Learning Templates from Fuzzy Examples in Structural Pattern Recognition

Kwok-Ping Chan, *Member, IEEE*

***Abstract*— Fuzzy-Attribute Graph (*FAG*) was proposed to handle fuzziness in the pattern primitives in structural pattern recognition. *FAG* has the advantage that we can combine several possible definition into a single template. However, the template require a human expert to define. In this paper, we propose an algorithm that can, from a number of fuzzy instances, find a template that can be matched to the patterns by the original matching metric.**

I. Introduction

Fuzzy set theory was first introduced by Professor L. A. Zadeh in 1965 [1] and is used as a formal mathematical tools to investigate problems pertaining to uncertainty, ambiguity and vagueness. It can be applied on concepts that has no exact boundary between membership and nonmembership and the change is gradual rather than abrupt.

There are two approaches in pattern recognition – the *decision-theoretic* and *structural* approach. In decision-theoretic approach, the pattern is represented by a series of "features". Somehow the object is transformed to a feature vector, and the classification is done in the feature space, using either a discriminant function, or *Bayes classifier*. In structural approach, the object is represented by its structures. Tsai and Fu [2] proposed to use attributed graph in structural pattern recognition. The graph representation provide much richer structural content than grammatical approach, where a string is assumed. Although high-dimensional grammar can be used, this will be very complicated. Another possibility is to use a *Guarded Grammar* [3] which can fulfill most function of the high dimensional grammar while retaining simplicity.

It is the author's belief that human perception is achieved by finding recognizable subpatterns in a complex scene and then compose a meaningful picture from these subpatterns. During this matching process, we have some *definitions* of the subpatterns, which by itself, is not fuzzy. However, each instance of this definition is fuzzy. Hence we are matching a non-fuzzy template with fuzzy objects.

The attributed graph proposed do not contain any fuzzy information. Hence the author proposed an extension, which is called *Fuzzy-Attribute Graph (FAG)* [4]. Also, some nice properties of FAG has been derived. In this paper, we further present an algorithm for learning templates from several instances of FAGs, and hence present a complete framework that can be used in structural pattern recognition.

The author is with the Department of Computer Science, University of Hong Kong, Pokfulam Road, Hong Kong.

II. Fuzzy-Attribute Graph

Attributed graph was introduced by Tsai and Fu [2] for pattern analysis. It gives a more straightforward representation of structural patterns. The vertices of the graph represent pattern primitives describing the pattern while the arcs are the relations between these primitives. However the pattern often possess properties that are fuzzy in nature and it has been extended to include fuzzy informations into the attributes.

In a fuzzy-attribute graph, each vertex may have attributes from the set $Z = \{z_i | i = 1, \ldots, I\}$. For each attribute z_i, it may take values from $S_i = \{s_{ij} | j = 1, \ldots, J_i\}$. The set of all possible attribute-value pair is $\tilde{L}_v = \{(z_i, \tilde{A}_{S_i}) | i = 1, \ldots, I\}$ where $\tilde{A}_{S_i}$ is a fuzzy set on the attribute-value set S_i. A valid pattern primitive is just a subset of $\tilde{L}_v$ in which each attribute appears only once, and $\tilde{\Pi}$ represent the set of all those valid pattern primitives.

Similarly, each arc may have attributes from the set $F = \{f_i | i = 1, \ldots, I'\}$ in which each f_i may take values from $T_i = \{t_{ij} | j = 1, \ldots, J_i'\}$. $\tilde{L}_\alpha = \{(f_i, \tilde{B}_{T_i}) | i = 1, \ldots, I'\}$ denotes the set of all possible relational-attribute value pair, where $\tilde{B}_{T_i}$ is a fuzzy set on the relational attribute-value set T_i. A valid relation is just a subset of $\tilde{L}_\alpha$ in which each attribute appears only once. The set of all valid relation is denoted $\tilde{\Theta}$.

Definition 1 A *Fuzzy-Attribute Graph (FAG)*, $\tilde{G}$ over $\tilde{L} = (\tilde{L}_v, \tilde{L}_\alpha)$ with an underlying graph structure $H = (N, E)$ is defined to be an ordered pair

0-7803-1896-X/94 $4.00 ©1994 IEEE

$(\tilde{V}, \tilde{A})$, where $\tilde{V} = (N, \tilde{\sigma})$ is called a fuzzy vertex set and $\tilde{A} = (E, \tilde{\delta})$ is called a fuzzy arc set and

$\tilde{\sigma}$: $N \rightarrow \tilde{\Pi}$ is called a fuzzy vertex interpreter,
$\tilde{\delta}$: $E \rightarrow \tilde{\Theta}$ is called a fuzzy arc interpreter.

This definition also applies when there are non-fuzzy attributes, since a crisp set (non-fuzzy) can always be represented as a special case of fuzzy set.

Example 1 When we use *FAG* to represent Chinese characters, each node of the graph will represent a stroke, and the relation between the nodes are represented by the edges. Each stroke may have two different attributes *STROKE-TYPE* and *LENGTH*. The attributes for relations between the nodes are *JOINT-TYPE*, *VERT-REL* and *HORI-REL*. The values that can be taken in each of the attributes are,

$$\begin{aligned}
\{\textit{STROKE-TYPE}\} &= \{\textit{Vertical, Horizontal, Slant45, Slant135}\} \\
\{\textit{LENGTH}\} &= \{\textit{Long, Short}\} \\
\{\textit{JOINT-TYPE}\} &= \{\textit{T-from, T-into, Ht, Cross, Parallel}\} \\
\{\textit{VERT-REL}\} &= \{\textit{On-top-of, Below-of, No-vert-relate}\} \\
\{\textit{HORI-REL}\} &= \{\textit{Left-of, Right-of, No-hori-relate}\}
\end{aligned}$$

Using these attributes and values, we can construct a *FAG* to represent a Chinese character. A typical fuzzy vertex set and fuzzy arc set is given below,

$$\begin{aligned}
\tilde{\pi}_1 = \{&(\textit{STROKE-TYPE}, \{0.7/\textit{Vertical}, 0.85/\textit{Slant45}, 0.01/\textit{Horizontal}, 0/\textit{Slant135}\}), \\
&(\textit{LENGTH}, \{0.6/\textit{Long}, 0/\textit{Short}\})\}
\end{aligned}$$

and,

$$\begin{aligned}
\tilde{\theta}_1 = \{&(\textit{JOINT-TYPE}, \{0.7/\textit{T-from}, 0.65/\textit{Cross}, 0/\textit{T-into}, 0/\textit{Ht}, 0/\textit{Parallel}\}), (\textit{VERT-REL}, \\
&\{0.9/\textit{On-top-of}, 0/\textit{Below-of}, 0.25/\textit{No-vert-relate}\}), (\textit{HORI-REL}, \\
&\{0.2/\textit{Left-of}, 0.4/\textit{Right-of}, 0.77/\textit{No-hori-relate}\})\}.
\end{aligned}$$

The *FAG* data structure is general enough to be applied to most objects in typical recognition problems. The fuzzy attributes are used to handle those fuzzy properties as well as non-fuzzy enumerable attributes. For those attributes with continuous values, such as height of a man, we can easily abstracted out the fuzzy concepts such as ***tall*** and ***short***, and represent them in fuzzy sets. This is what is performed by human being.

III. A Recognition Framework based on *FAG*

In this section, we will describe a framework of general pattern recognition system based on the *FAG*. Before going further, the author would like to reiterate his view that human perception is based on recognizing some identifiable parts from a complex scene, and then based on these identified parts, reconstruct the whole picture. Also, the ***ideal*** objects, or the ***definition*** of the object is not fuzzy at all. It is an ***instance*** of this definition that is fuzzy. The whole approach is based on these views.

The following is a generalization of the approach from the author's original application on Chinese character recognition [4] First of all, we define a set of small subpatterns (similar to syntactic pattern recognition). The definition of these subpattern is represented in a Hard (non-fuzzy) *FAG* or *HFAG*, which is a *FAG* which all fuzzy set crisped, i.e. with membership of either 0 or 1. These are used as the matching templates. The pattern to be matched is represented as *FAG*. The fuzzy graph is then matched against the non-fuzzy graph to determine whether there is a matching found. One problem that immediately follows is the way to define a matching.

To derive the matching formula, we first notice that the *FAG* representation has some meaning behind it. Actually, the membership value of a fuzzy set can be interpreted as the compatibility of a member and its properties. Consider the following examples of a fuzzy vertex set,

$$\tilde{A}_{1S_i} = \left\{ \mu_{\tilde{A}_{1S_i}}(s_{ij})/s_{ij} | s_{ij} \in S_i \right\}$$

where S_i is the i-th attribute and s_{ij} are the attribute values. This can be interpreted as

> vertex v_i possesses property s_{ij} with truth value $\mu_{\tilde{A}_{1S_i}}(s_{ij})$.

To match between the definition and the pattern, one would like to check the truth value of the following statement:

> template $\tilde{H}$ possess property s_{ij} *AND* pattern $\tilde{G}$ possess property s_{ij}

and this can be obtained by the following fuzzy expression

$$\mu_{\tilde{H}S_i}(s_{ij}) \wedge \mu_{\tilde{G}S_i}(s_{ij})$$

Now, when we consider over all values for an attribute, we would like to take *disjunction* over all values, for the attribute S_i,

> template $\tilde{H}$ possess property s_{i1} *AND* pattern $\tilde{G}$ possess property s_{i1} *OR*
> template $\tilde{H}$ possess property s_{i2} *AND* pattern $\tilde{G}$ possess property s_{i2} *OR* ...

This is for one attribute and we can repeat this for all attributes by taking *conjunction* over them. Hence we have the following definition:

Definition 2 Let $\tilde{G}_1$ and $\tilde{G}_2$ be two *FAGs* with the same underlying graph $H_1 = (N_1, E_1)$ of $\tilde{G}_1$ being monomorphic to the underlying graph H_2 of $\tilde{G}_2$. The ***degree of matching*** γ is defined as

$$\gamma(\tilde{G}_1, \tilde{G}_2) = \bigwedge_{i \in N_1} \alpha(i, h(i)) \bigwedge_{(i,j) \in E_1} \beta(e_1(i,j), e_2(h(i), h(j)))$$

where $h(i)$ is the vertex in $\tilde{G}_2$ that matches vertex i in $\tilde{G}_1$, and $e_k(i,j)$ is the arc joining vertices i and j in $\tilde{G}_k$.
The value $\alpha(i,j)$ is the matching between vertex i and vertex j and is obtained as

$$\alpha(i,j) = \bigwedge_{m=1}^{I} \bigvee_{n=1}^{J_I} \left\{ \left(\mu_{\tilde{A}_{1S_m}}(s_{mn}) \wedge \mu_{\tilde{A}_{2S_m}}(s_{mn}) \right) \right\}$$

where $\tilde{A}_{kS_i}$ is the fuzzy set of the attribute S_i of graph k, $k = 1, 2$.
Similarly, the value $\beta(e_i, e_j)$ is the matching between arc e_i and arc e_j and is obtained as

$$\beta(e_1, e_2) = \bigwedge_{m=1}^{I'} \bigvee_{n=1}^{J'_m} \left\{ \left(\mu_{\tilde{B}_{1T_m}}(t_{mn}) \wedge \mu_{\tilde{B}_{2T_m}}(t_{mn}) \right) \right\}$$

where $\tilde{B}_{kT_i}$ is the fuzzy set of the attribute T_i of graph k, $k = 1, 2$.

The degree of matching so defined has its physical meaning which corresponds to the logical expression above. This logical expression is both intuitive and understandable and in fact it has some very nice property that makes it very useful in pattern recognition.

IV. Learning from examples of *FAGs*

When using the above described framework, one can notice that the template should be defined by the user. In the previous section, we have discussed how matching can be achieved between a fuzzy instance and the non-fuzzy definition. This definition was provided by human experts. However, we would like to have a mechanism such that the definition can be derived from a set of fuzzy examples. In this section, we will try to describe such an algorithm that can learn the template by showing the system examples. This is not a trivial "learning from example" problem as the examples is fuzzy in nature, and the result should be λ-monomorphic with all the fuzzy examples.

Theorem 1 Given a *FAG* $\tilde{G}$, and a *HFAG* $\tilde{H}$, $\tilde{G}$ is λ-monomorphic with $\tilde{H}$ iff for each nodal attribute z_i, $\exists s_{ij} \in S_i$ such that $\mu_{\tilde{G}_{S_i}}(s_{ij}) \bigwedge \mu_{\tilde{H}_{S_i}}(s_{ij}) \geq \lambda$ and for each relational attribute f_i, $\exists t_{ij} \in T_i$ such that $\mu_{\tilde{G}_{T_i}}(t_{ij}) \bigwedge \mu_{\tilde{H}_{T_i}}(t_{ij}) \geq \lambda$.

The proof follows directly from the definition of λ-monomorphic.

This theorem, although simple, allows us to consider each attribute (whether it is nodal or relational) individually. Hence the following discussion will be concentrated on the learning of a single attribute, and the result can then be combine together back to form the template.

Definition 3 Consider a set $X = \{x_i | i = 1, \ldots, n\}$. The extended product $\odot$ between two fuzzy subset $\tilde{Y}_1$ and $\tilde{Y}_2$ of X is defined as

$$\tilde{Y}_1 \odot \tilde{Y}_2 = \bigvee_{i=1}^{n} \left(\mu_{\tilde{Y}_1}(x_i) \wedge \mu_{\tilde{Y}_2}(x_i) \right)$$

Definition 4 $\tilde{Y}_1$ and $\tilde{Y}_2$ are said to be λ-matched if $\tilde{Y}_1 \odot \tilde{Y}_2 \geq \lambda$.

With the help of theorem 1 we can now rephrase the problem to a simpler form: Consider a set $X = \{x_i | i = 1, \ldots, n\}$. Given a set of fuzzy subset of X, $\tilde{Y}_i, i = 1, \ldots, m$, we are going to find the smallest set $Z \subseteq X$ (i.e. $|Z|$ is smallest) such that $\tilde{Z}$, the fuzzy set representation of Z, is λ-matched with $\tilde{Y}_i, i = 1, \ldots, m$. This is for one attribute and we repeat this for all nodal and relational attributes.

Definition 5 Given a universe $X = \{x_i | i = 1, \ldots, n\}$. A *polynomial form* P is a summation of the form

$$P = \sum_{k=1}^{m} a_k X_k$$

where X_k is a product of any number of x_i's, and a_k are constants.

For each fuzzy set $\tilde{Y}$ on the universe X, we can rewrite $\tilde{Y}$ in the polynomial form

$$\sum_{i=1}^{n} \mu_{\tilde{Y}}(x_i) x_i$$

where each X_i's is a singleton.

Definition 6 The *f-product* between two polynomial form P_1 and P_2 is another polynomial form $Q = P_1 \otimes P_2$ such that if $P_1 = \sum_{k=1}^{m} a_k X_k$ and $P_2 = \sum_{i=1}^{m'} a'_k X'_k$, then

$$Q = \sum_{i=1}^{m}\sum_{j=1}^{m'} \min(a_k, a'_k) X_i X_j$$

with the convention that $x_i x_i = x_i$, and

$$a_1 X + a_2 X = max(a_1, a_2) X$$

Example 2

$$\begin{aligned} & (0.5x_1 + 0.3x_2 + 0.7x_3) \otimes (0.3x_1 + 0.2x_2 + 0.8x_3) \\ = \; & (0.3x_1 + 0.2x_2 + 0.7x_3) + (0.2x_1x_2 + 0.3x_1x_2) \\ & + (0.5x_1x_3 + 0.3x_1x_3) + (0.3x_2x_3 + 0.2x_2x_3) \\ = \; & 0.3x_1 + 0.2x_2 + 0.7x_3 + 0.3x_1x_2 + 0.5x_1x_3 + 0.3x_2x_3 \end{aligned}$$

Theorem 2 Let $\tilde{Y}_1$ and $\tilde{Y}_2$ be two fuzzy sets with polynomial forms P_1 and P_2 respectively, and Q be the *f-product*

$$Q = P_1 \otimes P_2 = \sum_{k=1}^{m} a_k X_k$$

Then the *HFAG* $\tilde{H}_k$ constructed form the k-th term $a_k X_k$ such that

$$\mu_{\tilde{H}_k}(x_i) = \begin{cases} 1 & \text{if } x_i \text{ is in } X_k \\ 0 & \text{otherwise} \end{cases}$$

is a_k-matched with both $\tilde{Y}_1$ and $\tilde{Y}_2$.

Proof: Note that P_1 and P_2, the polynomial form of $\tilde{Y}_1$ and $\tilde{Y}_2$ contain only single x_i's in the summation. Let $P_1 = \sum_{i=1}^{n} a_i x_i$ and $P_2 = \sum_{i=1}^{n} a'_i x_i$. Then for a term containing $x_i x_j$ in the product (note that $x_1 = x_1 x_1$) with a coefficient b_{ij},

$$b_{ij} = \max(\min(a_i, a'_j), \min(a_j, a'_i))$$

Hence,

$$\begin{aligned} & (\mu_{\tilde{H}_k}(x_i) \wedge \mu_{\tilde{Y}_1}(x_i)) \bigvee (\mu_{\tilde{H}_k}(x_j) \wedge \mu_{\tilde{Y}_1}(x_j)) \\ = \; & \mu_{\tilde{Y}_1}(x_i) \vee \mu_{\tilde{Y}_1}(x_j) \\ = \; & \max(a_i, a_j) \geq b_i j \end{aligned}$$

The same argument applies to $\tilde{Y}_2$. Hence the result.

It can easily be seen that the above argument can be extended to more than two $\tilde{Y}$'s. From theorem 1 we can separate the learning into individual attributes for nodes and relations. Theorem 2 allow us to find the attribute for individual attribute. It can be further noted that we can delete intermediate terms which has a value $< \lambda$.

Definition 7 The λ-cut of a polynomial form $P = \sum_{i=1}^{n} a_i X_i$ is another polynomial form $P_\lambda = \sum_{i=1}^{n} b_i X_i$ such that

$$b_i = \begin{cases} a_i & \text{if } a_i \geq \lambda \\ 0 & \text{otherwise} \end{cases}$$

Theorem 3 The λ-cut operation is distributive with respect to *f-product*, i.e.

$$P_{1\lambda} \otimes P_{2\lambda} = (P_1 \otimes P_2)_\lambda$$

From definition 6, it can be seen that any term that contains a coefficient $< \lambda$ will produce another term which is $< \lambda$ in the summation. This will be discarded after the λ-cut operation. Hence we can ignore any terms that contain coefficients $< \lambda$.

With this theorem, we can delete all intermediate terms with coefficient $< \lambda$ during the multiplication process. This will prune off all those unpromising terms. Taking this into account, we have the following algorithm for finding the "smallest" templates.

Algorithm

Repeat the following steps for each nodal and relational attributes.

(1) Represent the fuzzy set of the attribute of all training samples in a polynomial form, namely, $P_1, P_2, \cdots, P_n$.
(2) Assign λ-cut of P_1 to Q.
(3) Find the *f-product* $Q \leftarrow \lambda$-cut of $(Q \otimes P_2)$
(4) Repeat step 3 until all input samples exhausted
(5) Find the term with smallest number of x_i's and with coefficient $\geq \lambda$.

Finally, construct a *HFAG* from these term.

The complexity of this algorithm is $O(nm2^m)$ for each attribute where n is the number of examples and m is the number of attribute-values within the attribute. For each multiplication, one of them is originally a *FAG* and its polynomial form contain only m terms. The other have at most 2^m terms. Hence each step take no more than $m2^m$. Also note that the value of m is fixed and is usually very small, for example ≤ 10. In our previous example, the *STROKE-TYPE* attribute actually contain only 4 values, i.e. $m = 4$ and $m2^m = 64$. Hence the algorithm is essentially linear with respect to the number of training examples. Finding the smallest term is also constant (in this case $O(2^m)$).

V. An Illustrative Example

Chinese character recognition is a typical example that require fuzziness in the description of the pattern primitives. Each stroke and their relations can

id	*Horizontal*	*Vertical*	*Slant45*	*Slant135*
1	0.85	0.00	0.22	0.00
2	0.83	0.00	0.19	0.00
3	0.37	0.00	0.00	0.49
4	0.91	0.00	0.13	0.00
5	0.84	0.00	0.21	0.00
6	0.98	0.00	0.39	0.00
7	1.00	0.00	0.03	0.00
8	0.94	0.00	0.31	0.00

(a) Stroke 1

id	*Horizontal*	*Vertical*	*Slant45*	*Slant135*
1	0.00	1.00	0.03	0.03
2	0.00	1.00	0.03	0.03
3	0.00	0.93	0.09	0.00
4	0.00	0.43	0.32	0.00
5	0.00	0.99	0.05	0.02
6	0.00	1.00	0.03	0.03
7	0.00	0.95	0.08	0.00
8	0.10	0.30	0.39	0.00

(b) Stroke 2

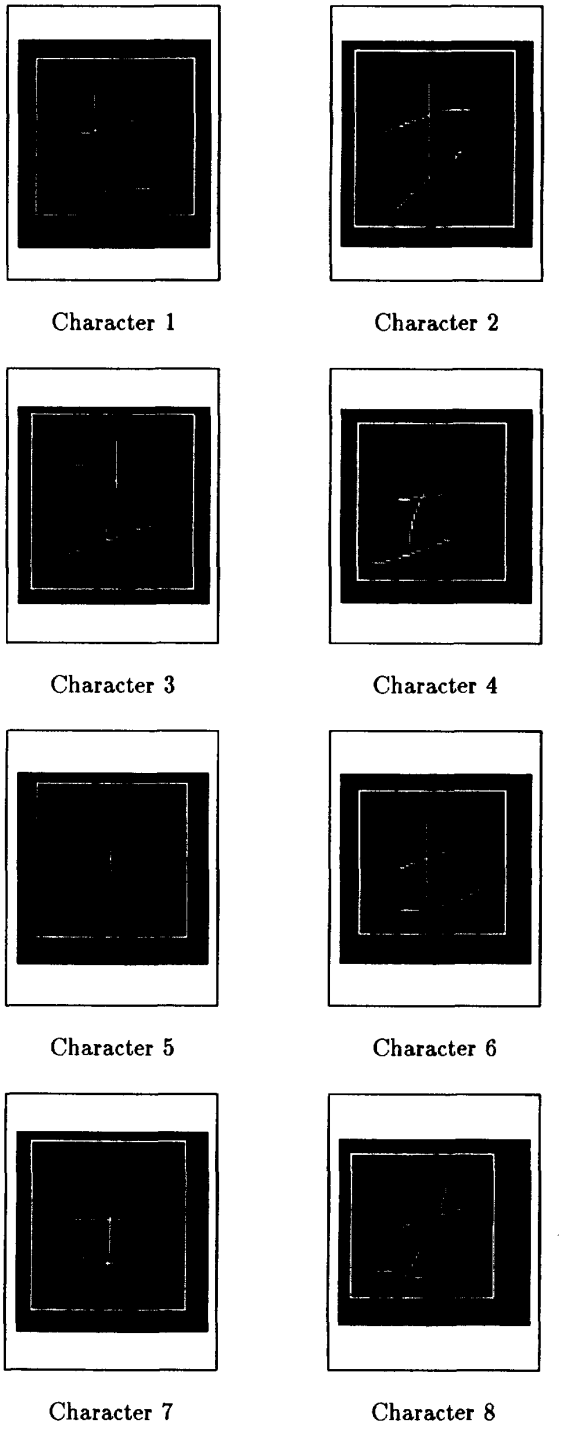
id	*Horizontal*	*Vertical*	*Slant45*	*Slant135*
1	0.96	0.00	0.09	0.00
2	0.86	0.00	0.94	0.00
3	0.89	0.00	0.25	0.00
4	0.85	0.00	0.22	0.00
5	0.31	0.00	0.00	0.61
6	0.84	0.00	0.18	0.00
7	0.81	0.00	0.00	0.20
8	0.85	0.00	0.00	0.18

(c) Stroke 3

Table 1: Membership value of the fuzzy set *STROKE-TYPE*

have fuzzy attribute as described in example 1 of section II. To test the effectiveness of the algorithm we try the approach on the character 土. Figure 1 shows 10 samples of the character 土. For simplicity, we only use 1 attribute – the stroke type, for illustration. The stroke type may take four values – *Horizontal*, *Vertical*, *Slant45* and *Slant135* which are the main direction of strokes in a Chinese characters (two in horizontal and vertical and two in the diagonals). They are represented as a fuzzy set. Table 1 shows the membership values of the fuzzy set *STROKE-TYPE*.

The evaluation of the membership can be found in [5] First of all, each character is represented by



Character 1 Character 2 Character 3 Character 4 Character 5 Character 6 Character 7 Character 8

Figure 1: A set of training sample used

a *FAG*. The *FAGs* are first matched against each other to find vertex correspondence. This can be achieved by a graph matching algorithm proposed by A. K. C. Wong [6] After the vertex correspondence are found, the algorithm can then be applied. When applying the above algorithm, taking $\lambda = 0.25$, we will get the following result:

Let H, V, S and X represent the attribute-values *Horizontal*, *Vertical*, *Slant45* and *Slant135* respectively, and Q_1, Q_2 and Q_3 be the resultant polynomial form for the three strokes in the character $\pm$. We get

$$\begin{aligned} Q_1 &= 0.37H + 0.49HX + 0.37HS + 0.39HSX \\ Q_2 &= 0.3V + 0.39VS \\ Q_3 &= 0.31H + 0.31HS + 0.61HX + 0.61HSX \end{aligned}$$

Taking the term with the smallest number of attributes, then

$$\begin{aligned} \text{Stroke 1} &= \textit{Horizontal} \\ \text{Stroke 2} &= \textit{Vertical} \\ \text{Stroke 3} &= \textit{Horizontal} \end{aligned}$$

This definition is the same the what we would expect.

When we matched this template with the original 8 patterns, the degree of matching are respectively 0.85, 0.83, 0.37, 0.43, 0.31, 0.84, 0.81 and 0.30, all exceed our threshold $\lambda = 0.25$. Hence correct result is produced by the algorithm.

VI. Conclusion

In this paper, we have discussed a pattern recognition approach based on the *Fuzzy-Attribute Graph*. It is the author's belief that human perception is performed by looking for identifiable subpatterns in a complex scene and based on these parts, make decision about the scene. Hence we approach the problem by finding subpatterns using matching of *FAGs*. and then combine the matched ones to form the pattern.

Also the template used for matching is a kind of definition which should not contain any fuzziness, while any "instance" of the template is fuzzy. Hence we define the matching between a fuzzy instance and a non-fuzzy template. However, we do not want to define the template by ourselves. This will be very tedious and error prone, especially when we have many templates to define, such as in Chinese character recognition. We would like to have an automatic mechanism to do this instead.

In this paper, we propose an algorithm which can find the smallest template from a set of training samples based on the matching metric. We have proved that the resultant *FAG* from the algorithm is in fact λ-monomorphic to all the learning pattern. As an illustration, the algorithm was applied to 8 character $\pm$ and the above verified. The result also match with our definition.

Finally, the algorithm is linear on the number of training samples. Although it is exponential with respect to the number of values in each attribute, this is usually very small. In our application on Chinese character recognition, the largest number is 6. Hence this can be assumed to be a constant. Thus, an efficient algorithm has been developed for the learning of *HFAG* from a set of examples of *FAGs*.

Reference

[1] L. A. Zadeh, "Fuzzy Sets," *Inf. Control*, Vol. 8, pp. 338–353, 1965.

[2] W. H. Tsai and K. S. Fu, "Error Correcting Isomorphism of Attributed Relational Graphs for Pattern Analysis," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 9, No. 12, pp. 757–768, Dec, 1979.

[3] K. P. Chan, "Guarded Fuzzy-Attribute Context Free Grammar and its Application on Structural Pattern Recognition," *Int. J. Pattern Recognition and Artificial Intelligence*, Vol. 6, No. 5, 1992.

[4] K. P. Chan and Y. S. Cheung, "Fuzzy-Attribute Graph with Application to Chinese Character Recognition," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 22, No. 4, pp. 402–410, Mar, 1992.

[5] K. P. Chan, "Fuzzy-theoretic approach to Handwritten Chinese Character Recognition," Univ of Hong Kong, Ph. D. Thesis, 1989.

[6] A. K. C. Wong, M. You and S. C. Chan, "An algorithm for graph optimal monomorphism," *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 20, No. 3, pp. 628–638, May, 1990.