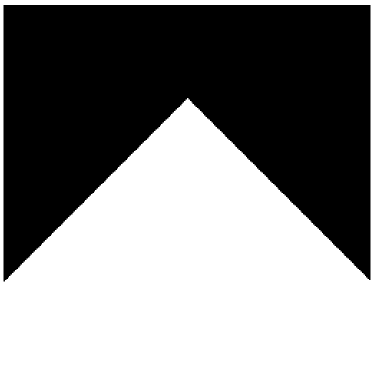
Number of steps	Stopping time	Velocity	Angle
1000	10	0.965	94.1
2000	20	0.957	95.0
5000	50	0.969	93.7
10000	100	0.976	92.8

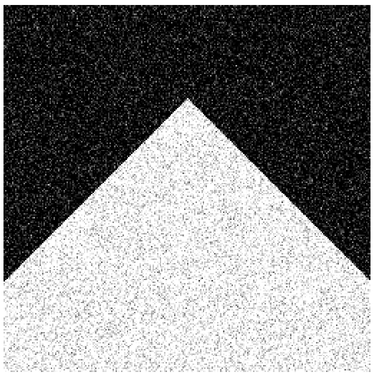
For `corner02_50.pgm` we obtain:

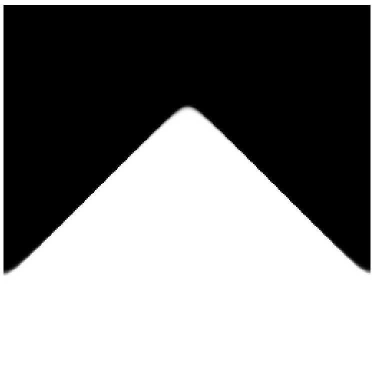
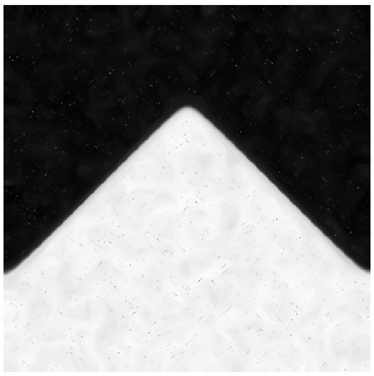
Number of steps	Stopping time	Velocity	Angle
1000	10	1.588	43.3
2000	20	1.416	53.0
5000	50	1.368	56.2
10000	100	1.378	55.6

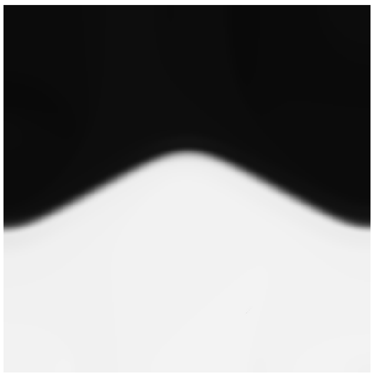
We conclude that the process is very robust with respect to noise. On the one hand, this comes from the least-squares fit that naturally increases the robustness. On the other hand, the affine morphological scale-space removes the noise relatively fast.

We show some examples of the evolution for `corner01.pgm` and its noisy version:


original image `corner01.pgm`


with Gaussian noise, $\sigma = 50$



stopping time $t = 10$


stopping time $t = 100$