Korea Advanced Institute of Science and Technology

School of Electrical Engineering

EE636 Digital Video Processing Fall 2018

Programming Assignment 1

Student's Name: Dinh Vu

Student's ID: 20184187

1. Implementation of Lucas-Kanade algorithm to estimate an optical flow

a. Estimate Optical flows

Assume that $s_c(x_1, x_2, t)$ present the continuous space-time intensity distribution [1]. The Optical Flow Equation (OFE) is:

$$\frac{ds_c(x_1, x_2, t)}{dt} = 0$$
$$\Rightarrow \frac{\partial s_c(\mathbf{x}, t)}{\partial x_1} v_1(\mathbf{x}, t) + \frac{\partial s_c(\mathbf{x}, t)}{\partial x_2} v_2(\mathbf{x}, t) + \frac{\partial s_c(\mathbf{x}, t)}{\partial t} = 0 \text{ (Using the chain rule)}$$

Where:

$$v_1(\mathbf{x}, t) = \frac{dx_1}{dt} \text{ and } v_1(\mathbf{x}, t) = \frac{dx_2}{dt}$$

However, practically, an image is not processed in the continuous domain. Suggested by Lucas and Kanade, an image is divided into square blocks have the same size, in a block $\boldsymbol{B}$:

$$\mathbf{v}(\mathbf{x}, t) = \begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix} \text{ for } \mathbf{x} \in \boldsymbol{B}$$

The error in the optical flow equation over the block $\boldsymbol{B}$ become:

$$E = \sum_{\mathbf{x} \in \boldsymbol{B}} \left(\frac{\partial s_c(\mathbf{x}, t)}{\partial x_1} v_1(t) + \frac{\partial s_c(\mathbf{x}, t)}{\partial x_2} v_2(t) + \frac{\partial s_c(\mathbf{x}, t)}{\partial t} \right)^2$$

Computing the partials of the error E with respect to (w.r.t) $v_1(t)$ and $v_2(t)$, then put them equal to 0, we have:

$$\sum_{\mathbf{x}\in\boldsymbol{B}}\left(\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\hat{v}_1(t)+\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\hat{v}_2(t)+\frac{\partial s_c(\mathbf{x},t)}{\partial t}\right)-\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}=0$$

$$\sum_{\mathbf{x}\in\boldsymbol{B}}\left(\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\hat{v}_1(t)+\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\hat{v}_2(t)+\frac{\partial s_c(\mathbf{x},t)}{\partial t}\right)-\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}=0$$

Solving the above equations simultaneously, we have:

$$\begin{bmatrix}\hat{v}_1(t)\\ \hat{v}_2(t)\end{bmatrix}=\begin{bmatrix}\sum_{\mathbf{x}\in\boldsymbol{B}}\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\frac{\partial s_c(\mathbf{x},t)}{\partial x_1} & \sum_{\mathbf{x}\in\boldsymbol{B}}\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\\ \sum_{\mathbf{x}\in\boldsymbol{B}}\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\frac{\partial s_c(\mathbf{x},t)}{\partial x_2} & \sum_{\mathbf{x}\in\boldsymbol{B}}\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\end{bmatrix}^{-1}\begin{bmatrix}-\sum_{\mathbf{x}\in\boldsymbol{B}}\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\frac{\partial s_c(\mathbf{x},t)}{\partial t}\\ -\sum_{\mathbf{x}\in\boldsymbol{B}}\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}\frac{\partial s_c(\mathbf{x},t)}{\partial t}\end{bmatrix}$$

Approximately, to calculate $\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}$ and $\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}$, the `imfilter()` function is used with kernel $[-1 \quad 1]$ and $\frac{\partial s_c(\mathbf{x},t)}{\partial t}$ equals to the difference intensity between two adjacent frames.

Each element of matrix such as $\sum_{\mathbf{x}\in\boldsymbol{B}}\frac{\partial s_c(\mathbf{x},t)}{\partial x_1}\frac{\partial s_c(\mathbf{x},t)}{\partial x_2}$ is determined by sum of element-wise multiplying.

To display the optical flows of each block, the coordinates of each block are calculated and two functions `imshow()` and `quiver()` are used to present the 10th gray frame and the blue motion vectors, respectively.

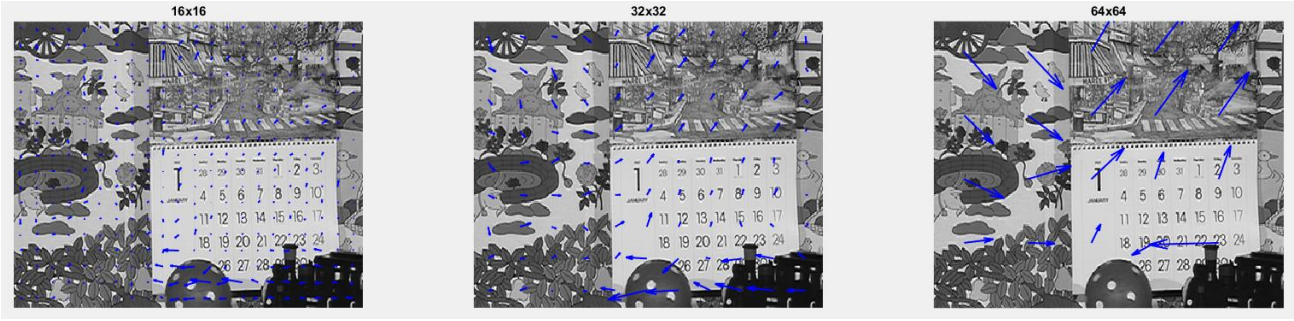

Figure 1.1. The optical flows estimated by Lucas-Kanade algorithm with block size of 16×16, 32×32 and 64×64

b. Calculate PSNR

The motion vectors between two adjacent frames are estimated following part 1a. Each pixel in the reconstructed frame is corresponding with a motion vector by calculating the coordinates of the block. Because the motion vector is real number, we only can locate 4 neighboring pixels based on the motion vector. Then, the motion compensation is determined by bi-linear interpolation from the intensity of four pixels.

After, getting all pixels in the reconstructed image, the PSNR between the reference and reconstructed images is calculated by `psnr()` function in Matlab. Figure 1.2 shows that when the block size increases, the PSNR decreases.

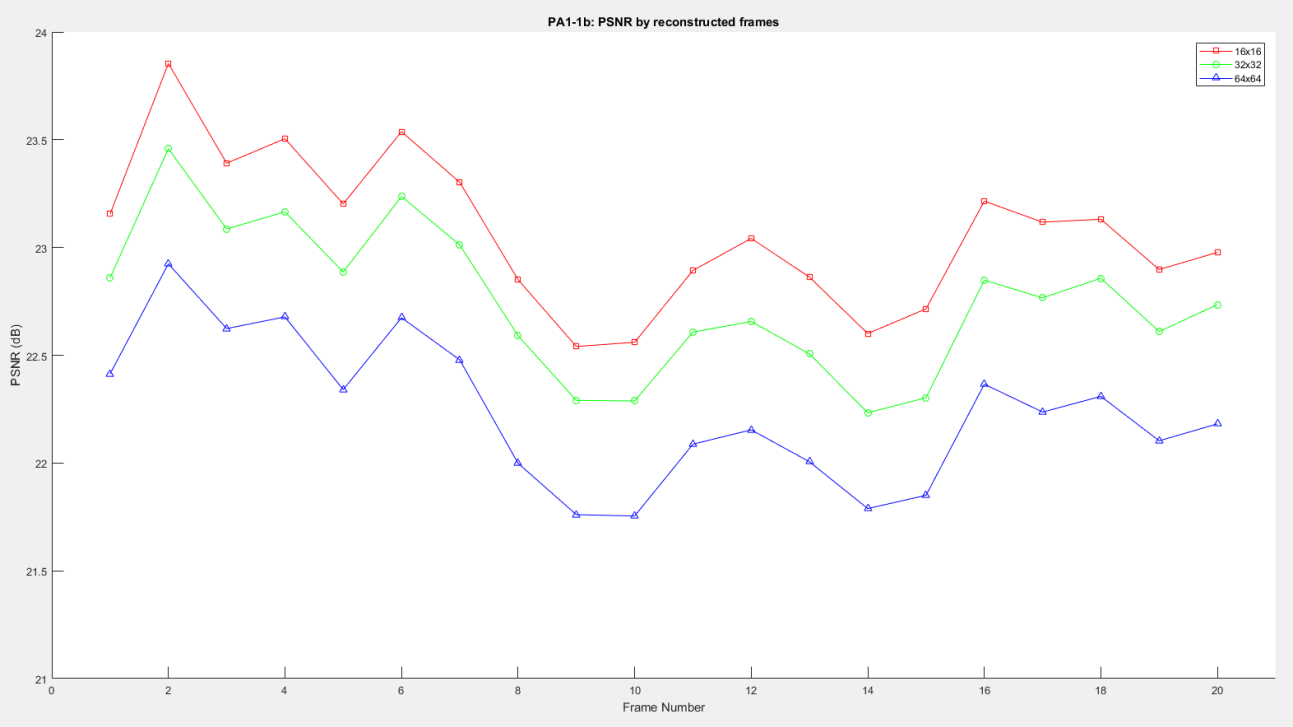

Figure 1.2. The PSNR of 20 first reconstructed frames using Lucas-Kanada algorithm with block size of 16×16, 32×32 and 64×64

Table 1.1. The min, max and average PSNR of 20 first frames using Lucas-Kanada algorithm with block size of 16×16, 32×32 and 64×64

Block Size	**16×16**	**32×32**	**64×64**
Min (dB)	22.5406	22.2331	21.7541
Max (dB)	23.8537	23.4583	22.9243
Average (dB)	23.0675	22.7498	22.2365

c. Difference between reference and motion compensated frames

After reconstruction with motion compensation, the difference between the reconstructed image and the 10^{th} image with the difference between the 11^{th} and 10^{th} image are computed by L1 loss using function `abs()`. Figure 1.3 shows that they are very similar with various block size.

Figure 1.3. The difference between the reconstructed and 11th with 10th frames with block size of 16×16, 32×32 and 64×64

d. Conclusion and aperture problem

The results of block size 32×32 and 64×64 are already shown in Figure 1.1, 1.2 and 1.3. As a result, it can be seen that with the smaller block size, the motion vectors can be estimated more accurately. If the block size is too large, not only the number of motion vectors is reduced, but also the result of combining multiple motions represents one block. Hence, the motion vectors with the block size of 16×16 and 32×32 are similar. As shown in Figure 1.2 and 1.3, the higher PSNR for the reconstructed frames along with the smaller block size and it can reflect the detailed motion. However, if the block size is too small, there is a risk that the gray-level patterns will be mapped to other blocks which are not related to the motion. Therefore, it is important to select an appropriate block size.

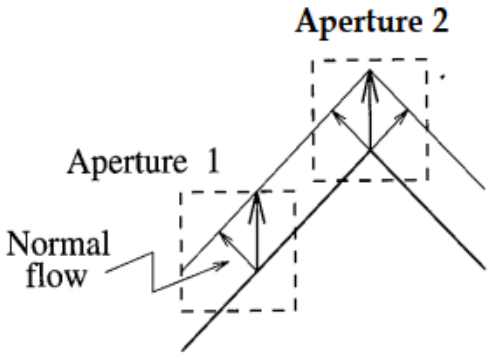

Figure 1.4. The aperture problem

The aperture problem occurs because we can only observe displacements perpendicular to the direction of the intensity gradient (edges). When the block size is too small, it is difficult to predict the exact motion because it is difficult to obtain two required vertical gradients to solve the OFE. It is consistent with the above mentioned, it is a risk that the gray-level patterns are not related to the motion but correspond to other blocks with similar gray-level patterns.

2. Implementation of Horn-Schunck algorithm to estimate an optical flow

a. Estimate optical flows

Horn and Schunck seek a motion that satisfies the OFE with the minimum pixel-to-pixel variation among the flow vectors [1]. Let

$$\mathcal{E}_{of}\big(\mathbf{v}(\mathbf{x},t)\big) = \big(\nabla s_c(\mathbf{x},t), \mathbf{v}(\mathbf{x},t)\big) + \frac{\partial s_c(\mathbf{x},t)}{\partial t}$$

denote the error in the OFE. Observe that the OFE is satisfied when this error equals to 0. In practice with noise, the square of this error is minimized subject to the smoothness constrain

$$\mathcal{E}_s^2\big(\mathbf{v}(\mathbf{x},t)\big) = \|\nabla v_1(\mathbf{x},t)\|^2 + \|\nabla v_2(\mathbf{x},t)\|^2 = \left(\frac{\partial v_1}{\partial x_1}\right)^2 + \left(\frac{\partial v_1}{\partial x_2}\right)^2 + \left(\frac{\partial v_2}{\partial x_1}\right)^2 + \left(\frac{\partial v_2}{\partial x_2}\right)^2$$

The the Horn-Schunck minimizes a weight sum of error in the OFE and a measure of the pixel-to-pixel variation of the velocity field

$$\min_{\mathbf{v}(\mathbf{x},t)} \int \Big(\mathcal{E}_{of}^2\big(\mathbf{v}(\mathbf{x},t)\big) + \alpha^2 \mathcal{E}_s^2\big(\mathbf{v}(\mathbf{x},t)\big)\Big)\, d\mathbf{x}$$

The minimization of the above function requires solving two equations below:

$$\left(\frac{\partial s_c}{\partial x_1}\right)^2 \hat{v}_1(\mathbf{x},t) + \frac{\partial s_c}{\partial x_1}\frac{\partial s_c}{\partial x_2}\hat{v}_2(\mathbf{x},t) = \alpha^2 \nabla^2 \hat{v}_1(\mathbf{x},t) - \frac{\partial s_c}{\partial x_1}\frac{\partial s_c}{\partial t}$$

$$\frac{\partial s_c}{\partial x_1}\frac{\partial s_c}{\partial x_2}\hat{v}_1(\mathbf{x},t) + \left(\frac{\partial s_c}{\partial x_2}\right)^2 \hat{v}_2(\mathbf{x},t) = \alpha^2 \nabla^2 \hat{v}_2(\mathbf{x},t) - \frac{\partial s_c}{\partial x_2}\frac{\partial s_c}{\partial t}$$

Using the Laplacians of the velocity components:

$$\hat{v}_1^{(n+1)}(\mathbf{x},t) = \bar{v}_1^{(n)}(\mathbf{x},t) - \frac{\partial s_c}{\partial x_1}\frac{\frac{\partial s_c}{\partial x_1}\bar{v}_1^{(n)}(\mathbf{x},t) + \frac{\partial s_c}{\partial x_2}\bar{v}_2^{(n)}(\mathbf{x},t) + \frac{\partial s_c}{\partial t}}{\alpha^2 + \left(\frac{\partial s_c}{\partial x_1}\right)^2 + \left(\frac{\partial s_c}{\partial x_2}\right)^2}$$

$$\hat{v}_2^{(n+1)}(\mathbf{x},t) = \bar{v}_2^{(n)}(\mathbf{x},t) - \frac{\partial s_c}{\partial x_2}\frac{\frac{\partial s_c}{\partial x_1}\bar{v}_1^{(n)}(\mathbf{x},t) + \frac{\partial s_c}{\partial x_2}\bar{v}_2^{(n)}(\mathbf{x},t) + \frac{\partial s_c}{\partial t}}{\alpha^2 + \left(\frac{\partial s_c}{\partial x_1}\right)^2 + \left(\frac{\partial s_c}{\partial x_2}\right)^2}$$

To calculate the gradient, Horn and Schunck proposed averaging four finite difference:

$$\frac{\partial s_c}{\partial x_1} \approx \frac{1}{4}[s(n_1+1,n_2,k) - s(n_1,n_2,k) + s(n_1+1,n_2+1,k) - s(n_1,n_2+1,k) + s(n_1+1,n_2,k+1) - s(n_1,n_2,k+1) + s(n_1+1,n_2+1,k+1) - s(n_1,n_2+1,k+1)]$$

$$\frac{\partial s_c}{\partial x_2} \approx \frac{1}{4}[s(n_1,n_2+1,k) - s(n_1,n_2,k) + s(n_1+1,n_2+1,k) - s(n_1+1,n_2,k) + s(n_1,n_2+1,k+1) - s(n_1,n_2,k+1) + s(n_1+1,n_2+1,k+1) - s(n_1+1,n_2,k+1)]$$

$$\frac{\partial s_c}{\partial t} \approx \frac{1}{4}[s(n_1,n_2,k+1) - s(n_1,n_2,k) + s(n_1+1,n_2,k+1) - s(n_1+1,n_2,k) + s(n_1,n_2+1,k+1) - s(n_1,n_2+1,k) + s(n_1+1,n_2+1,k+1) - s(n_1+1,n_2+1,k)]$$

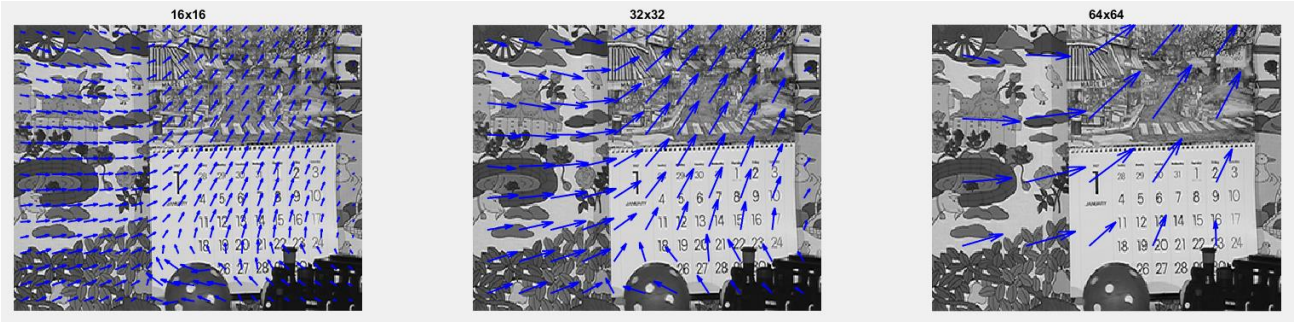

Figure 2.1. The optical flows estimated by Horn-Schunck algorithm with block size of 16×16, 32×32 and 64×64

b. Calculate PSNR

The experimental results about PSNR of the reconstructed images using Horn-Schunck method with difference block size are displayed in Figure 2.2. The graph presents that PSNR reduces while increasing the block size.

Otherwise, I also compare the performance between two methods with various block size in Figure 2.3. It is noticeable to see that Horn-Schunck method reconstructs the frames better than Lucas-Kanada because of smoothness constrain.

Table 2.1. The min, max and average PSNR of 20 first frames using Horn-Schunck algorithm with block size of 16×16, 32×32 and 64×64

Block Size	**16×16**	**32×32**	**64×64**
Min (dB)	25.1121	24.0157	22.6882
Max (dB)	25.7785	24.6739	23.3901
Average (dB)	25.4693	24.3715	23.0441

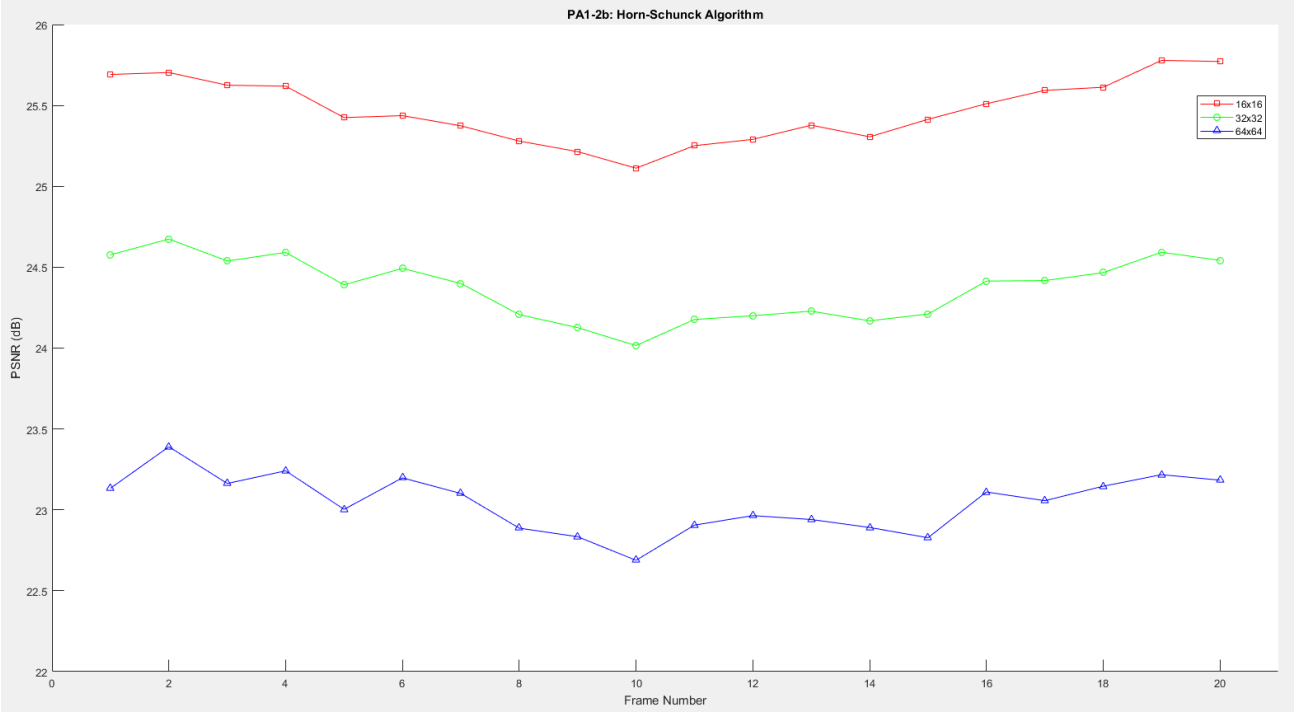

Figure 2.2. The PSNR of 20 first reconstructed frames using Horn-Schunck algorithm with block size of 16×16, 32×32 and 64×64

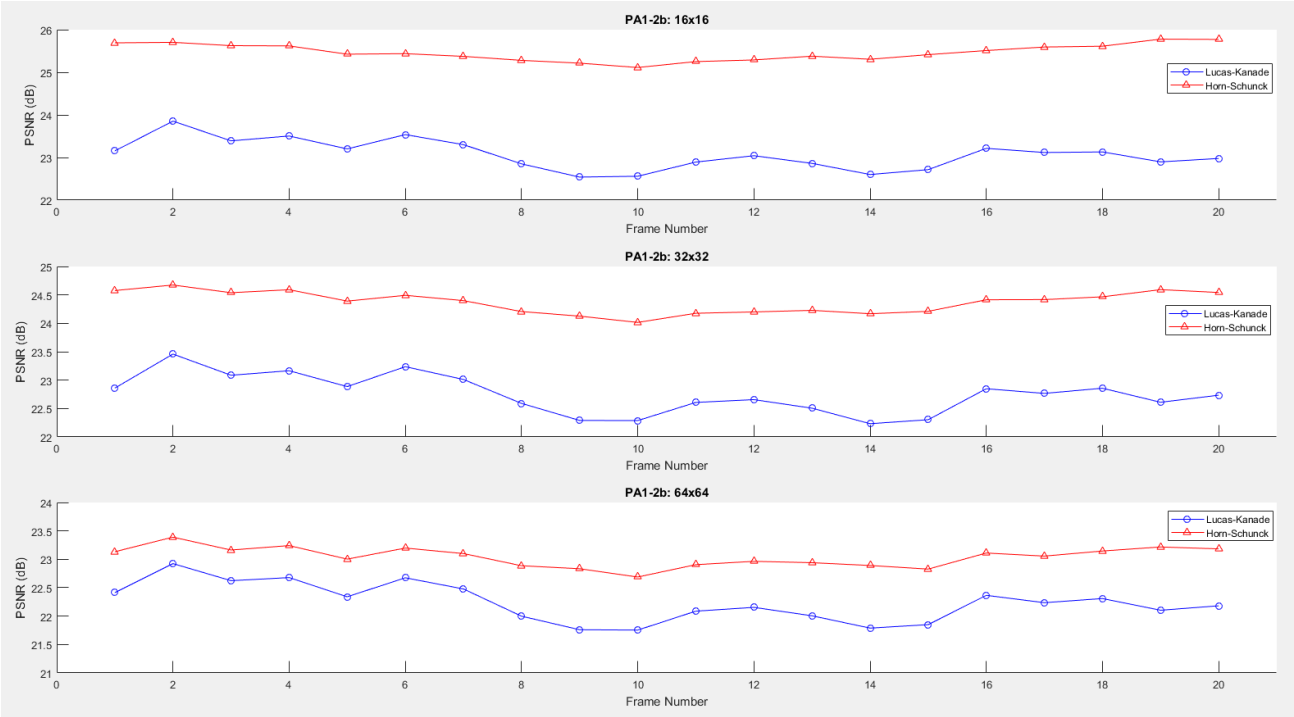

Figure 2.3. The PSNR comparison between Lucas-Kanada and Horn-Schunck algorithm with block size of 16×16, 32×32 and 64×64

c. Difference between reference and motion compensated frames

Figure 2.4. The difference between the reconstructed and 11th with 10th frames with block size of 16×16, 32×32 and 64×64

d. The occlusion problem

The occlusion problem is caused by the movement of the object as shown in the figure below, making covered and uncovered background in the image, which also happened in Calendar_CIF30.yuv. In particular, the ball and train move to the left, causing the occlusion problem when obtaining the motion vector. As a result, the difference image shows that both methods have difficulty in reconstructing a moving object, a ball and a train. To solve this problem, it is better to use a bi-directional motion estimation method that restores both past and future frame information.

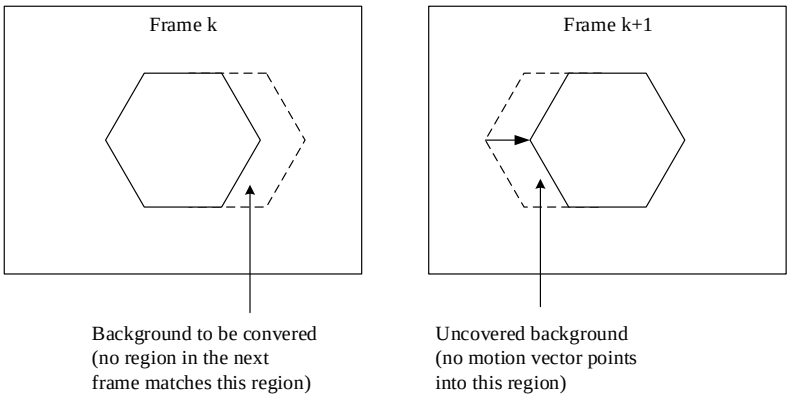

Figure 2.5. The occlusion problem

Reference

[1] A. Murat Tekalp, “Digital Video Processing”, Prentice Hall

[2] Bruce D. Lucas and Takeo Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision”, *Proceeding of the 7th International Joint Conference on Artificial Intelligence*, volume 2, pages 674-679, Canada 1981.

[3] Berthold K.P. Horn and Brian G. Schunck, “Determining Optical Flow”, *Artificial Intelligence*, volume 17, issue 1-3, pages 185-203, August 1981.

[4] Darun Kesrarat and Vorapoj Patanavijit, “Tutorial of Motion Estimation Based on Horn-Schunck Optical Flow Algorithm in Matlab”, *AU Journal of Technology*, volume 15, no. 1, July 2011.