Basic study toward automatic generation of glove-type command input device with optimal number of sensors

Y Tabata[1], T Kuroda[2], K Okamoto[3]

[1]Department of Radiation Technology, Kyoto College of Medical Science,
1-3 Imakita, Oyama-higashi, Sonobe, Nantan, JAPAN

[2,3]Department of Medical Informatics, Kyoto University Hospital,
54 Kawahara-cho, Shogo-in, Sakyo-ku, Kyoto, JAPAN

[1]*yoshi-t@kyoto-msc.jp*, [2]*tomo@kuhp.kyoto-u.ac.jp*, [3]*kazuya@kuhp.kyoto-u.ac.jp*

[1]*www.kyoto-msc.jp*, [2,3]*www.kuhp.kyoto-u.ac.jp*

ABSTRACT

Data-gloves are one of the most essential devices for VR systems. Although most of conventional data-gloves are designed to capture “analogue” hand postures, most of information systems require just “digital” hand postures corresponding to commands. This paper proposed a method to calculate a data-glove with optimal number of sensors to obtain given set of “digital” hand postures. The authors applied the proposed method to JSL. The result tells that a data-glove with just eight sensors can obtain all hand postures given in JSL.

1. INTRODUCTION

The glove-type input device (data-glove), which can measure hand postures, is one of the most essential devices to develop virtual reality (VR) environments (Dipietro et al, 2008). Although, most of foregoing data-gloves are designed to capture hand postures as they are, most of information systems quantize obtained hand postures to know user’s intention as commands. For example, the first VR development environment RB2 maps several certain hand postures to several commands such as “go forward” or “go backward”. Sign language translators are other typical example. Except for pure motion capture systems, no information system requires “analogue” hand posture.

The authors have developed a data-glove named “StrinGlove” (Kuroda et al, 2004), which can quantize given postures into a set of notation codes by itself. However, as the authors designed StrinGlove referring to conventional data-gloves, StrinGlove doesn’t have enough sensors to capture certain hand postures given in sign languages. On the other hand, StrinGlove seems overengineered device for certain applications, such as VR CAD. As prices of data-gloves are defined simply by number of sensors in many cases, a data-glove should be designed with minimum number of sensors to distinguish a set of “digital” hand postures required for the targeting application.

This paper proposes a method to provide optimal number of sensors to distinguish a given set of “digital” hand postures.

2. RELATED WORKS

2.1 *Data-gloves*

Hand is most common output device of human beings to manipulate external space directly. Therefore, the many foregoing researches have been trying to emerge direct manipulation environment using human hand. Although several researches such as Iwai et al (1996) and Wang and Popović (2009) tried to obtain hand postures from their outlook, most of researches developed sensor unit to be put on. Such sensor unit, commonly called data-gloves, are widely known as one of the three indispensable devices for VR.

After invention of VPL Dataglove (Zimmerman et al, 1982), many data-gloves have been developed. As most researches assume that combinations of joint angles emerges any hand postures, most of data-glove measures joint angles by various sensors, such as optical-fibers, piezo-registers, carbon ink, etc. However, as motions around metacarpal joints (MPJ) and trapeziometacarpal joint (TMJ) is far more complicated from common hand joint model shown in Figure 1, no existing data-glove can obtain detailed hand posture as it is, even they are designed to do so. Some data-gloves like Pinch Glove equip sensors to obtain contact of fingertips. LaViola (1999) claimed that a glove with only contact sensors could obtain over 1000 postures.

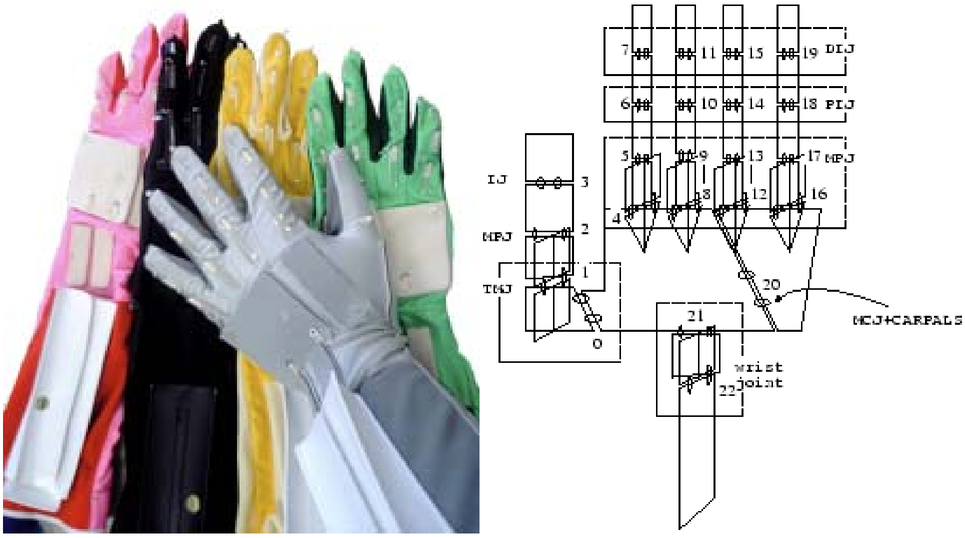

Figure 1. *StrinGlove® (left) and its joint model (right)*

Although some applications, such as motion capture for animations or motor skill evaluation for rehabilitation requires “analogue” hand posture as it is, most of applications intend to obtain “digital” hand posture as a command as the first VR development environment RB2. New data-glove applications, such as wearable computers or motion-based games, also quantize given hand postures into a set of command words. Sign language translators are other typical and conventional example. According to the review by Dipietro et al (2008), most of conventional and new applications of data-gloves can be replaced by keyboards and mice, that is command input devices and pointing devices. Augmented Reality based input systems, such as HIT-wear (Sasaki et al, 2006) and SixthSense (Mistry and Maes 2009), give another typical hand-gesture-based input interface to obtain commands.

2.2 Sign Notations

Sign language is a group of most sophisticated coding system of human motion. Thus, we may refer sign notation codes as a “gold standard” of hand posture.

Foregoing researches tried to develop sign notation systems. Stokoe et al (1965) is the first and most well known trial to define phoneme of sign languages. They claimed that phoneme of any sign languages consisted of “Tab” (location), “Dez” (hand posture) and “Sig” (movement), and denoted each identical element by their original characters. Thus, as each hand posture is denoted by a single notation character, to know detailed posture of each finger from the notation code is rather difficult. Many following researches either simply used the Stokoe’s notation characters or invented their own characters as SignWriting (Sutton, 2009) did.

On the other hand, HamNoSys (Hanke 2004) and researches on non-verbal or motion-based human computer interaction, such as Kurokawa (1992), denoted hand posture as a set of finger postures. These systems denote a hand posture as a combination of contacts of fingers and bending angles of each joint.

As most of foregoing notation systems classifies hand posture from subjective point of view, the notations must reflect human perception of hand posture. Thus, the detailed analysis of the notations may give good insights of human perception model. An analysis tells that most of notation systems classify bending angle of each joint into three levels (full bend, weak bend, and stretch). This finding also agrees conventional pathological insights. As two muscles sandwiching the joint controls the joints, the three postures can be encoded as combination of statuses (relaxation and contractions) of the muscles.

3. FINDING MINIMUM REQUIRED SENSORS

3.1 Overview of proposed method

As discussed in 2.2, human hand posture intentionally given can be denoted as combination of rough bending angles and contacts between fingers. Therefore, a data-glove equips a contact sensor on each segment of fingers and bending sensors on each joint as shown in Figure 2 could obtain any "digital" hand postures. Here, the contact sensors are not simple switches or pressure sensors to know whether a sensor pressed by another (anonymous) segment, but coupling sensors to tell which (identified) segment touches the sensor. For example, StrinGlove (Kuroda, 2004) equips coupling sensors, consists of small coils to emit wireless signal to give identifier and small coils to receive the identifier on fingertips

However, a single application may not require whole hand posture human beings can express; an application may need a set of certain selected postures. For example, a sign language (like ASL or Japanese Sign Language (JSL)) uses only limited number of hand postures. A set of required hand postures would define a set of required sensors.

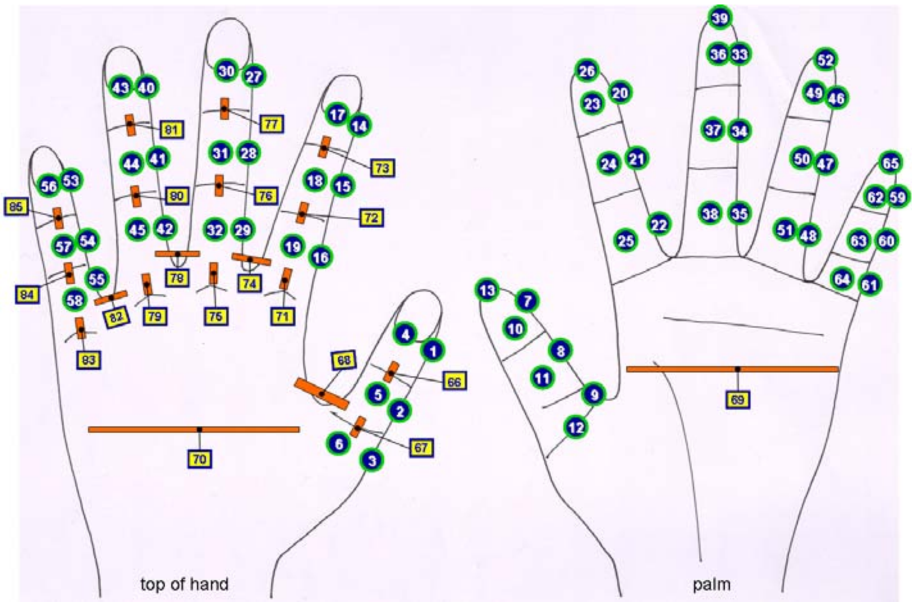

Figure 2. *Sensor Arrangement of Fully Furnished Data-glove*

Once fully furnished data-glove is given, a hand posture (code) can be denoted as a set of sensor values (code-sensor relation table). When two identical codes are given, the difference between given two sets of sensor values tells which sensor is indispensable to distinguish given two hand postures. Thus, the problem to find a set with minimum number of indispensable sensors for a given set of codes must be reduced into a mathematical problem to find a set with minimum number of items to cover given universe, widely known as set-cover problem. The following sections give details of the method.

3.2 Creating Code-Sensor Relation Table

The fully furnished data-glove given in Figure 2 equips 85 sensors; 65 contact sensors (no.1 to 65) and 20 bending sensors (no. 66 to 85). As each contact sensor is a coupling sensor, sensor output becomes multi-value data consists of status of contact (0: non-contact, 1: contact) and an ID of touching sensor. On the other hand, output of bending sensor is status of bending angle (0: full-stretch, 1: weak-bend, 2: full-bend).

Under above discussed notation system, each hand posture (code) of a given set is denoted as a set of 85 sensor values of the data-glove as shown in Table 1. When we plot only statuses of contacts and bending angles, Table 1 becomes Table 2. Here, Table 2 actually provides a set of sensor output when the contact sensors are simple switches. Therefore, when all row of Table 2 is distinct, the given set of codes can be distinguished by a glove equips simple switches instead of coupling sensors. Table 2 tells that a data-glove equips simple switches can distinguish case A not case B.

3.3 Reduction to the Set Cover Problem

If all the codes are not identified with simple switches, a brute-force method seems required because the relation between the number of selected sensors and a set of distinguishable codes is complicated. On the other hand, if all the codes are identified with simple switches, we can reduce the proposed optimizing problem to the set cover problem. Then, the problem can be solved via algorithms for the set cover problem.

The set cover problem is one of the most famous NP-complete problems (Karp, 1972). An instance (U, V) of the set cover problem consists of a set U of n elements and of a set V={V_1, …, V_m} where V_1, …, V_m are

subsets of U. The set cover problem is to find the minimum collection of subsets $V'(\subseteq V)$ whose union is equal to U. Since the set cover problem is NP-complete, it is hard to solve it in polynomial time. We can refer a brute-force algorithm and a dynamic programming algorithm as exponential time algorithms to solve the problem though there is unfortunately much fewer studies on exponential time algorithms than on approximation algorithms (Johnson, 1974; Lovász, 1975; Slavík, 1997; Feige, 1998; Goldschmidt, 1993).

Table1. *Code-Sensor Relation Table (with coupling sensors)*

Case A

No.	hand shape (example)	sensor number													sensor number													sensor number									
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	α	0	0	0	0	0	0	0	0	0	1(17)	0	0	0	0	0	0	1(10)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	β	0	0	0	0	0	0	0	0	0	1(17)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1(27)	0	0	0	0	0	0	0	0	0
3	γ	0	0	0	0	0	0	0	0	0	1(26)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1(10)	0	0	0	0	0	0	0	0	0	0

Case B

No.	hand shape (example)	sensor number													sensor number													sensor number									
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	■	0	0	0	1(20)	0	0	0	0	0	1(27)	0	0	0	0	0	0	0	0	0	1(4)	0	1(29)	0	0	0	0	1(10)	0	1(22)	0	0	0	0	0	0	0
2	●	0	0	0	1(22)	0	0	0	0	0	1(29)	0	0	0	0	0	0	0	0	0	1(27)	0	1(4)	0	0	0	0	1(20)	0	1(10)	0	0	0	0	0	0	0
3	▲	0	0	0	0	0	0	0	0	0	1(26)	0	0	0	0	0	0	1(36)	0	0	0	0	0	0	0	0	1(10)	0	0	0	0	0	0	0	0	0	1(17)

Table2. *Code-Sensor Relation Table (with simple switches)*

Case A

No.	hand shape (example)	sensor number													sensor number													sensor number									
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	α	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	β	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3	γ	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Case B

No.	hand shape (example)	sensor number													sensor number													sensor number									
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	■	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0
2	●	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0
3	▲	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1

3.4 Reduction to the Set Cover Problem

If all the codes are not identified with simple switches, a brute-force method seems required because the relation between the number of selected sensors and a set of distinguishable codes is complicated. On the other hand, if all the codes are identified with simple switches, we can reduce the proposed optimizing problem to the set cover problem. Then, the problem can be solved via algorithms for the set cover problem.

The set cover problem is one of the most famous NP-complete problems (Karp, 1972). An instance (U, V) of the set cover problem consists of a set U of n elements and of a set $V=\{V_1, \ldots, V_m\}$ where $V_1, \ldots, V_m$ are subsets of U. The set cover problem is to find the minimum collection of subsets $V'(\subseteq V)$ whose union is equal to U. Since the set cover problem is NP-complete, it is hard to solve it in polynomial time. We can refer a brute-force algorithm and a dynamic programming algorithm as exponential time algorithms to solve the problem though there is unfortunately much fewer studies on exponential time algorithms than on approximation algorithms (Johnson, 1974; Lovász, 1975; Slavík, 1997; Feige, 1998; Goldschmidt, 1993).

The reduction from the proposed problem to the set cover problem is as follows. Recall that the input of an instance of the proposed problem is a code-sensor relation table. A row and a column of the table are corresponding to a code and a sensor, respectively. The first step is to create a pairwise code-sensor relation table from the original code-sensor relation table. Let all the codes and all the sensors be $c_1, \ldots, c_p$ and $s_1, \ldots, s_q$, respectively. A row and a column of the pairwise code-sensor relation table are corresponding to a pair of codes $\{c_i, c_j\}$ $(1 \le i < j \le p)$ and a sensor s_k $(1 \le k \le q)$, respectively. The value of table element $(\{c_i, c_j\}, s_k)$ is 1, if code c_i and code c_j are distinguished by sensor s_k. Otherwise, the value is 0. Table 3 shows the pairwise code-sensor relation table of case A in Table 2.

Table 3. *The pairwise code-sensor relation table of case A*

No.	pairwise hand shape	sensor number													sensor number													sensor number									
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	$\alpha-\beta$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
2	$\alpha-\gamma$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	$\beta-\gamma$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0

The second step is to make an instance (U, V) of the set cover problem. Each pair of codes is regarded as an element of U, and the subset of pairs in which a certain sensor's value is 1 is regarded as an element of V. For example, in Table 3, as the sensor no.17 distinguishes code α and β (the pair no.1) and code α and γ (the pair no.2), the set of pairs {1, 2}, namely V_{17}, becomes an element of V. The subset corresponds to a group of code pairs that can be distinguished by a single sensor no.17. The reduction from the proposed problem to the set cover problem is complete because finding the minimum collection of subsets $V'(\subseteq V)$ whose union is U equals to finding the sensors of the minimum number which distinguish all the pairs of codes. For example, in Table 3, the answer is $\{V_{17}, V_{26}\}$, $\{V_{17}, V_{29}\}$ or $\{V_{26}, V_{29}\}$.

3.5 Solving the Set Cover Problem

Exact algorithms for the set cover problem take exponential time to solve an instance. The more subsets of an instance there are, the more run-time increases. Therefore, it is important to decrease the solution space of an instance. The first step of the preprocessing to decrease the solution space is to omit some sensors, whose corresponding columns do not include 1s, i.e., these sensors are not contributing. The second step of the preprocessing is to omit sensors whose corresponding columns are identical to another sensor, i.e., these sensors are replaceable.

The third step, the main procedure, is to solve the reduced instance using the exponential time algorithms. If the exponential time algorithm cannot solve it in realistic time, approximation algorithms are applied. If $P \neq NP$, the worst-case approximation ratios of approximation algorithms are known as θ (log m) (m is the number of subsets) (Johnson, 1974; Lovász, 1975; Raz, 1997), which means that a solution of the approximation algorithms can be O (log m) times as large as the optimal solutions. However, the approximation ratio is close to 1 experimentally (Gomesa, 2006).

4. EVALUATIONS

4.1 Target dataset

The authors selected JSL as a source of target dataset. Takemura (1999) and Kanda (2010) examined hand postures given in JSL and found that 50 hand postures shown in Figure 3 can express at least 600 JSL words. Thus the authors selected the 50 hand postures as a target dataset. Another target dataset is 28 hand postures surrounded by dotted line, which are shown in Japanese finger characters.

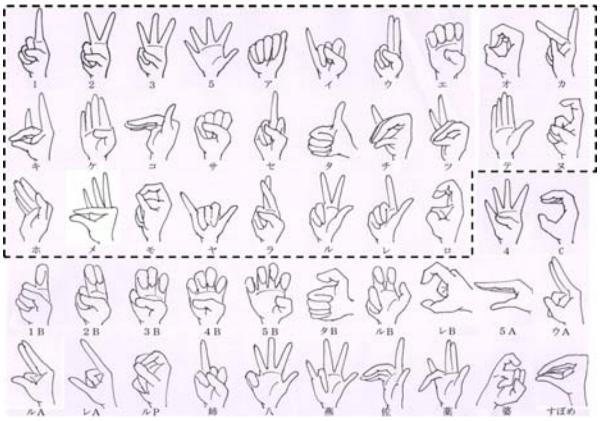

Figure 3. *50 hand shapes of JSL and their notation codes.*

4.2 Result

At first, the authors developed code-sensor relation tables of JSL and Japanese finger spelling. Table 4 shows the code-sensor relation table of Japanese finger spelling. Recall that sensors from No.1 to No.65 are contact sensors and sensors from No.66 to No.85 are bending sensors.

As all the rows of both tables were distinct, the authors picked only contact status of contact sensors and constructed pairwise code-sensor relation tables. The preprocessing to omit non-contributing sensors omitted 34 sensors in the case of JSL and 35 sensors in the case of Japanese finger characters. The second step to omit replaceable sensors omitted 10 sensors in JSL and 11 sensors in Japanese finger characters.

After preprocessing, the authors tried to solve the reduced instances of set cover problem using an exponential time algorithm that takes a great deal of time. However, the authors could get optimal solutions in both cases because the solution space was relatively small. The minimum number of sensors to distinguish all the codes in JSL is eight and there are 133 combinations. On the other hand, the minimum number of sensors to distinguish all the codes in Japanese finger character is six and there are just 2 combinations; $\{V_{35}, V_{66}, V_{68}, V_{72}, V_{74}, V_{85}\}$ and $\{V_{35}, V_{67}, V_{68}, V_{72}, V_{74}, V_{85}\}$.

Table 4. *The code-sensor relation table of Japanese finger characters.*

No.	hand shape of notaion code	sensor number																
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…	85
1	1	0	0	0	0	0	0	0	0	0	1(30)	0	0	0	0	0		2
2	2	0	0	0	0	0	0	0	0	0	1(43)	0	0	0	0	0		2
3	3	0	0	0	0	0	0	0	0	0	1(56)	0	0	0	0	0		1
4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
5	ア	0	0	0	0	0	0	0	0	0	1(16)	0	0	0	0	0		2
6	イ	0	0	0	0	0	0	0	0	0	1(30)	0	0	0	0	0		0
7	ウ	0	0	0	0	0	0	0	0	0	1(43)	0	0	0	0	0		2
8	エ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		2
9	オ	0	0	0	0	0	0	1(26)	0	0	0	0	0	1(39)	0	0		1
10	カ	0	0	0	0	0	0	0	0	0	1(37)	0	0	0	0	0		0
11	キ	0	0	0	0	0	0	0	0	0	1(36)	0	0	0	0	0		0
12	ケ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
13	コ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
14	サ	0	0	0	0	0	0	0	0	0	1(43)	0	0	0	0	0		2
15	セ	0	0	0	0	0	0	0	0	0	1(43)	1(17)	0	0	0	0	…	1
16	タ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		2
17	チ	0	0	0	0	0	0	1(23)	0	0	1(36)	0	0	0	0	0		0
18	ツ	0	0	0	0	0	0	1(23)	0	0	1(36)	0	0	0	0	0		0
19	テ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
20	ヌ	0	0	0	0	0	0	0	0	0	1(43)	0	0	0	0	0		2
21	ホ	0	0	0	0	0	0	0	0	0	1(16)	0	0	0	0	0		1
22	メ	0	0	0	0	0	0	0	0	0	1(23)	0	0	0	0	0		1
23	モ	0	0	0	0	0	0	0	0	0	1(23)	0	0	0	0	0		1
24	ヤ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
25	ラ	0	0	0	0	0	0	0	0	0	1(43)	0	0	0	0	0		1
26	ル	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		1
27	レ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		1
28	ロ	0	0	0	0	0	0	0	0	0	1(43)	0	0	0	0	0		1

4.3 Discussions

The dramatic decrease of required sensors for JSL and Japanese sign characters indicates that the proposed method is capable as the method to select indispensable sensors.

There were 133 sets of minimum numbers of sensors to distinguish JSL. The best one out of 133 set may be defined from prices of sensors, manpower to mounting sensors on glove, and favor of users. Although waited set-patter problem seems applicable, the parameters to determine the weight of each sensors, such as required manpower, are not independent each other. Manual process to select a best set according to given criteria may be required.

The sensor no.73 is in all 133 sets. The fact means the sensor is indispensable to distinguish 50 hand postures given in JSL. The detailed analysis clears that the sensor is the bending sensor to measure angle of distal interphalangeal joint (DIJ) of index finger and is indispensable to distinguish two hand postures shown in Figure 4. Most of conventional data-gloves don't equip a sensor to measure bending of DIJ under a hypothesis that angle of DIJ is always proportional to angle of proximal interphalangeal joint (PIJ) and that DIJ is not important to get hand postures. The result denies the hypothesis of foregoing data-gloves. The fact clearly confirms the effectiveness of the proposed method.

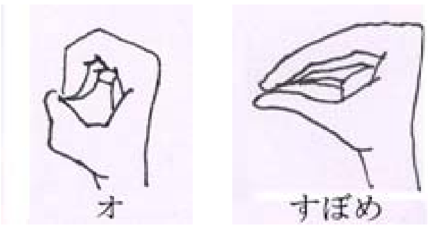

Figure 4. *Hand postures distinguished by the sensor no. 73.*

In this research, the authors start from a code-sensor relation table where a single code defined as a single combination of sensor values. However, the "digital" hand posture cannot be always defined as a specific combination of sensor values. Subjective definition of hand posture cannot be so rigid. To accept variations for each code, we need to develop several rows for a certain code in code-sensor relation table. Although we need to omit rows of pair of identical codes on the pairwise code-sensor relation table, the same proposed algorithm will provide sets of optimum number of sensors for a given set of codes.

5. CONCLUSIONS

This paper proposed a method to work out a set of minimum number of indispensable sensors to distinguish a given set of hand postures by reducing the problem into simple set-cover problem. The developed method clears that a data-glove to distinguish JSL requires only eight sensors, and a data-glove to distinguish Japanese finger characters requires only six sensors. The developed method enables us not only to produce low-cost data-gloves for each use cases, but also to discuss the communication bandwidth of gesture communication of human beings.

Acknowledgements: The authors would like to thank Dr. Shinobu Kawagishi of Sakai City Office and Prof. Kazuyuki Kanda of Chukyo University for their support for defining hand posture set of JSL. The authors also would like to thank Teiken Limited, Fujita Corp. and AMITEQ Corp. for their continuous support to develop StrinGlove®.

6. REFERENCES

L. Dipietro, A.M. Sabatini and P. Dario (2009), A Survey of Glove Based Systems and Their Applications, *IEEE Trans. Man. Cyber.,* **38**, *4*, pp.461-482.

O. Goldschmidt, D. S. Hochbaum and G. Yu (1993) A Modified Greedy Heuristic for the Set Covering Problem with Improved Worst Case Bound, *Info. Proc. Lett.*, **48**, pp.305-310.

F.C. Gomesa, C.N. Menesesb, P.M. Pardalosb and G.V.R. Viana (2006) Experimental Analysis of Approximation Algorithms for the Vertex Cover and Set Covering Problems, *Comp. & Ope. Res.*, **33-12**, pp.3520-3534.

T Hanke (2004) HamNoSys - Representing Sign Language Data in Language Resources and Language Processing Contexts, *Proc. LREC*, pp.1-6.

Y. Iwai, K. Watanabe, Y. Yagi and M. Yachida (1996) Gesture Recognition using Colored Gloves, *Proc. ICPR*, **1**, pp.662-666.

D.S. Johnson (1974) Approximation Algorithms for Combinatorial Problems, *J. Comp. Sys. Sci.*, **9**, pp.256-278.

K.Kanda (2010), *A Study of Linguistic Characteristics of Japanese Sign Language*, Fukumura Press, Tokyo, *Japanese*.

R.M. Karp (1972) Reducibility Among Combinatorial Problems, In *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, Eds.), Plenum Press, New York, pp. 85-103.

T. Kuroda, Y. Tabata, A. Goto, H. Ikuta and M. Murakami (2004) Consumer Price Data-Glove for Sign Language Recognition, *Proc. ICDVRAT*, pp.253-258.

T. Kurokawa (1992) Gesture Coding and a Gesture Dictionary for a Nonverbal Interface, *IEICE Trans. Fund. Elect.*, **75**, pp.112-121.

J.J. LaViola (1999) A Survey of Hand Posture and Gesture Recognition Techniques and Technology, *Tech. Rep. of Brown Univ.*, CS-99-11.

L. Lovász (1975) On the Ratio of Optimal Integral and Fractional Covers, *Discrete Math.*, **13**, pp.383-390.

P. Mistry and P. Maes (2009) SixthSense – a Wearable Gesture Interface. *Proc. SIGGRAPH Asia*.

R. Raz and S. Safra (1997) A Sub-Constant Error-Probability Low-Degree Test, and Sub-Constant Error-Probability PCP Characterization of NP, *Proc. STOC*, pp.475-484.

H. Sasaki, T. Kuroda, P. Antoniac, Y. Manabe and K. Chihara (2006) Hand-Menu System: a Deviceless Virtual Input Interface for Wearable Computers, *CEAI J.*, **8**, *2*, pp.24-33.

P. Slavík (1997) A Tight Analysis of the Greedy Algorithm for Set Cover, *J. Algo.*, **25**, pp.237-254.

W.C. Stokoe, D.C.Casterline and C.G. Croneberg (1965) *A Dictionary of American Sign Language on Linguistic Princeples*, Gallaudet College Press, Washington DC.

V. Sutton (2009) *SignWriting*, The SignWriting Press, Lo Jolla.

S.Takemura(1999), *Shuwa-Nihongo-Daijiten*, Kosaido Press, Tokyo, *Japanese*.

R.Y. Wang and J. Popović (2009), Real-Time Hand-Tracking with a Color Glove, *Proc. SIGGRAPH*, **28-3**, p.*63*.

T.G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson and Y. Harvill (1987) A Hand Gesture Interface Device, *Proc. SIGCHI*, pp.189-192.