A Visual Knowledge Representation Language for Layout Problem

Tetsuto Yoshikawa

ASTEM RI
17 Chudoji Minami-machi, Simogyo, Kyoto 600 JAPAN

Abstract

This paper proposes a new knowledge programming approach to solve a layout problem. The new approach describes a problem in the two-dimensional language called a visual language, applies to the problem a form of the knowledge that is also described in the visual language, solves the problem, and represents the solution, also in the visual language.

The knowledge is represented by a type of production rules called visual production rules. A visual production rule has conditional and action parts that are represented by the views written in the visual language. A view is composed of visual objects, visual predicates, and visual pattern operators. The reasoning algorithm employed in this approach is forward chaining that is based on the pattern matching represented in the visual language.

The new approach makes the description of the knowledge about layouts simpler than conventional lines of the approach. This provides greater benefits in knowledge acquisition. It also allows one to describe ambiguous knowledge about the layout of the target objects without detailed coding. The new approach arouses great interest as a means of implementing an inference process that executes while leaving intact ambiguous information such as that which humans handle.

1. Introduction

In a problem solving process using knowledge that includes two-dimensional information, we usually represent two-dimensional information as single dimensional symbol (textual language). This approach is effective, because we can use a conventional algorithm for problem solving. However, it has two problems. One is that it is difficult to translate two-dimensional information to single dimensional information, and the other is that a vague knowledge is lost when things come to the worst. To resolve these problems, we propose a new approach that represents all two-dimensional information as a two-dimensional symbol that is called a visual language.

In this approach, a problem is described in the visual language, and it is solved using knowledge described in the visual language, and a solution of the problem are described in the visual language as shown in Fig.1. We call a language which describes the knowledge in the visual language as visual knowledge representation language (VKRL), and call this approach "visual knowledge programming".

![Fig.1 Problem Solving Architecture](image)

For example, we think a concrete layout problem given in Fig.2. In this problem, there are one table and two chairs in a room with a door. If an expert of room layout thinks it is no good that the table is near the door, he may change the positions of each furniture shown in Fig.3 in his mind.

In a practical problem, experts solve problems by visual pattern operation in a two-dimensional space. The proposed approach reproduces an expert's thinking process like above example, and it provides a natural and easy knowledge representation language.
In this paper, we discuss a new approach for knowledge based layout problem solving. We outline the basic idea of the visual knowledge programming in section 2. From this perspective, we discuss a VKRL in section 3 and an inference method in the section 4. Finally, we discuss an extended VKRL.

2. Framework

We regard that a typical process of a two-dimensional conceptual thinking for a layout is a matching and an operation of a pictorial pattern. Therefore, we think it is suitable to formalize this process as a production system. For this reason, we introduce a concept of visual production system as a framework of problem solving. We call this production system VPS.

Basic faculty of VPS are follows. For a given problem in the visual language, VPS solves the problem using rules described in the visual language, and displays a solution in the visual language. We call the rule described in the visual language a visual rule.

To realize this faculty, the next six facilities are needed.

1. Pictorial problem Describer
   Accept a problem described by the visual language.

2. Inference Engine
   Pattern match of two visual language sentence

3. Visual Pattern Compiler
   Parse a visual language sentence and translate to a internal description

4. Visual Rulebase
   Store and manage of visual rules

5. Visual Working Memory
   Memorize a state of inference by geometric information

6. Visual Solution Display
   Display a solution by a picture

Fig.4 shows relation of these six facilities.

3. Representation of Visual Rule

In this section, we define a key concept of the visual rule at first. Secondly, we define a description of conditional and action parts of the visual rules. Finally, we define the visual rule.

3.1 Key concept

A basic elements of the visual rule are visual object, pattern operator, and view.

Visual object

A visual object is a primitive element of the visual rule. There are two kinds of a visual object. One is class visual object, and the other is instance visual object. A concept of class object and instance object are same as them in a usual object oriented- programming language. A structure of the visual object is shown in Fig.5. The visual object is a compound of object name, class declaration, object_type (class/instance) declaration, image representation, class of object shape, and attributes. A position and a shape are represented as the attribute.
The visual objects are defined by the users. The classes are also defined by the user except two classes (e.g., text class, frame class). The classes in advance are as follows:

1. Text class
   A text class is a class object without the image representation. This class object is used to express a knowledge by the textual language. A symbol is a mark which is put on a befit position.

2. Frame class
   A frame class object makes the frame. The frame is a rectangle that encloses all visual objects. This class object is used to express an absolute position of each visual object in a fixed space.

   The image representation is a bitmap of the object. The shape has a class as rectangle, circle, and so on. The attributes are additional information for the visual object.

   Following is an example of the visual object definition.

   [example of the visual object]
   Object_name : table
   class : class
   Object_type : class
   Class of Object shape : rectangle
   Image representation : (bitmap file)
   position : left_under
   position x : 100
   position y : 100
   shape_high : 10
   shape_wide : 20

Pattern operator
Pattern operators show operations to arrange the visual objects. The VSP has seven pattern operators, i.e., ADD, DELETE, EXCHANGE, MOVE, REPLACE, ALIGN and ROTATE.

View
A view corresponds to an element of a conditional part or an action part of a usual production rule. The view is a drawing composed of the visual objects and the pattern operators in the two-dimensional space. The conditional and the action parts of visual rule are sets of views with a logical relation. We will discuss a detail representation at later section. A example of the view is shown in Fig.6. In Fig.6, the view is constructed from next visual object, table(rectangle), chair(circle), room(rectangle), and door (bar).

![Fig.6 Example of View](image)

There is a special view which is composed of only the text class objects. In this view, the text class objects are shown as the nodes, and their relations, AND, OR, and NOT are shown as the links, and the condition is expressed as a logic chart. The logic chart expression is shown in the later section.

3.2 Definition of the conditional part
The conditional part of visual rule is defined as a set of views and a relation among them. The relations are as follows;

Threshold of the view
When a view pattern matches with working memory pattern, similarity degree which indicates how much the view pattern similar to the working memory pattern is calculated. The view pattern is regarded to match with the working memory pattern when the similarity degree is over a threshold value.

The logical relation
A relation among the views is represented by AND/OR relation. Further, each view can take the NOT relation which means the view matches the working memory when the similarity degree is under the threshold.

To describe the conditional part, a logic chart is used.
In the logic chart, next graphical symbols are used.

- \( \square \times \) : Threshold of the View
- \( \square \) : AND relation among the views
- \( \circ \) : OR relation among the views
- \( \Box \) : NOT of the view

### 3.3 Definition of the action part

VPS has following seven action operator; ADD, DELETE, EXCHANGE, MOVE, REPLACE, ALIGN and ROTATE. Each operator has a graphical symbol shown as below. An operand is shown by putting the graphical operator on the visual object. We discuss each operator using the view shown in Fig.6 as example.

1. **ADD operator**
   ADD operator adds a new visual object to the working memory. A graphical symbol is \( + \) and put it on the visual object which is added. An example view that the ADD operator adds the chair(circle) object to the view of Fig.6 is shown in Table-1.

2. **DELETE operator**
   DELETE operator delete a object included in the working memory. A graphical symbol is \( \times \) and put it on a visual object which is deleted. An example view that DELETE operator deletes the chair(circle) object in Fig.6 is shown in Table-1.

3. **EXCHANGE operator**
   EXCHANGE operator exchanges a position of two visual objects in a working memory. A graphical symbol is \( \leftrightarrow \) and put its arrows on the visual objects which are exchanged. An example that the EXCHANGE operator exchanges the chair(circle) objects and table (rectangle) object is shown in Table-1.

4. **MOVE operator**
   MOVE operator moves one visual object from a original position to a new position in a working memory. A graphical symbol is \( \rightarrow \) and put its start point on the moved object and put its end point(arrow) on a new position. An example that MOVE operator moves a table (rectangle) object to another position is shown in Table-1.

### Table-1 Pattern Operator

<table>
<thead>
<tr>
<th>Operator name</th>
<th>expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD operator</td>
<td>( + )</td>
</tr>
<tr>
<td>DELETE operator</td>
<td>( \times )</td>
</tr>
<tr>
<td>EXCHANGE operator</td>
<td>( \leftrightarrow )</td>
</tr>
<tr>
<td>MOVE operator</td>
<td>( \rightarrow )</td>
</tr>
<tr>
<td>REPLACE operator</td>
<td>( \leftarrow \rightarrow )</td>
</tr>
<tr>
<td>ALIGN operator</td>
<td>( \uparrow \downarrow )</td>
</tr>
<tr>
<td>ROTATE operator</td>
<td>( \circ )</td>
</tr>
</tbody>
</table>
(6) **REPLACE** operator

REPLACE operator replaces one visual object which is included in the working memory to other visual object which is not included. A graphical symbol is $\rightarrow$, and put its start point on a included object and put its end point on a not included object. An example that REPLACE operator replaces one chair(circle) object to another chair(circle) object is shown in Table-1.

(6) **ALIGN** operator

ALIGN operator put all object on a line. A graphical symbol is $\rightarrow$, and put its start point on a moved object and put its end point on a line. An example that ALIGN operator align two chair(circle) object on the line is shown in Table-1.

(7) **ROTATE** operator

ROTATE operator moves multiple objects keeping their topological relation. A graphical symbol is sequence of arrow, and put its start point on one object and put its end point on another object which moves at same time. An example that ROTATE operator move two chair(circle) objects and a table(rectangle) object shown in Table-1.

3.4 The visual rule representation

The visual rule is described by the definition of the condition and the action sequence on the rule shown in Fig.7.

4. Inference algorithm

An inference is a recognize and action cycle. In this section, we discuss an algorithm how to match the view pattern to the working memory pattern, and how to modify the working memory according to the action view.

4.1 Control of the inference

A detail flow of recognize and action cycle in VPS is as follows.

a) One view in the evaluated visual rule is got out.
b) The similarity degree between the view and working memory pattern is calculated by the algorithm shown in later section.
c) If the similarity degree is over the threshold, the next view is got out.
d) When the similarity degree of the all view have been calculated or one similarity degree is under the threshold, a next visual rule is evaluated.
e) After the evaluation of all rules the action part of the visual rule which has the highest applicable degree is executed. A calculation algorithm of the similarity degree is discussed later, and the applicable degree is calculated from the similarity degree by the rule as follow;

\[
\text{AND} : \times \\
\text{OR} : \text{minimum} \\
\text{not} : [1 - (\text{similarity degree})]
\]
f) The action is executed.

We explain the detail algorithm of b) and f) which is the feature of our approach.

4.2 Working Memory Structure

We discuss structure and content of the working memory of VPS before discussion about a pattern matching algorithm.

The working memory shows a current arrangement of layouted object. The working memory is constructed from the instance visual objects shown in Fig.5. As known from Fig.5, information in the working memory is represented by a frame structure. It includes geometric information which is constructed from the position of the visual object, the shape, the type of the shape and so on.

4.3 Pattern matching algorithm

A pattern in the view is a picture as a language, so it is not necessary to think absolute size. Consequently, two pictures are regard as matching when the topological relation between the visual objects is same. From this viewpoint, we propose the next algorithm.
Pattern matching algorithm

Let the pictorial pattern in the working memory is \( A \), and one in the thinking view is \( X \).

<step1>
Establish a correspondence between objects. Make sure that pattern \( A \) always contains a visual object that belongs to the same class and shape as a visual object in case \( X \).

<step2>
Designate one of the objects in view \( A \) extracted in step1 as reference object \( A_0 \) and its center of weight \( a_0 \) as the reference point. Also designate the corresponding object in \( X \) as reference object \( X_0 \) in \( X \) and its center of weight \( x_0 \) as the reference point.

<step3>
Consider a two-dimensional coordinate system \( AA_x \) with \( a_0 \) taken as its origin and find the angle \( \theta_{Ai} \) with the \( X \) axis and distance \( D_{X} \) from the \( X \) axis for all corresponding objects \( A_i \) (\( i=1, n \)) in \( A \).

<step4>
Similarly, find angles \( \theta_{Xi} \) and distances \( D_{Xi} \) for objects \( X_i \) (\( i=1, n \)) in the two-dimensional coordinate system \( XAX \) with \( x_0 \) taken as the origin.

<step5>
Prepare in advance the function \( f_\theta \) (\( 0 < f_\theta < 1 \)) that represents the angular error and the weight for similarity and compute the relative similarity \( S_{i\theta} \) of the individual corresponding objects.

\[
S_{i\theta} = f_\theta (\theta_{Ai} - \theta_{Xi})
\]

where \( f_\theta \) may be, for example, a function shown in Figure 8. \( f_\theta \) may be the same for all objects representing the features of the drawing or may be represented by a function that gives relatively larger values for objects that feature the pattern in question.

<step6>
Find the similarity \( S_\theta \) representing the overall angle. \( S_\theta \) is the weighted mean of the similarities of all elements. It is computed as shown below to match the human feeling that more elements are there that have higher similarities, the higher is the similarity of the whole view.

\[
S_\theta := S_{\theta 1} + (1 - S_\theta) \cdot S_{\theta i} (i=2, \ldots, n)
\]

\[
S_\theta := S_{\theta n}
\]

<step7>
Find the similarity of the distance as follow.
1) Calculate the next values for each objects.

\[
d_i = D_{Xi}/D_{Ai}
\]

The average of this values \( d_{ave} \) is calculated by

\[
d_{ave} = 1/n \sum d_i
\]

Here, \( d_{ave} \) means the scale of \( A \) for \( X \).
2) Prepare the weighting function of the similarity for the offset of the distance \( f_d \) (\( 0 < f_d < 1 \)) and calculate the similarity with the distance \( S_d \) for each objects.

\[
S_d = f_d(D_{Ai} \cdot d_{ave} - D_{Xi})
\]

<step8>
Find the similarity \( S_d \) for the distance from the all objects. \( S_d \) is calculated by the next expression.

\[
S_d := S_{d0} + (1 - S_{d0}) \cdot S_{di} (i=2, \ldots, n)
\]

\[
S_d := S_{dn}
\]

<step9>
Find the similarity \( S \) of the over all pattern from the similarity of the angle and the distance by the next expression

\[
S = S_\theta + (1 - S_\theta) \cdot S_d.
\]

In this algorithm, it is possible to match the pattern like to human sense by the method of the load average and the method to give the weighting function.

4.4 Algorithm for executing action

An executing action means operation for the working memory. DELETE, EXCHANGE, and ROTATE operator are easy to realize if the corresponding object can be found. ADD, MOVE, and ALIGN operator are difficult. To realize these operators, it is necessary to get out the
action which is given the pictorial information, and execute the pictorial operation according to the working memory scale. As consider this, the algorithm of the operation for the working memory is as follows:

**<step1>**
Check whether there is the corresponding view object to the working memory object.

In this step, the corresponding object means that two objects included in X and A are the same class. If not exist, the action is not executed.

If the operator is one of the DELETE, EXCHANGE and CHANGE, the action is executed in this step. If the operator is ADD or MOVE or ALIGN then go to the next step.

**<step2>**
Find the scale of the view for the working memory.

Let one object in the pattern A be the basis object Ao in A. And let the object correspond to Ao be basis object Xo in X. Further, let the distance of Ai from the Ao be DAi, and let the distance of Xi from the Xo be DXi. Calculate the average of the distance s

\[ s = \frac{1}{n} \sum DAi / DXi \]

and, let s be the scale.

**<step3>**
Modify the working memory by the action scaled by s.

The class object in the view change to the instance object when the object add to working memory.

### 5. Improvement of availability

In this section, we discuss how to enhance the availability to represent the knowledge. We discuss this problem from two view point. One is an availability to represent conditions of visual rules. The other is to represent the actions.

**Visual Predicate**

The knowledge representation language shown in Section 4 represents only an instance information. Therfore, this language can not provide an abstract expression. However, we need abstract expression to represent conceptual knowledge.

To enhance availability to represent an abstract knowledge, we introduce a concept of visual predicate. The visual predicates are predicates that are illustrated as graphical symbols and have visual object as arguments.

In the VPS, we can think six types visual predicate, EXIST, ZONE, CONTACT, OVERLAP, ALIGN, and ROTATE. We will explain a outline of the six visual predicate.

1. **EXIST predicate**

   EXIST predicate gives an arrangement of visual objects in a view. This predicate provides a positions of each object. As shown in Table-2. EXIST predicate does not have visual symbol.

#### Table-2 Visual Predicate

<table>
<thead>
<tr>
<th>predicate</th>
<th>example of expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIST</td>
<td><img src="image" alt="Exist Example" /></td>
</tr>
<tr>
<td>ZONE</td>
<td><img src="image" alt="Zone Example" /></td>
</tr>
<tr>
<td>CONTACT</td>
<td><img src="image" alt="Contact Example" /></td>
</tr>
<tr>
<td>OVERLAP</td>
<td><img src="image" alt="Overlap Example" /></td>
</tr>
<tr>
<td>ALIGN</td>
<td><img src="image" alt="Align Example" /></td>
</tr>
<tr>
<td>ROTATE</td>
<td><img src="image" alt="Rotate Example" /></td>
</tr>
</tbody>
</table>
(2) ZONE predicate

ZONE predicate presents that visual objects are in a specific zone. A graphical symbol of ZONE predicate is a rectangle shown in Table-2. A expression by ZONE predicate is constructed from the visual objects and rectangle surrounding them.

(3) CONTACT predicate

CONTACT predicate gives whether specific two visual objects contact each other. A graphical symbol of CONTACT predicate is CT with circle shown in Table-2. A expression by CONTACT predicate is a picture in which CONTACT object cover both visual objects.

(4) OVERLAP predicate

OVERLAP predicate gives whether specific two visual objects overlap. A graphical symbol of OVERLAP predicate is OV with circle shown in Table-2. A truth expression by OVERLAP predicate is a picture in which OVERLAP predicate cover both visual objects.

(5) ALIGN predicate

ALIGN predicate presents that visual objects form a line. A graphical symbol of ALIGN predicate is a line shown as Table-3. A truth expression by ALIGN predicate is a picture in which all visual object are on the align line.

(6) ROTATE predicate

ROTATE predicate presents that specific visual object are same by means of topological relation ship. A graphical symbol of ROTATE object is an arrow which links among related visual objects. A truth condition by ROTATE predicate is that topological relation among related visual objects is same for rotate or slid actions.

The expression shown in section 4 is correspond to EXIST expression. For another predicate, we have to develop a pattern matching algorithm. This is a main furture work relative to VPS.

Point specification method in operator

The visual predicates enhance availability for knowledge representation of the condition part of the visual rules. In other hand, it is important to specify visual objects with a convenient way at the action part.

A current method of specification is to bring only one object to a point. For the convenience, we need to think additional three specification methods

(1) bring only one object to a ZONE
(2) bring multiple objects to a ZONE
(3) bring Object in a ZONE to a ZONE.

Using these ways, the representation availability for knowledge of the action is enhanced.

6. Discussions

Limitation

A problem discussed in this paper is layout problem in two-dimensional space. An arrangement of layouted objects are decided by the topological relation of the visual objects with the attribute.

In a room layout problem, experts perhaps thinks at two phase as follow:

(1) Decide how to put on
   (conceptual design)
   In this phase, qualitative knowledge, such as "it's desirable that a chair is near a table", is used.

(2) Decide where every object put on
   (detailed design)
   In this phase, according to the layout decided in (1), the place where every object put on is decided. Deciding it, the shape and the size of the object are considered as the constraint. If the troubles occur in this phase, the designer returns to the phase (1).

The basic concept of this approach is to treat the picture as a language. Therefor, it is difficult to treat qualitative information, and it is the limit of this approach. Consequently, the target is phase (1), not phase (2). By this approach, it is easy to describe the knowledge for the phase (1), but it is not easy to describe the knowledge for the phase (2).

Applicability

There are many equivalent problem to the problem in this paper. We can part them into the next three types from the view of a semantics of the relation among the objects in the space.

(a) relation is position (layout problem)
(b) relation is time (scheduling problem)
(c) relation is concept (pictorial thinking)

Another furture research

In section 5, we point out that the visual predicate and the multiple specification method are need to enhance knowledge representation availability. Additionally, we have to think a mixture with the textual language.

There are many knowledge problem which can not be represented by a picture in the layout problem. For example, the knowledge, "If the noise in the TV exists, put the PC on far away", can not be represented only in the visual language. However the proposed approach can represent these knowledge by the text class visual object, it can not be mixed with the knowledge described in the
visual language. Therefore, it may be impossible to solve the complicated problems. A mixture of these two languages is an important problem.

7. Conclusion

In this paper, we propose VPS as a new knowledge representation language. Further, we present the new knowledge-based problem solving approach using this language. In this approach, we represent a problem, a knowledge and a solution in the visual language. The inference process is geometric pattern matching of the visual language. Finally, we discuss the extension of the proposed language in order to enhance the availability to represent an abstract knowledge. As a result, we propose the visual predicate and the multiple methods to specify the operand.

By this approach, it became easy for some problem to represent two-dimensional information which has been difficult. A system according to this approach is developing now.

On the other hand, we confirm an importance of a mixed descriptions with the visual language and the textual language. We need to research for this problem.

Finally, the visual knowledge programming has the another aspect as a user interface to acquire knowledge. From this aspect, there is a study to describe the IF-THEN rule as a flowchart(1). This study use the picture (flowchart) to make rule description easy. In our approach, not only the structure of the rule, also the semantics is represented by the picture (visual language). It means that our approach give the different viewpoint for describing the knowledge and it is excellent from the usual approach.

[REFERENCE]