Image Flows and One-Liner Graphical Image Representation

Vadim Makhervaks Gill Barequet Alfred Bruckstein

Faculty of Computer Science, The Technion—Israel Institute of Technology
Haifa 32000, Israel. vadik@il.ibm.com , {barequet,freddy}@cs.technion.ac.il

Abstract

In this paper we introduce a novel graphical image representation comprising a single curve—the one-liner. The first step involves the detection and linking of image edges. We use a new technique, so-called "edge exploration," to simultaneously perform both tasks. This process is based on "image flows." It uses a gradient vector field and a new operator to explore image edges. Estimating the derivatives of the image is performed by using local Taylor expansions in conjunction with a weighted least-squares estimation method. This process finds all the possible image edges without any pruning, and collects information that allows us to prioritize the found edges. This enables us to select the most important edges, that form a "skeleton" of the sought representation. The next step connects the selected edges into one continuous curve—the one-liner. It orders the selected edges and finds curves connecting between them. We solve these two problems separately. Since the abstract graph setting of the first problem is NP-complete, we reduce it to a variant of TSP and compute an approximate solution to it. We solve the second problem by using Dijkstra's shortest-path algorithm. We have a full software implementation for the entire one-liner determination process.

1. Introduction

A one-liner representation of an image is a continuous curve passing through the important image edges. This curve is not necessarily built only from the image edges. Building a one-liner representation of a given image is a complex process, composed of multiple phases and affected by multiple factors. Picasso and Calder are well-known artists that have produced beautiful one-liner images. While a computerized system cannot compete with artists' rendition, we feel that such work proves that "good" image representations with one-liners are indeed possible.

The first step consists of detecting and linking of image edges. The quality of this step is crucial for the success of the creation of the one-liner. Conventional edge-detection schemes include three operations: smoothing, edge detection, and edge labeling. Image derivative estimation is necessary for detecting local grey-level changes, smoothing is performed for reducing the noise, and labeling is necessary in localizing the edges and suppressing the false edges. Sobel [7] and Prewitt [11] provide the best-known operators for edge detection. They focus on detecting pixels with high gradient magnitude by convoluting the original image with a filter. These operators are very sensitive to noise. To overcome this problem, smoothing operators preceding edge detection were proposed by Marr and Hildreth [8] and by Torre and Poggio [12]. Marr and Hildreth detect edges by using the zero-crossing of the Laplacian of a Gaussian convolved with the image. Together with the positive effect of noise reduction, the smoothing operator also has the negative effect of information loss. The edge detectors of Sobel and Prewitt (mentioned above) are based on thresholding the gradient modulus. This results in the need to "thin" the found edges. Canny [1] proposed an algorithm which labels local maxima along the direction of the gradient vector.

Eua-Anant and Udpa [4] proposed a novel approach based on a vector field image model. They defined a new edge operator and developed a boundary-extraction algorithm based on particle motion in a force field. Our approach is similar to that of Eua-Anant and Udpa. We refer to the process of edge-detection and linking as edge exploration. Conventional edge detectors concentrate on looking for the points with locally-maximal gradient magnitude. Our approach extends this idea to exploring an edge once we have found a seed position of it. An association of the gradient vector field of the image with a hill gives us a good intuition of this approach. The height of every point of the hill is defined as the gradient magnitude in the corresponding image point. Then, an edge in the image corresponds to a ridge of this hill. The idea is to take advantage of the gradient orientation and to use both altitude and orientation to "climb" on the ridge of the hill and then to stay there. Thus, the edge-exploration process is combined of selecting the starting point, moving toward the ridge of the hill, reaching it, and then moving in the direction that keeps the gradient magnitude locally maximal. The proposed process finds all the possible image edges without any pruning. A complex image may have many explored candidate edges, with different levels of importance. Selecting a high quality subset of representative edges is crucial, so we prioritize the edges according to image-driven information collected during the edge exploration. The selected set of image edges does not necessarily form a connected curve. We present an elegant way to resolve the problems of ordering the selected edges and connecting them into a one-liner by using image-driven curves. This problem is NP-complete in its abstract form. We reduce the problem of ordering the edges to an instance of the classic Traveling Salesman Problem and use a freely-available TSP solver to solve it. To complete a full one-liner representation of the image we connect the edges according to the TSP solution. We connect two endpoints of consecutive edges by using the well-known Dijkstra's shortest-path algorithm to find a path going through points with maximal gradient magnitude.

1051-4651/02 $17.00 © 2002 IEEE

2. Edge Exploration

The edge-exploration step detects the image edges without any additional processing, such as thinning, skeletonization, etc. The algorithm is based on image flows, using an edge-exploration operator in the gradient vector field. In a complex image almost every pixel has a non-zero gradient magnitude and there are many local maxima. Setting a threshold on the minimum accepted gradient magnitude may cause a significant loss of information in a very early stage of the processing. Therefore we use another approach. We start with finding all the edges regardless of their contribution to the image representation. Then we prioritize them using information collected during the edge exploration. Finally we select the most important edges. A typical image consists of a large number of edges (up to hundreds of thousands). Some of the edges represent a crucial part of the image and contribute to its representation. Other edges are the result of noise or belong to the background. The classification of image edges is difficult, an the quality of the image representation greatly depends on the success of the selection process.

2.1. Image Flows

We compute the gradient vector field of the image: magnitude and orientation for every pixel. We use the Taylor expansion for estimating the gradient vector and its low-order derivatives. This provides a system of linear equations with the image derivatives and estimation errors as unknowns. To minimize the error we use a weighted least-square method. The gradient vector field in each point consists of a vector orthogonal to the image gradient and having the gradient's magnitude. This vector points in a direction that preserves image intensity. The vector field defines "flows" in the image, similarly to flows in fluids. The local force yielding this flow will move an imaginary particle in the direction of the field, following the same image intensity level regardless of the gradient magnitude. This will not work in complex images, where the intensity changes along image edges. Thus we introduce an additional force that drives the flow in the direction of increasing gradient magnitude. Thus we apply two forces: one leading an object on an equal intensity level of the image, and the other pushing toward the higher gradient magnitudes. This strategy allows us not only to explore the edge starting from an edge point, but also to discover the nearest edge from any point in the field.

2.2. Edge-Exploration Operator

The edge-exploration operator provides us with the location of the next edge pixel, when it is applied to the vector of the gradient vector field. It works on a subpixel level to improve the quality of the resulting edges. Denote an input image by $\mathbf{I}(x, y)$ and the corresponding gradient vector field by $\nabla\mathbf{I}(x, y) = \left[p(x, y), q(x, y)\right]$, where p and q are the partial derivatives of the image in the pixel (x, y). The operator is:

$$\frac{\partial P}{\partial t} = \alpha \frac{\left(\nabla \mathbf{I}(x, y)\right)_{\perp}}{\|\nabla \mathbf{I}(x, y)\|} + (1 - \alpha) \frac{\nabla \|\nabla \mathbf{I}(x, y)\|}{\left\|\nabla \|\nabla \mathbf{I}(x, y)\|\right\|} \quad (1)$$

where $0 \leq \alpha \leq 1$. The left-hand side of Eq. (1) represents a shift in both coordinates, required to move to the next pixel of the edge. The right-hand side contains two terms. The first term represents a motion in the direction that preserves image intensity. This is achieved by pointing orthogonally to the gradient direction. The second term represents a motion in the direction of maximum change of the image gradient magnitude. The exploration algorithm does not assume that it starts from a pixel of an image edge. When approaching an edge, the second term points to the "fastest" way to reach the edge and has the major influence on the selection of the next subpixel. After the edge was reached, the first term takes the leading role for following the edge, whereas the second term slightly fixes the direction toward a growing gradient magnitude.

The parameter α controls the proportion between the two components of the operator. A value of $\alpha = 0.1$ suits all grey-scale images with which we have experimented. For some b/w images this parameter can be reduced even further to $\alpha = 0.05$. We explain this by the fact that b/w images have a very sharp change of the gradient magnitude near the edges, letting us reduce the influence of the second component. Notice that both components of the operator are normalized, allowing to control the displacement between two consecutive points and keep it in a subpixel resolution. A large value of the gradient magnitude or a fast change in the gradient magnitude may lead to a step of multiple pixels. This may not only hamper the beauty of the explored edge, but may also cause losing it. The fine tuning of the granularity in this step can be achieved by multiplying Eq. (1) by a scaling factor δ. We used $\delta = 0.25$ in order to enforce a quarter-pixel granularity. The real coordinate space is used to minimize the estimated errors. Due to the nature of Eq. (1), the movement is not assumed to be discrete and it allows a smooth transition between pixels. However, the first and second derivatives of the image altitude are calculated for discrete pixels. To approximate the derivatives on a subpixel level we use a linear interpolation method that uses values at the four corners of a pixel.

2.3. The Algorithm

We use a scan-line algorithm to explore and prioritize all image edges. For each pixel the algorithm finds a candidate edge starting from it. The algorithm moves for a pixel to the next pixel along the edge by using our edge-exploration operator. The algorithm assumes that the resolution of the operator is at least a single pixel. The algorithm has three termination conditions: (1) Exploring a "white" pixel (a pixel with zero gradient magnitude); (2) Exploring a stream of pixels belonging to an already-explored edge; and (3) Attempting to move to a pixel that is located outside of the image boundary.

After the candidate edge is found, a set of heuristic algorithms attempt to improve its quality. The edge is considered to be a "good" or a "stable" edge if it has a relatively smooth distribution of the overall gradient magnitude. The result is a subdivision of a candidate edge into several parts which are considered separately in the sequel.

At the same time, the algorithm collects the information necessary for edge prioritization. This involves two parameters calculated for each edge: the number of edges converging to the edge (indicating its importance), and the average gradient magnitude along it (indicating its intensity). The edges are prioritized according to both parameters, and the user selects the most important edges.

3. Linking the Edges

Finally we connect the selected edges into a continuous curve—a one-liner. This goal can be divided into two separate tasks: (1) Order the set of edges so that their connected version is "nice" and intuitive; and (2) Compute a set of curves that connect between the edges. We accomplish each task separately and combine the solutions together.

3.1. Ordering the Edges

We seek an "optimal" sequence of the edges in which they will be connected. Since formally the one-liner is not required to not intersect itself, the one-liner problem can be reduced to the problem of connecting a given set of edges into one curve with an upper bound on the length of the connectors. We prove that this abstract setting of the problem is NP-complete by a reduction from the Hamiltonian Path/Circle problem. (The proof is omitted here.) Thus it is unlikely to find efficiently the optimal order to connect the edges. Instead we convert it to an instance of the Traveling Salesman Problem and use a publically-available TSP solver to resolve it approximately.

The one-liner problem slightly differs from the classic TSP: Our problem not only contains $2n$ vertices and all possible weighted edges connecting between them, but it is also constrained by a set of n edges that must be part of the solution. To satisfy this we add a new point in the middle of every such edge, and assign distances (edges weights) between the points so as to force the TSP solution to contain all the preselected edges. Although the TSP problem is NP-complete [5, 10], many special cases of it, in particular, our one-liner problem, can be approximated efficiently. We use the publically-available LK program (by Neto [9]), which implements the Lin-Kernighan heuristic. It mostly follows the design outlined by Johnson and McGeoch [6].

3.2. Connecting the Edges

For obtaining a one-liner similar to the image, we use image-driven curves that connect smoothly between the edges. To connect between two endpoints of different edges, we seek a curve with maximal average gradient magnitude, which does not have loops and is located in the neighborhood of the connected endpoints. We use the graph shortest-path algorithm of Dijkstra [3]. Given the two connector's endpoints, we build a graph that has a vertex for each pixel in the neighborhood and an edge connecting the pixel with its neighboring pixels. The weight of the edge is the reciprocal of the gradient magnitude of the neighboring pixel. We use an implementation of the algorithm found in SPLIB (Cherkassky, Goldberg, and Radzik [2]).

4. Experimental Results

We have implemented the entire algorithm, except using two external packages: the LK program for solving the Euclidean TSP problem, and an implementation of Dijkstra's shortest-path algorithm. We experimented extensively with the software. Here is a sample of figures that demonstrate the various stages of the algorithm. Figure 1 demonstrates the operation of the edge-exploration step on some simple images. It requires very few edges to obtain nice representations. Figure 2 shows the application of the same algorithm on a few more complex images. Figures 3 and 4 show various stages of the algorithm. These sequences demonstrate the effect of varying the prioritization parameters and show one-liner representations of each image. Finally, Figure 5 shows one-liner representations of various grey-scale and b/w images. We ran our experiments on a P-III 600Mhz personal computer with 256MB of SRAM. Table 1 shows some running times, measured for three grey-scale images differing in their dimensions and complexities. For all the images we used the parameter $\alpha = 0.1$ for the edge-exploration step, whose statistics are shown in the the first section of the table. The second section shows statistics for the TSP and Dijkstra's implementations.

Edge exploration			
Image	**Lamp**	**Paolina**	**Felix**
Dimensions	256256	512480	240,346
Derivatives estimation (MS)	841	3135	1072
Path exploration	7190	21531	9954
Number of explored paths	7703	31865	10515
Connecting selected edges			
Selected edges	122	574	182
TSP vertices	183	861	273
TSP execution (MS)	140	1422	250
Graph vertices (ave.)	2301	2248	2428
Dijkstra's algorithm (MS)	531	2213	801

Table 1. Performance of the algorithm

References

[1] J.F. Canny, A computational approach to edge detection, *IEEE Trans. on Pattern Anal. and Machine Intel.*, 8(1986), 679–698.

[2] B.V. Cherkassky, A.V. Goldberg, and T. Radzik, *Shortest paths algorithms: Theory and experimental evaluation*, Math. Programming, 73(1996), 129–174.

[3] E.W. Dijkstra, A note on two problems in connection with graphs, *Numerical Mathematics*, 1(1959), 269–271.

[4] N. Eua-Anant and L. Udpa, Boundary detection using simulation of particle motion in a vector image field, *IEEE Trans. on Image Processing*, 8(1999), 1560–1572.

[5] M.R. Garey, R.L. Graham, and D.S. Johnson, Some NP-complete geometric problems, *Proc. ACM Symp. on Theory of Computing*, 1976, 10–22.

[6] D.S. Johnson and L.A. McGeoch, The traveling salesman problem: A case study, *Local Search in Combinatorial Optimization*, John Wiley & Sons, 1997.

[7] E.P. Lyvers and O.R. Mitchell, Precision edge contrast and orientation estimation, *IEEE Trans. on Pattern Anal. and Machine Intel.*, 10(1988), 927–937.

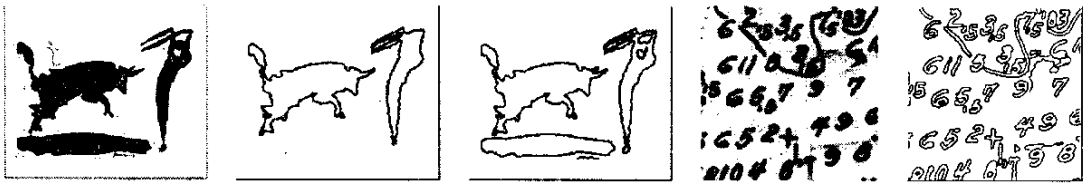
[8] D. Marr and E.C. Hildreth, Theory of edge detection, *Proc. of the Royal Society of London*, 207(1980), 187–217.

[9] D. Neto, Efficient cluster compensation for Lin-Kernighan heuristics, Ph.D. thesis, Dept. of CS, Univ. of Toronto, 1999.

[10] C.H. Papadimitriou, Euclidean TSP is NP-complete, *Theoretical Computer Science*, 4(1977), 237–244.

[11] J.M.S. Prewitt, Object Enhancement and Extraction, in *Picture Processing and Psychopictorics* (B.S. Lipkin and A. Rosenfeld., Eds.), Academic Press, New York, 1970.

[12] V. Torre and T.A. Poggio, On edge detection, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(1980), 147–163.



(a) Image (b) Two edges (c) Eight edges (d) Image (f) Edges

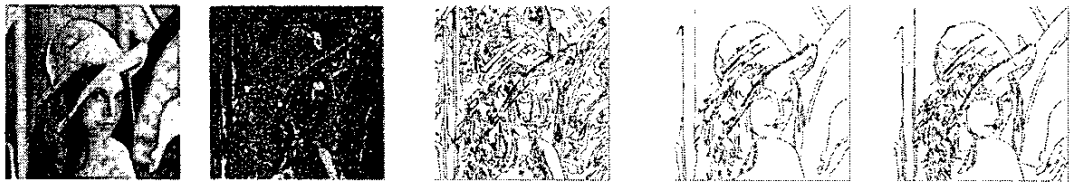
Figure 1. Main edges in simple images

(a) Paolina (b) Venere Vincitirce (c) Amore e Psiche (d) Bullfight (e) Lena

Figure 2. Main edges in complex images

(a) Image (b) All edges (c) Unlimited gradient (d) 23 selected edges (f) One-liner

Figure 3. Paolina: one-liner image representation



(a) Image (b) All edges (c) Unlimited gradient (d) 23 selected edges (f) One-liner

Figure 4. Lena: one-liner image representation

(a) Lamp (b) Felix (c) Venere Italica (d) Ebe (f) Horse

Figure 5. Various one-liners