Code Structure

Tensor Voting:

main ==> pixel2tensor ==> vote_kernel ==> get_eig ==> main (generate the result)

Steerable Filter:

main ==> steerable_filter ==> main (generate the result)

Steerable Filter with Tensor Voting:

main ==> steerable_filter ==> steer_tv_kernel ==> main (generate the result)

Tensor Voting

1) vote_kernel

Because I am using a fixed kernel size, I can pre-define the relationships between a targeted pixel and its neighbour. Let's say I decide to use a 3x3 kernel. The relationship is described as this:

-1, -1	0, -1	1, -1
-1, 0	0, 0	1, 0
-1, 1	0, -1	1, 1

Based on this relationship, I can calculate theta, then get parameters for decay function and the projection matrix. The SV projection matrix is 2x2 and is calculated as N' * N ==> $[-\sin 2\theta,\ \cos 2\theta]' * [-\sin 2\theta,\ \cos 2\theta]$.

TV = SV + BV for images, where BV = $\int$ SV dθ

It is impractical to integrate for DF, so we only integrate for the projection matrix. The kernel like this: TV = DF * [N' * N + $\int$ (N' * N) dθ]

Then, I stretch it into a matrix. Each p component is a 2x2 projection matrix. How many of them is determined by the kernel size. For a 3x3 kernel, there are 9 projection matrix.

$$[[p_1][p_2][p_3] \cdots\cdots [p_{n^2}]]$$

2) pixel2tensor

Two main steps in this function: 1) convert a pixel value into a 2x2 tensor, 2) get a targeted pixel's neighbour and apply the kernel.

If I decide to use a 3x3 kernel, for each targeted pixel I will get 9 tensors. Then apply the tensor to its according projection matrix, and sum up to become a new tensor. The output is M x N x 2 x 2

$$\begin{array}{ccccc} [[p_1] & [p_2] & [p_3] & \cdots\cdots & [p_{n^2}]] \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\ [[t_1] & [t_2] & [t_3] & \cdots\cdots & [t_{n^2}]] \end{array}$$

3) get_eig

It takes the output of the pixel2tensor as its input. The outcome is a M x N matrix.

Steerable Filter with Tensor Voting

1) steerable_filter
this function can generate two different results

1) steerable filter
2) steerable filter with tensor voting

The 4^{th} and 5^{th} input are for the interpolation function. If they are same number, it refers to use a steerable filter with tensor voting. Otherwise, it refers to use a regular steerable filter. The output of the function is a M x N matrix.

The steerable filter is very straight forward, no need to explain the code. For the steerable filter with tensor voting, I did not use an interpolation function. After applying a Gaussian filter, I use the projection matrix to connect with the [Gx   Gy]. I did not apply eigen decomposition, but simply sum up the result to generate a M x N matrix. Interestingly, the final result is not much different from the regular tensor voting technique.