Constructing Spatial Representations of Variable Detail for Sketch Recognition

Andrew Lovett Morteza Dehghani Kenneth Forbus
{andrew-lovett, morteza, forbus}@northwestern.edu
Qualitative Reasoning Group, Northwestern University
2133 Sheridan Road, Evanston, IL 60201 USA

Abstract

We describe a system which constructs spatial representations of sketches drawn by users. These representations are currently being used as the input for a spatial reasoning system which learns classifiers for performing sketch recognition. The spatial reasoning system requires representations at a level of detail sparser than that which the representation constructor normally builds. Therefore, we describe how the representation constructor ranks the expressions in its output so that the number of expressions in the representation can be decreased with minimal loss of information. We evaluate the overall system, showing that it is able to learn and utilize classifiers for complex sketches even when the representation size is sharply diminished.

1. Introduction

One of the major problems in spatial reasoning systems is generating initial representations to be used as input to the systems. If a researcher is primarily interested in testing the system, such representations can be, and often are, generated by hand. However, if a system is to be truly useful, there needs to be a component which automatically constructs the representation, based directly on visual data. Depending on the task, this data may vary from movies, to photographs, to hand-drawn sketches, to simple line drawings. Obviously, the problems that arise in constructing a representation vary across these media. However, there are certain constraints that arise regardless of the type of image being processed.

There are two general constraints on any system which constructs spatial representations. First, the representations must be sufficiently detailed for accomplishing the task at hand. For example, in a recognition task, a representation of a coffee mug will probably need to incorporate the information that the image contains a large, upright cylinder and a second, curved cylinder (the handle) that connects to the larger cylinder at each of its ends. Note that the degree of detail needed will vary depending on the task. For example, if the task is to reason about heat transfer from a coffee mug, then it may also be necessary to represent the wavy lines indicating liquid in the cup, or even the lines floating above the cup that indicate it is giving off heat.

Unfortunately, the first constraint cannot be met by simply providing the maximum possible degree of detail. The representations must also be simple enough to avoid overwhelming the spatial reasoning system. This constraint can be seen as a simplified form of the classic framing problem in AI. A visual scene, even a simple one, has a huge number of objects, properties of objects, and relations between objects that could potentially be attended to. However, in most cases, a small subset of this information is all that is necessary for performing a task. The rest of the information will at best distract the spatial reasoning system, and at worst cause it to fail. It is the job of the representation constructor to determine which information is necessary and which information can be left out.

Given the tradeoff between the sufficient detail constraint and the simplicity constraint, we believe one goal in making representation constructors should be to design systems that are flexible in the degree of detail they include in their representations. This will allow them to be used in varying tasks which demand different degrees of detail, as well as allowing them to be used with varying reasoning systems that can handle different degrees of detail.

One general way in which a system can vary the detail in its output representations is to compute a representation to the fullest possible detail but assign a priority ranking to each expression in the representation. For example, consider the case of a simple rectangle. The fact that this is a closed shape with four sides is particularly important, and it should probably be included in any representation of the shape. The fact that two of the sides are longer than the other two is less important, and it might be left out of the representation in cases where a sparser representation is needed. Note that a priority ranking might be task-general, meaning that certain expressions are always considered more important than other expressions, or it might be task-specific, meaning that certain expressions are considered more important for certain tasks.

In this paper, we will be discussing our approach to constructing and using spatial representations. Our work is based on reasoning about hand-drawn sketches. Sketches are, of course, much easier to process than photographic images. Because they consist only of the edges of objects, they allow us to skip edge detection, one of the more difficult tasks in visual processing. However, processing sketches is far from trivial. Hand-drawn sketches lack straight lines or clear corners between lines. They tend to contain a large amount of noise, depending on how carefully they were drawn. Thus, the problem of sketch perception involves taking a set of noisy lines and representing them in a clean format useful for higher-level spatial reasoning.

In the following sections, we will describe our system in greater detail. We will begin by discussing open-domain sketch perception, the task for which our system is being used. We will then cover the spatial reasoning system that performs this task. We will describe how the representation constructor builds its representation. We will talk about how the constructor ranks expressions in the representation so that the degree of detail can be controlled. Finally, we will give experimental evidence showing that this is an effective means of controlling the representation size.

2. The Sketch Recognition Task

Our interest in sketching stems from working with sKEA, the *sketching Knowledge Entry Associate* (Forbus et al. 2004). sKEA is an open-domain sketch understanding system. It builds representations of sketches drawn by users and incorporates these representations in various reasoning tasks, such as comparing sketches and looking for differences. sKEA is able to build a representation of a sketch by any user without any prior expectations regarding what the user will be sketching for one key reason: it does not perform recognition. Rather, it relies on the user to divide a sketch up into units, called glyphs, and to tell it what object each unit represents. Labels for each glyph can be picked from a knowledge base containing over 25,000 categories. sKEA then computes a set of spatial relations between the glyphs found in a sketch. It combines this information with its semantic knowledge about each glyph's label to build a representation of the sketch that can be used in spatial reasoning tasks.

One of sKEA's drawbacks is the requirement that users manually segment their sketches into glyphs and label each glyph. This limits sKEA's user base to those who understand the concept of a glyph and are capable of identifying the most appropriate category for each glyph. Even among experienced users, the requirement that every glyph be labeled can become onerous, particularly when a user is setting up an experiment that requires drawing many instances of the same object.

sKEA's limitations led us to ask whether it would be possible to add a recognition component to sKEA without sacrificing its domain-independence. Unfortunately, open-domain sketch recognition is a significant problem. Even a classifier designed to identify sketches of a single object, such as a book, must deal with significant variability in the way that object can be sketched. As the set of possible objects increases, the potential for confusion between the objects also increases. There may be greater similarity between a sketch of a book and a sketch of a box, or a computer screen, than between two sketches of books. In an unconstrained drawing task, there is no way to predict what object a user will sketch, so the set of possible objects must include every conceivable object.

For these reasons, most sketch recognition systems are constrained to a narrow domain containing a small set of possible objects (e.g., circuit diagrams: Liwicki and Knipping 2005; simple symbols: Anderson, Bailey, and Skubic 2004; architectural objects: Park and Kwon, 2003). By limiting the domain, the creators of these systems can identify every object the systems will be required to identify. They can then either hand-code classifiers for each object or train the classifiers on a large body of data (700 images for Liwicki and Knipping 2005). Even systems designed to work in multiple domains require that the classifiers for each domain be hand-coded (Alvarado et al. 2002). The systems created in this way are powerful when used within their domains, but they fail when given any input outside of those domains.

We believe the key to sketch recognition in the absence of domain expectations is efficient, online learning. Our goal is to build a system that constructs classifiers for different objects on-the-fly during a sketching session. The first time a user sketches an object, the system will have no way of knowing what the object is. However, after the user labels the object, the system will remember the representation for that object. In the future, if the system sees a similar object, it will guess that the new object is a new instance of the old object's category. If the system is wrong, the user still has the option of using sKEA's labeling function to correct it.

As we have stated above, different sketches of an object may vary widely. The best way to learn to recognize an object is by considering multiple examples of the object and identifying the features most common to those examples. However, we do not wish to build a system that requires some minimum number of examples before it can learn to classify an object (and certainly not a number in the hundreds). If the system is to be useful within a single sketching session, it should build a usable classifier based on the very first example of an object it sees. It should then use each additional example of the object to fine-tune the classifier. Thus, this task requires a spatial reasoning system capable of efficient, incremental learning of classifiers for sketched objects.

3. Comparison and Generalization

In this section, we will describe how our system builds classifiers for each encountered object and uses these classifiers to categorize new sketches. Note that these processes work independent of the component which constructs the representation for each sketch. They constrain that component in terms of the size of the representation they can handle and the format in which the representation must be encoded, but they are otherwise agnostic as to the content of the sketch representations. This allows us to study the problem of building spatial representations independent of the problem of sketch recognition. The component which constructs the representations will be described in the following section.

Our system classifies sketches through a comparison process. Once a representation of a sketched object has been produced, the system compares it to the representations of previously encountered object. If a sketch's representation is sufficiently similar to another object, the system concludes that the sketch is another instance of that object. In cases where multiple instances of an object have been encountered, a generalization of the representations of all those instances is computed. This generalization can then be compared to the new sketch's representation in the same way.

We perform comparison using SME, the Structure Mapping Engine (Falkenhainer, Forbus, and Gentner 1989). SME is a computational model of similarity and analogy. It is based on Gentner's (1983) structure-mapping theory, which states that humans draw analogies between two cases by aligning their common structure. SME works on structured representations, consisting of entities, attributes of entities, and relations between entities or between other relations. Given two cases, a base and a target, it begins by finding all possible correspondences between entities, attributes, and relations in the two cases. It combines consistent correspondences to find a global mapping between the cases with the maximum systematicity. Systematicity is the depth of the aligned structure. Correspondences between higher-order relations, i.e., relations between relations, are deeper than and thus preferred to simple relations between entities.

We perform generalization using SEQL (Kuehne, Forbus, and Gentner 2000; Halstead and Forbus 2005). SEQL models generalization as a process of progressive alignment. Each known category is represented by a generalization. When a category is first created, this generalization is simply the representation of the first known instance in the category. When a new instance of a category is found, its representation is aligned with the generalization using SME. Expressions in the generalization that have a corresponding expression in the new instance are strengthened. Expressions in the generalization that do not have a corresponding expression are weakened. This is done by simply associating a probability value with each expression in the generalization. This probability value tells the percentage of instances of the category containing an expression which aligns with that particular expression. Note that when new instances are added to a category, any expressions in their representation that fail to align with anything in the generalization can be added to the generalization, with an associated probability value of 1 / (the total number instances).

4. Building Spatial Representations

The representation constructor begins with a set of lines drawn in sKEA. Its processes can be divided into two steps. In the first step, it segments the rough sketch into a set of edges. In the second step, it produces a qualitative representation of the edges. These two steps are described below. For a more detailed description, see Lovett, Dehghani, and Forbus (2006).

Perceptual Elements

Our system utilizes a bottom-up approach for sketch perception. Given a set of polylines, lists of points representing lines drawn by the user, it begins by segmenting them into atomic units called segments. Each segment is a short straight line. These segments can then be grouped together to form progressively larger, more complex perceptual elements.

Before segments can be grouped together, their endpoints must be classified. The endpoints, i.e., the points at the beginning and end of each segment, can be classified as connections, corners, or terminations. A connection is a point at which two segments along the same edge meet. A corner is a point at which two segments along different edges meet. For example, in the case of a square, all the segments along each edge are joined by connections. The last segment of one edge and the first segment of an adjacent edge are joined by a corner. Finally, terminations are endpoints of a segment that are not joined to any other segment. Closed shapes such as squares contain no segments with terminations. On the other hand, a single straight line will have terminations at each of its ends.

Once endpoints have been classified, grouping segments is a trivial matter. Chains of segments joined by connections are grouped together to form edges. If two segments are joined by corners, then the edges those segments are each grouped into are also joined by corners. Thus, the grouping process creates a set of edges with various corners connecting them. These corners are used to group the edges into *connected edge groups*. A connected edge group is a maximal list of sequentially connected edges. Connected edge groups in which the first and last edge are the same, i.e., edge groups that form a closed shape, are *cycles*, while connected edge groups that begin and end with a termination are *paths*. Once the edges, connections between edges, and connected edge groups have been calculated, the system can begin building the representation.

Qualitative Representation

Our system builds qualitative representations of sketches, meaning it avoids using absolute values. This is vitally important for any task involving comparisons between sketches. Absolute values, such as the length of a particular edge, may vary greatly across different sketches, depending, for example, on the scale at which each sketch is drawn, but qualitative values like the relative lengths of the edges are much more likely to remain constant.

One of our primary tasks in building this system was to design an appropriate qualitative vocabulary for representing the edges, attributes of edges, and relationships between edges in a sketch. This vocabulary needed to be general enough that it could be used to represent any sketch, regardless of the type of object being drawn. Fortunately, in building our vocabulary, we were able to draw on the work done by previous researchers who had shared an interest in qualitative representations of sketches or line drawings. In particular, we looked at the qualitative vocabularies used by Ferguson and Forbus (1999), Museros and Escrig (2004), and Veselova and Davis (2004). Many of the terms in our vocabulary were used by one or more of those researchers. However, those researchers were primarily concerned with representing relatively simple sketches. We found that in order to properly represent sketches of greater complexity, and particularly in order to represent sketches of three-dimension objects, it was necessary to add a few terms to the vocabulary.

The terms in the vocabulary can be split into three types: attributes, pairwise relations, and anchoring relations. Attributes are simple descriptors for individual edges. Each edge is classified as one of three possible types: straight, curved, or elliptical. Straight edges can also be classified as horizontal or vertical, if they align with the x- or y-axes.

Pairwise relations describe basic relationships between pairs of edges, including relative length, relative orientation, and relative location. Because relationships exist between every pair of edges in the sketch, the number of pairwise relations can grow at an alarming rate. Furthermore, many of the relationships are irrelevant or redundant. Therefore, we follow Veselova and Davis (2004) in only asserting pairwise relations between adjacent edges, i.e., edges that do not have another edge between them.

In additional to the pairwise relations described above, there are also pairwise relations describing connections between edges. Edges may be connected because their endpoints are joined as corners, or they may be connected because they intersect each other. In cases where a corner between edges is located along a cycle, that corner can be further classified as convex or concave.

The final set of terms are the anchoring relations. They are called anchoring relations because, unlike the simpler attributes and pairwise relations, they describe more than two edges and contain greater structural depth. Because of SME's systematicity bias, this greater amount of structure causes them to be matched first when two representations are being compared. Thus, they play the role of anchoring the SME mappings between representations.

There are two types of anchoring relations. The first type describe simple closed shapes. Anchoring relations are asserted for each thee-sided or four-sided closed shape in the sketch. The second type describe junctions between edges. Junctions are asserted for points at which three edges join. These junctions are classified as arrow junctions, fork junctions, tee junctions, or other junctions. Positional relations between junctions are also asserted, to provide additional structure for anchoring the match.

5. Controlling the Representation Size

One early discovery we made regarding our representation constructor was that the representations it built were simply too large. A simple sketch, such as a cylinder might be represented by as few as 50 expressions, but larger, more complicated sketches, contained as many as 600 expressions in their representations. Representations of this size were simply too much for SME and SEQL to handle. Our representation constructor was committing a fatal violation of the simplicity constraint.

We chose to address this problem by allowing the spatial reasoning system to restrict the representation to an arbitrary number of facts. That is, the system could choose to accept only the first N expressions in the representation, and to ignore the remaining ones. Of course, this is a dangerous strategy if the ordering of the facts in the representation is unknown. If N expressions were taken from the representation at random, it is possible that one or two vitally important expressions would be left out. Thus, it became necessary to assign a priority ranking to expressions in the representation, so that instead the N most important expressions could be chosen.

We rank expressions by two factors: the predicate of the expression and the edges involved in the expression. Predicates are divided into three groups: defining attributes, anchoring relations, and others. These groups mostly align with the attributes, anchoring relations, and pairwise relations described above. Defining attributes, which tell what type of edge an entity represents, receive the highest ranking because any time an entity is included in a representation, the representation ought to describe what the entity is. Anchoring relations receive the second-highest priority because of the important role they play in anchoring a match. Other terms receive the lowest priority. These include pairwise relations and the attributes stating whether a straight edge is vertical or horizontal.

Edges are grouped into the following categories: external, external-adjacent, external-connected, and internal. An external edge is an edge which touches the outer bounds of the entire sketch. These edges are deemed the most important because they can provide the most information for recovering the overall shape of the sketch. External-adjacent edges are edges that connect to external edges. These edges can also provide useful information about the sketch's shape. External-connected edges are edges that are part of a connected edge group that contains at least one external edge. The internal edges do not connect to outer edges. They can provide information about the details of a sketched object, but they usually do not help to describe its shape.

Expressions are ranked first by predicate, and then by edges. That is, all expressions with defining attributes and anchoring relations are ranked above all expressions with other predicates. Among the defining attributes and anchoring relations, and then among the other predicates, expressions are ranked by the lowest-ranked edge. Thus, a pairwise expression between an edge-adjacent edge and an edge-connected edge would be ranked at the same level as a pairwise expression between two edge-adjacent edges. Expressions of the same rank are ordered randomly, with the exception of the following somewhat arbitrary ranking of the other predicates: corners and connections, relative orientation and length, and relative position.

The rankings as they have been described above were chosen to be task- and domain-general. That is, we believe that anchoring relations and external edges should be important for representing any sketch. However, we made one additional decision that we believe would not generalize to all other tasks, or even to all other stimuli within the recognition task: we do not assert relations about internal edges. This means that, for example, in a sketch containing a larger square with two smaller circles inside it, the representation would say that were two ellipses inside the square, but it would not give their positions relative to each other or to the sides of the square. Thus far, we have found that this heuristic has simplified the problem of sketch recognition, but again, we do not claim that it would generalize. Fortunately, it could be easily removed if the need arose.

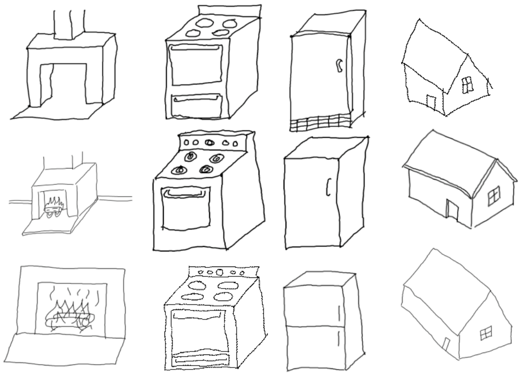

Figure 1. Sketches drawn by subjects

6. Experiment

We evaluated our sketch recognition system using sketches of eight everyday objects: a house, a brick, an over, a cup, a refrigerator, a bucket, a fireplace, and a cylinder. We selected an example of each of these objects from *Sun up to Sun Down* (Buckley 1979), a book which uses simple drawings to illustrate physical processes such as heat transfer. 10 subjects were asked to sketch the eight objects, using the example illustrations as guides. These examples were provided so that all subjects would draw the same type of object from the same perspective. However, subjects were instructed to only include those parts of the illustration that they believed were necessary to indicate what the object was. As we had hoped, there was significant cross-subject variation in the way each object was drawn, but there tended to be core features common to most or all sketches of an object (see Figure 1 for examples). Thus, we decided the task we had created for our system was difficult, but not impossible. One subject's sketches were thrown out because the subject failed to follow directions. The remaining 9 subject's 72 total sketches made up our corpus of training and test data.

We tested the system by running a series of trial runs. In a single trial run, the 9 subjects' sketches were randomly divided into a training set, containing 5 sketches of each object, and a test set, containing the remaining 4 sketches of the objects. Generalizations for each object were built from the representations of the 5 sketches in the training set. The sketches in the test set were then evaluated by comparing them to each generalization and picking the most similar generalization.

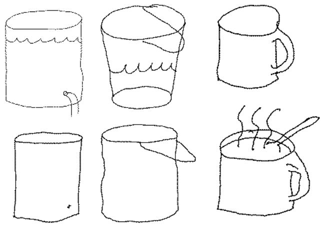

Figure 2. Cylinders, buckets, and cups drawn by subjects

Preliminary tests indicated that three objects were commonly confused: the cup, the bucket, and the cylinder. This is hardly surprising, as these objects are quite similar, and there were sometimes more differences between sketches of the same object than there were between sketches of the different objects (see Figure 2). We therefore chose to evaluate our system using two criteria. Under the strong criterion, only a classification into the correct object category was counted as correct. Under the weak criterion, a classification into any of the three cylindrical categories was counted as correct as long as the object belonged in any of the three categories.

One of our goals in evaluating the system was to determine how limiting the number of expressions in the representation of a sketch would affect results. Recall that when the entire representations were allowed, some representations grew to as large as 600 expressions. Our goal was to cut the representation size to about a quarter of that. We therefore tried four max representation sizes: 100, 150, 175, 225.

Results

For each representation size, we ran 80 trial runs and averaged the results. We measured accuracy according to both the strong and weak criteria. Note that chance performance on the strong criterion would be 12.5%, while chance performance on the weak criterion would be 16.7%. We found that performance far exceeded chance with all representation sizes (see Figure 3). The greatest results were achieved with 175 expressions in the representations. With this representation size, the strong criterion was met 77% of the time, while the weak criterion was met 93.5% of the time. We found that the increase in performance between a representation size of 100 and 150 was statistically significant (p < .05). However, there were no significant differences between the results with sizes of 150, 175, and 225.

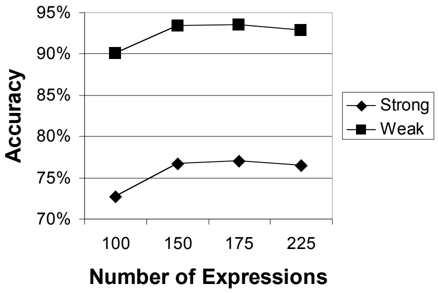

Figure 3. Results with varying representation sizes

7. Conclusion

Our system was effective at classifying sketches into the eight object categories. Performance was far above chance, and performance on the weak criterion was near-perfect. Furthermore, the best results were achieved with a max representation size of 175 expressions, close to a quarter of the size of the full size of the most complex sketches. Increasing the representation size from 175 to 225 failed to improve performance. In fact, it resulted in a slight dip in performance, although the change was not statistically significant. Therefore, we think it likely that a max representation size of around 175 is optimal, given our current representation scheme. The optimal size might change if a different set of sketches were used, but given the wide range in the complexity of these sketches, with the full representation size varying from 50 to 600, we think it likely that a limit of 175 is generally appropriate for the task of sketch recognition.

While we have demonstrated that our system for constructing representations of sketches is effective for the sketch recognition task, it remains to be seen whether the system is flexible enough to deal with other spatial reasoning tasks. We are particularly interested in determining whether the ranking of expressions used here can be used in other tasks which require a greater or lesser level of detail. Examples of other spatial reasoning tasks being studied in our group include geometric analogy and spatial preposition use. In the future, we hope to show that a single sketch representation scheme can be used, in coordination with different spatial reasoning systems, to solve a variety of such tasks.

Acknowledgments

This research was supported by a grant from the Computer Science Division of the Office of Naval Research.

References

Alvarado, C., Oltmans, M., and Davis, R. 2002. A Framework for Multi-Domain Sketch Recognition. In *2002 AAAI Spring Symposium on Sketch Understanding*. Palo Alto, CA.

Anderson, D., Bailey, C., and Skubic, M. 2004. Hidden Markov Model Symbol Recognition for Sketch-Based Interfaces. In *Making Pen-Based Interaction Intelligent and Natural*, 15-21. Arlington, VA: AAAI Press.

Buckley, S. 1979. *Sun Up to Sun Down*. McGraw Hill: New York.

Falkenhainer, B., Forbus, K. and Gentner, D. 1989. The Structure-Mapping Engine: Algorithms and Examples. *Artificial Intelligence* 41: 1-63.

Ferguson, R. W., and Forbus, K. D. 1999. GeoRep: A Flexible Tool for Spatial Representations of Line Drawings. In *Proceedings of the 13th International Workshop on Qualitative Reasoning (QR'99)*, 84-91. Loch Awe, Scotland.

Forbus, K., Lockwood, K., Klenk, M., Tomai, E., and Usher, J. 2004. Open-Domain Sketch Understanding: The nuSketch Approach. In *AAAI Fall Symposium on Making Pen-based Interaction Intelligent and Natural*, 58-63. Washington, DC: AAAI Press.

Gentner, D. 1983. Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive Science 7*: 155-170.

Halstead, D., and Forbus, K. 2005. Transforming between Propositions and Features: Bridging the Gap. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*. Pittsburgh, PA: AAAI Press.

Kuehne, S., Forbus, K., Gentner, D., and Quinn, B. 2000. SEQL: Category Learning as Progressive Abstraction Using Structure Mapping. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, 770-775. Philadelphia, PA.

Liwicki, M., and Knipping, L. 2005. Recognizing and Simulating Sketched Logic Circuits. In *Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, 588 – 594. Melbourne, Australia: LNCS.

Lovett, A., Dehghani, M., and Forbus, K. 2006. Efficient Learning of Qualitative Descriptions for Sketch Recognition. In *Proceedings of the 20th International Workshop on Qualitative Reasoning (QR'06)*. Hanover, NH.

Museros, L., & Escrig, M. T. 2004. A Qualitative Theory for Shape Representations and Matching. In *Proceedings of the 18th International Workshop on Qualitative Reasoning (QR'04)*. Evanston, IL.

Park, J., and Kwon, Y-B. 2003. Main Wall Recognition of Architectural Drawings Using Dimension Exension Line. In *Proceedings of the Fifth IAPR International Workshop on Graphics Recognition (GREC'03)*, 116-127. Barcelona, Spain: Springer.

Veselova, O., and Davis, R. 2004. Perceptually Based Learning of Shape Descriptions. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI'04)*, 482-487. San Jose, CA: AAAI Press.