Grid Smoothing: A Graph-Based Approach

Guillaume Noel, Karim Djouani, and Yskandar Hamam

French South African Institute of Technology
Tshwane University of Techology, Pretoria, South Africa

Abstract. In the past few years, mesh representation of images has attracted a lot of research interest due to its wide area of applications in image processing. In the mesh framework, an image is represented by a graph in which the nodes represent the pixels and the edges reflect the connectivity. The definition of the most adapted mesh for a given image is a challenge in terms of computation cost and information representation. In this paper, a new method for content adaptive mesh representation of gray scale images, called grid smoothing, is presented. A cost function is defined using the spatial coordinates of the nodes and the gray levels present in the image. The minimisation of the cost function leads to new spatial coordinates for each node. Using an adequate cost function, the grid is compressed in the regions with large gradient values and relaxed in the other regions. The result is a grid which better fits the objects in the image. The mathematical framework of the method is introduced in the paper. An in-depth study of the convergence is presented as well as results on real gray scale images.

Keywords: Content adaptative mesh, grid smoothing, image coding, non-linear optimisation.

1 Introduction

Mesh representation of images has received a lot of attention in the recent years due to its wide range of applications in the image processing domain such as image compression and coding [1], low rate video coding[2], [3] and image processing for medical application [4]. In the mesh representation of a gray scale image, the information is no longer coded into a matrix of real numbers. Instead, a graph is defined, in which the nodes represent the pixels and the edges reflect the connectivity between the pixels. The main challenges faced when generating the mesh representation of an image is the accuracy of the representation of the information contained in the image, the size of the mesh and the computation time. Various methods have been proposed for content adaptative mesh generation, the common trend between them being to select particular pixels in the image (the one carrying the largest part of the information) and to create a mesh based on this set of point using a Delaunay triangulation scheme. For example, Yang [5] analysed the second order derivative of a pixel to consider it as a significant pixel and then a node in the mesh while Ramponi [6] selected the meaningful pixels by looking at their normalized skewness. Using a different

I. Bloch and R.M. Cesar, Jr. (Eds.): CIARP 2010, LNCS 6419, pp. 153–160, 2010.

methodology, Sarkis [7] generated the mesh by dividing, in a recursive manner, an initial triangle. The decision of dividing a triangle into two is based on its ability to represents the lying pixels in the triangle. The method presented in this paper differs in the approach. The main idea of the grid smoothing is, starting from a uniform grid, composed by squares or triangles depending on the connectivity chosen, to reshape the grid according to the information (gray levels) contained in the image. The grid smoothing relies on the minimisation of a cost function leading to a compression of the grid in the regions with large gradient values and a relaxing in the other regions. Section 2 of this paper presents the graph-based representation of an image while section 3 exposes the mathematical framework of the grid smoothing as well as the convergence. Simulations results and example of grid smoothing on real images may be found in Section 4. Conclusion and recommendations are underlined in section 5.

2 Graph-Based Image Representation

Our input data is a graph $G = (V, E)$, embedded in the 3D Euclidian space. Each edge e in E is an ordered pair (s, r) of vertices, where s (resp. r) is the sending (resp. receiving) end vertex of e [8]. To each vertex v is associated a triplet of real coordinates x_v, y_v, z_v. Let C_{ve} be the node-edge incidence matrix of the graph G, defined as:

$$C_{ve} = \begin{cases} 1 \text{ if } v \text{ is the sending end of edge } e \\ -1 \text{ if } v \text{ is the receiving end of edge } e \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

In the rest of the paper, the node-edge matrix C_{ve} will also be denoted C.

Considering an image with M pixels, X, Y and Z respectively represent $[x_1, ..., x_M]^t$, $[y_1, ..., y_M]^t$ and $[z_1, ..., z_M]^t$. X and Y are at first uniformly distributed (coordinates of the pixels in the plane), while Z represents the gray level of the pixels. Each pixel in the image is numbered according to its column and then its rows. We define L as the number of edges in the graph. C is consequently a matrice with L rows and M columns.

3 Optimisation-Based Approach to Grid Smoothing

3.1 General Framework

A cost function is introduced to fit the object of the image with the grid. The main idea is that the regions where the variance is small (low gradient) require fewer points than the regions with a large variance (large gradient). The grid smoothing techniques will move the points of the grid from small variance regions to large variance regions. To achieve this goal, a cost function J is defined as follows:

$$J = J_X + J_Y \tag{2}$$

where

$$J_X = \frac{1}{2}\left[\left(X - \hat{X}\right)^t Q\left(X - \hat{X}\right) + \theta\left(X^t A X\right)\right] \tag{3}$$

and

$$J_Y = \frac{1}{2}\left[\left(Y - \hat{Y}\right)^t Q\left(Y - \hat{Y}\right) + \theta\left(Y^t A Y\right)\right] \tag{4}$$

where $\hat{X}$ and $\hat{Y}$ are respectively the initial values of X and Y, A being equal to $C^t \Omega C$.
The matrix Ω is defined as follows:

$$\Omega_{k,k} = (z_i - z_j)^2 \tag{5}$$

where node i is the sending end of the edge k and node j the receiving end. Ω and Q are square diagonal matrices which dimensions are respectively $L \times L$ and $M \times M$.

The first term in the expression of the cost function is called the *attachment* as it penalises the value of the cost function if the coordinates are too far from the original values. It is introduced to avoid large movement in the grid [8]. θ is a real number and is acting as weighing factor between the terms of the cost function. As a result of the definition of Ω, the minimisation of J is leading to the reduction of the areas of the triangle formed by two connected points and the projection of one of the point on the Z-axis. The edges in the image act as attractors for the points in the grid. As a consequence, the edges are better defined in terms of location and steepness in the smoothed grid.

3.2 Convergence of the Cost Function

The following sub-sections present the proof of convergence in two scenari: minimisation of a cost function with attachment and a cost function with attachment and fixed points.

Cost function with attachment. This section focuses on proving the existence of a unique solution for the minimisation problem presented above. The solution is presented for J_X only. The proof for J_Y can be derived in a similar manner. The cost function of the first order with attachment may be expressed as:

$$J_X = \frac{1}{2}\left[\left(X - \hat{X}\right)^t Q\left(X - \hat{X}\right) + \theta\left(X^t A X\right)\right] \tag{6}$$

The gradient of the first order cost function J_X with attachment is:

$$\nabla_x J_X = Q\left(X - \hat{X}\right) + \theta A X \tag{7}$$

At the optimum, the gradient is equal to zero. Let X_{opt} be the optimal solution for X. X_{opt} may be expressed as:

$$X_{opt} = \left(Q + \theta A\right)^{-1} Q\hat{X} \tag{8}$$

The above equation shows that a unique optimal solution (it may be shown that $Q+\theta A$ is inversible) exists for the minimisation problem and that for small scale problem, the solution may be obtained easily. For large scale problem, a gradient descent method may be used. Let X_{n+1} and X_n be respectively the values of X at iteration $n+1$ and n. X_{n+1} is equal to

$$X_{n+1} = X_n - \alpha_n \nabla_x J_X = X_n - \alpha_n \left(Q\left(X - \hat{X}\right) + \theta A X \right) \tag{9}$$

α_n is the step and may be chosen optimal or not. An optimal step leads to a smaller number of iterations while increasing the processing power required for the optimisation. The optimal step α_n may be expressed by:

$$\alpha_n = \frac{\nabla_x J^t \nabla_x J}{\nabla_x J^t \left(Q + \theta A\right) \nabla_x J} \tag{10}$$

Cost with fixed points and attachment. The cost function with attachment results in a grid whose size might differ from the original grid size. A solution to conserve the original size is to fix the coordinates of the outer points of the grid. Let the X coordinates be partitioned into two parts, variable coordinates 'x' and fixed coordinates 'a' giving

$$X = \begin{bmatrix} x \\ a \end{bmatrix} \tag{11}$$

Then the first order cost function without attachment is

$$J_x = \frac{1}{2} \left(\left[(x - \hat{x})^t \; 0 \right] Q \begin{bmatrix} (x - \hat{x}) \\ 0 \end{bmatrix} + \theta \left[x^t \; a^t \right] \begin{bmatrix} C_x^t \\ C_a^t \end{bmatrix} \Omega \left[C_x \; C_a \right] \begin{bmatrix} x \\ a \end{bmatrix} \right) \tag{12}$$

Expanding the above equation gives

$$J_x = \frac{1}{2} \left[(x - \hat{x})^t Q_x (x - \hat{x}) + \theta x^t C_x^t \Omega C_x x + 2\theta x^t C_x^t \Omega C_a a + \theta a_t C_a^t \Omega C_a a \right] \tag{13}$$

The gradient of J_x with respect to x is

$$\nabla_x J_x = Q_x (x - \hat{x}) + \theta C_x^t \Omega C_x x + \theta C_x^t \Omega C_a a \tag{14}$$

Setting the gradient to zero gives

$$x = -\left[Q_x + \theta C_x^t \Omega C_x \right]^{-1} \left[Q_x \hat{x} - \theta C_x^t \Omega C_a a \right] \tag{15}$$

This gives the exact solution for the coordinates x.
Let x_{n+1} and x_n be x at iteration $n+1$ and n then

$$x_{n+1} = x_n - \alpha_n \nabla_x J_x \tag{16}$$

The gradient of J_x at the point x_{n+1} is equal to

$$\nabla_x J_{x_{n+1}} = \nabla_{x_n} J_x - \alpha_n Q_x \nabla_x J_{x_n} - \alpha_n \theta C_x^t \Omega C_x \nabla_x J_{x_n} \tag{17}$$

The optimal step condition may by expressed by $\nabla_x J_{x_n}^t . \nabla_x J_{x_{n+1}} = 0$
It leads to:

$$\alpha_n = \frac{\nabla J^t \nabla J}{\nabla J^t \left(Q_x + \theta C_x^t \Omega C_x\right) \nabla J} \tag{18}$$

Stopping criterion. As mentioned earlier, for large scale problem, the minimisation uses a gradient descent algorithm as it is computationally expensive to inverse very large matrices. Three gradient methods are used for the simulation, namely the steepest descent gradient with fixed step, the steepest descent gradient with optimal step and the conjugate gradient with optimal step. The descent gradient methods are iterative process and require a stopping criterion ϵ to stop the iterations. The chosen criterion is the simulation is the norm of the gradient ∇J. The iterative process continues while $\nabla J^t \nabla J \geq \epsilon$. When it is possible, the comparison between the exact coordinates given by the inversion of the matrix and the result of the gradient descent algorithm is small and is of the order of ϵ. For example, if $\epsilon = 10^{-3}$, the difference between the exact coordinates (matrix inversion) and the coordinates obtained through the gradient descent is 10^{-3} of the width of a pixel.

4 Simulations

The simulations were performed using a standard laptop (1.87 GHz processor, 2GB RAM and *Windows Vista SP1* as operating system) and *Matlab R14 Service Pack 2*. The algorithms are tested on an image coming from the *Matlab* library.The computing time is obtained for 15 executions of the program and the mean value is indicated. Tables 1 and 2 show that the conjugate gradient with optimal step performs much better than the fixed step and optimal step descent gradient methods. For example, for an image of 300×300 pixels (90000 nodes), the number of iterations is limited to 76 while the computing time is 5s. The number of iterations for the optimal step gradient descent is almost ten times this figure while the fixed step method requires 20 times more iterations. The computation times for the fixed and optimal step methods are in the same range. The computation of the optimal step at each iteration doubles the time of the fixed-step iteration. The conjugate gradient descent is consequently the most suitable method out of the three tested for the grid smoothing. Consequently, the rest of the simulations uses the conjugate gradient descent. Table 3 shows that the stopping criterion has little effect on the number of iterations and the computation time. From a qualitative point of view, and $\epsilon = 10^{-4}$ looks like a good compromise between the result and the computing time. Table 4 displays the number of iteration and the computing time required for various values of θ. It may be seen that θ influences deeply the convergence. Small values of θ lead to speedy convergence whereas large values require more iteration. θ is the weighing factor between the two terms of the cost function. With θ small, the attachment term dominates the cost function and very little displacement of the points is allowed (quick convergence). The opposite effect is obtained with a large θ. From a qualitative point of view, $\theta = 0.05$ looks like a convenient value. This however depends on the application. Fig. 1 shows the results of the grid smoothing on two images for various values of θ. It may be seen that the level of compression (and noise) in the grid increases with θ. In both cases, it may be observed that the grid fits the object present in the images.

Table 1. Convergence in iterations for $\epsilon = 10^{-4}$ and $\theta = 0.005$

Number of points	Fixed step	Optimal step	Conjugate gradient
100	363	190	23
2500	543	282	37
10000	1174	593	60
90000	1440	707	76

Table 2. Convergence in seconds for $\epsilon = 10^{-4}$ and $\theta = 0.005$

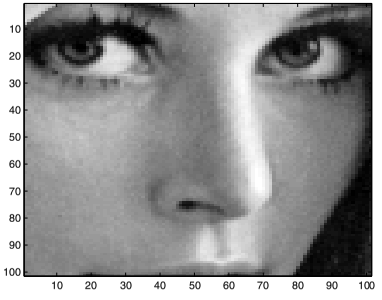
Number of points	Fixed step	Optimal step	Conjugate gradient
100	1.9×10^{-4}	2.3×10^{-4}	1.8×10^{-4}
2500	8.3×10^{-1}	9.0×10^{-1}	5.9×10^{-2}
10000	6.8	7.6	3.3×10^{-1}
90000	1.2×10^{2}	1.3×10^{2}	5.0

Table 3. Convergence of the conjugate gradient in iterations and seconds for $\theta = 0.05$

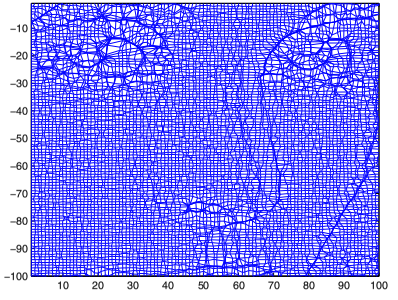
	$\epsilon = 10^{-3}$	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$
Iterations	240	263	287	312
Time (s)	15.1	16.4	17.9	19.7

Table 4. Convergence of the conjugate gradient in iterations and seconds for $\epsilon = 10^{-4}$

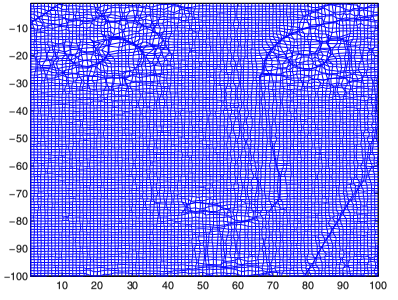
Number of points	θ	Iterations	Time (s)
2500	5×10^{-3}	37	7.4×10^{-2}
	5×10^{-2}	126	1.6×10^{-1}
	5×10^{-1}	432	5.1×10^{-1}
	5	1116	1.3
10000	5×10^{-3}	60	3.3×10^{-1}
	5×10^{-2}	206	1.3
	5×10^{-1}	701	3.6
	5	2162	1.1×10^{1}
40000	5×10^{-3}	66	1.8
	5×10^{-2}	228	5.7
	5×10^{-1}	784	1.9×10^{1}
	5	2660	6.8×10^{1}
90000	5×10^{-3}	76	5.0
	5×10^{-2}	263	1.6×10^{1}
	5×10^{-1}	920	6.1×10^{1}
	5	3183	2.1×10^{2}



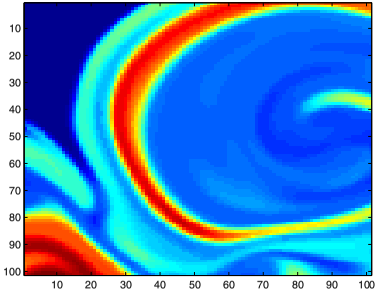
(a) Lena (source: Matlab)



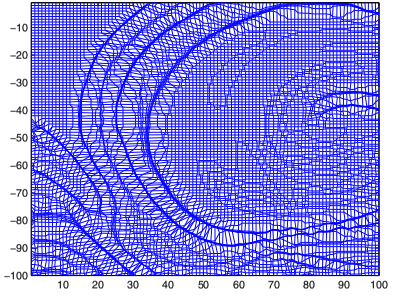
(b) Grid smoothing with $\theta = 1$



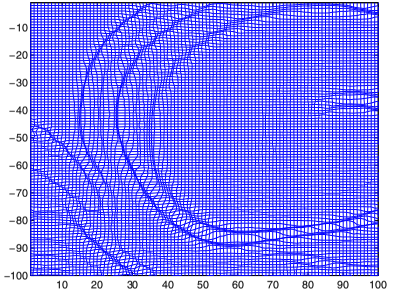
(c) Grid smoothing with $\theta = 0.005$



(d) Turbulent flux (source: Matlab)



(e) Grid smoothing with $\theta = 0.5$



(f) Grid smoothing with $\theta = 0.05$

Fig. 1. Result of the grid smoothing

5 Conclusions

A new framework to represent an image is presented in the paper. Based on the graph representation of an image, the grid smoothing process modifies the coordinates of the points in the grid to fit the objects in the image. The convergence is shown and the conjugate gradient descent is recommended for the optimisation in large scale problems. The results obtained are promising and may be used in many applications (edge detection, compression or super-resolution). The main challenge of the method presented is the choice of the parameter θ. This parameter should be chosen according to the desired application, keeping in mind the computation time implication of this choice. The computational cost of the grid smoothing approach is high even when working with sparse matrices. Properties af the various matrices will be analysed in detail to achieve a faster grid smoothing. Finally, the grid smoothing approach will be combined with the graph-based mesh smoothing method presented in [8] to detect edges in noisy complex grayscale images.

References

1. Demaret, L., Dyn, N., Iske, A.: Image compression by linear splines over adaptive traingulations. IEEE Trans. on Signal Processing 86(7), 1604–1616 (2006)
2. Han, S.-R., Yamasaki, T., Aizawa, K.: Time-varying mesh compression using an extended block matching algorithm. IEEE Trans. on Circuits and Systems for Video Technology 17(11), 1506–1518 (2007)
3. Bu, S., Shiina, T., Yamakawa, M., Takizawa, H.: Adaptive dynamic grid interpolation: A robust, high-performance displacement smoothing filter for myocardial strain imaging. In: Ultrasonics Symposium, IUS 2008, vol. 2(5), pp. 753–756. IEEE, Los Alamitos (2008)
4. Prassl, A.J., Kickinger, F., Ahammer, H., Grau, V., Schneider, J.E., Hofer, E., Vigmond, E.J., Trayanova, N.A., Plank, G.: Automatically generated, anatomically accurate meshes for cardiac electrophysiology problems. IEEE Trans. on Biomedical Engineering 56(5), 1318–1329 (2009)
5. Yang, Y., Wernick, M.N., Brankov, J.G.: A fast approach for accurate content-adaptative mesh generation. IEEE Trans. on Image Processing 12(8), 866–880 (2003)
6. Ramponi, G., Carrato, S.: An adaptive sampling algorithm and its application to image coding. Image Vis. Comput. 19(7), 451–460 (2001)
7. Sarkis, M., Dieplod, K.: Content adaptive mesh representation of images using binary space partitions. IEEE Trans. on Image Processing
8. Hamam, Y., Couprie, M.: An Optimisation-Based Approach to Mesh Smoothing: Reformulation and Extension. In: Torsello, A., Escolano, F., Brun, L. (eds.) GbRPR 2009. LNCS, vol. 5534, pp. 31–41. Springer, Heidelberg (2009)