CS 7690, Advanced Image Processing
Project2 Anisotropic Diffusion
Xiang Hao

Implement the PDE of the anisotropic diffusion

For one dimension:
The PDE of the anisotropic diffusion is $I_t = d(c(x, t) I_x)/dx$

$d(c(x,t)I_x/dx = (c_r * I_r - c_l * I_l)/\Delta x$ ....(1)
$c_r * I_r - c_l * I_l = c_r * (I(x + \Delta x) - I(x))/\Delta x - c_l * (I(x) - I(x-\Delta x))/\Delta x$ ....(2)
From (1) and (2), we can get:
$d(c(x,t)I_x/dx = (1/\Delta x)^2 * (c_r * (I(x + \Delta x) - I(x)) - c_l * (I(x) - I(x-\Delta x)))$

Since $I_t = (I(t+\Delta t) - I(t))/\Delta t$, so
$I(t+\Delta t) = I(t) + \Delta t((1/\Delta x)^2 * (c_r * (I(x + \Delta x) - I(x)) - c_l * (I(x) - I(x-\Delta x))))$ ....(a)

From the above equation, we know that, we can compute $I(t)$ by an iteration.
At $t = 0$, $I(0) = I$, I is the original image.
At $t = 0 + \Delta t)$, we compute $I(t + \Delta t)$ by using the above equation.
We do the above step over and over again, the number of the iteration is decided by the user.
At each step, before we compute $I(t + \Delta t)$, we need to compute c and the gradient at each point.

For two dimension:
The equation (a) becomes:
$I(t+\Delta t) = I(t) + \Delta t((1/\Delta x)^2 * (c_{x\_r} * (I(x + \Delta x) - I(x)) - c_{x\_l} * (I(x) - I(x-\Delta x))) + (1/\Delta y)^2 * (c_{y\_r} * (I(y + \Delta y) - I(y)) - c_{y\_l} * (I(y) - I(y-\Delta y))))$

We compute $I(t)$ in the same way as the was we compute $I(t)$ in one dimension.

Description of the conductivity function

Usually, the conductivity function is $\exp-(|I_x|/K)^2$ or $1/(1 + |I_x|^2/K)$.

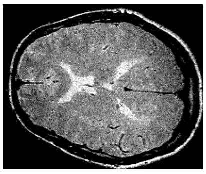
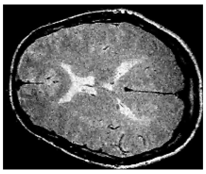
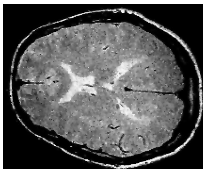
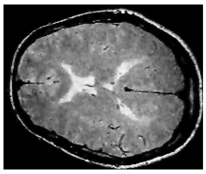
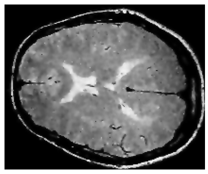
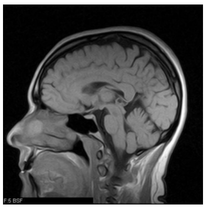
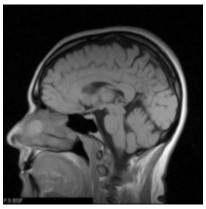
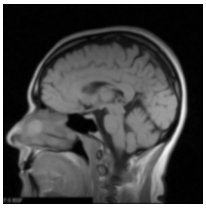
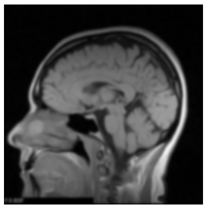
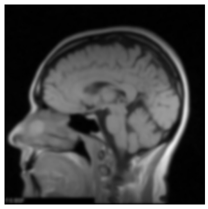
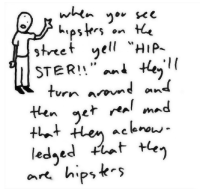
If we treat I_x as a variable, K as a constant:
Both of the functions are monotone decreasing, which means:
If a region of a image has a higher gradient, it will suffer a smaller diffusion. On there other hand, if the region has a lower gradient, it will suffer a higher diffusion.
So, use these conductivity function will preserve the features with high gradients.

In a different view point, if we treat K as a variable, the diffusion degree of each pixel will increase as the increase of K.

Here I choose the function $1/(1 + |I_x|^2/K)$ since it is much faster than the first one.

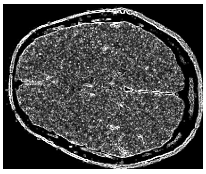
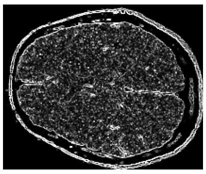
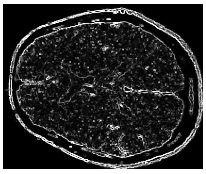
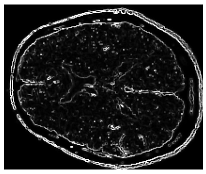
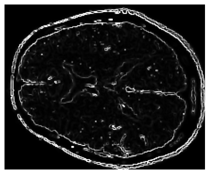
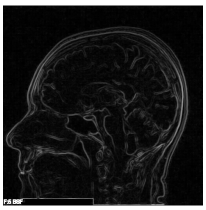
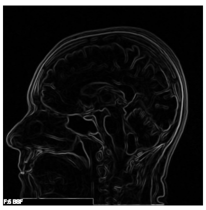
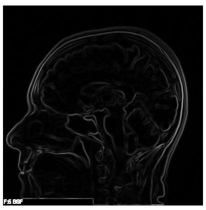
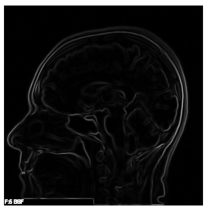
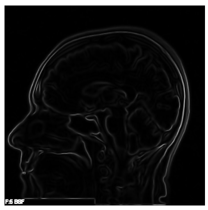
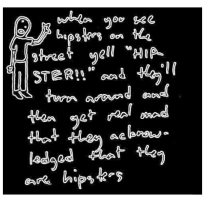
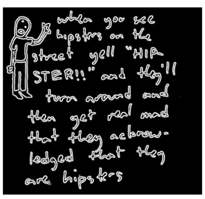
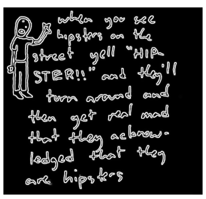
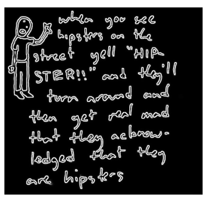
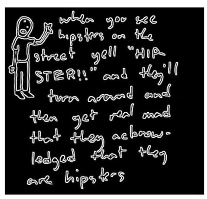
Application

The diffusion image:

Kappa = 20	Iteration1	Iteration5	Iteration10	Iteration15	Iteration20
Noisy MRI					
MRI					
Text					

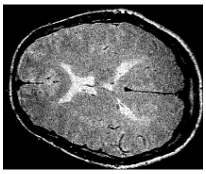
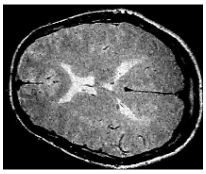
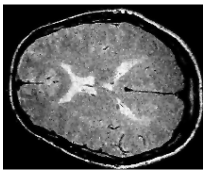
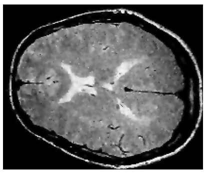
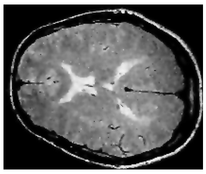
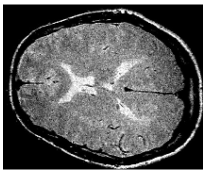
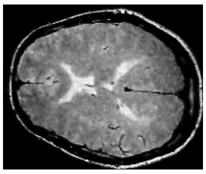
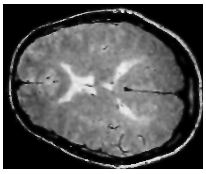
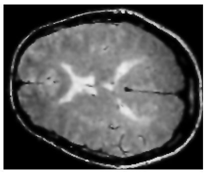
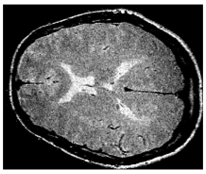
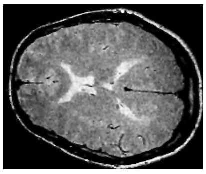
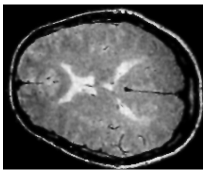
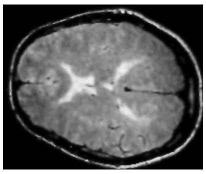
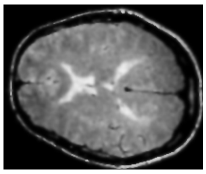
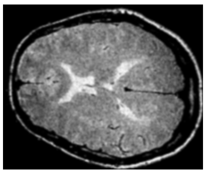
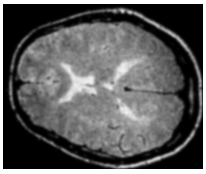
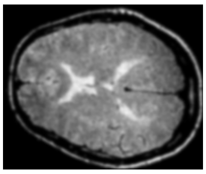
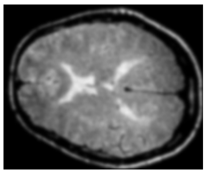
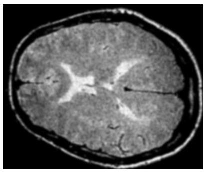
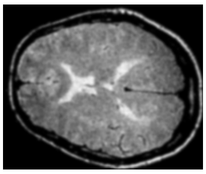
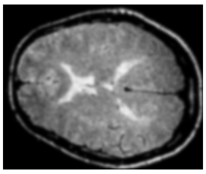
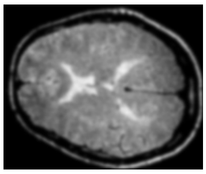
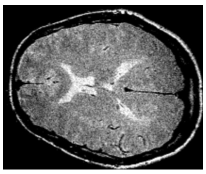
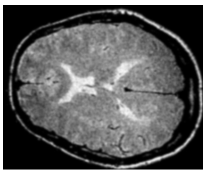
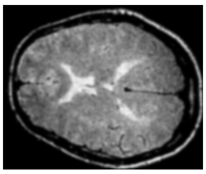
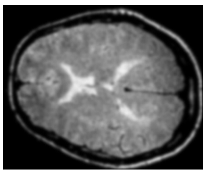
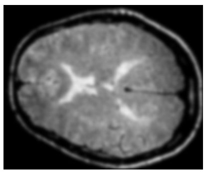
From the above pictures, we can see in the anisotropic diffusion, the edges are preserved and the flat area and noises are blurred.

The gradient images:

Kappa = 20	Iteration1	Iteration5	Iteration10	Iteration15	Iteration20
Noisy MRI					
MRI					
Text					

From the gradient images, we can see the area with high gradients will not change two much. The area with low gradients will become darker and darker. This also explains why the anisotropic diffusion can preserve the edges.

Performance under different k-values

	Iteration1	Iteration5	Iteration10	Iteration15	Iteration20
Kappa = 20					
Kappa = 35					
Kappa = 50					
Kappa =300					
Kappa = 3000					
Kappa = 300000					

Compare the above two results:

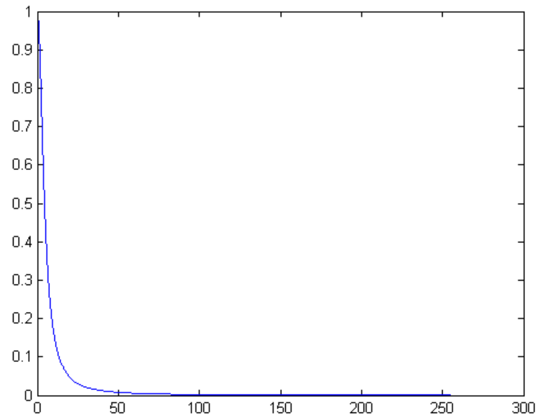
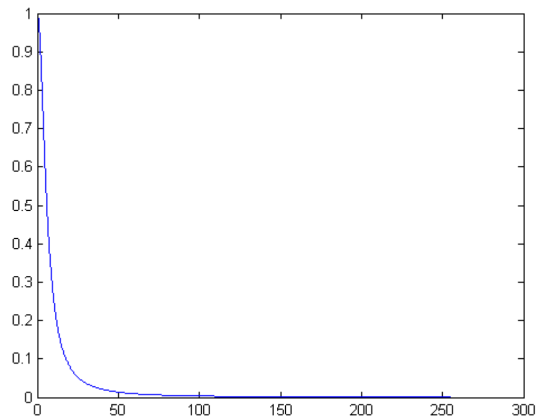
When Kappa = 30, we preserve more edges since the larger the gradient is, the less diffusion it will suffer.

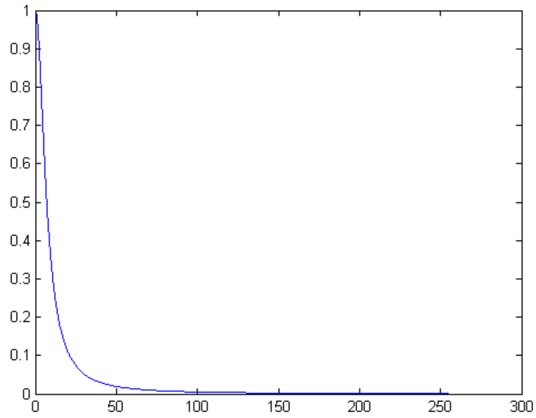
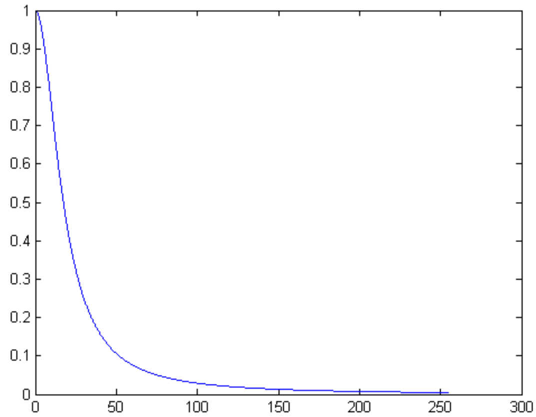
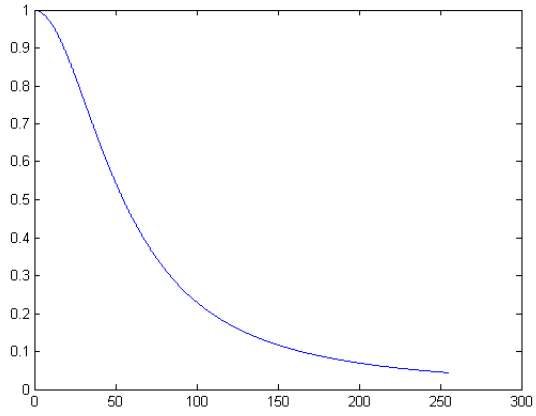
When Kappa is very large, the whole image suffer the similar diffusion, so it is similar a linear diffusion in this case.

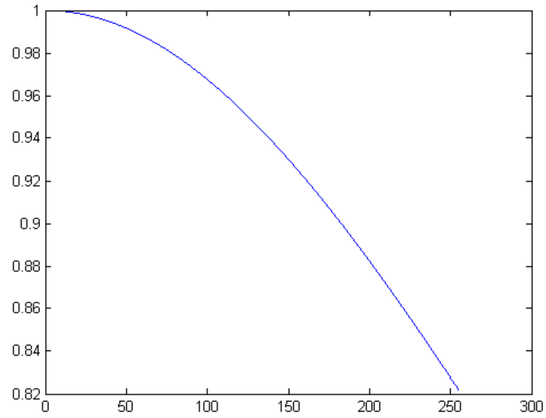
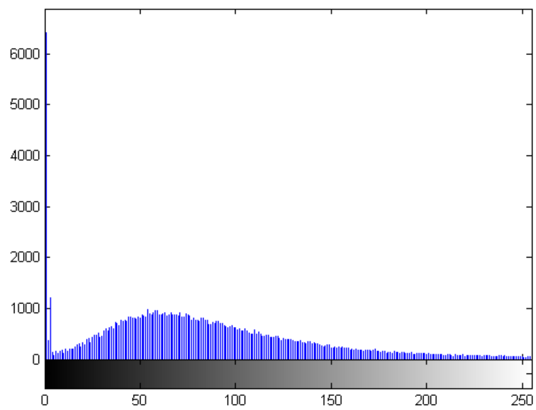
In addition, the images, especially the CSF boundary, do not change too much from kappa = 300 to kappa = 300000. Even form kappa = 50 to kappa = 300, the changes are not very significant. The reasons are:

1) The diffusion degree is increasing as the increases of the kappa.
2) As kappa is increasing. For some features(For example, the csf edges in the above pictures), before kappa goes to certain value(for the edges of the above picture, I guess the value is around 25), the features will have few diffusion, since their gradients are large enough. But, as the kappa increases larger and larger, the features' gradients are not big enough to get few diffusion, so these features will be blurred.

I plot the conductivity functions here:

Kappa	Conductivity function
20	
35	

50	1 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0 0 50 100 150 200 250 300
300	1 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0 0 50 100 150 200 250 300
3000	1 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0 0 50 100 150 200 250 300

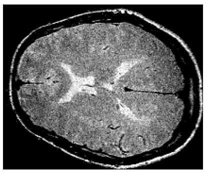
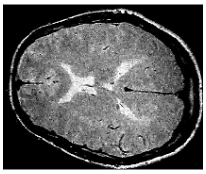
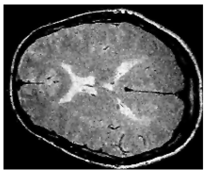
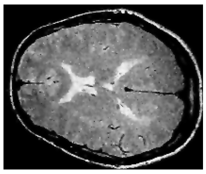
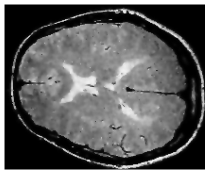
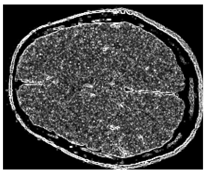
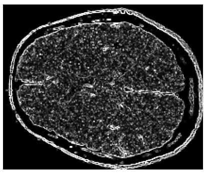
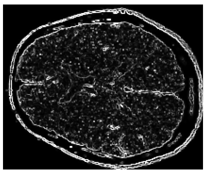
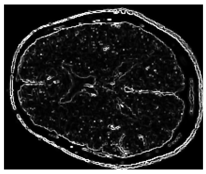
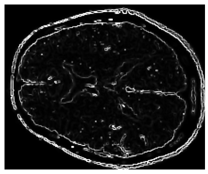
<table>
<tr>
<td>300000</td>
<td>
</td>
</tr>
<tr>
<td>Histogram of the gradient(Itertaion #1)</td>
<td>
</td>
</tr>
</table>

From the above pictures,
1)We can see that when kappa = 300000, the range of the y axis(the amount of the diffusion) is 0.82~1, when the gradient is less than 255. That's why the anisotropic diffusion is similar with isotropic diffusion when kappa is very large

2)From the histogram of the gradients, we know that most of the gradients are less than 100. When kappa is larger 300, the c(I_x > 100) we get from the conductivity function is become larger and larger, which means most of the image will be blurred. This also explains why the images does not change too much from kappa = 300 to kappa = 300000.

3)Base on the histogram of the gradients, we can come up a way to automatically choose the kappa parameter.

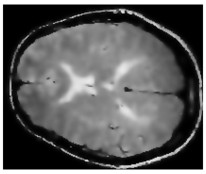
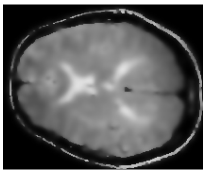
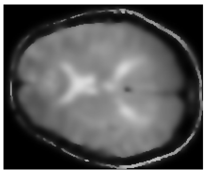
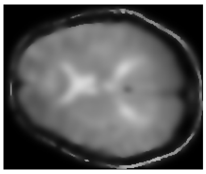
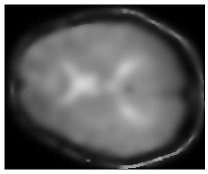
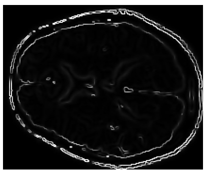
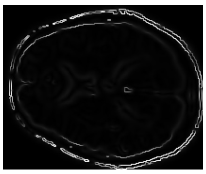
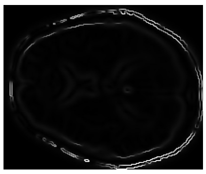
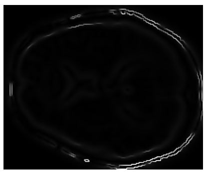
Performance under different iteration numbers

Kappa = 20	Iteration1	Iteration5	Iteration10	Iteration15	Iteration20
Noisy MRI					
Gradient					

In the Noisy MRI images, as the number of iteration increases, the boundary does not change too much, but the noise is blurred as the number of iteration increases.
In the gradient images, the boundary, which is the very bright, does not change too much either. The dark areas become darker and darker.

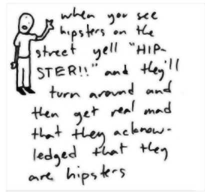
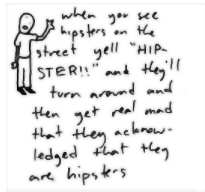
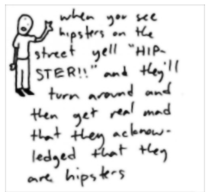
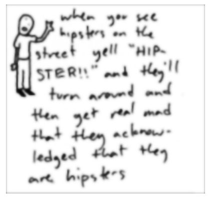
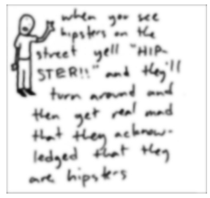
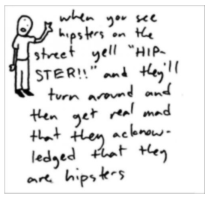
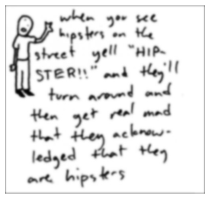
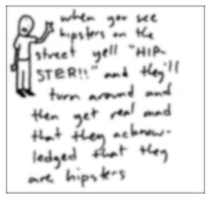
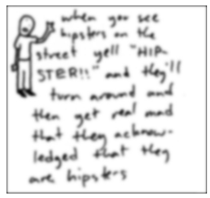
So, when kappa is equal to 20, as the number of iteration increases, the edges(boundaries) will be preserved and the noises and flat areas will be blurred.

However, I am wondering if the edges will also be blurred when the number of the iteration goes to infinite. So I increased the maximum of the number of the iteration from 20 to 200.

Kappa = 20	Iteration50	Iteration80	Iteration120	Iteration150	Iteration200
Noisy MRI					
Gradient					

From the above results, obviously everything is blurred as the number of iteration goes to a very larger number.
We can also guess that as the number of iteration goes to infinite, every pixel of the image will have the same intensity.

Discussion of preservation of details

	Iteration1	Iteration5	Iteration10	Iteration15	Iteration20
Kappa = 20					
Kappa = 40					
Kappa = 80					
Kappa = 150					

As the kappa increases, the overall diffusion is increasing. We may use this property to segment the region of the each word first by using large kappa and then segment the words into characters.

In order to segment the text into words, we can also use a lot of iteration.

The conductivity function I choose here is a monotone decreasing function. This kind of functions preserve pixels with high gradient and blur pixels with low gradient. It is kind of a "high pass filter".

Similarly, we can also construct some "low pass filter" and "band pass filter" if we need.
For example, by band pass filter, I mean we can set the conductivity function to be a Gaussian(mu, sigma) function. In this case, the pixels with the gradient as mu will be preserved.