Information Modelling and Knowledge Bases XXXII
M. Tropmann-Frick et al. (Eds.)

doi:10.3233/FAIA200818

A Concept for Control and Program Based on the Semantic Space Model

Xing Chen[1], Maimai Prayongrat[2] and Yasushi Kiyoki[3]
[1] *Department of Information & Computer Sciences*
Kanagawa Institute of Technology, Japan
[2] *Mechanical Engineering, Chulalongkorn University, Thailand*
[3] *Graduate School of Media and Governance, Keio University, Japan*

Abstract. The most important mechanism of the computer is that various functions are implemented based on programs stored in it. Programs are developed by program languages implementing functions of models. One of the efficient methods to construct a model is to construct it by semantic computation models. Using semantic computation models, we can construct a model in a semantic space. In this paper, we present a mechanism to execute models presented by the semantic spaces. We have presented a mechanism to implement combinational and sequential logic computations based on the semantic space model. The combinational and sequential logic computations are the basic functions in computer systems. However, we still need a control mechanism like that in computers. In this paper, we present a control mechanism based on the semantic space model and some of execution examples. The most important contribution of this paper is that we first present a concept for control and program based on the semantic space model. In order to demonstrate the efficiency of the proposed mechanism, we performed a demonstration experiment. In the experiment, an agent is constructed for unmanned ground vehicle control with the control mechanism. A video camera is used to determine the position of the vehicle and obstacles on the road. The control signals, including "turn left," "turn right," "go ahead" and "stop" outputted from the agent are used to demonstrate the efficiency of the mechanism.

Keywords. Artificial intelligent, semantic space, semantic computation, unmanned control

1. Introduction

Control mechanism is a basic required mechanism for automatic control systems. In a computer system, the control unit controls the arithmetic and logic calculation to implement the basic functions of the computer according to programs stored in the memory. Programs are developed by program languages implementing functions of models. One of the efficient methods to construct a model is to construct it by semantic computation models [1, 2, 3, 4]. The semantic computation models are based on multiple matrix calculation which is also utilized for implementing artificial neural networks and deep-learning [5]. By using semantic computation models, we can create human understandable models.

[1] Xing Chen, 1030 Simo-Ogino, Atsugi-shi, Kanagawa 243-0292, Japan; chen@ic.kanagawa-it.ac.jp

The combinational and sequential logic computations are the basic functions in computer systems. We have presented a mechanism to implement combinational and sequential logic computations based on the semantic space model [6]. In our semantic computing models, data are presented as points in semantic orthogonal spaces [1, 2, 3, 4], and the semantic calculation is transmitted to calculate Euclidean distances of those points. For example, in the case for implementing semantic query, a query data set is mapped into a semantic space and summarized as a point in the space. Retrieval candidate data are also mapped into the semantic space and summarized as other points. Euclidean distance is calculated between the query point and each retrieval candidate point. When the distance of a retrieval candidate is shorter than a given threshold, its relative retrieval candidate is extracted as the query output.

Mapping matrixes are used to map input into the semantic space. Different mapping matrixes are required when the semantic space model is applied in different application areas [7- 15]. Therefore, we developed many methods to create mapping matrixes and apply the model in the areas of semantic information retrieving [10, 11, 15], semantic information classifying [12], semantic information extracting [13], and semantic information analyzing on reason and results [15], etc. We furtherly developed a method to create the mapping matrixes through deep-learning [16].

In order to apply the semantic computation model for implementing the combinational and sequential logic computations, mapping matrixes are created according to truth tables and state transition tables. Each rows of the tables are multiplied with the mapping matrixes and mapped into the semantic spaces. In this way, the logical computation is implemented as calculating Euclidean distances of mapped points in a semantic space. Thus, we obtained the basic functions for implement the intelligent systems. However, we still need a control mechanism like that in computers. In this paper, we present a control mechanism based on the semantic space model and some of execution examples. The most important contribution of this paper is that we first present a concept for control and program based on the semantic space model. In order to demonstrate the efficiency of the proposed mechanism, we performed a demonstration experiment. In the experiment, an agent is constructed for unmanned ground vehicle control with the control mechanism. A video camera is used to determine the position of the vehicle and obstacles on the road. The control signals, including “turn left”, “turn right”, “go ahead”, “stop”, etc., outputted from the agent are used to demonstrate the efficiency of the mechanism.

In the following, we first briefly review the semantic space model in Section 2. In Section 3, we review the concept of the combinational and sequential logic computations implemented based on the semantic space model. After that, we present the control mechanism and execution examples in Section 4. This mechanism addresses the concept of time, clock, state memory and subspace creation. In section 5, we present the demonstrate experiment and the experimental results. Finally, we will present our conclusions in section 6.

2. The semantic space model

2.1. Semantic Feature Extracting Model (SFEM)

In the semantic space model, we create a semantic space by a pre-selected training data set into several clusters. In SFEM model [1, 2], a data set is used where each of the

clusters has a feature that some common features of data frequently appear among the data sets in the same cluster but rarely appear in the data sets of the other clusters. The common features which frequently appear in a cluster C_i are referred to as C_i's key features. By using the training data clusters, we construct a matrix, which is referred to as *K-C* matrix. In the *K-C* matrix, each of the rows corresponds to a key feature set K_i, and each of the columns corresponds to a cluster. The ij^{th} entry of the matrix is the number of the key features in the set K_i appearing in the cluster C_j. Because key features in the set K_i only appear in the cluster C_i, therefore, if i is not equal to j, $i \neq j$, the value of the ij^{th} entry is 0. Therefore, the *K-C* matrix is a diagonal matrix. That is to say that an orthogonal space is created. The value of the ii^{th} entry of the matrix is the number of the elements of the set K_i, $|K_i|$.

Next step in the semantic space model is to map data into the semantic space. By using the *K-C* matrix, each cluster is represented as a q dimensional vector. We use $\mathbf{C}_i$ to represent the vector of the cluster C_i. We use a unit vector $\mathbf{c}_i$, where its norm is 1 ($|\mathbf{c}_i|=1$), to represent the cluster vector $\mathbf{C}_i$ as $\mathbf{C}_i=|K_i|\mathbf{c}_i$. When the data are classified into q clusters, we obtain q cluster vectors. Therefore, we define q unit vectors $\mathbf{c}_1$, $\mathbf{c}_2$, ..., $\mathbf{c}_q$, to represent the q cluster vectors. We refer the vector space constructed by the q unit vectors to as the "*space*". Because the inner product of two different unit vector is 0, $(\mathbf{c}_i \cdot \mathbf{c}_j)=0$, $i \neq j$, and there are q unit vectors, the space is a q dimensional orthogonal space.

When data d_j, is vectorized according to the key features, the count of the occurrences of the C_i's key features in the data d_j is defined to $e_{i,j}$. We set the value of $e_{i,j}$ based on the following rule: *when a key feature in the key feature set K_i appearing in the data d_j, it is counted only once*. If the number of the key feature sets is q, the data d_j is vectorized to a q dimensional vector. We represent the vector of the data d_j as $\mathbf{d}_j$:

$$\mathbf{d}_j = \begin{bmatrix} e_{1,j} \\ e_{2,j} \\ \vdots \\ e_{q,j} \end{bmatrix}$$

We use v_t to represent the counted value of a key feature t. In the following, we use the expression $\sum_{t \in K_i} \{v_t\}$ to represent the sum of $v_{t_1} + v_{t_2} + \cdots + v_{t_a}$, where, t_1, t_2 ... t_a are the elements of the key feature set K_i:

$$\sum_{t \in K_i} \{v_t\} = \left\{ v_{t_1} + v_{t_2} + \cdots + v_{t_a} \middle| t_1 \in K_i, t_2 \in K_i, \cdots, t_a \in K_i \right\}.$$

In this way, the calculation for $e_{i,j}$ is represented by the following formula:

$$e_{i,j} = \sum_{t \in K_i} \left\{ v_t \mid if\ t \in d_j\ v_t = 1\ else\ v_t = 0 \right\},$$

where, K_i is the set of the C_i's key features and v_t is the counted value of one of the C_i's key feature t. If the data d_j contains the key feature t, v_t is set to 1. If the data d_j does not contain the key feature t, v_t is set to 0.

With the definition of the retrieval space, we express the data vector $\mathbf{d}_j$ as

$$\mathbf{d}_j = \sum_{i=1}^{q} e_{i,j}\mathbf{c}_i \ .$$

In this way, data are mapped onto the q dimensional space.

The third step is to calculate Euclidean distances for data retrieval, classification or recognition. Take the data query as an example. In the processing of data query, the retrieval candidates are mapped in the semantic space and summarized as retrieval candidate points. A query is also mapped in the semantic space and summarized as a query point. We use two methods to implement the Euclidean calculation. The first method is to calculate the distances between the retrieval candidate points and query point. The second method is to is to select a subspace by a given query and mapped the query into the original point of the subspace and calculate the length of each retrieval candidate points from the original point. That is to calculate the norms of the retrieval candidate points. By ranking the retrieval candidates based on the norms of their relative points, the query result is obtained.

When a subspace is selected from the semantic space based on queries, the subspace is a v-dimensional space which is a part of the q-dimensional space, where v is smaller than q. The v-dimensional subspace correlates to v clusters. The subspace is selected by the following steps.

(1) When a query Q is given, the data which contain the same features in the query is searched. Data that have the same feature as those in the query are extracted.

(2) From all the component items of the selected data vector, the cluster vector $\mathbf{c}_i$ is extracted where the related component item $|e_{i,j}\mathbf{c}_i|$ has the maximum value.

$$\left|e_{i,j}\mathbf{c}_i\right| = MAX\left(e_{1,j}, e_{2,j}, \cdots, e_{q,j}\right)$$

We use a vector $\mathbf{q}$ referred to as the query vector to represent the extracted clusters. We add the extracted cluster vector $\mathbf{c}_i$ to the vector $\mathbf{q}$,

$\mathbf{q}=\mathbf{q}+\mathbf{c}_i$; where the initial value of $\mathbf{q}$ is $\mathbf{0}$.

(3) A retrieval subspace S corresponding to the query is selected from the entire retrieval space by calculating the inner product of $\mathbf{c}_i$ and $\mathbf{q}$. If the value of the inner product $\mathbf{c}_i \cdot \mathbf{q}$ is greater than or equal to a threshold ε, which is referred to as the subspace selection threshold, $\mathbf{c}_i$ is added to S.

These steps (2) and (3) are repeated for each extracted data vector.

When the subspace S is obtained, the data vector on the subspace is projected and represented as

$$\mathbf{d}_j = \sum_{i=1}^{q} \left\{ e_{i,j}\mathbf{c}_i \middle| \mathbf{c}_i \in S \right\}.$$

In the second step, the data are ranked by calculating the norms of the data vectors on the selected subspace:

$$\left|\mathbf{d}_j\right| = \left| \sum_{i=1}^{q} \left\{ e_{i,j}\mathbf{c}_i \middle| \mathbf{c}_i \in S \right\} \right|.$$

2.2. Mathematical Model of Meaning (MMM)

In MMM, the semantic interpretation is performed as projections of the semantic space dynamically, according to contexts, as shown in Figure 1.

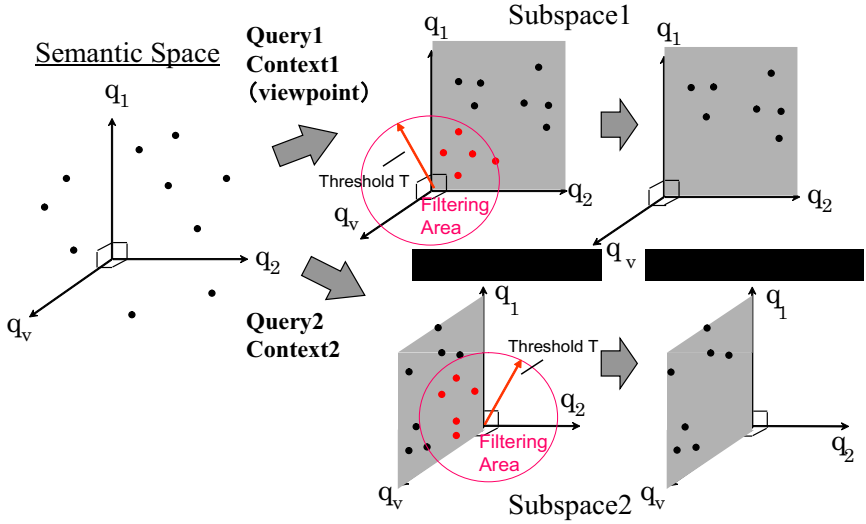


Figure 1. Semantic interpretation according to contexts in MMM

In the Mathematical Model of Meaning (MMM) [4, 7], an orthogonal semantic space is created for semantic associative search. Retrieval candidates and queries are mapped onto the semantic space. The semantic associative search is performed by calculating the correlation of the retrieval semantic space.

In MMM, the acquisition of information or knowledge is performed by semantic computations. Context-dependent interpretation means that information is dynamically extracted by a semantic computation with context-recognition. The method realizes the computational machinery for recognizing the meaning of contexts and obtaining the semantically related information to the given context. MMM is essentially different from those methods. The essential difference is that this method provides dynamic recognition of the context. That is, the "context-dependent interpretation" is realized by dynamically selecting a certain subspace from the entire semantic space. The other methods do not provide the context dependent interpretation, that is, their space is fixed and static. The outline of MMM [4, 7] is summarized as the following:

The semantic associative computing algorithm is extended to include a deep-learning process in the MMM semantic space in the following steps:

(1) A set of *m* words is given, and each word is characterized by *n* features. That is, an *m* by *n* matrix ***M*** is given as the data matrix.

(2) **"Context words" and "image"** are characterized as **"context"** by using the *n* features and representing them as *n*-dimensional vectors.

(3) The context words and **"image"** are mapped into the orthogonal semantic space by computing the Fourier expansion for the *n*-dimensional vectors.

(4) A set of all the projections from the orthogonal semantic space to the invariant subspaces (eigen spaces) is defined. Each subspace represents a phase of meaning, and it corresponds to "**context.**"

(5) A subspace of the orthogonal semantic space is selected according to the given **"context"** expressed in *n*-dimensional vectors, which are given as **"context"** represented by **"a sequence of words"** and **"image."**

(6) The most correlated information resources to the given **"context"** are extracted as the selected subspace by applying the metric defined in the semantic space.

3. Implementing combination logic calculation by multiple matrix computation

3.1. The combinational logic computations implemented based on the semantic space model

Logical design is performed based on truth tables. In the truth table, all output values are given to all possible input values. The input data values and output data vaudes are Boolean values. That is, the value can only be '1' or '0'. For example, if there are two logical input data, x_1 and x_2, all the possible input data values of the input data x_1 and x_2 are (0, 0), (0, 1), (1, 0) and (1, 1), in which (a, b) means the value of x_1 is 'a' and the value of x_2 is 'b'. As shown in Table 1, each input data pair is given a corresponding output value in the truth table. The output values '0', '0', '0' and '1' are given to the input values (0, 0), (0, 1), (1, 0) and (1, 1) in the "and" logic truth table. Based on the truth table, logical formulas are derived. For example, for the "and" logic, an equation

$$y=x_1*x_2,$$

is derived, where '*' presents logic "and". In the same way, an equation

$$y=\tilde{}x_1*x_2+x_1*\tilde{}x_2$$

is also derived for the "xor" logic, where '*', '+' and '~' present logic "and", "or" and "not", respectively.

Table 1. Truth Table of "and," "or" and "xor" logic

"and" logic			"or" logic			"xor" logic		
x_1	x_2	y	x_1	x_2	y	x_1	x_2	y
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

A logical system is constructed by the derived formulas from the truth table. Boolean algebra is applied to simplify the derived formulas in order to reduce the complexity of the designed system. For example, the formula

$$a*b+a*c+\tilde{}a*\tilde{}c+b*c$$

can be simplyfied as

$$b+a*c+\tilde{}a*\tilde{}c.$$

In the following, we use an example to illustrate how to implement "and," "or" and "xor" combination logical calculation based on the semantic calculation model. First we present a data set with two inputs x1 and x2 and three outputs corresponding to "and," "or" and "xor," as shown in Table 2.

Table 2. State transition table

x_1	x_2	Y_{and}	Y_{or}	Y_{xor}
0.2	0.1	0.1	0.2	0.1
0.1	0.9	0.2	0.8	0.9
0.9	0.1	0.1	0.9	0.8
0.9	0.8	0.9	0.8	0.2

The values in data set are set as: when a value is close to 0, it might be logic '0;' when a value is close to 1, it might be logic '1.' If a data value is 0.5, it might be logic '0' or logic '1.' For the given data set, $x_1 = 0.9$ and $x_2 = 0.1$, it means that the input might be $x_1 = 1$ and $x_2 = 0$. The "and" output of it might be '0.' The "and" output of it might be '0.' The "or" output of it might be '1,' and the "xor" output might be '1.'

Set a data set as a matrix M. A well-known method of the principal component analysis is the Singular Value Decomposition (SVD), which is a matrix computation widely used in spectral analysis, eigenvector decomposition and factor analysis. The computation is performed on a matrix with different entities on the rows and the columns. When SVD is performed on the matrix M, this matrix is decomposed into three other matrixes that contain "singular vectors" and "singular values". We call these three matrixes as U, S and V:

$$M = U*S*V'$$

where, S is a diagonal matrix that contains singular values, matrixes U and V are left and right matrix of S, respectively. V' is the transposed matrix of V. The matrix V has orthonormal columns, that is

$$V'*V=I$$

where I is the identity matrix.

We call the space U*S is the semantic space created by the matrix M. As

$$M = U*S*V'$$
$$M*V = U*S*V'*V$$
$$M*V = U*S*I$$
$$M*V = U*S,$$

we call the matrix V as the *space mapping matrix*. That is, any matrixes of data sets which have the same number of columns of the matrix M can be mapped to the semantic space through the mapping matrix V.

When SVD is performed to the matrix of the given data set,

$$M = \begin{bmatrix} 0.2 & 0.1 & 0.1 & 0.2 & 0.1 \\ 0.1 & 0.9 & 0.2 & 0.8 & 0.9 \\ 0.9 & 0.1 & 0.1 & 0.9 & 0.8 \\ 0.9 & 0.8 & 0.9 & 0.8 & 0.2 \end{bmatrix},$$

we get three matrixes:

$$U = \begin{bmatrix} -0.1 & -0.1 & 0.1 & -1.0 \\ -0.5 & 0.6 & -0.6 & 0.0 \\ -0.5 & 0.3 & 0.8 & 0.1 \\ -0.6 & -0.7 & -0.2 & 0.1 \end{bmatrix} \quad S = \begin{bmatrix} 2.5 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.9 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.8 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.5 & -0.4 & 0.6 & 0.1 & -0.5 \\ -0.4 & 0.0 & -0.7 & -0.2 & -0.5 \\ -0.3 & -0.6 & -0.3 & 0.5 & 0.5 \\ -0.6 & 0.1 & 0.1 & -0.6 & 0.5 \\ -0.4 & 0.7 & 0.1 & 0.6 & 0.0 \end{bmatrix}.$$

The semantic space U*S is

$$\begin{bmatrix} -0.3 & 0.0 & 0.1 & 0.0 & 0.0 \\ -1.3 & 0.6 & -0.5 & 0.0 & 0.0 \\ -1.3 & 0.2 & 0.6 & 0.0 & 0.0 \\ -1.6 & -0.7 & -0.2 & 0.0 & 0.0 \end{bmatrix}.$$

This is a five dimensional space. Each row of the matrix represents a mapping point of the data set. As it is a four rows matrix, that means four points are mapped to the semantic space. It is worth to notice that the values of the last two columns of the matrix is zero, therefore, we remove the last two rows and get a new matrix P,

$$\begin{bmatrix} -0.3 & 0.0 & 0.1 \\ -1.3 & 0.6 & -0.5 \\ -1.3 & 0.2 & 0.6 \\ -1.6 & -0.7 & -0.2 \end{bmatrix}.$$

The meaning of removing the last two rows of the matrix is that the semantic space is compressed from a five dimensional space to a three dimensional space.

Strictly speaking, that is each of the value of the last two rows is smaller than a threshold. In this example, the absolute value of the threshold is set to 0.003.

As U'*U=I, that is

$$U'*U = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix},$$

the data set is mapped to an orthogonal space. This orthogonal characteristic is not changed to the compressed space:

$$P'*P = \begin{bmatrix} -0.3 & -1.3 & -1.3 & -1.6 \\ 0.0 & 0.6 & 0.2 & -0.7 \\ 0.1 & -0.5 & 0.6 & -0.2 \end{bmatrix} * \begin{bmatrix} -0.3 & 0.0 & 0.1 \\ -1.3 & 0.6 & -0.5 \\ -1.3 & 0.2 & 0.6 \\ -1.6 & -0.7 & -0.2 \end{bmatrix}$$

$$= \begin{bmatrix} 6.2 & 0.0 & 0.0 \\ 0.0 & 0.8 & 0.0 \\ 0.0 & 0.0 & 0.6 \end{bmatrix}.$$

As the data set is mapped to an orthogonal space, Euclidean distance calculation can be applied to calculate the distance of a new mapped point to the points already in the space.

For example, give a new input set $x_1 = 0.7$, $x_2 = 0.3$. As is different from the given data set, we should calculate four outputs Y_{and}, Y_{or} and Y_{xor} for the given input data set. As the output value is not known, we set the values of the three outputs as 0.5, that means it might be logic '0' or logic '1.' Thus we get a vector

$$[0.7 \;\; 0.3 \;\; 0.5 \;\; 0.5 \;\; 0.5].$$

Mapping the vector to the semantic space,

$$[0.7 \;\; 0.3 \;\; 0.5 \;\; 0.5 \;\; 0.5]^* V,$$

we get a mapping point p_x represented as a three dimensional vector,

$$p_x - [-1.1 \qquad -0.2 \qquad 0.2].$$

Dividing each rows of matrix P, we get four mapping points of the given data set in the semantic space, p_1, p_2, p_3 and p_4, represented as four vectors:

$$p_1 = [-0.3 \qquad 0.0 \qquad 0.1],$$
$$p_2 = [-1.3 \qquad 0.6 \qquad -0.5],$$
$$p_3 = [-1.3 \qquad 0.2 \qquad 0.6],$$
$$p_4 = [-1.6 \qquad -0.7 \qquad -0.2].$$

Calculating the Euclidean distances of p_x to p_1, p_2, p_3 and p_4, we get four values: 0.64, 1.01, 0.44 and 0.58. Among them, the smallest value is the third one, 0.44. That is, point p_x is most close to the point p_3. The value of p3 in the data set is

$$[0.9 \qquad 0.1 \qquad 0.1 \qquad 0.9 \qquad 0.8],$$

thus we set the output value as $Y_{and} = 0.1$, $Y_{or} = 0.9$ and $Y_{xor} = 0.8$ for the input value $x_1 = 0.7$ and $x_2 = 0.3$.

3.2 The sequential logic computations implemented based on the semantic space model

Sequential logic calculation is required for Spatio-temporal data processing. The Spatio-temporal data processing includes the processing of text understanding, sound recognition and motion processing, etc. We use an example case to illustrate the mechanism. In the example, we have two values s_0 and s_1 to present a state (s_0, s_1). When the state (s_0, s_1) is (0, 0), its next state is (0, 1). When the state is (0, 1), its next state is (1, 0). When the state is (1, 0), its next state goes back to (1, 0). The output in the state (1,0) is 1. The output is 0 in the states (0, 0) and (0, 1). This is illustrated in Figure 2 and summarized in Table 3.

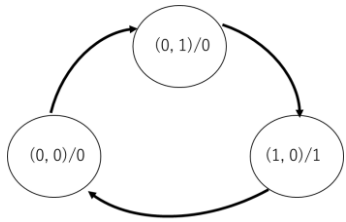


Figure 2. State transition diagram

Table 3. State transition table

S_1	S_0	next S_1	next S_0	output
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1

From Table 3, we get a matrix M,

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

When SVD is performed to the matrix of the given data set, we get three matrixes:

$$U=\begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \quad S=\begin{bmatrix} 1.41 & 0 & 0 & 0 & 0 \\ 0 & 1.41 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad V=\begin{bmatrix} 0 & -0.7 & 0 & -0.5 & -0.5 \\ -0.7 & 0 & 0 & -0.5 & 0.5 \\ -0.7 & 0 & 0 & 0.5 & -0.5 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -0.7 & 0 & 0.5 & 0.5 \end{bmatrix}.$$

The semantic space U*S is that

$$U*S=\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ -1.4 & 0 & 0 & 0 & 0 \\ 0 & -1.4 & 0 & 0 & 0 \end{bmatrix}.$$

This is a five dimensional space. As the values of the last two columns of the matrix is zero, we compress the semantic space by removing the last two rows and get a new matrix P,

$$\begin{bmatrix} 0 & 0 & -1 \\ -1.4 & 0 & 0 \\ 0 & -1.4 & 0 \end{bmatrix}.$$

In this example, the absolute value of the threshold for the semantic space compressing is set to 0.001.

Dividing each rows of matrix P, we get three mapping points of the state table in the semantic space, p_1, p_2, p_3 and p_4, represented as three vectors:

$p_1 = [0 \quad 0 \quad -1]$,
$p_2 = [-1.4 \quad 0 \quad 0]$,
$p_3 = [0 \quad -1.4 \quad 0]$.

During the processing of text understanding, sound recognition and motion processing, etc., tt is common that input data are not logical data. Suppose that the input data set is s0 = 0.2 and s1 = 0.9, we set the input vector as

[0.2 0.9 0.5 0.5 0.5].

This vector is mapped to the semantic space by multiply the vector V. the Euclidean distances of the input vector and p1, p2, p3 are 1.2, 0.8 and 1.4, respectively. The input vector is closest to p2. Therefore, we get out vector as [0 1 1 0 0]. That is, the processing state now is (0, 1). Its next state is (1, 0) and its output is 0.

4. The control mechanism and execution examples

In this section, we use a case study to illustrate the control mechanism. We suppose that there is an agent with a control unit created based on the control mechanism. The agent moves on a plane surface with obstacles. We set a start point and an end point for the agent. We define a path to as the traces of the agent moved from the start point to the end point and a matrix **X** to represent the paths and obstacles on the surface.

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ S & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & G \end{bmatrix} \quad (1).$$

This matrix **X** represents that the plane surface is divided into 16 areas. Each element of the matrix **X** represents an area. When an area is on the path, the relative value of the matrix **X** is set to '1'. If there is obstacle in the area, that is the agent cannot path through the area, the relative element of **X** is set to '0'. The area with the start point is set to 'S' and that with goal point to 'G'. In this example, the value of X(2,1) is 'S', and that of X(4,4) is 'G'. All the '1' value elements are X(2,2), X(2,3), X(2,4), X(3,2), X(3,4), X(4,2) and X(4,3).

We design a control unit with a state memory access function, a clock function, and a control signal memory access function. The state memory access function is used to access the state memory. The control unit works based on the clock function. The outputs of the clock function are execution step signals, Step-1, Step-2, Step-3, ... The control signal memory access function is used to access the control signal memory. Along with the step signals, the state memory access function and the control memory signal access function are executed. The control unit works as follows:

Step-1: Execute the state memory accessing function to obtain the next step state matrix and control signal matrix. Execute the clock function and go to Step-2.

Step-2: Execute the state interpretation function to obtain the control signals. Execute the clock function and go to Step-3.

Step-3: Executing the sate memory accessing function to store states into the state memory. Execute the clock function and go to Step-4.

Step-4: Execute the clock function. The output of this function is the next step signal. If the output is Step-1, go to Step-1. If the output is Step-4, stay in this step.

The clock function detects the states of the control unit. If a state is changed, this function will output a next step signal. Otherwise, the next step signal will be the same one of this time. This is based on ***concept of time*** in our mechanism. Here we use an example to illustrate this concept. Suppose that we have one state and a timer in a system. The time expressed by the timer in the system is t1, t2, t3, … We use T1, T2, T3, …, to express the time in the outside of the system. If we observe the state from the

outside of the system, and we find that state is change at T3, but inside of the system, the time is lasted from t1 to t2. That is, the times inside and outside of the system are different. The time lasts speed depends on the detection of state changes.

We use matrixes to present states. Suppose an agent with the control mechanism moves on the path described by equation (2).

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ S & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & G \end{bmatrix} \quad (2)$$

If the positions of the agent on the surface are defined as states, the states s_1, s_2, s_3, s_4, s_5 and s_6 are defined as:

$$s_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, s_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, s_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$s_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, s_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, s_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

If the traces of the agent on the surface are defined as states, the states s_1, s_2, s_3, s_4, s_5 and s_6 are defined as:

$$s_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, s_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, s_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$s_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, s_5 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, s_6 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

When we put the agent at the begin point, the state s_1 appears. This triggers the clock and the control unit in the agent starts to go to the Step-1 processing. Suppose that the state memory and the control signal memory are empty at the beginning. In this processing, a semantic control subspace, sp_1 is created. We call it a subspace because we refer to the space mapped all the states as the full space. On the subspace, the movement of the agent from the position (2,1) to (2,2) as shown in the equation (2) is presented as a point p_R. If we define four movement actions, "Left," "Right," "Up" and "Down" as four points in the subspace, the point p_R that presents the movement from the position (2,1) to (2,2) is interpreted as "move right" action and set close to the point "Right," as shown in Figure 3. The agent cannot move from the position (2,1) to (1,1). The reason of this is interpreted as "Block" on the subspace and a point p_U is set close to it. In the same reason, the movement from the position (2,1) to (3,1) is also interpreted as the "Block" and a point p_D is set close to it. The agent cannot move from

the start point to the "Left" direction because the start position is at edge of the plain. Thus, it is interpreted as "Forbidden" and a point p_I presenting this movement is set close it.

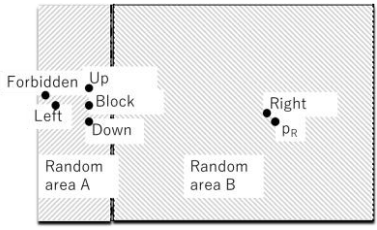


Figure 3. The semantic control subspace with interpretations and actions

During the processing of Step-1, if the state memory is not empty, the semantic control subspaces are read out from the memory. When the processing of this step is finished, the processing of Step-2 is started. In this processing, four passible action signals presenting "move left," "move up," "move right" and "move down", are randomly generated and mapped to the semantic control subspace. In the case of presented in Figure 3, "move left" is mapped close to "Forbidden," therefore, it is interpreted as to *disable action*. Same as that, both "move up" and "move down" are interpreted as to *disable action*. Only "move right" is an effective action, therefore "move right" signal is outputted. In this step, if a random generated action signal is a disable action, the action signal generation is continuing to perform until an effective action signal is generated.

The random action signal generation processing is performed based on the "random area" as shown in Figure 3. As shown in Figure 3, the area of "Random area A" is smaller that of "Random area B", the occurrence probability of the actions in "Random area A" is lower than that of in "Random area B". That is, in this example, the signal of "move right" has a high probability to be generated. In the case that the agent is at the position (2,2) as shown in equation (1), it can move to two positions, (2,3) or (3,2) mapped as p_R and p_D on the semantic subspace. In this case, the "random area" for each movement has the same area "Random area B" and "Random area C" as shown in Figure 4. Thus, the agent has the same possibility to go "right" or go "down."

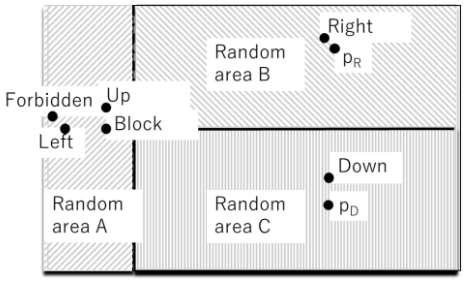


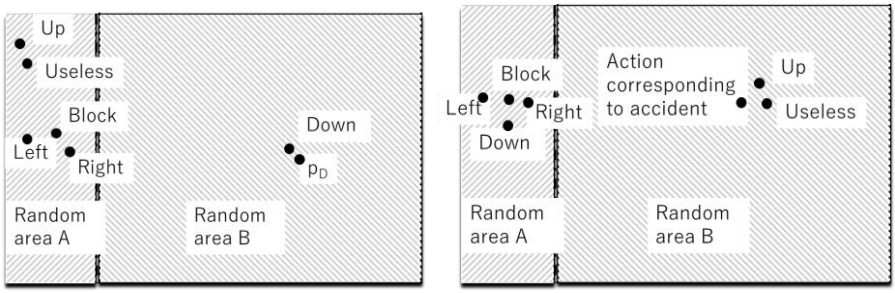
Figure 4. The semantic control subspace with two same size random areas

After the processing Step-2 is finished, the processing of Step-3 is started. In this step, the new generated semantic control subspace is stored together with the state s_1 is into the state memory.

It will be occurred that a path is blocked by some accident. For example, the path, presented by the points (2,1), (2,2), (3,2), (4,2), (4,3), (4,4) is blocked by an accident that an obstacle is appeared at place (4,2), as shown by equation (3).

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ S & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & \mathbf{0} & 1 & G \end{bmatrix} \quad (3)$$

When the agent happened moves to the position (3,2), it cannot move to the position (4,2) because there is an obstacle there. A new semantic control subspace is created as shown in Figure 5 (b). In Figure 5 (a), the semantic control subspace shows that the agent can go down from the position (3,2) to (4,2). In Figure 5 (b), it shows that the agent must be go back, that is go up from the position (3,2) to (2,2) because an action is happened that position (4,3) is blocked.

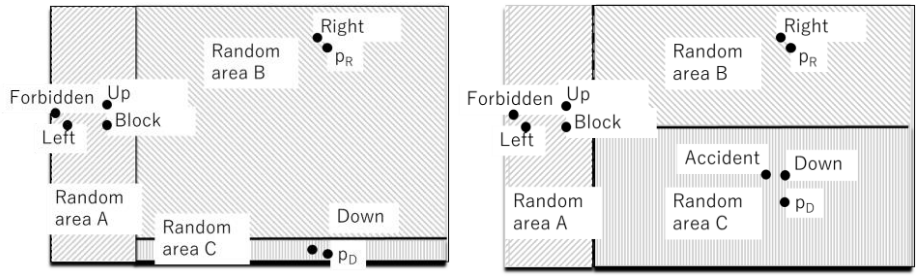


(a) Possible to move from (3,2) to (4,2)

(b) Impossible to move from (3,2) to (4,2)

Figure 5. The semantic control subspace when an accident happened

When the agent goes back to the position (2,2), a new semantic control subspace is created as shown in Figure 6. We can create two kinds of the subspace as shown in Figure 6 (a) and (b).



(a) Setting a small size of "random area C"

(b) Adding "Accident" concept close to the point "Down"

Figure 6. The semantic control subspace of s_2 when an accident happened

The first one is to set a small size of "Random area C" as shown in Figure 6 (a). In this way, the agent will move to the "right" direction in high possibility and in small possibility try to go to the "down" direction testing if the obstacle is removed out or not. The other one is to add "Accident" concept and map this concept as a point close to the point of "Down". If a "move down" signal is generated, it will be interpreted as a useless action because it is close to the "Accident" point.

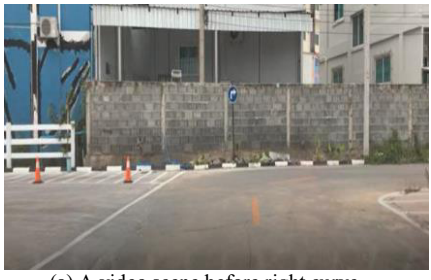
5. The demonstrate experiment and the experimental results

In the demonstrate experiment, we construct an agent for unmanned ground vehicle control with the control mechanism. We set a video camera in a car driven at a driving school. The video camera is used to determine the position of the vehicle and obstacles on the road. In our experiment, we use recorded video as inputs of the agent. The control signals, including "turn left," "turn right," "go ahead" and "stop," etc., outputted from the agent are used to demonstrate the efficiency of the mechanism.

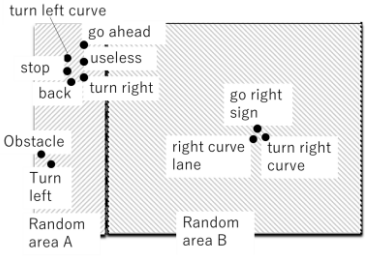
State are constructed by several basic parts, "go right sign," "straight," "forward arrow," "lane," "oncoming lane," "no entry sign," "one way sign" and "obstacle", etc., as shown in Figure 7. We constructed a convolutional neural network (CNN) to catch these parts from the recorded video. As the car moved along the lane, different parts are caught and the states are changed as the new parts are caught.

Figure 7. Parts for constructing states

The control subspaces are constructed with actions, "turn right curve," "turn left curve," "go ahead," "turn left," "turn right," "back" and "stop," and concepts same as that of the state parts. Actions and concepts are set on the control subspaces based on the driver's actions at each state. Figure 8 shows an example.



(a) A video scene before right curve

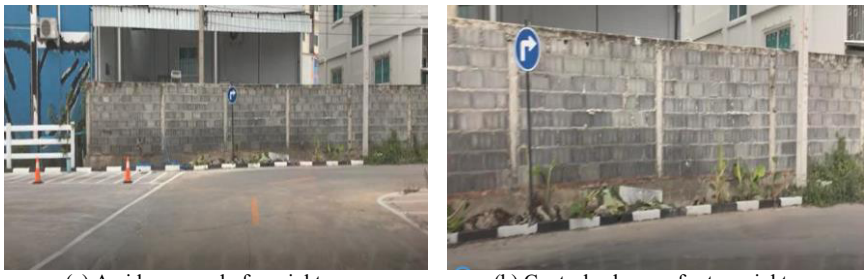


(b) Control subspace for turn right

Figure 8. An example of video scene and control subspace

Figure 8 (a) shows the scene when the car arrives at the right curve. Therefore, "turn right curve" signal should be outputted. The created control subspace is shown in Figure 8 (b). As shown in the figure, "turn right curve" signal is close "right curve lane" and "go right sign".

The state with the parts "right curve lane" and "go right sign" are used as the teacher data. Recorded video is used to train the CNN network with the teacher data. After the training, when the recorded video is played again, states are generated by the network with the video as its input. The scenes used for training and playing are different. The scene extracted from the video used for training is not used during the playing as shown in Figure 9. Figure 9 (a) is the scene used for training and (b) is the scene used in the playing.

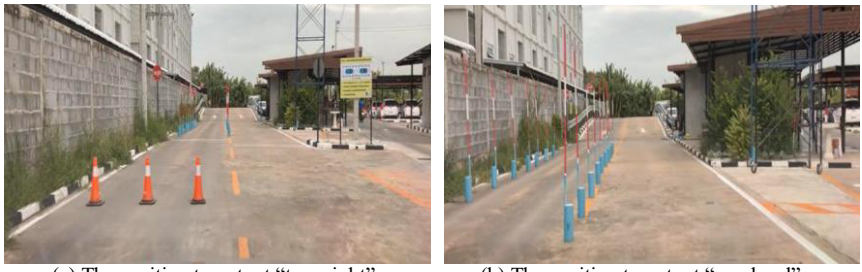


(a) A video scene before right curve (b) Control subspace for turn right

Figure 9. Parts for constructing states

We use different scene as the input of the agent to imitate it moving. We exam the output control signals of the agent to demonstrate the efficiency of the mechanism proposed in this paper. In the following, we present part of experimental results.

As shown in Figure 10 (a), there is an obstacle on the road. The agent must avoid the obstacle by turning right. It should go ahead through the "oncoming lane." Therefore, we expected that the output signal at Figure 10 (a) position is "turn right" and that at Figure 10 (b) is "go ahead." The agent correctly outputs the two signals at these two positions.



(a) The position to output "turn right" (b) The position to output "go ahead"

Figure 10. Check the output signals "turn right" and "go ahead"

In Figure 11 (a), the scene shows that the agent is in the lane of "oncoming lane." The agent must go back to the correct lane. Therefore, we expected that the output signal at Figure 11 (a) position is "turn left." In Figure 11 (b), the scene shows that there is an obstacle on the lane of "oncoming lane." As the obstacle is not on the lane

that the agent will go through, “go ahead” signal is expected. The agent correctly outputs the two signals at these two positions.

(a) The position to output “turn left”

(b) The position to output “go ahead”

Figure 11. Check the output signals “turn left” and “go ahead”

The next experimental results are shown in Figure 12, in which the agent correctly outputted two signals, “turn right curve” and “stop.” In Figure 12 (a), a state is shown that the agent is in front of a right curve. In Figure 12 (b), it is shown that the road is blocked.

(a) The state for output “turn right curve”

(b) The state for output “stop”

Figure 12. Check the output signals “turn left” and “go ahead”

In Figure 13, we present our experiment to test if the agent can correct its action from an error state. In Figure 13 (a), it can be find that at right side, there is a “no entry sign.” We set the agent in a state trying to enter the right side road. We expect that a “turn left” signal can be outputted from the agent. In our experiment, we obtained the expected result.

(a) At right side, there is a “no entry sign”.

(b) A “turn left” signal is outputted.

Figure 13. Check the output signals “turn left” and “go ahead”

When the agent arrives to the goal, we set a series actions, “slow down” and “soft-break”. The “slow down” relates the action of “easing the accelerator.” The “slow down” actions relates the action of “stepping on the brake lightly.” This action will

perform several times till the agent arrived at goal point. During our experiment, the expected signals are corrected outputted from the agent.

(a) A signal "slow down" is outputted.

(b) A "soft-break" signal is outputted.

Figure 14. Actions before and reaching "goal"

Based on the above presented experimental results we demonstrate the efficiency of the proposed mechanism.

6. Conclusion and future work

In this paper, we presented control mechanism based on the semantic space model. This mechanism is the base for implementing the programs presented by the semantic spaces, referred to as the control subspaces. It is the most important contribution of this paper. In the paper, we presented design idea of the mechanisms. A state memory is designed for storing states and control subspaces. In the mechanism, states are the input and the control subspaces are the output of the memory. Like the programs stored in the memory of computers, the control subspaces implement the required functions. We use examples to illustrate the basic idea on the control subspace construction. Another contribution of the paper is the idea of indicating control signals on the semantic space instead of coding program with program language. Using the semantic space to indicate control signals makes it possible to add both the reason and the explanation for the control signals on the space. As all those are presented as points on the space, we can implement semantic interpretation for the control signals by Euclidean distance calculation. This is the third contribution of this paper that we presented a method for constructing artificial intelligence systems based on the semantic and knowledge. We furtherly introduced the concept, ***time***, and presented the clock mechanism for the execution flow of this mechanism. In order to demonstrate the efficiency of the proposed mechanism, we performed a demonstration experiment. In the experiment, an agent is constructed for imitating an unmanned ground vehicle control with the control mechanism. We use recorded video as inputs of the agent. The control signals, like "turn left," "turn right," "go ahead" and "stop" outputted from the agent are used to demonstrate the efficiency of the mechanism. As all the expected control signals are obtained at proper times, we confirmed the efficiency of the proposed mechanism. As our future work, we will use this mechanism to implement some artificial intelligence systems to furtherly confirm the effectiveness of these mechanisms in practice.

Acknowledgment

We are particularly grateful for the assistance given by Sorawit Sirimongkol in taking video clips used in our experiments.

References

[1] Chen, X. and Kiyoki, Y., "*A query-meaning recognition method with a learning mechanism for document information retrieval*," Information Modelling and Knowledge Bases XV, IOS Press, Vol. 105, pp.37-54, 2004.

[2] Chen, X. and Kiyoki, Y., "*A dynamic retrieval space creation method for semantic information retrieval*," Information Modelling and Knowledge Bases XVI, IOS Press, Vol. 121, pp.46-63, 2005.

[3] Kiyoki, Y. and Kitagawa, T., "*A semantic associative search method for knowledge acquisition*," Information Modelling and Knowledge Bases, IOS Press, Vol. VI, pp.121-130, 1995.

[4] Kitagawa, T. and Kiyoki, Y., "*A mathematical model of meaning and its application to multidatabase systems*," Proc. 3rd IEEE International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems, pp.130-135, April 1993.

[5] Negnevitsky, M., "Artificial Intelligence: A Guide to Intelligent Systems," Pearson Education, ISBN 0-321-20466-2, Second edition published 2005

[6] Chen, X. and Kiyoki, Y., "*On Logic Calculation with Semantic Space and Machine Learning*," Information Modelling and Knowledge Bases XXXI, IOS Press, Vol. 321, pp.324-343, 2019.

[7] Chen, X., Kiyoki, Y. and Kitagawa, T., "*A multi-language oriented intelligent information retrieval system utilizing a semantic associative search method,*" Proceedings of the 17th IASTED International Conference on Applied Informatics, pp.135-140, 1999.

[8] Chen, X., Kiyoki, Y. and Kitagawa, T., "*A semantic metadata-translation method for multilingual cross-language information retrieval,*" Information Modelling and Knowledge Bases XII, IOS Press, Vol. 67, pp.299-315, 2001.

[9] Kiyoki, Y., Kitagawa, T. and Hitomi, Y., "*A fundamental framework for realizing semantic interoperability in a multidatabase environment*, " International Journal of Integrated Computer-Aided Engineering, Vol.2, No.1(Special Issue on Multidatabase and Interoperable Systems), pp.3-20, John Wiley & Sons, Jan. 1995.

[10] Kiyoki, Y., Kitagawa, T. and Hayama, T., "*A metadatabase system for semantic image search by a mathematical model of meaning*, " ACM SIGMOD Record, Vol.23, No. 4, pp.34-41, Dec. 1994.

[11] Kiyoki, Y, Chen, X. and Kitagawa, T., "*A WWW Intelligent Information Retrieval System Utilizing a Semantic Associative Search Method*," APWeb'98, 1st Asia Pacific Web Conference on Web Technologies and Applications, pp. 93-102, 1998.

[12] Ijichi, A. and Kiyoki, Y.: "*A Kansei metadata generation method for music data dealing with dramatic interpretation*," Information Modelling and Knowledge Bases, Vol.XVI, IOS Press, pp. 170-182, May, 2005.

[13] Kiyoki, Y., Chen, X. and Ohashi, H.: "*A semantic spectrum analyzer for realizing semantic learning in a semantic associative search space*," Information Modelling and Knowledge Bases, Vol.XVII, IOS Press, pp.50-67, May 2006.

[14] Takano, K. and Kiyoki, Y.: "*A causality computation retrieval method with context dependent dynamics and causal-route search functions*," Information Modelling and Knowledge Bases, ISO Press, Vol.XVIII, pp.186-205, May 2007.

[15] Chen, X. and Kiyoki, Y.: "*A visual and semantic image retrieval method based on similarity computing with query-context recognition*," Information Modelling and Knowledge Bases, IOS Press, Vol.XVIII, pp.245-252, May 2007.

[16] Nitta T, "*Resolution of singularities introduced by hierarchical structure in deep neural networks*," IEEE Trans Neural Netw Learn Syst., Vol.28, No.10, pp.2282-2293Oct. 2017.

[17] Wiatowski, T. and Bölcskei, H., "*A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction*," IEEE Transactions on Information Theory, PP(99) · Dec. 2015.

[18] Hochreiter, S., Bengio, Y., Frasconi, P. and Schmidhuber, J. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer, S. C. and Kolen, J. F. (eds.), *A Field Guide to Dynamical Recurrent Neural Networks*," IEEE Press, 2001.

[19] Hochreiter, S. and Schmidhuber, J., "*Long short-term memory*,". Neural computation, Vol.9, No.8, pp.1735-1780, 1997.

[20] Kalchbrenner, N., Danihelka, I. and Graves, "*A. Grid long short-term memory*," CoRR, abs/1507.01526, 2015.