Associative Memory for Geon-Based Object Identification

Simon Léonard[1], Richard Lepage[1] and Tanneguy Redarce[2]

[1]Laboratoire d'imagerie, de vision et d'intelligence artificielle
École de technologie supérieure, Montréal (Québec), Canada
email: sleonard@livia.etsmtl.ca, lepage@livia.etsmtl.ca

[2]Laboratoire d'Automatique Industrielle
Institut Nationale des Sciences Appliquées, Villeurbanne, France
redarce@lai.insa-lyon.fr

Abstract

On-line inspection for defects in manufactured parts is a challenging problem and a crucial issue for any production process. Precise and extended CAD data describe the manufactured part under inspection and can be stored in a database in order to perform comparisons between the measured observed part (with a 3D laser camera, for instance) and its CAD description. The problem is very slow access time to find the right CAD model, because of the huge amount of data necessary to represent even a very simple manufactured piece. We propose a fast access to the database based on a vision-derived representation of the part, much less precise than the CAD representation but which offers a very low storage requirement. CAD representations of the models are converted off-line into a vision-based representation used to train a neural network. Volumetric projected components (geons) and their inter-relationship are extracted from the digitized image of the piece under inspection and input to a neural network whose outputs point to the most probable models in the large database. Corresponding CAD data is then available for the inspection module.

1 Introduction

Many applications require real-time analysis in order to identify a particular feature. This requirement becomes particularly critical when the information needed for such an analysis is complex and requires a considerable amount of processing. A good approach to solve such a problem is to perform the analysis as soon as a new information becomes available. This paper describe the implementation of a neural network shift-reduced parser based on SARDSRN [1] for analyzing sequences of volumetric projected components (geons) in order to perform object identification. The SARDSRN is mainly used to parse sentences, thus in a noise free environment, where the sequence order is relevant and where the input comes from only one channel. This paper describes how the SARDSRN can be modified for an environment where the available information is corrupted and comes without any order. We also extend the network to use a second channel of information in order to increase the performance of the object identification process.

2 Automated Inspection System

Automated inspection of manufactured parts is a difficult problem because of the huge amount of information to be processed in real-time. CAD data, necessary for tolerance verification, is a complete and very precise description of the part under inspection. Problem arises when the object under inspection is not known *a priori*. It is near to impossible, or at least too long, to query each of the CAD models stored in a database in order to just determine which object is under inspection. We add a CCD camera to the inspection setup and use a perceptual visual representation, that is much more dense but less precise, in order to provide quickly an index for selecting the right CAD models.

The overall automated inspection system is shown in Fig 1. Manufactured objects have to be inspected according to their CAD representation stored in a database. The inspection module must use the CAD representation and a 3D image of the object under inspection obtained by a laser camera to perform the comparison. Laser image and CAD representation involve too much data for indexing the database efficiently. Therefore, a higher level representation that can abstract the details is required for fast indexing. The proposed solution is to represent the objects by their volumetric projected components or geons [2]. The

indexing is performed by a neural network trained with the geon representation extracted from all the CAD descriptions in the database. When an object is under inspection, the geon representation extracted from the CCD image is input to the neural network which then identifies the object and indexes the database.

3 Geons Representation

The purpose of decomposing an object into geons is to grossly describe the object by its main elementary components. Geons include volumetric components such as: cylinders, parallelepipeds, cones, prisms, etc. Basically, geons are defined according to four attributes: axis type, edge type, symmetry and sweep dimension. This set has been extended to include negative components (holes) by adding a fifth attribute: material type [3], for a total of 72 geons, half of them being negative. Fig 2 shows the visual effect of the four basic attributes.

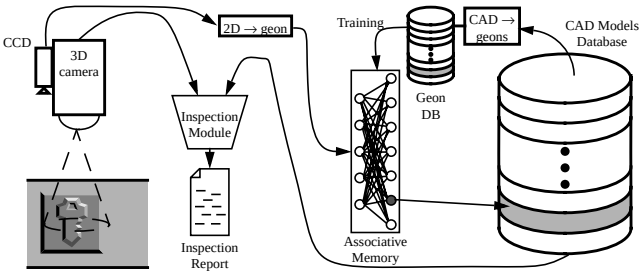

Figure 1. Automated Inspection System.

According to their axis, the way two geons are connected can be described in five different ways: colinear, angular, parallel, crossed, included.

Geons are qualitative entities, mostly aimed to describe families of objects, rather than a specific one. For instance, any flashlight could be *geonized* as "a combination of two straight cylinders joined at their ends, with a parallelepiped lying along one of the cylinders". If a generic description is obviously too vague for any task that requires some accuracy (metrology, quality control, etc.), it could be well suited on the other hand as an indexing key in a large database. As a matter of fact, Biederman calculates than more than 154 million qualitative different objects can be derived from all possible combinations of three geons.

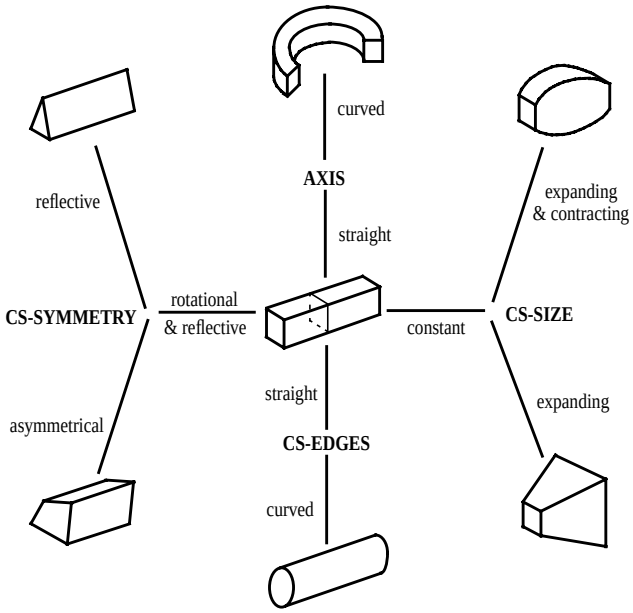

Figure 2. Geon Attributes.

4 Constraints for Accurate Real-Time Identification

The first constraint for object identification comes from the extraction process. The real-time constraint requires that the identification be achieved within a minimum of time interval, thus with minimal information. This constraint requires that the extraction module sends information as soon as it is available, making the channel between the extraction and the identification modules serial. Identification should be achieved as soon as there is enough components to identify an object amongst others.

Geons offer a relatively good constraint for identification. However, it is likely that some objects will be composed, in whole or in part, of the same geons. Connections between geons offers a better identification clue than geons solely, mainly because a connection involves two geons connected together with a connectivity attribute. This also involves an extra processing time cost. However, it is still possible to have two objects having the same list of connections and look totally different. The last information we can add for the decomposition is the number of connections in which each geon is involved.

The last constraint rises from the unordered sequence in which the geons are transmitted. There is no *a priori* knowledge of the order of decomposition of the objects, so that two equivalent sets of geons should produce the same output even if their order is different.

4.1 Identifying a Sequence of Geons

Our algorithm used to identify a sequence of geons is based on the shift-reduce algorithm used for parsing sentences by reading one word at a time [1]. First, the shift-reduce algorithm tries to reduce the representation of the input words by applying grammar rules. If no grammar rule applies, then the next word will be read. The algorithm iterates these steps until a final representation is reached. The shift-reduce algorithm can be easily adapted to solve our problem by designing a set of «construction rules» instead of grammar rules. This is achieved by creating a set of rules that determine if it is permitted to build an object, in whole or in part by using an unordered sets of geons. Instead of parsing a sentence word by word, an object is build geon by geon, until the final representation of the object is given. For example, the set of rules for the object (object #2 in our database) illustrated in Fig 4 is shown below.

<Final State 2> → <Intermediate State><#5>
<Intermediate State> → <#15> | <#47> | <#15,#15> |
<#15,#47> | <#47,#47> |
<#15,#15,#47> | <#15,#47,#47> |
<#15,#15,#47,#47>

Figure 3. Grammar rules producing the final state of the object shown in Fig 4.

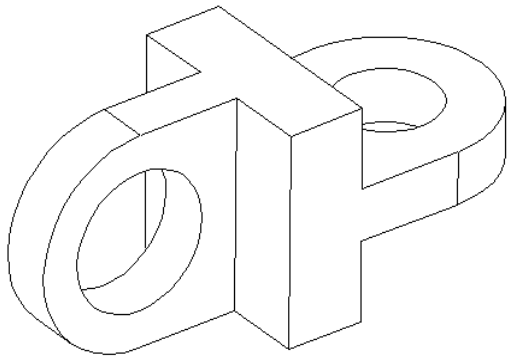

Figure 4. Example of an object used in the database.

Each tag corresponds to one of the 72 geons. For instance, the geon #47 is the circular hole shaped like a cylinder, the geon #5 is the box in the middle and the geon #15 corresponds to an hybrid (straight lines and curves) arm. When parsing sentences, final states are produced when the words of a sentence are seen as grammatically complete and coherent. In our case, final states are produced when there is enough evidences in the sequence to identify an object. Since there are many objects, a different final state is associated to each object. Also, since there are several ways to identify the same object, there are several ways to produce each final state. For example, if we look back at the grammar rules shown in Fig 3, we see that the object can be identified by one of the following unordered sets: {#5, #15}, {#5, #47}, {#5, #15, #15} {#5, #47, #47} {#5, #15, #47} {#5, #15, #15, #47}, {#5, #15, #47, #47}, {#5, #15, #15, #47, #47}. Thus, these sets are the different ways to identify this particular object. So the problem can be reduced to finding all the unordered sets of geons that uniquely identify each object. Being concerned with real-time constraint, there is no need to deal with unnecessary information but to look only for sets who can identify an object with a minimum of information. For example, if an object can be identified by the set {#5, #5}, there is no need to use the fact that the same object can also be identified by the set {#5, #5, #5}, since it can already be identified after the second geon. This important feature has the following consequences. First, it identifies objects with less information, thus in less time. Second, it greatly reduces the number of combinations necessary to identify each object, which greatly reduces the number of training patterns and consequently the errors in the network. Third, it simplifies the writing of grammar rules and training examples. Finally, there are states that are common to many objects. In the example shown in Fig 3, we see that the concatenation of the state <Intermediate State> and the geon #5 clearly identify the object #2 by generating the final state <Final State 2>. This indicates that the state <Intermediate State> is common to at least two objects, otherwise it would represent an object and make the concatenation with the geon #5 unnecessary. These intermediate steps don't give any information on the identity of the object under inspection. They are just an iteration step in the identification process.

One useful characteristic for using the shift-reduce parsing for geon-based identification is the simplicity of the parsing. For example, with the rules shown in Fig 3, the ordered sequence of geons #15, #47, #5 respectively produce the outputs: <#15>, <#15,#47> and <Final State 2>. The ordered sequence #47, #15, #5 produces a similar output: <#47>, <#15,#47>, <Final State 2>. Notice that the order of the geons #47 and #15 in the second sequence is different from the first one, but produces the same output <#15,#47>. This has no effect on the performance, but greatly reduces the size of the lexicon, eliminating all the possible permutations (<#15, #47> and <#47, #15> in this example). This shows that since the order of the input sequence does not affect the output, and because the structure is much simpler than grammatical ones, a reduction occurs at every step.

5 SARDSRN

The design of the associative memory is based on the SARDSRN parser which is a simple recurrent network (SRN) [4]. This network is usually used to parse sentences by using the shift-reduce algorithm.

The first step is to identify the geons according to their attributes. A self-organizing map called SARDNET [5] (Sequential Activation Retention and Decay NETwork) is used in the parser for sequence classification. SARDNET has activation retention and decay that allows to represent ordered sequences. The activation retention allows a unit to remain active throughout a complete sequence. Every time there is a new input, a new unit is activated and can't be reused for further mapping in the sequence. This means that any input will be mapped on the closest available unit. SARDNET is very effective to represent similar input in clusters by mapping similar (or identical) inputs in a near neighborhood. As we will see later, this feature is used to its full extent for handling noise. The decay is a feature that allows to represent chronology amongst the activated units. Every time a new unit is activated, all the other activated units are decreased by a constant factor. However, in this application, the order in which the geons are received is not relevant. Therefore, the decay feature is not used in the current implementation. This allows a sequence of *n* geons, whatever the order, to always be mapped the same way instead of *n!* when the decay is used and greatly improves the network performance, i.e. all activated units are set to «1» instead of *n* different activations which implies an *n-permutation*.

The SARDSRN uses one hidden layer with three inputs as shown in Fig 5: the geon input (which is the same as for the SARDNET), the SARDNET representation, and the hidden layer representation of the previous step. At each step in the sequence, the hidden layer is copied by propagating itself to a separate layer. This copy will then be used as an input for the hidden layer at the next step. All the state representations are distributed over all the output units, i.e. there is no dedicated output units and all the input and output representations are contained in a lexicon.

5.1 Input Noise

There are two error sources caused by the decomposition process. The first is decomposition errors, and can be handled up to a certain limit by the network. The second is related to the complex problem of decomposition multiplicity and has not yet been fully solved.

Decomposing an object into geons is not a simple task, and not surprisingly mistakes can be made in this process. For example, a cone could be interpreted as a cylinder. One way

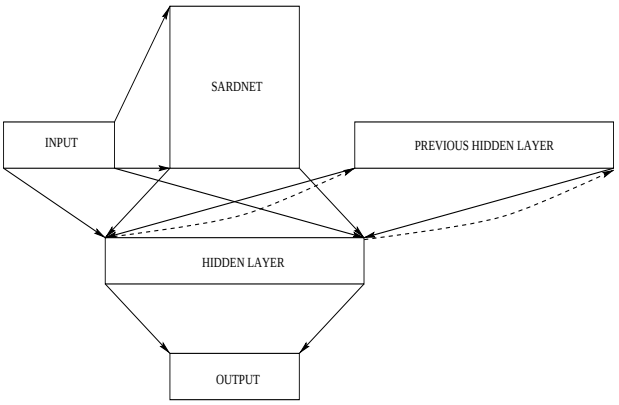

Figure 5. Basic configuration of SARDSRN.

to help the network to deal with such misinterpretation is to use a map that projects similar geons into a close neighborhood. Since SARDNET is a self-organizing map, one can control the mapping by pondering the number of input units for each attribute according to their confidence level. For example, there should be a big difference in the mapping of positive and negative geons (solids or holes). The importance of each attribute according to their confidence level has been established as: 1- type, 2- edge type, 3- axis type, 4- symmetry, 5- sweep dimension. Therefore, by using a strong ponderation (more input units) for the geon type, we can split the map into two parts, one for the positive geons and one for the negative geons. This can also be done for the other attributes as well. Also, by using a relatively tight map, we can closely compress similar geons. The result is that the network will output the closest known representation even when an extraction error is made, or it will output the right representation if no error was made. So, the SARDNET has to be trained in such a way that similar geons, from a visual point of view, form tight neighborhoods.

5.2 Extending the Network for Handling Connections

Modeling connections for sequential processing poses few problems. To keep a good real-time behavior requires working with sequential information. This means that the information on connections will also have to be processed serially. The information on a connection involves the identification of two implied geons and a connection attribute. This brings the problem on how to represent the information of the connections. At first, it would seem logical to add the connection information as a sixth attribute at each geon, but this method has two major drawbacks. First, how can we find information about how and with which geon to connect when its neighbors might not have been detected yet? Second, how can we efficiently represent

with the same dimensionality (with a fixed size attribute) a geon connected to only another one and a geon connected to many different ones? In other words, how can we represent with the same attribute various numbers of connections knowing that for each connection there are 72 different kinds of neighbors that can be connected in five different ways? These questions clearly indicate that the connection information can't be concatenated to the geons five attributes, but still have to be sent serially.

The best way to handle this is to split the information into two different data types, each one using a dedicated channel. The first one representing individual geons with five attributes, and the second one representing individual connections with eleven attributes (ten for the two geons and one for the connectivity). This configuration allows to process geons and connections simultaneously and independently and it effectively uses the information available. Since connections imply more information (two geons and one connectivity attribute) they offer better constraints for identification, i.e. a geon can belong to many objects but it is less likely to be connected the same way with the same neighbor in these objects. These data structures require an additional input in the network that can also process sequential information. The SARDSRN can easily be extended to support this new input by simply adding an additional SARDNET that would process the connection sequences as shown in Fig 6. Therefore, the SARDSRN can combine both SARDNET representation for even faster and efficient identification. With this configuration, the geon extraction module has the flexibility required and can send without any order the information on individual geons or connections as soon as information is found.

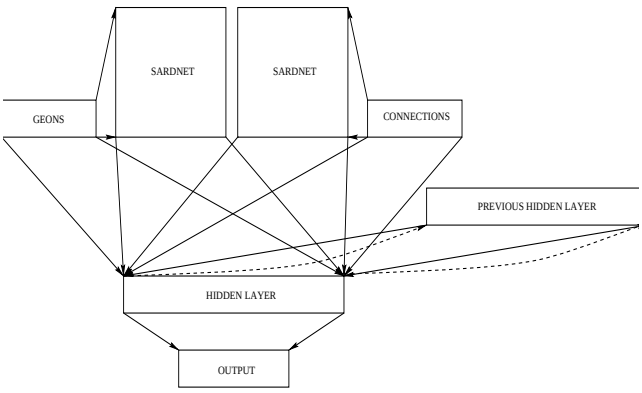

Figure 6. Extending the SARDSRN to handle connections by adding a second input.

6 Results

Experiences have been done on a database containing information about ten manufactured objects. The number of geons varies from two to ten for each object, and the total number of different geons used is eleven. The network was trained by using sequence examples so the network had to extrapolate the grammar rules. The total number of sequences necessary for training with these objects was 331, including all the relevant permutations that leads to a final state, i.e. the sequences: #15, #47, #5 and #47, #15, #5. The last geon in a sequence can never be permuted with the others since it is the one that is making the difference between an intermediate state and a final state. The shortest sequence was composed of only one geon (objects that are composed of one geon that no other object has) and the longest was composed of six geons, which means that no object has more than five geons in common with another one.

To accelerate the training, the SARDNET was trained separately from the rest of the network and once it is correctly trained, it can be reused for any other database as well. All of the geons, even those who are never used in the parsing, are part of the examples to train the SARDNET so it can accurately map geons that could be encountered in noisy inputs. As for the parser, since some objects are more complex than others, they have more possibilities to reach their final state, thus they necessitate more training sequences than others. In order to balance the training, this requires equilibrating the number of examples by duplicating those of objects who are penalized by showing only a few sequences. After the equilibration, the total number of training examples for the parser is 1030. For this database, 200 epochs were sufficient to train the network.

When testing the network with noise free inputs, the network always showed the correct output. On the other hand, with noisy sequences, its behavior varied depending on the noise severity. When an error is made in the geon attributes, the network produces the closest known representation of the sequence so far. When the mistakes are in the less important attributes, such that the geon is mapped relatively near the right one, the network will output the right representation. An interesting case is when the network is in an intermediate state and the noisy representation of the next geon belongs to another object who also share the previous state. Then, the network produces the learned output of the other object and even identify an object if it can, but it falls back in the right track when the subsequent geons are input (as long as they are not too noisy). For example, let O1 and O2 be two objects defined respectively by the following sets of geons {#5,#12,#15} and {#5,#11,#47}. Now, suppose that the object O1 is under inspection and the geon sequence

is in the following order: #5, #11 and #15 with a mistake in the geon #11 instead of #12. First, the network will output the intermediate representation for the geon #5. Then, since the second geon is #11 and that the network had learned the representation <#5,#11> from O2, it will output this representation. Finally, since there are enough differences between #15 and #47, the network will fall back in the right track and output the representation of O1 when the geon #15 is input. This even works when the network wrongly identifies an object. For example, if <#5,#11> occurred to be final state for O2, the network would immediately have identified O2 after the geon #11, but it would have «change its mind» for O1 after receiving the geon #15. When severe noise is added like replacing a solid geon by a hole, the network won't consider the last geon and remain at the same state. In parsing sentences, this can be compared to when a word is shifted but no grammar rules can be applied so another input is shifted again.

7 Conclusion

The shift-reduce is a very well suited algorithm for processing sequential information. By extending the number of final states, it can be efficiently used to parse sequences that lead to different conclusions. Also, this algorithm has the advantage that it is completely independent of the sequence length. This paper indicate how to use the SARDSRN in a parsing applications where input can be noisy and the sequence order is not relevant. The SARDSRN had already shown its accuracy and efficiency in using the shift-reduce algorithm to parse relatively simple sentences in the english language. Using the SARDSRN with geon sequences, a much simpler structure than natural language, yielded excellent results. The network showed great capacities in handling multiple final states and noise by rejecting too noisy inputs and changing its mind when there is evidences it was misleading. The network is totally independent of the extraction module and can also be easily extended to handle connections letting all the flexibility to the extraction module on the order of extraction.

8. References

[1] Mayberry, M.R., Miikkulainen, R. (1999). SARDSRN: A Neural Network Shift-Reduce Parser. To be published in the *Proceedings of the 16th Annual Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden.

[2] Biederman, I. (1987). Recognition by components: ATheory of Human Image Understanding. Psychological Review, 94, 115-147.

[3] Grégoire, N., Lepage, R., and Redarce, T. (1999). From CAD to geons: a simplified data representation for rapid indexing of large CAD models database. *Proceedings of the IS&T/SPIE Conference on Machine Vision Applications in Industrial Inspection VII*, held at San Jose, California, January, 3652, 224-231.

[4] Elman, J.L. (1990). Finding Structure in Time. Cognitive *Science*, 14, 179-221.

[5] James, D.L., Miikkulainene, R. (1995). SARDNET: A Self-Organizing Feature Map for Sequences. In Tesauro, G., Touretzky, D.S., Leen, T.K. (editors), *Advances in Neural Information Processing Systems*, 7, 577-584. Cambridge, MA: MIT Press.