Edge-cycles: A qualitative sketch representation to support recognition

Matthew D. McLure, Scott E. Friedman, Andrew Lovett, and Kenneth D. Forbus
Qualitative Reasoning Group, Northwestern University
2133 Sheridan Rd., Evanston, IL, 60201, USA
{mclure@u., friedman@, andrew-lovett@, forbus@}northwestern.edu

Abstract

Qualitative representations can play an important role in sketch understanding, by providing stable relational descriptions that support learning for recognition. A common approach is to represent a sketch of an object as a set of edges and relations between edges. However, hand-drawn sketches are generally noisy, with unintended gaps, jitter, and other glitches that cause artifacts in edge representations. We describe a new higher-level representation, *edge-cycles*, that is more stable than edge-based representations, and demonstrate that it provides significantly better results in learning to classify hand-drawn sketches of everyday objects.

1 Introduction

Software that works with people via sketching could have a revolutionary impact on human-computer interaction. Consequently, understanding hand-drawn sketches is an important problem. Hand-drawn sketches are noisy: Lines intended to be straight often aren't, lines jitter, and gaps often occur in what was intended to be a continuous outline. Qualitative representations provide a natural approach to sketch understanding, because they can abstract out the quantitative variations that constitute noise, and recover more of the intended shapes. Previous work on CogSketch [Forbus *et al*., 2011; Lovett *et al*., 2008] has shown that qualitative representations of digital ink in terms of properties of shapes, groups, and edges can provide useful results in a variety of tasks. However, as shapes get more complex, edge-level representations tend to bog down. For example, the hand-drawn sketches of everyday examples in [Lovett *et al*., 2006] initially contained as many as 86 edges, with many more relationships connecting those facts, making matching and generalization quite difficult. This paper defines a new higher-level representation, *edge-cycles*, that provides a higher level of qualitative abstraction. An edge-cycle is a sequence of edges connected end-to-end, whose last edge connects back to its first edge. Edge-cycles are computed from the edge-level representation, but because they are more abstract, they should provide a more stable level of representation. We show that edge-cycles significantly out-perform edge-level representations on the object recognition task of [Lovett *et al*., 2006].

We begin by reviewing CogSketch, our sketch understanding system, and the analogical processing techniques being used for learning and classification. Then we describe the edge-cycle representation, followed by an experiment where we compare edge-cycle and edge-level representations. We close with a description of other related work and future work.

2 Background

We begin by summarizing CogSketch and the analogical processing models used for learning and classification.

2.1 CogSketch

CogSketch [Forbus *et al.*, 2011] is an open-domain sketch understanding system. It computes a variety of qualitative spatial representations on digital ink, including qualitative topological relationships [Cohn *et al*., 1997], positional relationships (e.g. `above`, `rightOf`), symmetry (using MAGI [Ferguson, 1994]), and relative sizes. Its representations are hierarchical, with the default *shape* level describing pieces of the ink that the user labeled as an entity. (In the sketches here, the entire sketch is always one entity.) The *edge-level* representation is the more detailed level of representation that decomposes a shape into its edges and encodes their attributes (e.g. straight, curved, or elliptical) and relationships between them (e.g. corners, whether a corner is concave or convex). The highest level representation is the *group* level, where CogSketch groups objects together based on proximity and similarity. By default, CogSketch computes only shape-level representations, with edge or group level representations being computed on demand by systems that use CogSketch. In the experiments reported here, the edge-level representations are always computed.

2.2 Models of analogical processes

We use SAGE, a model of analogical generalization, for learning and MAC/FAC, a model of analogical retrieval, for classification. These both in turn use SME, a model of analogical matching, so we start with it.

The Structure-Mapping Engine

SME, the Structure-Mapping Engine [Falkenhainer *et al*., 1989] is a domain-general computational model of analogy and similarity, based on Gentner's [1983] structure-mapping theory of analogy. Its inputs are two cases, the *base* and *target*, consisting of structured, relational representations. SME produces one or more *mappings* between the base and the target. Each mapping contains: (1) *correspondences* that match relations and entities in the base with relations and entities in the target; (2) a numerical *similarity score*, provided the correspondences; and (3) *candidate inferences* that assert what might hold in the target, provided the correspondences with the base. Two relevant factors to note about SME for this paper: (1) here we normalize the usual similarity score by dividing it by the self-similarity score, i.e. the maximum score attained by matching either the base or target to itself. This ensures that the similarity score is always between zero and one. (2) Mappings are constrained to be 1:1, i.e. each item in the base or target can map to at most one other item in the other. There is strong psychological evidence for this constraint, and it simplifies the matching process considerably. However, it does put more pressure on encoding systems to be accurate in their grouping operations, as shown below.

MAC/FAC

MAC/FAC [Forbus *et al*., 1995] is a domain-general computational model of similarity-based retrieval. Its inputs are a *case library*, again consisting of structured, relational representations, and a *probe*, which is a case. MAC/FAC retrieves one or more cases from the case library that are similar to the probe. It uses a two-stage filtering process. The first stage is coarse, using a vector representation automatically computed from the structured representation to estimate similarity between the probe and the contents of the case library by computing dot products in parallel. It returns the best, plus up to two others, if sufficiently close. The second stage uses SME to compare the structured representation of the probe with the structured representations of the outputs of the first stage. Again, it returns the best, or up to three if the others are very close. The mappings it computes are available for subsequent processing. Here, MAC/FAC is used for classification: During testing, a sample is used as the probe, and analogical retrieval used to find the most similar generalization, which constitutes its classification, using the union of all generalization contexts (see next section) as the case library.

Category Learning with SAGE

Our category learning approach uses SAGE, a computational model of analogical generalization[1]. SAGE is designed for learning multiple categories using positive examples. Each category (e.g. Brick) has an associated *generalization context*, which contains a set of previously seen examples of the category, and a set of *generalizations*, which are automatically created by SAGE. SAGE learns incrementally. Given a new example, SAGE uses MAC/FAC to retrieve similar prior examples and/or generalizations, using the generalization context as its case library. If the similarity between the best retrieved item and the example is higher than a *similarity threshold* S, then the retrieved item and the example are merged. If the retrieved item is itself an example, a new generalization is formed. If the retrieved item is a generalization, the example is assimilated into it. In both cases, the assimilation process consists of calculating probabilities for each statement in the updated (or new) generalization, based on frequency of occurrence of corresponding facts in the examples that have gone into it. Entities that are not identical are replaced by arbitrary new entities, not logical variables. As examples accumulate, incidental properties fade away (by becoming less probable), whereas characteristic properties are strengthened (by remaining more probable). If no sufficiently similar item is retrieved for the example, the example itself is stored in the generalization context corresponding to its label.

3 Representation

There is psychological evidence that while people's representations of space are computed from the bottom up, they are attended to from the top down during problem-solving [Hochstein and Ahissar, 2002]. Higher-level representations are typically better places to start because they are sparser, providing lower working-memory loads, and reducing the number of distractors in matching. Consequently we want our representations to be as sparse as possible, while retaining features that are important for categorization. Edge-cycles, we argue, provide a useful and cognitively plausible representation for learning and recognition.

Constructing representations at both the edge level and edge-cycle level depends on the segmentation of ink into edges. We begin by describing our segmentation algorithm. We then describe relevant aspects of the Lovett *et al.* [2006] edge-level representation in more detail, since we use this as a baseline for our evaluation in section 4. We follow with a description of the three types of edge-cycles used in the present encoding approach.

3.1 Segmentation of ink into edges

The algorithm used by CogSketch to segment ink into edges has changed since 2006; what is described in this section is the present segmentation algorithm, since this is what is used to compute the edge-cycle representation[2]. For more details on the segmentation algorithm used in 2006, see [Lovett *et al*., 2006].

[1] SAGE is the descendant of SEQL [Kuehne et al 2000], extended with probabilities [Halstead and Forbus, 2005].

[2] The present segmentation algorithm computes significantly more edges than the baseline algorithm. Consequently, the segmentation algorithm alone is not responsible for making the present representation sparser than the baseline.

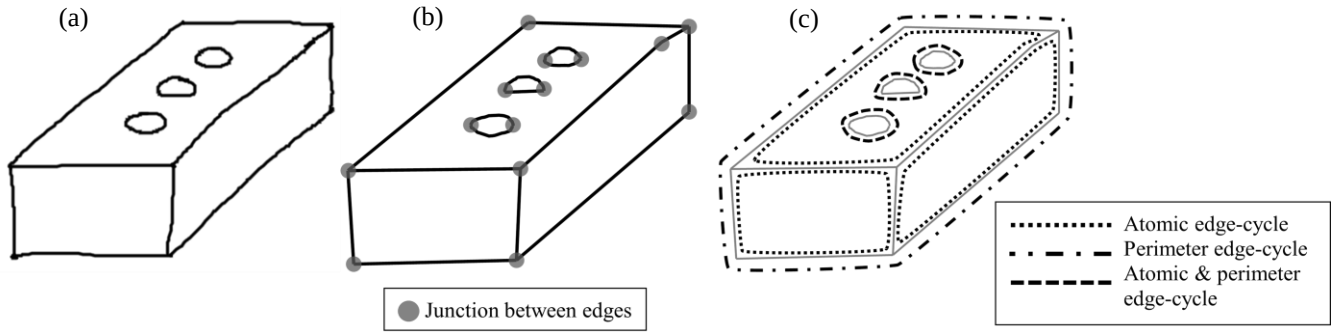

Figure 1: A sketch of a brick drawn by one of the participants (*a*), and a visualization of each encoding. The edge-level encoding (*b*) has 16 entities (all edges), and ends up generating 114 facts. The right-hand edge of the top surface is segmented into two entities due to noise in the curvature. The edge-cycle encoding (*c*) has 11 entities: 4 edge-connected objects, 3 atomic edge-cycles that are also perimeter edge-cycles, 3 atomic edge-cycles that are not perimeters, and 1 perimeter edge-cycle that is not atomic. It generates only 46 facts. The edge-cycle representation is unaffected by the curvature-related segmentation error.

CogSketch captures digital ink as sets of polylines, with each point represented by a 2D coordinate and timestamp. CogSketch resamples the polylines to factor out velocity artifacts and describe them according to changes in curvature. Polylines are initially segmented at intersection points, and then further decomposed based on discontinuities in curvature [Lovett *et al*., 2009]. This segmentation process results in edges, whose end points can be connected to each other via junctions.

3.2 Edge-level (baseline) representation

In this section, we describe the edge-level representation produced by CogSketch in Lovett *et al.* [2006], since this encoding produced the representations that we use as a baseline for our evaluation in section 4.

Of the edges created during segmentation, each is classified either as a `straightEdge` or a `curvedEdge`, with curved edges also sometimes qualifying as an `ellipseEdge`. Additional attributes may apply to edges of each type (e.g. `vertical` and `horizontal`). Pairwise relations are computed between neighboring edges to capture relative position, length and orientation (e.g. `above`, `longerThan` and `perpendicular`).

Junctions are represented by (1) relations that describe the connections between edges and (2) attributes that make qualitative distinctions between their shapes (e.g. `arrowJunction`, `forkJunction`, and `tJunction`). Junctions connecting three or more edges are related to one another via positional relations.

A precursor to edge-cycles is the detection and use of closure, i.e. when a group of connected edges closes in on itself. Closure is used to identify three-sided and four-sided shapes, and to identify corners that are convex with respect to some enclosed region. Edge-level properties that are constant within a shape (e.g., all of the edges are `straightEdge` or all of the corners are convex), are propagated upward to the shape-level representation of the object of which they are a part.

It should be noted that edge-level representations can be quite large: The sketches used in the experiment described here sometimes involved over 600 facts [Lovett *et al*., 2006], making analogical matching quite challenging. In fact, pruning heuristics were used in the 2006 experiments to eliminate many of the edges, based on giving priority to edges that were connected to the exterior of an object. Experimentally, it was found that keeping no more than about 175 assertions yielded the best results for learning and classification. This result suggests that the edge-level representation, while crucial for understanding shapes, is too detailed for these purposes. This led us to the edge-cycle representation, described next.

3.3 Edge-cycle representation

The edge-cycle representation abstracts away from specific edges, to construct sparser representations. The sketch in Figure 1 illustrates the difference between the encodings.

There are two types of entities in the edge-cycle representation: *edge-connected objects* and *edge-cycles*. An edge-connected object is a maximal set of connected edges. For example, the brick in Figure 1 contains four objects: the brick itself and the three holes. An edge-cycle is a sequence of edges which is closed, i.e. traversing the path along the edges eventually leads back to the start point, without encountering any dead-ends. A single object may contain several edge-cycles or it may contain none at all.

There are two types of edge-cycles: atomic edge-cycles and perimeter edge-cycles. We define each below and then describe the attributes and relationships that are computed for objects and edge-cycles.

Atomic edge-cycles

An atomic edge-cycle is one that contains no other cycles that share any of its edges. For example, in Figure 1(c), the

brick's outer contour is *not* an atomic edge-cycle because it contains three edge-cycles that share edges with it (the three surfaces of the brick). However, each of these is an atomic edge-cycle.

Atomic edge cycles are found via the closure computation in the edge-level representation. We believe atomic edge-cycles are an appropriate basic entity type, since people have been found to compute closure early on in spatial processing [Treisman and Paterson, 1984]. Atomic edge-cycles form salient closed shapes, and properties that are uniform within them are propagated from the edges to the shape itself (e.g. straight edges, convex corners).

Perimeter edge-cycles

A perimeter edge-cycle describes the shape of an object's outer contour. It consists of the cycle of edges that form the exterior of an edge-connected object (provided there is such a cycle. As with atomic edge-cycles, perimeter edge-cycles are closed shapes, so shape-level attributes and relationships are also computed over them. In some cases, a perimeter edge-cycle may also be atomic, e.g., the holes in the brick in Figure 1(c). Since outer contours are highly salient [Hoffman and Richards, 1984], a non-atomic term (using the logical function `PerimeterFn`) is used to describe such entities. This increases the order (in the structure-mapping sense of how nested an expression is), thereby causing the exterior to provide more weight in the mapping.

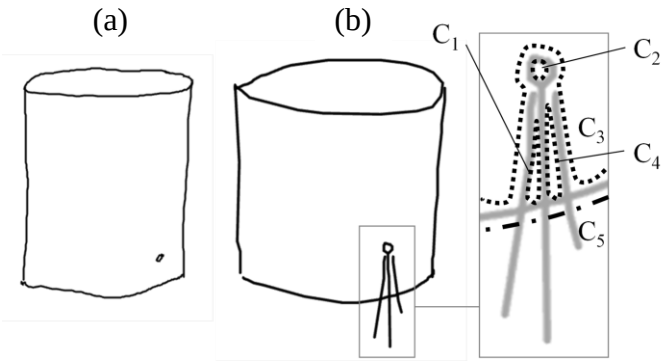

Figure 2: Two participants' sketches of cylinders, both drawn with water spouts. In (*a*), the spout and the cylinder are different edge-connected objects. In (*b*), the spout and the cylinder are connected by the stream of fluid, and thus, part of the same edge-connected object. The close-up shows the atomic (C_1-C_4) and perimeter (C_5) edge-cycles in the vicinity of the stream depiction. The symmetric `sharesEdgesWith` relation holds between the pairs: {C_1: C_4}, {C_1: C_3}, {C_1: C_5}, {C_4: C_3}, {C_4: C_5}, {C_3: C_5} {C_3: C_2}. The `touchesDirectly` relation, also symmetric, holds between all of these same pairs as well as {C_1: C_2} and {C_4: C_2}. C_2 is related to C_3 by `edgeSubsetOf`, since all of the edges that comprise C_2 (in this case only one edge) are also in C_3. The three bottom edges are not part of any edge-cycle because they terminate at one end, but they add an `edgeProtrudedShape` attribute to the perimeter edge-cycle C_5.

Attributes and relations

The attributes and relations computed for edge cycles are primarily those used in CogSketch's shape representation, including the ability to propagate shared edge properties upward to the shape, e.g. `2DShape-Convex` (all corners and curved edges in the shape are convex) and `2DShape-AxisAligned` (every edge that makes up the shape is axis-aligned). Similarly, positional relationships are computed (`above/rightOf` and `enclosesVertically` / `enclosesHorizontally`), as well as relative-orientation relations (`parallelElements` / `perpendicularElements`). Two additional attributes are computed to encode the presence of edges that are not part of any edge-cycle. These edges, called *terminal edges*, include (1) any edge that is not connected to any other edge on one of its ends, and (2) any edge that is only connected only to terminal edges on one of its ends. Terminal edges inside an atomic edge cycle are encoded with as `edgeIntrudedShape`, similarly, terminal edges outside of a perimeter edge cycleis encoded as `edgeProtrudedShape` (e.g., C_5 in Figure 2(b)).

Intersection relationships between edge-cycles are important because they signal connectivity in what they depict. Edge-cycles that share an edge are related by `sharesEdgesWith`, and any two edge cycles that share either a junction or an edge are related by `touchesDirectly`, which provides a notion of adjacency. Positional relations and relative-orientation relations are only computed by default between adjacent edge-cycles. A third intersection relation, `edgeSubsetOf`, is used in the rare case when all of the edges in one cycle form a subset of those in another cycle (e.g., C_2 and C_3 in Figure 2(b)).

Containment relationships between atomic edge-cycles and edge-connected objects are important for encoding nested imagery within a sketch. The `atomicPartOf` relation connects an atomic cycle to the edge-connected object it is part of. Conversely, the `containsObject` relation holds between an atomic edge-cycle and an edge-connected object if the cycle spatially contains the object without sharing any edges or junctions with it (e.g. the brick's holes in Figure 1).

4 Experiment

We evaluated our cycle-based encoding approach on the learning task of [Lovett *et al.*, 2006], using the original edge-level representations as a baseline. This experiment was based on a corpus of hand-drawn sketches from 10 participants depicting each of 8 everyday objects from the book *Sun Up to Sun Down* [Buckley, 1979], a book that uses simple drawings to illustrate physical processes such as heat transfer. The sketched concepts in the dataset are a brick, a bucket, a cup, a cylinder, a fireplace, a house, an oven, and a refrigerator. Participants were instructed to sketch each object in CogSketch using the corresponding picture from the book as a guide, so that the general features and orientations of the sketches would be similar.

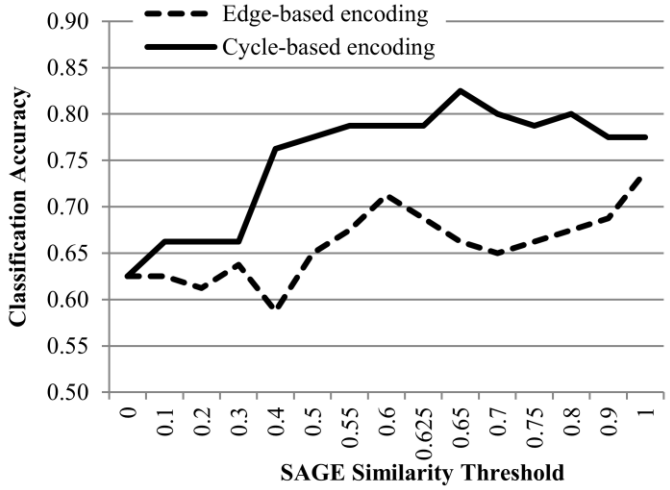

Figure 3: The classification accuracy of the SAGE-based learner at various similarity thresholds (*S*), when run on each encoding scheme. The difference in performance is significant with $p < 0.05$ for all $0.4 \leq S \leq 0.8$, with one exception at $S = 0.6$.

4.1 Evaluation Procedure

All 80 sketches in the corpus were automatically encoded using two different methods: the baseline edge-based encoding, and the present edge-cycle-based encoding. To test their effectiveness, we used the same supervised learning task from [Lovett *et al*., 2006]. That is, in each condition, categories were learned via SAGE, using the process described in Section 2.2. During testing, unlabeled examples were classified using MAC/FAC. We used a 10-fold cross-validation, with the 8 sketches generated by a particular participant constituting a fold. This resulted in 10 rounds of training and testing per condition, where the system does analogical learning over nine participants' sketches and categorizes those of the tenth. To explore how SAGE's similarity threshold affects the results, we ran this experiment across a range of values for this parameter, from 0 to 1.

4.2 Results

Figure 3 summarizes the results. Recall that the similarity threshold *S* is the minimum similarity score (computed by SME) required between an example and another example (or generalization) for SAGE to combine them. With $S = 0$, SAGE always combines descriptions, and the two encodings achieved equal performance. For every other value, the cycle-level encoding outperformed the edge-level encoding. Moreover, for similarity thresholds between 0.4 and 0.8, with the exception of 0.6, the performance was significantly different ($p < 0.05$, one-tailed paired t-test). The difference between the maximum accuracies achieved in the two conditions, which was at $S = 0.65$ for the cycle-based encoding and $S = 1.0$ for the edge-based encoding, was marginally significant at $p < 0.06$.To better understand these results and the tradeoffs involved, it is useful to look more closely at the relationships between the representations computed. Figure 4 shows the confusion matrices produced by the classifier in the cycle-based and edge-based conditions, with the similarity threshold set to 1.0. With that high of a similarity threshold, SAGE never generalizes, and hence classification consists of finding the closest exemplar. This factors out analogical generalization, so that we can focus on properties of the representations produced.

We found two main reasons for confusability in the edge-level representations. The first was the need for pruning, as mentioned above. Ovens were the worst for the edge-level encoding relative to the cycle encoding, so a closer look can be revealing. With edge-level representations, houses and refrigerators were often retrieved. A closer look revealed that the particular sketches of ovens that were labeled as houses and refrigerators shared an unusual feature along their exteriors where four edges meet, which showed up in the edge-based representations as a high order relation (and therefore very influential in determining structural similarity). This feature was not prevalent across all ovens, but it was among houses and refrigerators. Two of the three ov-

Cycle-Based, *S* = 0.65

Actual \ Predicted	Brick	Oven	Fridge	House	Fireplace	Cup	Bucket	Cylinder
Brick	10	0	0	0	0	0	0	0
Oven	1	9	0	0	0	0	0	0
Fridge	0	0	8	1	0	1	0	0
House	0	0	0	10	0	0	0	0
Fireplace	0	0	0	2	7	1	0	0
Cup	0	0	0	1	0	6	3	0
Bucket	0	0	0	0	0	1	7	2
Cylinder	0	0	0	0	0	2	3	5

Edge-Based, *S* = 1.0

Actual \ Predicted	Brick	Oven	Fridge	House	Fireplace	Cup	Bucket	Cylinder
Brick	9	0	1	0	0	0	0	0
Oven	1	6	1	2	0	0	0	0
Fridge	1	0	8	0	1	0	0	0
House	0	2	0	8	0	0	0	0
Fireplace	0	0	0	0	10	0	0	0
Cup	0	0	0	0	0	6	3	1
Bucket	1	0	0	0	0	3	5	1
Cylinder	0	0	0	0	0	2	1	7

Figure 4: The confusion matrices produced by the classifier on the cycle-based (*left*) and edge-based (*right*) encodings at their maximum performances ($S = 0.65$ and $S = 1.0$, respectively).

ens that were confused with houses and refrigerators also had truncated edge-level representations that contained no mention of the round edges depicting the stovetop burners and knobs, which are intuitively the most salient differences between the sketches of ovens and those of houses and refrigerators. This illustrates that an outline-based pruning heuristic can lead to critical internal features being omitted from the representation.

The second reason for confusability in edge-level representations was segmentation errors. Recall that digital ink generally artifacts that can make accurate segmentation difficult. Edges that connect end-to-end at 2-way junctions can be especially difficult. This can result in unintended junctions and failure to recognize intended junctions. This can have a significant impact on the edge-level representation. For example, in Figure 5(a), the cylinder's top edge is split into two. This means that different entities are participating in the three-way junctions on either side of the top. Since SME requires mappings to be 1:1, this substantially reduces the similarity computed for the edge-level representation. The edge-cycle representation is robust relative to extra junctions because it abstracts the edges away to cycles.

On the other hand, a close analysis of the confusion matrix for the edge-cycle representation reveals that it has its own set of problems. Consider the cylinder in Figure 5(b), which contains a gap in the top. Not closing this gap leads to an edge-cycle description with half of the atomic edge-cycles of the cylinder in 4(a), changes the perimeter edge-cycle to be convex, and removes an edge-protrusion attribute. These are substantial changes that radically reduce perceived similarity by the system, although people seem to "fill" these gaps reasonably easily. The edge-level representations are more stable when gaps occur, since most of the constituents will remain unchanged.

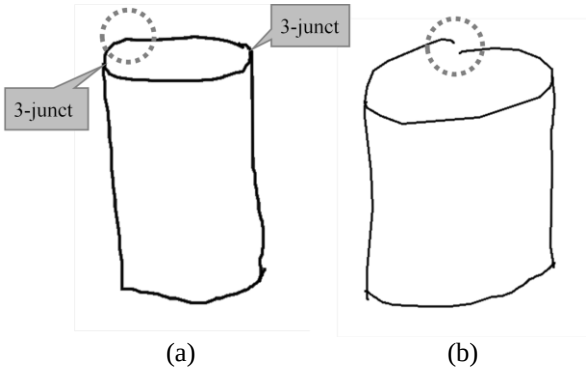

Figure 5: Sketches of cylinders whose edge segmentations were problematic. In (*a*), an extra junction was detected due to a sudden change in curvature that was a presumably unintentional. In the edge-based representation this split up a critical edge entity involved in two three-way junctions. In (*b*), two edges that were supposed to be connected were not due to a gap in the ink. The cycle-based encoding subsequently lost an entire atomic edge-cycle and misconstrued the perimeter shape.

Another problem with our current edge-cycle representation is that our qualitative vocabulary isn't expressive enough to capture a number of useful properties of shapes. Figure 6 shows one participant's sketch of a fireplace (*left*) along with the most similar examples retrieved in the cycle-based (*middle*) and edge-based (*right*) conditions. A detailed examination of the correspondences is revealing: The textures on the bottom row indicate the correspondences between the sample (*bottom-left*) and the retrievals from the edge-cycle (*bottom-middle*) and edge-level representation (*bottom-right*). The fire corresponds with the window in the edge-cycle representation because they both have many straight lines, although the number of lines in both is quite different. Relative size is not taken into account, e.g. the match between the top of the mantle and the roof. Adding these and other higher-level properties (e.g. that the door is almost entirely contained within the wall of the house, whereas the inner wall of the mantle is more exposed to the exterior) could help improve both retrieval and mapping.

Another problem with our current edge-cycle representations is that they are still too sensitive to small variations in drawing style. Seemingly minor depiction differences can lead to large differences in the number of edge-connected objects present. For example, fluids were sometimes depicted with lines in the foreground of the scene, connecting otherwise separate edge-connected objects (e.g. the stream falling from the cylinder's spout in Figure 2(b)). One fireplace was depicted with several small lines of rising smoke, causing the number of edge-connected objects in the scene to triple. Such "decorations" do not cause similar problems for people, although they can cause serious trouble for any current recognition-based approach. We return to the decoration problem below.

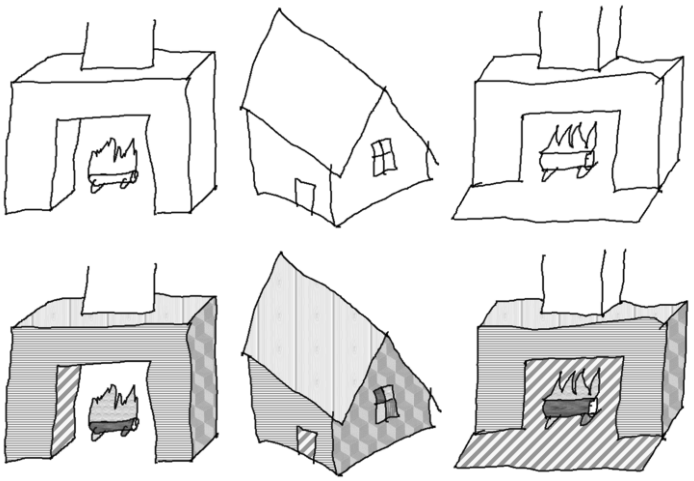

Figure 6: Across the top row are an unlabeled example encountered during testing (*top-left*) along with the top retrievals from memory in the cycle-based (*top-middle*) and edge-based (*top-right*) conditions. The bottom row uses textures to depict the structural mappings between each retrieval (*bottom-middle* and *bottom-right*) and the unlabeled example (*bottom-left*).

5 Related Work

Several previous systems generated symbolic representations at the edge level to support shape recognition [Ferguson and Forbus, 1999; Museros and Escrig, 2004; Veselova and Davis, 2004]. Though we represent shapes at a higher level of abstraction (edge-cycles), there are parallels between these previous approaches and our own.

Our use of adjacency to restrict what edge-cycle relationships are computed is similar to Veselova and Davis' [2004] use of adjacency in their edge-level representations. When determining which constraints to infer from a sketch, one heuristic they use is looking for special cases of geometric configurations (e.g. perpendicular or collinear lines). Our strategy for propagating edge-level attributes and relations to edge-cycles, which is to do so only when they are shared across an entire shape, places a similar emphasis on special cases at the edge-cycle-level.

Like our cycle-encoding, Museros and Escrig's [2004] qualitative shape descriptions are heavily based on closure, since they target a task that involves recognizing geometric tiles. Their vocabulary was designed to handle shapes that are less organic and noisy than those found in our sketches, and is not equipped with as rich a vocabulary for relating intersecting closed shapes to one another, as our approach does. It does, however, use edge-level descriptors that our approach could benefit from (e.g. acute/obtuse angles).

There have been decades of work on reconstructing 3-D object models from 2-D line-drawings via line-labeling [Clowes, 1971; Huffman, 1971; Malik, 1987; Cooper, 2007], but these techniques are highly specialized for 3-D modeling tasks. They typically deal with sketches that are far less noisy than the sketches here, in which all lines are taken to be relevant to the 3-D contour of the object.

The segmentation algorithm used in our approach draws some ideas from scale-space techniques for segmenting based on discontinuities in curvature [Mokhtarian and Mackworth, 1986; Saund, 1990; Witkin, 1989]. Saund's [1999] work on subjective contour segmentation using deterministic annealing is promising for overcoming errors due to gaps in digital ink.

We use SAGE to generalize across structured qualitative descriptions of edge-cycles. Recent work in the object recognition community has mirrored this approach with efforts to abstract generic models from multiple images, by reducing segmented images to graph representations and then employing editing and matching operations. Keselman and Dickinson [2005] create region adjacency graphs from segmented input images, and search for approximate *least common abstractions* of multiple examples based on edit-distance (effectively adding/removing edges between regions). Todorovic and Ahuja [2008] learn region-based hierarchical models for object categories, while incorporating geometric and photometric properties into the similarity measure that is used to match regions. Both of these approaches allow for many-to-many mappings between regions, which SME does not handle. Fidler and Leonardis [2007] learn a large hierarchy of part compositions for object categories in which simpler parts at lower levels are shared across objects at the higher levels. The part-hierarchy supports robust matching going from the top down and efficient indexing going from the bottom up. Our approach addresses these same concerns via SME's systematicity constraint and MAC/FAC's coarse first-stage filter, respectively. However, these other approaches have not been tested on hand-generated line drawings. Sketches are subject to noisy ink and depiction differences across participants, and they usually lack many of the segmentation clues that real-world images contain, such as color, texture and lighting. Our qualitative ink descriptions are robust against noise because minor quantitative variance is ignored.

6 Summary and Future Work

We have introduced the idea of edge-cycles, a higher-level qualitative representation for sketches that is generally more sparse and stable than the edge-level representation it is computed from. This resulted in a significant improvement in learning and classification compared to our previous edge-level representation.

However, as the analysis above indicated, there is still more work needed. First, incorporating better gap-filling strategies, such as Saund's work on subjective contours, could be useful. Solving the Decoration Problem very likely requires a combination of a richer vocabulary for describing edge-cycles and a more incremental approach to encoding and retrieval (e.g. recognizing parts, like fires in fireplaces, burners on stoves, and handles on doors). This in turn requires building up robust models of depiction [Lockwood *et al*., 2008]. We plan to combine crowd-sourced gathering of labeled sketches with active learning [Settles, 1994], by developing heuristics for generating new drawings from edge-cycle representations.

Acknowledgments

This work is funded by the Air Force Office of Scientific Research, the Office of Naval Research, and the NSF Spatial Intelligence and Learning Center.

References

[Buckley, 1979] Shawn Buckley. *Sun Up to Sun Down*, McGraw-Hill. 1979.

[Clowes, 1971] M. B. Clowes. On seeing things. *Artificial Intelligence* 2(1): 79-116. 1971.

[Cohn *et al*., 1997] Cohn, A. G., Bennet, B., Gooday, J., and Gotts, N. M. 1997. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *GeoInformatica* 1: 275-316.

[Cooper, 2007] Martin C. Cooper. Constraints Between Distant Lines in the Labelling of Line Drawings of

Polyhedral Scenes. *International Journal of Computer Vision*. 2007.

[Falkenhainer *et al*., 1989] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence* 41(1): 1-63. 1989.

[Ferguson, 1994] Ron W. Ferguson. (1994). *MAGI: Analogy-based encoding using symmetry and regularity*.

[Ferguson and Forbus, 1999] Ron W. Ferguson and Kenneth D. Forbus. GeoRep: A flexible tool for spatial representation of line drawings. *Proceedings of QR99*, Loch Awe, Scotland. 1999.

[Fidler and Leonardis, 2007] Sanja Fidler and Ales Leonardis. Towards scalable representations of object categories: Learning a hierarchy of parts. *Computer Vision and Pattern Recognition*. 2007.

[Forbus *et al*., 2011] 1. Forbus, K., Usher, J., Lovett, A., and Wetzel, J. (2011). CogSketch: Sketch understanding for Cognitive Science Research and for Education. *Topics in Cognitive Science*. pp 1-19.

[Forbus *et al.*, 1995] Kenneth D. Forbus, Dedre Gentner, and Keith Law. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science* 19(2): 141-205.

[Gentner, 1983] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science* 7(2): 155-170. 1983.

[Halstead and Forbus, 2005] Halstead, D. and Forbus, K. (2005). Transforming between Propositions and Features: Bridging the Gap. *Proceedings of AAAI-2005*. Pittsburgh, PA.

[Hochstein and Ahissar, 2002] Shaul Hochstein and Merav Ahissar. View from the Top: Hierarchies and Reverse Hierarchies in the Visual System. *Neuron* 36(5): 791-804. 2002.

[Hoffman and Richards, 1984] D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition* 18(1-3): 65-96. 1984.

[Huffman, 1971] Dennis D. Huffman. Impossible objects as nonsense sentences. In: B. Meltzer and D. Mitchie (Eds.), *Machine Intelligence 6*, Edinburgh University Press, Endinburgh, pp. 295–323. 1971.

[Keselman and Dickinson, 2005] Yakov Keselman and Sven Dickinson. Generic model abstraction from examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 27(7): 1141-1156. 2005.

[Kuehne *et al*., 2000] Kuehne, S., Forbus, K., Gentner, D. and Quinn, B. (2000). SEQL: Category learning as progressive abstraction using structure mapping. *Proceedings of CogSci 2000*, August.

[Lockwood *et al.*, 2008] Kate Lockwood, Andrew Lovett, Kenneth D. Forbus, Morteza Dehghani, and Jeffrey Usher. A Theory of Depiction for Sketches of Physical Systems. In the *Proceedings of QR 2008*. 2008.

[Lovett *et al.*, 2006] Andrew Lovett, Morteza Dehghani, and Kenneth D. Forbus. Efficient Learning of Qualitative Descriptions for Sketch Recognition. *Proceedings of the 20th International Qualitative Reasoning Workshop*, Hanover, New Hampshire. July. 2006.

[Lovett *et al.*, 2008] Andrew Lovett, Kate Lockwood, Kenneth D. Forbus. A computational model of the visual oddity task. In the *Proceedings of the 30th Annual Conference of the Cognitive Science Society*. Washington, D.C. 2008.

[Lovett *et al.*, 2009] Andrew Lovett, Emmett Tomai, Kenneth D. Forbus, Jeffrey Usher. Solving Geometric Analogy Problems Through Two Stage Analogical Mapping. *Cognitive Science* 33(7): 1192-1231. 2009.

[Malik, 1987] Jitendra Malik. Interpreting line drawings of curved objects. *International journal of computer vision* 1(1): 73-103. 1987.

[Mokhtarian and Mackworth, 1986] Farzin Mokhtarian and Alan Mackworth. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1): 34-43. 1986.

[Museros and Escrig, 2004] Lledó Museros and M. Teresa Escrig. *A qualitative theory for shape representations and matching*, Citeseer. 2004.

[Saund, 1990] Eric Saund. Symbolic construction of a 2-D scale-space image. *IEEE transactions on pattern analysis and machine intelligence*: 817-830. 1990.

[Saund, 1999] Eric Saund. Perceptual organization of occluding contours of opaque surfaces. *Computer Vision and Image Understanding* 76(1): 70-82. 1999.

[Settles 1994] Burr Settles. Active Learning Literature Survey. *Machine Learning* 15(2): 201-221. 1994.

[Todorovic and Ahuja, 2008] Sinisa Todorovic and Narendra Ahuja. Region-based hierarchical image matching. *International Journal of Computer Vision* 78(1): 47-66. 2008.

[Treisman and Paterson, 1984] Anne Treisman and R. Paterson. Emergent features, attention, and object perception. *Journal of Experimental Psychology: Human Perception and Performance* 10(1): 12-31. 1984.

[Witkin, 1987] Andrew P. Witkin. Scale-space filtering. *Readings in computer vision: issues, problems, principles, and paradigms*. 1987.

[Veselova and Davis, 2004] Olya Veselova and Randall Davis. Perceptually based learning of shape descriptions. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI'04)*, pages 482-487, San Jose, CA, 2004. AAAI Press.