Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition Letters 24 (2003) 1051–1057

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Contraction kernels and combinatorial maps

Luc Brun [a,*], Walter Kropatsch [b,1]

[a] Laboratoire d'Études et de Recherche en Informatique, I.U.T. de Reims, 51.059 Reims, France
[b] Institute for Computer-aided Automation, Pattern Recognition and Image Processing Group, Vienna University of Technology, Vienna, Austria

Abstract

Graph pyramids are made of a stack of successively reduced graphs embedded in the plane. Such pyramids overcome the main limitations of their regular ancestors. The graphs used in the pyramid may be region adjacency graphs, dual graphs or combinatorial maps. Compared to usual graph data structures, combinatorial maps offer an explicit encoding of the orientation of edges around vertices. Each combinatorial map in the pyramid is generated from the one below by a set of edges to be contracted. This contraction process is controlled by kernels that can be combined in many ways. This paper shows that kernels producing a slow reduction rate can be combined to speed up reduction. Conversely, kernels decompose into smaller kernels that generate a more gradual reduction. We also propose one sequential and one parallel algorithm to compute the contracted combinatorial maps.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Segmentation; Combinatorial maps; Hierarchical representation; Combinatorial pyramids

1. Introduction

Using graphs many vision problems may be formulated in an abstract setting with solid theoretical foundations from graph theory. Additionally, graphs allow to abstract the exact shape and positioning of the objects under study and thus improve the potential applications and the efficiency of many graph based vision algorithms. The use of graphs within the computer vision framework is not new. However, there is a growing interest toward an explicit formulation of vision problems as graph problems. The recent trends in this field concern (Dickinson et al., 2001): the graph partitioning, graph indexing, graph matching and clustering and the graph generalization problems. The potential applications of such problems are respectively the segmentation, the image data base retrieval, the object recognition and classification and the object recognition and modeling.

However, graphs used in vision may be quite big and many graph algorithms have a high computational cost. For example, a simple graph encoding a 512×512 regular grid is defined by $\mathcal{O}(512^2)$ vertices and edges. In the same way the brute force approach of graph matching requires a

* Corresponding author.
E-mail addresses: luc.brun@univ-reims.fr (L. Brun), krw@prip.tuwien.ac.at (W. Kropatsch).
[1] This work was supported by the Austrian Science Foundation under P14445-MAT.

0167-8655/03/$ - see front matter © 2002 Elsevier Science B.V. All rights reserved.
PII: S0167-8655(02)00251-9

computational cost of $\mathcal{O}(n!)$, where n is the number of vertices.

An irregular pyramid (Meer, 2001) is defined as a stack of successively reduced graphs. Each vertex in the pyramid is associated to a connected set of vertices in the level below named its reduction window. Each vertex is the parent of the vertices in its reduction window. This parent–child relationship may be iterated down to the base level and the set of child of one vertex in the base level is named its receptive field.

Given one level of the pyramid, the set of receptive fields defines a partition of the base level graph. A direct application of irregular pyramids concerns thus the graph partitioning. Gdalyahu et al. (2001) improve the robustness of the decimation process by using stochastic reduction rules and building M samples of the pyramid. Each edge is then labeled by the number of times it survives up to a given level r. The irregular pyramid is then used to attach a global value to each edge of the base level graph. Paillonc y et al. (1998) use two other interesting features of irregular pyramids for graph matching:

(1) Given a pyramid, a first rough solution may be determined on the top of the pyramid and then refined from level to level down to the base.
(2) The notion of hierarchy defined by irregular pyramids may be used to add further constraints on graph algorithms. For example, within the graph matching framework we can require not only a match between the two base level graphs but between the base level graphs and their reduced versions.

These two features may both improve the robustness and reduce the computational cost of graph algorithms.

Irregular pyramids were first defined by Montanvert et al. (1991). Such irregular pyramids are based on simple graphs (i.e. graphs without multiple edges nor self-loops) and the reduction operation lies on the definition of a set of surviving vertices. Such a set fulfills the two following requirements: (1) any non-surviving vertex must be adjacent to a surviving one and (2) two surviving vertices cannot be adjacent. A set of vertices fulfilling these last requirements is called a maximal independent vertex set (MIVS). Given a MIVS, each non-surviving vertex is mapped to a surviving neighbor which becomes its parent. The definition of a MIVS requires several iterations of a parallel algorithm. Jolion (2001) removes the iterations induced by the MIVS by performing asynchronous reductions: some non-surviving vertices may be mapped to surviving ones beside the fact that the whole set of surviving vertices is not yet fully computed. Within the segmentation framework, the first surviving vertices reaching the top of the pyramid should, according to Jolion, correspond to the main object of a scene.

The reduction process used to build one level from the level below influence thus both the computational cost and the properties of an irregular pyramid. The type of graph used within the pyramid is an other important parameter which determines the features which may be readily computed. For example, if the pyramid encodes a hierarchy of image partitions, multiple boundaries and inclusion relationships between regions cannot be readily encoded using a simple graph data structure. Note that such features may be required, for example by segmentation algorithms.

This last drawback may be overcome by using the dual graph pyramid introduced by Kropatsch (1995). A dual graph pyramid is defined as a stack of dual graphs $(G, \overline{G})$ successively reduced. Within such pyramids the mapping of a non-surviving vertex to a surviving one is performed by the contraction of their common edge. The contraction of a graph reduces the number of vertices while maintaining the connections to other vertices. As a consequence some redundant edges such as self-loops or double edges may occur. These redundant edges may be characterized in the dual of the contracted graph and suppressed by a removal step. The resulting graph encodes multiple boundaries between regions. Moreover, edges encoding the adjacency between one region and a surrounding one may be characterized in the dual graph.

Dual graph pyramids have been successfully applied to encode the relationships between objects embedded in the plane (see e.g. Kropatsch, 1995). However, such a graph encoding may not

be readily extended to higher dimensions. Moreover, the orientation of the plane is not explicitly encoded by dual graph pyramids. These two drawbacks maybe overcome by using the combinatorial pyramid framework. We present in Section 2 the combinatorial map model together with its main properties. In Section 3 we introduce the notion of contraction kernel. Such kernels specify the set of edges to be contracted between two levels of the pyramid. These kernels may be either combined or decomposed into smaller kernels. This last property provides an efficient control on the decimation ratio. One sequential and one parallel algorithm computing the contracted combinatorial map are provided in Section 4.

2. Combinatorial maps

A combinatorial map (Gareth and Singerman, 1978) may be understood as a planar graph encoding explicitly the orientation of edges around a given vertex. Fig. 1 demonstrates the derivation of a combinatorial map from a plane graph. First edges are split into two half edges called *darts*, each dart having its origin at the vertex it is attached to. The fact that two half-edges (darts) stem from the same edge is recorded in the reverse permutation α. A second permutation σ, called the successor permutation, defines the (local) arrangement of darts around a vertex. Each orbit of σ is associated to one vertex and encodes the sequence of darts encountered when turning counterclockwise around this vertex (e.g. the σ-orbit $(4,5,6)$ in Fig. 1).

The symbols $\alpha^*(d)$ and $\sigma^*(d)$ stand, respectively, for the α and σ-orbits of the dart d. More generally, if d is a dart and π a permutation we denote the π-orbit of d by $\pi^*(d)$.

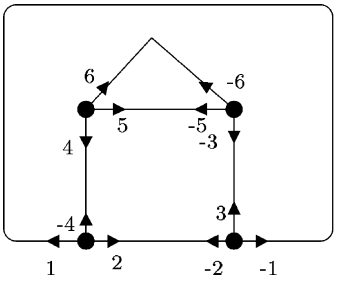
A combinatorial map G is the triplet $G = (\mathscr{D}, \sigma, \alpha)$, where $\mathscr{D}$ is the set of darts and σ, α are two permutations defined on $\mathscr{D}$ such that α is an involution: $(\forall d \in \mathscr{D}\ \alpha^2(d) = d)$.

A combinatorial map is defined up to a labeling of darts. This equivalence between two combinatorial maps may be formally defined by using the notion of isomorphism (Gareth and Singerman, 1978): given two combinatorial maps $G_1 = (\mathscr{D}_1, \sigma_1, \alpha_1)$ and $G_2 = (\mathscr{D}_2, \sigma_2, \alpha_2)$ an application ψ from $\mathscr{D}_1$ to $\mathscr{D}_2$ defines an isomorphism between G_1 and G_2 iff ψ is bijective and

$$\forall d \in \mathscr{D}_1 \quad \begin{cases} \alpha_1(d) = \psi^{-1}(\alpha_2(\psi(d))) \\ \sigma_1(d) = \psi^{-1}(\sigma_2(\psi(d))) \end{cases} \tag{1}$$

As dual graphs, combinatorial maps allow to encode multiple boundaries between regions and surrounding relationships. However, combinatorial maps provide the following additional advantages within the image analysis framework:

(1) Combinatorial maps explicitly encode the orientation of darts around one vertex. This information should be helpful to differentiate some configurations within the graph matching and clustering frameworks. Note that this information is not encoded by region adjacency graphs nor explicitly available in dual graph data structures.
(2) Given a combinatorial map $G = (\mathscr{D}, \sigma, \alpha)$, its dual $\overline{G}$ is defined on the same set of darts by the permutations $\varphi = \sigma \circ \alpha$ and α. The efficiency of this transformation avoids an explicit encoding of the dual graph. Therefore, only one data structure has to be encoded and



$$\alpha = (1,-1)(2,-2)(3,-3)(4,-4)(5,-5)(6,-6)$$

$$\sigma = (1,2,-4)(-2,-1,3)(-3,-6,-5)(5,6,4)$$

Fig. 1. One planar graph encoded by a combinational map.

maintained along the pyramid (Brun and Kropatsch, 1999b).

(3) Combinatorial maps may be defined in any dimensions (Lienhardt, 1989). The design of 3D split and merge algorithms based on combinatorial maps is an active research field (e.g. Bertrand et al., 2001).

3. Contraction kernels

Using the reduction scheme introduced by Kropatsch (1995) (see also Section 1) the reduction operation is performed in two steps: a first contraction step maps non-surviving vertices to surviving ones and a removal step suppresses redundant edges. The contraction and removal operations may be defined in order to preserve the orientation of the initial combinatorial map (Brun and Kropatsch, 1999b). In order to preserve the number of connected components of the initial combinatorial map, bridges and self-loops must be respectively excluded from removal and contraction operations (Brun and Kropatsch, 1999b). Since the contraction operation is not defined for self-loops, several contraction operations may be performed simultaneously only if we ensure that no self-loops may be contracted. This constraint may be solved by using a contraction kernel:

Definition 1 (*Contraction kernel*). Given a connected combinatorial map $G = (\mathscr{D}, \sigma, \alpha)$, the set $K \subset \mathscr{D}$ will be called a contraction kernel iff it is a forest of G not including all darts of G: $\mathscr{S}\mathscr{D} = \mathscr{D} - K \neq \emptyset$. The set $\mathscr{S}\mathscr{D}$ is called the set of surviving darts.

Since the set of darts to be contracted forms a forest of the initial combinatorial map, no self-loop may be contracted and the contraction operation is well defined. Note that the forest K is not required to be a spanning one. This last point allows to apply contractions on some parts of the initial combinatorial map leaving the other parts unchanged. More generally, the set of child of a given vertex defined in the contracted combinatorial map may vary from a single vertex to a tree with any height. Moreover, since the vertices of the graph are implicitly defined by their darts we must require that at least one edge survives.

Given an initial combinatorial map G_0 contracted by a contraction kernel K_1 into a reduced combinatorial map $G_1 = G_0/K_1$, any contraction kernel K_2 defined on G_1 is called a successor of K_1. This relation is denoted $K_1 \prec K_2$. Moreover, given two contraction kernels K_1 and K_2 both defined on G_0, we say that K_2 includes K_1 ($K_1 \subset K_2$) iff the set of darts of K_1 is included in the one of K_2. The successive applications of two successive kernels K_1 and K_2 on a combinatorial map G_0 defines a contracted combinatorial map $G_2 = (G_0/K_1)/K_2$. We have shown (Brun and Kropatsch, 1999a) that the union of the darts of K_1 and K_2 defines a new contraction kernel $K_3 = K_1 \cup K_2$ of G_0. Applied on G_0, this kernel provides the same contracted combinatorial map than the successive applications of K_1 and K_2. Conversely, given two contraction kernels K_1 and K_3 of G_0 such that $K_1 \subset K_3$, the set of darts of K_3 which are not in K_1 defines a contraction kernel K_2 of $G_1 = G_0/K_1$ such that $G_1/K_2 = G_0/K_3$ (see proof in Brun and Kropatsch, 1999a).

Therefore, the successive applications of small kernels may be replaced by the application of a bigger one. Conversely, one contraction kernel may be decomposed into smaller kernels in order to define some intermediate contracted combinatorial maps. Such results provide a better flexibility in the design of contraction kernels and allow us to adapt more efficiently the decimation rate to the data.

4. Computation of the contracted combinatorial map

Given a combinatorial map $G = (\mathscr{D}, \sigma, \alpha)$ and a contraction kernel K, the set of darts of the contracted combinatorial map is equal to $\mathscr{S}\mathscr{D} = \mathscr{D} - K$ (Definition 1). These surviving darts are connected in G by connecting walks:

Definition 2 (*Connecting walk*). Given a connected combinatorial map $G = (\mathscr{D}, \sigma, \alpha)$, a contraction kernel K and a dart $d \in \mathscr{S}\mathscr{D}$, the *connecting walk* associated to d is equal to

$$\mathrm{CW}(d) = d, \varphi(d), \ldots, \varphi^{n-1}(d)$$
$$\text{with } n = \mathrm{Min}\{p \in \mathbb{N}^* | \varphi^p(d) \in \mathscr{S}\mathscr{D}\}$$

Note that only the first dart of each connecting walk survives. Each connecting walk $\mathrm{CW}(d)$ connects the surviving darts d and $\varphi^n(d)$ by a sequence of non-surviving darts. Moreover, the set of connecting walks defines a partition of the initial set of darts $\mathscr{D}$ (Brun and Kropatsch, 1999a). We have thus

$$\mathscr{D} = \bigsqcup_{d \in \mathscr{S}\mathscr{D}} \mathrm{CW}(d) \tag{2}$$

where $\bigsqcup$ denotes an union of disjoint sets.

If $\mathscr{D}_K$ denotes the set of connecting walks defined on G by the contraction kernel K, the following applications define permutations on $\mathscr{D}_K$ (see formal proof in Brun and Kropatsch, 1999a):

$$\alpha_K \begin{pmatrix} \mathscr{D}_K & \to & \mathscr{D}_K \\ \mathrm{CW}(d) & \mapsto & \mathrm{CW}(\alpha(d)) \end{pmatrix},$$
$$\varphi_K \begin{pmatrix} \mathscr{D}_K & \to & \mathscr{D}_K \\ \mathrm{CW}(d) = d, \ldots, \varphi^{n-1}(d) & \mapsto & \mathrm{CW}(\varphi^n(d)) \end{pmatrix}$$

Indeed, the permutation α_K satisfies $\alpha_K^2(\mathrm{CW}(d)) = \mathrm{CW}(\alpha^2(d)) = \mathrm{CW}(d)$. The permutation α_K is thus an involution (and therefore a permutation). Moreover, the dart $\varphi^n(d)$ is uniquely defined from the dart d. Since each connecting walk contains a unique surviving dart, $\mathrm{CW}(\varphi^n(d))$ is uniquely determined from $\mathrm{CW}(d)$. The application φ_K is thus injective from $\mathscr{D}_K$ to $\mathscr{D}_K$. It is thus also surjective and defines a permutation on $\mathscr{D}_K$. Since α_K and φ_K define two permutations on $\mathscr{D}_K$, $\sigma_K = \varphi_K \circ \alpha_K$ is a permutation as the composition of two permutations. The two permutations α_K and σ_K structure the set of connecting walks $\mathscr{D}_K$ into a combinatorial map $G_K = (\mathscr{D}_K, \sigma_K, \alpha_K)$ (Brun and Kropatsch, 1999a). Moreover, since each connecting walk contains only one surviving dart, we can consider CW as a bijective application from $\mathscr{S}\mathscr{D}$ to $\mathscr{D}_K$. If $G' = (\mathscr{S}\mathscr{D}, \sigma', \alpha')$ denotes the contracted combinatorial map, CW defines an isomorphism between G' and G_K (see proof in Brun and Kropatsch, 1999a). Therefore (see Eq. (1)):

$$\forall d \in \mathscr{S}\mathscr{D} \quad \alpha'(d) = \mathrm{CW}^{-1}(\alpha_C(\mathrm{CW}(d)))$$
$$= \mathrm{CW}^{-1}(\mathrm{CW}(\alpha(d))) = \alpha(d)$$

In the same way, we obtain for any dart d in $\mathscr{S}\mathscr{D}$

$$\begin{aligned} \sigma'(d) &= \mathrm{CW}^{-1}(\sigma_C(\mathrm{CW}(d))) \\ &= \mathrm{CW}^{-1}(\varphi_K \circ \alpha_K(\mathrm{CW}(d))) \\ &= \varphi^n(\alpha(d)) \end{aligned} \tag{3}$$
$$\text{With } n = \mathrm{Min}\{p \in \mathbb{N}^* | \varphi^p(\alpha(d)) \in \mathscr{S}\mathscr{D}\}$$

Note that, $\varphi^{n-1}(\alpha(d))$ is the last dart of $\mathrm{CW}(\alpha(d))$. Therefore, the permutation α remains unchanged in the contracted combinatorial map while the permutation σ' maps each surviving dart d to the φ-successor of the last dart of $\mathrm{CW}(\alpha(d))$. Therefore, the computation of the σ-successor of a dart d in the contracted combinatorial map requires to traverse $\mathrm{CW}(\alpha(d))$. Sequential Algorithm 1 computes the σ-successor of all the surviving darts in the contracted combinatorial map. Since the set of connecting walks forms a partition of $\mathscr{D}$ (Eq. (2)), Algorithm 1 has to traverse $|\mathscr{D}|$ darts. Its complexity is thus equal to $\mathscr{O}(|\mathscr{D}|)$.

Algorithm 1 (*Computation of the permutation* σ')

```
dart contracted_sigma (G = (𝒟, σ, α), K)
{
  For each d ∈ 𝒮𝒟 = 𝒟 − K
  do
    d' = φ(α(d)) = σ(d) // Second dart of
         CW(α(d))
    while (d' ∈ K) // computation of
         CW(α(d))
      d' = φ(d')
    σ'(d) = d'
  done
}
```

The basic idea of a parallel implementation of Algorithm 1 is to compute concurrently for each dart d, the first surviving dart encountered when

traversing the φ-orbit of d from d. The result of this parallel computation is stored in an array survive initialized to survive$[d] = d$ for each dart (Algorithm 2, lines 3–4). We have thus

$$\forall d \in \mathscr{D} \quad \begin{cases} \text{survive}[d] & = \varphi^{q}(d) \quad \text{with } q = \text{Min}\{p \in \mathbb{N} | \varphi^{p}(d) \in \mathscr{S}\mathscr{D}\} \\ & = \varphi^{n-1}(d) \quad \text{with } n = \text{Min}\{p \in \mathbb{N}^{*} | \varphi^{p-1}(d) \in \mathscr{S}\mathscr{D}\} \end{cases}$$

Moreover, using Eq. (3):

$$\begin{aligned} \sigma'(d) &= \varphi^{n}(\alpha(d)) = \varphi^{n-1}(\varphi(\alpha(d))) \\ &= \varphi^{n-1}(\sigma(d)) \\ \text{with } n &= \text{Min}\{p \in \mathbb{N}^{*} | \varphi^{p}(\alpha(d)) \in \mathscr{S}\mathscr{D}\} \\ &= \text{Min}\{p \in \mathbb{N}^{*} | \varphi^{p-1}(\sigma(d)) \in \mathscr{S}\mathscr{D}\}. \end{aligned}$$

Therefore, (Algorithm 2, lines 9–10): $\sigma'(d) = \varphi^{n-1}(\sigma(d)) = \text{survive}[\sigma(d)]$.

Algorithm 2 (*Parallel computation of the permutation* σ')

```
1   dart parallel_contracted_sigma
    (G = (𝒟, σ, α), K)
2   {
3     for each d in 𝒟 do in parallel
4       survive[d] = d
5     for each d in 𝒟 do in parallel
6       while (survive[d] ∈ K)
7         survive[d] = survive[φ(d)]
8
9    for each d in 𝒮𝒟 do in parallel
10      σ'(d) = survive[σ(d)]
11  }
```

The parallel complexity of Algorithm 2 is determined by the number of elementary steps performed in the second loop (lines 5–7). This number is equal to the cyclic distance between d and its associated surviving dart. If we denote by D the maximum of these distances, the parallel algorithm will terminate after D steps. Therefore, worst case parallel complexity of our algorithm is linear in the cyclic max-distance between surviving darts.

5. Conclusion

We have defined in this article the notion of contraction kernel which allows to perform several contractions simultaneously. Successive kernels may be combined in an equivalent kernel or on the contrary one kernel may be decomposed into smaller ones. This notion of equivalent kernels provides a better flexibility in the design of kernels which allows to better fit the pyramid structure to the data. This article also provides the main theoretical results required to compute the reduced combinatorial map defined by a contraction kernel. These results are used to design one sequential and one parallel algorithm computing the contracted combinatorial map.

Finally, the combinatorial pyramid should be a nice framework to take up some of the challenges defined during GbR'2001. Indeed, a hierarchical matching should reduce the computation cost of matching algorithms for large graphs. The efficiency of matching algorithms should be further improved by using the explicit encoding of the orientation which defines an additional constraint. Compared to a RAG, a combinatorial map encodes multiple boundaries, inclusion relationships and the orientation of edges around each vertex. This model may be further enriched in order to "increase the intelligence of a pixel-based graph". Finally, combinatorial maps are defined in any dimension and offer thus a nice framework for applications using 3D or 4D data.

References

Bertrand, Y., Damiand, G., Fiorio, C., 2001. Topological map: minimal encoding of 3D segmented images. In: Jolion, J.M., Kropatsch, W., Vento, M. (Eds.), 3rd Workshop on Graph-Based Representations in Pattern Recognition, IAPR-TC15, CUEN, Ischia, Italy, pp. 64–73.

Brun, L., Kropatsch, W., 1999a. Pyramids with combinatorial maps. Tech. Rep. PRIP-TR-057, PRIP, TU Wien. Available from <www.prip.tuwien.ac.at/ftp/pub/publications/trs/tr57.ps.gz>.

Brun, L., Kropatsch, W.G., 1999b. Dual contraction of combinatorial maps. In: Kropatsch, W.G., Jolion, J.-M. (Eds.), 2nd IAPR-TC-15 Workshop on Graph-based Representations, vol. 126. Österreichische Computer Gesellschaft, Haindorf, Austria, pp. 145–154.

Dickinson, S., Pellilo, M., Zabih, R., 2001. Special section on graph algorithms and computer vision. Pattern Anal. Machine Intelligence 23 (10), 1049–1144.

Gareth, A.J., Singerman, D., 1978. Theory of maps on orientable surfaces. In: Proceedings of the London Mathematical Society, vol. 3. London Mathematical Society, pp. 273–307.

Gdalyahu, Y., Weinshall, D., Werman, M., 2001. Self-organization in vision: stochastic clustering for image segmentation, perceptual grouping, and image database organization. Pattern Anal. Machine Intelligence 23 (10), 1053–1074.

Jolion, J.-M., 2001. Data driven decimation of graphs. In: Jolion, J.-M., Kropatsch, W., Vento, M. (Eds.), Proceedings of 3rd IAPR-TC15 Workshop on Graph based Representation in Pattern Recognition, Ischia, Italy, pp. 105–114.

Kropatsch, W.G., 1995. Building irregular pyramids by dual graph contraction. IEE Proc. Vision, Image and Signal Process. 142 (6), 366–374.

Lienhardt, P., 1989. Subdivisions of *n*-dimensional spaces and *n*-dimensional generalized maps. In: Annual ACM Symposium on Computational Geometry, all, vol. 5.

Meer, P., 2001. From a robust hierarchy to a hierarchy of robustness. In: Davis, L.S. (Ed.), Foundations of Image Understanding. Kluwer, pp. 323–347. (Ch. 11).

Montanvert, A., Meer, P., Rosenfeld, A., 1991. Hierarchical image analysis using irregular tessellations. IEEE Trans. Pattern Anal. Machine Intelligence 13 (4), 307–316.

Pailloncy, J.-G., Kropatsch, W.G., Jolion, J.-M., 1998. Object matching on irregular pyramid. In: Jain, A.K., Venkatesh, S., Lovell, B.C. (Eds.), 14th International Conference on Pattern Recognition, vol. II. IEEE Computer Society, pp. 1721–1723.