Using Ontologies for Recognition: An Example

Mieczyslaw M. Kokar
Department of Electrical
and Computer Engineering
Northeastern University
Boston, MA 02115
kokar@coe.neu.edu

Jiao Wang
Department of Electrical
and Computer Engineering
Northeastern University
Boston, MA 02115
jiaowang@coe.neu.edu

Abstract – *In this paper we investigate a scenario in which the fusion process (the algorithm) could be synthesized at run time. This goal can be achieved in two steps: first synthesize a formal specification of the fusion process and then generate code from the specification. In this paper we show the first of these steps.*

Keywords: Ontologies, Unified Modeling Language, Automatic Target Recognition, symbolic information, dynamic process update.

1 Introduction

An ability to dynamically accommodate symbolic information by an automatic target recognition (ATR) system is a high value feature that allows for the dynamic inclusion of new types of target into the recognition algorithm. The new symbolic information might be obtained through various means, e.g., through an intelligence channel, through reading a description of a new type of target in a library of target specifications, or by receiving a message from a software agent that is part of a Situation Awareness system (SAW). Once such information is received, it has to be incorporated into the target recognition algorithm on the fly, i.e., at run time. After that, the algorithm can be used to process sensory information. To achieve this kind of goal the ATR system needs to "understand" both the symbolic and the sensory information. This capability is achieved through the use of an ontology. Both the fusion agent and the knowledge provider know the ontology, and both the symbolic information and the knowledge of the sensors is "annotated" using that ontology.

In this paper we describe an approach to solving such a problem, provide examples of symbolic target descriptions, describe an artificial ATR scenario, and use the example to show how some of the crucial steps of the whole process could be accomplished. We also summarize the advantages of the proposed approach over that in which the recognition algorithm is fixed. The main building ingredient of our approach is DAML (DARPA Agent Markup Language) in which ontologies are specified and symbolic information is annotated.

In Section 2 we present a simple scenario which we use to explain the idea of ontology-based target recognition. Then in Section 2.5 we describe the details of our system. Finally, in Section 3 we present conclusions.

2 Scenario

The main point that we want to stress in the scenario is that the types of target are not known in advance. In our experiments we simulated a world of simple geometric objects. We simulated two kinds of sensor input, an intensity image and a range image. Examples of images of two types of target (cube and pyramid) are shown in Figures 1 and 2. The images in the top row are from a range sensor and in the bottom from an intensity sensor. It is clear that an ATR system can take advantage of both sensors, since one of the sensors might not give enough confidence of a recognition decision. For instance, in Figure 1, where both sensors are positioned above the target, the intensity images look the same for both targets, while the range images are quite different. On the other hand, in Figure 2 the range sensor is positioned above, but far from, the target and thus it cannot distinguish a cube from a pyramid, while the intensity sensor positioned on one side of the target is able to see a difference in the shapes of the targets.

The scenario for developing a specification of a processing algorithm is as follows:

1. Agent receives an update of the ontology that includes description of target called `pyramid`.

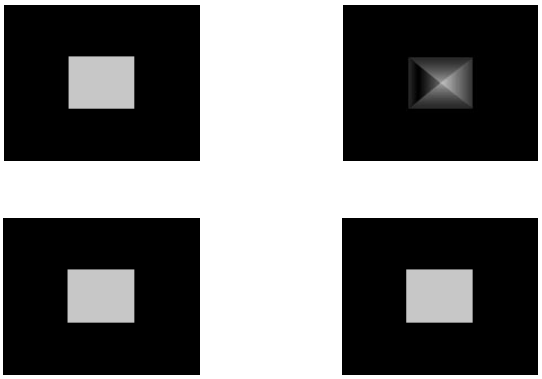

Figure 1: Examples of Images: Top View. Range images (top row), intensity images (bottom row).

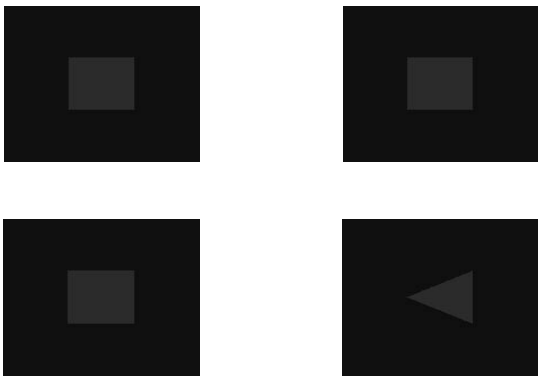

Figure 2: Examples of Images from Far Away. Range images from above (top row). Intensity images (bottom row), views from one side.

2. It incorporates this knowledge into its existing ontology.

3. Agent receives a request for information `TargetType(X, ''Pyramid'')`.

4. Agent posts such a query to its query processing engine.

5. The query processing engine invokes its generic processing algorithms and returns a result (either "yes" or "no").

2.1 Corner Detection Algorithms

In our experiment we assumed that there is a feature extraction preprocessing block. The features are *corners* classified into a number of types. To detect corners we used wavelet transforms. Towards this aim we implemented the corner detection algorithm presented in [5]:

1. For a given image $I(x, y)$, calculate wavelet coefficients $W_x(s, x, y), W_y(s, x, y)$ at a given scale s. For this purpose, the following basis wavelet was used [12]:

$$\varphi_1(x, y) = -x \cdot e^{(\frac{-(x^2+y^2)}{2})} \quad (1)$$

$$\varphi_2(x, y) = -y \cdot e^{(\frac{-(x^2+y^2)}{2})} \quad (2)$$

2. Compute the modulus M and the orientation O according to the following formulas:

$$M(s, x, y) = (|W_x(s, x, y)|^2 + |W_y(s, x, y)|^2)^{\frac{1}{2}} \quad (3)$$

$$O(s, x, y) = tan^{-1}(\frac{W_y(s, x, y)}{W_x(s, x, y)}) \quad (4)$$

3. Use the *scale proportion property*, i.e., $\frac{M(s_1,x,y)}{M(s_2,x,y)} - \frac{s_1}{s_2} = 0$, to find edge points of an image. This property is satisfied by corner points and edge points. We used $s_1 = 2$ and $s_2 = \sqrt{(2)}$, as in [5].

4. Eliminate edge points using the *orientation variance* property. This property captures the fact that the orientation variance near a corner point is much larger than near an isolated edge point. Using this property we can eliminate isolated edge points (might be spurious points, but unfortunately, it also eliminates corner points, like the tip of a pyramid). After this step, only corner points and edge points that are very close to the corner point are left.

5. Locate corners using the property of *scale invariance*. This property is that the orientation $O(x, y)$ at a corner point (and isolated edge points) is independent of scale. Since it depends on the scale at edge points, in this step, edge points are eliminated and only corner points are left. In this step we computed the values of $O(x, y)$ for $s_1 = 2$ and $s_2 = \sqrt{(2)}$, as in [5]. The points for which the values of $O(x, y)$ for the two scales were within the threshold of 0.3 were removed by the algorithm.

2.2 Corner Classification

We defined two kinds of corner for each sensor type. For the range sensor we had *RangeRectCorner* and *RangePyramidCorner*. Similarly, for the intensity sensor we had *IntensityRightCorner* and *IntensityAcuteCorner*. Examples of images of the two range types

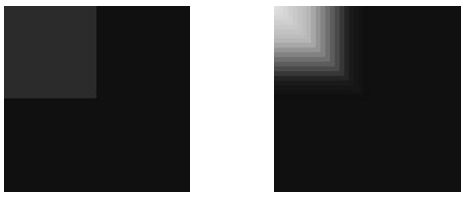

Figure 3: Range Corner Types: *RangeRectCorner* (left) and *RangePyramidCorner* (right)

Figure 4: Intensity Corner Types: *IntensityRightCorner* (left) and *IntensityAcuteCorner* (right)

of corner are shown in Figure 3 and for intensity in Figure 4.

RangeRectCorner is a rectangular corner, with all the high-intensity points within the corner having roughly the same value.

RangePyramidCorner is a rectangular corner in which the edge points and the corner point have the same value, but for other points the value increases toward the center of the object.

IntensityRightCorner is a corner of 90^o or more with all the high-intensity points within the corner having the same value.

IntensityAcuteCorner is an acute corner, with all the high-intensity points within the corner having the same value.

To classify a corner, the algorithm needs to first compute the degree of the corner and then analyze the variability of the values within the corner. To determine the degree, the algorithm positions the corner in a small rectangular region so that the corner is in the center of the region. Then the high and low value points are counted. The ratio of the numbers of high and low values points gives the angle.

2.3 Target Ontology

An ontology captures the basic terminology (concepts) of the domain of interest and the relationships among the concepts [6, 8]. In the following we show a small ontology for target recognition. It is essentially the same ontology as we used in [7]. First, in Section 2.3.1 we present the ontology in the UML language [4]. UML is a graphical language and thus is easier to understand by both the developer of an ontology and by the reader. However, we need a computer processable representation. For this purpose we use DAML. A small sample of the DAML representation of the ontology is given in Section 2.3.2.

2.3.1 UML Representation

A small piece of an ontology (in UML) for the scenario of geometric object recognition is shown in Figure 5. The two main constructs in UML are *Class* and *Association*. Classes are represented as rectangles, while Associations as arrows. Multiplicities of associations are shown as numbers (or symbols) at the association ends. We also show one special kind of relationship between Classes, called *generalization*. Generalization is represented by a hollow arrow. It means that one class is a subclass of another.

As can be seen from the Figure, there are two main kinds of Class, called *Target* and *Corner*. These two classes are further subclassified into *Cube* and *Pyramid*. *Pyramid* is a generalization of two subclasses, *Pyramid_1* and *Pyramid_2*.

There are two kinds of *Corner*, *RangeCorner* and *IntensityCorner*, each of which is further subclassified into two: *RangeRectCorner*, *RangePyramidCorner* and *IntensityRightCorner* *IntensityAcuteCorner*, respectively.

The most important information in the ontology, from the point of view of target recognition, is the information about the relationships between types of target and types of corner. At the top level of the ontology we show the relationship *hasCorner*, which is then specialized to *hasRangeCorner* and *hasIntensityCorner*. Strictly speaking, in UML, the higher level and the lower level relations are different relations. But in DAML the lower level relations can be treated as *subProperties* of the higher level relation. At the bottom level we show the *multiplicities* of each of the relations. For instance, we can see that a target of type *Cube* must have four corners of type *IntensityRightCorner* and four corners of type *RangeRectCorner*. A target of type *Pyramid_2* must have one corner of type *IntensityRightCorner*, two corners of type *IntensityAcuteCorner* and four corners of type *RangePyramidCorner*.

2.3.2 DAML Representation

A fragment of the DAML representation of the ontology shown in Figure 5 is shown below. This fragment shows the description of the corner type *IntensityRightCorner* and of the target type *Cube*.

```
<daml:Class rdf:ID="IntensityRightCorner">
  <rdfs:label>
```

```
    IntensityRightCorner
  </rdfs:label>
  <rdfs:comment>
     Corners in the intensity image are
     either rectangular or obtuse.
  </rdfs:comment>
  <rdfs:subClassOf
        rdf:resource="#IntensityCorner"/>
</daml:Class>

<daml:Class rdf:ID="Cube">
  <rdfs:label>Cube</rdfs:label>
  <rdfs:comment>
    Cube is a cube.
  </rdfs:comment>
  <rdfs:subClassOf rdf:resource="#Target"/>
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="4">
      <daml:onProperty
       rdf:resource="#hasRangeCorner"/>
      <daml:toClass
       rdf:resource="#RangeRectCorner"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="4">
      <daml:onProperty
       rdf:resource="#hasIntensityRightCorner"/>
      <daml:toClass
       rdf:resource="#IntensityRightCorner"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

2.4 Input Information

Information from both the sensors and other (external) sources needs to be annotated using the language defined by the ontology shown in Section 2.3.2. Below we show a sample of such annotation format.

```
<Target rdf:ID="T1">
<rdfs:label>T1</rdfs:label>
<rdfs:comment>T1 is a target.</rdfs:comment>
<hasIntensityRightCorner rdf:resource="#IC56"/>
<hasIntensityRightCorner rdf:resource="#IC57"/>
<hasIntensityRightCorner rdf:resource="#IC58"/>
<hasIntensityRightCorner rdf:resource="#IC58"/>
<hasRangeRectCorner rdf:resource="#RC59"/>
<hasRangeRectCorner rdf:resource="#RC60"/>
<hasRangeRectCorner rdf:resource="#RC61"/>
<hasRangeRectCorner rdf:resource="#RC62"/>
</Target>
```

This information must come from some sources. For instance, the information can be from a feature (corner) extraction and classification algorithm. It is assumed that algorithms for corner detection and classification as described in Sections 2.1 and 2.2 preprocess sensory information. Additionally, the information about corners is encoded in the language defined by the given ontology. Such an encoding is called *annotation*.

2.5 Target Recognition

The main idea behind the ontology-based recognition is that a particular object is classified as a target of a given type if that object satisfies all the constraints that this type of object is shown to satisfy in the ontology. In other words, for an object to be of type *Cube* it must have the corners as specified in the ontology. This means that the ontology should be used by some executable code. There are various approaches to solving this problem:

- Use a generic theorem prover to prove that the facts known about this target satisfy all the constraints that a target of this type must satisfy. This approach can be used off-line, especially for the proof of concept.
- Use an expert system that invokes its inference engine to check all the rules on the data provided by the feature extraction module. The expert system can derive that it is a target of some type (unique conclusion), or that this could be an instance of a number of types, or none.
- Generate code from a formal software specification (cf. [9]).
- Use a generic inference mechanism, like Prolog, to check the satisfiability of constraints of a given target type.

In all the approaches listed above both the target ontology and the annotation must be first translated to the language that a given inference engine can "understand" (process). We experimented with two different approaches: a Prolog implementation and a Slang implementation. The Prolog solution was described in [7]. In this paper we report on the Slang based approach.

In order to be able to invoke a reasoner on the DAML ontology and the DAML annotated facts, the reasoner must be able to understand the DAML language. While such reasoners are being developed by the DAML community, in the absence of such reasoners at this time, we used the Snark theorem prover [3, 11] within the Specware tool [10, 1, 2]. Instead of using a DAML reasoner, we translated the target ontology represented in DAML and

the facts annotated with DAML into the Slang language, the language of Specware. This translation process was carried out manually. An automatic translator from DAML to Slang is under development. We then invoked the Snark theorem prover on a query (conjecture) of interest, for instance:

```
Type(T1, Cube)
```

For Snark to prove or disprove such a conjecture, it has to have a theory of targets. Such a theory was obtained from the translation process, as described above. The theory consists of a number of axioms.

Here are some of the axioms of the theory of targets.

```
op Cube : Element
axiom cube-is-a-target is
SubClass(Cube, Target)
axiom cube-is-restriction is
Type(Cube, Restriction)
axiom onproperty-of-cube is
PropertyValue(OnProperty, Cube, HasRangeCorner)
axiom toclass-of-cube is
PropertyValue(ToClass, Cube, RangeRectCorner)
op Card : Target * HasCorner -> Nat
op Four : Nat
axiom four-cardinality is
Card(t, h) = Four <=>
ex(c1:Corner, c2:Corner, c3:Corner, c4:Corner)
PropertyValue(h,t,c1) & PropertyValue(h,t,c2) &
PropertyValue(h,t,c3) & PropertyValue(h,t,c4) &
fa(cc:Corner) PropertyValue(h,t,cc) =>
cc=c1 or cc=c2 or c =c3 or cc=c4
axiom cube-has-four-rangerectcorner is
fa(t:Cube)(ex(h:HasRangeCorner)(Card(t,h)=Four ))
axiom cube-has-four-intensityrightcorner is
fa(t:Cube)
(ex(h:HasIntensityRightCorner)(Card(t,h)=Four))
```

Some examples of the facts translated to Slang are shown below.

```
Type(T1,Target)
Type(IC56,Corner)
Type(IC57,Corner)
Type(IC58,Corner)
Type(IC59,Corner)
Type(RC60,Corner)
Type(RC61,Corner)
Type(RC62,Corner)
Type(RC63,Corner)
hasCorner(T1,IC56)
...
hasCorner(T1,RC63)
Type(IC56,IntensityRightCorner)
...
Type(RC63,RangeRectCorner)
```

The theorem prover is given the following conjecture:

```
conjecture is-cube is
Type(T1,Cube)
```

In this experiment, we traced a number of proves for this kind of inputs. The goal was to test all of the components involved in this experiment: the ontology, the annotations, the translations. But the ultimate goal was to check the feasibility of this approach to incorporating symbolic knowledge about targets into the processing algorithms of an automatic target recognition system.

3 Conclusions

As we stated in the beginning of this paper, the main goal of this exercise was to investigate the feasibility of an approach to Automatic Target Recognition in which not all of the targets are known at the design time of the ATR system. Instead, we assume that such knowledge can become available during the operation of the system. It can come from various sources - from the operator, from intelligence channels, or simply from a communication link. The question is then what kind of design is needed to accommodate such requirements. In particular, we were interested in the use of the DAML language for this purpose. In other words, we assumed that the symbolic information would be communicated to the system in the DAML language. We then assumed that the sensory information would also be annotated using DAML.

Although we did not implement a fully automatic system, we implemented the whole process using elements that are, at least in principle, automizable. We say "in principle", since our approach involves theorem proving, which may not be practical in real time. There are various ways of approaching this problem. The approach by the designers of the DAML language involves constraining the language to make inferencing tractable. This philosophy comes from the community of Descriptive Logics. While not everybody in the DAML community agrees with this philosophy, it guarantees the tractability of the inferencing. The main critique of this approach is that the annotation process is very tedious. Other approaches are still being considered.

Acknowledgments

This research was partially supported by a grant from the Air Force Office of Scientific Research under contract No: F49620-98-1-0043 and from the Air Force Research Laboratory under contract No: F30602-00-C-0188.

References

[1] Specware: Language manual, version 2.0.3. Technical report, Kestrel Institute, 1998.

[2] Specware: User guide, version 2.0.3. Technical report, Kestrel Institute, 1998.

[3] SNARK: SRI's new automated reasoning kit, 2002. `http://www.ai.sri.com/ stickel/-snark.html`.

[4] G. Booch, I. Jacobsen, and J. Rumbaugh. *OMG Unified Modeling Language Specification*, March 2000. Available at `www.omg.org/technology/documents/formal/-unified_modeling_language.htm`.

[5] C-H. Chen, J-S. Lee, and Y-N. Sun. Wavelet transformation for grey-level corner detection. *Pattern Recognition*, 28, No. 6:853–861, 1995.

[6] N. Guarino. Formal ontology in information systems. In N. Guarino, editor, *Proc. of Formal Ontology in Information Systems*, pages 3–15, Trento, Italy, 6-8 June 1998. IOS Press, Amsterdam.

[7] M. M. Kokar and J. Wang. An example of using ontologies and symbolic information in automatic target recognition. In *Sensor Fusion: Architectures, Algorithms, and Applications VI*, pages 40–50. SPIE, 2002.

[8] D. McGuinness. Ontologies and online commerce. *IEEE Intelligent Systems*, 16(1):8–14, 2001.

[9] D. R. Smith. KIDS: a semi-automatic program development system. *IEEE Transactions on Software Engineering*, 16 (9):1024–1043, 1990.

[10] Y. V. Srinivas and R. Jullig. Specware^{TM}: Formal support for composing software. Technical Report KES.U.94.5, Kestrel Institute, 1994.

[11] M. Stickel, R. Waldinger, M. Lowry, T. Pressburger, and I. Underwood. Deductive composition of astronomical software from subroutine libraries. In *Proceedings of the Twelfth International Conference on Automated Deduction (CADE-12)*, pages 341–355, 1994.

[12] Y. Y. Tang, L. H. Yang, J. Liu, and H. Ma. *Wavelet Theory and Its Application to Pattern Recognition*. World Scientific Publishing Co. Pte. Ltd., 2000.

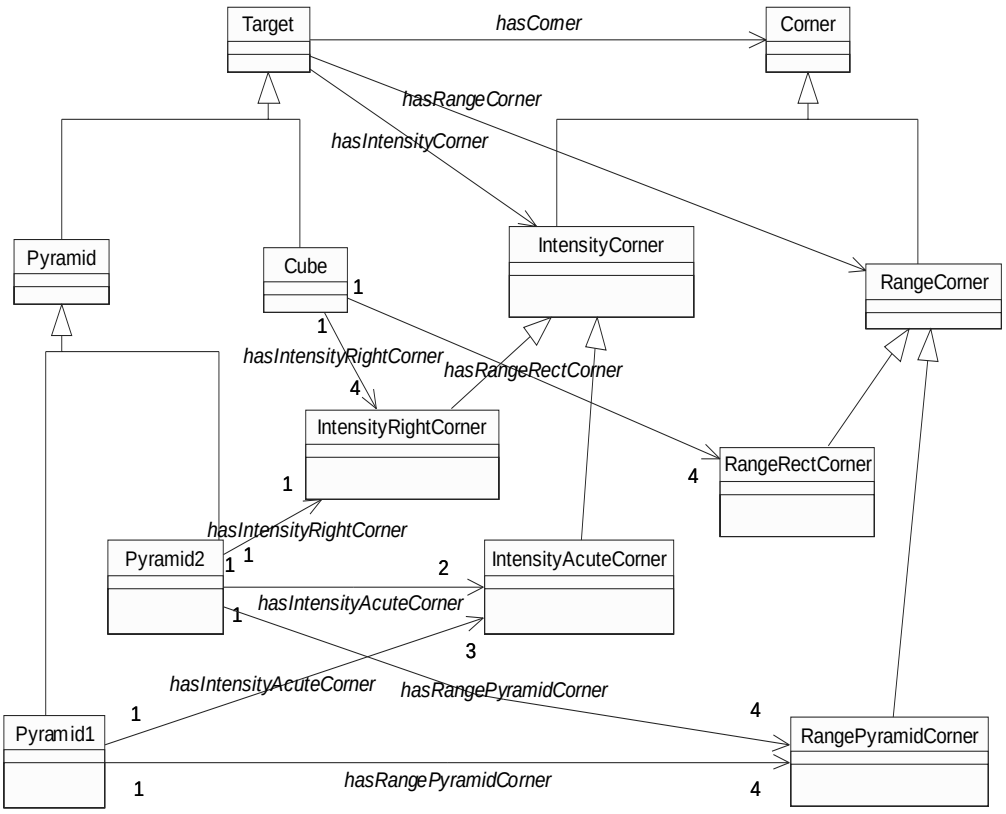

Figure 5: UML Representation of an Ontology of Geometric Objects