Finding Textures in Sketches using Planar Ising Models

Matthew D. McLure, Subu Kandaswamy, and Kenneth D. Forbus
Northwestern University

Abstract

Creating software that can understand the range of sketches that people produce is a challenging problem. One source of difficulty is that people often include textures in their drawings. This paper shows how to use Ising models, a technique from computer vision at the level of pixels, for decomposing digital ink into a hierarchy of edge-based structures that provide more concise qualitative representations of textures in hand-drawn sketches. We analyze the compression efficacy, strengths and weaknesses of this qualitative representation technique using a subset of a large-scale sketch corpus.

Introduction

Sketching is a natural way for people to interact, but automated sketch understanding is a difficult challenge because raw ink is noisy. Qualitative, structured representations of edges and shapes have performed well for learning sketched concepts from only a handful of examples (Lladós et al. 2001, Lovett et al. 2007, Lee et al. 2007, McLure et al. 2015). However, structured representation schemes can get bogged down by ink containing textures – regions of repeating, richly connected visual features. Textures are commonly included in drawings of certain categories of sketched object (e.g. turtle, pumpkin, pizza, feather, from Eitz et al. (2012) – all examples in this paper are drawn from this corpus of sketches), presumably because they are salient features of these concepts, facilitating recognition. To produce effective qualitative representations of sketched objects for analogical learning, we need to be able to construct concise qualitative representations of textured regions, to take advantage of this information.

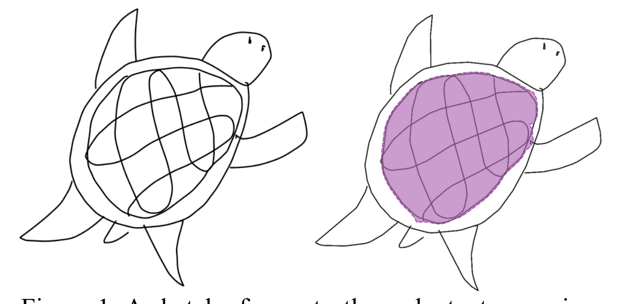


Figure 1: A sketch of a sea turtle, and a texture region detected by the algorithm (shaded).

This paper shows how to segment textures in digital ink using the Ising model, which originated in statistical mechanics as a model of ferromagnetic particle interactions. It became popular in computer vision for segmenting images into regions based on local features like pixel intensity, in part because it can be solved efficiently and exactly for certain classes of graphical models, including planar graphs. Here, we present methods for producing planar graphs at multiple levels of granularity for describing ink, so that an Ising model can be used to group them into qualitative descriptions, within which texture is in some sense uniform. We analyze the strengths and weaknesses of our approach to texture detection, as well as its compression efficacy, using a subset of the Eitz et al. (2012) corpus.

Background & Related Work

CogSketch

CogSketch (Forbus et al. 2011) is an open-domain sketch understanding system in which users can draw digital ink. Every pen stroke produces a *polyline*, a series of points. Sets of polylines can be manually grouped into conceptually meaningful objects called *glyphs,* either on-the-fly or post-hoc, which may then be assigned conceptual labels from the Cyc knowledge base[1]. Performing analogies between perceptual part-models of glyphs has the potential to facilitate recognition, making sketch interaction more fluid, but how to best organize the ink into salient, informative parts is an open problem.

CogSketch already constructs multiple levels of representations for digital ink within a glyph, based on a combination of evidence from studies of human vision and computational explorations of what information is salient and relevant for

[1] www.cyc.com/platform/opencyc

multiple tasks. Its lowest level of representation decomposes the ink into a network of edges and junctions, and characterizes the visual attributes of edges (e.g. curvature, arc length) and their relationships (e.g. acute/obtuse angles, relative orientation, relative length) with qualitative predicate calculus statements. CogSketch is also capable of describing ink at the level of cycles of edges, to which it applies a vocabulary of shape attributes (e.g. major axis orientation) and relationships (e.g. between). Finally, CogSketch can compute attributes and relations for edge-connected objects (ECOs); in the graph of edge/junctions, these are the *connected components* – the maximal connected subgraphs. ECOs are described using many of the same descriptors as edge-cycles (e.g. between), excluding those that characterize connectivity. See McLure et al. (2011) for details. Here, these three levels are used to construct the planar graph embeddings in which we define our Ising models.

Analogical Learning

The qualitative representations of ink we generate here are meant to be input to a suite of analogy models based on Gentner's (1983) structure-mapping theory of analogy and similarity. The Structure-Mapping Engine (SME, Falkenhainer et al. 1989) takes as input two structured relational representations (*cases*), and computes one or more mappings in polynomial time, using a greedy algorithm (Forbus et al. 1994). Analogical retrieval is modeled by MAC/FAC (Forbus et al. 1995), which uses a two-stage model to provide scalable similarity-based retrieval. Analogical generalization is modeled by SAGE (McLure et al. 2010), which can cluster and compress a series of positive examples of a concept into a set of structured prototypes and outlying examples. The generalizations are probabilistic, and creating them does not involve introducing logical variables. MAC/FAC and SAGE can be used in concert to classify examples, by retrieving over the union of concept prototypes. ALIGN (McLure et al. 2015) further extends SAGE by adding automatically constructed near-misses, which help improve discriminability.

The Goldilocks Hypothesis (Finlayson and Winston 2006) predicts that the most productive representations for analogical retrieval and matching describe input in terms of its intermediate properties – not too big, not too small; not too simple, not too complex. Graph-based approaches to sketch recognition that describe ink at the level of visual primitives such as edges or closed regions have been effective for recognizing relatively simple symbols (each containing on the order of 10 primitives; Lladós et al. 2001, Lee et al. 2007). But a sketch recognizer in an open domain may encounter richer stimuli, which, when described at the level of primitives, can overwhelm an analogical or otherwise graph-based matcher. Lovett et al. (2007) applied analogical learning to rich sketch stimuli by using a ranking technique to prune facts from oversized cases. Our strategy is to compress at an earlier stage, when ink is being perceptually organized into entities.

The Ising Model

In machine learning, undirected graphical models express energy functions as a sum of individual energy functions over every edge and every node in a graph. They can be used to infer labels for the nodes in the graph, where lower-energy states are more probable and the lowest-energy (ground) state is the maximum a posteriori (MAP) estimate.

The Ising model is an undirected graphical model that makes two assumptions: (1) The node labels are binary, and (2) every edge energy can be expressed as a *disagreement cost* – a fixed energy cost incurred only when the labels for the two nodes connected by the edge have different labels. Here we are specifically interested in the zero-field Ising model, which additionally assumes that (3) there are no node energies. With these three constraints, the energy function boils down to

$$E(y) = \sum_{\{i,j\}} [y_i \neq y_j] \, D_{ij}$$

where y_i is the label for node i, y_i the label for node j, and D_{ij} is the disagreement cost associated with the edge $\{i, j\}$. Because the energy is just a sum of disagreement costs, and because a binary labeling over nodes is equivalent to a graph cut, solving for the ground state of the model reduces to a min-cut/max-flow problem on a weighted, undirected graph, where the edge weights are the disagreement costs.

Efficient Solutions: Submodularity vs. Planarity

The minimum-weight cut for an arbitrary undirected weighted graph can be found in polynomial time if there are no negative disagreement costs in the graph, as this ensures submodularity. This approach requires some localized background knowledge about labels, often in the form of extra *terminal nodes* with fixed labels that can be connected to nodes in the original model to bias them (Boykov and Veksler 2006). Otherwise, the empty cut (a uniform labeling) would always be the lowest energy state.

However, the Ising model does not need to be submodular to be solved in polynomial time if it is defined on a graph that has a planar embedding – informally, a way to lay it out on a plane such that no two edges cross (Kasteleyn 1961; Fisher 1961; Globerson and Jaakkola 2007; Schraudolph and Kamenetsky 2009). In this case the edges in the model can have real-valued disagreement costs, and the ground state is computed via the minimum-weight graph cut directly, with no terminal nodes required.

The real-weighted (planar) model is more flexible than the non-negative (submodular) model in that it does not require any nodes to be biased towards a particular label. On

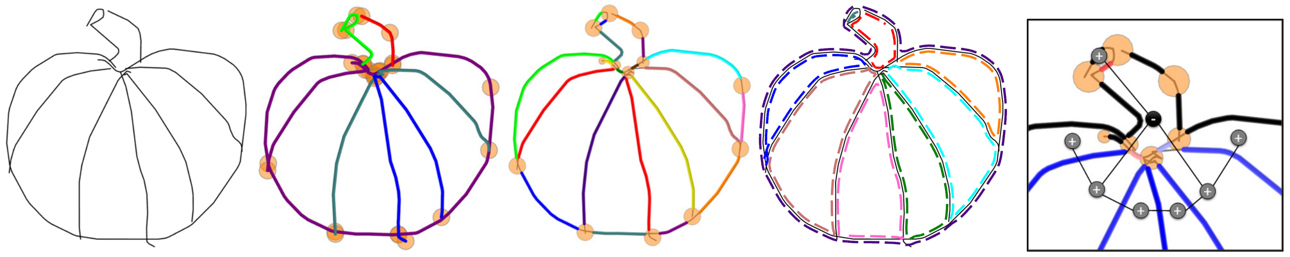


Figure 2: (1) A sketch of a pumpkin, overlaid with (2) its raw polylines and primitive junctions, (3) its final edges and junctions, (4) its edge-cycles, and (5) its ground-state Ising model, with edges colorized to reflect relative disagreement cost (blue = affinity, red = anti-affinity). In this case, the ground state ends up violating the preference of one of the edges (the short affinity edge in the center); to allow every edge its preference would not be a legal cut in this case.

the other hand, it requires a stronger bias on the edges; whereas the submodular model just calls for a partial ordering over the edges' individual (quantitative) preferences to group the nodes that they connect, real-weights additionally require that we encode whether each edge should have an *affinity* – a qualitative preference to group – or an *anti-affinity* – a preference to separate. This is a trade-off, but the real-weighted approach has an appealing property for our application: It can induce an arbitrary number of connected components in the graph quite naturally with a single cut, whereas the submodular approach requires strategically biasing at least a corresponding number of nodes to do so – a non-trivial task.

Approach

Here we present methods for finding textures in digital ink using planar Ising models. We begin with an account of how CogSketch decomposes a glyph's ink into a network of edges/junctions, edge-cycles, and ECOs, and how we use them to determine the structure of our Ising models. We then describe the local geometric features used to assign disagreement costs to the edges of each model. Next, we discuss our method for performing a real-weight graph cut to find the ground state of the model, and explain how we extract textures from the results of the cut. The final subsection explains how the texture entities are described qualitatively for analogy.

Determining Ising Model Structure

In this subsection we explain how CogSketch constructs its network of edges, junctions, edge-cycles and ECOs, and how these structures are used to produce planar embeddings on which we define our Ising models.

Establishing Planarity at the Edge Level

We begin at the lowest level – the placement of edges and junctions to form a planar embedding, where CogSketch's edges correspond to the edges in the embedding and its junctions correspond to the nodes. Planarity is enforced by working within the following constraints:

(a) Every edge is a simple curve (no self-intersections).

(b) No two edges intersect.

(c) Edges can only intersect junctions at their endpoints.

(d) Every edge endpoint must intersect the boundary of exactly one junction.

(e) No two junctions intersect.

The goal at this stage is to place junctions and edges in a way that abides by these constraints while best representing the geometry of the raw ink. Our strategy is to first place junctions such that all of the intersections and endpoints of the raw ink polylines are covered by some junction and no two junctions intersect. We then construct edges from each maximal ink segment that isn't covered by a junction.

Junction placement begins by finding all polyline intersections (self or mutual) and polyline endpoints in the raw ink. These points are marked as critical. CogSketch also

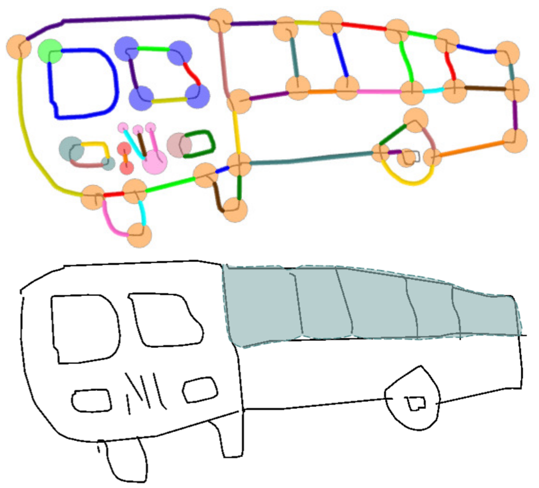


Figure 3: (Top) Sketch of a bus overlaid with its edge/junctions embedding, in which the 7 separate ECOs are marked with different junction colors. (Bottom) A single (shaded) texture found in the outermost ECO.

finds corners and inflection points in the ink using a modification of the Curvature Scale Space (CSS) corner detector (Lovett et al. 2012; Mohktarian and Suomela 1998), and marks these points non-critical. For every critical and non-critical point, a new junction with a starting radius proportional to the size of the glyph is centered on the point. These primitive junctions are allowed to intersect. Clusters of overlapping junctions are identified, and junctions in each cluster either shrink to produce smaller clusters, or merge such that all critical points overlapping the cluster are still overlapping the new, merged junction. Our heuristic is to shrink up to a minimum junction size, at which point we merge the remaining clusters. Junction placement terminates when no two junctions intersect.

ECOs and their Edge-cycle Embeddings

Recall that the ECOs are the connected components in the edge/junction graph embedding. In this paper, we assume that a texture cannot span multiple ECOs, so we create and solve an Ising model for each ECO in isolation. The algorithms we describe below could, in theory, be applied to one big Ising model for the entire sketch. Our decision to build them per ECO primarily affects how disagreement costs are normalized, which is discussed in the next section. Ideas for grouping ECOs themselves into more abstract textures are revisited in the Future Work section.

To build an Ising model for an ECO, we begin with its edge/junction embedding. As a subgraph of the overall edge/junction embedding, it inherits planarity. CogSketch creates an edge-cycle around each *face* in the ECO's edge/junction embedding, i.e. every region of white space created by the ink. In Figure 2, the pumpkin is a single ECO and its edge-cycles are marked with dashed lines. We then construct a new graph embedding for an ECO at the edge-cycle level by creating a node for every cycle except the perimeter cycle, and placing an edge between the nodes for any pair of cycles that share an edge at the edge level, such that the new edge intersects the shared edge. In graphical terms, the ECO's edge-cycle embedding is the *dual* of its edge/junction embedding, excluding the node for the *infinite face* – the whitespace outside the perimeter that isn't bounded by any ink – along with the edges incident to that node. As such, the edge-cycle embedding inherits planarity. This embedding is the structure on which the ECO's Ising model is built; every edge between adjacent edge-cycles is assigned a disagreement cost in the model. The next section explains how these disagreement costs are computed.

Determining Disagreement Costs

The disagreement cost between two neighboring edge-cycles is determined by summing measurements of their similarity along 8 dimensions, listed in Table 1. The dimensions of similarity were chosen to correlate with likely texture groupings. Dimensions 1-4 are meant to capture size and

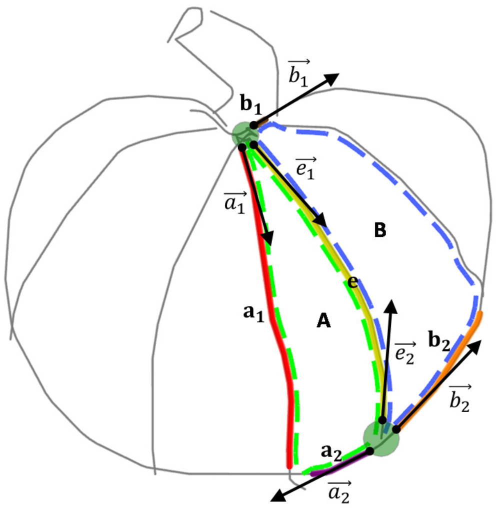


Figure 4: The geometry used to compute the disagreement cost for edge **e** (yellow) that is shared between edge-cycles **A** (green) and **B** (blue). Edges $\mathbf{a_1}$ (red) and $\mathbf{b_1}$ (brown) form an ordered pair of *cross edges* for **e**, as do $\mathbf{b_2}$ (orange) and $\mathbf{a_2}$ (purple). The vector $\overrightarrow{a_1}$ denotes the local orientation at end of $\mathbf{a_1}$ that intersects the shared junction. Note that while the cross edge polylines are always oriented in the clockwise direction (for curvature comparisons), the vectors for their endpoint orientations are always pointing away from the shared junctions.

Area	The areas of the edge-cycles' polygons
Major-axis orientation	The orientations of the 3D major axes for the cycles' polygons, where roundness is a *z* axis
Perimeter complexity	The number of edges in the cycle
Contents	The number of ECOs contained by the cycles (e.g. Six are contained by the cycle surrounding the front of the bus in Figure 3)
Cross-edge orientation	The local orientations of the cross edges in each pair as they pass through the shared junctions (e.g. the angular distance between $\overrightarrow{-\boldsymbol{a_1}}$ and $\overrightarrow{\boldsymbol{b_1}}$ in Figure 4)
Cross-edge curvature	The (signed) average curvatures of the cross edges in each pair
Cross-edge length	The length of the cross edges in each pair
Adjacent angles	The adjacent angles created between the shared edge and each cross edge (e.g. the absolute difference between the angular distance between $\overrightarrow{\boldsymbol{a_1}}$ and $\overrightarrow{\boldsymbol{e_1}}$ and the angular distance between $\overrightarrow{\boldsymbol{b_1}}$ and $\overrightarrow{\boldsymbol{e_1}}$ in Figure 4). Important for radial textures, e.g. wheel.

Table 1: The dimensions of similarity that factor into the disagreement cost, and what exactly each one compares.

shape similarities in the cycles themselves, taken as polygons (e.g. two neighboring edge-cycles with similar areas are more likely to be intended as part of the same texture, ceteris paribus). Dimensions 5 and 6 reflect the expectation that neighboring edge-cycles in a texture are more likely to have good edge continuity where their boundaries meet, as is the case along the bottom of the pumpkin in Figure 4 or amid the cross-hatching in the skyscraper in Figure 5 (left). Dimensions 7 and 8 are meant to capture radial textures (e.g. a bicycle wheel), which tend to produce similarly sized adjacent angles and edges at their centers.

Note that in this paper, the disagreement cost is a sum, not a weighted sum; all dimensions of similarity are taken to be equally important. We leave for future work the addition of parameters corresponding to the dimensions, to make the model tunable, or better yet, learnable.

The similarity between two neighboring edge-cycles *i* and *j* along dimension *s* is computed as the (absolute) difference between them w.r.t. that dimension, $\Delta_s(i,j)$, inverted and normalized by some mean (absolute) difference for elements that dimension, $\overline{\Delta_s}$. Our equation for determining disagreement cost is then

$$D_{ij} = \sum_{s \in S} \frac{\overline{\Delta_s} - \Delta_s(i,j)}{\overline{\Delta_s}}$$

So whether neighbors i and j have an affinity/anti-affinity depends on the extent to which their difference along each dimension s is less/greater than $\overline{\Delta_s}$, summed across all dimensions. We do not compute the mean difference $\overline{\Delta_s}$ over the neighboring pairs of edge-cycles alone – the same set of pairs that defines the Ising model edges. While straight-forward, this would constrain the distribution of disagreement costs to have a mean of zero. This is a problem because sometimes affinities should be allowed to dominate (e.g. to determine that the skyscraper in Figure 5 (left) is one whole texture), and the same goes for anti-affinities (e.g. to determine that the sea turtle in Figure 5 (right) has no textures). Instead, we compute the mean difference $\overline{\Delta_s}$ over all pairs of elements measurable along a similarity dimension. For dimensions 1-4, this includes all pairs of edge-cycles in the ECO – not just neighbors. For dimensions 5-7, the mean is computed over all pairs of edges in the ECO that share a junction. For dimension 8, the mean is computed over all pairs of adjacent angles inside the interior of the ECO. This allows us to capture, for example, in Figure 5 (right), the fact that each pair of neighboring edge-cycles is more different than the average pair.

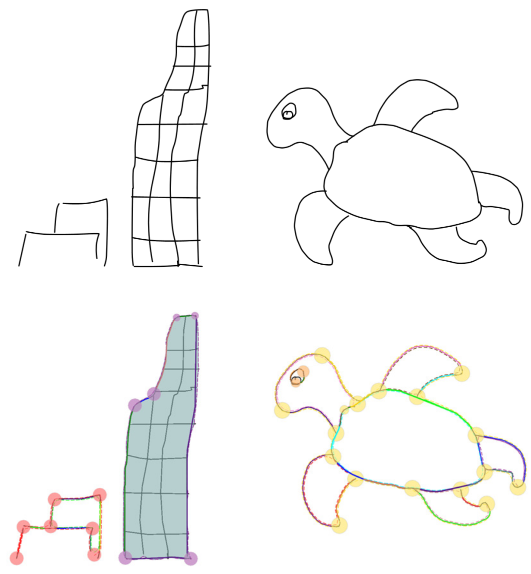


Figure 5: A skyscraper ECO identified as a single, complete texture (shaded), and a sea turtle identified as having no textures. Neither would be possible if we simply normalized using the mean of differences between neighbors.

Computing the Minimum-weight Cut

To find the minimum-weight cut of our planar, real-weight Ising model, we use the technique of Schraudolph and Kamenetsky (2009). It finds the minimum-weight cut by computing an *expanded dual* graph for the model – like the dual but with q-cliques in place of nodes, where q is the number of edges connected to the node – and solving for its maximum-weight perfect matching using the blossom-shrinking algorithm. The complement of this perfect matching in the original graph is its minimum-weight cut. We use the Blossom V implementation for blossom-shrinking (Kolmogorov 2009).

Extracting Textures: Measure Once, Cut Twice

Recall that a minimum weight cut on a real-weight graph can induce an arbitrary number of connected components in the graph (by removing the edges in the cut). Assuming that textures are contiguous, we construct a texture object for every one of these connected components that has grouped a sufficient number of edge-cycles – we use a minimum of three. To avoid false positives, we count on our Ising model to produce many small or singleton connected components in the non-texture regions of the ink.

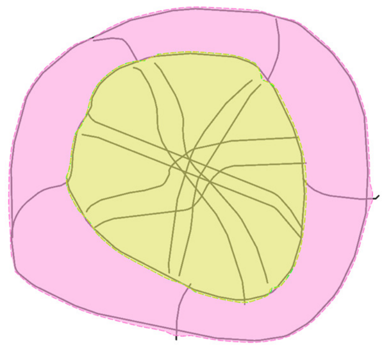
While the graph cut can induce an arbitrary number of components in the model, it cannot produce them in arbitrary configurations, because it is only performing a binary labeling. The four-color theorem tells us we need four labels to separate regions in an arbitrary planar map (Appel and Haken 1977). This problem is exemplified by the sketch of a bus in Figure 3. The detected texture would not have been possible to detect with a binary labeling because the front of the bus needs to disagree with the front-most window as well as with the side of the bus, the two of which additionally need to disagree with one another. A graph cut ends up forcing the windows of the bus to group with the side panel.

We address this problem by cutting twice in the same Ising model. The disagreement costs do not update, but before taking the second cut, we remove all of the cut edges that resulted from the first cut. We remove the union of both sets of cut edges from the edge-cycle embedding before looking for textures among the remaining connected components.

Qualitative Representations for Textures

Geometrically, the texture objects extracted are *multiple-connected regions*, as opposed to edge-cycles, which are *simply-connected polygons*. Informally, the difference is that the former is allowed to have holes, which allows us to capture textures like the outer one in Figure 6. To qualitatively represent a texture region in CogSketch's structured predicate calculus representations, we require an entity to represent the region itself, an edge-cycle entity to represent its perimeter, and *n* additional edge-cycle entities to represent its *n* holes. New edge-cycle entities are created whenever equivalent edge-cycles do not already exist. These new edge-cycles are naturally incorporated into CogSketch's existing scheme for representing edge-cycle configurations. We add additional facts to link texture entities to their perimeter and hole edge-cycles. We also assign attributes to the texture entities based on the same similarity dimensions described above. For each dimension, if the mean difference across all edges in the texture is lower than the overall mean difference $\overline{\Delta_s}$, then we assign the texture an attribute corresponding to that dimension. These texture-specific representations are sampled in Figure 6.

Our goal is to group CogSketch's more primitive elements into texture regions in order to compress qualitative representations for sketches without losing representativeness. Textures provide compression at the edge-cycle level by grouping entities into more abstract entities, but they also compress the edge level of description by pruning edges on the interiors of texture regions and potentially allowing edges along the their boundaries to be grouped if they have good continuity. For example, the wheel in Figure 6 goes from producing 868 facts at the edge-cycle level to producing 42 facts with texture compression (including texture-specific facts). At the edge level, the number of facts drops from 636 to 100.



```
(hasPerimeterEdgeCycle Texture-16
  (PerimeterEdgeCycleFn ECO-173))
(hasHole Texture-16 (HoleFn Texture-16 0))
(isa Texture-16 PerceptualTexture-CycleSize)
                    …
```

Figure 6: A sketch of a tire and a sample of facts from the qualitative representations with textures.

Analysis

We performed an analysis of texture compression on 10 concepts from the Eitz corpus. We asked a third party – someone not familiar with the research or the algorithm – to rank the 10 concepts by the extent to which they expected sketches of the concepts to include texture, from most texture to least. We added that larger, more complex textures should bear more weight in the ranking. We also encoded these 10 concepts (80 examples each) using CogSketch,

Label	Facts	Entities	Compressed Facts	Compressed Entities	Fact Compression	Entity Compression
Tire	648	45	275	25	0.58	0.44
Hot Air balloon	649	43	288	25	0.56	0.42
Bicycle	1333	82	633	52	0.53	0.37
Guitar	838	54	433	35	0.48	0.35
Suitcase	459	32	314	25	0.32	0.22
Cell phone	956	67	639	54	0.33	0.19
Carrot	556	41	383	34	0.31	0.17
Bear	586	46	502	43	0.14	0.07
Arm	252	21	224	20	0.11	0.05
Donut	424	29	358	28	0.16	0.03

Table 2: Average compression rates for 10 concepts from the Eitz corpus.

Predicted Rank	**Fact compression rank**	**Entity compression rank**
Bicycle	Tire	Tire
Hot air balloon	Hot air balloon	Hot air balloon
Cell phone	Bicycle	Bicycle
Tire	Guitar	Guitar
Guitar	Cell phone	Suitcase
Carrot	Suitcase	Cell phone
Suitcase	Carrot	Carrot
Donut	Donut	Bear
Bear	Bear	Arm
Arm	Arm	Donut
Correlation	**0.88** ($p<0.01$)	**0.81** ($p<0.01$)

Table 3: The predicted rankings in terms of textured content compared to the ranking in terms of the compression provided by our algorithm, with Spearman's rank correlations.

both with and without texture compression. We measured two types of compression: Fact compression and entity compression. The former is the proportional reduction in facts, and the latter is the proportional reduction in entities. We hypothesized that the order of the concepts by compression rate (decreasing) should correlate with the order predicted by the human. The data underlying the compression rates is in Table 2. The predicted (human) ordering and orderings by compression rates are in Table 3. The compression rankings were both strongly correlated with the human rankings. This gives us reason to believe that the compression is at least somewhat effective in targeting regions intended as textures, but a more rigorous evaluation involving an analogical learning task is necessary.

A common source of false negatives (textures that are overly fragmented or incomplete) seems to be errors early on in the segmentation process that propagate up to the

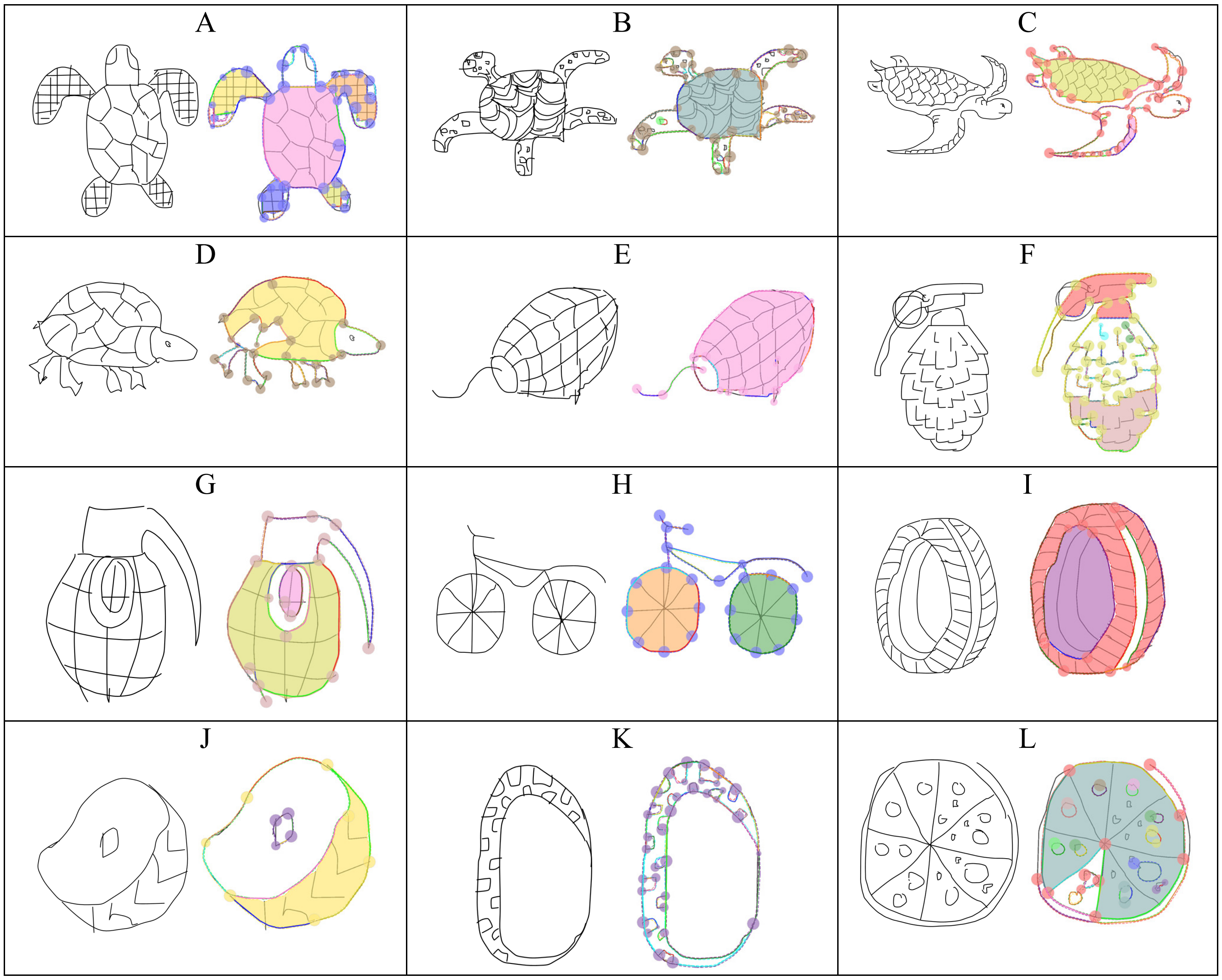


Figure 7: 12 examples from assorted categories in the Eitz corpus, with detected texture regions shaded.

edge-cycle level, making neighboring cycles look deceptively different. For example, The sea turtles in Figure 7A and 7D have edges in their textures that don't quite reach their respective perimeters because junctions shrank when they would have merged, resulting in unexpectedly merged cycles. The pizza in 7L has one slice excluded from the texture for the opposite reason; the junctions in the toppings got too close to the edge of the slice and merged into the ECO, altering the cycle's contents, size, and complexity.

Some textures are not possible to detect using edge-cycles directly because the regularity that defines the texture is across open-shaped pieces of ink. Figure 7F is an example of how an artistic rendering of a 3D texture results in a conglomeration of edges and corners whose shapes are similar, but whose closures (cycles) are unpredictable.

Other textures involve edge-cycles, but not ones that neighbor one another according to our strict definition involving shared edges. The tire in Figure 7K has many similar edge-cycles that are joined by a common, dissimilar neighbor cycle, which acts as a kind of "negative space" in the texture. The spots on the limbs of the sea turtle in 7B are mostly free-floating. Using connectivity as a proxy for adjacency seems too rigid in cases such as these.

Future Work

An analogical learning task will be necessary to evaluate the effectiveness of our texture segmentation/compression techniques. We expect our approach to perform on par with, or better than CogSketch's current representations in terms of accuracy. More importantly, we expect these techniques to make such an experiment dramatically more tractable on large, open-domain datasets such as on the entire 20,000-example Eitz corpus.

These Ising models are very fast to solve, which points to opportunities to apply them iteratively. Recall that we have the added flexibility of real-weighted edges because we are not constrained by submodularity. One approach for factoring in top-down information is to add a shared constant to the disagreement costs of all edges in the model – a *base* disagreement cost. This component can be seen as the general propensity to group elements; If the number of elements (e.g. edge-cycles) extracted from the Ising model is more/less than expected, we could increment/decrement the base cost, respectively.

All of the dimensions of similarity from which disagreement costs are computed are currently equally important. We would like to add parameters to weight these dimensions and explore techniques for adjusting them automatically, perhaps by performing parameter estimation using labeled data, or with a model of focus that gets primed in an unlabeled fashion.

The techniques used here can be extended with other methods for producing planar graphs in the ink. For example, a Voronoi Diagram (Edwards and Moulin 1998) defines a planar adjacency graph for any arrangement of disconnected shapes, and can therefore be applied to edges, which are kept disconnected by their junctions, or ECOs, which are disconnected by definition. The Ising models built at these alternative levels might help detect the grenade texture in Figure 7F and the pizza-topping texture in Figure 7L, respectively. We believe that strategies for allowing multiple layers of planar Ising models to mutually constrain each other would provide more robust results.

Acknowledgements

This work was funded by the Air Force Office of Scientific Research.

References

Appel, K., and Haken, W. 1977. Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics* 21(3): 429-490.

Boykov, Y., and Veksler, O. 2006. Graph cuts in vision and graphics: Theories and applications. In Handbook of mathematical models in computer vision, 79-96. Springer US.

Edwards, G., and Moulin, B. 1998. Toward the simulation of spatial mental images using the Voronoi model. In *Representation and Processing of Spatial Expressions*, 163-184. Hillsdale, NJ: LEA Press.

Eitz, M., Hays, J., and Alexa, M. 2012. How do humans sketch objects? *ACM Transactions on Graphics* 31(4).

Falkenhainer, B., Forbus, K., & Gentner, D. 1989. The structure-mapping engine: Algorithm and examples. Artificial Intelligence 41(1): 1-63.

Finlayson, M. & Winston, P. 2006. Analogical retrieval via intermediate features: The Goldilocks hypothesis. Technical Report MIT-CSAIL-TR-2006-071, MIT Computer Science and Artificial Intelligence Laboratory.

Fisher, M. E. 1961. Statistical mechanics of dimers on a plane lattice. *Physical Review* 124(6): 1664–1672.

Forbus, K., Ferguson, R., and Gentner, D. 1994. Incremental Structure-Mapping. *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, August.

Forbus, K., Gentner, D., and Law, K., 1995. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science* 19, 141-205.

Forbus, K. D., Usher, J., Lovett, A., Lockwood, K. and Wetzel, J. 2011. Cogsketch: Sketch understanding for cognitive science research and for education. *Topics in Cognitive Science* 3, 648-666.

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

Globerson, A., and Jaakkola, T. 2007. Approximate inference using planar graph decomposition. In *Advances in Neural Information Processing Systems 19*, 473. MIT Press.

Kasteleyn, P. W. 1961. The statistics of dimers on a lattice: I. the number of dimer arrangements on a quadratic lattice. *Physica* 27(12): 1209–1225.

Kolmogorov, V. 2009. Blossom V: A new implementation of a minimum cost perfect matching algorithm. In Mathematical Programming Computation (MPC), 1(1): 43-67.

Lee, W., Kara, L.B. and Stahovich, T. 2007. An efficient graph-based recognizer for hand-drawn symbols. In *EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*.

Lladós, J., Martí, E. and Villanueva, J. 2001. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(10): 1137–1143.

Lovett, A., Dehghani, M., and Forbus, K. 2007. Constructing spatial representations of variable detail for sketch recognition. In *Proceedings of AAAI 22*. Vancouver.

Lovett, A., Kandaswamy, S., McLure, M., and Forbus, K. 2012. Evaluating qualitative models of shape representation. In *Proceedings of the 26th International Workshop on Qualitative Reasoning*. Los Angeles, CA.

McLure, M., Friedman, S. & Forbus, K. 2010. Learning concepts from sketches via analogical generalization and near-misses. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society (CogSci)*.

McLure, M., Friedman, S., & Forbus, K. 2015. Extending Analogical Generalization with Near-Misses. In *Proceedings of AAAI 29*. Austin, TX.

McLure, M. D., Friedman, S. E., Lovett, A., and Forbus, K. D. 2011. Edge-cycles: A qualitative sketch representation to support recognition. In *Proceedings of the 25th International Workshop on Qualitative Reasoning*.

Mokhtarian, F., & Suomela, R. (1998). Robust image corner detection through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 20(12), 1376-1381

Schraudolph, N. N., and Kamenetsky, D. 2009. Efficient exact inference in planar Ising models. In *Advances in Neural Information Processing Systems 22*, 1417-1424.