

Hierarchical Overlapping Class Definitions

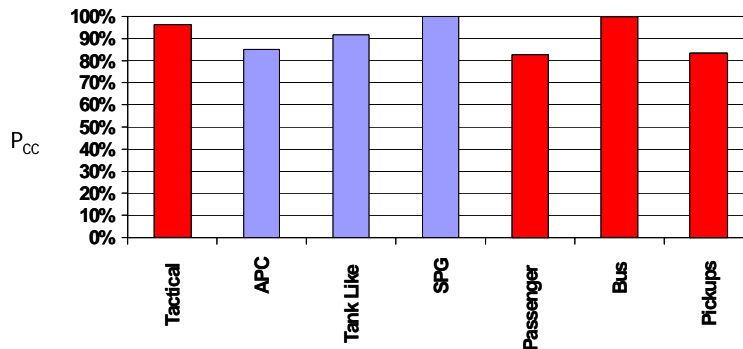
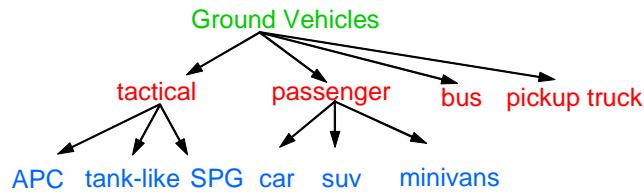


Figure 51. Initial classification performance on hierarchical overlapping classes is promising. Observed targets are assigned to classes based on the 3D geometric relations between class-specific features. While our initial results are promising several straight forward enhancements (discussed in the text) promise greatly improved performance.

3.1.6. References

1. ALPHATECH, Model-Based 3D LADAR ATR Development and Evaluation, Final Report, Contract #F33615-01-M-1947, January, 2002.
1. Grimson, W. E. L., Lozano-Perez, T., (1987). "Localizing overlapping parts by searching the interpretation tree." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:469-482.
2. Duda, R. O., Hart, P. E., (1973). *Pattern classification and scene analysis*. Wiley, New York.
3. Devijver, P. and Kittler, J., (1982). *Pattern Recognition: A Statistical Approach*, Englewood Cliffs, NJ: Prentice-Hall.

3.2. W.E.L. GRIMSON. OBJECT RECOGNITION BY COMPUTER: THE ROLE OF GEOMETRIC CONSTRAINTS, MIT PRESS, 1990. CUNY FAST RELATIONAL MATCHING

CUNY explored fast relational matching using relation extraction. The relational matching paradigm is based on the comparisons or matching of relational descriptions that are used to represent objects. Relational descriptions are firstly composed of basic

elements called primitives that are used to describe the basic parts, such as points, lines, regions, etc. that makeup an object. Descriptions that lack contextual information among the primitives are of very little use since they do not specify how the parts that compose an object fit together. Moreover they lack important information which is necessary to constrain the large space of possible mappings between the primitives. Thus, a description of the interrelationships among the primitives is necessary. This description of primitives and their interrelationships is called a *structural* or *relational* description [1]. The objective of a relational matching scheme is to compare two relational descriptions and find the "best correspondence" between them. Informally, the best correspondence is a mapping that maps the primitives of one object to another, and under this mapping the corresponding primitives and their relations show the best similarity in their attribute values, where the definition of similarity is application dependent. Due to the combinatorial nature of this problem, finding such a mapping is considered to be one of the hardest problems in computer vision, and is classified as an NP-hard problem. The aim of this report is to propose a new relational matching technique.

An object recognition or matching task usually consists of three stages. In the first stage a segmentation procedure must be executed to extract the basic parts that make up the object. Because we will be using 3D-car point clouds to demonstrate the effectiveness of our matching technique, the primitives in our case are chosen to be planer surfaces, which closely correspond to the basic parts that compose a car. Bellow we present a novel segmentation technique which we use to extract planer surfaces from 3D-car point clouds. Upon the completion of this stage a feature extraction phase must take place to compute the features that are used to describe the object parts, that is, to compute the relational description. In section 3 we show how the relational description of the segmented car surfaces is computed. The last stage in an object recognition task consists of the matching process, which is the comparison of one relation description with another to determine their level of correspondence or similarity. In section 4 we define what a match is, in section 5 we discuss the difficulties with the problem of relational matching, outline existing techniques, and finally in section 6 present our approach. The underlying idea is to execute hundreds or thousands of tree searches used to find the best correspondence between many pairs of relational descriptions in the training set. We do this offline, and even in a brute force manner. From all these executions we collect information that distinguishes the more promising paths in the tree search from the futile ones. This information is later used online to guide the search to the more promising paths, and to prune the futile paths, thus speeding up the objective which is to find the best correspondence as early as possible.

3.2.1. Segmentation

To convert the raw data into relational descriptions we first need to extract the primitives, a process generally known as segmentation. Once these primitives are extracted the acquisition of the relations between them is usually straight forward. Since the data we use are 3D-car point clouds, the basic elements the segmentation procedure should generate must closely correspond to the basic parts/surfaces that makeup a car, such as wheels, hoods, doors, etc. Each of these elements is later converted into planer surfaces, or a combination of planer surfaces, that make up our set of primitives.

3.2.2. Extracting the Primitives

We use a novel clustering routine which is based on spatial proximity thresholding. This routine repeatedly bi-partitions the data in the 3D point cloud (given in X,Y,Z coordinate form) and builds clusters with the property that their set of constituting points are located (distance wise) close to an oriented and bounded plane.

More specifically the idea is to sample three points from the data that will form the basis of the plane, and then construct a spatial proximity histogram of the distances of each point in the data from this plane. Then using the Kittler and Illingworth minimum error thresholding technique [2] we find a threshold that separates the data, based on the densities of the distances from the plane into two partitions/classes. The quality of these partitions or the goodness of the separation primarily depends upon the points that were sampled to create the plane. Sampling only one triple of points will generally not yield the best possible partitioning of the data. We therefore sample a certain number of triples, and select the triple that generates the best separation of the data, exhibited by the largest valley in the histogram. This process is repeatedly applied, splitting the data into smaller and smaller partitions until the data cannot be further partitioned in which case a cluster is assumed to be found. The points constituting this cluster should then closely correspond to one (or part) of the basic elements of car, such as the hood of a car. The next step is to convert these points into a planer surface. This is done by computing the three principle components (eigenvectors) of the covariance matrix of the points, and selecting the two that correspond to the two largest eigenvalues, i.e. the ones that account for the largest variance in two of the three dimensions. Once they are computed, the data is projected onto the subspace spanned by these two components. These projections then constitute a planer surface which closely corresponds to one (or part) of the basic elements. A more elaborate description of this segmentation procedure is presented in [1]. An illustration of the result of this process is provided in Figure 52 and Figure 53. Figure 52 demonstrates our initial results, and Figure 53 our current improved results obtained by learning what caused the profusion of surfaces and tuning the routine accordingly.

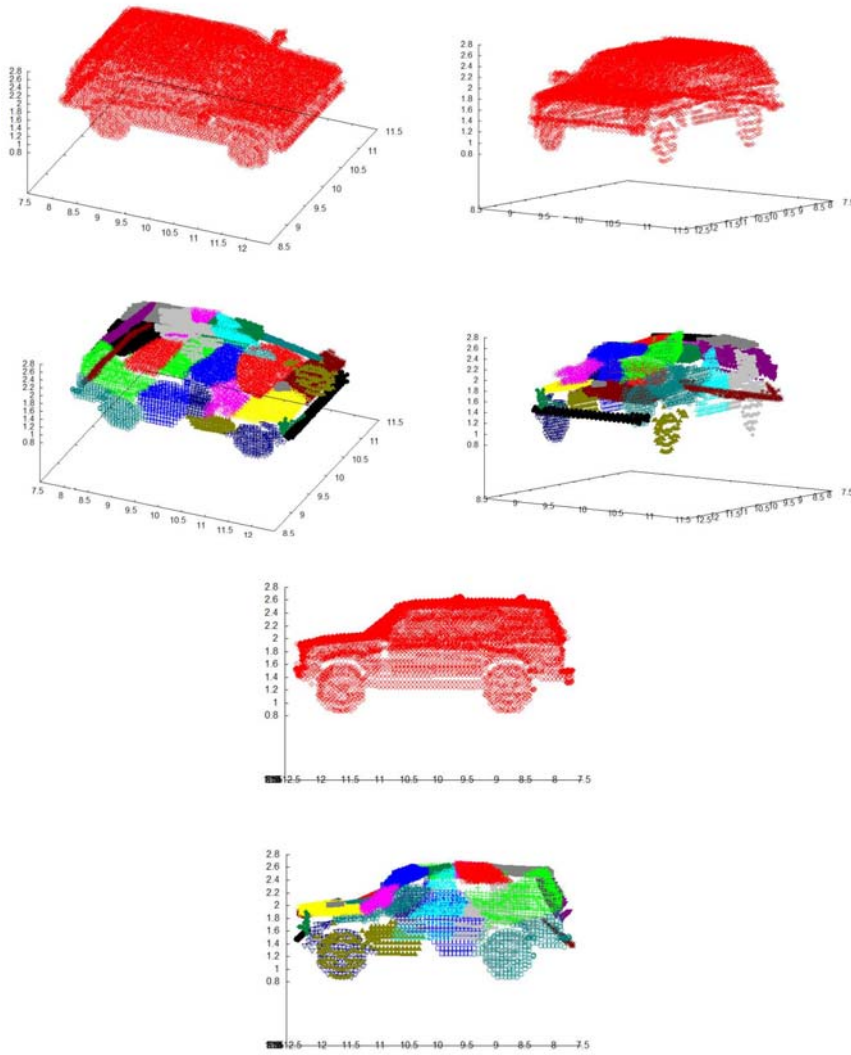


Figure 52: Three different views of a Ford-Explorer-91 3D point cloud and initial results of the segmentation procedure which extracted 26 surfaces.

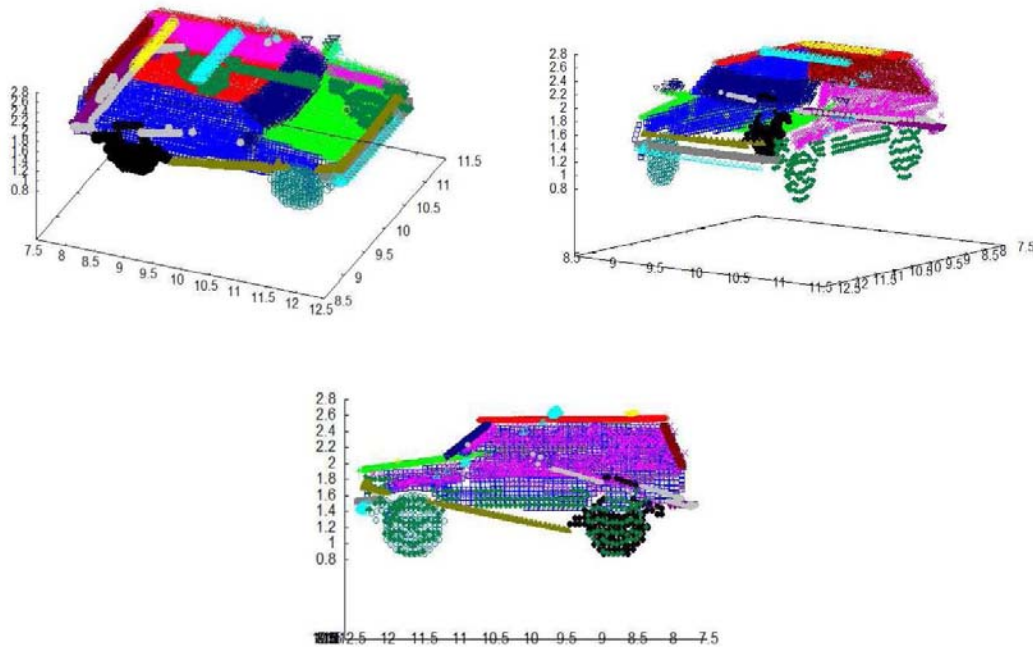


Figure 53. The same three views the Ford-Explorer-91 3D point cloud demonstrating the current improved results of the segmentation procedure which extracted only 15 surfaces.

3.2.3. Relation Extraction

After generating the surfaces of a 3D-car point cloud by the segmentation procedure, the next step is to extract the relational description of that car by an automated process. Before describing this process we give an overview of the relational model.

3.2.4. The Relational Model

The data of 3D objects can generally be described at different levels of abstraction. At the highest level the data descriptions that are being matched contain not only the description of the basic elements being matched, but also the interrelationships between them in form of relational descriptions. As mentioned before these descriptions are needed in order to constraint the large space of possible mappings between the basic elements. They intend to provide a rough characterization of the structure of the objects being matched by describing the interrelationships between the basic elements that comprise them. In our case they are used to describe how the planer surfaces are put together, and oriented with respect to one another to form the car. For example, describe whether two surfaces are connected, or whether they are parallel to each other.

3.2.5. The Relations

The basic elements of a relational description are called primitives. Generally, there are many types of primitives such as points, lines, or regions. We use only planer

ALPHATECH Inc.

surfaces as our primitives to describe cars. The reason, as mentioned before, is that these surfaces closely correspond to be the basic parts that make up a car, such as wheels, hoods, doors, etc.

Formally [1] a relational description D is defined by a set of primitives P and the interrelationships R among the primitives.

$$D=(P,R)$$

The primitives p_i in $P= \{p_1,p_2,\dots,p_N\}$ are usually described by a set of attribute value pairs, such as length, size, center of mass, etc. which can be numeric or symbolic in nature. Thus, each primitive p_i is a binary relation

$$p_i \subseteq A_p \times V_p$$

where A_p is a set of possible attributes and V_p is a set of possible values.

To describe surfaces we can use several different attributes such as area, perimeter, compactness, center of mass, etc. However we choose to use only the area attribute. The reason is that we want to keep the description as simple as possible, yet store enough information that can effectively be used by the matching process. Thus, $A_p = \{\text{area}\}$, $V_p = \mathbb{R}^+$ and each p_i in our relational description will be a pair such that

$$p_i (\{\text{area}\} \times \mathbb{R}^+)$$

For example the k th surface denoted by s_k will be represented as

$$s_k = \{(\text{area}, 4.545)\}$$

Next we concentrate on the interrelationships between the primitives, which are stored in R . Usually several different relations will be stored in R , for example the relation “angle” between two lines, or the relation “connectedness” between two regions. We use only one relation called “orientation” to describe the relationship between the surfaces extracted from our car point clouds.

Each relation is made of multiple events, and each event is described by a tuple of the primitives that constitute the event, and the attribute value pairs of the event [4]. For example, for our car objects there will be several pairs of surfaces that make an event of the “orientation” relation, and for each pair of surfaces there will be three attributes: angle, distance, and above/below specifying whether one surface is above or below the other, paired with their values. All events are collected into one set, denoted R_k for the k -th relation. R_k paired with the name of the relation denoted NR_k , is called a named relation and is denoted by PR_k . Thus,

$$R= \{PR_1,PR_2,\dots,PR_k\}$$

where $PR_k=(NR_k,R_k)$.

Since we have only one relation

$$R= \{PR_1\}$$

where $PR_1=(\text{orientation},R_1)$.

The set of relation tuples of relation k over M_k primitives and L_k attribute value pairs is a subset of all possible combinations, i.e.

$$R_k \subseteq P^{M_k} \times (A_{R_k} \times V_{R_k})^{L_k} .$$

In our case $M_1 = 2$, i.e. “orientation” is a binary relation over the set of surfaces, describing the orientation between two surfaces using three attribute value pairs, i.e. $L_1 = 3$. Thus

$$R_1 \subseteq (P \times P) \times (A_{R_1} \times V_{R_1})^3$$

where

$$A_{R_1} = \{ \text{angle, distance, above/below} \}$$

and

$$V_{R_1} = \{ V_{\text{angle}} \cup V_{\text{distance}} \cup V_{\text{above/below}} \} .$$

We now need to define the values that each attribute may take, i.e the set V_{R_1} , for our car objects. Since we are only interested to know whether two surfaces are parallel, perpendicular to each other, or neither, we quantize the values computed for the angles between surfaces. Thus “angle” will be a ternary symbolic attribute taking three values, $V_{\text{angle}} = \{1,2,3\}$, where 1-indicates that two surfaces are parallel, 2-indicating that the surfaces are perpendicular to each other, and 3-indicating neither. We let $V_{\text{distance}} = \mathbb{R}^+$, i.e. V_{distance} is regular distance measure. Finally, “above/below” will also be a ternary symbolic attribute taking values, $V_{\text{above/below}} = \{-1,1,0\}$, where 1-indicates the first surface is above the other, (-1)-indicating the first surface is below the other, and 0-indicating neither. To illustrate these concepts we give the following example expressing the orientation relation between several labeled surfaces s_1, s_2, \dots, s_k .

$$R = \{ \text{orientation, } \begin{array}{l} \{ s_1 s_2 \text{ (angle 1) (distance 0.034) (above/below -1) } \\ \{ s_3 s_5 \text{ (angle 2) (distance 0) (above/below 1) } \\ \vdots \\ \{ s_9 s_{12} \text{ (angle 3) (distance 1.42) (above/below 0) } \\ \{ s_{10} s_k \text{ (angle 1) (distance 0.722) (above/below -1) } \} \end{array} \}$$

3.3 Computing The Relational Description

To compute the relational description, we first need to compute the part that describes the surfaces, that is the area of each surface. We assume that most surfaces are rectangular or close to rectangular, an assumption that is confirmed by the segmentation results. Therefore the area of a surface can be approximated by taking the product of the lengths of each of its sides. The lengths can be found by computing the principle components of each surface, and projecting the points of each surface onto the basis spanned by the two most significant principle components (eigenvectors). Then the problem of computing the lengths is translated to the problem of finding the minimum and maximum of each component of the projections. More formally let v_1, v_2 be the two eigenvectors corresponding to the two most significant principle components. Let X be

ALPHATECH Inc.

the set of points that make up the surface, and x is a member of X be a point on the surface. Then $x = v_1^T x$ is the projection of x onto v_1 , and $y = v_2^T x$ is the projection of x onto v_2 . The minimum and maximum of the projections onto each of the principle components is defined by

$$\begin{aligned} x'_{\min} &= \min \{x' | x' = v_1^T x, x \in X\} \\ x'_{\max} &= \max \{x' | x' = v_1^T x, x \in X\} \\ y'_{\min} &= \min \{y' | y' = v_2^T x, x \in X\} \\ y'_{\max} &= \max \{y' | y' = v_2^T x, x \in X\} \end{aligned}$$

and the area of a surface is approximated by computing

$$\text{area} = (x'_{\max} - x'_{\min}) \times (y'_{\max} - y'_{\min})$$

Next we need to compute the part that describes the interrelationship between the surfaces, i.e., the tuples of the “orientation” relation. To do this we go through all possible pairs of surfaces (since we are interested in describing the relation between any pair of surface), and for each pair compute the three attributes.

1. To compute the quantized (dihedral) angle between two surfaces, we first compute the normal to each surface n_1, n_2 . This is done as before, by computing the three principle components (eigenvectors) of each surface, and selecting the least significant component, the one corresponding to the smallest eigenvalue. Then the angle θ between the surfaces is computed by $\theta = \cos^{-1}(n_1 \cdot n_2)$. This angle is then quantized by the following definition:

$$\text{angle} = \begin{cases} 1 & 0 \leq \theta \leq \epsilon_\theta \text{ or } (180 - \epsilon_\theta) \leq \theta \leq 180 \\ 2 & (90 - \epsilon_\theta) \leq \theta \leq (90 + \epsilon_\theta) \\ 3 & \text{if neither 1 or 2} \end{cases}$$

where ϵ_θ is a control parameter specifying error ranges for which we decide how to classify the angles.

2. The value of the distance attribute is defined as the shortest L_2 distance between any pair of points, where each point in the pair comes from a different surface. A special case is when the surfaces are adjacent, in which case the distance is defined to be zero. Two surfaces s_i, s_j are defined to be adjacent if $|d(s_{i_p}, s_{j_q})| \leq \epsilon_d \geq \epsilon_q$. That is, if there are at least ϵ_q pairs of points that are at most ϵ_d units separated from each other. ϵ_d is a control parameter specifying the largest distance for which we consider a pair of points to be adjacent. Likewise ϵ_q is a control parameter specifying the minimum amount of pairs for which we consider two surfaces as adjacent.

3. Surface s_i is defined to be above surface s_j (attribute value equal to 1), if s_i 's smallest z coordinate is larger than s_j 's largest z coordinate. Likewise surface s_i is defined to be below surface s_j (attribute value equal to -1), if s_i 's largest z coordinate is smaller than

s_j 's smallest z coordinate. If neither of the above holds, the attribute value will equal to 0.

Figure 54 illustrates the outcome of this process applied on several surfaces extracted from the front part of a Ford-Explorer-91 3D model. The results are summarized in a tabular form.

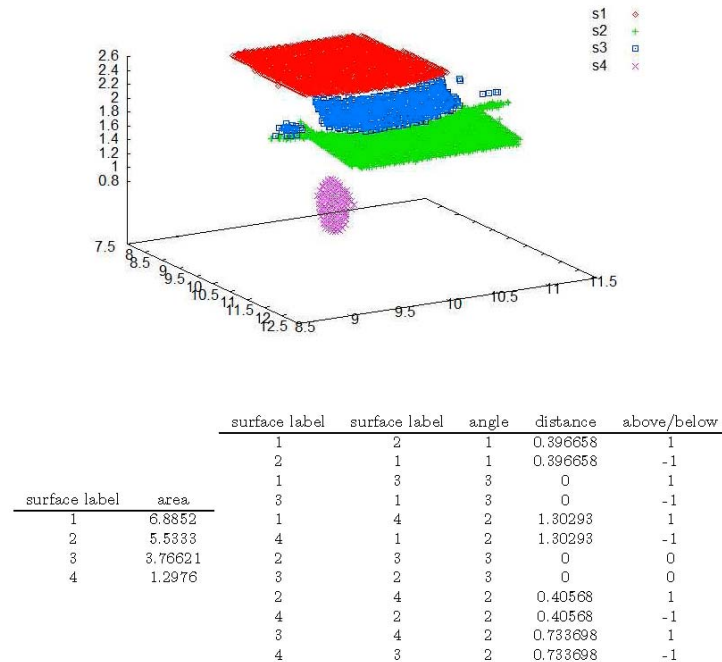


Figure 54. Several surfaces extracted from the front part of a Ford-Explorer-91 point cloud, and the relational description extracted by the relational description extraction process.

3.2.6. References

1. R. Haralick and R. Harpaz, Estimating high dimensional probability distributions, Tech. report, Department of Computer Science The Graduate Center City University of New York, 2003.
2. J. Kittler and J. Illingworth, Minimum error thresholding, Pattern Recogn. 19 (1986), no. 1, 41–47.
3. G. Shapiro and R.M. Haralick, Structural descriptions and inexact matching, IEEE Trans. Patt. Anal. Mach. Intell. 3 (1981), 514–519.
4. G. Vosselman, Relational matching, lecture notes in computer science, number 628, Springer Verlag, Berlin, 1992.