Meshed Snakes

Cheng-Yuan Liou, Quan-Ming Chang

Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan, R.O.C.
Tel: 8862-3625336 ext 515, Fax: 8862-3628167, Email: cyliou@csie.ntu.edu.tw.

Abstract

A mesh model which is composed of many element units what are called snkaes is presented. The boundary sides surrounding a hole on the mesh form an element snake. These element snakes configure the whole mesh. We devise a Hopfield type energy function for these snakes. This energy function will guide the mesh for various applications, such as image coding, surface modeling, image recognition.

1 Introduction

Mesh generation has been studied for object modeling, nonuniformly sampling, reconstruction visual data, and so on. A mesh is designed for representing an image with a small number of nodes that will convey most of the information about the image. Weiss [8] presented a method of reconstructing a shape from incomplete and noisy data. A major advantage of the method is its ability to handle shapes containing sharp corners or edges using the same mechanism. Terzopoulos et al. [4, 5] proposed the adaptive mesh model which is a dynamic network of nodal masses interconnected by adjustable springs. The mesh can automatically adapt to the variations in the input data. Wang and Lee [7] proposed an active mesh representaion for image sequences tracking and presented an energy minimization approach for mesh generation.

In this paper, we present an energy minimization approach for mesh generation. Our approach is quiet different from that described in [7]. The mesh we generate is composed of many basic units- "snakes" [3]. We define an energy function for the mesh and then minimize it following the Hopfield neural network [2]. The nodal points in the mesh are more densely distributed in regions containing interesting features such as edges and corners. Furthermore, we can generate meshes with variable shapes and deal with image sequences tracking.

2 The snakes

Kass et al.[3] proposed an energy-minimizing active contour model (snake) which could drive a set of points to lie on features of interest ,e.g. edges, in image. The model has the advantage that the final form of a contour can be influenced by feedback from a high level process. If we represent the position of a snake parametrically by $\mathbf{v}(s) = (x(s), y(s))$, the energy function can be written as

$$\begin{aligned} E^*_{snake} &= \int_0^1 E_{snake}(\mathbf{v}(s))ds \\ &= \int_0^1 [E_{int}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) + E_{con}(\mathbf{v}(s))]ds \end{aligned} \quad (1)$$

0-7803-3210-5/96 $4.00

where E_{int} represents the internal energy of the spline due to bending or discontinuities, E_{image} represents the image forces, and E_{con} is the external constrain forces. The internal spline energy can be written

$$E_{int} = (\mu(s)|\ \mathbf{v}_s(s)\ |^2 + (\nu(s)|\ \mathbf{v}_{ss}(s)\ |^2)/2 \tag{2}$$

In the above equation, the first-order term causes the snake to behave like a string (i.e. resist stretching) and the second-order term causes the snake to behave like a rod (i.e. resist bending). Adjusting the weights $\mu(s)$ and $\nu(s)$ controls the relative importance of the string and rod terms. The image force in eqn(1) is sourced from various events such as lines or edges.

Many methods have been proposed to minimize this energy function [1, 3, 6, 9]. Basing on the idea presented in [6, 9], we solve the problem with Hopfield neural network. Our approach is different from that described in [6] and is more suitable for generating mesh. We will show how the snake model can be extended to form the mesh in next section.

For simplicity, we consider the internal term and the image term in eqn(1). When we discretize the energy function, we can write

$$E_{snake} = \sum_{i=1}^{n} \mu_i\ |\ \mathbf{v}_i - \mathbf{v}_{i-1}\ |^2 + \nu_i\ |\ \mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1}\ |^2 + E_{image}(\mathbf{v}_i) \tag{3}$$

where $E_{image}(\mathbf{v}_i) = -\gamma_i\ |\ \nabla I(\mathbf{v}_i)\ |^2$ denoting the negated intensity gradient at point $\mathbf{v}_i$. In most cases, the weighting factors μ_i and ν_i may be assumed to be two constants. However, it is not the case for our model and the choice of them will be discussed later. Note that every term in eqn(3) should be normalized, and the first term and the second term can be chosen in other forms. The details can be found in [9].

To find the minimization of eqn(3), a two-dimensional Hopfield neural network is used. The network consists of $n * m$ mutually interconnected neurons where n is the number of points of the snake and m is the number of neighbors around each point of the snake. The energy function is computed at $\mathbf{v}_i$ and each of its neighbors (Fig. 1). The location having the smallest value of energy function is chosen as the new position of $\mathbf{v}_i$. Let $v_{i,p}$ denote the binary state of the (i,p)th neuron (1 for firing and 0 for resting). Let $x_{i,p}$ and $y_{i,p}$ be the x and the y coordinates, respectively, of the neighboring point (i,p), and $g_{i,p}$ be the negated intensity gradient at the neighboring point (i,p) defined as $g_{i,p} = -\gamma_i\ |\ \nabla I(x_{i,p}, y_{i,p})\ |^2$.

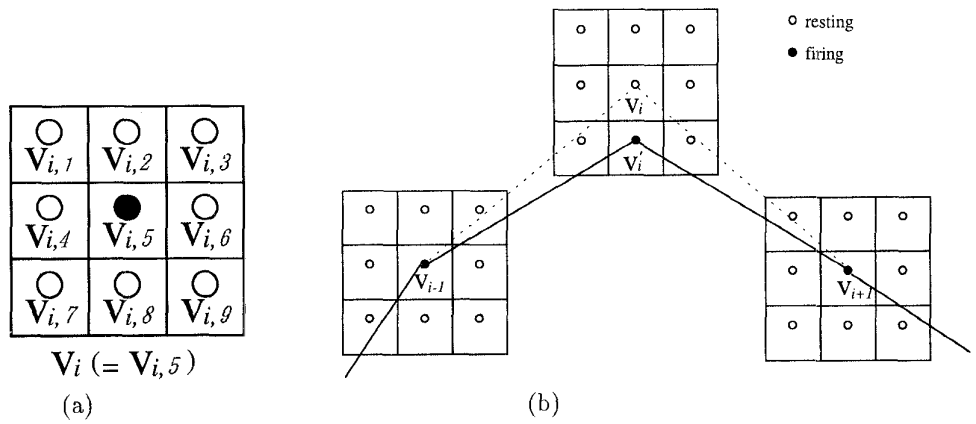


Figure 1: (a) Each node in the snake has eight neighbors. Thus, $m = 9$ in our case (including itself). (b) The energy function is computed at $\mathbf{v}_i$ and each of its eight neighbors. The location $\mathbf{v}_i'$ which has the smallest value of the energy function is selected as the new position of $\mathbf{v}_i$.

Then the Hopfield type energy for eqn(3) is

$$
\begin{aligned}
E_{snake} = & \sum_{i=1}^{n}\{w_1\alpha_{i,i-1}[(\sum_{p=1}^{m}x_{i,p}v_{i,p}-\sum_{p=1}^{m}x_{i-1,p}v_{i-1,p})^2+(\sum_{p=1}^{m}y_{i,p}v_{i,p}-\sum_{p=1}^{m}y_{i-1,p}v_{i-1,p})^2] \\
& +w_2\beta_{i-1,i+1}[(\sum_{p=1}^{m}x_{i-1,p}v_{i-1,p}-2\sum_{p=1}^{m}x_{i,p}v_{i,p}+\sum_{p=1}^{m}x_{i+1,p}v_{i+1,p})^2 \\
& +(\sum_{p=1}^{m}y_{i-1,p}v_{i-1,p}-2\sum_{p=1}^{m}y_{i,p}v_{i,p}+\sum_{p=1}^{m}y_{i+1,p}v_{i+1,p})^2] \\
& +w_3\gamma_i(\sum_{p=1}^{m}g_{i,p}v_{i,p})+w_4\varepsilon_i[(1-\sum_{p=1}^{m}v_{i,p})^2]+w_5\tau_i\sum_{p=1}^{m}v_{i,p}(1-v_{i,p})\}
\end{aligned} \tag{4}
$$

where the fourth term is included to imply that only one point can be selected from the neighboring points $(i,1),(i,2),...,(i,m)$ for each i. The last term is called self-connection eliminating term, which will change the diagonal value of the interconnection matrix of the Hopfield network (see next section). Consequently, it will influence the convergence property of the network. Note that μ_i and ν_i in eqn(3) are substituted by $\alpha_{i,i-1}$ and $\beta_{i-1,i+1}$ respectively. $\alpha_{i,j}$ and $\beta_{i,j}$ are defined as:

$$
\begin{cases}
\alpha_{i,j} = 1-(g_{i,5}+g_{j,5})/10 \\
\beta_{i,j} = 15+(g_{i,5}+g_{j,5})/40
\end{cases}
$$

The choice of $\alpha_{i,j}$ and $\beta_{i,j}$ are not limited. Different types will produce different results. Here we employ the simple forms to illustrate the concept that those neighboring nodes locate on the points having larger gradient are closer than those which locate on the points having smaller gradient. The value of $g_{i,p}$ are normalized to range from 0 to -255. The constants w_1, w_2, w_3, w_4 and w_5 are used to weight the five terms.

3 From snake to mesh

A mesh is formed with a finite number of nonoverlapping polygonal elements. To present clearly, we only consider the case which all the elements in mesh have the same number of nodes. Fig. 2(a) illustrates a quadrilateral mesh. Each element in the mesh is taken to be a snake. Thus, if we have $row * col$ nodes in the mesh, we'll have $(row-1)*(col-1)$ snakes. Each element (snake) will shrink to one point if there is no constrains imposed upon it. However, since every internal node in the mesh is shared by four snakes, the forces impose upon the node will keep balance if there is no image forces (Fig. 2(b)). The corner nodes are fixed and the boundary nodes are constrained to slide along the boundary the image frame.

Then, we can define the energy function of the mesh as

$$
E_{mesh} = \sum_{s=1}^{Sn} E_{snake(s)} \tag{5}
$$

where Sn is the total number of snakes in the mesh and $E_{snake(s)}$ is defined in eqn(4). The Lyapunov function of a two-dimensional Hopfield network is written as:

$$
E_{Hopfield} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{p=1}^{m}\sum_{j=1}^{N}\sum_{q=1}^{m}W_{i,p;j,q}v_{i,p}v_{j,q}-\sum_{i=1}^{N}\sum_{p=1}^{m}I_{i,p}v_{i,p} \tag{6}
$$

where N is the number of nodes of the mesh. Rearranging eqn(5) and comparing it with eqn(6) we have:

$$
W_{i,p;j,q} = (T_1(i,j)+T_2(i,j)+T_3(i,j))*(x_{i,p}x_{j,q}+y_{i,p}y_{j,q})+T_4(i,j) \tag{7}
$$

where

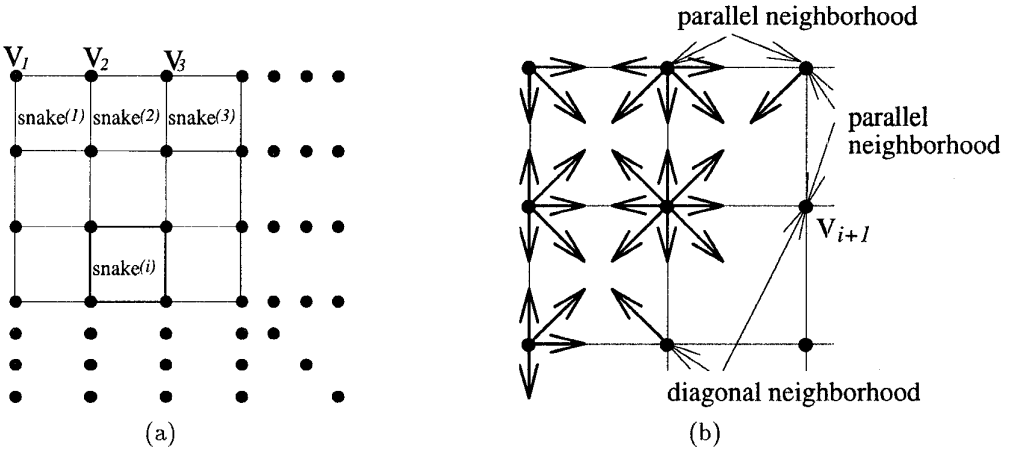


Figure 2: (a) The mesh is composed of many snakes. Each snake has four nodes in it. (b) The snake has the tendency to shrink if there are no constraints imposed upon it. The internal nodes in the mesh is influenced by eight forces.

$$T_1(i,j) = [\sum_{k=1}^{N} -2w_1\alpha_{i,k}\phi_1(i,k)(2 - B(i)B(k)) - 4w_2\beta_{i,k}\phi_2(i,k) \\ -8\sum_{l=1}^{N} w_2\beta_{k,l}\phi_1(i,k)\phi_1(i,l)\phi_2(k,l)](2 - \delta_{p,q})\delta_{i,j}$$

$$T_2(i,j) = 4w_1\alpha_{i,j}\phi_1(i,j)[2 - B(i)B(j)]$$

$$T_3(i,j) = 8w_2\{\sum_{k=1}^{N}[\beta_{i,k}\phi_1(j,k)\phi_2(i,k) + \beta_{j,k}\phi_1(i,k)\phi_2(j,k)]\phi_1(i,j) - \beta_{i,j}\phi_2(i,j)\}$$

$$T_4(i,j) = [\sum_{k=1}^{N}(-2w_4\varepsilon_i(2 - \delta_{p,q}) + 2w_5\tau_i\delta_{p,q})\phi_2(i,k)(2 - B(i)B(k))]\delta_{i,j}$$

and

$I_{i,p} = \sum_{k=1}^{N}[(-w_3\gamma_i g_{i,p} + 2w_4\varepsilon_i - w_5\tau_i)\phi_2(i,k)(2 - B(i)B(k))]$
respectively, where $\delta_{i,j}$ is the Kronecker delta function and $B(i) = 1$ if node i if the boundary node (including corner nodes), else $B(i) = 0$. $\phi_1(i,j)$ and $\phi_2(i,j)$ are the neighborhood functions defined as:

$$\phi_1(i,j) = \begin{cases} 1 & \text{,if i and j are parallel neighborhood (fig. 3(b)).} \\ 0 & \text{,otherwise} \end{cases}$$

$$\phi_2(i,j) = \begin{cases} 1 & \text{,if i and j are diagonal neighborhood (fig. 3(b)).} \\ 0 & \text{,otherwise} \end{cases}$$

A neuron (i,p) in the network receives weighted $W_{i,p;j,q}$ input from each neuron (j,q) and a bias input $I_{i,p}$. The total input, $net_{i,p}$, of the (i,p)th neuron is computed as

$$net_{i,p} = \sum_{j=1}^{N}\sum_{q=1}^{m} W_{i,p;j,q}v_{j,q} + I_{i,p} \tag{8}$$

Then the (i,p)th neuron is updated as follows:

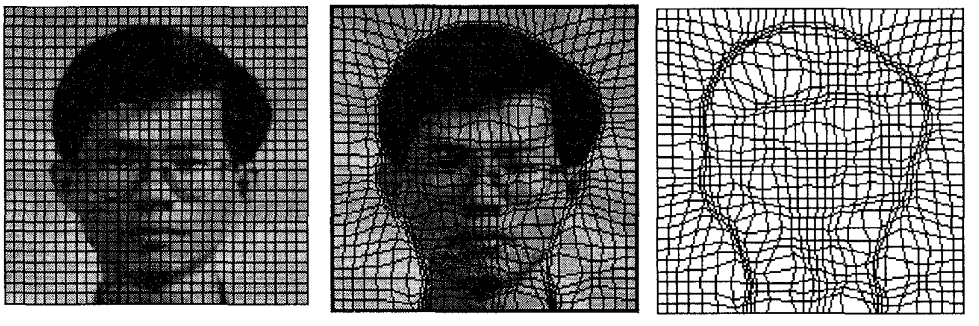
$$v_{i,p} = \begin{cases} 1 & \text{,if } net_{i,p} \geq 0 \\ 0 & \text{,if } net_{i,p} < 0 \end{cases}$$

The rule described above for updating the neuron is applied in an asynchronous fashion. This means that for a given time, only a single neuron (which is selected randomly) is allowed to update its output. The iterative algorithm is presented below:

Procedure

```
do      /* loop to update the mesh */
        initialize the matrix W ans I used in eqn(8);
        Moved = 0;
        do      /* loop to update the network until it converges in this epoch*/
                ChangeCount=0;
                UpdateCount=0;
                do      /* loop to move points to new position among the neighboring points */
                        randomly select a node from the mesh for updating;
                        update the position of this selected node;
                        (calculate those neurons corresponding to this node and its neighbors)
                        if one of states of these neurons changes ,ChangeCount+ = 1 and Moved = 1;
                        UpdateCount+ = 1;
                until Updatecount = N;          /* N is the number of the nodes of the mesh */
        until ChangeCount = 0;
until moved = 0;
```

The experimental results is showed in Fig. 3.



(a) (b) (c)

Figure 3: (a) The initial mesh. (b) The final mesh. (c) The converged mesh only.

4 Variable region mesh

The corner nodes and the boundary nodes of the mesh model we define above are constrained. These constraints can be removed to produce meshes with variable regions. Initially, γ_i is set to be zero for internal nodes. By this way, the mesh will shrink until the boundary nodes of the mesh are bounded on those points which have large gradient (Ideally, they are the contour of the object.). Then, we let the mesh behave the same as the mesh model we describe in previous section. The advantage of forming variable region mesh is that it can reduce the data to record the object in the image. In many cases we are interested in the object itself but not the background. Furthermore, the object encoded by the variable mesh is more accurate than that by the normal mesh since the snake can find the contour of the object accurately. The experimental results is showed in Fig. 4. This mesh can track the motion of the head, such as pitch, swing, and yaw.

5 Conclusion

In this paper we have presented a new method of mesh generation. The method is based on an energy-minimizing approach which we employ Hopfield neuron network to solve it. The experimental results show that the nodes in the mesh are more densely distributed in regions containing interesting features such as edges. Also, our model can generate variable region mesh which is showed in Fig. 4. Furthermore, image sequences tracking and image recognition can be achieved. Although we only show the 2-D case, this model can easily extended to two types of 3-D mesh. The first type is used in object surface modeling.

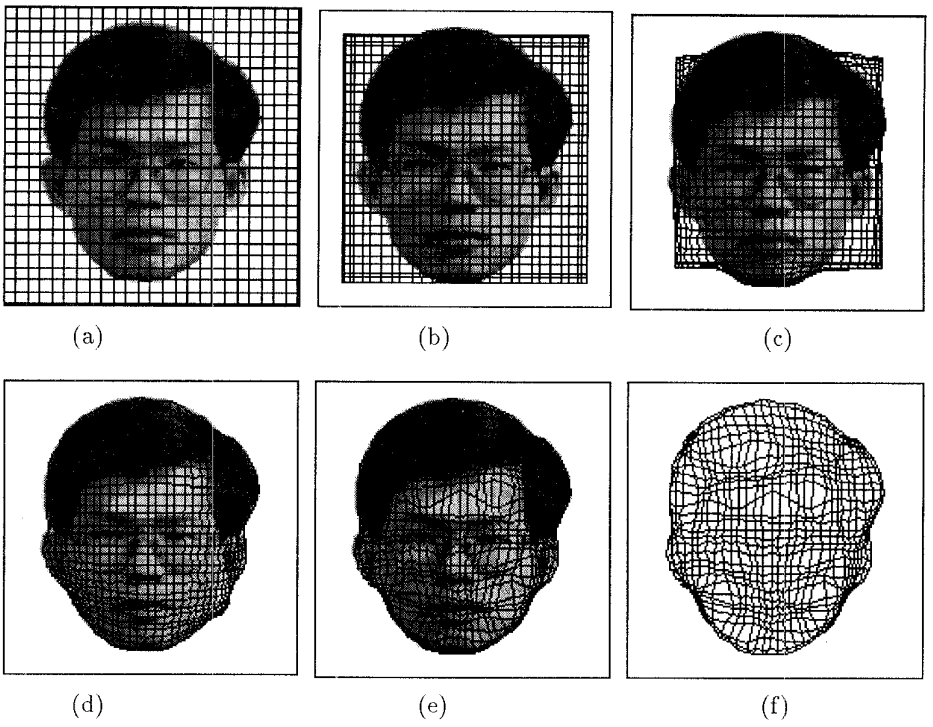


Figure 4: (a) The initial mesh. (b)(c)(d) The intermediate states. (e) The final result. (f) The convergenced mesh only.

In the second type, the mesh is composed of cubes which is composed of six snakes. Thus, each internal nodes is shared by twelve snakes.

References

[1] A. A. Amini, S. Tehrani, and T. E. Weymouth, "Using Dynamic Programming for Minimizing the Energy of Active Contours in the Presence of Hard Constraints," *Proc. IEEE Conf. Computer Vision*, pp. 95–99, 1988.

[2] J. J. Hopfield, and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biolog. Cybern.*,**52**, pp. 141–152, 1985.

[3] M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active Contour Models," *Proc. of Int. Conf. on Computer Vision*, pp. 259–268, 1987.

[4] D. Terzopoulos and M. Vasilescu, "Sampling and Reconstruction with Adaptive Meshes," *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR'91)*, pp. 70–75, 1991.

[5] M. Vasilescu and D. Terzopoulos, "Adaptive Meshes and Shells," *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR'92)*, pp. 829–831, 1992.

[6] C. T. Tsai, "Minimising The Energy of Active Contour Model Using A Hopfield Network," *IEE Proceedings-E*, vol. 140, no. 6, pp. 297–303, 1993.

[7] Y. Wang and O. Lee, "Active Mesh — A Feature Seeking and Tracking Image Sequence Representation Scheme," *IEEE Trans. on Image Procedding*, vol. 3, no. 5, pp. 610–624, 1994.

[8] I. Weiss, "Shape Reconstruction on a Varying Mesh," *IEEE Trans. on Pattern Analysis and Machine Intelligence.*, vol. 12, no. 4, pp. 345–362, 1990.

[9] D. J. Williams and M. Shah, "A Fast Algorithm for Active Contours and Curvature Estimation," *CVGIV: Image Understanding*, vol. 55, no. 1, pp. 14–26, 1992.