Pulling, Pushing and Grouping for Image Segmentation

Guoping Qiu[1] and Kin-Man Lam[2]

[1] School of Computer Science, The University of Nottingham
qiu@cs.nott.ac.uk
[2] Dept. of Electronic and Information Eng. The Hong Kong Polytechnic University
enkmlam@polyu.edu.hk

Abstract. This paper presents a novel computational visual grouping method, termed pulling, pushing and grouping, or PPG for short. Visual grouping is formulated as a functional optimisation process. Our computational function has three terms, the first pulls similar visual cues together, the second pushes different visual cues apart, and the third groups spatially adjacent visual cues without regarding their visual properties. An efficient numerical algorithm based on the Hopfield neural model is developed for solving the optimisation process. Experimental results on various intensity, colour and texture images demonstrate the effectiveness of the new method.

1. Introduction

Visual grouping is thought to be a crucial process in both human and computer vision systems. The volume of computer vision/image processing literature relating to the topic, often under different names, such as image segmentation, figure ground separation, and perceptual organization, is huge, and recent examples include [1-4]. Although a large amount of effort has been put into the study of the topic, many computational issues remain very difficult. In this paper, we present a new computational approach to visual grouping.

We consider visual grouping as a combined process of pulling, pushing and grouping, or PPG for short. Two visual cues that belong to the same visual group will pull each other together to become members of the same group. Conversely, two visual cues that belong to different visual groups will push each other apart to become members of the different groups. Simultaneously, local neighboring visual cues, regardless of their visual properties, will group together to become members of the same group. Similar to many other approaches in the literature, the PPG process is formulated as a functional optimization process. A PPG computational function, consisting of three terms, namely, the pulling, the pushing and the grouping terms, is constructed in such a way that the minimum of the function will correspond to a good solution to the visual grouping problem. An efficient numerical solution based on a neural computing model is developed to solve the optimization problem.

The organization of the paper is as follows. Section 2 presents the new PPG visual grouping method. Section 3 presents experimental results of the application of PPG to the segmentation of intensity, colour, and texture images. Section 4 very briefly discusses related prior work in the literature. Section 5 concludes the presentation.

2. Perceptual Organization based on the Pulling, Pushing and Grouping (PPG) Principle

Perceptual grouping is the process of organising visual elements into groups that manifest certain regularities. The Gestalt psychologists have suggested that visual grouping should follow some basic laws, including proximity, similarity, good continuation, amongst many others. To formulate computational models for the perceptual grouping process, a common practice is to optimise some cost functions which measure the quality of the grouping. Although the Gestalt grouping laws are intuitive, casting them into a computational function proves to be difficult. A common approach is to capture local interaction of visual cues in a global optimisation framework. Once a cost function is formulated, computing an optimal solution is again very challenging. In this section, we propose a pushing, pulling and grouping principle to formulate a perceptual grouping computational energy and develop a neural network based optimisation algorithm.

2.1 The PPG Computational Energy Function

We first consider the case where the image is to be partitioned into two groups (figure and ground). Let V_i, $i = 1, 2, ...N$, be the binary variables, $V_i = 0$ represents pixel i belong to group 0 and $V_i = 1$ represents pixel i belongs to group 1. If two pixels i and j belongs to the same group, we define a pulling energy, $A(i, j)$, between the pixels. If two pixels i and j belong to different groups, we define a pushing energy, $R(i, j)$, between the pixels. Then the PPG computational function is defined as

$$\begin{aligned} E = \lambda_1 \Bigg(& \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} V_i V_j A(i,j) + \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} (1-V_i)(1-V_j) A(i,j) + \\ & \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} V_i (1-V_j) R(i,j) + \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} (1-V_i) V_j R(i,j) \Bigg) + \\ & \lambda_2 \Bigg(-\sum_{i=1}^{N} \sum_{j\in n_i, j\neq i} (2V_i - 1)(2V_j - 1) \Bigg) \end{aligned} \qquad (1)$$

where n_i is a local neighbourhood of i. λ_1, λ_2 are non-negative weighting constants. The values of V_i, $i = 1, 2, ...N$, that correspond to the minimum of E represent an effective partition of the image into two perceptual groups.

At a first glance, it may seem strange to write (1) in such a redundant way. As will become clear immediately, the current form of (1) helps an intuitive explanation of the meanings of each term. The first term measures the cost of putting two pixels into the same group (pulling). The second term measures the cost of putting two pixels into different groups (pushing). The third term measures the cost of putting a pixel and its neighbours into the same group regardless of the values of the original pixels (grouping). The relative importance of these terms is determined by their respective weighting constants. However, the importance of the first and second terms is the same, so the same weighting constant is used.

If two pixels have similar visual properties, such as similar intensity, similar colour, similar texture or similar edgels, it is likely that they belong to the same group. If such pixels are close to each other in the spatial domain, this likelihood (that they belong to the same group) will increase. On the other hand, if two pixels have dissimilar visual properties, different intensities, different colours, different textures or different edgels, it is likely that they will belong to different groups. If they are far apart in the spatial domain, this likelihood (that they belong to different groups) will increase. Based on these reasoning and the form of (1), we can now define the form of the pulling energy function $A(i, j)$ and the pushing energy function $R(i, j)$.

When two pixels have very similar visual cues or are very close to each other spatially, $A(i, j)$ should be small and $R(i, j)$ should be large such that the first term in (1) is favoured, effectively forcing a solution that V_i and V_j will have the same value, i.e., pulling pixels i and j into the same group. Conversely, when two pixels have very different visual cues and far apart spatially, $A(i, j)$ should be large and $R(i, j)$ should be small such that the second term in (1) is favoured, effectively forcing a solution that V_i and V_j will have different values, i.e., pushing pixels i and j to different groups. Let $F(i)$, i =1, ...N, be the visual cue of pixel i, one possible definition of the pulling and pushing energy functions is (2) (other similar definitions are possible)

$$A_S(i,j)=\begin{cases}\frac{|i-j|}{\sigma_S} & \text{if} |i-j|\le\sigma_S\\ 0 & \text{Otherwise}\end{cases}\quad R_S(i,j)=\begin{cases}\frac{\sigma_S-|i-j|}{\sigma_S} & \text{if} |i-j|\le\sigma_S\\ 0 & \text{Otherwise}\end{cases} \tag{2}$$

$$A_F(i,j)=\exp\left(-\frac{\|F(i)-F(j)\|}{\sigma_F}\right)\quad R_F(i,j)=1-\exp\left(-\frac{\|F(i)-F(j)\|}{\sigma_F}\right)$$

$$A(i,j)=A_S(i,j)A_F(i,j)\qquad R(i,j)=R_S(i,j)R_F(i,j)$$

Fig.2 shows the shape of the visual components and the spatial components of the pulling and pushing energy. The shape is controlled by the free parameters σ_F and σ_S, which determine when the pulling power outweighs the pushing power for the pair of pixels, or vice versus. The pulling and pushing interactions amongst pixels are confined to local regions. The rationale is that when two pixels are too far apart, they will not affect one another.

2.2 A Numerical Algorithm

The Hopfield neural network models [5 -7] have been successfully used in solving optimization problems. Because a neural network is a naturally parallel structure, it has the potential for solving many complex optimization problems much more rapidly than other sophisticated and computationally intensive optimization techniques. Neurons can be highly and selectively interconnected so as to give rise to collective computational properties and create networks with good computational efficiency. Collective computational properties emerge from the existence of an energy function of the states of the neuron outputs, namely the computational energy. The computational energy function of the Hopfield network has the following form

$$E_H=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}V_iT_{ij}V_j-\sum_{i=1}^{N}I_iV_i \tag{3}$$

where V_i, i = 1, 2, ...N, is the outputs of the network, T_{ij} is the connection strength between neuron i and neuron j, I_i, i =1, 2, ...N, is the external input to neuron i. Hopfield has shown that if the connection matrix is symmetrical with zero diagonal elements (i.e., T_{ij} = T_{ji}, and T_{ii} = 0), the function of the neuron output in (4) is decreased by any state changes produced by the following asynchronous rule:

$$V_i = \begin{cases} 1 & \text{if } H_i \geq 0 \\ 0 & \text{if } H_i < 0 \end{cases} \quad \text{where } H_i = I_i + \sum_{j=1, j\neq i}^{N} T_{ij} V_j \tag{4}$$

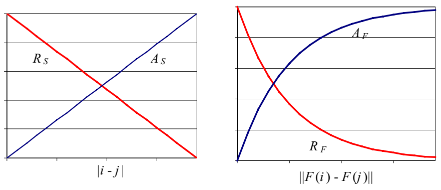

Fig. 1. The spatial and visual components of the pulling and pushing energy function.

For a given initial state, the network updates its output V_i according to equation (4) asynchronously, that is one neuron output at a time. When the network reaches its stable state in which further iterations do not change the output of V_i, the network outputs correspond to a local minimum of the computational energy function of the form (3). It follows that the interconnection strength between neurons, and the external inputs to the neurons, have the following relationship with the computational energy function:

$$T_{ij} = \frac{\partial^2 E_H}{\partial V_i \partial V_j} \qquad I_i = \left.\frac{\partial E_H}{\partial V_i}\right|_{V_j = 0, \forall j} \tag{5}$$

Clearly, to use the collective computational properties of the Hopfield network to solve a problem, a cost function (computational energy) has to be defined in such a way that the lowest value represents an effective solution to the problem. For our current application, we can use the Hopfield neural model to optimize the PPG energy (1).

To find the optimum solution to (1), we find it numerically convenient to separate the visual feature dependent part and the visual feature independent part, and minimize each part in turn. We re-write (1) as

$$\begin{aligned} E_{PP} = \lambda_1 \Bigg(& \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} V_i V_j A(i,j) + \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} (1 - V_i)(1 - V_j) A(i,j) + \\ & \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} V_i (1 - V_j) R(i,j) + \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} (1 - V_i) V_j R(i,j) \Bigg) \end{aligned} \tag{6}$$

$$E_G = \lambda_2 \left(-\sum_{i=1}^{N} \sum_{j \in n_i, j \neq i} (2V_i - 1)(2V_j - 1) \right)$$

We can construct one Hopfield network to optimize E_{PP} and another Hopfield network to optimize E_G. For the E_{PP} network, we have the connection strengths and external inputs as

$$\begin{aligned} T_{ij}^{PP} &= -4\lambda_1 \left(A(i,j) - R(i,j) \right), j \neq i \\ I_i^{PP} &= 2\lambda_1 \sum_{j=1, j\neq i}^{N} \left(A(i,j) - R(i,j) \right) \end{aligned} \tag{7}$$

and for the E_G network, we have the connection strengths and external inputs as

$$T_{ij}^{G} = 4\lambda_2,\ j \neq i, j \in n_i \qquad I_i^{G} = -2(|n_i| - 1)\lambda_2 \tag{8}$$

Then the PPG visual grouping algorithm can be summarized:

```
The PPG Visual Grouping Algorithm

Step 1, Initialize $V_i$, $i$ = 1, 2, ...$N$, to random values in the range (0, 1)
Step 2, For all $i$ = 1, 2, ...$N$, update $V_i$, one at a time according to
    $V_i = \begin{cases} 1 & \text{if } H_i^{PP} \geq 0 \\ 0 & \text{if } H_i^{PP} < 0 \end{cases}$   where $H_i^{PP} = I_i^{PP} + \sum_{j=1, j \neq i}^{N} T_{ij}^{PP} V_j$
Step 3, For all $i$ = 1, 2, ...$N$, update $V_i$, one at a time according to
    $V_i = \begin{cases} 1 & \text{if } H_i^{G} \geq 0 \\ 0 & \text{if } H_i^{G} < 0 \end{cases}$   where $H_i^{G} = I_i^{G} + \sum_{j=1, j \neq i}^{N} T_{ij}^{G} V_j$
Step 4, go to Step 2, until converge.
```

From (8), it is not difficult to see that Step 3 performs majority voting. A pixel will be assigned to the membership of that of the majority of its neighbours. It is also easy to see that the weighting parameter λ_2 for the grouping term becomes irrelevant in the decision making. The complexity of this Step is $O(mNn_i)$ binary operations, where N is the number of pixels, n_i the size of the local window and m the number of iterations. To perform Step 2, the connections and external inputs in (7) need only to be done once with a complexity of $O(N\sigma_S)$, where N is the number of pixels and σ_S is size of the local region within which the pulling and pushing are active. The complexity of Step 2 after computing the T_{ij} and I_i is $O(mN\sigma_S)$ addition operations. Again, the weighting parameter λ_1 becomes irrelevant in the decision making. We have observed that $n_i = 3 \sim 5$, $\sigma_S = 7 \sim 15$, $m = 5 \sim 10$ worked well in our experiments. An important parameter in the algorithm is σ_F in (2). Typically, this should be set based on the statistics of the visual features used in the interaction. We found setting it according to the variance of the feature vectors worked well.

If it is required that the image be partitioned into more than two groups, the algorithm can be applied recursively to the resultant groups, until the desired number of groupings are generated.

3. Application to Scene Segmentation and Results

We have implemented the PPG algorithm based on intensity, colour and texture features. For the intensity images, the visual feature $F(i)$ in (2) takes the intensity of the pixel. For the colour image, we used the HSV colour space and each pixel was represented by a 3-d vector formulated as $F(i) = (v(i)*s(i)*\sin(h(i)), v(i)*s(i)*\cos(h(i)), v(i))$, where $v(i)$, $s(i)$, $h(i)$, are the Value, Saturation and Hue of pixel i. For texture images, we use Law's filtering approach to extract the texture features. The filter masks were t_1 = [1, 4, 6, 4, 1], t_2 = [-1, -2, 0, 4, 1], t_3 = [-1, 0, 2, 0, -1], t_4 = [-1, 2, 0, -2, 1], and t_5 = [1, -4, 6, -4, 1]. The resulting 25 filter outputs for each pixel position were used to form a 25-d feature vector for each pixel [4]. In each case, the variance of the feature, var_F, was calculated as

$$\text{var}_F = \frac{1}{N}\sum_{i=1}^{N} \| F(i) - M_F \|, \qquad M_F = \frac{1}{N}\sum_{i=1}^{N} F(i) \tag{9}$$

The pulling and pushing energy parameter in (2) is set as $\sigma_F = \alpha var_F$, where α is a constant, which is tuned manually. We found setting $0 < \alpha \leq 1$ worked well.

Fig. 2 shows results of grouping grey scale intensity images. It is seen that the results are very good. The two salient groups of visual pattern have been successfully separated.

Fig. 3 shows results of grouping images using colour features. It is again seen that the two salient groups of visual pattern have been successfully separated.

Fig. 4 shows results of grouping images using texture features. It is again seen that the two salient groups of visual pattern have been successfully separated. Notice that in the grouping process, only texture features extracted from the achromatic channel were used and colour information was not used. The images are shown in colour for visually more appealing presentation.

4. Related Work

The volume of literature in the area of image segmentation is huge, so we do not intend to list all possibly related prior work but merely to point out some recent work we believe are most related to our current work. We believe the present work is most closely related to recent work on graph based frameworks for image segmentation [1-3]. Whilst [1,3] proposed eigenvector or graph spectral theory based solutions, [2] used methods based on maximum likelihood graph clustering. Our method also can be regarded as representing the segmentation or grouping using a graph. However, the weights of the graph edges in our current work are defined differently, which not only measure the similarity between the nodes but also explicitly measure the dissimilarity of the nodes, i.e., the nodes not only pulling each other together, they also pushing one another apart. We have formulated the energy function in such a way that it can be numerically solved by a Hopfield neural network model. Our solution can be considered very efficient numerically because only addition and thresholding operations are involved. We are investigating whether there are deeper relations between our current algorithm and those in the literature.

5. Concluding Remarks

We have presented a novel algorithm for visual grouping. Our pulling, pushing and grouping principle was proposed based on the rationale that, (i) if two pixels have similar visual properties, it is likely that they belong to the same group and this likelihood increases if they are close to each other spatially as well, on the other hand, (ii) if two pixels have dissimilar visual properties, it is likely that they will belong to different groups and this likelihood will increase if they are far away from each other spatially, and (ii) spatially close pixels are more likely belong to the same group regardless of their photometric values. We then cast this principle in a computational energy function and developed a neural network based solution. We have presented experimental results to show that the new algorithms work very effectively and give very encouraging results.

As is the same with many other algorithms, there are several free parameters in our algorithm that needed to be decided by empirical means. We have provided some guidelines to select them and fortunately, we found that it was not difficult to find their values that would work well on a variety of data. We also found the algorithm converged very fast. However, one possible problem is that the algorithm may converge to a local minimum, and in some cases, the algorithm may perform unbalanced grouping resulting in one group being too large the other too small. Work is ongoing to test the algorithm using a variety of data to gain deeper understanding of its behaviour.

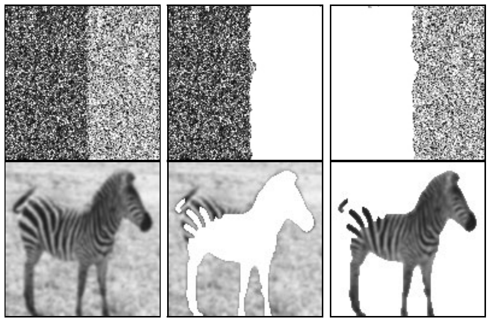

Fig. 2. Results of PPG on intensity images. Images in the 1st column are the original images. The two visual groups are shown in the 2nd and 3rd columns. All results are for parameters $\sigma_S = 5$, $n_i = 2$, $\alpha = 1$ and after 5 iterations.

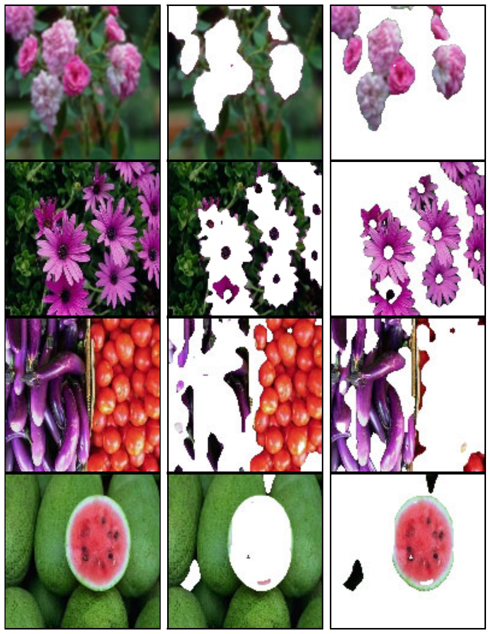

Fig. 3. Results of PPG on colour images. Images in the 1st column are the original images. The two visual groups are shown in the 2nd and 3rd columns. All results are for parameters $\sigma_S = 11$, $n_i = 2$, $\alpha = 0.5$ and after 15 iterations.

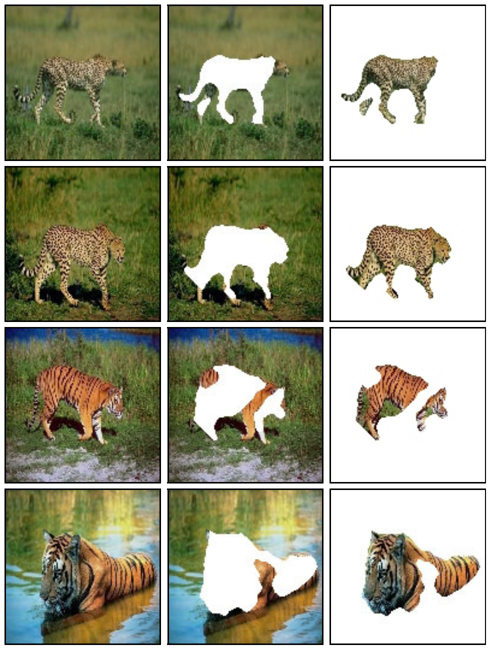

Fig. 4. Results of PPG based on texture features (notice that only achromatic signals were used in the grouping). Images in the 1st column are the original images. The two visual groups are shown in the 2nd and 3rd columns. All results are for parameters $\sigma_S = 11$, $n_i = 2$, $\alpha = 0.5$ and after 15 iterations.

6. References

[1] J. Shi and J. Malik, "Normalized cuts and image segmentation", IEEE Trans PAMI, vol. 22, no. 8, pp. 888 - 905, 2000

[2] A. Amir and M. Lindenbaum, "A generic grouping algorithm and its quantitative analysis", IEEE Trans PAMI, vol. 20, no. 2, pp. 168-185,1998

[3] Y. Weiss, "Segmentation using eigenvectors: a unifying view", ICCV 1999

[4] T. Randen and J. Husøy, " Filtering for Texture Classification: A Comparative Study", IEEE Trans PAMI, Vol 21, No. 4, pp. 291 - 310, 1999

[5] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proc. Natl. Acad. Sci., pp. 2554 - 2558, April 1982,

[6] J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", Proc. Natl. Acad. Sci., pp. 3088 - 3092, April 1984,

[7] J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems", Biol. Cyern., pp. 141 - 152, 1985