ChemInk: A Natural Real-Time Recognition System for Chemical Drawings

Tom Y. Ouyang
MIT CSAIL
32 Vassar St. Room 239
Cambridge, MA 02139
ouyang@csail.mit.edu

Randall Davis
MIT CSAIL
32 Vassar St. Room 239
Cambridge, MA 02139
davis@csail.mit.edu

ABSTRACT
This paper describes a new sketch recognition framework for chemical structure drawings that combines multiple levels of rich visual features using a jointly trained conditional random field. This joint model of appearance at different levels of detail makes our framework less sensitive to noise and drawing variations, improving accuracy and robustness. The result is a recognizer that is better able to handle the wide range of drawing styles found in messy freehand sketches. Our system handles both graphics and text, producing a complete chemical structure. It works in real time, providing visual feedback about the recognition progress. On an existing dataset of chemical drawings our system achieved an accuracy rate of 97.4%, an improvement over the best reported results in literature. Our work presents a novel approach to corner detection that learns the features of corners in our domain, achieving an accuracy of over 99%. In our preliminary user study we found that the participants were on average over twice as fast at generating diagrams using our new system compared to using ChemDraw, a popular CAD-based tool for chemical diagrams, even though most of them had years of experience using ChemDraw and little or no experience using a Tablet PC. Participants in the study also rated our interface as both faster and easier to use than ChemDraw.

INTRODUCTION
Sketches and diagrams are an essential means of communicating information and structure in many different domains, and can be an important part of the early design process, helping people explore rough ideas and solutions in an informal environment. Despite the ubiquity of sketches, there is still a large gap between how people naturally interact with diagrams and how computers understand them today.

One field where sketches and diagrams are especially widely used is in chemistry. When chemists need to describe the structure of a compound to a colleague, they typically do so by drawing a diagram (e.g., Figure 1). When they need to convey the same structure to a computer, however, they must re-create the diagram using programs like ChemDraw, that still rely on a traditional point-click-and-drag style of interaction. While such programs offer many useful features and are used almost universally by chemists, these CAD-based systems simply do not provide the ease of use or speed of simply drawing on paper.

Our goal is to develop an intelligent sketch understanding system that provides a more natural way to specify chemical structures to a computer. To preserve the familiar experience of drawing on paper, our interface allows users to use the same set of standard chemical notations and symbols they used before. However, unlike real pen and paper, sketches created and interpreted using digital ink are recognized and understood by our system, converting them to a format that can be readily exported to other tasks such as structure analysis, visualization, and database/literature search.

This paper presents a new sketch recognition framework and applies it to hand drawn chemical diagrams. It combines multiple levels of rich visual features into a joint model using a discriminatively trained conditional random field. This hierarchical approach to recognition allows our framework to take advantage of information at multiple levels of detail, making it less sensitive to noise and drawing variations and significantly improving robustness and accuracy.

The key research contributions of this paper are:

- A symbol recognition architecture that combines vision based features at multiple scales and levels of classification.
- A new approach to corner detection that learns a domain-specific model of how to segment strokes based on training data.
- A discriminatively trained graphical model that unifies the predictions at each classification level and captures the relationships between symbols.
- A new clustering based algorithm for inferring the connectivity structure of sketched symbols.
- A real-time sketch recognition interface that has been evaluated by intended end-users and compared against the most

Submitted for review to IUI 2011.

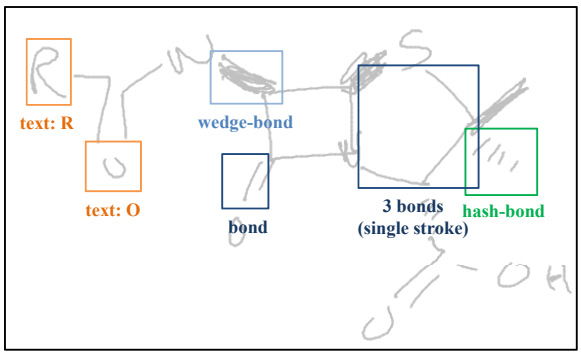

Figure 1. An example of a chemical drawing that our system is designed to recognize. The notation consists of element abbreviations (e.g., "N", "O"), group abbreviations (e.g., "R"), straight bonds, hash bonds, and wedge bonds. Wedge and hash bonds show the 3-D structure of a molecule (its stereochemistry): hash bonds angle down beneath the plane and wedge bonds angle up.

popular existing technique for chemical diagram authoring.

HIERARCHICAL SKETCH RECOGNITION

Our system interprets each sketch using three levels of classification : inkpoints, segments, and candidate symbols.

InkPoints

"Inkpoints" are data points sampled at a regular spatial interval from each stroke (Figure 2). The features we compute at inkpoints are similar to the local descriptors used in computer vision [3, 8], but are not located only at special interest points.

The features computed at each inkpoint were selected to provide a rich description of the surrounding patch of ink. Motivated by our previous work on sketched symbol recognition [12], the features employed by our recognizer are primarily based on visual appearance. This emphasis on visual properties makes our method less sensitive to stroke level differences, improving robustness and accuracy.

For each inkpoint, our method uses four sets of feature images to describe the local appearance at varying scales and orientations. Each set contains four individual images, each of which acts as orientation based filter (at 0, 45, 90, and 135 degrees). Each image captures only the ink that was drawn at the specified orientation. For example, in the 0-degree feature image a bright pixel indicates that the ink at that point is perfectly horizontal, a darker pixel indicates that the ink is somewhat horizontal, and a black pixel means that there is no ink or the ink is diagonal or vertical.

We designed these descriptors to be invariant to scale by normalizing the size of the ink patch based on L, an estimate of the scale of the sketch (described in the next section). We then compute two version of each feature image, one of which is rotated so the orientation of pen stroke is horizontal, thus making the feature image invariant to rotation. This dual representation is useful because it helps the system model both orientation-independent symbols like bonds

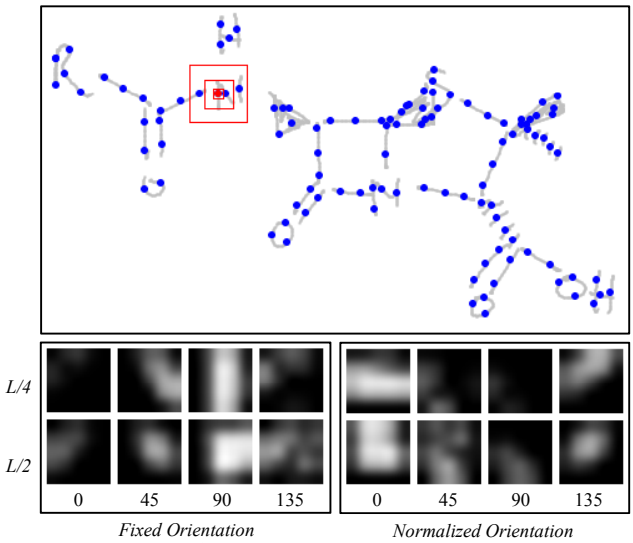

Figure 2. Shows the set of inkpoints (blue) that were extracted from a chemical diagram. For the inkpoint highlighted in red, the two red boxes show the size of the $L/4$ and $L/2$ feature regions. The figures below show the set of feature images generated for the same inkpoint. Note: the features images generated for segments are nearly identical except that they span a larger region.

as well as fixed orientation symbols like element and group abbreviations.

These ink features are rendered onto four 10x10 pixel feature images. We perform Gaussian smoothing on each image to improve robustness and reduce sensitivity to small distortions and noise. We downsample each image by a factor of 2 to a final size of 5x5 pixels. The result is a set of 400 feature values per inkpoint (25 pixels * 4 images * 2 scales * 2 orientations).

Segment Extraction

The second level of features is computed on the set of stroke segments extracted from the sketch (Figure 3). Segments are generated by dividing strokes at corner points. Chemists often draw straight bonds using a single long polygonal stroke (see Figure 1), relying on the reader to infer that they are actually drawing multiple individual bonds connected by implicit carbons.[1] Therefore, in our domain corners have a special meaning because they determine the breaks between straight bonds.

Corner detection is a well studied problem in sketch recognition. Previous approaches have explored looking at minima in curvature and time [15], temporal patterns in the stroke orientation [14], and alternative approximations to stroke curvature [22]. These methods often use hand coded thresholds to achieve good performance and customized heuristics to deal with common error cases. Prior work also focus primarily on finding well defined corners in isolated shapes, where there is a clear distinction between corners, curves,

[1] Carbons atoms are so common in chemistry that they are typically left out of the drawing and are assumed to be present anywhere that two bonds connect without a intermediate atom.

Feature	Description
Cost	The cost of removing the vertex, from Equation 1.
Diagonal	The diagonal length of the stroke's bounding box.
Ink Density	The length of the stroke divided by the diagonal length.
Max Distance	The distance to the farther of its two neighbor (p_{i-1} or p_{i+1}) normalized by the distance between the two neighbors.
Min Distance	The distance to the nearer of its two neighbor normalized by the distance between the two.
Sum Distance	The sum of the distances to the two neighbors normalized by the distance between the two.

Table 1. List of features for corner detection.

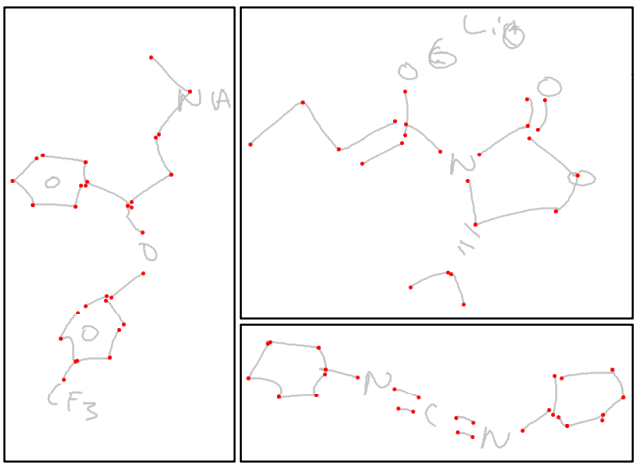

Figure 3. Examples of the result of our segment extraction algorithm on three chemical drawings. Segment endpoints are highlighted in red. Note that we only show corners from strokes that represent straight-bonds.

and straight lines. However, as we see in Figure 3, corners in real-world chemical drawings are often messy and unclear.

To deal with these challenges, we designed a novel corner detection algorithm that learns how to segment a stroke. Rather than forcing the developer to define thresholds and parameters beforehand, we train our corner detector from labeled sketch data. To the best of our knowledge this is the first trainable corner detector used in sketch recognition. This allows our detector to learn the specifics of what it means to be a corner for chemical diagrams, which may be different from what it means to be a corner in a different domain, or in a domain-independent shape recognizer.

Instead of starting by deciding which points are corners, our system instead repeatedly removes the point that is *least likely* to be a corner. The process stops when the system decides that all of the remaining points are likely to be corners. Specifically, our algorithm repeatedly discards the point p_i that introduces the smallest cost when removed, with cost defined as:

$$cost(p_i) = \sqrt{\text{mse}(\boldsymbol{s}_i; p_{i-1}, p_{i+1})}\ \text{dist}(p_i; p_{i-1}, p_{i+1}) \quad (1)$$

where $\boldsymbol{s}_i$ is the subset of points in the original stroke between point p_{i-1} and point p_{i+1} and $\text{mse}(\boldsymbol{s}_i; p_{i-1}, p_{i+1})$ is the mean squared error between the set $\boldsymbol{s}_i$ and the line segment formed by (p_{i-1}, p_{i+1}). The term $\text{dist}(p_i; p_{i-1}, p_{i+1})$ is the distance between p_i and the line segment formed by (p_{i-1}, p_{i+1}).

Instead of using a hard threshold to determine when to stop removing vertices, our system learns the likelihood of a vertex being a corner from training data. For each vertex elimination candidate p_i it extracts the set of features shown in Table 1. During classification, if the classifier decides that p_i is not a corner, it removes the vertex and continues to the next elimination candidate. If, on the other hand, it decides that it is a corner, the process stops and all remaining vertices are returned as corners.

One important feature of our approach is that in each iteration the system makes its decision based on the set of corner candidates that are still remaining, taking advantage of the partial solution generated so far. To illustrate this point, consider the case where there are two high curvature regions next to each other. One of them is an intended corner, but the other is just an accident due to noise (this is a common problem for corner detectors in general). When both high-curvature points are still in the list of candidates, neither one will have a large cost since removing one will simply cause the other to "pick up the slack." However, once one of them is eliminated, the cost of removing the remaining point becomes much larger. Of course in our implementation the other features from Table 1 will factor into the decision, so this is an illustrative but much simplified description.

At the end of the segment extraction stage the system records the length of the longest segment L (after excluding the top 5% as outliers). This value is used in subsequent stages as a rough estimate for the overall scale of the sketch.

Segment Features

We compute two types of features of segments. The first type consists of the same feature images that we use for ink-points, except in this case the image regions are centered at the midpoint of the segment and the width and height of the regions are set to L for the first scale and $2L$ for the second scale. The number of pixels in the feature images is the same as before, producing the same set of 400 feature values. We add to this the set of geometric properties listed in Table 3 as features.

Symbols

Symbols are the final unit of classification in our hierarchy. We define a symbol as a group of one or more seg-

Feature	Description
Length*	The length of the segment.
Ink Density	The length of the stroke region matching the segment divided by the length of the segment.
Segment Count	The total number of segments extracted from the parent stroke
Stroke Diagonal*	The diagonal length of the parent stroke's bounding box
Stroke Ink Density	The length of the parent stroke divided by the diagonal length of the parent stroke's bounding box

Table 2. List of geometric features for segment classification. (*) means we include two version of this feature, one normalized by L and the other unnormalized.

Feature	Description
Stroke Count	The number of strokes in the candidate (discrete valued feature, ceiling=10).
Segment Count	The number of segments in the candidate (discrete valued feature, ceiling=10).
Diagonal*	The diagonal length of the candidate's bounding box
Ink Density	The cumulative length of the strokes in the candidate divided by the diagonal length of the candidate.

Table 3. List of geometric features for candidate classification. (*) means we include two version of this feature, one normalized by L and the other unnormalized.

ments that represents a complete entity in the domain (e.g., bonds, element abbreviation, etc.). Our algorithm searches for candidate symbols (henceforth referred to as candidates) among groups of temporally or spatially contiguous strokes. It forms the set of temporally candidates by considering all possible sequences of up to $n = 8$ consecutively drawn strokes, evaluating how well each sequence matches the symbols in the domain. It forms the set of spatial candidates by combining groups of strokes that are close to each other. This process starts with all possible groups of size 2 (each stroke and its nearest neighbor) and successively expands each group by including the next nearest stroke (e.g., each stroke and its 2 nearest neighbors, then its 3 nearest neighbors, etc). This expansion ends when either the size of the group exceeds a spatial constraint or when the group contains more than 4 strokes. Note that this spatial grouping algorithm allows temporal gaps in each candidate, so symbols do not need to be drawn with consecutive strokes. An illustration of this process is shown in Figure 4.

The features we use for candidates encode the visual appearance of the candidate, based on our previous work in [13]. For each symbol we generate a set of five 20x20 feature images, four orientation filters and one "endpoint" image that captures the location of stroke endpoints. These feature images are computed only on stroke segments that belong to the candidate, unlike features in the other levels, which include all the ink in a local patch. In order to improve robustness to differences in aspect ratio, we stretch each symbol image so that it has the same standard deviation of ink in both the x and y axes. As before, we smooth and downsample each image by a factor of 2. An example of these feature images is shown in Figure 4. Notice that the "S" is stretched horizontally in the feature image so that the standard deviation of the ink in the x-axis is the same as that in the y-axis.

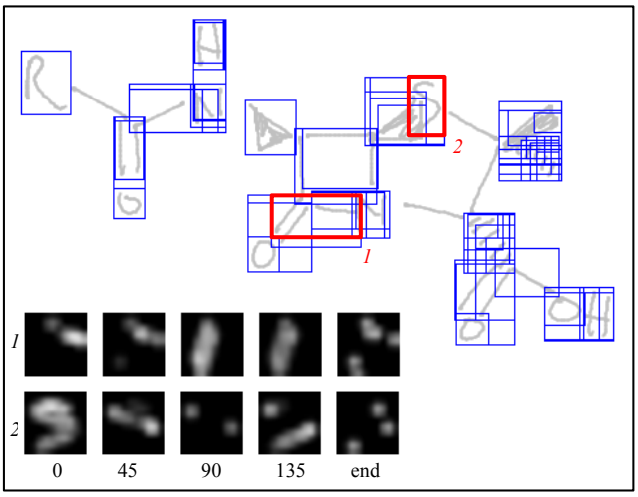

Figure 4. Shows the set of candidates extracted from a chemical diagram. The figures below show the feature images generated for the two candidates highlighted in red.

In addition to these five symbol feature images, we include another set of four images that characterize the ink in a patch around the candidate. These are identical to those used for segments, and here they are centered at the center of the candidate and have a region size of L. We also include the set of geometric properties listed in Table 3 as features.

Feature Image Templates

In the sections above we described how the system generates sets of feature images for each classification entity (i.e., inkpoints, segments, and candidates). However, we do not use the image values directly as features for classification. Instead, we compare the images against a set of stored templates taken from the training data and record the match distances to the nearest template neighbor in each class (we use the L2 distance). Next, we convert these distances into match scores (score = 1.0 - distance) and use as features both the label of the nearest neighbor and the best match distances to each class. For example, a segment whose nearest neighbor is a straight-bond might have the following features: (nearest="bond", bond=.1, wedge=.7, N=.3, O=.4, etc.).

As an optimization to improve running speed and reduce memory usage, we use Principal Component Analysis to reduce the dimensionality of the combined images for each entity to 256. For example, we would compress the 400 image values extracted from an inkpoint to a set of 256 principal

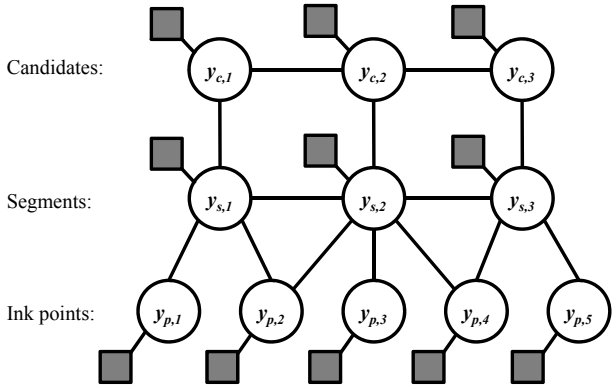

Figure 5. An illustration of our Conditional Random Field model. Circles represent label nodes (y), edges represent relationships, and dark boxes represent evidence nodes (x) that connect the labels to its corresponding features.

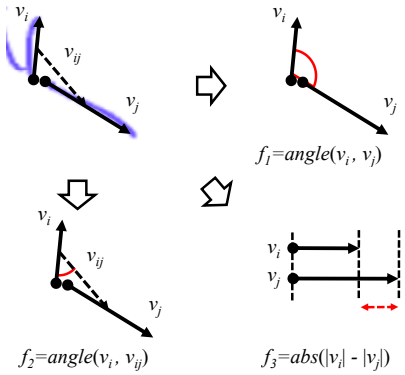

Figure 6. Spatial relationships: The three geometric features used in the pairwise context potential.

components. During the template matching process we calculate distances based on these principal components rather than the original image values.

JOINT GRAPHICAL MODEL CLASSIFIER

We propose a new model for sketch recognition based on Conditional Random Fields (CRF) that combines the features from the three levels in the classification hierarchy. A CRF can be seen as a probabilistic framework for capturing the statistical dependencies between the different entities we wish to model (i.e., the set of inkpoints, segments, and candidates).

An alternative way to accomplish this is by training an independent classifier at each level, then using some type of voting scheme to combine the predictions from each level. But this simple approach has two major disadvantages. First, by treating each layer independently it ignores any joint dependencies between features at different levels. Second, it requires the designer to specify a weighting scheme for each layer (e.g., deciding that the predictions in the candidate symbol layer should be worth 2x those in the inkpoint layer), something that could be very difficult to do.

Figure 5 shows an illustration of our CRF graph structure. The nodes in the bottom row represent labels for inkpoints (V_p), nodes in the middle row represent labels for segments (V_s). Inkpoint and segment nodes each have four possible labels: bond, hash, wedge, and text. The "text" label temporarily condenses the specific letters and abbreviations (e.g., "H", "O", "R") into a single label. Examples of these labels are shown in Figure 1. When classification is finished, any candidate symbol recognized as a "text" is converted to the letter identity of its nearest template match.

Nodes at the top level (V_c) represent candidate labels. However, rather than assign one node per candidate, we create one for each segment and then distribute the candidate labels to their corresponding segments. Essentially, assigning a node to a given candidate means the system thinks the node's segment is part of that symbol. For example, assume there are 2 candidates that contain segment 2: c_2 containing only segment 2 and $c_{1,2}$ containing segments 1 and 2. Under this framework, node $y_{c,2}$ will have 4 possible labels: c_2-bond, $c_{1,2}$-hash, $c_{1,2}$-wedge, and $c_{1,2}$-text (we assume that single segment candidates can only be straight bonds).

Our CRF model encodes the following four types of relationships:

Entity features to label mapping: This relationship determines the mapping between an entity's features and its label (e.g., straight-bond, text, etc). This is equivalent to a local classifier that classifies each entity independently of the others.

Each local potential function ϕ represents the entity features to label mapping, where $\mathbf{x}_i$ is the set of features for entity i and y_i is the label.

$$\phi_p(y_i, \mathbf{x}_i; \theta) = \sum_l f_{p,l}(y_i, \mathbf{x}_i)\theta_{p,l} \qquad (2)$$

Here f is a feature function defining the set of features for the given entity (e.g., the 256-valued PCA vector for inkpoints). There are three versions of this relationship: ϕ_p (shown above) for inkpoints, ϕ_s for segments, and ϕ_c for candidates (the same is true for f, $\mathbf{x}$, and θ). Note that ϕ is linear to the parameters θ, making the joint model in Equation 6 (below) log-linear.

In the case of candidate nodes, the feature function extracts features from the candidate corresponding to label y_i. For instance, if $y_{ci} = c_{1,2}$-wedge, then the features for the potential function would be those from candidate $c_{1,2}$.

Cross-level label consistency: This is a pairwise constraint specifying that predictions at each level be consistent predictions at other levels (e.g., an inkpoint, its parent segment, and its parent candidate all need to have the same label).

$$\psi(y_i, y_j) = \begin{cases} 0, & \text{if } y_i = y_j \\ -\text{inf}, & \text{otherwise} \end{cases} \qquad (3)$$

Candidate to candidate overlap consistency: This is a pairwise constraint that prevents the system from choosing conflicting interpretations for a single sketch region. For example, if the system predicts that segment 1 belongs to $c_{1,2,3}$, segments 2 and 3 also need to be assigned to $c_{1,2,3}$.

$$\psi_c(y_i, y_j) = \begin{cases} 0, & \text{if } y_i = y_j \text{ or} \\ & y_i \text{ does not overlap } y_j \\ -\text{inf}, & \text{otherwise} \end{cases} \quad (4)$$

Segment to segment spatial context: This pairwise relationship captures the contextual relations between pairs of segments and their respective labels. This relation enables our system to classify each segment jointly with its context, allowing neighboring interpretations to influence each other. Since segment predictions are tied to inkpoints and candidates, this joint classification propagates to the other two levels in the hierarchy.

$$\psi_s(y_i, y_j, \mathbf{x}_i, \mathbf{x}_j; \theta) = \sum_l f_{ss,l}(y_i, y_j, \mathbf{x}_i, \mathbf{x}_j)\theta_{ss,l} \quad (5)$$

Here the feature function $f_{ss,l}$ contains the 3 spatial relationships shown in Figure 6. The system discretizes f_1 and f_2 into bins of size $\pi/8$ and f_3 into bins of size $L/4$.

The joint probability function over the entire graph is given by:

$$\begin{aligned} \log P(\mathbf{y}|\mathbf{x}, \theta) = & \sum_{i \in V_p} \phi_p(y_i, \mathbf{x}_i; \theta) + \sum_{i,j \in E_{ps}} \psi(y_i, y_j) \\ & + \sum_{i \in V_s} \phi_s(y_i, \mathbf{x}_i; \theta) + \sum_{s_i, c_j \in E_{sc}} \psi(y_i, y_j) \\ & + \sum_{i \in V_c} \phi_c(y_i, \mathbf{x}_i; \theta) + \sum_{c_i, c_j \in E_{cc}} \psi_c(y_i, y_j) \\ & + \sum_{i,j \in E_{ss}} \psi_s(y_i, y_j, \mathbf{x}_i, \mathbf{x}_j; \theta) - \log(Z) \end{aligned} \quad (6)$$

where E_{ps} is the set of label consistency edges from inkpoints to segments, E_{sc} is the set of label consistency edges from segments to symbols, E_{cc} is the set of overlap consistency edges from candidates to candidates, and E_{ss} is the set of spatial context edges from segments to segments. Z is a normalization constant.

Inference and Parameter Estimation

During training the system estimates the parameters θ in a maximum likelihood framework. The goal is to find $\theta^* = \text{argmax} L(\theta)$, where, following the previous literature on CRF's [7], we define:

$$L(\theta) = \log P(\mathbf{y}|\mathbf{x}, \theta) - \frac{1}{2\sigma^2}||\theta||^2 \quad (7)$$

Here the second term is a regularization constraint on the norm of θ to help avoid overfitting. In our experiments we use a value of $\sigma = 10$.

We optimize $L(\theta)$ with a gradient ascent algorithm, calculating the gradient for each parameter $\frac{\delta}{\delta\theta_i} L(\theta)$. This process requires us to compute the marginals $P(y|x_i, \theta)$. Loops in the graph make exact inference intractable ; in response we calculate these marginals using Belief Propagation [23], an approximate inference algorithm. We employ a randomized message passing schedule and run the Max-Sum algorithm for up to 100 iterations.

For gradient-ascent algorithm we use L-BFGS ([9]), which has been applied successfully to other CRF problems in the past [17]. We use the same belief propagation algorithm during inference.

Real-Time Recognition

In our evaluations it takes about 1 second to classify a sketch on a 3.2ghz processor running in a single thread. While this is likely sufficient for real time recognition, we also took steps to make sure that our system is fast enough to run on slower Tablet PC's. First we implemented an incremental recognition model that updates the interpretation only of strokes and segments that have been modified or added since the last pass. Second, we made the most time consuming step of the process, generating features and template matches, parallel so that we could take advantage of multicore CPUs. In our user study the system, a 1.8ghz Tablet PC, was able to easily keep up with the user's drawings.

STRUCTURE GENERATION

After choosing the final set of predicted symbols, our system produces a connectivity pattern between the different symbols, to form a complete structure. An example is shown in Figure 7. The process of connecting symbols is based on a set of three pairwise distance metrics:

- Bond-element distance: The distance between a bond and an element is the distance from the bond endpoint to the nearest point in the element symbol. We impose an additional penalty if the bond does not point towards the element.
- Element-element distance: The distance between two letter symbols is defined as the minimum distance between the two at their closest point.
- Bond-bond distance: The distance between two bonds is defined as the distance between their respective endpoints. We impose a penalty if the bonds do not point towards each other (e.g., if one bond is pointed to the midpoint of the other) or if they are nearly parallel (parallel double bonds are technically connected to each other, but we are more interested in determining the two elements joined at their endpoints).

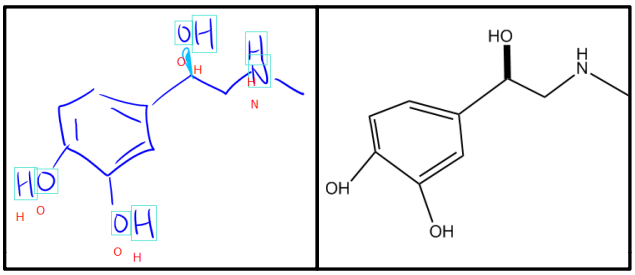

Figure 7. An illustration of the structure interpretation process: (left) an interpreted sketch with detected symbols highlighted and (right) the generated structure exported and rendered in ChemDraw.

We use an agglomerative clustering algorithm to generate the set of symbol connections. This algorithm iteratively merges the two nearest symbols or symbol clusters, using the maximum distance between the two groups of entities as the clustering metric (i.e., complete-link). We empirically set the threshold to stop clustering at $0.4L$. Since all symbols should be connected to at least one other symbol, the system reduces the distance metric by a factor of two if there are only two symbols in the cluster. This encourages it to connect isolated symbols first, effectively lowering the clustering threshold.

OFF-LINE EVALUATION

We recruited 10 participants who were familiar with organic chemistry and asked each of them to draw 12 real world organic compounds (e.g., Aspirin, Penicillin, Sildenafil, etc) on a Tablet PC. We performed a set of user-independent performance evaluations, testing our system on one user while using the examples from the other 9 as training data. By leaving out sketches from the same participant, this evaluation demonstrates how well our system would perform on a new user.

Because one goal of our research is to build a system that can handle the range of drawings styles found in natural, real world diagrams, the program used to collect these drawings behaved simply like a piece of paper, i.e., capturing the sketch but providing no recognition or feedback. This ensured that the system did not inadvertently provide guidance in how to draw.

Corner Detection

Our trainable corner detector was able to find corners in bond strokes 99.91% of the time, with a precision of 99.85% (these measurements include bond endpoints and single bond strokes) . In comparison, we tested a simpler version of the detector that uses a fixed threshold[2] on the cost metric from Equation 1. The results are shown in Table 4.

Symbol Detection

The results in Table 5 show that our system was able to accurately detect and classify 97.4% of the symbols from the sketches in the dataset. Our result also represents an improvement on the best previously reported accuracy of 97.1%[12]. While the increase in performance seems modest, it is worth noting that performance on the dataset was already very high and may be beginning to plateau. Despite this, our new approach was able to remove over 10% of the remaining errors.

[2]The threshold was chosen to produce similar values for the recall and precision.

Method	Recall	Precision
Trained detector	**0.9991**	**0.9985**
Fixed threshold	0.9879	0.9904

Table 4. Evaluation of the corner detection component of our system. We only count corners in strokes labeled as bonds and compare against the hand labeled ground truth.

Method	Recall	Precision
ChemInk (context)	**.974**	**.956**
ChemInk (no context)	.969	.951
O&D 2009 [12] (context)	.971	-
O&D 2009 [12] (no context)	.958	-

Table 5. Evaluation of the recognition accuracy of our system. The (no context) version does not employ spatial relationships between segments.

Note that for completeness we report precision as well as recall. However, for this task we believe that recall (the fraction of true symbols detected) is a more appropriate metric than precision (the fraction of detections that are true symbols) because, unlike in traditional object detection, there are no overlapping detections and every stroke is assigned to a symbol. Thus, a false positive always causes a false negative. Second, precision can be a less reliable metric because similar mistakes are not always counted equally. Misclassifying a 3-segment "H" as straight bonds, for instance, generates 3 false positives, while misclassifying it as a hash bond generates only one.

REAL-TIME COMPARATIVE EVALUATION

We conducted a second user study to evaluate the usability and speed of our system, asking a number of chemistry graduate students to draw a set of five pre-selected diagrams on a Tablet PC. While they were drawing, our recognition engine was running in real time and constantly providing feedback about the recognition progress by highlighting symbols detected so far. Users were asked to correct any errors the system made by simply erasing and redrawing the troubled region. Some of the collected diagrams are shown in Figure 10.

We compared our system to an existing popular chemistry authoring tool called ChemDraw, asking users to produce the same diagrams using its traditional mouse-and-keyboard interface. We recorded each session and measured the amount of time taken to construct the diagrams using both interfaces. We also asked the users for their opinions about how fast and easy it was to use each program.

Note: this study was conducted using an earlier version of our recognition engine, combining parts of our work in [12].

Demographics

We had a total of 9 participants, all with prior experience with chemistry, gained either through coursework only (1 user) or research only (1 user) or both (7 users). All of them had experience drawing chemical compounds on paper, reporting an average of 5.9 out of 7 (1=novice, 7=expert). Most also had extensive prior experience using ChemDraw, rating themselves on average a 5.0 out of 7. Conversely, most had little or no prior experience using Tablet PC's, rating themselves an average of 2.2 out of 7.

Quantitative Analysis

Figure 8 shows the average time that the users took to complete a diagram using both ChemInk and ChemDraw. It shows that they were on average more than twice as fast using our ChemInk interface, averaging 36 seconds per sketch, compared to ChemDraws average of 79 seconds. The difference in sketching time between the two interfaces is statistically significant (paired one-sided t-Test, $p < .05$). This was a surprising finding for us since many of the participants mentioned that they had years of experience using ChemDraw and use it daily in their research. This finding would also likely surprise those users who did not rate ChemInk as being significantly faster in the subsequent survey (Figure 9).

As Figure 8 shows, User 6 had an especially difficult time using ChemDraw, taking on average over 2 minutes per sketch. To make sure that the outlier was not biasing our results we repeated the analysis with User 6 omitted. The average time spent per sketch becomes 35 seconds for ChemInk and 61 seconds for ChemDraw, showing that our interface is still nearly twice as fast, and the difference is still statistically significant ($p < .05$).

Not surprisingly, the two users who had the lowest prior experience using ChemDraw (both rated themselves 1 out of 7) were also the slowest ChemDraw users (#6 and #8) and there was a highly negative correlation between reported ChemDraw experience and time spent per sketch (corr = -0.738, $p < .05$). This suggests that prior experience and training is very important in using ChemDraw proficiently. In contrast, all of the users were able to use ChemInk effectively regardless of prior experience with Tablets PC's (corr = 0.111, $p > .05$).

Qualitative Analysis

On average users rated ChemInk as being faster (6.2 vs. 4.4) and easier to use (6.2 vs. 2.2) than ChemDraw (both differences are statistically significant). In their comments about our system most of the users were very satisfied with the speed and performance, in many cases comparing it favorably against the traditional ChemDraw program. Some of the comments were: "Awesome!", "The program was very good at recognizing my drawing even though I have fairly messy handwriting...", and "In classroom setting, ChemDraw is too slow, whereas this is almost as fast as paper for taking notes."

Some also had suggestions for making it faster: "It would

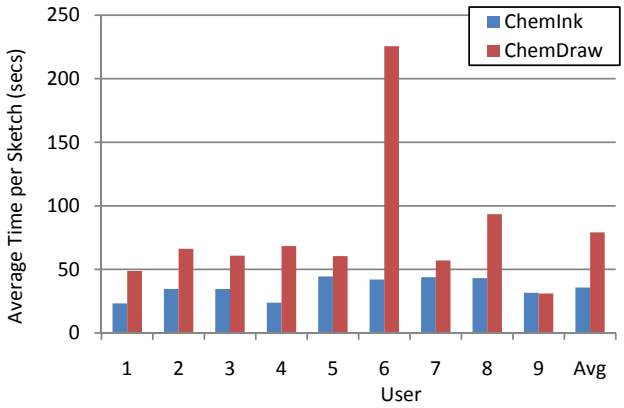

Figure 8. The average time taken by each of the study participants to draw a chemical diagram using ChemInk and ChemDraw.

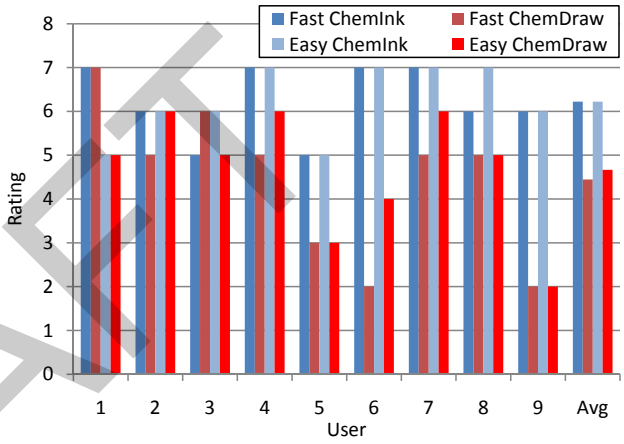

Figure 9. The ratings given by the study participants on how fast and how easy it was to use ChemInk and ChemDraw.

be even faster if you have predrawn sections that you could insert..." and "...if other letters or abbreviations were recognized, it would be even faster (for example recognizing that ph = [phenyl group])".

One user made the interesting comment that there is something fundamentally different about sketching vs. typing: "I like drawing structures by hand rather than in ChemDraw. Just like w/ typing hand-written notes are easier to remember / visualize / understand. Ability to sketch in 3D is very important too. great work guys! :)"

RELATED WORK

One popular approach to sketch recognition focuses on the relationships between geometric primitives like lines, arcs, and curves, specifying them either manually [1, 4, 5] or learning them from labeled data [16]. Recognition is then posed as a constraint satisfaction problem, as in [4, 5], or as an inference problem on a graphical model, as in [1, 16, 18, 20]. Szummer [20] proposed using a CRF to classify segments in a diagram, modeling the spatial relationships between neighboring segments. Their work is very different from ours in that recognition and feature extraction was only

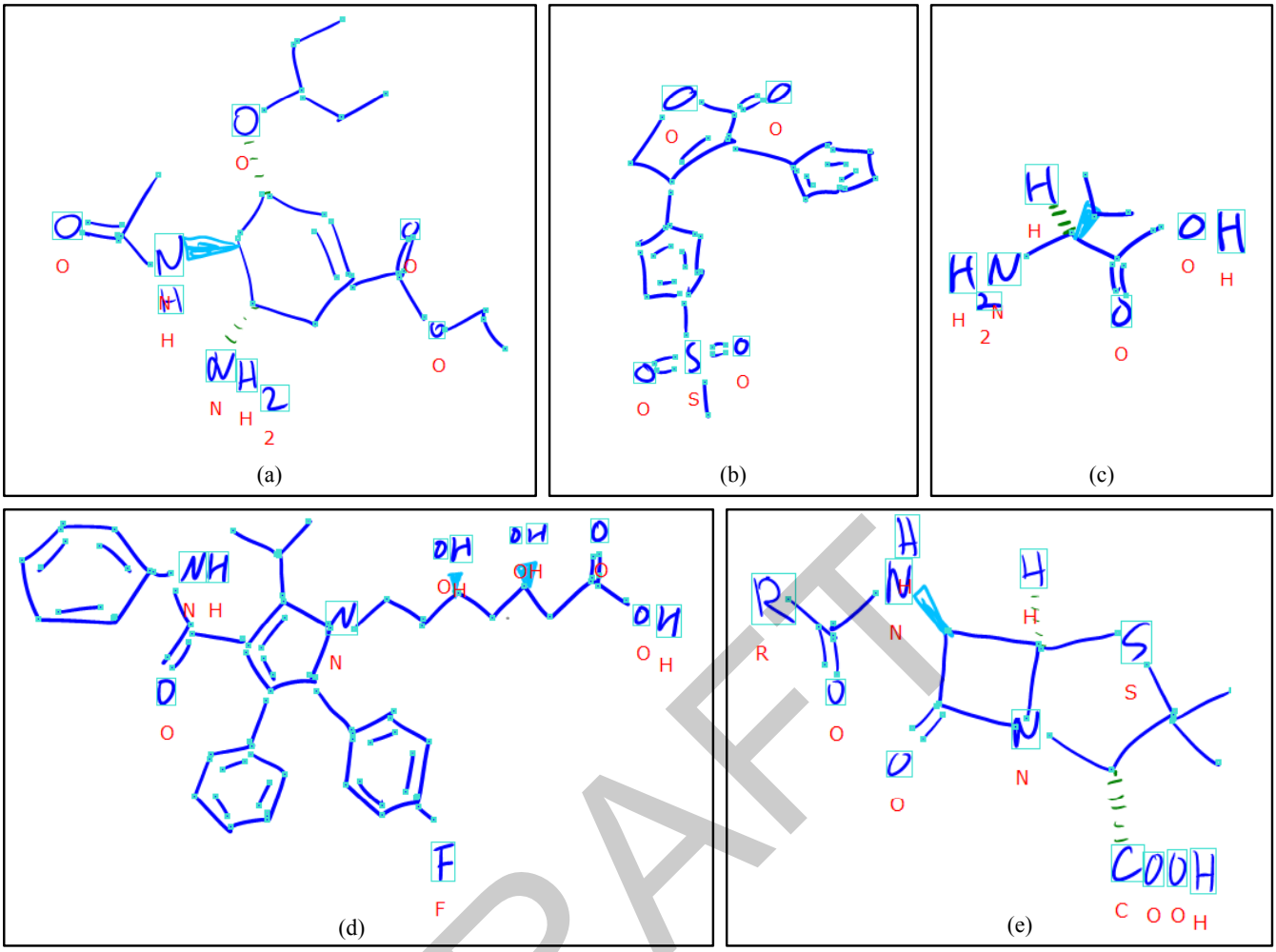

Figure 10. Examples of sketches collected from the real-time user study. The system's interpretation is highlighted as: boxed = text, light blue = wedge bond, green = hash bond, boxes = bond endpoints, blue = straight bond.

done at the segment level and there were only two possible labels.

Another group of related work focuses on the visual appearance of shapes and symbols. These include parts-based methods [10, 19], which learn a set of discriminative parts or patches for each symbol class, and template-based methods [6, 13], which compare the input symbol to a library of learned prototypes. The main advantage of vision-based approaches is their robustness to many of the drawing variations commonly found in real-world sketches, including artifacts like over-tracing and pen drag. However, these methods do not model the spatial relationships between neighboring shapes, relying solely on local appearance to classify a symbol.

There have also been efforts to recognize chemical sketches and diagrams. Tenneson and Becker [21] developed a sketch-based system that helps students visualize the three dimensional structure of an organic molecule. Unlike our system, theirs avoids many of the challenges in symbol detection by requiring that all symbols be drawn using a single stroke. It also does not handle implicit structure such as omitted carbon and hydrogen atoms. Casey et al. [2] developed a system for extracting chemical graphics from scanned documents, but that work focused on printed chemical diagrams rather than freehand drawings. Also, unlike our system their approach did not handle non-planar chemical notations such as wedge and hash bonds.

The work closest to this paper is our previous research on chemistry sketch recognition [11] and multi-domain symbol detection [12]. The work in this paper delivers better recognition performance and presents a new principled approach to combining multiple feature representations using a jointly trained CRF model. It also introduces a new learning based approach to corner detection that achieves nearly perfect results in our evaluation.

CONCLUSIONS

This paper introduced a new sketch recognition architecture for hand drawn chemical diagrams. It combines a rich visual description of the sketch at multiple levels with a joint clas-

sification model that captures the relationships between the features at each level. In our evaluation our system was able to correctly recognize 97.4% of the symbols in our dataset, improving on the best result previously published in literature. We also present a novel trainable corner detector that is able to correctly identify over 99% of the time. Our evaluation demonstrated that our new interface is over twice as fast as the existing method for authoring chemical diagrams, even for novice users who had little or no experience using a tablet.

Acknowledgements
This research was supported in part by a DHS Graduate Research Fellowship, a grant from Pfizer, Inc., and NSF Grant Number 0729422.

REFERENCES

1. C. Alvarado and R. Davis. Sketchread: A multi-domain sketch recognition engine. In *Proceedings of UIST*, 2004.
2. R. Casey, S. Boyer, P. Healey, A. Miller, B. Oudot, and K. Zilles. Optical recognition of chemical graphics. *Document Analysis and Recognition*, pages 627–631, Oct 1993 1993.
3. L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.
4. M. Gross. The electronic cocktail napkina computational environment for working with design diagrams. *Design Studies*, 17(1):53–69, 1996.
5. T. Hammond and R. Davis. Ladder: a language to describe drawing, display, and editing in sketch recognition. In *International Conference on Computer Graphics and Interactive Techniques*, 2006.
6. L. Kara and T. Stahovich. An image-based trainable symbol recognizer for sketch-based interfaces. *AAAI Fall Symposium: Making Pen-Based Interaction Intelligent and Natural*, 2004.
7. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 282–289. Citeseer, 2001.
8. D. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, volume 2, pages 1150–1157. Corfu, Greece, 1999.
9. J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2000.
10. M. Oltmans. *Envisioning Sketch Recognition: A Local Feature Based Approach to Recognizing Informal Sketches*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, May 2007.
11. T. Ouyang and R. Davis. Recognition of hand drawn chemical diagrams. In *Proceedings of AAAI*, 2007.
12. T. Ouyang and R. Davis. Learning from neighboring strokes: Combining appearance and context for multi-domain sketch recognition. In *Advances in Neural Information Processing Systems 22*, pages 1401–1409, 2009.
13. T. Y. Ouyang and R. Davis. A visual approach to sketched symbol recognition. In *Proceedings of the 2009 International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.
14. B. Paulson and T. Hammond. Paleosketch: accurate primitive sketch recognition and beautification. In *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 1–10, New York, NY, USA, 2008. ACM.
15. T. Sezgin and R. Davis. Sketch based interfaces: Early processing for sketch understanding. In *International Conference on Computer Graphics and Interactive Techniques*. ACM New York, NY, USA, 2006.
16. T. Sezgin and R. Davis. Sketch recognition in interspersed drawings using time-based graphical models. *Computers & Graphics*, 32(5):500–510, 2008.
17. F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003.
18. M. Shilman, H. Pasula, S. Russell, and R. Newton. Statistical visual language models for ink parsing. *AAAI Spring Symposium on Sketch Understanding*, 2002.
19. M. Shilman, P. Viola, and K. Chellapilla. Recognition and grouping of handwritten text in diagrams and equations. In *Frontiers in Handwriting Recognition*, 2004.
20. M. Szummer. Learning diagram parts with hidden random fields. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 1188–1193, 2005.
21. D. Tenneson and S. Becker. Chempad: Generating 3d molecules from 2d sketches, 2005.
22. Y. Xiong and J. LaViola Jr. Revisiting ShortStraw: improving corner finding in sketch-based interfaces. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 101–108. ACM, 2009.
23. J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, pages 239–236, 2003.