Successive Projection Graph Matching

Barend Jacobus van Wyk[1,3], Michaël Antonie van Wyk[2], and Hubert Edward Hanrahan[3]

[1] Kentron, a division of Denel
Centurion, South Africa
ben.van.wyk@kentron.co.za
[2] Rand Afrikaans University
Johannesburg, South Africa
mavw@ing.rau.ac.za
[3] University of the Witwatersrand
Johannesburg, South Africa
h.hanrahan@ee.wits.ac.za

Abstract. The *Successive Projection Graph Matching* (SPGM) algorithm, capable of performing full- and sub-graph matching, is presented in this paper. *Projections Onto Convex Sets* (POCS) methods have been successfully applied to signal processing applications, image enhancement, neural networks and optics. The SPGM algorithm is unique in the way a constrained cost function is minimized using POCS methodology. Simulation results indicate that the SPGM algorithm compares favorably to other well-known graph matching algorithms.

1 Introduction

In image processing applications, it is often required to match different images of the same object or similar objects based on the structural descriptions constructed from these images. If the structural descriptions of objects are represented by attributed relational graphs, different images can be matched by performing *Attributed Graph Matching* (AGM). Because of the combinatorial nature of the AGM problem, it can be efficiently solved by an exhaustive search only when dealing with extremely small graphs.

According to [1], graph matching algorithms can be divided into two major approaches. In general, the first approach constructs a state-space which is searched using heuristics to reduce complexity. Examples of algorithms belonging to this group are those proposed by You and Wong [2], Tsai and Fu [3, 4], Depiero *et al.* [5], Eshera and Fu [6], Bunke and Shearer [7], Bunke and Messmer [8] and Allen *et al.* [9].

The second approach, which is also our approach, is based on function optimization techniques. This approach includes earlier techniques such as the symmetric *Polynomial Transform Graph Matching* (PTGM) algorithm of Almohamad [10], the *Linear Programming Graph Matching* (LPGM) algorithm of Almohamad and Duffuaa [**?**] and the *Eigen-decomposition Graph Matching*

T. Caelli et al. (Eds.): SSPR&SPR 2002, LNCS 2396, pp. 263–271, 2002.

(EGM) method of Umeyama [11]. More recent techniques include a multitude of Bayesian, genetic, neural network and relaxation-based methods.

The *Graduated Assignment Graph Matching* (GAGM) algorithm of Gold and Rangarajan [1] proved to be very successful. It combines graduated non-convexity, two-way assignment constraints and sparsity. The literature on optimization methods for graph matching has also been complemented by the work of Hancock and his associates [13–17]. Their work builds on a relational consistency gauged by an exponential probability distribution.

This paper focuses on matching fully-connected, undirected attributed graphs using a *Projections Onto Convex Sets* (POCS) method. POCS methods have been successfully applied to signal processing applications, image enhancement, neural networks and optics [22]. In this paper, the *Successive Projection Graph Matching* (SPGM) algorithm is presented. This algorithm is unique in the way a constrained cost function is minimized using POCS methodology. Although the algorithm of Gold and Rangarajan [1] also uses a successive approximation approach, our algorithm is significantly different. We do not use graduated non-convexity, and we do not enforce constraints using repeated row and column normalization. Instead, constraints are enforced by mapping onto appropriate convex sets.

The outline of the presentation is as follows: In section 2 we introduce a constrained cost function, and in section 3 we show how successive projections can be used to obtain a constrained optimum. Numerical results, obtained during the evaluation of our algorithm, are presented in section 4.

2 Cost Function Formulation

The focus of this paper is on matching graphs where a duplicate graph, say

$$G = \left(V, E, \{\mathbf{A}_g\}_{g=1}^{r}, \{\mathbf{B}_h\}_{h=1}^{s}\right) \tag{1}$$

is matched to a reference graph, say

$$G' = \left(V', E', \{\mathbf{A}'_g\}_{g=1}^{r}, \{\mathbf{B}'_h\}_{h=1}^{s}\right) \tag{2}$$

where $\mathbf{A}_g \in \mathbb{R}^{n\times n}$, $\mathbf{B}_h \in \mathbb{R}^{n\times 1}$, $\mathbf{A}'_g \in \mathbb{R}^{n'\times n'}$and $\mathbf{B}'_h \in \mathbb{R}^{n'\times 1}$ represent the edge attribute adjacency matrices and vertex attribute vectors respectively. The reference and duplicate graphs each have r edge attributes and s vertex attributes. The number of vertices of G' (respectively, G) is $n' := |V'|$ (respectively, $n := |V|$). Here we consider the general case of sub-graph matching. *Full-graph Matching* (FGM) refers to matching two graphs having the same number of vertices (i.e. $n' = n$) while *Sub-graph Matching* (SGM) refers to matching two graphs having a different number of vertices (i.e. $n' > n$).

We say that G is *matched* to some sub-graph of G' if there exists a matrix $\mathbf{P} \in \mathrm{Per}(n, n')$, where $\mathrm{Per}(n, n')$ is the set of all $n\times n'$ permutation sub-matrices, such that

$$\mathbf{A}_g = \mathbf{P}\mathbf{A}'_g\mathbf{P}^T, \qquad g = 1, ..., r \tag{3}$$

and

$$\mathbf{B}_h = \mathbf{P}\mathbf{B}'_h, \qquad h = 1, ..., s. \tag{4}$$

As shown in [19], the *Attributed Graph Matching* (AGM) problem can be expressed as a combinatorial optimization problem. However, due to the difficulty in solving this combinatorial optimization problem, we construct an approximate solution. Following an approach similar to [1], we can express the undirected AGM problem as finding the matrix $\overline{\mathbf{P}}$, such that the objective function,

$$J(\mathbf{p}) = -\mathbf{p}^T\mathbf{X}\mathbf{p} - \mathbf{y}^T\mathbf{p}, \tag{5}$$

is minimized, where $\mathbf{p} = \text{vec}\left(\overline{\mathbf{P}}\right)$ is subject to

$$0 \leq \overline{P}_{ij} \leq 1, \tag{6}$$

$$\sum_{j=1}^{n'} \overline{P}_{ij} = 1, \qquad i = 1, .., n \tag{7}$$

and

$$\sum_{i=1}^{n} \overline{P}_{ij} \leq 1, \qquad j = 1, ..., n'. \tag{8}$$

Here $\overline{\mathbf{P}} := \left(\overline{P}_{ij}\right)$ and vec$(\cdot)$ denote the vectorization operation from linear algebra. The elements of the matrix $\mathbf{X} \in \mathbb{R}^{nn' \times nn'}$ are given by

$$X_{kl} = \frac{\alpha}{\left(\sum_{g=1}^{r} \left| A^g_{l-\left\lfloor \frac{l-1}{n} \right\rfloor n,\, k-\left\lfloor \frac{k-1}{n} \right\rfloor n} - A'^g_{\left\lfloor \frac{l-1}{n} \right\rfloor + 1,\, \left\lfloor \frac{k-1}{n} \right\rfloor + 1} \right| + \alpha\right)}$$

except when $k = l$, $\left\lfloor \frac{k-1}{n} \right\rfloor + 1 = \left\lfloor \frac{l-1}{n} \right\rfloor + 1$ or $k - \left\lfloor \frac{k-1}{n} \right\rfloor n = l - \left\lfloor \frac{l-1}{n} \right\rfloor n$, in which case $X_{kl} = 0$. Here $\mathbf{A}_g := (A^g_{ij}) \in \mathbb{R}^{n \times n}$, $\mathbf{A}'_g := (A'^g_{ij}) \in \mathbb{R}^{n' \times n'}$, $|\cdot|$ denotes the absolute value, $k = 1, ..., nn'$, $l = 1, ..., nn'$ and α is a parameter controlling the steepness of the compatibility function, normally chosen equal to one. The elements of the vector $\mathbf{y} \in \mathbb{R}^{nn' \times 1}$ are given by

$$y_k = \frac{\alpha}{\left(\sum_{h=1}^{s} \left| B^m_{k-\left\lfloor \frac{k-1}{n} \right\rfloor n} - B'^m_{\left\lfloor \frac{k-1}{n} \right\rfloor + 1} \right| + \alpha\right)}$$

where $\mathbf{B}_h := (B^h_i) \in \mathbb{R}^{n \times 1}$, $\mathbf{B}'_h := (B'^h_j) \in \mathbb{R}^{n' \times 1}$, $k = 1, ..., nn'$.

Relaxation methods such as [17] and [18] in general only enforce the constraint given by Eq. 6 and the row constraint given by Eq. 7. In addition, the SPGM algorithm also enforces the column constraint given by Eq. 8. Similar to our algorithm, the GAGM algorithm [1] also enforces a column constraint, but uses a significantly different approach. Central to our method is the projection of a vector onto the intersection of two convex sets formed by the row and column constraints. As noted by [17], the row constraints form a closed convex set, which can be expressed as

$$\mathbf{C}_r = \left\{ \mathbf{p}_r \in \mathbb{R}^{nn'} : \begin{array}{l} \mathbf{p}_r = [\bar{\mathbf{p}}_1, ..., \bar{\mathbf{p}}_n], \\ \bar{\mathbf{p}}_i = [\overline{P}_{i1}, ..., \overline{P}_{in'}], \\ \sum_{j=1}^{n'} \overline{P}_{ij} = 1, \\ 0 \leq \overline{P}_{ij} \leq 1 \end{array} \right\}.$$

In a similar manner, the column constraints can also be expressed as the set,

$$\mathbf{C}_c = \left\{ \mathbf{p}_c \in \mathbb{R}^{nn'} : \begin{array}{l} \mathbf{p}_c = [\bar{\mathbf{p}}_1, ..., \bar{\mathbf{p}}_{n'}], \\ \bar{\mathbf{p}}_j = [\overline{P}_{1j}, ..., \overline{P}_{nj}], \\ \sum_{i=1}^{n} \overline{P}_{ij} \leq 1, \\ 0 \leq \overline{P}_{ij} \leq 1 \end{array} \right\},$$

which is also closed and convex. Note that the intersection of the convex sets $\mathbf{C}_r$ and $\mathbf{C}_c$, denoted by $\mathbf{C}_0 = \mathbf{C}_r \cap \mathbf{C}_c$, is non-empty.

3 Projected Successive Approximations

3.1 The SPGM Algorithm

The following pseudo-code describes the SPGM algorithm with index $k > 0$:
while $(k < I$ and $\delta > \epsilon)$ or $(k < 3)$
$\widetilde{\mathbf{p}}^{k+1} = \mathbf{p}^k - \frac{1}{s_0^k}\left[\bigtriangledown J\left(\mathbf{p}^k\right)\right]$
$\mathbf{p}^{k+1} = T_0\left(\widetilde{\mathbf{p}}^{k+1}\right)$
$\delta = \left(\mathbf{p}^{k+1} - \mathbf{p}^k\right)^T \left(\mathbf{p}^{k+1} - \mathbf{p}^k\right)$
$k = k + 1$
end

The vector $\mathbf{p}^0$ is initialized to $\left[\frac{1}{n'}, ..., \frac{1}{n'}\right]^T$. $T_0\left(\widetilde{\mathbf{p}}^{k+1}\right)$ denotes the projection of $\widetilde{\mathbf{p}}^{k+1}$ onto $\mathbf{C}_0$, which will be discussed in section 3.2. We obtain $\widetilde{\mathbf{p}}^{k+1}$ by approximating Eq. 5, using a spherical function given by

$$\widetilde{J}\left(\widetilde{\mathbf{p}}^{k+1}\right) = J\left(\mathbf{p}^k\right) + \bigtriangledown^T J\left(\mathbf{p}^k\right)\left(\widetilde{\mathbf{p}}^{k+1} - \mathbf{p}^k\right) + \frac{1}{2}\left(\widetilde{\mathbf{p}}^{k+1} - \mathbf{p}^k\right)^T S_o^k \left(\widetilde{\mathbf{p}}^{k+1} - \mathbf{p}^k\right), \tag{9}$$

where $S_o^k = \text{diag}\left(s_0^k, ..., s_0^k\right) = s_0^k \mathbf{I}$, $\bigtriangledown J\left(\mathbf{p}^k\right)$ denotes the gradient vector of Eq. 5, given by $-2\mathbf{X}\mathbf{p}^k - \mathbf{y}$, and s_0^k is a curvature parameter. The vector $\widetilde{\mathbf{p}}^{k+1}$ is calculated as the minimum of the spherical function, occurring where

$$\bigtriangledown^T J\left(\mathbf{p}^k\right) + \left(\widetilde{\mathbf{p}}^{k+1} - \mathbf{p}^k\right)^T S_o^k = 0. \tag{10}$$

Since $S_o^k = s_0^k \mathbf{I}$, we obtain the simple update rule $\widetilde{\mathbf{p}}^{k+1} = \mathbf{p}^k - \frac{1}{s_0^k}\left[\bigtriangledown J\left(\mathbf{p}^k\right)\right]$. Although the curvature parameter, s_0^k, can be varied every iteration, the best results were obtained by keeping it constant.

The convergence parameter, ϵ, is normally chosen $< 10^{-3}$. The iteration parameter, I, limits the maximum number of iterations. Once the algorithm

has terminated, devec($\mathbf{p}$), is used to obtain an estimate to $\mathbf{P} \in \mathrm{Per}(n, n')$, by setting the maximum value in each row equal to one and the rest of the values in each row equal to zero. Here devec$(\cdot)$ denotes the inverse of vec$(\cdot)$, the matrix vectorization operation from linear algebra.

3.2 Projection onto $\mathbf{C_0}$

Once we obtain $\widetilde{\mathbf{p}}^{k+1}$, our objective is to find $\mathbf{p}^{k+1} \in \mathbf{C}_0$ such that

$$\left\|\widetilde{\mathbf{p}}^{k+1} - \mathbf{p}^{k+1}\right\| = \min_{\mathbf{z} \in \mathbf{C}_0} \left\|\widetilde{\mathbf{p}}^{k+1} - \mathbf{z}\right\|. \tag{11}$$

From the POCS theory [21], Eq. 11 implies that $\mathbf{p}^{k+1} = T_0\left(\widetilde{\mathbf{p}}^{k+1}\right)$, the projection of $\widetilde{\mathbf{p}}^{k+1}$ onto the set $\mathbf{C}_0$. By applying certain fundamental POCS results, detailed in [22–23] to our problem, we can construct a sequence that converges to $T_0\left(\widetilde{\mathbf{p}}^{k+1}\right)$. The algorithm for obtaining $T_0\left(\widetilde{\mathbf{p}}^{k+1}\right)$ is described by the following pseudo-code:

Calculating $T_0\left(\widetilde{\mathbf{p}}^{k+1}\right)$:
initialization: $k_C = 1,\ \delta_{\mathbf{C}_0} > 0.05$
while $(\delta_C > 0.05$ and $k_C < n')$
$\mathbf{p}_h = \mathbf{p}$
$\mathbf{p} = T_r(\mathbf{p})$
$\mathbf{p} = T_c(\mathbf{p})$
$\delta_{\mathbf{C}_0} = (\mathbf{p}_h - \mathbf{p})^T (\mathbf{p}_h - \mathbf{p})$
$k_C = k_C + 1$
end

$T_r(\mathbf{p})$ denotes the projection of $\mathbf{p}$ onto the set $\mathbf{C}_r$ and $T_c(\mathbf{p})$ denotes the projection of $\mathbf{p}$ onto the set $\mathbf{C}_c$, given by the following pseudo-code. The superscript $k+1$ has been omitted for simplicity. The notation $\mathbf{p}(i : j : k)$ indicates that we select every j–th element from the vector $\mathbf{p}$, starting with the i–th element and ending with the k–th element. The operation $[\mathbf{s}, \mathbf{d}] = \mathrm{sort}[\bar{\mathbf{p}}]$ indicates that we sort the elements of $\bar{\mathbf{p}}$ in ascending order, where $\mathbf{s}$ is the sorted vector and $\mathbf{d}$ is a vector containing the pre-sorted positions of the sorted elements.

Calculating $T_r(\mathbf{p})$:
for $i = 1 : n$
$\phi = \mathrm{sum}[\mathbf{p}(i : n : n(n'-1) + i)]$
$\sigma = n'$
$\bar{\mathbf{p}} = \mathbf{p}(i : n : n(n'-1) + i)$
$[\mathbf{s}, \mathbf{d}] = \mathrm{sort}[\bar{\mathbf{p}}]$
for $j = 1 : n'$
$\mathbf{s}(j) = \mathbf{s}(j) + \frac{1-\phi}{\sigma}$
if $\mathbf{s}(j) < 0$
$\mathbf{s}(j) = 0$
$\phi = \phi - \bar{\mathbf{p}}(\mathbf{d}(j))$

```
                σ = σ − 1
            end
            p̄(d(j)) = s(j)
        end
        p(i : n : n(n′ − 1) + i) = p̄
end
```

When $n = n'$, the approach used to calculate $T_c(\mathbf{p})$ is similar to that which we used to calculate $T_r(\mathbf{p})$. The pseudo-code below is for the case $n < n'$:

```
Calculating T_c(p):
for j = 1 : n′
        φ = sum[p(n(j − 1) + 1 : nj)]
        if φ > 1
            σ = n
            p̄ = p(n(j − 1) + 1 : nj)
            [s, d] = sort[p̄]
            for i = 1 : n
                s(i) = s(i) + (1 − φ)/σ
                if s(i) < 0
                    s(i) = 0
                    φ = φ − p̄(d(i))
                    σ = σ − 1
                end
                p̄(d(i)) = s(i)
            end
            p(n(j − 1) + 1 : nj) = p̄
        end
end
```

4 Simulation Results

In order to evaluate the performance of the SPGM algorithm, the following procedure was used: Firstly, the parameters n', n, r and s were fixed. For every iteration, a reference graph G' was generated randomly with all attributes distributed between 0 and 1. An $n \times n'$ permutation sub-matrix, $\mathbf{P}$, was also generated randomly, and then used to permute the rows and columns of the edge attribute adjacency matrices and the elements of the vertex attribute vectors of G'. Next, an independently generated noise matrix (vector, respectively) was added to each edge attribute adjacency matrix (vertex attribute vector, respectively) to obtain the duplicate graph G. The element of each noise matrix/vector was obtained by multiplying a random variable —uniformly distributed on the interval $[-1/2, 1/2]$— by the noise magnitude parameter ε. Different graph matching algorithms were then used to determine a permutation sub-matrix which approximates the original permutation sub-matrix $\mathbf{P}$.

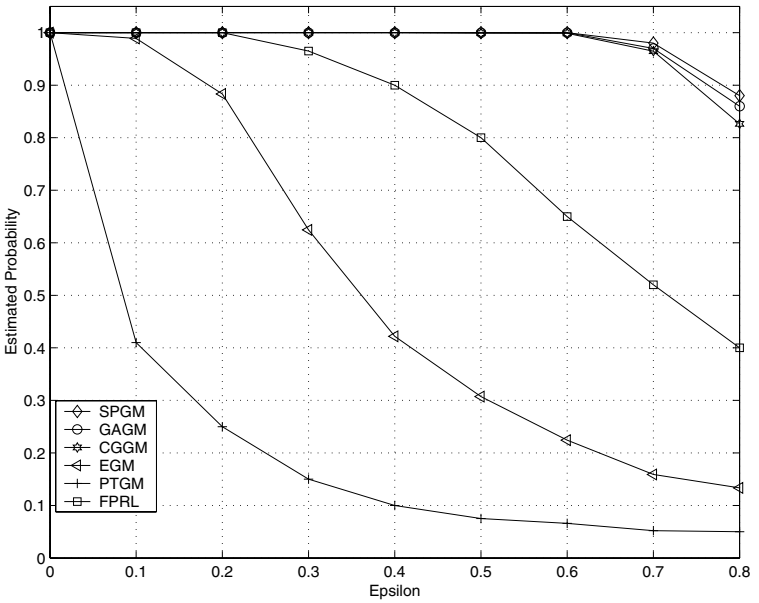

Fig. 1. Matching of (30,3,3) attributed graphs: Estimated probability of correct vertex-vertex matching versus ε

In figure 1, the performance of the SPGM algorithm is compared to the performance of the GAGM [1], PTGM [10], EIGGM [11] and CGGM [20] algorithms for $n' = 30$, $n = 30$, $r = 3$ and $s = 3$. The performance of the SPGM algorithm is also compared to the performance of the well-known *Faugeras-Price Relaxation Labelling* (FPRL) method [18]. The EIGGM algorithm has been adapted for attributed graph matching by calculating separate permutation sub-matrices for each attribute, and then selecting the permutation sub-matrix associated with the minimum cost. The FPRL algorithm was implemented using a step-size parameter of 0.1. The probability of a correct vertex-vertex assignment was estimated for a given value of ε after every 300 trials. From a probabilistic point of view, this provides us with an approximation of how well the proposed algorithm performs for a given noise magnitude.

In figure 2, the sub-graph matching performance of the SPGM is compared to the performances of the GAGM, CGGM, and FPRL algorithms for $n' = 20$, $n = 5$, $r = 3$ an $s = 3$. The EIGGM and PTGM algorithms are not suitable for performing sub-graph matching. The performance of the CGGM algorithm severely degrades when more than half the nodes are missing. The GAGM algorithm was implemented using the default parameters described in [1].

From the results it is evident that the SPGM algorithm is an extremely robust algorithm for performing full- and sub-graph matching. For $n' = 30$ and $n = 30$, the algorithm took on average 5.5 iterations to converge for $\epsilon = 10^{-3}$, $\varepsilon < 0.5$ and $s_0 = 30$. For $n' = 20$ and $n = 5$, the algorithm took on average 13.7

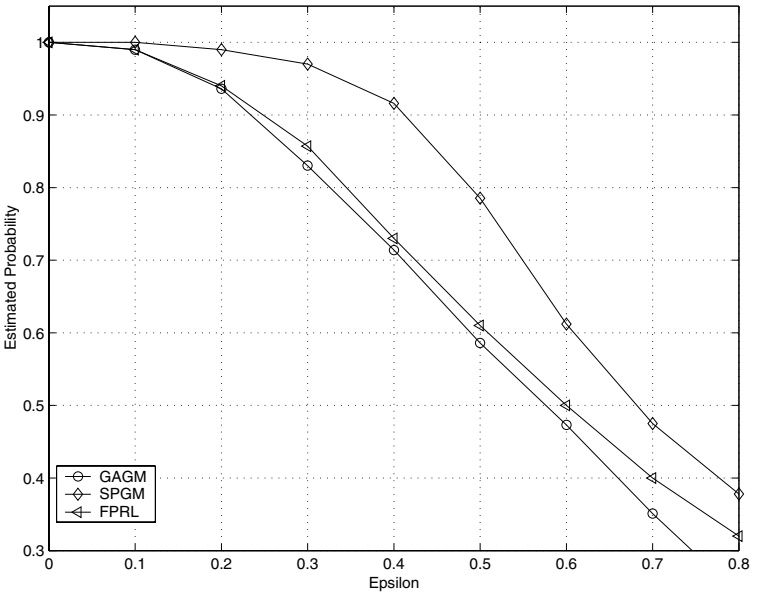

Fig. 2. Matching of (20/5,3,3) attributed graphs: Estimated probability of correct vertex-vertex matching versus ε

iterations to converge for $\epsilon = 10^{-3}$, $\varepsilon < 0.5$ and $s_0 = 30$. The complexity of the SPGM algorithm is $O(n^4)$ per iteration.

5 Conclusion

A novel algorithm for performing attributed full- and sub-graph matching was presented. The SPGM algorithm is unique in the way a constrained cost function is minimized using POCS methodology. Simulation results indicate that the SPGM algorithm is very robust against noise and performs as well or better than the algorithms it was compared against. The SPGM algorithm incorporates a general approach to a wide class of graph matching problems based on attributed graphs, allowing the structure of the graphs to be based on multiple sets of attributes.

References

1. Gold, S., Rangarajan, A.: A Graduated Assignment Algorithm for Graph Matching, IEEE Trans. Patt. Anal. Machine Intell, Vol. 18 (1996) 377–388 263, 264, 265, 269
2. You, M., Wong, K. C.: An Algorithm for Graph Optimal Isomorphism, Proc. ICPR. (1984) 316–319 263
3. Tsai, W.-H., Fu, K.-S.: Error-Correcting Isomorphisms of Attributed Relation Graphs for Pattern Recognition, IEEE Trans. Syst. Man Cybern., Vol. 9 (1997) 757–768

4. Tsai, W.-H., Fu, K.-S.: Subgraph Error-Correcting Isomorphisms for Syntactic Pattern Recognition, IEEE Trans. Systems, Man, Cybernetics, Vol. 13 (1983) 48–62
5. Depiero, F., Trived, M., Serbin, S.: Graph Matching using a Direct Classification of Node Attendance, Pattern Recognition, Vol. 29, No. 6, (1996) 1031–1048 263
6. Eshera, M. A., Fu, K.-S.: A Graph Distance measure for Image Analysis, IEEE Trans. Systems, Man, Cybernetics, Vol. 13 (1984) 398–407 263
7. Bunke, H., Shearer, K.: A Graph Distance Metric Based on the Maximal Common Subgraph, Pattern Recognition Letters, Vol. 19 (1998) 255–259 263
8. Bunke, H., Messmer, B.: Recent Advances in Graph Matching, Int. J. Pattern Recognition Artificial Intell. Vol. 11, No. 1 (1997) 169–203 263
9. Allen, R., Cinque, L., Tanimoto, S., Shapiro, L., Yasuda, D.: A Parallel Algorithm for Graph Matching and Its MarPlas Implementation, IEEE Trans. Parallel and Distb. Syst., Vol. 8, No. 5 (1997) 490–501 263
10. Almohamad, H. A. L.: Polynomial Transform for Matching Pairs of Weighted Graphs, Appl. Math. Modelling, Vol. 15, No. 4 (1991) 216–222 263, 269
11. Umeyama, S.: An Eigendecomposition Approach to Weighted Graph Matching Problems, IEEE Trans. Patt. Anal. Machine Intell., Vol. 10, No. 5 (1988) 695–703 264, 269
12. Cross, A. D. J., Wilson, C., Hancock, E. R.: Inexact Matching Using Genetic Search, Pattern Recognition, Vol. 30, No. 6 (1997) 953–970
13. Finch, A. M., Wilson, R. C., Hancock, R.: Symbolic Matching with the EM Algorithm
, Pattern Recognition, Vol. 31, No. 11 (1998) 1777–1790
14. Williams, M. L., Wilson, R. C., Hancock, E. R.: Multiple Graph Matching with Bayesian Inference, Pattern Recognition Letters, Vol. 18 (1997) 1275–1281
15. Cross, A. D. J., Hancock, E. R.: Graph Matching with a Dual Step EM Algorithm, IEEE Trans. Patt. Anal. Machine Intell., Vol. 20, No. 11 (1998) 1236–1253
16. Wilson, R. C., Hancock, E. R.: A Bayesian Compatibility Model for Graph Matching, Pattern Recognition Letters, Vol. 17 (1996) 263–276.
17. Hummel, R. A., Zucker, S. W.: On the Foundations of Relaxation Labelling Processes, IEEE Trans. Patt. Anal. Machine Intell., Vol. 5, No. 3 (1983) 267–286 265
18. Faugeras, O. D., Price, K. E.: Semantic Description of Aerial Images Using Stochastic Labeling, IEEE Trans. Patt. Anal. Machine Intell., Vol. 3, No. 6 (1981) 633–642 265, 269
19. van Wyk, M. A., Clark, J.: An Algorithm for Approximate Least-Squares Attributed Graph Matching, in Problems in Applied Mathematics and Computational Intelligence, N. Mastorakis (ed.), World Science and Engineering Society Press, (2001) 67-72 265
20. van Wyk, B. J., van Wyk, M. A., Virolleau, F.: The CGGM Algorithm and its DSP implementation, Proc. 3rd European DSP Conference on Education and Research, ESIEE-Paris, 20-21 September (2000) 269
21. Stark, H., Yang, Y.: Vector Space Projections: A Numerical Approach to Signal and Image Processing, Neural Nets and Optics., John Wiley and Sons (1998) 267
22. Youla, D. C.: Mathematical theory of image restoration by the method of convex projections, Chapter 2, in Image Recovery: Theory and Applications, H. Stark (ed.), Academic Press, Orlando, FL (1987)
23. Garey, M. R., Johnson, D. S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman (1979)