Pattern Recognition Pergamon Press 1971. Vol. 3, pp. 297–306. Printed in Great Britain

Reduction of the Storage Requirements of Bledsoe and Browning's *n*-tuple Method of Pattern Recognition

J. R. ULLMANN

Division of Computer Science, National Physical Laboratory, Teddington, Middlesex, England

(*Received* 21 *September* 1970 *and in revised form* 5 *April* 1971)

Abstract—Random superimposed coding has reduced the massive storage requirements of the Bledsoe and Browning Method of Pattern Recognition, applied to unconstrained hand-printed numerals with $n = 14$, by a factor of roughly four. A fourfold reduction in storage area can also be achieved by the use of associative memory, but at higher cost per bit. A third approach aims to achieve economy by exploiting any non-randomness of stored n-tuple states, but this is discussed only in outline.

1. INTRODUCTION

IN SMALL scale experiments reported previously,[10] BLEDSOE and BROWNING'S n-tuple method has been found to achieve about 92 per cent successful recognition of unconstrained hand-printed numerals, at the cost of about 14 million storage bits. It is possible that performance could be substantially improved by

(a) Using a very much larger training set, and carefully selected n-tuples.
(b) Employing edge detection preprocessing, as used for example by MUNSON, DUDA and HART.[6]
(c) Changing the binarization threshold and/or alignment of a rejected character, and making a further attempt at recognition.

At the same time, if large training sets were available, the best recognition performance would be obtained with large n, e.g. $n = 14$, and the storage required would be costly. If the store were organized as specified in BLEDSOE and BROWNING'S original paper,[1] the cost of storage could be mitigated by the fact that ordinary random access high speed computer store could be used. In applications where a pattern recognition machine and a general purpose digital computer were each required to work a limited number of hours per day, the same store could be used by both machines. This suggestion would come nearer to economic practicality if the storage requirements of the Bledsoe and Browning system could be substantially reduced. Reduction is in fact possible because it is possible to reduce the redundancy of the stored information.

In the hope of achieving a big reduction in storage requirements we have adapted and applied Calvin Mooers' idea of *random superimposed coding*,[3,4] otherwise known as *zatocoding*. Using this idea we can still use conventional computer memory, but, as will be explained below, a roughly equal reduction in storage can be achieved in the $n = 14$ case by the use of a rudimentary associative memory. The associative memory cannot straightforwardly be used as the high speed store of an ordinary digital computer, and its cost per bit is higher.

Experimentally we have not investigated modifications such as (a), (b) and (c) above, which are anyway well known. Instead we have worked directly with unconstrained unpreprocessed hand-printed numerals.

2. REDUCTION OF STORAGE REQUIREMENTS BY RANDOM SUPERIMPOSED CODING

For clarity the random superimposed coding technique is introduced in three stages.

Stage 1: *Coding of classes*

For recognizing v recognition classes, let us consider a non-weighted Bledsoe and Browning system employing altogether M n-tuples, where n is the number of pattern element locations constituting an n-tuple. Bledsoe and Browning's original system requires M blocks of 2^n storage words, each composed of v bits. Let us denote any unknown binary pattern by X, and the ith pattern in the rth class training set by X_{ri}. For all j from 1 to M, let us denote by $B_j(X)$ the word, in the block of 2^n words associated with the jth n-tuple, whose address is the state of the jth n-tuple in the pattern X.

Bledsoe and Browning's system also requires a set of v words $C_1, \ldots, C_r, \ldots, C_v$, each of v bits. No two of these words are the same, and each contains one 1 and $v-1$ 0's. In this work logical operators which are applied to words are in fact applied to corresponding bits, as illustrated for four bit words:

Words	Operator	Result
1010, 0011	&	0010
1010, 0011	$\vee$	1011
1010, 0011	$\not\equiv$	1001

At the start of "training" of the Bledsoe and Browning system the M blocks of 2^n words are all set to contain only zero's. Training consists of working once through all training set patterns for all classes, carrying out an operation typified here in terms of the ith training set pattern in the rth recognition class. The operation is: for all n-tuples, that is, for all j, update $B_j(X_{ri})$ according to

$$B_j(X_{ri}) = B_j(X_{ri}) \vee C_r. \tag{1}$$

To recognize an unknown pattern X, for each recognition class in turn the system counts the number of n tuples for which

$$B_j(X) \,\&\, C_r = C_r \tag{2}$$

and X is assigned to the recognition class for which this number is largest. If more than one class has the largest count, that is, there is a tie, X is rejected.

The first-stage zatocoding system is a simple modification of this. The number of bits in each word in the block of 2^n words associated with each n-tuple is reduced from v to a smaller number u. The number of bits in each of the words $C_1, \ldots C_v$ is also reduced from v to u. Furthermore, the contents of the words $C_1, \ldots C_v$ are altered as follows. No two of

these words are the same, and each contains $u-z$ 0's and z 1's in randomly chosen bit positions. The small number z is the same for all $C_1, \ldots C_v$.

As in Bledsoe and Browning's original system, training proceeds in accordance with (1), and recognition in accordance with (2). The advantage of the zatocoding modification is that it reduces the number of storage bits required in the proportion $u:v$. But it has the disadvantages of somewhat increased complexity and decreased reliability.

To illustrate the source of unreliability let us consider an (unrealistically small) example in which $u = 8$, $z = 3$, $n = 6$, and

$$C_6 = 01100010,$$
$$C_7 = 00101001,$$
$$C_8 = 01000011,$$

these three words corresponding to the numerals "6", "7" and "8", respectively. Let us consider for example the state 010110 of the third n-tuple. Suppose that this state has occurred in at least one "6" and at least one "7" in the training set, but not in any "8" in the training set. From (1) it follows that, after training, the word, in the block of 2^n words corresponding to the third n-tuple, whose address is 010110, will contain the 1's

$$-11-1-11, \text{ where "}-\text{" denotes "either 0 or 1".}$$

Let us now consider the recognition of an unknown pattern in which 010110 is the state of the third n-tuple. Expression (2) will be satisfied for C_8 since

$$-11-1-11 \,\&\, 01000011 = 01000011,$$

and a "1" will erroneously be added to the 8's count. The reason that this is an error is that the state 010110 has not occurred on the third n-tuple in any training set pattern belonging to the class "8". This error could not possibly occur in the original Bledsoe and Browning system.

In practice errors of this sort can be made fairly rare by careful choice of z and u, and the trade-off between store size and error rate has to be determined experimentally. The overall recognition decision is based on votes from *all* n-tuples, and spurious votes from a *few* n-tuples may not always be disasterous.

Stage 2: *Coding of* n-*tuples as well as classes*

Just for recognizing the ten numerals, stage 1 zatocoding would not be worthwhile. But zatocoding is more useful when we zatocode the n-tuples as well as the classes.

The stage 2 system does *not* employ 2^n words for each n-tuple. Instead it employs only *one* block of 2^n words, but each word in this block now has w bits, where w is greater than the number of recognition classes. For instance, in recognizing the ten numerals we have worked with $w = 48$. Let us denote by $D_j(X)$ the word, in the single block of 2^n words, whose address is the state of the jth n-tuple in X. $D_j(X)$ and $D_{j+1}(X)$ are now words in the *same* block of 2^n words, whereas in the stage 1 system $B_j(X)$ and $B_{j+1}(X)$ were words in *different* blocks.

In the stage 2 system the words $C_1, \ldots C_v$ are replaced by a two dimensional array

$$\begin{matrix} A_{11}, \ldots A_{1j}, \ldots A_{1M} \\ \vdots \\ A_{r1}, \ldots A_{rj}, \ldots A_{rM} \\ \vdots \\ A_{v1}, \ldots A_{vj}, \ldots A_{vM} \end{matrix}$$

of w bit words. This array includes one word per n-tuple per class. No two of these words are the same, and each contains $w-z$ 0's and z 1's in randomly chosen bit positions.

At the start of training the stage 2 zatocoding system, the D block is set to contain only zeros. As before, training consists of working once through all training set patterns for all classes, carrying out an operation typified here in terms of the ith training set pattern in the rth recognition class. The operation is: for all n-tuples, update $D_j(X_{ri})$ according to

$$D_j(X_{ri}) = D_j(X_{ri}) \vee A_{rj}.$$

To recognize an unknown pattern X, for each recognition class in turn the system counts the number of n-tuples for which

$$D_j(X) \,\&\, A_{rj} = A_{rj}$$

and X is assigned to the recognition class for which this count is largest, or rejected if there is a tie.

All our experiments were carried out on a set of unconstrained unpreprocessed hand-printed numerals which had been used in previous work[10] on the n-tuple method. The characters used for testing were always different to those used for training. With $n = 12$, $w = 48$, $v = 10$, training and testing with 630 and 20 patterns per class respectively, the stage 2 system gave 49 per cent recognition success, using 40 randomly chosen n-tuples. This was improved to 68.5 per cent as follows.

In most of the characters *very roughly* 25 per cent of the picture elements were black. So on randomly chosen n-tuples the average percentage of 1's in observed n-tuple states was also very roughly 25 per cent. Conversely, states with many more or many fewer than 25 per cent 1's tended to occur on fewer n-tuples and in fewer classes than states with about 25 per cent 1's. This meant that after training the stage 2 zatocoding system, some of the words of the D block contained hardly any 1's, and other words contained too many 1's.

To even this up, we permanently assigned a different random n-bit word to each n-tuple. For any given n-tuple, all observed states, both in training and recognition, were $\not\equiv$'d with the n bit word assigned to that n-tuple, and the result was used for addressing the D block. The meaning of "$\not\equiv$" was explained above. The effect of this operation is that some of the bits in observed n-tuple states are inverted, thus diversifying the n-bit words used as store addresses, and giving the important improvement in performance reported above. This modification was used in all subsequent work.

Experimental results given below show that, as one might expect, with fixed n, w, z, and training set size, performance peaks as the number of n-tuples is increased. In other words, there is a critical number of n-tuples, such that with more or fewer than this number of n-tuples, recognition performance is not so good. For $n = 14$, $w = 48$, $z = 4$, training

and testing with 600 and 50 per class respectively, we found that on our data the critical number of *n*-tuples was about 50, which gave about 85 per cent successful recognition. To improve this to over 90 per cent we adopted the obvious expedient of combining a number of different *D* blocks. This is the basis of the stage 3 system.

Stage 3: *Optimally packed blocks*

The stage 3 system is equipped with a number of *D* blocks, and with a set of *k* *n*-tuples corresponding to each block. The sets of *k* *n*-tuples corresponding to any two blocks are different. Training the system consists of "training" each block exactly as in the stage 2 system.

In recognizing an unknown pattern, each block produces a count for each class as in the stage 2 system. For each class in turn the stage 3 system adds the counts from all the blocks. The unknown pattern is assigned to the class with the highest count. If there is a tie the unknown pattern is rejected.

3. EXPERIMENTS WITH ZATOCODING

(i) *Variation of* k, z *and number of blocks*

Figure 1 shows a plot of performance against z, for $k = 52$, with four blocks. Figure 2 shows a plot of performance against k with $z = 4$ and four blocks; and Fig. 3 shows a plot of performance vs. number of blocks with $z = 4$ and $k = 50$. Figures 1–3 were obtained by training with 600 and testing with 50 per class, and using $w = 48$ and $n = 14$. $w = 48$ was a convenient choice because it is the word width of the KDF9 computer. $n = 14$ was dictated partly by the NPL KDF9 core store size (32K) and partly by previous experimentation[10] which showed only a slow improvement in performance as n was increased above 14.

From Fig. 3 it can be seen that with $n = 14$, $w = 48$, $k = 52$, and using four blocks, the stage 3 system gave 92.0 per cent recognition success. This required

$$\text{Number of blocks} \times w \times 2^n = 4 \times 48 \times 2^{14} = 3{,}145{,}728$$

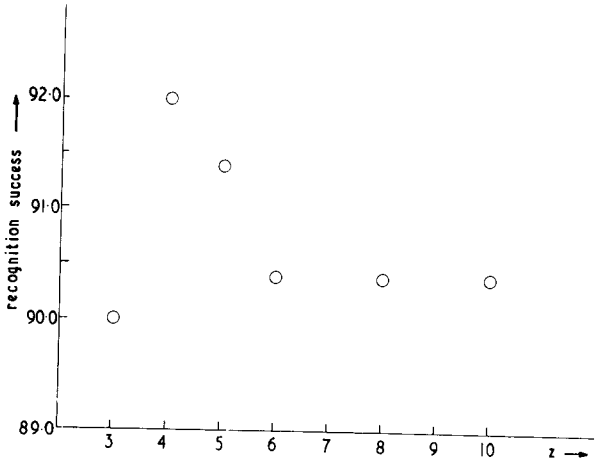

FIG. 1.

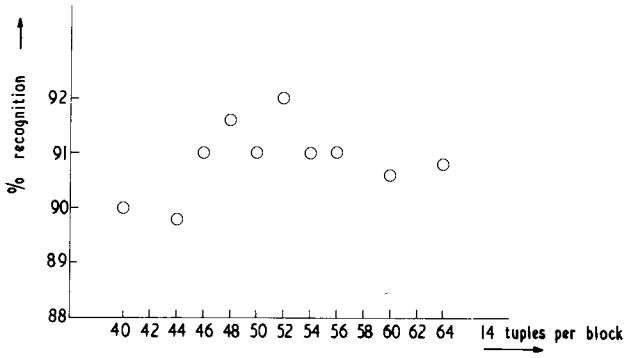

FIG. 2.

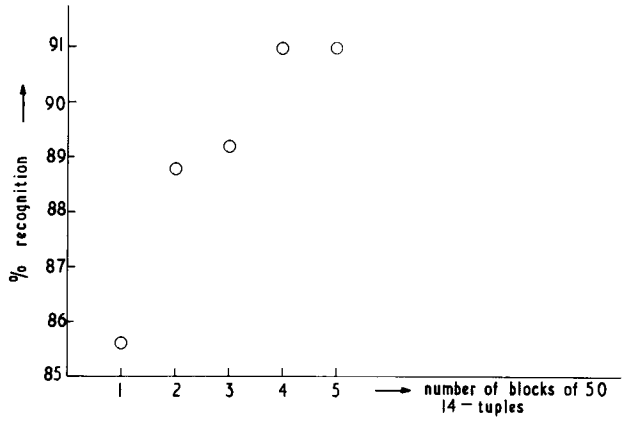

FIG. 3.

storage bits, for the D blocks. Storage of the A arrays corresponding to the D blocks was also necessary, and this required number of blocks $\times w \times k \times v = 4 \times 48 \times 52 \times 10 = 99{,}840$ bits which is fairly small in comparison with the size of the D blocks.

From Fig. 3 in reference (10) it can be seen that on this data the original Bledsoe and Browning system reached 92.0 per cent recognition success with $n = 14$ at 84 n-tuples, requiring

$$84 \times 10 \times 2^{14} = 13{,}762{,}560 \text{ bits.}$$

Thus it can be seen that the zatocoding system achieved a reduction in storage requirements by a factor of roughly four.

(ii) *Using the same zatocodes for all blocks*

The above results were obtained by using different A arrays for different D blocks. This meant that, for example, the word A_{rj} in the A array corresponding to the first D block was different to the word A_{rj} in the A array corresponding to the different D block. This called for 520 48-bit words per D block, when the A words were implemented by computer words. In a special purpose hardware implementation, each A word could be implemented by a z input "and" gate. If the A arrays for the different D blocks were the

same, the same 520 "and" units could serve all four blocks, which would be a worthwhile economy.

Using the same data and n-tuples as were used above to achieve 92.0 per cent recognition (actually 0.6 per cent reject, 7.4 per cent substitution), a further experiment was performed, differing from the previous experiment only in that the same A array was used for all four D blocks. Recognition success fell to 90.8 per cent (0.4 per cent reject, 8.8 per cent substitution). Presumably this signifies that different A arrays give rise to different errors which to some extent cancel out when the recognition counts from the different blocks are pooled.

4. ASSOCIATIVE MEMORY IN THE BLEDSOE AND BROWNING SYSTEM

An entirely different approach to the reduction of the storage requirements of the Bledsoe and Browning system stems from the use of associative memory. To introduce this we return briefly to the original form of the Bledsoe and Browning system outlined above.

Let $S_j(X)$ be the state of the jth n-tuple in pattern X, let L_{rj} be the set of all states of the jth n-tuple which occur in at least one training set pattern belonging to the rth class, and let t be the number of training set patterns in each class. At the start of training in the Bledsoe and Browning system, $B_j(X) \,\&\, C_r \neq C_r$ for all j, r and X, since the store is cleared. If and only if

$$\left.\begin{array}{ll} & S_j(X) = S_j(X_{r1}) \\ \text{or} & S_j(X) = S_j(X_{r2}) \\ \text{or} & S_j(X) = S_j(X_{r3}) \\ & \vdots \qquad\quad \vdots \\ \text{or} & S_j(X) = S_j(X_{rt}) \end{array}\right\} \tag{3}$$

the store will have been updated in accordance with (1) so that for $S_j(X)$ condition (2) will be satisfied since

$$(B_j(X) \vee C_r) \,\&\, C_r = C_r.$$

If $S_j(X)$ satisfies condition (3) it belongs to L_{rj}, by definition of L_{rj}. Conversely if $S_j(X)$ belongs to L_{rj} it satisfies condition (3). Therefore, $S_j(X) \in L_{rj}$ is equivalent to (2).

This means that the following simple reorganization does not affect the recognition performance of the Bledsoe and Browning system. The reorganized system does not store 2^n v bit words per n-tuple. Instead it stores v lists of n-bit words, the number of entries per list being variable. Training now consists, for all r, j, of entering in the list corresponding to the rth recognition class and jth n-tuple the states of the jth n-tuple which occur in at least one member of the rth class training set. After training, this list is a list of the members of L_{rj}. To recognize an unknown pattern X, for each recognition class in turn the system[7] counts the n-tuples for which $S_j(X) \in L_{rj}$. X is assigned to the class for which this count is largest, or rejected if there is a tie.

Storage of the lists calls for $v \times M \times e \times n$ storage bits, where e is the average number of entries per list. Comparing this with the requirement of the original Bledsoe and Browning

system for $v \times M \times 2^n$ bits, we see that the reorganized system is more economical than the original if $e < 2^n/n$. Figure 6 in reference (10) is a plot of e vs. n for our data, with 630 patterns per class in the training set. The slope decreases when n increases above roughly 12, whereas $2^n/n$ always increases when n increases, and for $n = 16$ the reorganized system becomes very much more economical than the original. The reorganized system was used for obtaining the $n > 14$ results reported in references (9) and (10). For $n = 11, 2^n/n = 186.2$, and $e = 183.0$, so for $n = 11$ both systems require roughly the same number of storage bits. Below $n = 11$, the original system is the more economical. For instance for $n = 8, 2^n/n = 32$ and $e = 84.1$.

For $n = 14$, $e = 294.7$ and $2^n/n = 1170.3$, so that the reorganized system requires only about one quarter of the storage bits required by the original system when $n = 14$. In comparing this with the economy achieved by zatocoding, costs of implementation should be taken into account. A computer simulation of the reorganized system is very much slower than simulation of the original or zatocoding systems, since recognition involves serial searching through lists. Serial searching can be reduced by hash coding, or can be avoided by storing the lists in associative memories which can be addressed in parallel.

For this purpose all that is required is a memory which gives a yes/no answer to the question "is the word which is used as the address identical to any of the words which are addressed?". SLADE and SMALLMAN[8] have called this rudimentary type of associative memory a "catalogue memory", because it tells us whether or not a given word is included in a stored catalogue of words. The present application would require $v \times M$ catalogues, each consisting of one of the lists L_{rj}. The cost per bit would be greater than in a conventional high speed memory.

5. RECOGNITION OF N ELEMENT PATTERNS

We now indicate briefly a further possible approach to economy. In recognizing an unknown pattern X, the reorganized system determines whether $S_j(X)$ belongs to L_{rj} for each r, j. This determination can itself be regarded as a pattern recognition problem, in which $S_j(X)$ is an unknown n bit pattern and the L_{rj}'s are the different recognition classes of n bit patterns. This differs from the usual recognition problem in that

(i) $S_j(X)$ can validly belong to a plurality of classes, that is, L_{rj}'s.
(ii) At the end of training the L_{rj}'s are known and available explicitly.
(iii) If the system is to be identical in performance to the original Bledsoe and Browning system, generalization from the training set is *not* required. That is, n-tuple states must only be classified as belonging to L_{rj} if they have been found to belong to L_{rj} during training.

This n-bit pattern recognition problem could be tackled by many well known techniques. For instance for each L_{rj} it would be possible to synthesize a network of linear threshold elements[2] to dichotomize L_{rj}/not L_{rj}. Piecewise linear separation of L_{rj}/not L_{rj} would be a special and perhaps clumsy case of this. Since the members of any L_{rj} could be expected not to be purely random, and could be expected instead to cluster at least to some extent, it is possible that the number of threshold logic units per L_{rj} would turn out to be small enough to make this approach worthwhile from the point of view of storage economy. In recognizing an unknown pattern X, the hyperplanes associated with L_{rj} would be used

for determining whether or not $S_j(X) \in L_{rj}$, and X would be recognized according to the rule used in the reorganized Bledsoe and Browning system.

Note that neither the zatocoding nor the associative memory approach attempts to exploit any non-randomness which may exist in the membership of any L_{rj}.

6. CONCLUSION

It is proper to conclude this paper with two cautionary remarks. The first is that our actual recognition performance figures do not mean very much because of the small amount of data used, because the scanner was hand operated and rather primitive, and because although the numerals were handprinted by 650 different people, this was done in the laboratory and not in the field.

To reduce the error rate to a practical level it would be necessary to make developments such as (a), (b) and (c) mentioned above in the introduction. A well designed preprocessor (e.g. edge-detector) would have a de-noising effect which would reduce the value of e, the average number of stored states per n-tuple per class. Because of this, it is to be expected that if two experiments were performed using stage 3 zatocoding, one using preprocessed data, the other using the same data not preprocessed, and both using the same training set size and value of n, the extent to which zatocoding could reduce the storage requirements while retaining a given level of recognition performance would differ in the two experiments. From CALVIN MOOERS' theory[5] we would expect a reduction of e to favour zatocoding. A reduction of e would necessarily be an advantage from the point of view of associative memory. In short, our second cautionary remark is that the reduction in storage requirements by zatocoding or associative memory would be *altered* by effective preprocessing. It would also obviously be altered by changes in n or training set size.

Previous results[10] indicate that as the training set and optimal n increase, the error rate of the Bledsoe and Browning system at optimal n tends to an asymptotic level above zero. We interpret this intuitively as meaning that the system does not assume enough *a priori* concerning recognition classes, or, in other words, that the logical structure prior to training is too simple. This raises basic questions which have not been tackled in the present paper which has been concerned instead, as an interim measure, with a purely symptomatic treatment of the storage gluttony of the Bledsoe and Browning system.

SUMMARY

The basic and well known pattern recognition method of Bledsoe and Browning has the advantages of simplicity and relatively high speed, but the disadvantage of massive storage requirements. In the application of this method to unconstrained hand printed numerals, using 14-tuples, a fourfold reduction in storage area has been achieved by the use of random superimposed coding. A similar reduction can be achieved by the use of associative memory, but at greater cost per bit. These reductions are achieved without loss of recognition performance. A third approach to economy is outlined which aims at exploiting any non-randomness which may exist in the collection of stored states of a given n-tuple for a given recognition class.

Acknowledgement—The work described above was carried out at the National Physical Laboratory.

REFERENCES

1. W. W. BLEDSOE and I. BROWNING, Pattern recognition and reading by machine, Proc. Eastern Joint Computer Conference, pp. 225–232 (1959).
2. P. M. LEWIS, II and C. L. COATES, *Threshold Logic*. Wiley, New York (1967).
3. C. N. MOOERS, Zatocoding applied to mechanical organisation of knowledge, *Am. Docum.* **2**, 20–32 (1951).
4. C. N. MOOERS, Choice and coding in information retrieval systems. *Trans. IRE* **PGIT-4**, 112–116 (1954).
5. C. N. MOOERS, The application of simple pattern inclusion selection to large-scale information retrieval systems, ASTIA document No. AD-215 434 (1959).
6. J. H. MUNSON, R. O. DUDA and P. E. HART, Experiments with Highleyman's data, *Trans. IEEE* **C-17** (4), 399 (1968).
7. D. RUTOVITZ, Pattern recognition, *J. R. Stat. Soc.* **129A**, part 4, 504–530 (1966).
8. A. E. SLADE and C. R. SMALLMAN, Thin-film cryotron catalog memory, *Solid-St. Electron.* **1**, 357–362 (1960).
9. J. R. ULLMANN and P. A. KIDD, Recognition experiments with typed numerals from envelopes in the mail, *Pattern Recognition* **1**, 273–289 (1969).
10. J. R. ULLMANN, Experiments with the *n*-tuple method of pattern recognition, *Trans IEEE* **C-18** (12), 1135–1137 (1969).

Applic

(*Re*

Abstract—Color is on
color and demonstrate
single scene through re
and regions which hav
color, shape, size and

SINCE the possibil
suggested by Sha
been studied.

ERNST[1] was th
with tactile and po
made an excellent
began research on
MINSKY and others
it to a robot syster
et al.[4] at Stanford
ears for testing ne
at Stanford Resear
move around in si
oratory are making

Motivation for
Robots find applic
application is to ex
provided. The rob
pattern recognition

* Present address: Osak