Appendix 2. 1. Motion generalization

The dominant motion can be generalized to a smooth motion with changing speed and curved path. The basic idea is that the smooth motion can be modeled as piece-wise linear motions, and the 3D model will be a view-based representation along the (curved) path. The content of this appendix is related to both Section 2.2 and Section 2.3, so it is advised to read this appendix after you finish reading these two sections.

1. Translation with varying speed

The basic vehicular motion model is based on the assumption that a vehicle translates on a straight line with constant speed in a considerable long time interval. In fact, the proposed algorithms for image stabilization and EPI analysis do not require such a strict 1D constant-speed translation in the entire time period [0, T]. Motion with a curved path and varying speed is also allowed. First, let us consider the situation when the dominant motion is a translation but with varying speed $V(t)$. The condition under which the image stabilization and EPI analysis work is (1) the speed is piecewise linear (e.g. inside m=64 frames, i.e. 2 seconds), or (2) the speed function $V(t)$ is known. Based on these assumptions, we have two approaches for the nonlinear translation.

(1). Temporal re-sampling - If the speed of the vehicle, $V(t)$, can be accurately measured, then the x coordinate of a point in frame (time) t can be represented as

$$x(t) = f\frac{X + \int_0^t V(w)dw}{Z} = x + \int_0^t v(w)dw \qquad \text{(a2.1-1)}$$

where $v(t) = f\,V(t) / Z$ is the image velocity of point (X,Y,Z) in time t. The function of such an x-t image sequence can be expressed as

$$g(x,t) = g_0(x - f\frac{\int_0^t V(w)dw}{Z}) = g_0(x - \int_0^t v(w)dw \qquad \text{(a2.1-2)}$$

where $g_0(x)$ is the 1D image function in time t=0. Under a nonlinear 1D translation, spatio-temporal (ST) loci in an EPI are curves instead of straight lines. It will make ST orientation detection or locus tracking much more complex. Therefore, we perform a temporal re-sampling and interpolation in the time axis. The new time (i.e. frame index) t' will be

$$t' = \int_0^t \frac{V(w)}{V_0}dw = \int_0^t \frac{v(w)}{v_0}dw$$

where V_0 is a target speed, for example, V_0 =E[$V(t)$], and $v_0 = f V_0 / Z$ is the image velocity of the point (X,Y,Z) after image re-sampling. Hence the re-sampled x-t image becomes

$$g(x,t') = g_0(x - f\frac{V_0}{Z}t') = g(x - v_0 t') \qquad \text{(a2.1-3)}$$

Eq. (a2.1-3) has the same form as Eq. (26), so after temporal re-sampling, we have an equivalent image sequence under 1D translation of constant speed V_0. The temporal re-sampling process is independent to the structure of the 3D scene.

(2). Piecewise linear motion model - If the motion can be approximated as a piecewise linear motion inside the window of size m (e.g. 64) that is used to detect the orientation of the locus at the center of the window, temporal re-sampling may not be necessary. A smooth translation, even with varying speed, does not change the occluding relations between objects with different depths. However, the depth estimates extracted from such an x-t image are a function of the speed $V(t)$. So if we know this speed function, we can normalize the depth estimates by using the speed information. Suppose that the image velocity of point (x,y,t) is v, then the 3D coordinates of that point should be

$$Z = F\frac{V(t)}{v}, Y = y\frac{V(t)}{v}, X = x\frac{V(t)}{v} - \int_0^t V(\tau)d\tau \qquad \text{(a2.1-4)}$$

2. Piecewise circular motion model

As a generalization of the "translation + fluctuation" motion, we have proposed a motion model that consists of piecewise "circular motions + fluctuations". A piece-wise circular motion has one of the following four forms(Fig. a2.1.1):

(1) *Pure rotation*: The camera rotates around its nodal point (C_5 in Fig. a2.1.1). The relation between two camera poses is a pure rotation (1D panning).

(2) *Inner-view rotation*: The camera moves along a circular path while its optical axis is always pointing to the center of the circle where the circular path is on(C_1 and C_3 in Fig. a2.1.1). It is the situation when the camera moves around an object. The relation between two poses of the camera is 2D rotation plus 2D translation.

(3) *Outer-view rotation*. The camera moves along a circular path while the opposite direction of its optical axis is pointing to the center of that circle (C_4 in Fig. a2.1.1). It is the situation when the camera does an off-nodal-point rotation around itself.

(4) *Translational motion.* The camera moves along a straight path while its optical axis is pointing to the orthogonal direction of the motion. It can be viewed as a special rotation of the camera on a circle whose center is at infinity (C_2 in Fig. a2.1.1).

A combination of the above four types of rotation can model a rather general motion of the camera on a curved path (Fig. a2.1.1).

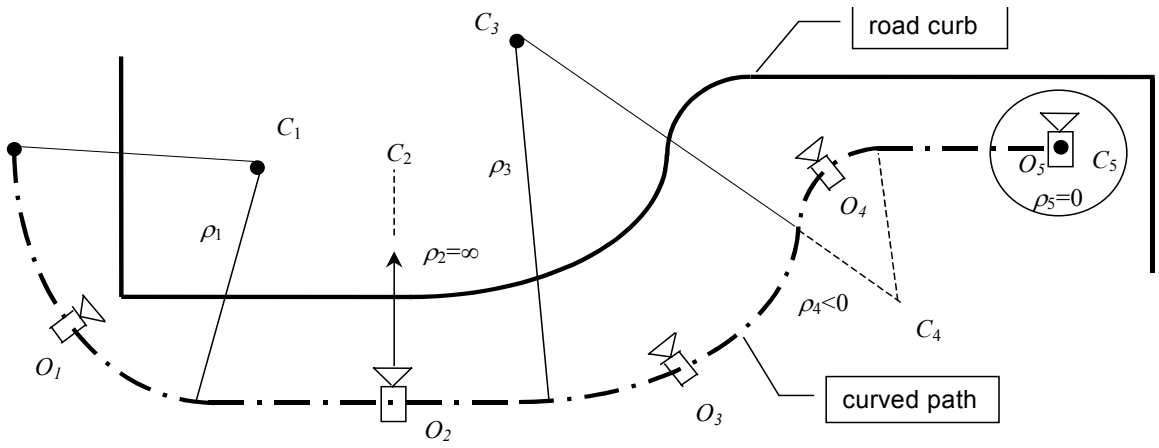

Fig. a2.1.1

(1) Mathematical model of the piecewise circular motion

We will first model the smooth motion part; the fluctuation of the camera will be added in afterwards. When the vehicle is moving on a planar road surface, the path of the camera can be represented as piecewise circular segment. Without loss of generality, we assume that the optical axis of the camera is always perpendicular to the motion direction and is parallel to the road surface. During the movement of the camera on a certain piece of the circular curve, the converging point C is defined as the origin of the world coordinate system $X_w Y_w Z_w$, and the three axes X_W, Y_W, Z_W are parallel to the three axes X_0, Y_0, Z_0 of the camera in the starting point of the circular arc (when the time t = 0). Thus the camera's motion is always a rotation around the Y_w axis (Fig. a2.1.2). Assume the radius of the rotating circle is ρ, then a point $P(X_0, Y_0, Z_0)$ in the camera coordinate system in time t can be represented in the world coordinate system as

$$X_w = X_0, Y_w = Y_0, Z_w = Z_0 - \rho$$

Denote the coordinates of that point in time t as X_t, Y_t, Z_t, and the rotating angle is ω (Fig. a2.1.2), the following relation holds:

$$\begin{pmatrix} X_t \\ Y_t \\ Z_t \end{pmatrix} = \begin{pmatrix} X_w \cos\omega - Z_w \sin\omega \\ Y_w \\ X_w \sin\omega + Z_w \cos\omega + \rho \end{pmatrix} = \begin{pmatrix} X_0 \cos\omega - (Z_0 - \rho)\sin\omega \\ Y_0 \\ X_0 \sin\omega + (Z_0 - \rho)\cos\omega + \rho \end{pmatrix} \qquad \text{(a2.1-5)}$$

where the motion is pure rotation when $\rho=0$, inner-view rotation when $\rho>0$, outer-view rotation when $\rho<0$ and pure translation when $\rho \to \pm\infty$.

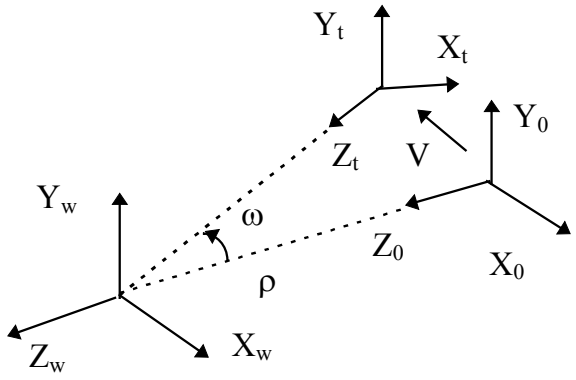

Fig. a2.1.2

1) When $|\rho| \to \infty$, we have $\omega \to 0$ therefore $\cos\omega \to 1$, $\sin\omega \to \omega$ and $\rho\omega \to V$ (a constant value), hence

$$\begin{pmatrix} X_t \\ Y_t \\ Z_t \end{pmatrix} = \begin{pmatrix} X_0 - (Z_0 - \rho)\omega \\ Y_0 \\ X_0\omega + (Z_0 - \rho) + \rho \end{pmatrix} \equiv \begin{pmatrix} X_0 + V \\ Y_0 \\ Z_0 \end{pmatrix} \qquad \text{(a2.1-6)}$$

It is the motion model of a pure translation.

2) When $\rho = 0$, we have

$$\begin{pmatrix} X_t \\ Y_t \\ Z_t \end{pmatrix} = \begin{pmatrix} X_0 \cos\omega - Z_w \sin\omega \\ Y_0 \\ X_0 \sin\omega + Z_0 \cos\omega \end{pmatrix} \qquad \text{(a2.1-7)}$$

It is the motion model of a pure rotation. The relationship between the corresponding image coordinates is

$$(x_t, y_t) = \left(f \frac{x_0 \cos\omega - f \sin\omega}{x_0 \sin\omega + f \cos\omega}, f \frac{y_0}{x_0 \sin\omega + f \cos\omega} \right) \qquad \text{(a2.1-8)}$$

In this case, we cannot obtain any depth information. The depth information is also unreliable if $|\rho|$ is too small. However the rotation angle ω can be calculated by image registration of the two images in pure rotation.

3) When $|\rho| >> Z_0$, i.e. $|\rho|$ is large enough, $V = \rho\omega$ approaches a constant value, and $\sin\omega \to \omega, \cos\omega \to 1$, we will have

$$\begin{pmatrix} X_t \\ Y_t \\ Z_t \end{pmatrix} = \begin{pmatrix} X_0 - (Z_0 - \rho)\omega \\ Y_0 \\ X_0\omega + Z_0 \end{pmatrix} = \begin{pmatrix} X_0 - \frac{Z_0}{\rho}V + V \\ Y_0 \\ X_0\frac{V}{\rho} + Z_0 \end{pmatrix} \approx \begin{pmatrix} X_0 + V \\ Y_0 \\ Z_0 \end{pmatrix} \qquad \text{(a2.1-9)}$$

In a short time interval (e.g. m=64 frames), we will have $\frac{Z_0}{\rho}V \to 0$ (if $|\rho| >> Z_0$), therefore we can model the camera motion as a translation along the tangent of the circular arc. Hence we can still use the assumption of the straight loci inside an $m \times m$ window of an EPI to find the image velocity *v* of the point at the center of the window. Then the 3D coordinates of that point in time t can be calculated as $(X_t, Y_t, Z_t) = \left(x\frac{V}{v}, y\frac{V}{v}, F\frac{V}{v}\right)$. The 3D coordinates (X_0, Y_0, Z_0) in the reference frame (t = 0) can be computed by using the transformation between the current frame *t* and the reference frame.

4) In the general case, radius $|\rho|$ and depth Z_0 are at the same scale. If we know *f*, ρ and ω, then

$$(x_t, y_t) = \left(f\frac{x_0\cos\omega - f\sin\omega}{x_0\sin\omega + f\cos\omega + f\frac{\rho}{Z_0}}, f\frac{y_0}{x_0\sin\omega + f\cos\omega + f\frac{\rho}{Z_0}} \right) \qquad \text{(a2.1-10)}$$

By finding the corresponding point pair in the two frames $(x_0, y_0), (x_t, y_t)$, the rotating angle ω and the depth Z_0 can be calculated by using Eq. (a.2.1-10). More accurate estimation can be achieved by tracking a feature point in multiple frames. We may not be able to extract epipolar plane images in this general case since the epipolar line of a point is a curve. In this case, feature point matching is needed (Ishiguro90, Murray95).

(2). Motion filtering under the generalized motion model

We model the dominant motion as a circular motion along a circular path with radius ρ and constant angular speed ω. The linear speed is $V = \rho\omega$. The fluctuation is modeled as a 3 DOF

rotation $R:(\Omega_x,\Omega_y,\Omega_z)$. The focal length of the camera in time *t* is *f,* the zooming factor is *s* between the current frame *t* and the reference frame *t-1*, i.e., $f=sf'$. Then the relationship between the coordinates in two successive images is

$$\boldsymbol{R}\begin{pmatrix}X\\Y\\Z\end{pmatrix}=\begin{pmatrix}X'-(Z'-\rho)\omega\\Y'\\X'\omega+Z\end{pmatrix} \tag{a2.1-11}$$

The relationship between the image coordinates (x',y') and (x,y) is

$$\begin{cases}x-\Omega_z y+f\Omega_y=f\dfrac{x'-f'\omega+f'\dfrac{\rho}{Z}\omega}{x'\omega+f'}\\[2ex] y+\Omega_z x-f\Omega_x=f\dfrac{y'}{x'\omega+f'}\end{cases} \tag{a2.1-12}$$

After a Taylor expansion of the above equation and then omitting the second-order terms (i.e. $xy',x'x,\Omega_x\omega$, etc., we have

$$\begin{cases}x-\Omega_z y+f\Omega_y=sx'-f\omega+f\dfrac{\rho}{Z}\omega\\ y+\Omega_z x-f\Omega_x=sy'\end{cases} \tag{a2.1-13}$$

Given N corresponding pairs (x_i,y_i) and $(x_i',y_i'),i=1,\cdots N$, we have

$$\begin{cases}sx_i'=x_i-\Omega_z y_i+T_{xi}\\ sy_i'=y_i+\Omega_z x_i+T_y\end{cases} \tag{a2.1-14}$$

where

$$T_{xi}=T_x+f\omega\left(1-\frac{\rho}{Z_i}\right),T_x=f\Omega_y,\ T_y=-f\Omega_x \tag{a2.1-15}$$

Using least mean square method, The *N*+3 unknowns($T_{xi},i=1,\cdots,N;\Omega_z,s$ and T_y) can be calculated given *N*≥3. Using the motion filtering techniques in Section 2.2, we can decompose the fluctuation and rectify the image sequence.

Appendix 2.2. Motion estimation

The inter-frame image displacements are estimated by using a pyramid-based matching algorithm. The hierarchical algorithm consists of four steps: pyramid construction, hierarchical block matching, match evaluation and robust estimation of motion parameters.

Step 1: Generate the pyramids for the current and the reference (preceding) images. For computational efficiency, the final image displacements are only given for non-overlapping image blocks of a given size, say 16×16, in the finest layer (i.e. original image) of the reference frame. The matching process is carried out from coarse to fine resolution layers, starting from a layer with certain image size, e.g., 2 times as large as the matching block size. The list of the blocks is represented by their center coordinates $\{(u_i,v_i)$, i=0, ...,B-1$\}$ in the reference frame.

Step 2: Determine the image displacements. For each block in a layer of the reference frame, the absolute difference operation (a simple version of correlation) is carried out in an *adaptive* search window over the current frame pixel by pixel. Matches with largest correlation values are determined and the one with smallest displacement is selected as the best match. Notice that there may be several best matches due to similar patterns within the search window. The search window is "adaptive" in that the initial size of the search window is about half the image size in the first layer, but it is reduced in the finer layers. The motion vectors for these blocks are presented by $\{(\Delta u_i, \Delta v_i)$, i=0, ...,B-1$\}$.

Step 3: Evaluate each match by combining a texture measure with the correlation measurement. This step is important because the confidence values will serve as weights in the parameter estimation. The evaluation of the matching itself is calculated from the normalized absolute difference of each block as

$$d_i = 1.0 - \frac{1}{255 N_w} \sum_{(u,v)\in W(u_i,v_i)} | I(u,v) - I'(u+\Delta u_i, v+\Delta v_i) | \qquad \text{(a2.2-1)}$$

where $W(u_i,v_i)$ is the block centered at (u_i,v_i), N_w is the pixel number in the block, *I(.)* and *I'(.)* are the intensity values (0-255) in the reference and current frames, respectively. The texture is measured as the normalized average magnitude of the gradient image of the reference frame inside a given block i

$$g_i = \frac{1}{g_{\max} N_w} \sum_{(u,v)\in W(u_i,v_i)} |(\frac{\partial I(u,v)}{\partial u}, \frac{\partial I(u,v)}{\partial v})| \qquad \text{(a2.2-2)}$$

where g_{max} is the maximum value of average magnitudes of all the blocks. The initial weight for the ith match is computed as

$$w_i^{(0)} = \frac{1 - e^{-\kappa d_i g_i}}{1 - e^{-\kappa}} \qquad \text{(a2.2-3)}$$

where κ = 8.0 in our experiments. Note that the weight is maximum (i.e. $w_i^{(0)} = 1$) when the match has the smallest difference and the best texture among all the blocks (i.e. d_i=g_i=1), and is minimun (i.e. $w_i^{(0)}$=0) if d_i=0 or g_i=0.

Step 4: Estimate inter-frame motion parameters. We use a weighted least mean square (WLMS) method to iteratively estimate the inter-frame parameters $\boldsymbol{W} = (a_1,...a_N,b',c,d',e,g,h)$ in equation (4) and Eq. (9). The objective function is

$$J = \min \sum_i w_i^{(k)} (r_i^{(k)})^2, \ (r_i^{(k)})^2 = | \mathbf{u}_i - \mathbf{W}^{(k)}(\mathbf{u}'_i) |^2 \qquad \text{(a2.2-4)}$$

where $\mathbf{u}_i = (u_i, v_i)^t$, $\mathbf{u'}_i = (u_i + \Delta u_i, v_i + \Delta v_i)^t$, i = 0,..., B-1, and the weight updating function is

$$w_i^{(k+1)} = \frac{w_i^{(0)}}{1 + (r_i^{(k)} / \rho)^2} \qquad \text{(a2.2-5)}$$

where the scale factor ρ is estimated as (Rousseeuw and Leroy 1987; Sawhney and Ayer 1996)

$$\rho = \underset{i}{median}(| r_i^{(k)} |) * 1.4826 \qquad \text{(a2.2-6)}$$

assuming that the residuals can be modeled as a noisy Gaussian distribution (residuals for the non-dominant components are the outliers). It has been pointed out in Sawhney and Ayer (1996) that a median-based estimate has excellent resistance to outliers. The iterative algorithm is given as follows.

---

(1): Initialize : k=0, $\mathbf{W}^{(-1)} = \mathbf{0}$.

(2): Find $\mathbf{W}^{k)}$ using WLMS method.

(3). Compute the distance $\Delta\mathbf{W}^{(k)} = |\mathbf{W}^{(k)} - \mathbf{W}^{(k-1)}|$, and estimate the scale factor ρ based on the current residuals.

(4). If $|\frac{\Delta\mathbf{W}^{(k)}}{\mathbf{W}^{(k)}}| < \varepsilon$ (e.g. $1.0e^{-3}$), or $\rho < \varepsilon$, or iterating count k > MaxK (e.g., 20), then stop; else update the weights $w_i^{(k)}$, assign k = k+1, and then go to step (2).

Appendix 2. 3. Energy model of the occluding boundary

Define the following functions for Eq. (30)

$$g_1(x,t) = u(x - v_2 t) f_1(x - v_1 t) \tag{a2.3-1}$$

$$g_2(x,t) = (1 - u(x - v_2 t)) f_2(x - v_2 t) \tag{a2.3-2}$$

It is easy to prove that

$$G_2(\xi,\omega) = (F_2(\xi) - F_2(\xi) * U(\xi))\delta(v_2\xi + \omega) \tag{a2.3-3}$$

Now we will prove

$$G_1(\zeta,\omega) = \frac{1}{v_1 - v_2} F_1(\frac{v_2\zeta + \omega}{v_2 - v_1}) U(\frac{v_1\zeta + \omega}{v_1 - v_2}) \tag{a2.3-4}$$

In Eq. (a2.3-1), let $x' = x - v_1 t$, then $x = x' + v_1 t$, $x - v_2 t = x' + (v_1 - v_2)t$, therefore

$$G_1(\xi,\omega) = \int_{x'}\int_t f_1(x') u(x' + (v_1 - v_2)t) e^{-j2\pi(\xi x' + (\xi v_1 + \omega))t} dx' dt$$
$$= \int_{x'} f_1(x') e^{-j2\pi\xi x'} \left\{ \int_t u(x' + (v_1 - v_2)t) e^{-j2\pi(\xi v_1 + \omega)t} dt \right\} dx'$$

Let $t' = x' + (v_1 - v_2)t$, then $t = \frac{t' - x'}{v_1 - v_2}$, hence

$$G_1(\xi,w) = \int_{x'} f_1(x') e^{-j2\pi\xi x'} \left\{ \int_{t'} u(t') e^{-j2\pi\frac{\xi v_1 + \omega}{v_1 - v_2} t'} e^{j2\pi\frac{\xi v_1 + \omega}{v_1 - v_2} x'} \frac{dt'}{v_1 - v_2} \right\} dx'$$
$$= \frac{1}{v_1 - v_2} \int_{x'} f_1(x') e^{j2\pi\frac{\xi v_2 + \omega}{v_2 - v_1} x'} dx' \int_{t'} u(t') e^{-j2\pi\frac{\xi v_1 + \omega}{v_1 - v_2} t'} dt' \tag{a2.3-3}$$
$$\frac{1}{v_1 - v_2} F_1(\frac{\xi v_2 + \omega}{v_2 - v_1}) U(\frac{\xi v_1 + \omega}{v_1 - v_2})$$

Add Eq. (a2.3-3) and Eq. (a2.3-5), We have Eq. (31).

Appendix 2.4. GFOD: a Fast Algorithm

The implementation of GFOD (Gaussian Fourier Orientation Detector) is based on a 1D Fast Fourier Transform (FFT) algorithm, and it is only performed in an m×m moving window along a 1D scanline (e.g. $x_0 = 0$). Generally speaking, a moving window technique can be designed to make use of the overlapping along the time axis in order to save computation time. However, the multiplication of a Gaussian window to a spatio-temporal epipolar plane image will increase the complexity of the re-use of the previous result. We have designed an fast GFOD algorithm that can use the temporal coherence as well as allow adaptive moving steps of the Gaussian window along the time axis. A 2D Gaussian function can be separated as the product of two 1D Gaussian function as

$$w(x,t) = w_1(x)w_1(t) \qquad \text{(a2.4-1)}$$

The m×m Fouier transform $G(\xi,\omega)$ of $g(x,t) = f(x,t)w(x,t) = f(x,t)w_1(x)w_2(t)$ is calculated in two steps: first, for each column in the x direction, performing a 1D FFT obtain an intermediate result $G(\xi,t)$; second, applying 1D FFTs along the t direction to obtain the final Fourier transform $G(\xi,\omega)$. When the Gaussian window is centered at (x_0, t_1), the origin of an m×m sub-image, the Guassian Fourier transform for column t in this sub-image is

$$G_{t_1}(\xi,t) = F\left[g(x,t+t_1)w_1(x)w_1(t)\right] = F\left[g(x,t+t_1)w_1(x)\right]w_1(t) \qquad \text{(a2.4-2)}$$

Fig. a2.4-1. Moving window method

Assume that the next location that a Gaussian window will be applied is $t_2 = t_1 + \Delta T$, then the Gaussian-Fourie transform with origin (x_0, t_2) should be

$$\begin{aligned} G_{t_2}(\xi,t) &= F\left[g(x,t+t_2)w_1(x)w_1(t)\right] \\ &= F\left[g(x,t+t_1+\Delta T)w_1(t+\Delta T)\right]w_1(t+\Delta T)\frac{w_1(t)}{w_1(t+\Delta T)} \qquad \text{(a2.4-3)} \\ &= G_{t_1}(\xi,t+\Delta T)\frac{w_1(t)}{w_1(t+\Delta T)} \end{aligned}$$

which means that the 1D Gaussian-Fourier transform of column t in the window of time t_2 can use the 1D Gaussian-Fourier transform of column t+ΔT in the window of time t_1, if we have $t + \Delta T \le \frac{m}{2}$, where $t \in \left[-\frac{m}{2}, \frac{m}{2}\right]$, and m is the size of the window. For such a column, the computation complexity is reduced to $O(m)$, comparing to $O(m \log_2 m)$ if 1D FFT is directly applied. When ΔT is smaller (i.e. the ST texture is richer hence the orientation estimation is denser and better) the speedup in computation is more obvious. In the extreme case when ΔT=1, For an M(row)×N(column) xt image, the multiplication and the addition of a 2D FFT (using 1D FFT directly) is $O\left(2Nm^2 \log_2 m\right)$. Using the proposed fast algorithm, the computation complexity reduces to $O\left(Nm^2 (\log_2 m + 1)\right)$, which means almost 2 times speedup.