Chapter 2

Motion Segmentation

2.1 Motion Estimation in Image Sequences

Motion estimation methods have been proposed in many digital video applications including surveillance systems, image stabilization, image registration, video coding such as MPEG-4, object tracking and detection, etc. For real time applications, multiple feature tracking algorithms are widely used [6, 7]. However, the precise estimation of motion is required for certain applications such as motion tracking under time-varying illuminations. Estimating motion patterns from gradient based methods give several numbers of advantages. Generally, these techniques are highly accurate and provide dense motion estimates without additional computation cost. Due to these advantages, gradient based structure tensor method is chosen to apply for the computation of dense motion fields for image sequences in our work.

2.1.1 Gradient Structure Tensor Method (GSTM)

3-D structure tensor method is the brightness change constraint which assumes that image brightness changes only due to motion. The tensor is constructed for every pixel within its neighborhood in which the local optical flow is assumed to be constant. Structure tensor is actually a conversion of optical flow estimation problem to an eigenvalue analysis problem.

When we assume that image intensity I at the location (x, y) and time t is constant over time, we have

$$I\,(x, y,t) = I\,(x + dx, y + dy,t + dt) \qquad (2.1)$$

where $I(x,y,t)$ indicates the image intensity at point (x,y) at time t, and $(\mathrm{d}x,dy)$ denotes the image displacement over time dt. By taking the first-order Taylor series expansion of the right hand side of Eq. (1), we obtain the well-known optical flow constraint equation (OFCE)

$$I_x \cdot u + I_y \cdot v + It = 0 \qquad (2.2)$$

where $(u,v) = (dx/\,dt\,,\,dy/\,dt)$ denotes motion vectors and $(I_x\,, I_y\,, I_t\,) = (\partial I\,(x, y,t)\,/\partial x\,, \partial I\,(x, y,t)\,/\partial y\,, \partial I\,(x, y,t)\,/\partial t)$ expresses the spatio-temporal image gradients [3]. Eq. (2) is apparently underdetermined because it contains two unknowns, u and v. The spatio-temporal gradient method (GM) solves Eq. (2) for the motion vectors (u,v) using the least-squares method where I_x and I_y are regarded as predictor variables and I_t as a response variable.

Meanwhile, the gradient structure tensor method (GSTM) solves Eq. (2) using the total squares approach [7]. In preparation, the gradient structure tensor (GST) is locally defined as

$$\mathrm{GST} = \begin{pmatrix} \sum_{BK} I_x^{\,2} & \sum_{BK} I_x I_y & \sum_{BK} I_x I_t \\ \sum_{BK} I_y I_x & \sum_{BK} I_y^{\,2} & \sum_{BK} I_y I_t \\ \sum_{BK} I_t I_x & \sum_{BK} I_t I_y & \sum_{BK} I_t^{\,2} \end{pmatrix} \tag{2.3}$$

where Σ denotes the integration within a small 3-D space such as a cube. The GST can be viewed as a correlation matrix of the spatio-temporal gradient vectors in principal component analysis [10].

Hence, in GSTM, motion estimation is formulated as a total squares problem or orthogonal regression in the 3-D space (I_x, I_y, I_t). Then the third eigenvector $\mathbf{V}_3 = [x_3 \quad y_3 \quad t_3]^T$, associated with the smallest eigenvalue λ_3 $(\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0)$ of the GST, indicates the motion trajectory in the spatio-temporal space (x, y, t). We may then obtain motion vectors (u, v) as

$$\begin{cases} u = x_3 / t_3 \\ v = y_3 / t_3 \end{cases} . \tag{2.4}$$

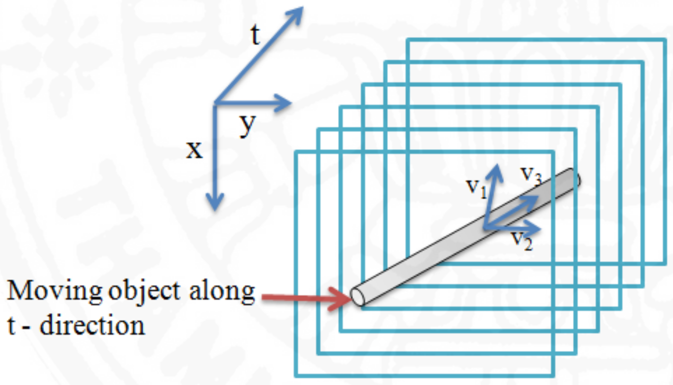


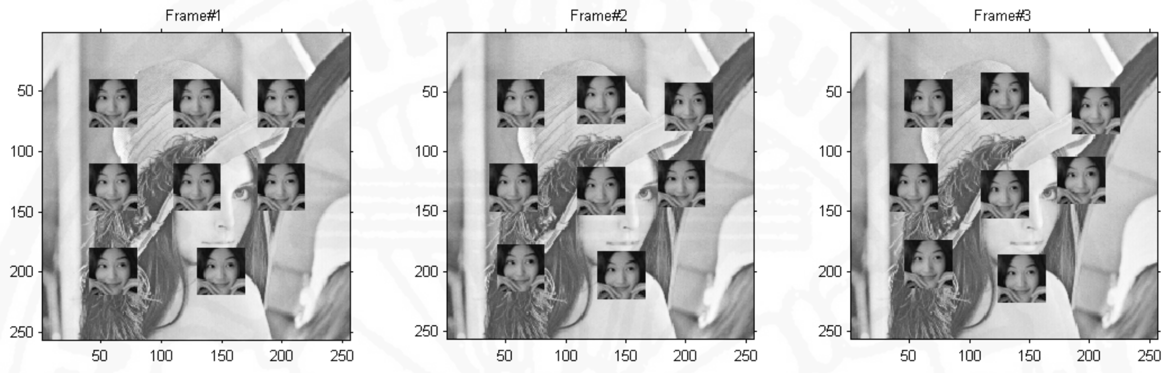
Figure 2.1 Motion trajectory on the sequence and corresponding eigenvector V$_3$

Figure 2.1 shows motion trajectory is represented by the third eigenvector V$_3$ in spatio-temporal space (x, y, t).

When the three eigenvalues have the relationship as $\lambda_1 \approx \lambda_2 >> \lambda_3 \geq 0$, the motion trajectory forms a linear structure in the spatio-temporal space, which indicates a high confidence of estimated motion. When the relationship $\lambda_1 >> \lambda_2 \approx \lambda_3 \geq 0$ holds, the motion trajectory is plane-like, which causes the aperture problem where we can estimate motion only along the gradient vector available. When the relationship $\lambda_1 \approx \lambda_2 \approx \lambda_3 > 0$ holds, the corresponding local region may have impulsive noise or motion discontinuities. Finally, if $\lambda_1 \approx \lambda_2 \approx \lambda_3 \approx 0$ holds, there is no gradient information and motion estimation is made impossible.

We test GSTM on both synthetic and real image sequences. For the synthetic sequence, a standard test image, Lena, of resolution 256×256 pixels with 8 bits gray levels is used as stationary background. Next, we superimpose 8 subimages of size 40 by 40 pixels on to the background. The eight subimages are translated by (−3,−3), (−3,0), (−3,3), (0,−3), (0,3), (3,−3), (3,0), and (3,3) pixels between two consecutive frames, respectively, generating 8 different motion vectors.

Figure 2.2 shows the three consecutive frames of synthetic image sequence. Motion vectors are calculated on the second frame.



(a) (b) (c)

Figure 2.2 Lena image with moving objects superimposed on it.

The gradients of an image are computed by using the first-derivative Sobel operators because of its reasonably good performance and ease of use. For the gradient tensor method, we used the 3-D version of the Sobel operators for computing I_x, I_y and I_t [10]. The three masks in Figure 2.3 form a cubic of 3×3×3 voxels and serves as a 3-D convolution mask. The orientation of 3-D mask must be selected properly to compute I_x, I_y and I_t.

1	2	1
0	0	0
-1	-2	-1

2	3	2
0	0	0
-2	-3	-2

1	2	1
0	0	0
-1	-2	-1

Figure 2.3 **3-D** Sobel operators that form a cube of 3×3×3 voxels.

GSTM is then applied on the synthetic image sequence. The size of the integration area is 8×8×3 voxels and we have 28×28 motion vectors in total as shown in Figure 2.4. Since false motion vectors may be occurred at the boundaries of moving objects and low-contrasted areas, we remove those erroneous vectors by applying the median filter. In the experiment here, we use the Median filter that covers 3×3 motion vectors. From the result, it can be seen that motions are estimated correctly according to the movements of each subimages in the image.

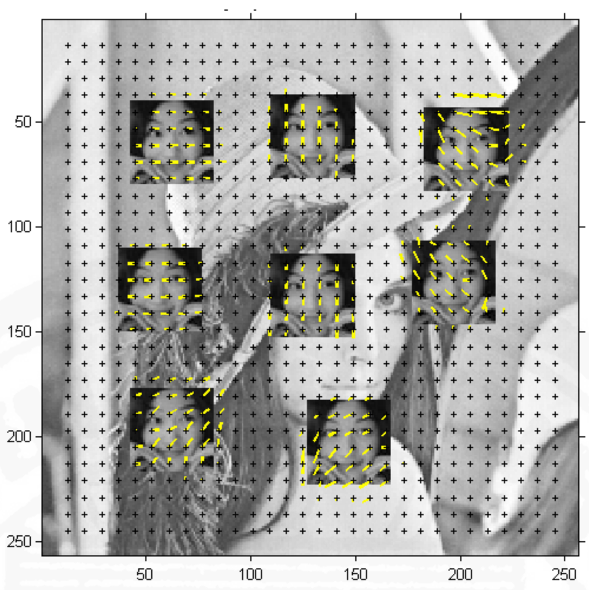


Figure 2.4 Motion vectors estimated by the gradient structure tensor method.

2.2 Motion Classification and Visualization

2.2.1 Self-Organizing Feature Map (SOM)

Self-Organizing Feature Map (SOM or SOFM) is one of the neural networks techniques and is a non-linear generalization of principle component analysis. The learning method of SOM discovers significant features within the input data without external supervision. It is used for extracting the most representative low-dimensional subspace from a high-dimensional pattern vector space.

The first part of a SOM is the data. The idea of the self-organizing feature map is to project n-dimensional data into something that be better understood visually. The second component to SOM is the weight vectors. Each weight vectors has two components which are data and its natural location. Weights are sometimes referred to as neurons since SOMs are actually neural networks.

SOM forms a topographic map or feature map (usually 1D or 2D) which represents the feature extracted from input data. Inside the map, there are features called neurons. Each neuron in the map is fully connected to all the source nodes in the input layer. Figure 2.5shows the example of Kohonen neural network with two inputs neurons and 4x4 output neurons.

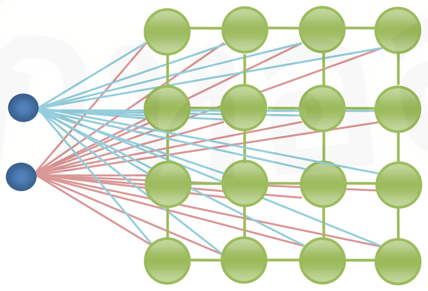


Figure 2.5 Structure of SOM neural network

Once the network has been properly initialized, there are three essential processes involved in the formation of the self-organizing map. They are competition, cooperation and synthetic adaption processes.

1. *Competition*: The neurons in the network compute their respective values of a discriminant function for every input pattern. This discriminant function provides the basis for competition among the neurons. The particular neuron with the largest value of discriminant function is declared winner of the competition.

2. *Cooperation*: The winning neuron determines the spatial location of a topological neighborhood of excited neurons, thereby providing the basis for cooperation among such neighboring neurons.

3. *Synthetic Adaption*: This last process enables the excited neurons to increase their individual values of the discriminant function in relation to the input pattern through suitable adjustments applied to their synthetic weights. The adjustments made are such that the response of the winning neuron to the subsequent application of a similar input pattern in enhanced.

The detailed procedures for each process are stated as follows.

1. *Competitive Process*

The most essential feature of SOM is to incorporate into the competitive learning rule some degree of sensitivity with respect to the neighborhood. It is the process in which output layer neurons compete among themselves to give the respective response for each input data. The neuron with largest response is the winner.

Let ***v*** be an input data and w be the synaptic weight vector.

$$\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m]^T \text{, where } m \text{ is the dimension of input data.} \quad (2.5)$$

$$\mathbf{w} = [\mathbf{w}_{j1}, \mathbf{w}_{j2}, ..., \mathbf{w}_{jm}]^T \quad , j = 1, 2, \ldots, l \quad (2.6)$$

where l is the total number of output neurons in the network and j is the index of them. The winning neuron (or winner) is determined by a minimum Euclidean distance criterion between an input motion vector and weights of the neurons in the networks.

$$ED = \|\mathbf{v} - \mathbf{w}_j\|, \qquad j = 1, 2, ..., l \quad (2.7)$$

The winning output neuron is the one with the shortest Euclidean distance.

2. *Cooperative Process*

The winning neuron locates the center of a topological neighborhood of cooperating neurons. Neighborhood function $\mathrm{h}_{j,i}(n)$ can be represented as Gaussian function in which $d_{j,i}$ denote the lateral distance between winning neuron i and excited neuron j.

$$\mathrm{h}_{j,i}(n) = \exp\left(-d_{j,i}^2 / 2\sigma^2(n)\right), \qquad n = 0, 1, 2, ... \quad (2.8)$$

The parameter σ is the effective width of the neighborhood which measures the degree to which excited neurons in the neighborhood of the winning neuron participate in the learning process. Figure 2.6 shows the Gaussian neighborhood function.

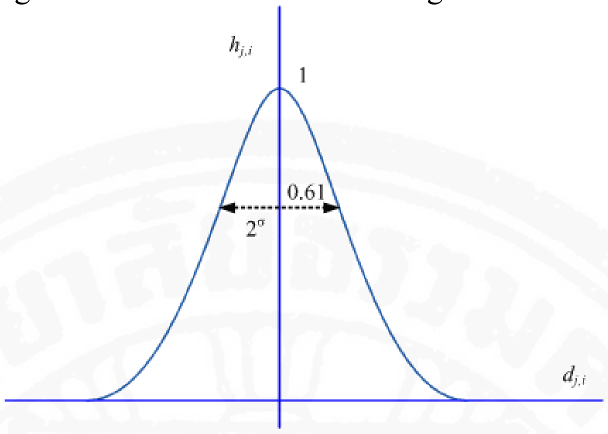


Figure 2.6 Gaussian neighborhood function.

Figure 2.7 below is the neighborhood which is centered around the winning neuron (yellow color) and the green arrow is the radius (width) of the neighborhood.

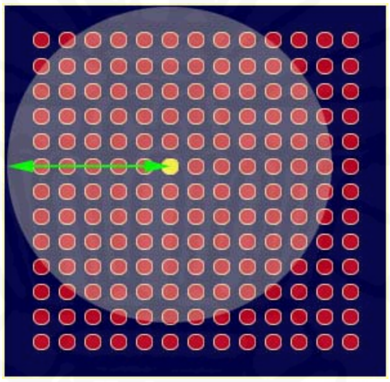


Figure 2.7 Neighborhood of SOM network with winning neuron at the center.

It is necessary that neighborhood $h_{j,i}$ be dependent on lateral distance $d_{j,i}$ between winning neuron i and excited neuron j in the output space rather than on some distance measure in the original input space. The distance $d_{j,i}$ is defined by

$$d_{j,i}^2 = \left\| \mathbf{r}_j - \mathbf{r}_i \right\|^2 \tag{2.9}$$

where the discrete vector $\mathbf{r}_j$ defines the position of the excited neuron j and $\mathbf{r}_i$ defines the discrete position of winning neuron i, both of which are measured in the discrete output space.

A unique feature of the Kohonen learning algorithm is that the area of the neighborhood shrinks over time. This is accomplished by making the radius of the neighborhood shrink over time. Therefore, $\sigma^2(n)$ decays exponentially as

$$\sigma(n) = \sigma_0 \exp(-n/\tau_1), \qquad \mathrm{n} = 0,1,2,\ldots.. \tag{2.10}$$

where σ_0 is the initial value of σ, and τ_1 is another time constant.

Thus, the neighborhood function $h_{j,i}(n)$ shrinks with every iteration as shown in Figure 2.8. Over time the neighborhood will shrink to the size of just one node which is the winning neuron only.

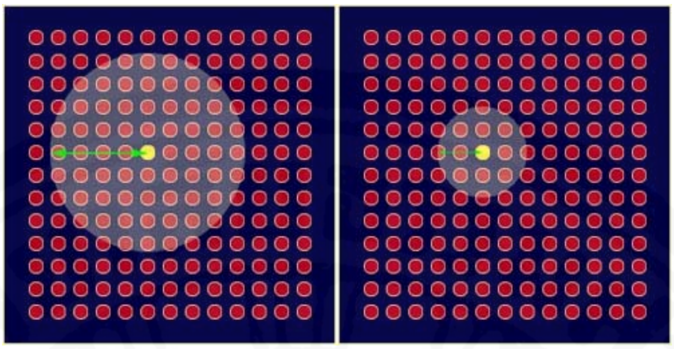


Figure 2.8 Shrinking neighborhood in every iteration.

3. *Adaptive Process*

For the network to be self-organizing, the synaptic weight vector $\mathbf{w}_j$ of neuron j in the network is required to change in relation to the input vector **v**.

The weights of the winner are then updated as

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta(n) \cdot h_{j,i}(n) \cdot (\mathbf{v} - \mathbf{w}_j(n)) . \qquad (2.11)$$

The function $\eta(n)$ defines the ratio of learning that should start at an initial value η_0, and then decrease gradually with increasing time n

$$\eta(n) = \eta_0 \exp(-n/\tau_2), \qquad n = 0, 1, 2, ... \qquad (2.12)$$

where τ_2 is a time constant.

SOM leads to a topological ordering of the feature map in the input space in the sense that neurons that are adjacent in the lattice will tend to have similar synaptic weight vectors [9].

Therefore, the segmentation of motion vectors can be done successfully by using self-organizing feature map. From the previous section, we tested GSTM on synthetic Lena image sequence in which eight objects are moving in eight different directions. Motion vectors estimated by GSTM to the sequence are sent to an SOM where there are two input neurons and 4×4 output neurons. The two input neurons which represent motion vectors in x and y directions (u, v) are connected to every output neuron (feature map) along which some weights are predetermined.

To visualize the classification results by the SOM, we map motion vectors to a color image where u is assigned to a red plane and v is to green one. As shown in Figure 2.9, classification results are expressed with 16 different colors in the output neurons or final feature map after the convergence of its learning step. Note that similar colors are automatically assigned to the motion vectors of similar directions and brightness of each color indicates the magnitude of the motion.

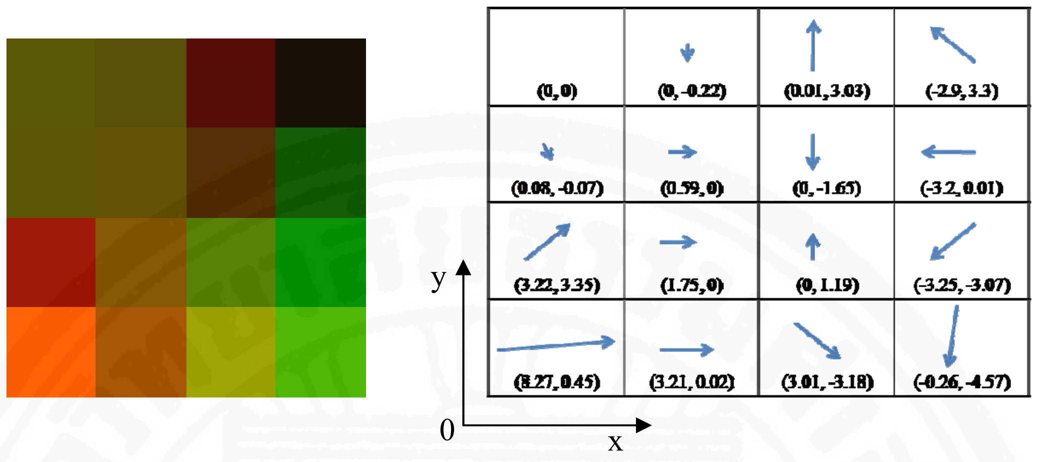


Figure 2.9 Final feature maps of SOM and corresponding motion vectors for each color

There are several parameters involved in the computation of SOM. The choice of values for each parameter is important for the computation since the parameters such as width of the neighborhood function and learning-rate may not be optimal. In the adaptive process, SOM algorithm may take learning as many as 1000 iterations or possibly more. The learning rate parameter $\eta(n)$ should begin with a value close 0.1 and after decrease gradually, it should remain above 0.01. The desirable values of initial learning rate η_0 and time constant τ_2 are 0.1 and 1000, respectively [9]. Since there are no fixed optimal values for all the parameters, we consider the value of each parameter empirically.

In this experiment, we use the initial width of the neighborhood function σ_0 to be 1.2 with time constant τ_1 to be 1000/log (σ_0) and initial learning rate η_0 to be 0.1 with time constant τ_2 to be 1000. Each parameter is chosen by trial and error method and they are suitable for the best tuning of SOM for our image sequences. The number of iterations for the training is set to be 60,000 for Lena synthetic image sequence. Using the final weights, the input image is segmented. The eight moving objects are successfully detected from the background.

Figure 2.10 shows the result of motion vectors estimated by GSTM and segmented image after 60,000 iterations. Each color represents the directions of the moving objects. In this case, the algorithm choose red for the motion in vertical, green for horizontal and colors between red and green for different diagonal directions.

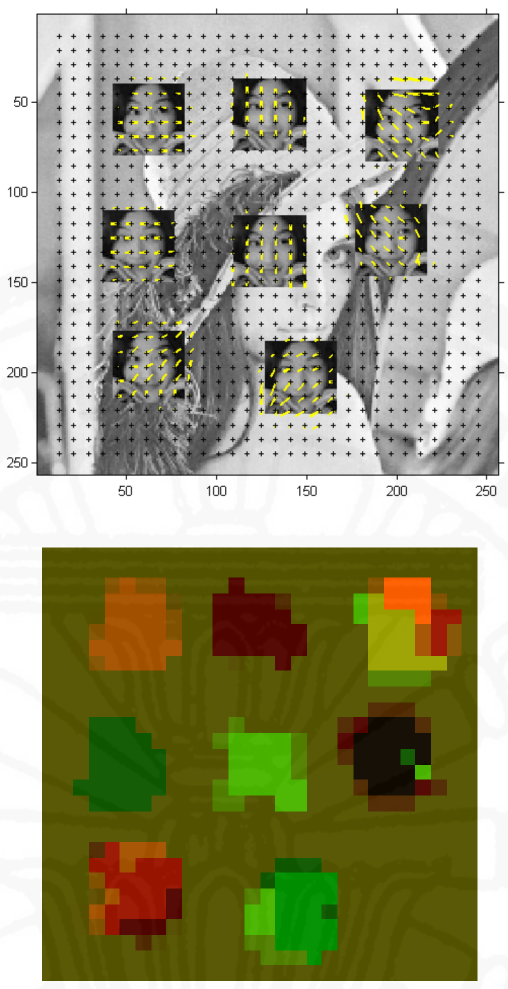


Figure 2.10 Classification Result corresponding to each motion vector