[11] S. G. Cao, N. W. Rees, and G. Feng, "Quadratic stability analysis and design of continuous-time fuzzy control system," *Int. J. Syst. Sci.*, vol. 27, pp. 193–203, 1996.
[12] ——, "Stability analysis and design for a class of continuous-time fuzzy control systems," *Int. J. Control*, vol. 64, pp. 1069–1089, 1996.
[13] G. Feng *et al.*, "H_∞ control of nonlinear discrete-time systems based on dynamical fuzzy models," *Int. J. Syst. Sci.*, vol. 31, pp. 229–241, 2000.
[14] J. C. Geromel, P. L. D. Peres, and S. R. Souze, "H_∞ control of discrete-time uncertain systems," *IEEE Trans. Automat. Control*, vol. 39, pp. 1072–1075, May 1994.
[15] K. Tanaka and M. Sano, "A robust stabilization problem of fuzzy control systems and its application to backing up control of a truck-trailer," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 119–134, Apr. 1994.

A PCA Approach for Fast Retrieval of Structural Patterns in Attributed Graphs

Lei Xu and Irwin King

***Abstract*—Attributed graph (AG) is a useful data structure for representing complex patterns in a wide range of applications such as computer vision, image database retrieval, and other knowledge representation tasks where similar or exact corresponding structural patterns must be found. Existing methods for attributed graph matching (AGM) often suffer from the combinatorial problem whereby the execution cost for finding an exact or similar match is exponentially related to the number of nodes the AG contains. In this paper, the square matching error of two AGs subject to permutations is approximately relaxed to a square matching error of two AGs subject to orthogonal transformations. Hence, the *principal component analysis* (PCA) algorithm can be used for the fast computation of the approximate matching error, with a considerably reduced execution complexity. Experiments demonstrate that this method works well and is robust against noise and other simple types of transformations.**

***Index Terms*—Attributed graph (AG), graph matching, principal component analysis (PCA).**

I. Introduction

In common information retrieval tasks when a large data set is involved, i.e., image database and data mining applications, a speedy and accurate way to locate objects of similar structures is essential for efficient and robust data access.

Attributed graph (AG) is a useful tool for representing structural knowledge since it is more expressive, flexible, and powerful than the typical vector representation, e.g., feature vectors. Thus, AG has been widely used in many image processing and computer vision tasks, e.g., road extraction [2], medical image databases [9], Chinese character recognition [11], and computer vision [6].

The key problem for information retrieval in databases with AG data structure is to match two AGs, called *attributed graph matching (AGM)* [3]. In the literature, different methods have been devised to implement an exact or similar match, e.g., the graduated assignment algorithm for subgraph isomorphism [5] and others [11]. However, for existing matching methods the computation of the matching error is highly coupled with the search of the one-to-one correspondence between each node and edge of two AGs. Thus, they usually suffer from the combinatorial problem whereby the computing matching error for finding an exact or similar match is exponentially related to the number of nodes in the AG. This makes the use of AGM impractical in many situations.

In practical applications, we often encounter a tradeoff between accuracy and speed during the retrieval process. In this paper, we attempt to separate the computation of the matching error from the search of the one-to-one correspondence between each node and edge of two AGs. We introduce a method to speed up the retrieval process by a two-stage approximate matching process. In the first stage, the exact square matching error of two AGMs is replaced by an approximation so that principal component analysis (PCA) [8], [13], [14], can be used to quickly calculate this approximate matching error. We prune away most of the AGs with large matching errors and keep a small subset of AGs for the second stage where a further processing can be made either interactively with human's help or by a more accurate matching method.

This paper focuses on the process which is being performed in the first stage. In the next section, we present the theoretical formulation of the PCA approach for a fast approximate AG matching. We then performed a number of experiments in Section III demonstrating the success of this approach. Finally, we conclude with some discussions in Section IV.

II. Attributed Graph Matching and its Approximate Stable Points by the PCA Approach

We consider an AG $G = [(V, V_a), (E, E_a)]$, where $V = \{v_1, \ldots, v_n\}$, $V_a = \{av_1, \ldots, av_n\}$ is the set of nodes and their attributes, respectively, and $E = \{e_{ij} \,|\, e_{ij} = e(v_i, v_j)\}$ where $e(v_i, v_j)$ is the edge from node v_i to v_j, $\forall i, j, i \neq j$, $E_a = \{ae_{ij}\}$ where ae_{ij} is the attribute of e_{ij}, $\forall e_{ij} \in E$ with av_i and ae_{ij} being real numbers. When there is no edge from v_i to v_j, we let ae_{ij} be zero. Moreover, we also assume that $ae_{ij} = ae_{ji}$. Specifically, an AG with n nodes reduces into an adjacency matrix for an ordinary graph when av_i and ae_{ij} are binary with "1" denoting the presence of the node or edge and "0" the absence.

The problem of matching two AGs $G = [(V, V_a), (E, E_a)]$ and $G' = [(V', V_a'), (E', E_a')]$ with node sets $V = \{v_1, \ldots, v_n\}$, $V' = \{v_1', \ldots, v_n'\}$ means setting up a one-to-one correspondences between the nodes of V and V'. Generally speaking, there are $n!$ combination for such correspondences (simply we denote a combination by c and the set of all the $n!$ combinations by S_c). Usually, the matching error of building a correspondence between $v \in V$ and $v' \in V'$ is defined as $d(v, v') = (av - av')^2$, and the matching error of building a correspondence between $e_{ij} \in E$ and $e_{kl}' \in E'$ is defined as $d(e_{ij}, e_{kl}') = (ae_{ij} - ae_{kl}')^2$. Thus, for each combination c, we can sum up all the $d(v, v')$ and $d(e_{ij}, e_{kl}')$ to get a total matching error

$$D_c(G, G') = \sum_{i=1}^{n} d\left(v_i, v'_{\pi(i)}\right) + \sum_{r=1}^{n} \sum_{q=1,\, q\neq r}^{n} d\left(e(v_r, v_q), e'\left(v'_{\pi(r)}, v'_{\pi(q)}\right)\right) \quad (1)$$

where $\pi(1, 2, \ldots, n)$ produces $(i_1, i_2, \ldots, i_n)$—a permutation of $1, 2, \ldots, n$, and $\pi(j) = i_j$. A specific permutation corresponds to a specific combination.

Manuscript received November 19, 1997; revised February 29, 2000 and May 31, 2001. This work was supported in part by Earmarked grants from the Hong Kong's Research Grants Council CUHK 4383/99E and CUHK 4407/99E. This paper was recommended by Associate Editor M. S. Obaidat.

The authors are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong.

Publisher Item Identifier S 1083-4419(01)08544-2.

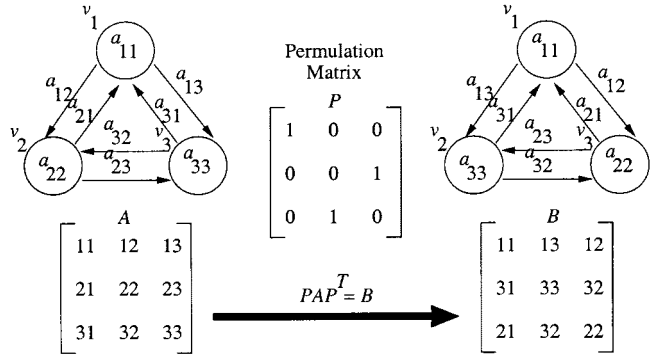


Fig. 1. Diagram illustrating the AG permutation process.

The goal of matching G to G' is to choose one that has the minimum value of

$$D(G, G') = \min_{c \in S_c} D_c(G, G')$$

and then to use this $D(G, G')$ as the matching error of G and G' to classify G according to the computed values $D(G, G_1), \ldots, D(G, G_m)$.

From the definition of $D(G, G')$, we know that its computation is a combinatorial search of correspondences between nodes and edges. This is a common feature that is shared by most existing matching methods. Hence, one of the ways to reduce this costly computation of searching is to modify the definition of $D(G, G')$ into an approximation to avoid a detailed and accurate calculation.

As shown in Fig. 1, given A as an $n \times n$ matrix with its diagonal elements $a_{ii} = av_i$, $i = 1, \ldots, n$ and off-diagonal elements $a_{ij} = ae_{ij}$, $i = 1, \ldots, n$, $j = 1, \ldots, i-1, i+1, \ldots, n$ and B as an $n \times n$ matrix with its diagonal elements $b_{ii} = av'_i$, $i = 1, \ldots, n$ and off-diagonal elements $b_{ij} = ae'_{ij}$, $i = 1, \ldots, n$, $j = 1, \ldots, i-1, i+1, \ldots, n$, we rewrite (1) into the following matrix form:

$$\begin{aligned} D_c(G, G') &= D_P(G, G') \\ &= \operatorname{tr}\left[\left(PAP^T - B\right)\left(PAP^T - B\right)^T\right] \end{aligned} \tag{2}$$

where P is a permutation matrix of π (and thus the combination c) and tr is the trace function which sums up the diagonal elements of the square matrix. In this case, both A and B are symmetric matrices. Usually, A and B are called the associate matrices of G and G', respectively.

With (2), the matching of G and G' can be expressed as a minimization problem as

$$\begin{aligned} D(G, G') &= D_{P^*}(G, G') \\ &\equiv \arg\min_P D_P(G, G') \\ &= \arg\min_P \operatorname{tr}\left[\left(PAP^T - B\right)\left(PAP^T - B\right)^T\right]. \end{aligned} \tag{3}$$

The resulting $D(G, G')$ is called the matching error of G and G'. Especially, G and G' are isomorphic when $D(G, G') = 0$. Here, the computation of $D(G, G')$ still interweaves implicitly with the combinatorial search of correspondences between nodes and edges, since the searching of P over all the permutation matrices is still a combinatorial problem.

The permutation matrix is a special type of orthogonal matrix. As an approximation to the $D(G, G')$ defined by (3), we relax the minimization of (3) with respect to an orthogonal matrix ϕ

$$\begin{aligned} D_o(G, G') &= D_{\phi^*}(G, G') \\ &\equiv \arg\min_\phi D_\phi(G, G') \\ &= \arg\min_\phi \operatorname{tr}\left[\left(\phi A\phi^T - B\right)\left(\phi A\phi^T - B\right)^T\right]. \end{aligned} \tag{4}$$

The main characteristics of the matching error function in (4) are described by the following theorem.

Theorem 1:

1) When G and G' are isomorphic, i.e., $PAP^T = B$ with P being a permutation matrix, we have $D_o(G, G') = 0$ if and only if $\phi = P$.
2) For any two AGs with the same number of nodes, we have $D_o(G, G') = D_o(G', G)$.
3) We call G a regular AG if there are no repeated eigenvalues in its associate matrix A. Given G and G' with the eigenvalues of A are $\lambda_1 > \lambda_2 > \cdots > \lambda_n$, and the eigenvalues of B are $\mu_1 > \mu_2 > \cdots > \mu_n$, we have $D_o(G, G') = \sum_{i=1}^n (\lambda_i - \mu_i)^2$.
4) Given G, G', and G'' with the same number of nodes, we have $D_o(G, G'') \le D_o(G, G') + D_o(G', G'')$.

Proof 1:

1) It follows directly from (3) and (4) that $D_o(G, G') = D(G, G') = 0$.
2) It follows that

$$\begin{aligned} &\min_\phi \operatorname{tr}\left(\left(\phi A\phi^T - B\right)\left(\phi A\phi^T - B\right)^T\right) \\ &\quad = \min_{\phi_v} \operatorname{tr}\left(\left(\phi_v B\phi_v^T - A\right)\left(\phi_v B\phi_v^T - A\right)^T\right) \end{aligned}$$

with $\phi_v = \phi^T$ being also an orthogonal matrix due to $\phi_v^T\phi_v = \phi_v\phi_v^T = I$.

3) It follows that

$$\begin{aligned} &\operatorname{tr}\left(\left(\phi A\phi^T - B\right)\left(\phi A\phi^T - B\right)^T\right) \\ &\quad = \operatorname{tr}\left(A^2\right) + \operatorname{tr}\left(B^2\right) - 2\operatorname{tr}\left(\phi A\phi^T B\right) \\ &\quad = \sum_{i=1}^n \lambda_i^2 + \sum_{i=1}^n \mu_i^2 - 2\operatorname{tr}\left(\phi A\phi^T B\right). \end{aligned}$$

Furthermore, when the eigenvalues of A are $\lambda_1 > \lambda_2 > \cdots > \lambda_n$, and the eigenvalues of B are $\mu_1 > \mu_2 > \cdots > \mu_n$, it follows from Brockett [1, Theorem 4] that the maximal value of $\operatorname{tr}(\phi A\phi^T B)$ is $\sum_{i=1}^n \lambda_i\mu_i$, which is reached when $\phi = U^T DPV$. Where U and V are orthogonal matrices such that UBU^T and VAV^T are diagonal and P is a permutation matrix, $D = \operatorname{diag}(\alpha_1, \ldots, \alpha_n)$ with $\alpha_i \in \{1, -1\}$. Therefore, we see that the minimal value of $\operatorname{tr}((\phi A\phi^T - B)(\phi A\phi^T - B)^T)$ is $\sum_{i=1}^n (\lambda_i - \mu_i)^2$, which is reached at the orthogonal matrix $\phi = U^T DPV$ since $\phi^T\phi = V^T P^T DUU^T DPV = I$. Hence, $D_o(G, G') = \sum_{i=1}^n (\lambda_i - \mu_i)^2$.

4) Let C be the matrix corresponding to G'' with eigenvalues $\eta_1 > \eta_2 \cdots > \eta_n$. We have

$$\sum_{i=1}^{n}(\lambda_i - \eta_i)^2 = \sum_{i=1}^{n}(\lambda_i - \mu_i + \mu_i - \eta_i)^2 \le \sum_{i=1}^{n}(\lambda_i - \mu_i)^2 + \sum_{i=1}^{n}(\mu_i - \eta_i)^2.$$

Thus $D_o(G, G'') \le D_o(G, G') + D_o(G', G'')$. □

Remarks:

1) The first point shows that $D_o(G, G') = 0$ reaches zero if and only if G and G' are isomorphic, i.e., $\phi = P$. However, if there is noise, we have $\min_\phi D_o(G, G') > 0$ and the corresponding solution ϕ is not guaranteed to be a permutation matrix and also may not be unique.
2) The first two points show that $D_o(G, G')$ is at least a pseudodistance measure on the space of orthogonal $n \times n$ matrices.
3) The third point shows that $D_o(G, G')$ can be calculated by finding the eigenvalues of A and B. The computation of eigenvalues has a time complexity on the order of $O(n^3)$ (e.g., by the Jacobi method). Thus, this matching error can be computed in polynomial time in comparison with the computational complexity that is exponential with n in the case of computing $D(G, G')$ by (3).
4) The first, second, and fourth points together indicate that the matching error function given by (4) is actually a distance measure on the set of regular AGs of n nodes.

We present three ways to compute $D_o(G, G')$. First, a direct way is to perform an eigen-analysis on A and B by Jacobi or one of other conventional methods and then to compute $D_o(G, G') = \sum_{i=1}^{n}(\lambda_i - \mu_i)^2$ by ordering their eigenvalues.

Second, an analog computation on $D_o(G, G')$ can be made by the following differential equation:

$$\begin{aligned} Q &= \phi A \phi^T B \\ \frac{d\phi}{dt} &= -Q\phi + B\phi A \\ D_o(G, G') &= \operatorname{tr}\left(A^2\right) + \operatorname{tr}\left(B^2\right) - 2\operatorname{tr}(Q) \end{aligned} \tag{5}$$

which is more plausibly to be performed by real world applications.

Finally, the above equations in the discrete form are given as

$$\begin{aligned} Q(t) &= \phi(t) A \phi^T(t) B \\ \phi(t+1) &= \phi(t) + \gamma_t [B\phi(t)A - Q(t)\phi(t)] \\ D_o(G, G')(t) &= \operatorname{tr}\left(A^2\right) + \operatorname{tr}\left(B^2\right) - 2\operatorname{tr}(Q(t)) \end{aligned} \tag{6}$$

where $0 < \gamma_t \le 1$ is the learning step size.

The reason for (5) and (6) comes from Brockett's maximization of $\operatorname{tr}(\phi A \phi^T B)$ [1] by

$$\frac{d\phi}{dt} = -\phi A \phi^T B \phi + B \phi A \tag{7}$$

in the space of $n \times n$ orthogonal matrices. Suppose that $UBU^T = D_B = \operatorname{diag}(\mu_1, \ldots, \mu_n)$ and $VAV^T = D_A = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ with U and V being orthogonal. Assuming that $\lambda_1 > \lambda_2 > \cdots > \lambda_n$ and $\mu_1 > \mu_2 > \cdots > \mu_n$, the stable point of (7) must be of the form $\phi = U^T D P V$, where P is an arbitrary permutation matrix and $D = \operatorname{diag}(\alpha_1, \ldots, \alpha_n)$ with $\alpha_i \in \{1, -1\}$. Moreover, $\operatorname{tr}(\phi^T A \phi B)$ has $2^n \cdot n!$ stationary points, exactly 2^n of which are local maxima and are of the form $\phi = U^T D V$, at which points $\operatorname{tr}(\phi^T A \phi B)$ takes on the value $\sum_{i=1}^{n} \lambda_i \mu_i$. These points are the stable points of (7) while the other stationary points are all saddle points.

Now let

$$\begin{aligned} J_o &= \operatorname{tr}\left[\left(\phi A \phi^T - B\right)\left(\phi A \phi^T - B\right)^T\right] \\ &= \sum_{i=1}^{n}\lambda_i^2 + \sum_{i=1}^{n}\mu_i^2 - 2\operatorname{tr}\left(\phi A \phi^T B\right). \end{aligned}$$

The maximization of $\operatorname{tr}(\phi A \phi^T B)$ provides the same solution to the minimization of J_o. This suggests that (5) performs a constrained gradient descent search for J_o. Starting from any initial $\phi(0)$ that is orthogonal [e.g., simply $\phi(0) = I$], the rule will move downhill on the landscape of J_o. This landscape has many stationary points and 2^n local minima points. Theoretically, the search given by (5) may get stuck in one of these stationary points. However, some random fluctuations (e.g., caused by quantizing errors in digital computing) will make the search eventually descent from the stationary points until it finally reaches one of the 2^n local minimum point at which J_o reaches its minimal value $\sum_{i=1}^{n}(\lambda_i - \mu_i)^2$ which is the matching error $D_o(G, G')$.

III. Experiments and Discussion

We demonstrate the proposed PCA approach for fast AGM in a number of experiments. The first set of experiments shows that how the PCA approach works. The second set of experiments compares the PCA approach with three other methods in terms of the computational requirement.

A. Symmetrical Attributed Graph Matching

Experiment I: We have used four randomly generated templates, represented in AGs with the associated matrices listed here.

$$G_1 = \begin{bmatrix} 1.5 & 1.0 & 3.0 & 0.4 & 2.0 \\ 1.0 & 2.0 & -0.3 & 1.0 & 0.0 \\ 3.0 & -0.3 & 1.0 & 2.0 & -0.5 \\ 0.4 & 1.0 & 2.0 & 4.0 & 0.2 \\ 2.0 & 0.0 & -0.5 & 0.2 & 1.0 \end{bmatrix}$$

and

$$G_2 = \begin{bmatrix} 2.5 & 2.0 & 0.3 & 0.4 & 1.0 \\ 2.0 & 2.0 & -0.3 & 1.0 & 0.1 \\ 0.3 & -0.3 & 1.0 & 2.0 & 0.5 \\ 0.4 & 1.0 & 2.0 & 1.0 & -0.2 \\ 1.0 & 0.1 & 0.5 & -0.2 & 2.0 \end{bmatrix}.$$

$$G_3 = \begin{bmatrix} -2.5 & -2.0 & 0.3 & 0.4 & 1.0 \\ -2.0 & 0.5 & -0.3 & 0.2 & 0.1 \\ 0.3 & -0.3 & 3.0 & 2.0 & -1.5 \\ 0.4 & 0.2 & 2.0 & -1.0 & -1.2 \\ 1.0 & 0.1 & -1.5 & -1.2 & 1.0 \end{bmatrix}$$

and

$$G_4 = \begin{bmatrix} 1.5 & 0.1 & 3.0 & 0.4 & -2.0 \\ 0.1 & 2.0 & 1.3 & 1.0 & -0.4 \\ 3.0 & 1.3 & 1.0 & 1.1 & -0.5 \\ 0.4 & 1.0 & 1.1 & 4.0 & 0.2 \\ -2.0 & -0.4 & -0.5 & 0.2 & 1.0 \end{bmatrix}.$$

Now, a query represented by G comes, which was obtained from G_1 by a permutation P

$$G = PG_1P^T = \begin{bmatrix} 2.0 & 1.0 & -0.3 & 0.0 & 1.0 \\ 1.0 & 4.0 & 2.0 & 0.2 & 0.4 \\ -0.3 & 2.0 & 1.0 & -0.5 & 3.0 \\ 0 & 0.2 & -0.5 & 1.0 & 2.0 \\ 1.0 & 0.4 & 3.0 & 2.0 & 1.5 \end{bmatrix}$$

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Using (4), we have $D_o(G, G_1) = 0$, $D_o(G, G_2) = 5.4429$, $D_o(G, G_3) = 16.4321$, $D_o(G, G_4) = 1.5032$. Here G is correctly matched into its isomorphic AG G_1.

Experiment II: For practical use, the input pattern is usually contaminated by noise. Thus G will not be exactly isomorphic to its template G_1. To observe the influence of noise, we added uniformly distributed random noise to each node and edge of G to obtain a noise-contaminated input G' as

$$G' = G + N_1 = \begin{bmatrix} 2.4615 & 1.1865 & -0.1811 & 0.4866 & 1.3754 \\ 1.1865 & 4.2779 & 2.1437 & 0.4709 & 0.6113 \\ -0.1811 & 2.1437 & 1.2206 & -0.2250 & 3.1381 \\ 0.4866 & 0.4709 & -0.2250 & 1.6436 & 2.4122 \\ 1.3754 & 0.6113 & 3.1381 & 2.4122 & 1.8446 \end{bmatrix}$$

$$N_1 = \begin{bmatrix} 0.4615 & 0.1865 & 0.1189 & 0.4866 & 0.3754 \\ 0.1865 & 0.2779 & 0.1437 & 0.2709 & 0.2113 \\ 0.1189 & 0.1437 & 0.2206 & 0.2750 & 0.1381 \\ 0.4866 & 0.2709 & 0.2750 & 0.6436 & 0.4122 \\ 0.3754 & 0.2113 & 0.1381 & 0.4122 & 0.3446 \end{bmatrix}.$$

Comparing N_1 with G, we can see that the noise added is not small in magnitude when compared to G. We have $D_o(G', G_1) = 1.4102$, $D_o(G', G_2) = 9.7695$, $D_o(G', G_3) = 23.9471$, and $D_o(G', G_4) = 3.1846$. Here, $D_o(G', G_1) = 1.4102$ is still the minimum one. Its value is still less than $D_o(G', G_4) = 3.1846$ of the competing template G_4.

Experiment III: This experiment tries to demonstrate the PCA approach in a large controlled data set. We randomly generated 10 000 AGs of size 5. The value in each attribute was generated with $\mathcal{U}(-1, 1)$. A randomly chosen AG was selected as the target among all AGs. To make the problem more difficult, we performed two operations on each of the AGs. First we added noise with a uniform distribution of $\mathcal{U}(-0.05, 0.05)$ to each AG in the data set. Second, we also permuted each AG in the data set to make certain that there is no straight forward match for the chosen AG. We then performed this test over 500 trials with random initial condition in each trial. The ranking of the AG matching result is shown in Table I. The result demonstrates that the PCA approach is robust against noise as shown previously.

We also tried in the case when the noise is skewed to the positive region, i.e., $\mathcal{U}(0, 0.5)$. The results are shown in Table II. Fig. 2 illustrates the ranking distribution of the result.

In this particular experiment, the PCA approach did not perform satisfactorily. Nonetheless, the worst result was ranked at 105th place occurring only once in 500 trials. Moreover, this worst case still placed the target at around the top 1% (105 out of 10 000) of the nodes. Hence, the experiment still illustrated the robustness of the proposed PCA approach's approximation power.

TABLE I
FREQUENCY DISTRIBUTION OF THE TOP THREE RANKING FOR CORRECTLY MATCHED ATTRIBUTED GRAPH OF SIZE 5 WITH NOISE ADDED

	1st Ranking	Second Ranking	Third Ranking
Frequency Count	473	25	2
Percentage (%)	94.6	5.0	0.4

TABLE II
FREQUENCY DISTRIBUTION OF THE TOP TEN RANKING FOR CORRECTLY MATCHED ATTRIBUTED GRAPH OF SIZE 5 WITH ONLY POSITIVE NOISE ADDED

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Frequency Count	117	71	49	42	37	24	13	14	23	10
Percentage (%)	23.4	14.2	9.8	8.4	7.4	4.8	2.6	2.8	4.6	2.0

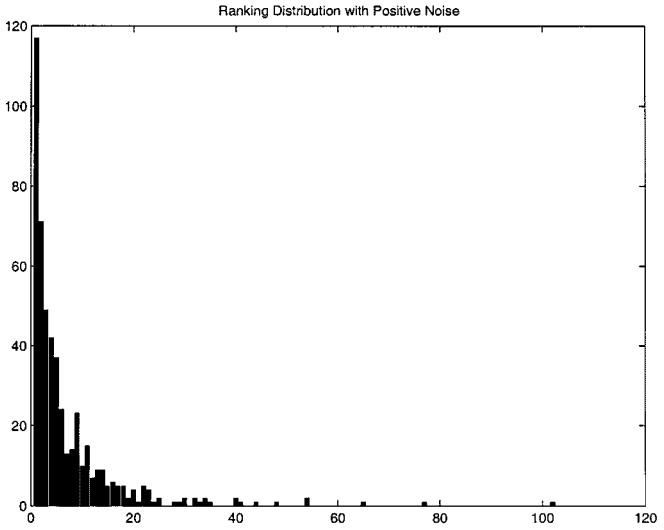


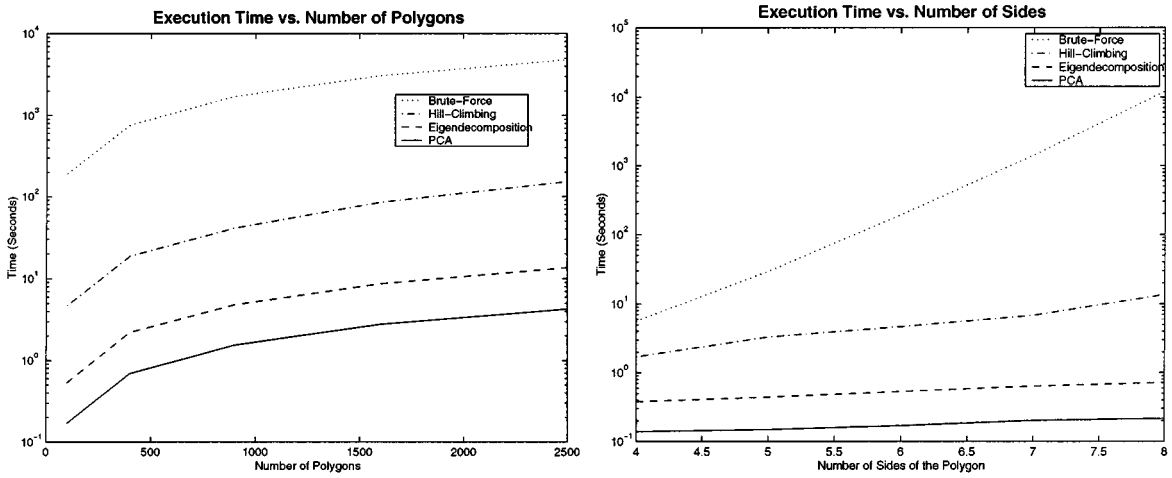
Fig. 2. Ranking frequency distribution chart.

B. Comparison With Other Algorithms

In this section, we compared our proposed PCA algorithm for similar shape matching with two other algorithms: 1) brute-force method and 2) eigendecomposition method [12]. We varied the sides of the polygon as $\{4, 6, 8\}$ and the number of polygons as $\{100, 400, 900\}$ to test the three algorithms. These experiments were all conducted on Sun's Ultra 1 workstation using Matlab 5.3 under the Solaris 2.6 environment.

The most obvious AGM solution is the brute-force method where each permuted graph is tested to locate the one having the smallest matching error. Although this method finds the optimal answer by enumerating all the possible permutations of the graph, this type of solution has a computational complexity of $O(n!)$, where n is the number of nodes in each graph. This is impractical and also unacceptable for large and arbitrary pairs of graphs.

The eigendecomposition approach seeks an approximate solution to the weighted graph matching problem for undirected graphs [12]. It



(a) (b)

Fig. 3. (a) Execution time versus number of polygons plot illustrates the computational time required for each of the four algorithms to perform for various polygons with six sides. (b) Execution time versus number of sides plot illustrates the computational time required for each of the four algorithms to perform for various sides of 100 polygons.

is an analytic approach based on the eigendecompositions of the adjacency matrices. The main difference between our proposed approach and the eigendecomposition's is that our method calculates the score directly from the eigenvalues while their method tries to find additional information to approximate the best permutation matrix for the best match.

Computational Complexity Experiment: Fig. 3 illustrates the running time of the three algorithms by varying the number of sides and the number of polygons in shape similarity matching tasks. Both (a) and (b) clearly demonstrate the fast execution speed of the proposed PCA method. Furthermore, Tables III and IV show the speed up factor for the tested algorithms using the proposed PCA algorithm as the baseline when we varied the number of sides of the polygon and also the number of polygons processed, respectively. Using the proposed PCA algorithm as the reference, the brute-force and the eigendecomposition algorithms, on the average, took approximately 1089 and 2.80 times more, respectively, as a function of the number of polygons processed.

Matching Accuracy Experiment: Fig. 4 shows the result of a typical accuracy experiment over many trials comparing the eigendecomposition algorithm and the PCA algorithm with the permutation matching result as the reference. The matching task set up in this experiment is similar to the previous experiment. However, this experiment focuses on the accuracy of the ranking result. Fig. 4(a)–(c) show the normalized and unweighted histogram 4-, 6-, and 8-sided 100 polygon ranking histogram results, respectively. Fig. 4(d)–(f) and (g)–(i) show the 400 and 900 polygon matching experimental results, respectively. On the other hand, the three columns show the 4-, 6-, and 8-sided polygon matching, respectively.

Given that the matching returns a reference ranking from the Brute-force method as $R_1 = (r_1, r_2, \ldots, r_n)$ and R_2, R_3, represent the ranking returned by the eigendecomposition and the PCA method, respectively. This set of experiments measures the accuracy of the algorithms by examining the histogram of the shifted position with respect to the reference ranking, i.e., the histogram of $|R_2 - R_1|$, $|R_3 - R_1|$. For example, let $R_1 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ and $R_2 = (4, 2, 1, 5, 3, 6, 7, 8, 9, 10)$, then the position shift for the list is $|R_1 - R_2| = (3, 0, 2, 1, 2, 0, 0, 0, 0)$ and the bin values in the histogram will contain $(50, 10, 20, 10, 0, 0, 0, 0, 0, 0, 0)$. We treat the individual ranking of the element equally so their relative position is important when they are near the front or near the end of the ranking.

TABLE III
SPEED UP FACTOR AMONG TESTED ALGORITHMS BY VARYING THE SIDES WITH 100 POLYGONS

	4-sided	6-sided	8-sided
Brute-force	38.25	1,083.53	55,069.95
Eigendecomposition	2.57	2.80	3.11

TABLE IV
SPEED UP FACTOR AMONG TESTED ALGORITHMS BY VARYING THE NUMBER OF POLYGONS PROCESSED WITH 100 POLYGONS

	100	400	900
Brute-force	1,083.53	1,091.53	1,092.81
Eigendecomposition	2.80	2.80	2.84

The brute-force method proved to be the most accurate, but it also took the longest time to find the optimal match since it enumerates all possible combinations of the match. Hence it is chosen as the reference for the accuracy ranking experiment. The accuracy of the eigendecomposition method is inferior to the PCA method as shown in Fig. 4(b), (c), (d), (e), (f), (h), and (i). The eigendecomposition method's accuracy dropped when the number of sides and the number polygons were increased. Not only the PCA method is faster than the eigendecomposition method since there is on need to approximate the permutation matrix, but also provides more the accurate results than that by the eigendecomposition method.

IV. FINAL REMARKS

There are many algorithms for solving the graph matching or isomorphic problems. Usually, these algorithms have a computing com-

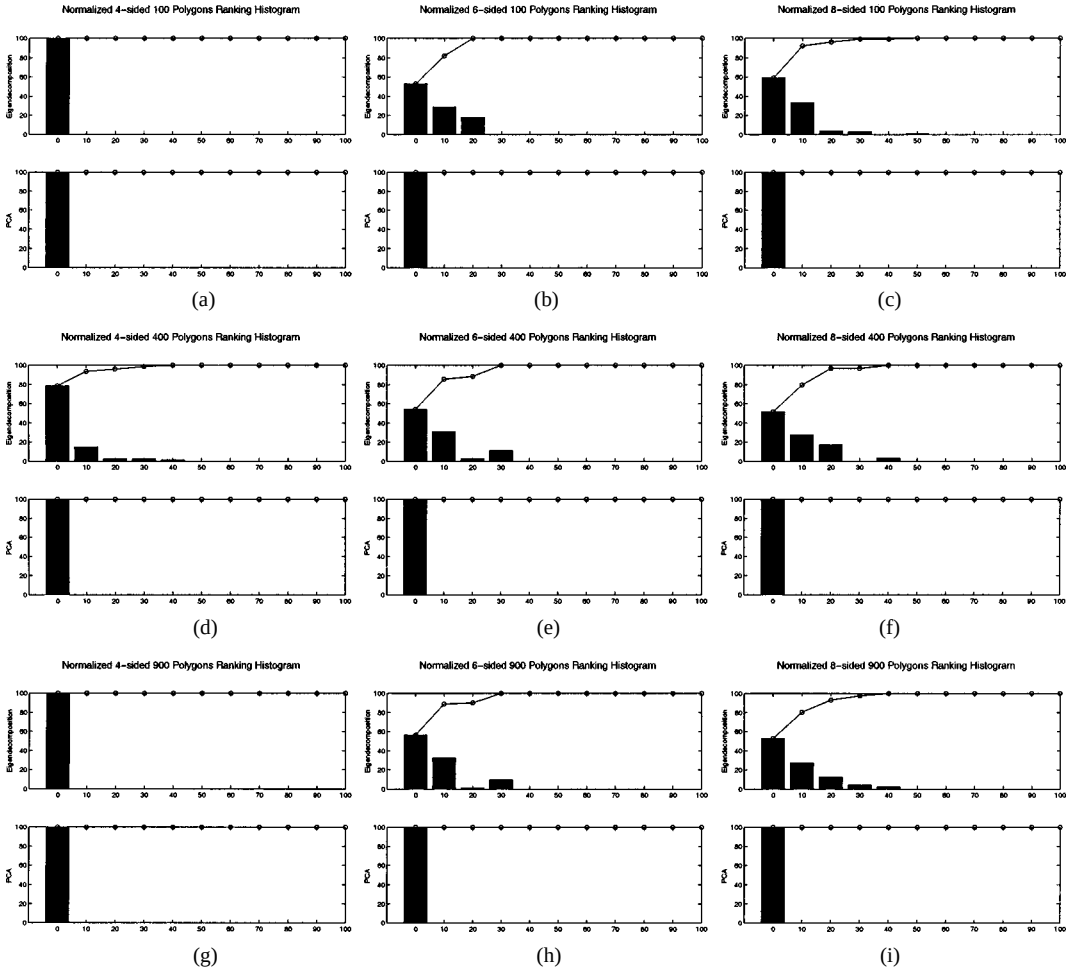


Fig. 4. Polygon matching accuracy result.

plexity of an exponential order of n [4]. Our proposed approach does not seek for exact isomorphic match of two graphs. Rather, we use a more relaxed constraint to find plausible candidates through the fast computable but approximate matching error in help of AG's eigenvalues. There are several ways to compute the eigenvalues of a square symmetric matrix. Many good algorithms can be found in [10]. Typical examples are the Householder transformation method and the QL method [7]. These algorithms are polynomial in nature, usually $O(n^2)$ and no more than $O(n^3)$, which is a considerable reduction in the execution cost.

As demonstrated by the previous experiments, the proposed PCA approach has a promising potential as a fast retrieval technique in large scale databases which processes images or other structural patterns efficiently. Moreover, it also has a favorable feature that it follows from (4) that any rotation of the matrix A will not affect the matching error and thus the approach is planar rotationally invariant.

ACKNOWLEDGMENT

This work was performed in part at the Harvard Robotics Laboratory, Harvard University, when L. Xu worked with Professor A. Yuille. L. Xu would like to thank Professor Yuille for his insightful comments on this work.

REFERENCES

[1] R. W. Brockett, "Least square matching problems," *Lin. Algebra Applicat.*, vol. 122–124, pp. 761–777, 1989.

[2] W. J. Christmas, J. Kittler, and M. Petrou, "Structural matching in computer vision using probabilistic relaxation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 749–764, Aug. 1995.

[3] E. Eshera and K. S. Fu, "A graph distance measure for image analysis," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-14, pp. 396–408, Mar. 1984.

[4] L. R. Foulds, *Graph Theory Applications*. New York: Springer-Verlag, 1992.

[5] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 377–388, Apr. 1996.

[6] H. Kälviäinen and E. Oja, "Comparisons of attributed graph matching algorithms for computer vision," Research Rep. 19, Lappeenranta Univ. Technol., Dept. Inform. Technol., Lappeenranta, Finland, 1990.

[7] J. H. Mathews, *Numerical Methods for Mathematics, Science, and Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[8] E. Oja, "A simplified neuron model as a principal component analyzer," *J. Math. Biol.*, vol. 16, pp. 267–273, 1982.

[9] E. G. M. Petrakis and C. Faloutsos, "Similarity searching in medical image databases," *IEEE Trans. Knowl. Data Eng.*, vol. 9, pp. 435–447, May/June 1997.

[10] W. H. Press *et al.*, *Numerical Recipes in C*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[11] P. N. Suganthan and H. Yan, "Recognition of handprinted Chinese characters by constrained graph matching," *Image Vis. Comput.*, vol. 16, no. 3, pp. 191–201, Mar. 1998.

[12] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, pp. 695–703, May 1988.

[13] L. Xu and A. L. Yuille, "Robust principal component analysis by self-organizing rules based on statistical physics approach," *IEEE Trans. Neural Networks*, vol. 6, pp. 131–143, Feb. 1995.

[14] L. Xu, "Least mean square error reconstruction for self-organizing neural-nets," *Neural Networks*, vol. 6, pp. 627–648, 1993.