Reduction of BRL/CAD Models and Their Use in Automatic Target Recognition Algorithms*

Mark R. Stevens and **J. Ross Beveridge** and **Michael E. Goss**
Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873

Abstract

We are currently developing an Automatic Target Recognition (ATR) algorithm to locate an object using multi-sensor data. The ATR algorithm will determine corresponding points between a range (LADAR) image, a color (CCD) image, a thermal (FLIR) image and a BRL/CAD model of the object being located. The success of this process depends in part on which features can be automatically extracted from the model database. The BRL/CAD models we have for this process contain more detail than can be productively used by our ATR algorithm and must be reduced to a more appropriate form.

This paper presents algorithms we are developing in order to reduce the BRL/CAD models a level of detail appropriate for the ATR algorithm. A secondary feature of these algorithms is to also maintain a parallel version with details sufficient to appear realistic when rendered. This rendering enables the ATR system to animate its search procedure for monitoring and debugging. Model reduction begins by converting the Constructive Solid Geometry BRL/CAD model to a set of polyhedra. All polyhedra that are completely contained within other polyhedra are automatically discarded. We will also allow the user to discard objects based on a specified minimum bounding box.

From the reduced model, we can extract 3D point sampled surface information as well as 3D features generated by the model silhouette. These features are then used by the ATR algorithm. Algorithms for determining the visible silhouette surfaces and the features for an arbitrary viewpoint are developed in this paper. The ATR algorithm will use the three dimensional features to refine a pose estimate of the model relative to sensors. This pose in turn provides a basis for measuring the quality of the match between 3D model features and sensor features.

*This work was sponsored by the Advanced Research Projects Agency (ARPA) under grant DAAH04-93-G-422, monitored by the U. S. Army Research Office.

1 Introduction

Model-based object recognition techniques often identify modeled objects in a scene by relating stored geometric models to features extracted from sensor data [Bes88]. A common goal is to place a 3D model in a scene at roughly the correct position, with similar scale, and orientation to the object in the scene. In this way, the fidelity between the hypothesized instance of the modeled object and the actual feature data can be compared. To perform this comparison requires that relevant and comparable information be extracted from both the sensor data as well as the model database. If object geometry is to constrain recognition, then the representation must allow the recognition system to reason about the geometric entities in a scene, and therefore must represent the geometric form of the objects. However, most geometric modeling schemes are designed for purposes other than automated recognition, and they typically do not make explicit the information needed most by the recognition process.

The models used by the automatic target recognition (ATR) process originate in the Ballistic Research Laboratory's Computer Aided Design (BRL/CAD) model format. The BRL/CAD model format is an excellent example of a mature modeling scheme in which extremely detailed and complete information about a vehicle can be represented. However, due to the implicit nature of the Constructive Solid Geometry (CSG) format of the BRL/CAD models, explicit 3D vertex, edge, and face information is not directly available. Thus extracting feature information from a CSG model can be computationally expensive. What we need is a model format which allows easy extraction of 3D face and edge information and efficient removal of non-relevant features.

This paper presents our initial efforts to build a semi-automated tool for reducing a complete and detailed BRL/CAD vehicle models to a *three* dimensional form more appropriate for automated recognition. The first step in this process is to transform the CSG representation into polyhedral form. The next step is to provide an inter-

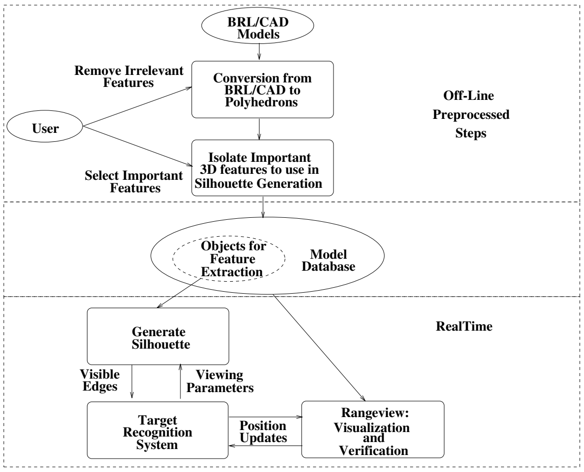


Figure 1: Overview of Entire Process

active application which allows a user to selectively reduce the complexity of the object. It is important that this application allow the user to visually inspect the consequences of each successive reduction and make modifications and corrections as appropriate. The third step must automatically extract 3D features directly comparable to measurable features in LADAR, FLIR and CCD imagery. The principle topic of this paper will be this third and final phase. The complete sequence of operations is shown in Figure 1.

2 The ATR Algorithm

The ATR system being developed at Colorado State University will attempt to locate military vehicles using input data from several different sensors. The data set used to test the algorithms was obtained in November of 1993 at Fort Carson, Colorado [BPY94]. Data from three sensor modalities was collected: color (CCD) images, thermal (FLIR) imagery, and depth (LADAR) information.

The ATR algorithm being developed will use a three stage strategy [BHP94]. First, a detection process suggests regions of interest within the image worth further consideration as possible targets. An innovation at this stage is the use of color as an additional detection queue [BDHR94]. The second stage extends LADAR probing techniques [BJLP92] to generate target type and target pose hypotheses. Finally, given object type and pose hypotheses, an error reduction approach will generate a best-fit match between the sensor and model features [aJRB94]. Using both multi-sensor data and detailed geometric constraints derived from the models, it is expected this final stage will be highly reliable and discriminating. A sample of the multi-sensor data is shown in Figures 2, 3, and 4. These are images show the same scene, containing an M113, in each of the three sensor modalities.

Figure 2: CCD Image of M113

It is crucial to understand that is not necessary, nor even desirable, for the model reduction algorithm to produce a perfect, continuous 3D silhouette contour. Even defining what is meant by a ‘perfect’ silhouette can become problematic for complex models, and setting such definitional issues aside, the amount of computation required to produce a continuous well formed silhouette can be very large. What drives the required detail of the models is the quality of the sensor data, and typically the sensor data is low resolution and noisy. Effort expended extracting detail from object models not manifest in the sensor data is effort wasted.

To help us better understand the character and quality of the multi-sensor data, we have developed a visualization and verification tool known as RangeView [GBSF94], [GBSF95]. RangeView allows all three sensor images to be viewed simultaneously as well as the current position of the model in relation to the scene. The user also has the ability to interactively modify any of the viewing parameters to allow a better interpretation of the scene. RangeView also provides a means of animating the ATR process. The ATR algorithm and RangeView communicate, thus enabling RangeView to update the display of the hypothesized target model relative to the data.

3 Converting CSG to Polyhedra

Polyhedral representations can be used to describe any type of object that can be bounded by a set of planar faces [Sny92]. The two main advantages of polyhedral models over CSG representations are: the ease with which they can be rendered on modern graphics hardware, and the ability to readily analyze their shape.

For matching to object silhouettes in FLIR and CCD data, the relevant model representation is a set of 3D lines rep-

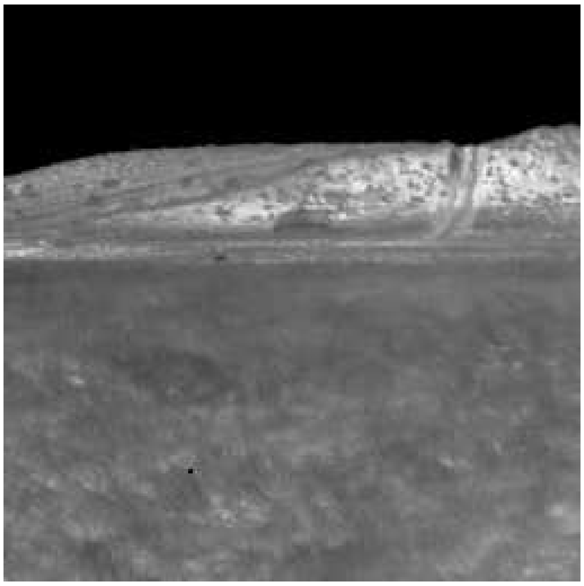
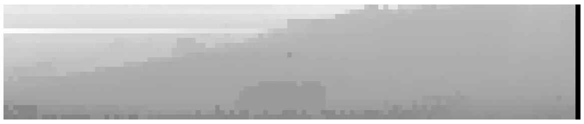
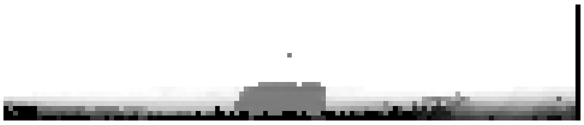


Figure 3: FLIR Image of M113(Grey scale)



a. full 100 to 1,000 foot range



b. range highlighted about vehicle

Figure 4: LADAR Image of M113 (Grey scale)

resenting visible silhouette edges. It is key that these edges still be represented in 3D, as opposed to projected into 2D templates, because the 3D structure can constrain the object pose refinement portion of the recognition process [KH94; aJRB94].

Due to the implicit nature of the BRL/CAD CSG representation, vertex, edge and face information needed in order to derive 3D silhouette information is not directly available. However, model point/line information is readily available in a polyhedral model, and thus the first step is to transform models into a polyhedral representation.

Several different algorithms have been developed to convert a model from CSG into a set of polyhedra [RV85; LTH86; PS86; NAT90]. These methods are all based on the principal that each object used in a boolean operation can be split into a set of planar faces, each of which lies completely inside or outside the other object. A classification algorithm can then be used to determine which faces are retained.

3.1 The Conversion Algorithm

We have chosen to utilize the algorithm presented in [LTH86] for convert the CSG representation into polyhedra. The conversion algorithm is summarized below along with some minor refinements which we found improved the quality of the resulting model. For a complete description of the algorithm the reader is referred to [LTH86], or [RV85] which provides a discussion of the mathematical basis for the algorithm.

The conversion algorithm traverses the CSG tree transforming the leaf nodes from BRL/CAD primitives into polyhedra. Currently our algorithm only supports a subset of the available BRL/CAD primitives: five types of arbitrary convex polyhedra (ARB4, ARB5, ARB6, ARB7, ARB8), right circular cylinder (RCC), and an interpolated curve (ARS). The conversion method used is based on the approximations Xmged uses to visualize the BRL/CAD models.

The leaf nodes are converted first, after which each node in the tree is visited with an in-order traversal. The two children of each node are merged into one object by applying the appropriate regularized set (R-set) operation. The R-set operation will generate a polyhedral operation which can then be merged with other objects in the tree.

The R-set operations begin by subdividing an object so that none of its faces intersect any face of the other object. After the splitting phase has occurred, a classification process labels each face of each object as inside or outside the other object. Then the classification routine casts a ray from the center of each face in the direction of the face normal. Based on the intersection of the ray with all the faces in the other object, the face is labelled as either INSIDE or OUTSIDE.

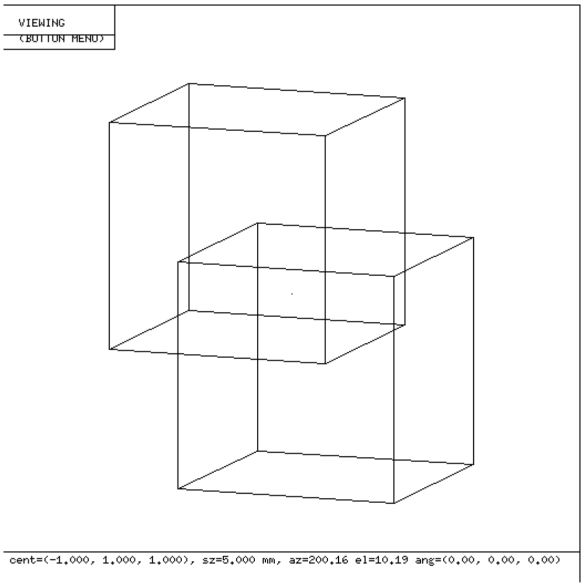
The final phase uses a table look-up to determine which faces of each object are kept, based on the label and the appropriate R-set operation from the CSG tree. The end result is a new solid object which can be used again as an object in the entire operation. The process continues until the entire CSG tree has been traversed and the model has been converted to its new polyhedral representation.

We have made several additions to the algorithm presented by Laidlaw [LTH86]. Their splitting algorithm tends to fragment the faces of an object into many smaller faces when one of the objects is small compared to the other. We have added a user specified parameter to discard the second object if the magnitude of the bounding box diagonal is below a certain percentage of the diagonal of the first object. By setting this parameter correctly a user can accomplish two tasks: the resulting model will contain a desired level of detail, and the fragmentation problem can be avoided.

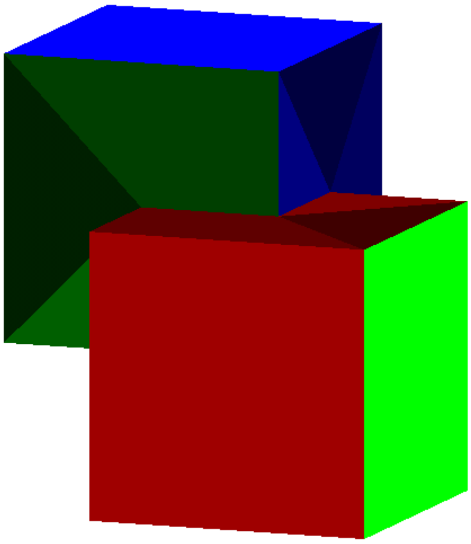
We have also modified the data structures used by Laidlaw. We have chosen to use a boundary representation (B-rep) structure. The B-rep allows easy access to information about which two faces share an edge, as well as the edges corresponding to a face. These relationships make the addition of a merging phase to the algorithm quite simple. After the final stage of the conversion to polyhedra is accomplished, each edge in the model is examined. If a pair of faces that share an edge have equal normals, they are coplanar and are merged to form one face. The merging reduces the number of planar facets, and reduces the complexity of the final model.

An example of a model that has been converted to a polyhedral representation is shown in Figures 5, and 6. Figure 5a shows an image captured from Xmged, and Figure 5b shows the polyhedral model, without the merge operation being applied. Figure 6 shows both the Xmged and the polyhedral versions of an M113 vehicle. In order to reduce the complexity of the M113 model, the splitting parameter was set high. The result is an object with a small number of splits. In the polyhedral images, each face has been rendered with a unique color.

Many of the vehicle models that we are using contain large numbers of structures that are completely contained within larger structures. For example, the M35 BRL/CAD model contains many structures representing the engine block. However, most of these structures are completely contained within the vehicle shell. The ATR algorithm is concerned with matching the shell of the vehicle and is not concerned with the interior engine components, which are never visible. Initially, it was believed that a separate algorithm would be needed to remove these occluded objects. After further consideration, it became apparent that the conversion can be slightly modified to remove these occluded objects. The modifications alter the classification and lookup phases of the algorithm, by rejecting faces contained within a larger object.



a. Xmged image



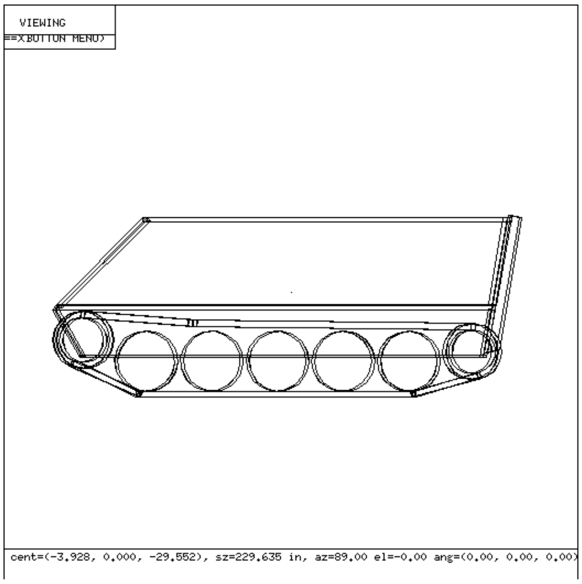
b. Unique color rendering of polyhedra

Figure 5: Union of Two Cubes

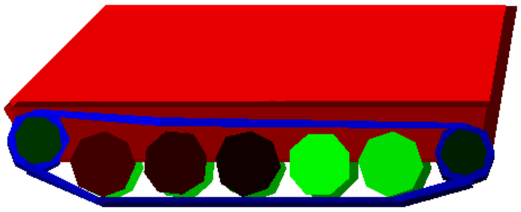
4 Reducing Model Complexity

The BRL/CAD models we have contain far more information than we need. Since converting the models from their initial CSG form does not decrease the level of model detail, we are forced to manually reduce the model complexity. Currently we are using Xmged to remove model primitives before the conversion process takes place. However, this method is tedious and error-prone.

We are developing a very simple interface which will allow the user to select particular facets of the model for use in the feature extraction process. The interface used for interactive reduction presents three different windows. The first renders all faces of a model. The second is a control box that allows the user to modify the current viewing parameters, and the third contains the visible faces the user has chosen for inclusion in the reduced model. By using the mouse, the user can select faces on the original



a. Xmged image



b Unique color rendering of polyhedra

Figure 6: M113 Vehicle

model in the first window, and the features picked will appear in the third window.

5 Extracting 3D Silhouette Features from Polyhedral Objects

After reducing the detail in the model to a level appropriate for recognition, the next step is to produce model features which are directly comparable with features extracted from sensor data. Our ATR algorithms dynamically update the 3D position and orientation of the object model relative to the sensors so as to align the visible contours in FLIR and CCD with silhouette inducing edges in the 3D model [KH94; aJRB94].

The ATR algorithm needs to be aware of which silhouette edges are visible for any given viewpoint. To generate a list of visible 3D edges, each face is projected onto a 2D image plane. Then, a reverse mapping procedure is used to recover the 3D information that has been lost by this projection. The 3D line segments which produce silhouette edges are provided to the ATR algorithm. The ATR algorithm is initiated with an estimated viewpoint and will make a request to the silhouette generation procedure, which will provide an initial set of visible features. If, during recognition, this viewpoint is modified significantly, then the ATR algorithm can dynamically request the visible edges for the modified viewpoint.

Earlier feature extraction processes have focused on development of a model representation known as the aspect graph. Aspect graphs are founded on the notion that there are regions on a view sphere in which all viewpoints in that region share a constant model topology [KvD76; KvD79]. The arcs in the graph represent movement from one set of visible features to another, and this characteristic has been exploited in the ATR process [PD87], [Pla88]. Other techniques have used projection techniques similar to ours [Bon86; GM88; SJ89]. A more relevant discussion of feature extraction appears in [SD92], and describes a simple test we implement for determining whether or not an edge represents a silhouette edge.

The aspect graph is a stored representation in which visible features are precomputed for all relevant viewpoints. We have chosen instead to take a demand driven approach in which visible features are determined as they are needed. However, whether or not a precomputed or dynamic approach is better suited for our task is not yet clear. It is our hope that the on-demand approach will prove reasonable using suitable hardware graphics accelerators. However, the same underlying algorithm can be used off-line to build up a stored aspect graph [KD87]. The more important issue is the nature and form of the algorithm used to generate the visible 3D features for different viewpoints.

Our viewing geometry parameters are specified in spherical coordinates, with each viewpoint represented as an elevation and a counterclockwise azimuth pair $(\phi,\ \theta)$, and a radius or distance, d, from the sphere center. The model is assumed to have a fixed orientation, and is centered at the sphere origin. The (ϕ, θ) parameters therefore give the representation two degrees of freedom.

5.1 The Silhouette Feature Extraction Algorithm

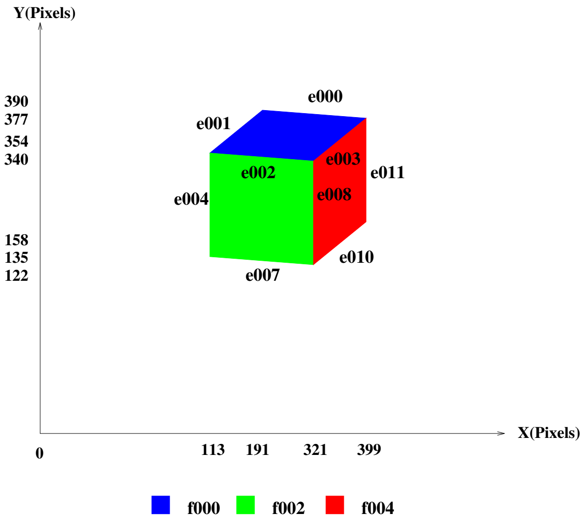
Feature extraction techniques have typically centered around the notion of projecting each face of a model onto a viewing plane in order to determine which faces are visible for a specific view point. Our algorithm utilizes accelerated graphics hardware to increase the efficiency of this operation. The main problem is not how to accomplish the projection, but rather how to interpret the projected results and obtain the 3D endpoints of the line segments which correspond to the silhouette of the model. Solving this reverse mapping problem turns out to be a non-trivial task.

The efficiency of our algorithm is dependent upon the capabilities of the accelerated graphics hardware it is running on. Currently, we are running the feature extraction algorithms on a Hewlett-Packard 715/50 CRX24Z, and a Sparc 10ZX. Both machines have 24-bit image planes. We initially chose to use the PEX graphics package as our interface to the accelerated hardware, but plan to move to the OpenGL package. The graphics package chosen does not affect the quality of the results, but does alter the speed at which they are obtained, the portability of the application to other platforms, and the ease with which the functionality can be implemented.

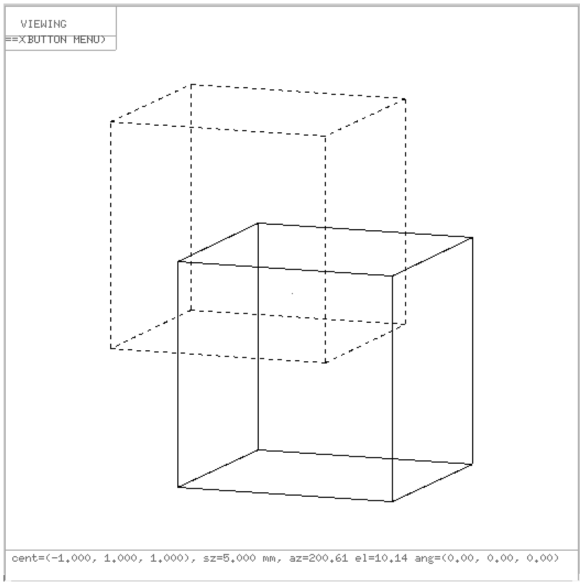
```
foreach (face in model)                          ; Step 1
    Assign face a unique color
    Add face to look-up table
foreach (face in model)                          ; Step 2
    Render in unique color
    (with Hidden Surface Removal)
foreach (pixel in projected image)               ; Step 3
    Obtain visible face based on color
    If silhouette pixel
        Mark face as visible
        Set bit in bitmap
foreach (face marked visible)                    ; Step 4
    foreach (edge of current face)
        if (edge is already marked visible)
            Then mark edge not visible
            Else mark edge visible
foreach (edge marked visible)                    ; Step 5
    Convert to pixel coordinates
    Follow pixel line to find t_in and t_out
    If (line too short) Discard line
Return 3D endpoints
```

Table 1: Pseudo-code for feature extraction algorithm

The algorithm breaks down into a five step process. The pseudo-code for the algorithm is presented in table 1. The first step of the algorithm labels each face in the model with a unique color, and stores it in a color look-up table. We are using the colors as face identifiers, so that for any given color an index into the look-up table can be computed. The table will allow us to determine which faces are visible based on the colors that appear in the rendered image plane. The only limit to the number of faces our algorithm can handle is the number of colors available to the machine (over 16 million). This should be more than adequate for the models we are using.



a. Unique Color Rendering with labels



b. Xmged image

Figure 7: Intersection of Two Cubes

The second step is equivalent to the traditional feature extraction process of projecting each facet to determine the visible portion. We are using orthographic projection to render each face of the polyhedral model in its unique color. Orthographic projection is an adequate approximation since the objects being located will be a significant distance from the sensor in relation to the size of the model. Orthographic projection also simplifies the calculations used in the recovery of the 3D feature information.

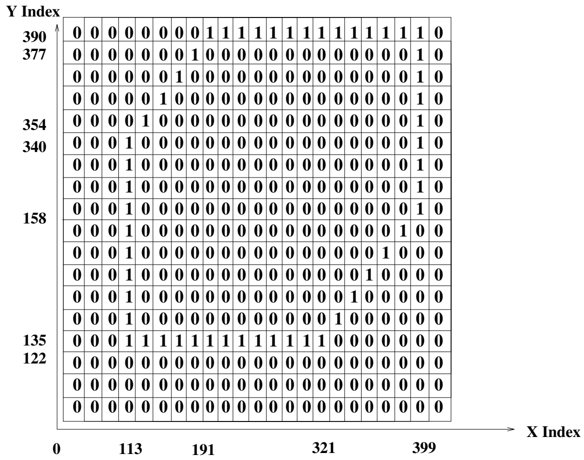
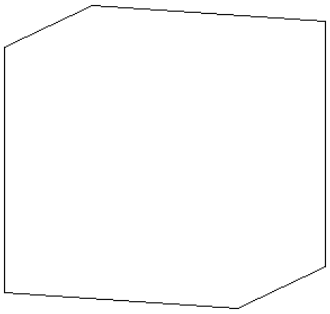


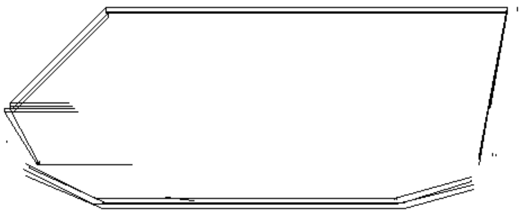
Figure 8: Bit map representing visible silhouette features

The viewing geometry used in the projection is determined by the ATR algorithm, and represents the estimated position of the model relative to the sensors. Shown in Figure 7a is a sketch of the faces for the intersection of two cubes, each rendered in its unique color, with edge and face labels. Figure 7b shows a similar model from the same viewpoint in Xmged. This simple example shows the difficulty of obtaining the visible silhouette features directly from the implicit CSG representation. The 3D vertex, edge and face information explicitly available in our polyhedral model greatly simplifies the reverse mapping stage where the 3D silhouette features are recovered from the 2D projection.

The third step determines which pixels represent silhouette pixels, and which faces generated them. A bitmap, similar to Figure 8 but at a higher resolution, is generated from the projected image with the bit for each pixel representing whether or not the pixel is on the silhouette. A pixel is classified as a silhouette pixel if any of the eight connected neighbors of the pixel are the same color as a predetermined background color. As each bit in the bitmap is assigned an “On” value, the face which generated the pixel is marked as visible. Locating the correct face is just a simple table look-up based on the color of the pixel. Each edge associated with a visible face is also marked as visible. We use a simple heuristic to reduce the number of edges needing to be clipped due to occlusion. An edge cannot be on the silhouette if both faces which share the edge are visible [SD92]. As an edge is marked as visible, a simple check is made to determine if the other face sharing the edge is visible. If the other face is visible, the edge can not lie on the silhouette and it is marked not visible. All edges marked visible are stored in a visible



a. Intersection of two Cubes



b. Simplified m113

Figure 9: Results of feature extraction algorithm

edge list for use in the next phase of the algorithm.

The list of visible edges is traversed to determine the portion of each edge actually visible, and which endpoints need to be clipped due to occlusion. The end points of each line are then converted to pixel coordinates, and a parametric representation is used to march along the line formed by the two pixel endpoints. The line following algorithm begins at the pixel of the first vertex for the edge. At each distinct pixel along the line, a neighborhood around the pixel in the bitmap is examined. The traversal continues until an edge is found in the bitmap.

Edge	Face1		Face2		Edge	Vertex	
	Label	Visible	Label	Visible	Visible	Start	End
e000	f000	Yes	f005	No	Yes	(-1.5, 1.5, 1.5)	(-0.5, 1.5, 1.5)
e001	f000	Yes	f001	No	Yes	(-0.5, 1.5, 1.5)	(-0.5, 1.5, 0.5)
e002	f000	Yes	f002	Yes	No	-	-
e003	f000	Yes	f004	Yes	No	-	-
e004	f001	No	f002	Yes	Yes	(-0.5, 0.5, 0.5)	(-0.5, 1.5, 0.5)
e005	f001	No	f005	No	No	-	-
e006	f001	No	f003	No	No	-	-
e007	f003	No	f002	Yes	Yes	(-0.5, 0.5, 0.5)	(-1.5, 0.5, 0.5)
e008	f002	Yes	f004	Yes	No	-	-
e009	f003	No	f005	No	No	-	-
e010	f004	Yes	f003	No	Yes	(-0.5, 0.5, 1.5)	(-0.5, 0.5, 0.5)
e011	f004	Yes	f005	No	Yes	(-0.5, 0.5, 1.5)	(-1.5, 1.5, 1.5)

Table 2: Visible 3D segments for Intersection Example

The first pixel found on the edge is used to compute a t value, called t_{in}, for the parametric line. This t value is stored as the beginning portion of the visible edge segment. The line following then continues until either the end point is reached, or an edge bit in the bitmap cannot be found. At this point, another t value is computed called t_{out}, and is stored as the exit point of the line. Due to the affine nature of orthographic projection, proportional distance is preserved between parameterized lines in 2D pixel and 3D model coordinates. We can therefore compute the parametric line in model coordinates, and use the t values obtain from the pixel parametric equation to obtain the 3D endpoints of each visible edge segment. The lines are then given to the ATR algorithm as the visible silhouette features of the model for the given viewpoint. Table 2 indicates the 3D silhouette edges for the intersection of two cubes example introduced in Figure 7

Figure 9a shows the silhouette lines found by the algorithm for the two cubes example, which contains only six different faces. We also applied the feature extraction process to a simplified m113 model which contained approximately four hundred faces. The results were not as exact as the intersection example. Floating point imprecision can cause errors in the conversion from modelling to pixel coordinates. The rasterization results in jaggies and other aliasing problems that can lead the line following algorithm astray.

To solve this problem, we were forced to increase the area of the bitmap searched when clipping the visible edge. However, as the area searched increased, so did the number of edges incorrectly clipped. To address this problem, we added a user definable minimum Euclidean distance required between the endpoints for the line retained. For the images generated we set this parameter to fifteen percent of the diagonal of the model bounding box. Another parameter was later added to remove lines with less than a certain percentage visible. The results, after these modifications, are shown in Figure 9b, and as can be seen there are still some extraneous lines. We expect refinements to the algorithm will further improve these results and a higher quality silhouette will be obtained.

6 Conclusion

We have presented a method to convert a BRL/CAD model to a set of polyhedra which can then be used by our feature extraction algorithm. The feature extraction algorithm is not a conceptually unique approach, but we are using several different implementation strategies in an effort to increase the performance. The next major step is to integrate the extraction routines into the ATR algorithm.

All of the algorithms presented can be considered temperamental, and their performance is dependent upon the proper parameterization. We are continuing to refine each phase of the overall process in attempts to reduce the sensitivity to parameters and increase the stability of the system. The goal is to eliminate, as much as is possible, the intervention of the user in the conversion and extraction routines, and to provide a more robust system.

References

[aJRB94] Anthony N. A. Schwickerath and J. Ross Beveridge. Object to multisensor coregistration with eight degrees of freedom. In *Proceedings: Image Understanding Workshop*, pages 481–490. ARPA, nov 1994.

[BDHR94] Shashi Buluswar, Bruce A. Draper, Allen Hanson, and Edward Riseman. Non-parametric Classification of Pixels Under Varying Outdoor Illumination. In *Proceedings: Image Understanding Workshop*, pages 1619–1626, Los Altos, CA, November 1994. ARPA, Morgan Kaufmann.

[Bes88] Paul J. Besl. Geometric modeling and computer vision. *Proceedings of the IEEE*, 76(8):936–958, August 1988.

[BHP94] J. Ross Beveridge, Allen Hanson, and Durga Panda. Integrated color ccd, flir & ladar based object modeling and recognition. Technical report, Colorado State University and Alliant Techsystems and University of Massachusetts, April 1994.

[BJLP92] James Bevington, Randy Johnston, Joel Lee, and Richard Peters. A Modular Target Recognition Algorithm for LADAR. In *Proc of the 2nd Automatic Target Recognizer Systems and Technology Conference*, pages 91 – 104, Fort Belvoir, VA, March 1992.

[Bon86] Luisa Bonfigliolo. An algorithm for silhouette of curved surfaces based on graphical relations. *Computer-Aided Design*, 18(2):95–101, March 1986.

[BPY94] J. Ross Beveridge, Durga P. Panda, and Theodore Yachik. November 1993 Fort Carson RSTA data collection final report. Technical Report CS-94-118, Computer Science Dept., Colorado State University, January 1994.

[GBSF94] Michael E. Goss, J. Ross Beveridge, Mark R. Stevens, and Aaron Fuegi. Visualization and verification of automatic target recognition results using combined range and optical imagery. In *Proceedings: Image Understanding Workshop*, pages 491–494. ARPA, nov 1994.

[GBSF95] Michael E. Goss, J. Ross Beveridge, Mark R. Stevens, and Aaron Fuegi. Three-dimensional visualization environment for multisensor data analysis, interpretation, and model-based object recognition. In *Proceedings: Visual Data Exploration and Analysis*. IS&T/SPIE, February 1995.

[GM88] Ziv Gigus and Jitendra Malik. Computing the aspect graph for line drawings of polyhedral objects. *IEEE Proceedings on Computer Vision and Pattern Recognition*, pages 654–661, 1988.

[KD87] Matthew R. Korn and Charles R. Dyer. 3d multiview object representations for model-based object recognition. *Pattern Recognition*, 20(1):91–103, 1987.

[KH94] Rakesh Kumar and Allen R. Hanson. Robust methods for estimating pose and a sensitivity analysis. *CVGIP:Image Understanding*, 11, 1994.

[KvD76] J. J. Koenderink and A. J. van Doorn. The singularities of visual mapping. *Biological Cybernetics*, 24:51–59, 1976.

[KvD79] J. J. Koenderink and A. J. van Doorn. The internal representation of shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.

[LTH86] D.H Laidlaw, W.B. Trumbore, and J.F. Hughes. Constructive solid geometry for polyhedral objects. *Computer Graphics*, 20(4):161–170, August 1986. Proceedings of SIGGRAPH 86.

[NAT90] Bruce Naylor, John Amanatides, and William Thibault. Merging BSP trees yields polyhedral set operations. *Computer Graphics*, 24(4):115–124, August 1990. Proceedings of SIGGRAPH 90.

[PD87] Harry Platinga and Charles Dyer. Visibility, occlusion, and the aspect graph. Technical Report 736, University of Wisconsin - Madison, December 1987.

[Pla88] William Harry Plantinga. *The ASP: A Continuous, Viewer-Centered Object Representation for Computer Vision*. PhD thesis, University of Wisconsin at Madison, 1988.

[PS86] L. K. Putnam and P. A. Subrahmanyam. Boolean operations on n-dimensional objects. *IEEE Computer Graphics and Applications*, 6(6):43–51, June 1986.

[RV85] A. A. G. Requicha and H. B. Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE*, 73(1), January 1985.

[SD92] W. Brent Seales and Charles R. Dyer. Modeling the rim appearance. In *Proceedings of the 3rd International Conference on Computer Vision*, pages 698–701, 1992.

[SJ89] Thawach Sripradisvarakul and Ramesh Jain. Generating aspect graphs for curved objects. In *Proceedings of the IEEE Workshop on Interpretation of 3D Scenes*, pages 109–115. IEEE, 1989.

[Sny92] John M. Snyder. *Generative Modeling for Computer Graphics and CAD: Symbolic Shape Design Using Interval Analysis*. Academic Press, first edition, 1992.