An Example of Using Ontologies and Symbolic Information in Automatic Target Recognition

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

Abstract

The goal of this paper is to show an approach to target recognition (ATR) that allows for efficient updating of the recognition algorithm of a fusion agent when new symbolic information becomes available. This information may, for instance, provide additional characterization of a known type of target, or supply a description of a new type of target. The new symbolic information can be either posted on a web page or provided by another agent. The sensory information can be obtained from two imaging sensors. In our scenario the fusion agent, after noticing such an event, processes the new symbolic information and incorporates it into its recognition rules. To achieve this goal the fusion agent needs to "understand" the symbolic information. This capability is achieved through the use of an ontology. Both the fusion agent and the knowledge provider (it may be another software agent or a human annotator) know the ontology, and the web based information is annotated using that ontology. In this paper we describe the approach, provide examples of symbolic target descriptions, describe an ATR scenario, and show some initial results of simulations for the selected scenario. The discussion in this paper shows the advantages of the proposed approach over that in which the recognition algorithm is fixed.

1 Introduction

The problem of automatic target recognition (ATR) has been the subject of research for many years. For various types of scenario very efficient algorithms exist that guarantee good accuracy of recognition. In most of those cases there is an assumption that the

types of target are known at the time of design of an ATR system. While this kind of an assumption is sufficient for many practical situations, there are still situations in which such an assumption is not valid. We call such situations *dynamic environments*. In dynamic environments the types of target are known in advance and thus cannot be hard coded into an ATR system. Instead, an ATR system for dynamic environments should be able to incorporate knowledge of new targets during the run time, rather than during the design time.

In this paper we investigate a scenario in which the knowledge of targets is stored in an *ontology* [3, 4, 6]. We do not assume that the ontology is known in advance; it can be updated at the run time of the ATR system. One of the questions in such a scenario would be - what language to use to represent the target knowledge? The selection of a language is not a trivial question at all. A language may be either too simple, and thus not have enough expressive power, or too complex, and thus unmanageable. Another feature of a language could be its *compactness*, i.e., the ability to express information in short and concise fashion. But perhaps even more importantly, the language should be accepted by the user community. Even the most perfect language would be useless if nobody wanted to use it. Fortunately, recently a number of languages have become "de-facto" standards; they have been accepted by wide communities of users. In software engineering, UML (Unified Modeling Language) [1] has become a standard and is officially maintained by the Object Management Group (OMG). The knowledge representation community seems to be converging on a standardized language called DAML (which stands for DARPA Agent Markup Language) [7, 5]. This standard is maintained by the World Wide Web Consortium (W3C).

In this paper we present a case study of using DAML for representing ontologies for target recognition. The idea is that either an interactive user, or another source of information (e.g., intelligence) can provide information about new targets. The information is communicated in DAML. A DAML-aware ATR system can read such information and incorporate it into its processing algorithms.

In Section 2 we present a simple scenario which we use to explain the idea of ontology-based target recognition. Then in Section 3.4 we describe all the details of our system. Finally, in Section 4 we present conclusions.

2 Scenario

The main aspect of the scenario is that the types of target are not known in advance. In our experiments we simulated a world of geometric objects. Examples of such two targets are shown in Figure 1. 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.

We simulated two kind of sensor, an intensity sensor and a range sensor. Images from those sensors, as is easy to imagine, would appear as compositions of two kinds of

geometrical shapes - rectangles and triangles (possibly skewed).

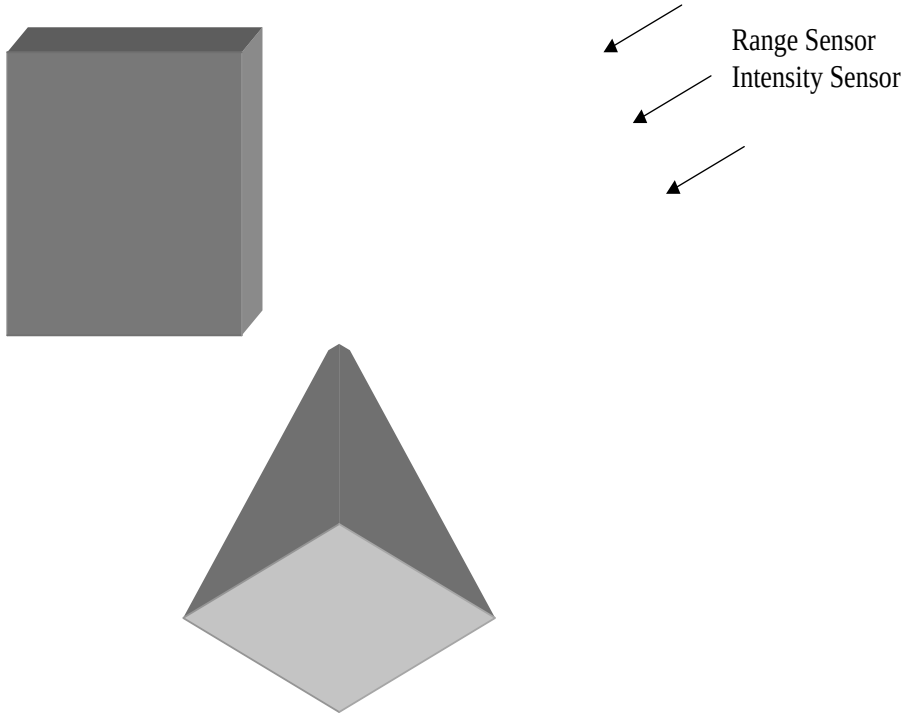

Figure 1: Examples of Targets

In our scenario we do not need to assume that we know any of these targets; the descriptions of the targets in the language defined by the ontology can be input at any time, for instance, at the time of operation of the system that implements the OBTRA.

3 Ontology Based Target Recognition

The Ontology Based Target Recognition Approach (OBTRA) is shown in Figure 2. OBTRA takes sensory inputs from two different sensors. It then processes signals (images) from the two sensors to extract *features*. In our example the features are *corners*. In

Section 3.2 we describe the types of corner we used in our experiments. Then in Section 3.1 we describe the algorithms we used for corner detection and characterization. The features extracted from two images are then passed to the target recognition algorithm labeled *Fusion and Recognition* in Figure 2. The Fusion and Recognition algorithm reads descriptions of targets from the Ontology. These descriptions are expressed in the DAML language [7, 5]. A fragment of a DAML Ontology describing targets is presented in Section 3.3. Note that the ontology is created by the user and that it can be changed dynamically during the operation of the system. In Figure 2 we indicate it by showing an icon representing the user. But this information can also we obtained through other means, for instance from intelligence channels. The recognition algorithm is described in Section 3.4.

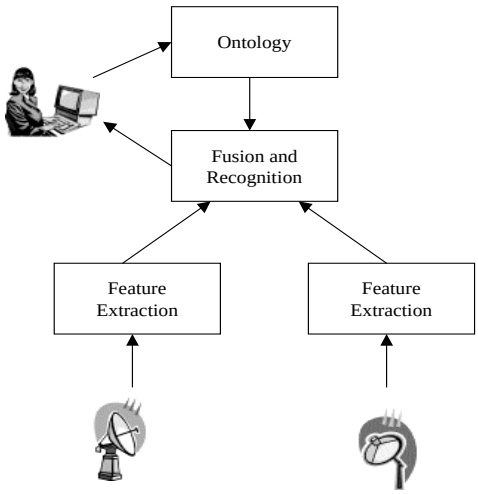

Figure 2: Ontology Based Target Recognition

3.1 Corner Detection Algorithms

As we mentioned earlier in the paper, we used *corners* as features for target recognition. We used wavelet transforms for corner detection. In this experiment, the following type of basic wavelet was adopted [8]:

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

and

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

We also used the modulus of the wavelet transformation at a given scale s as :

$$MC(s,x,y) = (|W_1C(s,x,y)|^2 + |W_2C(s,x,y)|^2)^{\frac{1}{2}} \tag{3}$$

and the orientation function

$$OC(s,x,y) = tan^{-1}(\frac{W_2C(s,x,y)}{W_1C(s,x,y)}) \tag{4}$$

Here $W_1C(s,x,y)$ and $W_2C(s,x,y)$ are wavelet transforms of a function $C(x,y)$ at scale s.

There are three properties that we can use for corner detection:

1. The *scale proportion property*, expressed as:

$$\frac{MC(s_1,x,y)}{MC(s_2,x,y)} = \frac{s_1}{s_2}$$

   that is satisfied by corner points and isolated edge points (edge points not very close to corner points). The ratio $\frac{s_1}{s_2}$ is called the *scale ratio*.

2. The property of the *orientation variance* near the corner point. It is much larger near the corner point than near the isolated edge points.

3. The property of *scale invariance*. This property is that the orientation at the corner point and the isolated edge point are scale independent.

The feature extraction (corner extraction) algorithm [2] consists of the following steps:

1. Use the first property, i.e., $\frac{MC(s_1,x,y)}{MC(s_2,x,y)} - \frac{s_1}{s_2} = 0$, to find edge points of an image.

2. Eliminate edge points using the second property. After this step, only corner a point and the edge points that are very close to the corner point are left.

3. Locate corners using the third property. In this step, the edge points are eliminated and the corner point is located.

3.2 Corner Types

We defined two kinds of corner for each sensor type. For the range sensor we had *rangeCornerType1* and *rangeCornerType2*. Similarly, for the intensity sensor we had *intensityCornerType1* and *intensityCornerType2*.

RangeCornerType1 is a rectangular corner, with all the high-intensity points within the corner having the same intensity. It is the corner found in the range image of a cube. *RangeCornerType2* is a rectangular corner in which the high-intensity points within the

corner have varying intensities. The edge points and the corner point have the same intensity, but the intensity increases toward the center of the object. It is the corner found in range images of a pyramid. *IntensityCornerType1* is a rectangular corner or an obtuse corner, with all the high-intensity points within the corner having the same intensity. It is the corner found in intensity images of a cube. *IntensityCornerType2* is an acute corner, with all the high-intensity points within the corner having the same intensity. There can be either three or two of this kind of corners and one *IntensityCornerType1* in an intensity triangular image.

The corner detection algorithm allows us to find the positions of corners. After finding a corner's position, we need to analyze a small rectangular area centered at the corner point. In this area, we need to count how many points in it have high intensity to find the ratio of high-intensity points to low-intensity points. The division between the high-intensity and the low-intensity points represents the angle (in degrees) of of the corner. This method can also be used to discriminate the two kinds of intensity corner. For range corners, we need to do one more thing - check the area of the corner and see whether all the high-intensity points have the same intensity or not. This allows us to discriminate between the two kinds of range corner.

3.3 Target Ontology

An ontology captures the basic terminology (concepts) of the domain of interest and the relationships among the concepts. In the following we show a small ontology for target recognition. First, in Section 3.3.1 we present the ontology in the UML language [1]. 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 taste of the DAML representation of the ontology is given in Section 3.3.2.

3.3.1 UML Representation

A small piece of an ontology for target recognition is shown in Figure 3. In this figure we use UML (Unified Modeling Language) [1] as a representation language. The two main constructs in UML are *Class* and *Association*. Classes are represented as rectangles, while Associations as arrows. In the figure we show only names of the Classes and of some Associations. Additionally, we show multiplicities of the Associations. 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 *TargetType1* and *TargetType2*. *TargetType2* is a generalization of two subclasses, *TargetType2_1* and *TargetType2_2*.

There are two kinds of *Corner*, *RangeCorner* and *IntensityCorner*, each of which

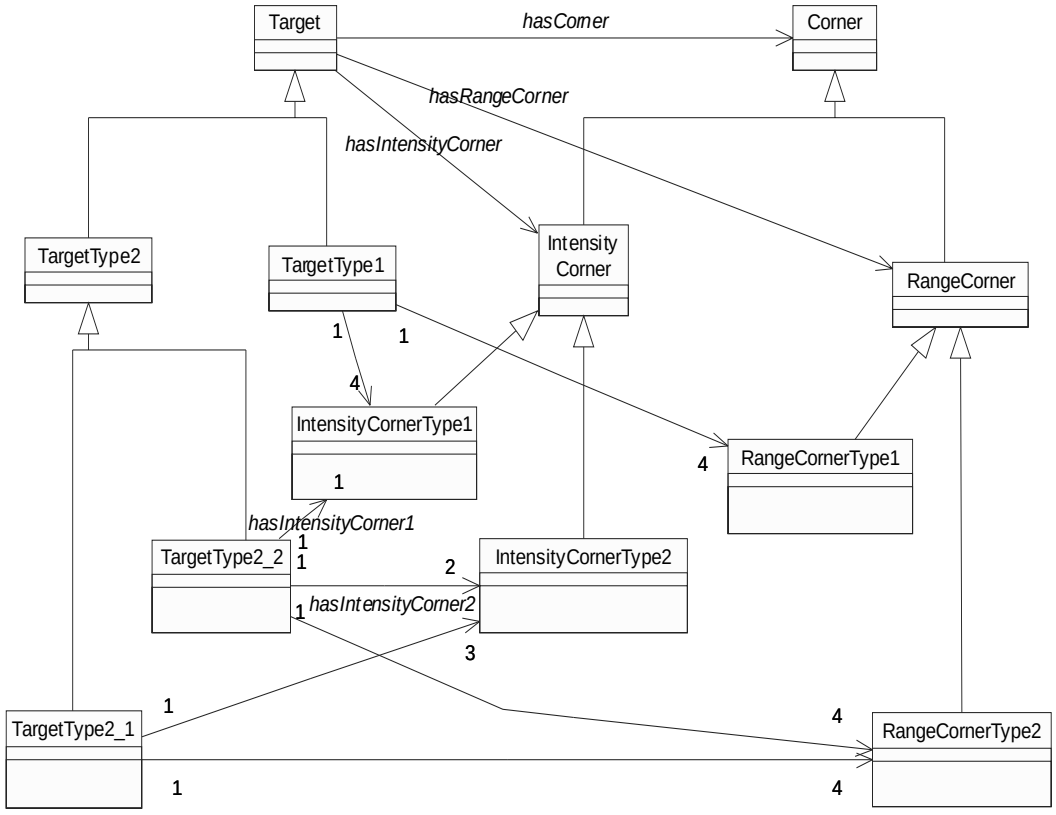

Figure 3: UML Representation of an Ontology for Target Recognition

is further subclassified into two: *RangeCornerType1*, *RangeCornerType2* and *IntensityCornerType1 IntensityCornerType2*, 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 *TargetType1* must have four corners of type *IntensityCornerType1* and four corners of type *RangeCornerType1*. A target of type *TargetType2_2* must have one corner of type *IntensityCornerType1*, two corners of type *IntensityCornerType2* and four corners of type *RangeCornerType2*.

3.3.2 DAML Representation

A fragment of the DAML representation of the ontology shown in Figure **??** is shown below. This fragment shows the description of the corner type *IntensityCornerType1* and of the target type *TargetType1*.

```
<daml:Class rdf:ID="IntensityCornerType1">
  <rdfs:label>IntensityCornerType1</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="TargetType1">
  <rdfs:label>TargetType1</rdfs:label>
  <rdfs:comment>
    TargetType1 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="#RangeCornerType1"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="4">
      <daml:onProperty rdf:resource="#hasIntensityCorner1"/>
      <daml:toClass rdf:resource="#IntensityCornerType1"/>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

3.4 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 *TargetType1* it must have the corners as specified in the ontology. This means that the ontology should be used by some executable code. In our experiment we wrote a simple program to translate such ontologies into Prolog rules.

Here are just a few examples of Prolog rules. The first two rules state capture the

fact that the ontology (see, for instance, Figure **??**) stipulates that *TargetType2* is either a member of the class *TargetType2_1* or of *TargetType2_2*.

```
targetType2(X) :- target(X), targetType2_1(X).
targetType2(X) :- target(X), targetType2_2(X).
```

As another example, the following rule states that a given object is of type *TargetType2_1*.

```
targetType2_1(X) :- target(X),intensityCornerType2(Y),
                    hasIntensityCorner2(X,Y),
                    multiplicity(hasIntensityCorner2,3),
                    rangeCorner(Z), hasRangeCorner2(X,Z),
                    multiplicity(hasRangeCorner,4).
```

This kind of rules are applied to a data base of facts generated by the Feature Extraction blocks, as shown in Figure 2. These blocks detect corners and classify them into the four classes, as described before. The data base facts are in the form:

```
target(t).
corner(c1).
corner(c2).
corner(c3).
...
corner(c7).
hasCorner(t, c1).
....
hasCorner(t,c7).
intensityCornerType1(c1).
.....
rangeCornerType1(c5).
```

4 Conclusions

The main goal of this work was to investigate a scenario of Automatic Target Recognition in which the targets are not know at the time of the design of the system. Instead, we presumed that the descriptions of targets are supplied through a communication channel. The descriptions are either generated by a human operator or are supplied by an intelligence source. The case of supplementing description of a known target by some additional information about its features is a special case of this scenario. This kind of scenario is not typical for ATR applications. In our approach to the solution of

such a problem we assumed that the descriptions of targets are expressed in the DAML language - the language for specifying ontologies. The goal of this analysis was to assess the feasibility of this kind of approach to ATR.

We were able to implement prototypes, mostly rather simple cases, of the elements of a system that would solve such a problem. Consequently, we can claim that this approach is feasible and promising. We must say that we are not making any claims about the implementation of our algorithms in terms of the accuracy of recognition for any special collection of objects (even the objects that were used in our simulations). This was not our goal in this investigation. For instance, we did not try to prove that our rules would correctly discriminate pyramids from cubes. But we do not see any reasons for not being able to develop robust algorithms for any kinds of scenarios like the one used in this research.

Various directions of future research need to be pursued to make the development process of such systems easy and robust. For one, there is a need for good tools that would help the ontologist to develop ontologies for various domains. The tools should be able to tell the user that the terms that s/he is trying to add to the ontology are already defined, that adding specific classes or properties would make the ontology inconsistent, that some of the facts that the ontologies is trying to add to the ontology are redundant. Independently of the tools, there is a need of developing an ontology for the domain of target recognition that would be acceptable to many users. (We do not believe it is possible to satisfy all the users with one ontology.) Such ontologies would capture knowledge of many targets of interest. The descriptions of the targets would be expressed in terms of different kinds of features. Once we admit the existence of multiple ontologies, it would be important to have tools that can use multiple ontologies, in spite of the inconsistencies that exist among them.

Acknowledgments

This research was partially supported by a grant from the Office of Scientific Research under contract No: F49620-98-1-0043.

References

[1] 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`.

[2] 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.

[3] J. Heflin, J. Hendler, and S. Luke. Coping with changing ontologies in a distributed environment. In *AAAI-99 Workshop on Ontology Management*. MIT Press, 1999.

[4] J. Heflin, J. Hendler, and S. Luke. SHOE: A knowledge representation language for Internet applications. Technical Report `www.cs.umd.edu/projects/plus/SHOE`, Institute for Advanced Studies, University of Maryland, 2000.

[5] J. Hendler and D. McGuinness. The DARPA Agent Markup Language. *IEEE Intelligent Systems*, 15, No. 6:67–73, 2000.

[6] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilde. An environment for merging and testing large ontologies. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000)*, 2000.

[7] DARPA Agent Markup Language Web Site. `www.daml.org`, 2001.

[8] 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.