Pixel Clustering for Skeletal Extraction in Digit Images

Mattias Fredriksson
Blekinge Inst. Technology
Karlskrona, Sweden

***Abstract*—Extracting features from digit images is complicated, and there are different approaches for interpreting and constructing feature representations. A common approach for extracting representations of the underlying shape consist of constructing a skeletal representation of the shape by reducing the image to a set of thin lines.**

A new approach to the problem of extracting skeletal representations is investigated. By clustering pixels within the image, a graph representation of the skeleton is formed from the clusters midpoints. The extracted skeleton is evaluated through a geometrical feature representation of the graph, where three different machine learning algorithms are applied on feature data generated from the MNIST data set. The results from the test data indicate that the method generates less accurate, but consistent representations of the image shape.

I. Introduction

Feature extraction is an integral part of any machine learning task. In the area of digit recognition multiple methods exist for extracting feature data from an image [1]. Different approaches to the problem vary in complexity, with simpler methods such as projection histograms counting the number of pixels in each image's columns and rows and using the sums as features. Recent development have led to increasingly elaborate methods with techniques utilizing similar ideas to how human interprets digits, e.g. using perceptual shape primitives to define the features [2]. Presented approach to digit recognition does not focus on constructing elaborate feature descriptions of the digit, instead the project approaches the problem of extracting skeletal representations of the image.

Extraction of a skeleton is a subject that has been studied extensively, but a new approach is proposed that differentiates from existing methods. Specifically, the question of *if it is possible to approach skeletal extraction by clustering image pixels?* is investigated.

By clustering the pixels within the image graph, a reduced graph is created from the initial shape of the object. Relevance for the new approach is that it locally adjusts to the image and outputs a reduced graph representation rather than a subset of pixels that require further processing. Pixels in the initial image are also associated with a single vertex in the final graph, which allows the result to be analyzed. Graphs produced by other methods, such as iterative thinning, can be harder to analyze as a pixels influence on the outcome can be complicated to discern.

To test the validity of the extraction method, an evaluation is performed using a simple approach to extract features from the skeletal representation. By further processing the graph, a geometrical representation is created that generalizes over the initial shape, extracted features are then used to evaluate if the skeleton consistently represents the initial shape in the image.

II. Background and Related work

Skeletal extraction, also denoted as 'thinning', is a process of reducing the image to a subset of pixels defined as the skeleton, representing the initial shape of the object as thin lines. A majority of the methods used come in the form of iterative morphological methods [3], thinning the image by iteratively removing the outer pixel layer according to a set of rules defined as kernel masks. The process is complete when only the subset of skeleton pixels remain [4].

Other common thinning algorithms include contour-based extraction methods, utilizing the image's contour to identify lines in-between the contour edges. Identification of the skeleton from a contour is approached differently, but multiple methods uses triangulation on the contour shape, creating the skeleton from the midpoints [3].

A. Medial Axis Transform

Topological skeleton has multiple definitions, but a common definition introduced by Blum (and improved upon in [5]) is the medial axis transform where the skeleton is defined by the medial axis. The planar shape is then defined by locus of center points from a set of maximum discs $D_m \subset D$, where D represents the set of all possible discs that fit inside the planar shape, and D_m represents the maximal discs which do not fit inside any other possible disc in D. The center points of the maximal disks then form a 1-dimensional graph, from where the shape can be reconstructed using the radii of the maximal disks associated with each point on the curved edges.

III. Method

The experiment was conducted by applying the skeletal extraction method on the MNIST [6] dataset. Geometrical features were extracted from the skeletal graph and evaluated by applying a kNN, Decision tree and SVC classification model on the test data. Classification algorithms was chosen based on suitability for multi-class classification and selected if proven to be suitable candidates in initial cross validation tests performed on the training data. Algorithm implementations used in the evaluation was provided through the sci-kit learn platform [7] and the algorithms for skeletal and feature extraction is explained in the following sections.

Due to time constraints, the evaluation of the skeletal extraction algorithm was limited to a simple classification test with little to no feature analysis or parameter tuning. No state-of-the-art feature extraction method was implemented nor applied for the purpose of generating comparable results with other digit classification methods.

IV. Skeletal Extraction Algorithm

A large part of the work is devoted to the algorithm for skeletal extraction, constructing a graph representation from the image using vertices in the form of center points of connected pixel clusters. The process of can be summarized in the following steps:

1) Graph construction, creation of connected pixel clusters from the binary image.
2) Reduction of the graph, removal of any cycles consisting of four vertices or less.
3) Culling, removing graph vertices that represent pixel clusters of small or no importance.
4) Line segmentation, merging vertices representing pixel clusters with small difference to the approximated graph direction.

Overview of results from the different steps is visible in Figure 1, 2 and details of each step are explained in the following subsections.

A. Graph Construction

First step in the process begins with reducing the image to a binary format differentiating the object from the background. The process amounts to a binary classification where each pixel in the image either are defined to represent the object or not. Binarization techniques suitable for separating an image depends on the image data and the type of degradation that has occurred [8].

The MNIST dataset [6] used for the purpose of evaluation consist of gray scale images constructed from processed binary images. Degradation over the image can therefore be assumed to be consistent and there exists a clear threshold separating foreground from background pixels. Otsu's method [9] selects the threshold maximizing between-class variance and was used for separating the gray scale image data into a binary representation.

Once binarization is performed, the image is converted into a graph constructed from a set of small pixel clusters representing the vertices V_g in the initial graph $G = (V_g, E_g)$. Clusters are created by dividing the image into a set of 2×2 kernels, with every four pixels further separated in disjoint sets depending on the adjacency between the pixels within the kernel. Adjacency here is defined in the same way as 4 adjacent pixels in iterative thinning methods [4], and only the closest pixels on the image's x- or y-axis is considered to be adjacent. Using the small 2×2 kernel, only two foreground pixels in a diagonal will be separated in two disjoint sets.

After initial sets are created within the kernels, adjacent kernels are connected by the same principle. If there exists one pixel that is 4 adjacent to a pixel in another kernel, an edge is created between the two related sets. The graph generated from the step is visible to the left in Figure 1, the edges (black lines) connect vertices constructed from the color-coded pixel clusters.

B. Graph Simplification

A simple graph, in graph theory, refers to a graph without any duplicate edges (connecting the same two vertices) and loops (edge connecting the same vertex). Here simplification is referred to as a reduction of the graph, removing all cycles in the graph that can be defined as a face. A face is then defined as an induced subgraph $S = (V_s, E_s)$ of G, where the subset $V_s \subseteq V_g$ is a set of at most four vertices, and there exists at minimum an equal number of edges in the subset as there are vertices $|V_s| \leq |E_s|$. The induced subgraphs consistent with the definition are cycles in G, with either three or four vertices.

By repeatedly reducing the number of faces, the graph will be reduced to a set of junction vertices and vertices defining line segments. Junctions are then vertices with a degree greater then 2 and the set consists of any vertex with three or more connected edges. The set of connected vertices with degree ≤ 2 then form the line segments of the skeleton representation. Pseudo code for the process is presented in Algorithm 1 and it performs the reduction of the graph by repeatedly merging a pair of vertices $uv \in V_g$, where uv is the two closest vertices considered within the graph. For two vertices to be considered, they must be part of the same subgraph $S_f \in F$, where F contains all subgraphs that exist in G that are consistent with the definition of a face. The distance used for finding the closest vertex pair is measured using the Euclidean norm.

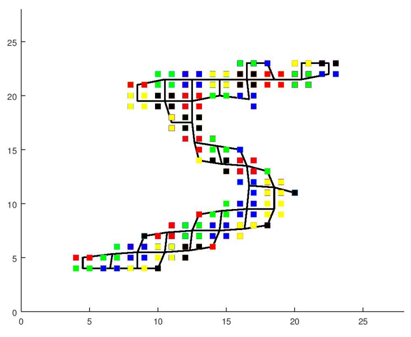
The center point of a new vertex (merged or otherwise) is calculated using the average of the pixel coordinates p_i (the mass center $\overline{p}$) of pixels associated with the vertex. The sums of the average can be separated as in (1) allowing efficient evaluation when merging uv. The result of the reduction process is visible in Figure 1.

$$\overline{p} = \frac{\sum_{i=1}^{m+n} p_i}{m+n} = \frac{\sum_{i=1}^{m} p_{1,i} + \sum_{j=1}^{n} p_{2,j}}{m+n} \Leftrightarrow p = \{x | x \in p_1 \vee x \in p_2\} \tag{1}$$

The merging process is greedy and uses a heuristic approach for finding the graph reduction representing line segments of the digit, while efficient, the definition of a face presents an issue with the shape representation. There can exist a cycle in the graph containing four connected vertices and still represents a set of valid line segments e.g. rectangular shapes, but the issue is minimal if initial clusters are small.

C. Graph Culling

Due to the properties of the original graph, vertices exist in the simplified graph that are dependent of the alignment


(a) Step I

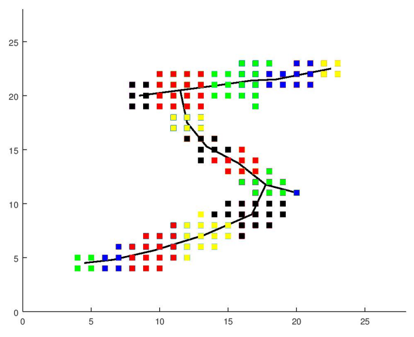
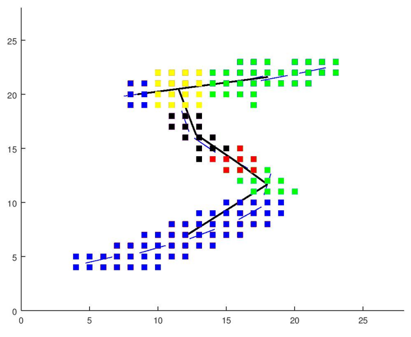

(b) Step II

Fig. 1. The result of the kernel pixel clustering to the left, the result of the simplification algorithm to the right.


(a) Final I

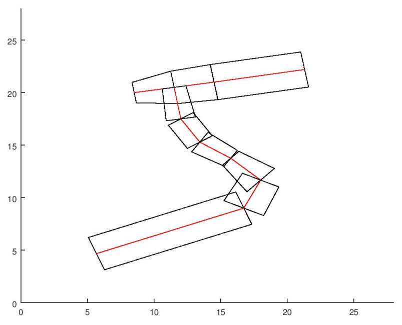

(b) Final II

Fig. 2. Final processing result after the fourth stage, to the left is the connected pixel clusters associated with the graph after the line segmentation is performed, the estimated graph direction is visible as thin blue lines positioned in the mass centers of the initial vertices. On the right, the reconstructed final graph where the ends of the line segments are connected (represented by the pixel clusters in the left image).

Algorithm 1 Simplify Graph

```
Input: Graph $G = (V_g, E_g)$
Output: Processed graph $G'$
 1: Merge face vertices in order of smallest norm first :
 2: while hasFace(G) do
 3:     Find related vertices that are part of a face :
 4:     $l = \emptyset$
 5:     for $uv \in V_g$ do
 6:         if $uv \in$ face(G) then
 7:             $l \leftarrow l + uv$
 8:         end if
 9:     end for
10:     Find the smallest distance between two vertices and
        merge them, reducing the graph :
11:     $uv \leftarrow$ min($l$)
12:     G $\leftarrow$ merge($uv$, G)
13: end while
14: return $G$
```

of the kernels during the creation of the initial graph clusters. The result is a single vertex formed from a redundant set of convex contour pixels ignored in the simplification process in Algorithm 1 since it is connected to a single neighbor. Issues with redundant clusters formed in the shapes contour are visible to the right in Figure 1.

Culling is then used to identify these vertices and merge them with their connected neighbor if one exists. Culling is necessary for reducing the noise in the graph, and the problem is common within skeletal extraction algorithms as it relates to the issue of spurious projection [3]. The pseudo code for the culling procedure can be found in Algorithm 2 which also accounts for noise in the image by removing unconnected vertices associated with few pixels.

D. Line Segmentation

The last processing stage reduces the number of vertices in the graph by finding an approximated direction $\mathbf{d_i}$ of the graph at each vertex position coordinate $\mathbf{p_i}$ using SVD - Singular Value Decomposition. SVD is used to calculate the least square fit of the pixel points associated with a vertex and all of its connected neighbors. The result is used to define a line with the formula $\mathbf{p_i} + t\mathbf{d_i}$, the lines are shown in the left image of Figure 2.

Using error quadrics [10], the error of joining two vertices can be defined as the distance (or residual) between two

Algorithm 2 Cull Graph

```
Input: Graph $G = (V_g, E_g)$
Output: Processed graph $G'$
    Find any vertex that should be removed :
 1: for $v \in V_g$ do
 2:     if pixels($v$) $< 5$ then
 3:         if edges($v$) $= 0$ then
 4:             $V_g \leftarrow V_g - v$
 5:         else if edges($v$) $= 1$ then
 6:             $uv \leftarrow$ edge($v$)
 7:             G $\leftarrow$ merge($uv$, G)
 8:         end if
 9:     end if
10: end for
11: return $G$
```

connected vertices $\mathbf{v_1}$ and $\mathbf{v_2}$, and the lines associated with each vertex (2). By finding the smallest error of any two connected vertices in the graph, the related edge is contracted, and the process can be repeated until the error of all edges are greater than a given threshold.

$$e = |\begin{pmatrix} -\mathbf{d_{1,y}} \\ \mathbf{d_{1,x}} \end{pmatrix} \bullet (\mathbf{p_2} - \mathbf{p_1})| + |\begin{pmatrix} -\mathbf{d_{2,y}} \\ \mathbf{d_{2,x}} \end{pmatrix} \bullet (\mathbf{p_1} - \mathbf{p_2})| \quad (2)$$

To retain the original shape of the graph as edges are contracted, the line equations of each original vertex is stored in the union, and the error between two remaining vertices $\mathbf{v_j}$ and $\mathbf{v_i}$ is calculated as the sum of the length of the two residual vectors (3). The residual vectors consists of the distances from the point $\mathbf{p_j}$ to each of the original line equations of the neighboring vertex $\mathbf{v_i}$, and the total equation to find the error for $\mathbf{p_j}$, $e_j = ||r_j||$ is then (4).

$$e = ||r_i|| + ||r_j|| \quad (3)$$

$$||r_j|| = \sqrt{\sum_{i=1}^{n} (\begin{pmatrix} -\mathbf{d_{i,y}} \\ \mathbf{d_{i,x}} \end{pmatrix} \bullet (\mathbf{p_j} - \mathbf{p_i}))^2} \quad (4)$$

V. Feature Extraction

Features describe the objects in the domain regarding the task we want to solve. If there is no correlation between the features used and the objects described, learning from the data assembled is an impossible task. Features can thus be described as the workhorse of machine learning [11].

Selecting the right feature descriptive of the object is an integral part of all classification problems, but as in the case of digit recognition, extracting the features themselves is not a trivial problem. In parts described earlier, our approach to reduce the complexity of the problem is explained. However, to construct features that correlates between different objects, a last processing step is required on the output from our extraction algorithm visible to the right in Figure 2.

The feature selection used is simple in relation to the previous processing steps, remaining edges in the graph describe a line segment where the following three parameters are used for feature description:

1) Angle difference from the direction of the previous connected segment.
2) Length of the segment.
3) Width of the segment.

To get a feature vector of equal length, smaller graphs are increased by a few empty segments, padding the vector to a limit defined for our solution. On the other hand, if the graph consists of too many segments, the fourth step is repeated with increasing threshold. Also note that, for the first segment, the angle difference from the unit vector $\mathbf{e_x}$ is used. This means that all the features are rotational invariant, except from the initial angle and the segment ordering associated with selection of the first segment, influenced by the rotation of the image.

Ordering the segment is important since it ensures that the graph is evaluated in a known order between images, limiting problems associated with trying to find correlation between unrelated line segments. To address the problem, a final processing step is performed where the edges in all disjoint subgraphs (and hence the subgraphs themselves) are ordered according to a simple scheme based on a principle of ordering occurring in contour segments.

The ordering scheme is based on methods in contour extraction [1], the contour can be traced in a clockwise or counterclockwise order. The behavior is utilized as segments are traced from the initial edge in a counterclockwise order, recursively ordering the rightmost edge connected to the previous edge in the traversed direction. The process is recursively performed for all connected edges until all edges of the vertex is ordered.

To get a starting point for the process, either the vertex of degree 1 (with only one neighbor) placed lowest on the y axis is used or, if no such vertex exists, the same scheme is applied to all vertices of any degree. The process is repeated for all subgraph until all edges are sorted.

VI. Results & Analysis

Metrics for the model created from the training data and applied on the testing data is presented in Table I. Graph representations of the feature extraction can be found in Figure 3. The result of the classification task is positive but not comparable with other solutions such as [2]. On the other hand, the results indicate that creating a skeletal representation from pixel clusters can prove to be an efficient method applicable if the data is assumed to primarily consist of curved line segments. With complicated shapes such as digits, other methods is better suited if the approach is not improved, as the current extraction method lose accurate representation in the joints (see Figure 4). Improvements in robustness and efficiency of the skeletal extraction method, and with a better approach for feature generation, classification results could be improved. Evaluating the approach in relation to other thinning methods could then show if the approach is relevant.

VII. Discussion

The skeletal extraction method creates a rough approximation of the skeleton in relation to the medial axis definition. Even if edges and joints in curved segments within the shape will deviate due to the linear approximation in the representation, if the assumption is made that the image is constructed from connected curved line segments with small deviations in width, errors can be minimal.

Assumption that shapes contain clearly distinguishable lines forms the basis for reasoning behind the implementation and is inherit from the clustering performed on the image. Formed clusters will deviate from the medial axis, but they will follow limitations defined by the radius r of the maximal discs D_m that defines the width of the curved line at the disk center. By continuously merging the closest clusters, the process ensures that the distance between clusters along the length of the line segment is limited by a factor of the width, and the radius r of the maximal disks.

If shapes are distorted there are cases where clusters extend beyond the width of a segment, due to discreet representation and kernel alignments, causing clusters to be malformed. Malformed clusters (see Figure 4) are created in joints where the space between two connected line segments are small, as graph edges can persist between the lines connected at the joint, and later joined when clusters are formed. Treating this as an edge case by splitting pixels associated with the edge would generate clusters that closer follows the constraints of the maximal disks.

Evaluating the deviation from the original image could also be of importance, defining how accurate the representation is from the medial axis representation of the skeleton as defined by Blum [5]. The metric could be used to determine relevance in comparison to other methods (such as iterative thinning).

VIII. Conclussion

Measured performance of the feature extraction method produced positive results but are not on par with state of the art solutions such as [2]. While underwhelming, the results indicate that clustering is a viable option for skeletal extraction if the data set consists of clearly separated line segments. Reaching reasonable results on the MNIST data set, clustering could be evaluated further and improved.

Improving performance of both the skeletal and feature extraction methods is possible. However, if accuracy in the generated skeleton is required for the feature extraction, other methods with a different approach would be better suited for the learning task, i.e. signed sequential Euclidean distance - SSED maps [12].

TABLE I
Classification Results

Classifier	Accurracy	F-measure	Training Time
KN-Neighbor	0.9059	0.9059	18.446
Decision Tree	0.8115	0.8115	4.794
SVC	0.9301	0.9301	113.564

References

[1] Ø.D. Trier, A.K. Jain and T. Taxt. ”Feature Extraction Methods for Character Recognition - A Survey”. *Pattern Recognition*, Vol. 29, Iss. 4, pp. 641-662, April 1996.

[2] K.S. Dash, N.B. Puhan and G. Panda. ”Unconstrained handwritten digit recognition using perceptual shape primitives”. *Pattern Analysis and Applications*, [online], 2016. Available: https://doi.org/10.1007/s10044-016-0586-3.

[3] J. Parker, *Algorithms for Image Processing and Computer Vision*, 2nd ed. Indianapolis, US: Wiley Publishing Inc, 2011, pp 209-247.

[4] L. Lam, S.W. Lee and C.Y. Suen ”Thinning Methodologies - A Comprehensive Survey”. *IEEE TPAMI*, Vol. 14, Iss. 9, pp. 869-885, September 1992.

[5] H. Blum and R.N. Nagel, ”Shape description using weighted symmetric axis features”. *Pattern Recognition*, Vol. 10, Iss. 3, pp. 167-180, January 1978.

[6] Y. LeCun, C. Cortes and C. Burges, ”MNIST Database”, 1999. [Online]. Available: http://yann.lecun.com/exdb/mnist/. Accessed: Jan. 19, 2018.

[7] F. Pedregosa et al., ”Scikit-learn: Machine Learning in Python”. *Journal of Machine Learning Research*, Vol. 12, pp. 2825-2830, 2011.

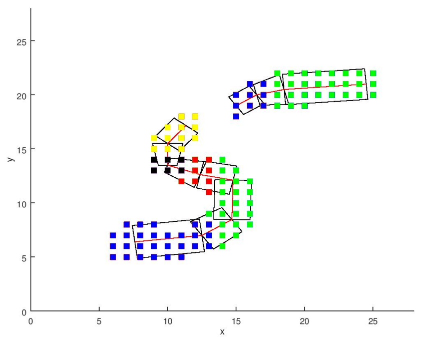
[8] Jyotsna, S. Chauhan, E. Sharma and A. Doegar, ”Binarization techniques for degraded document images - A review,” 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2016, pp. 163-166.

[9] O. Nobuyuki ”A Threshold Selection Method from Gray-Level Histograms”. *IEEE TSMC*, Vol. 9, Iss. 1, pp. 62-66, January 1979.

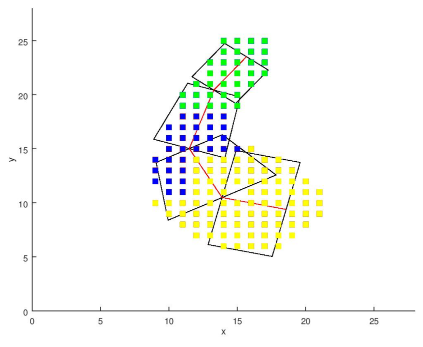
[10] M. Garland and P.S. Heckbert, ”Surface Simplication Using Quadric Error Metrics”. In *SIGGRAPH ’97 Proc.*, pp. 209-216, Aug. 1997.

[11] P. Flach, *Machine Learning: The Art and Sciece of Algorithms that Make Sense of Data*. Cambridge, UK: Cambridge University Press, 2012.

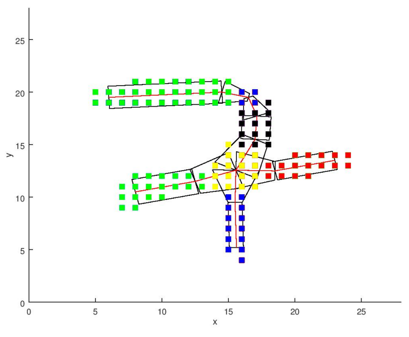
[12] W. Choi, K. Lam and W. Siu ”Extraction ofthe Euclidean skeleton based on a connectivity criterion”. *Pattern Recognition*, Vol. 36, Iss. 3, pp. 721-729, March 2003.



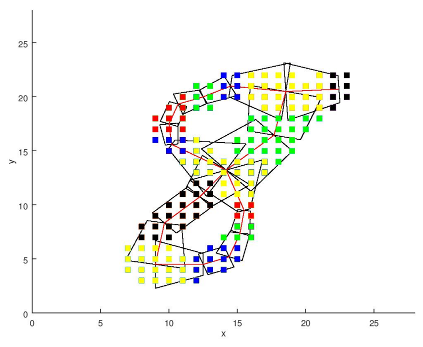
(a) Digit: 5



(b) Digit: 6

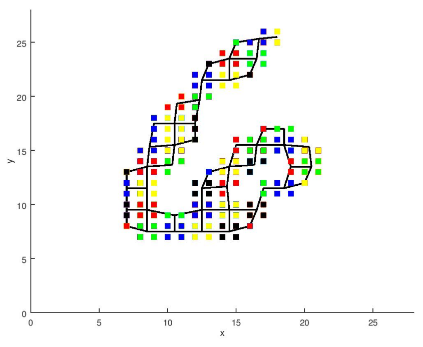


(c) Digit: 7

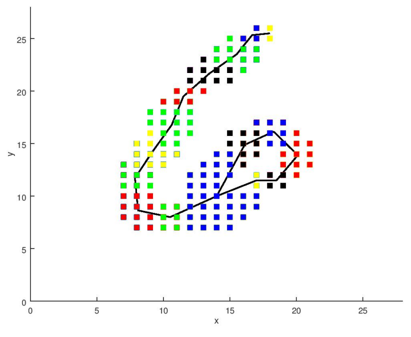


(d) Digit: 8

Fig. 3. Generated representations with all steps applied.



(a) Kernel aligned pixel clusters



(b) Clusters after faces/loops in the graph are merged

Fig. 4. Malformed cluster in the joint between the closed loop and the curved top segment (large blue area in the right image).