A Topological Graph-Based Representation for Denoising Low Quality Binary Images

Catherine Potts
Montana State University
Bozeman, MT, USA
catherine.potts@montana.edu

Liping Yang, Diane Oyen, Brendt Wohlberg
Los Alamos National Laboratory
Los Alamos, NM, USA
{liping.yang,doyen,brendt}@lanl.gov

Abstract

Scanned images of patent or historical documents often contain localized zigzag noise introduced by the digitizing process; yet when viewed as a whole image, global structures are apparent to humans, but not to machines. Existing denoising methods work well for natural images, but not for binary diagram images, which makes feature extraction difficult for computer vision and machine learning methods and algorithms. We propose a topological graph-based representation to tackle this denoising problem. The graph representation emphasizes the shapes and topology of diagram images, making it ideal for use in machine learning applications such as classification and matching of scientific diagram images. Our approach and algorithms provide essential structure and lay important foundation for computer vision such as scene graph-based applications, because topological relations and spatial arrangement among objects in images are captured and stored in our skeleton graph. In addition, while the parameters for almost all pixel-based methods are not adaptive, our method is robust in that it only requires one parameter and it is adaptive. Experimental comparisons with existing methods show the effectiveness of our approach.

1. Introduction

Image denoising is an important pre-processing technique in computer vision and image processing. Scanned images of patent or historical documents often contain localized zigzag noise (see Figure 2 (a) for an example of zigzag noise) introduced during the digitizing process that makes the automatic extraction of global structures substantially more difficult. Many images from scanned scientific documents only contain shape and topology information comprised of lines and curves; whereas, natural images have much richer features, such as intensity, color, and texture. Existing denoising methods (detailed in Section 2) work well for natural images, but not for binary diagram images (because those denoising methods are designed for natural images), which makes feature extraction for binary types of images difficult for computer vision and machine learning methods and algorithms.

Traditional computer vision feature descriptors, such as SIFT [24] and SURF [3], are often based on local scale. Recent work in computer vision and image processing has shown that topological graph-based methods [42, 43] have intuitive and clear advantages for image analysis [42, 41].

In this paper, we propose a topological graph-based image representation that combines both local and global features for denoising low quality binary images. Specifically, our method combines the power of topological graphs and skeletons to generate a hierarchical skeleton graph representation to tackle the denoising challenges. A skeleton is a central line representation of an object in an image obtained via thinning [22]. The skeleton emphasizes topological and geometrical properties of shapes [42]. In our approach, the skeleton serves as the essential bridge from image representation to topological graph representation.

Our approach and algorithms provide essential structure and lay an important foundation for computer vision challenges such as scene graph-based applications, because topological relations and spatial arrangement among objects in images are captured and stored in our skeleton graph. While the parameters for most pixel-based methods are not adaptive, our method is robust in that it only requires one parameter, and is adaptive. We compare our approach with six commonly used denoising methods (specifically, five designed for natural

2473-9944/19/$31.00 ©2019 IEEE
DOI 10.1109/ICCVW.2019.00222

images and one for document images); the experimental results show the effectiveness of our method and algorithms based on topological graphs and computational geometry (e.g. convex hull).

Here, we provide a road map to the rest of the paper. Section 2 covers related work, including a brief review of commonly used denoising methods for natural and document images, and the topological-based approaches that our method is built on. Section 3 is the core of the paper, focusing on our methods and algorithms. Specifically, Section 3.1 provides the workflow of our method (Figure 1), and Section 3.2 data structures, which will be used in our developed algorithms detailed in Section 3.3. In Section 4, we present our experiments and results. The paper concludes in Section 5 with discussion of limitations and future work, including potential applications.

For readability, we provide a list of abbreviations used in Appendix A. To provide background for readers, particularly those who are new to graph theory and computational geometry, in Appendix B, we introduce graph theory terms we used throughout the paper.

2. Related work

Most denoising methods are designed for the major types of noise present in natural images that follow different distributions (e.g., Gaussian noise, salt and pepper noise, speckle noise, and Brownian noise) [29, 38, 5, 20].

Here we briefly review some commonly used denoising methods, some of which we compare in our experiments (Section 4). Smoothing filters, such as Gaussian, median, and bilateral filters, are often the commonly used techniques to denoise images. *Gaussian filtering* is a linear operation. However, it does not necessarily preserve edges, as the Gaussian filter computes a weighted average of pixel values in the neighborhood, where the weights decrease with distance from the neighborhood center [36]; that is, the value of the variance governs the degree of smoothing. *Median filtering* is a non-linear operation [2, 17]. Unlike linear filters, as its name implies, median filters replace the pixel values with the median value in the local neighborhood. Median filtering, often for salt and pepper noise reduction, preserves edges while removing noise [19, 2, 17, 16], as the median value must actually be the value of one of the pixels in its neighborhood. *Bilateral filtering* is a non-linear, local, non-iterative and simple edge-preserving filtering method [36, 17]. As the name suggests, bilateral filtering was the combination of two types of filtering – domain filtering using pixel spatial closeness and range filtering using pixel value similarity [17].

Total variation (TV) denoising is non-linear, simple and relatively fast; it removes small scale noise while leaving important features such as edges intact, and thus is edge-preserving [31, 35, 17]. *TV-$l1$* – a modified version of TV – uses an ℓ_1 norm as a measure of fidelity between the observed and denoised images; the ℓ_1 norm makes it more geometric-based in the denoising process, as the regularization process has less dependence on the contrast of image features than on their shapes [7]. *Non-local means (NLM)* method uses a weighted averaging scheme to denoise images (it is non-local as the averages could be calculated over all pixels of the image); the authors [6] of NLM assume that in natural images a lot of structural similarities are present in different parts of the image and these repetitive structures can be used to restore images [6, 17]. Several extensions of NLM methods are developed (e.g., BM3D [10] and its extension BM4D [25]. However, although NLM and its extension are powerful and effective for denoising but the challenges of non-adaptive parameters (e.g., the sizes of smoothing kernel and neighborhood) still remains [17].

Relevant research on document images exists (as document images often contain noise introduced by the scanning process), but major methods on document images focus on binarization of images with thresholding techniques [33, 8, 16]. In [28], the different characters in the document are analyzed and a binary representation of the image is generated using thresholding, separating the text from a noisy background. Other methods of creating a binary representation of a text-based image include applying Otsu and Sauvola's thresholding [18], and Otsu thresholding alongside applying a median filter and other morphological techniques [46]. These processes are shown to remove salt and pepper noise, as well as smears [46, 28]. However, binarization is not the primary focus of this paper. We focus on how to remove the localized zigzag noise generated from scanning that are present in scanned binary diagrams. These localized noises are seen at the edges of the diagram and cannot be removed by many denoising methods, even the denoising methods designed for document images will not remove the zigzag noises. For example, in [46] a hybrid method is proposed, which uses the Otsu's method to initialize binarization, uses two dimensional median filters to reduce the leftover salt and pepper noises, and performs open and close operators to eliminate the remaining noises. *Wavelet* transform-based methods, often called *wavelet shrinkage*, use wavelet thresholding according to a shrinkage rule to denoise images. A challenge in the wavelet shrinkage process is to find

an adequate threshold value. Non-adaptive (e.g., VisuShrink [12]) and adaptive (e.g., SureShrink [11] and BayesShrink [9]) threshold estimation criteria were introduced for desired threshold estimation. Wavelets are used to denoise document images suffering from scanning introduced noises [4]. However, as [30] pointed out, wavelets are not shift and rotation invariant, and therefore they used curvelet transform (first introduced in [34]) to denoise document images.

There are other methods for document images, but to our knowledge, most of the methods are machine learning [23], neural network (e.g., cellular neural networks in [32]) or deep learning based (e.g., DnCNN used in [44, 16]), which requires (a large set of) training data. These approaches are beyond the scope of this work, as our approach provides a topological-based image representation and denoising method that requires no training data and is robust and whose parameter(s) are adaptive.

All the above denoising methods are pixel-based. Now we discuss some topological-based methods from which our denoising method builds upon. Yang *et al.* propose a topological graph-based image representation to automatically extract topological features that can be fed into different machine learning algorithms for image classification tasks [41]. Their topological graphs are generated automatically from image skeletons, which capture the topological and geometrical properties of shapes of objects present in images [21]. Yang *et al.* [41] uses the Zhang-Suen thinning algorithm [45], which is a well-known and robust skeleton extraction method, to extract skeletons from images. In our work, we also use the Zhang-Suen thinning algorithm to extract skeletons from binary diagram images. To further improve their graph representation, authors in [41] suggest two algorithms to simplify the geometries in the graph representation: the Douglas-Peucker [13] and the Visvalingam's [39]. In our method, we use the Douglas-Peucker algorithm because of its simplicity and robustness.

3. Methods and algorithms

In this section, we elaborate the proposed methods and algorithms, including workflow (Section 3.1), and data structures (Section 3.2) used in our developed algorithms (Section 3.3).

3.1. Workflow

The overview workflow of our method is provided in Figure 1. Given a diagram image, its skeleton is extracted using Zhang-Suen thinning algorithm [45]. The skeleton is then used to generate a skeleton graph. After that, important feature points defined as landmark nodes are detected from the skeleton graph. These landmark nodes are used to segment the graph into individual paths. In the next step, these paths and their geometry are used to create a hierarchical skeleton graph representing different levels of abstraction of the graph structure. We then compute an adaptive parameter to simplify each path in the hierarchical skeleton graph using Douglas-Peucker algorithm [13]. Once we get a simplified hierarchy skeleton graph G_s, we get a denoised image by drawing lines between those nodes that are connected in G_s.

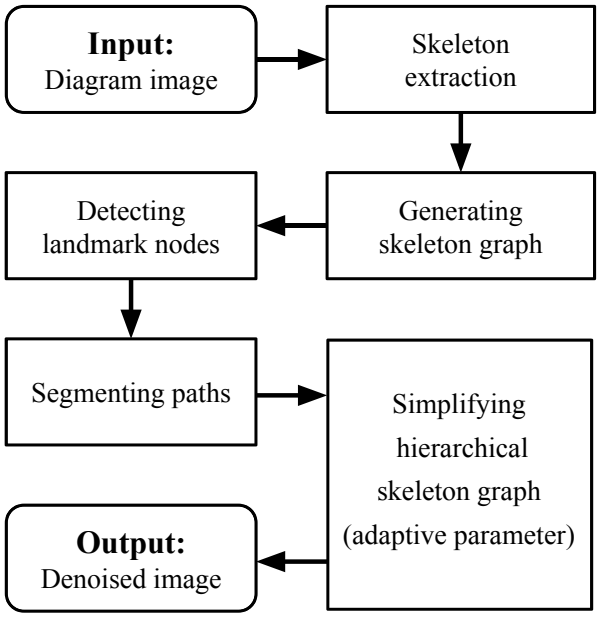


Figure 1. Workflow of our method.

3.2. Data structures

Below we introduce the data structures that will be used in the algorithms provided in Section 3.3.

Skeleton graph: A skeleton graph is an embedded graph $G = \{V, E\}$ generated from the skeleton of an image, where V is a set of white pixel-id, along with pixel coordinates as node attribute, and E is the edge list containing spatial connections among the nodes in V.

Hierarchical skeleton graph: A parent embedded graph $G_p = \{V_p, E_p\}$ contains simplified paths. Specifically, V_p contains collections of landmark nodes and E_p contains the path through those nodes.

Path: Each path is a child embedded graph $G_c = \{V_c, E_c\}$ that contains non-landmark nodes for paths identified by landmark nodes. Specifically, V_c contains collections of non-landmark nodes for a specific path defined by a member of V_p and E_c contains the edges that connect the non-landmark nodes through different paths.

3.3. Algorithms

3.3.1 Generating skeleton graph

Algorithm 1 provides detailed steps to generate a skeleton graph from a given diagram image. We use the Zhang-Suen thinning algorithm [45] to extract the skeleton for a given image, which is one pixel wide. From this skeleton a graph is constructed, where white pixels are represented by nodes, and edges are assigned between nodes whose pixels are nearest neighbors in the image skeleton. The result is an embedded graph $G = \{V, E\}$ where V holds the node id and planar coordinates and E holds the pairwise edge connections for each node. Figure 2 shows an example of a skeleton graph.

Algorithm 1: Generating skeleton graph

Input: A diagram image I
Output: An embedded graph $G = \{V, E\}$ representing the skeleton of I `// the nodes of an embedded graph contain coordinates, therefore the graph can be drawn uniquely on a plane`

$I \leftarrow$ imported image
`/* Pre-processing */`
$I \leftarrow$ binarized I
if *background (I) is white* **then**
  $I \leftarrow$ invert (I)
$I \leftarrow$ pad (I, 0) `// border extension`
$I \leftarrow$ dilate (I)
$S \leftarrow$ skeleton (I) `// Skeleton Extraction`
`/* Generating Skeleton Graph */`
$V \leftarrow$ nodes (S)
`// the coordinates of each node are stored in a list where the list index is the id/label for the individual node`
$E \leftarrow$ edges (S) `// two nodes are edge-connected if the pixels represented by the nodes are first nearest neighbors. A list of all edge connections is stored for each node.`
return V, E

3.3.2 Detecting landmark nodes

The next step is to segment the skeleton graph G into a collection of paths. In order to perform the segmentation, breakpoints (i.e., landmark nodes) need to be detected within the graph structure. Algorithm 2 provides the details for detecting landmark nodes.

In [43] *junction nodes*, *turning nodes*, and *end nodes* are defined based on the topological and geometric features found in the graphs. A *junction node* is where more than two paths meet, specifically any node that is connected to three or more other nodes. A *turning node* is where the direction of a path changes significantly, specifically a node where the angle between its two nearest neighbors is within a set tolerance range. An *end node* is any node where a path ends, specifically, any node that only has one neighbor.

3.3.3 Segmenting paths

After landmark nodes are detected, they are used to segment the skeleton graph into a collection of paths. The details for path segmentation using landmark nodes are provided in Algorithm 3.

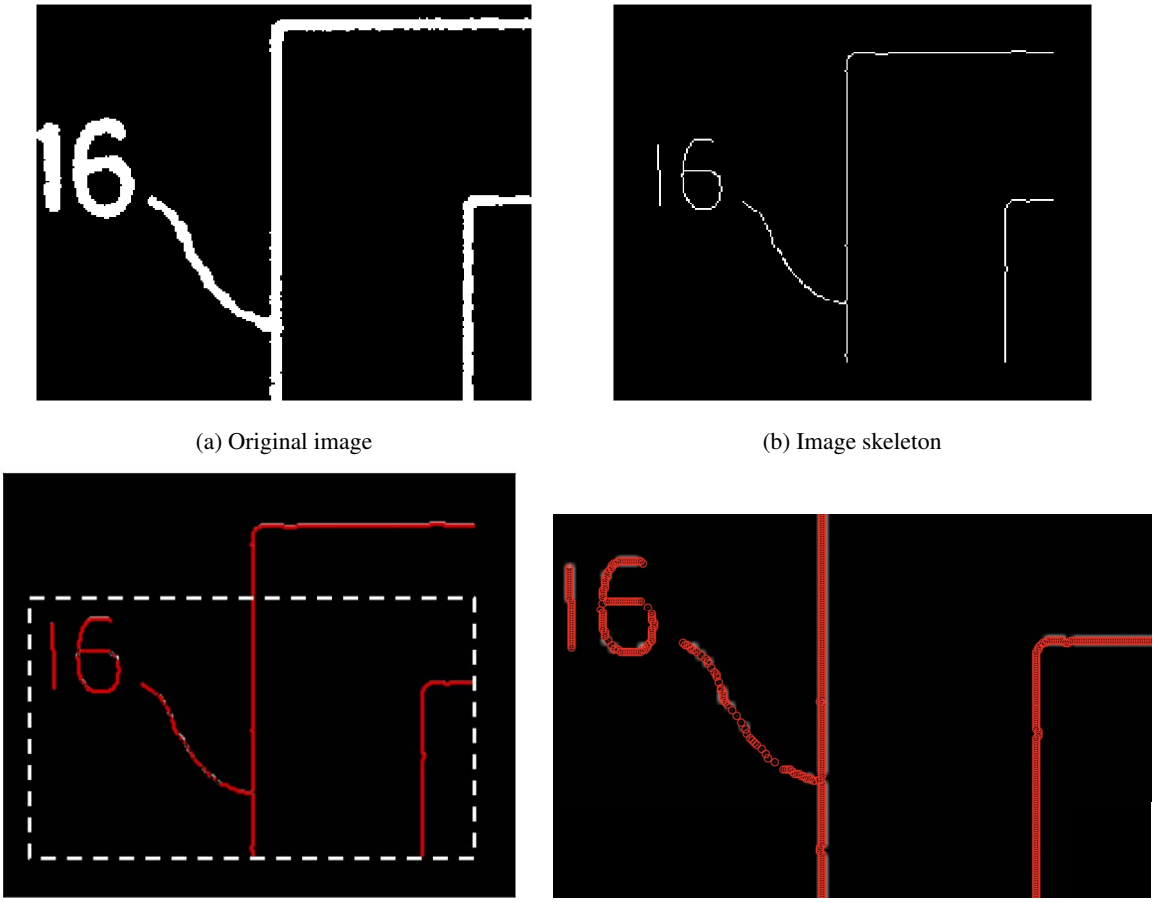
The paths containing junction nodes are firstly collected. Starting with a single junction node, a path stemming from that junction node is explored and stored as separate non-landmark nodes in the Path graph. The exploration occurs by taking one step at a time down the path and comparing the newly discovered node to the list of junction nodes and end nodes. The path ends when another junction node or an end node is found. After one path is completed, the process is repeated for all other paths stemming from that initial junction node.

The paths are explored for all junction nodes and for any end or turning nodes that have not already been added to a previous path. A portion of the graph without junction nodes could be a disconnected section of the skeleton graph. A circular curve of an image skeleton may produce a section of its corresponding skeleton graph with no junction nodes or end nodes, so turning nodes are also checked.

3.3.4 Simplifying hierarchical skeleton graph

Once paths are segmented, to create a simplified hierarchical skeleton graph, the segmented paths need to be simplified.

The hierarchical skeleton graph is simplified by using the Douglas-Peucker algorithm [13], which takes geometric curves composed of line segments, and reduces the number of points to represent the curve. Here the



(a) Original image

(b) Image skeleton

(c) Skeleton graph

(d) A zoomed in detail of (c)

Figure 2. Example of the graph generation process. Figure 2 (a) shows the original image before processing. Figure 2 (b) shows the skeleton of the original image. Figure 2 (c) plots the graph generated from the skeleton where each red dot is a node in the graph. Figure 2 (d) shows a zoomed in detail of Figure 2 (c), highlighting the nodes and edges in the skeleton graph.

curve is taken to be the collection of planar coordinates contained in the nodes of a given path. Detailed steps are given in Algorithm 4.

The simplified representation of the curve is assigned as a node in the hierarchical graph. The edges in the hierarchical graph connect the nodes together completing the path.

The Douglas-Peucker algorithm requires a parameter ϵ that governs the complexity of the simplified curve. In our algorithm, this parameter is adaptive, because it is computed automatically based on the convex hull of the coordinates of the nodes that comprise a specific path.

Once the hierarchical skeleton graph is simplified, a denoised image can be generated by drawing lines between those nodes that are connected in the simplified skeleton graph.

4. Experiments and results

To test and evaluate our method and algorithms, we run experiments on diagram images from the 2000 Binary Patent Image Database developed by Multimedia Knowledge and Social Media Analytics Laboratory (MKLab); the database contains images from patents maintained by the European Patent Office [1]. For the implementation, we use OpenCV, Scikit-image [37], SymPy [26], and NetworkX [14], as well as SParse Optimization Research COde (SPORCO) [40].

We compare our method with six commonly used denoising methods: Gaussian filter, median filter, bilateral filter, TV-$l1$, NLM, and wavelet; where the first five are

Algorithm 2: Detecting landmark nodes

Input: An embedded graph $G = \{V, E\}$ representing the skeleton of a diagram image

Output: A list of junction nodes $Junct\_pts$, a list of end nodes End_pts, and a list of turning nodes $Turn\_pts$ of the skeleton graph G `// A junction node is a node where more than two paths meet, an end node is a node where a path ends, and a turning node is a node where the angle of a path changes significantly.`

$Junct\_pts, End\_pts, Turn\_pts \leftarrow null$ `// Initialization`

foreach $v \in V$ **do**
  `/* count the number of edges that are incident to node v to determine the node type of v */`
  $n \leftarrow$ sum$(E(v))$
  if $n == 1$ **then**
    append v to End_pts
  if $n > 2$ **then**
    append v to $Junct\_pts$
  if $n == 2$ **then**
    `// exactly two incident edges implies a possible turning node`
    $first \leftarrow$ first neighbor of v
    $second \leftarrow$ second neighbor of v
    $angle \leftarrow$ angle_between$(first, second)$
    if $angle >= 45$ *and* $angle <= 135$ **then**
      `// the path turns at a significant angle, therefore v is a turning node`
      append v to $Turn\_pts$

return $End\_pts, Junct\_pts, Turn\_pts$

Algorithm 3: Segmenting paths

Input: An embedded graph $G = \{V, E\}$ representing the skeleton of a diagram image

Output: A list of paths P from the graph G separated by landmark nodes

$Junct\_pts, End\_pts, Turn\_pts \leftarrow$ detect_landmark_nodes (G) `// Detecting landmark nodes using Algorithm 2`
`/* Initialization */`
$P \leftarrow null$ `// completed paths`
$Visited \leftarrow null$ `// nodes already touched`
$Current \leftarrow null$ `// working path`
`/* Start with the junction nodes */`
foreach $v \in Junct\_pts$ **do**
  `/* edges represent path directions */`
  $Edges \leftarrow E(v)$
  foreach $neighbor \in Edges$ **do**
    `/* the next step in the path */`
    $next \leftarrow neighbor$
    `/* start walking the path */`
    append v to $Current$
    append v to $Visited$
    append $next$ to $Current$
    append $next$ to $Visited$
    while *there are more steps to take* **do**
      `/* Test to see if we reached the end of a path */`
      if $next \in Junct\_pts$ *or* $next \in End\_pts$ **then**
        `/* the path is finished */`
        break
      else
        `/* walk forward */`
        $first \leftarrow$ first neighbor of $next$
        $second \leftarrow$ second neighbor of $next$
        `/* add the node that isn't in the path */`
        if $first \notin Visited$ **then**
          $next \leftarrow first$
        else
          $next \leftarrow second$
      append $next$ to $Current$
      append $next$ to $Visited$
    append $Current$ to P
`/* There may be paths that do not contain a junction node */`
Repeat for nodes $N \in End\_pts$ that are $\notin Visited$
Repeat for nodes $N \in Turning\_pts$ that are $\notin Visited$
return P

designed for natural images and wavelet is for document images. These results can be found in the set of figures organized in Table 1 (for readability, only partial images are given here; pre-processing, such as border extension, is performed before denoising using the six existing methods, and before skeleton extraction using our method).

From the experiment results shown in Table 1, we can see that our method successfully removes the jagged zigzag edge noise found in scanned diagram images. As our method is developed to enhance the linear features of

Algorithm 4: Simplifying hierarchical skeleton graph

Input: Segmented paths P
Output: A simplified hierarchical skeleton graph S

$S \leftarrow null$ // Initialization
foreach *path* $p \in P$ **do**
  $N \leftarrow$ Node (p) // the nodes of p
  $n \leftarrow$ num (N) // the number of nodes
  if $n > 2$ **then**
    /* cacluate adaptive Douglas-Peucker parameter ϵ */
    $C \leftarrow$ convex hull (N)
    $\epsilon \leftarrow$ area of C / perimeter of C
    Simplify path p using Douglas-Peucker algorithm with parameter ϵ
    $new\_path \leftarrow$ simplified p
    append new_path to S
return S

diagram images, the width of the lines and curves are reduced in our denoising method while junctions between lines and linear directions are preserved well.

Looking closer at the experiment results in Table 1, we can see that in Images 4, 5, and 6, there are some curved lines that appear boxy in our denoised images, but the zigzag pattern in Image 7 is preserved well. For Image 6 and 7 we can see lines that are very close together in the original image being combined into one region to varying degrees in all denoising methods. The lines being smeared together is caused by image dilation, which is one of the pre-processing techniques. In addition, in Image 5 the original image has arrows, which are reduced to lines in the skeletonization process.

All of the six denoising methods require pre-setting of parameters, but the parameter in our method is adaptive. See Table 2 provided in Appendix C for parameter settings for the denoising methods shown in Table 1.

The computation time for each method and each sample image are shown in Table 3 in Appendix C (see Appendix D for our experiments' computing environment). In terms of computation time, our method is on par with NLM approach, outperforms the TV-$l1$ method, but is not as fast as the Gaussian, median, and bilateral filtering and wavelet methods.

5. Conclusion, limitation and future work

We have proposed a topological graph-based representation, along with algorithms built upon the representation, to denoise (binary) diagram images that contain zigzag noise introduced by digitization techniques such as scanning. Our hierarchical skeleton graph image representation can capture the major topological relations and spatial arrangement among entities present in an image, while eliminating undesired zigzag noise found at the edges of lines and curves in diagram images.

Our method is applied to diagram images from a patent database, and compared with six commonly used denoising methods; the experiment results show the effectiveness of our topological graph-based representation and algorithms. Specifically, our approach achieves very good and robust results with only one parameter (and most importantly, it is adaptive). By contrast, almost all other denoising methods require pre-setting of parameter(s), and this indicates fine-tuning is required to get optimized parameter(s) for desired results, which is time-consuming and thus not practical.

One major limitation of our method is that curves with large curvature may be oversimplified and thus resulting in boxy line segments (see images 4, 5, and 6 in Table 1). We will optimize our adaptive parameter to make our method able to cope with denoising both lines and curves with large curvature. In addition, the computation time of our method was mostly spent on the skeleton extraction step. We will improve skeleton extraction using the algorithm introduced in [42].

Our demonstrated denoising method and algorithms are applied to binary images, but we expect our topological graph-based image representation should work for other types of images. Non-binary images can be converted to grayscale and then binarization through thresholding, after which our method should be applicable.

Our hierarchical skeleton graph representation of (low quality) binary images using a topological graph-based approach provides essential structure and lays an important foundation for computer vision such as scene graph based applications. In this paper, we have demonstrated that the representation works effectively and robustly for denoising low quality binary images, but its potential has a much wider range. Many computer vision and image processing tasks and applications can benefit from our representation and algorithms, from fundamental and classic computer vision tasks such as line detection, image classification (see [41] for an example) and object recognition, to higher level tasks and applications such as text recognition using OCR, detection of lanes, stop signs, and crossing lines for autonomous driving, as well as scene understanding through scene graphs that can be generated from our hierarchical skeleton graph representation.

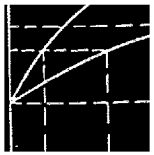
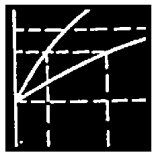
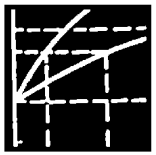
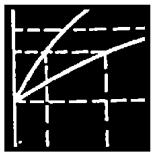
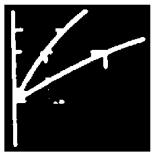
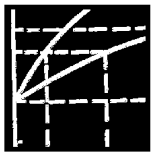
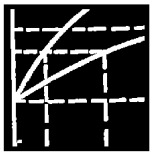
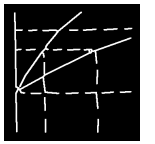
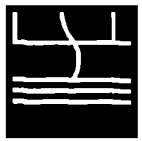
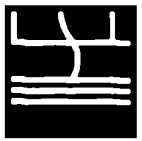
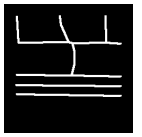
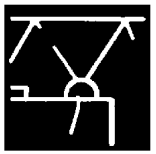
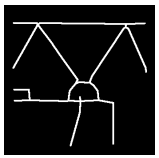
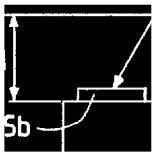
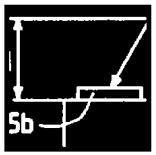
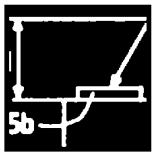
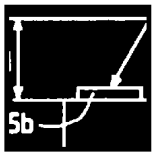
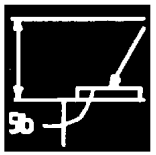
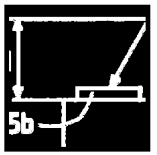
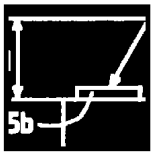
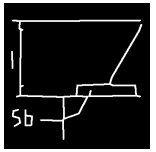
Image #	Original images	Gaussian filter	Median filter	Bilateral filter	TV-$l1$ denoising	NLM denoising	Wavelet denoising	Our method
Image 1								
Image 2								
Image 3								
Image 4								
Image 5								
Image 6								
Image 7								

Table 1. Experiment results of our and six commonly used denoising methods.

A. Abbreviations

In this appendix, we provide the abbreviations (ordered alphabetically) of terms we used in the paper.

BM3D	Block-matching and 3D filtering
DnCNN	Denoising convolutional neural network
NLM	Non-local mean
OCR	Optical character recognition
SIFT	Scale-invariant feature transform
SURF	Speeded up robust features
TV	Total variation
TV-$l1$	Total variation-$l1$ norm

B. Definition of terms used

In this appendix, we provide brief definitions to some concepts (ordered alphabetically; referenced [42, 43, 41, 15, 27]) in graph theory and computational geometry we used in our hierarchical skeleton graph based image denoising methods and algorithms.

Convex hull

Given a collection of 2D points P, the *convex hull* of P is the smallest convex set S that contains P; S is a subset of P, and S contains all possible convex combinations of P. For a set $\{p_1, p_2, ..., p_n\}$, a convex combination of those points is represented as another point $p = \sum_{k=1}^{n} \alpha_k p_k$ where $\sum_{k=1}^{n} \alpha_k = 1$.

Embedded graph

An *embedded graph* is a graph where each node has (planar) coordinates so the graph can be drawn on a plane uniquely.

Graph

A *graph* consists of a collection of nodes (also called vertices or points) and edges that connect nodes.

Skeleton

The *skeleton* of a binary image is a central line extraction of objects in the image via thinning.

Skeleton graph

The *skeleton graph* of a binary image B is an embedded graph generated by pixels (including coordinates) and connection among pixel neighbors from the skeleton extracted from B.

Path

In graph theory, a *path* is a set of nodes and their connecting relationships. From a given node, all other nodes in the path can be traversed by travelling through the edges.

C. Parameter settings and computation time

This appendix provides the parameter settings and computation time for denoising methods in Table 1.

Table 2 provides parameter settings for the denoising methods shown in Table 1 (For other parameters not listed specifically in the table, we took default parameters available in the tools used).

Table 3 provides the computation time for each method and each sample image (computing environment for the experiments is provided in Appendix D below).

D. Computing environment

In this appendix, we provide the computing environment that we ran our experiments. We ran all the experiments on a ThinkPad l580 (Linux: Ubuntu:version 18.04) with Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz (1 processor, and 4 CPU cores for each processor; 4 GiB RAM).

References

[1] Multimedia knowledge and social media analytics laboratory (MKLab). 2000 binary patent images database, 2010. Available online: https://mklab.iti.gr/results/patent-image-databases/ (accessed on August 14, 2019). 5

[2] E. Arias-Castro, D. L. Donoho, et al. Does median filtering truly preserve edges better than linear filtering? *The Annals of Statistics*, 37(3):1172–1206, 2009. 2

[3] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006. 1

[4] K. Berkner. Enhancement of scanned documents in besov spaces using wavelet domain representations. In *Document Recognition and Retrieval IX*, volume 4670, pages 143–154. International Society for Optics and Photonics, 2001. 3

[5] A. K. Boyat and B. K. Joshi. A review paper: noise models in digital image processing. *arXiv preprint arXiv:1505.03489*, 2015. 2

[6] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005. 2, 10

[7] T. F. Chan and S. Esedoglu. Aspects of total variation regularized l1 function approximation. *SIAM Journal on Applied Mathematics*, 65(5):1817–1837, 2005. 2, 10

[8] Y. Chen and L. Wang. Broken and degraded document images binarization. *Neurocomputing*, 237:272–280, 2017. 2

[9] H. A. Chipman, E. D. Kolaczyk, and R. E. McCulloch. Adaptive bayesian wavelet shrinkage. *Journal*

Denoising methods	**Parameters**	**Tools and methods used**
Gaussian filter	*zero-mean filter size*: 5x5	*GaussianBlur* method (OpenCV)
Median filter [19]	*aperture size*: 5	*MedianBlur* method (OpenCV)
Bilateral filter	*sigmaColor* value: 5; *sigmaSpace value*: 175	*BilateralFilter* method (OpenCV)
TV-$l1$ denoising [7]	*lambda* value: $40e-1$	*TVL1Denoise* method (SPORCO [40])
NLM denoising [6]	*patch size*: 7; *patch distance*: 11; *cut-off distance*: 0.5; *Gaussian noise standard deviation*: 0	*denoise_nl_means* method (Scikit-image [37])
Wavelet denoising	*sigma*: estimate_sigma; *wavelet*: db1; *mode*: soft; *wavelet_levels*: None; *method*: BayesShrink	*denoise_wavelet* method with *estimate_sigma* to estimate noise distribution (Scikit-image [37])
Our method	the parameter for simplification(i.e., ϵ) is **adaptive** (see line 5 in Algorithm 4), and this is the only parameter required for our denoising method	Implemented by the authors of the paper

Table 2. Parameter settings for denoising methods in Table 1.

Image #	Gaussian filter	Median filter	Bilateral filter	TV-$l1$	NLM	Wavelet	Our method
Image 1	< 1	< 1	< 1	6	2	< 1	4
Image 2	< 1	< 1	< 1	20	5	< 1	7
Image 3	< 1	< 1	< 1	7	2	< 1	2
Image 4	< 1	< 1	< 1	19	5	< 1	6
Image 5	< 1	< 1	< 1	16	5	< 1	7
Image 6	< 1	< 1	< 1	8	2	< 1	3
Image 7	< 1	< 1	< 1	6	2	< 1	3

Table 3. Computation time (in seconds) corresponding to denoising methods in Table 1.

of the American Statistical Association, 92(440):1413–1421, 1997. 3

[10] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007. 2

[11] D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, 90(432):1200–1224, 1995. 3

[12] D. L. Donoho and J. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425–455, 1994. 3

[13] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973. 3, 4

[14] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *In Proceedings of the 7th Python in Science Conference (SciPy)*, pages 11–15, 2008. 5

[15] J. M. Harris, J. L. Hirst, and M. J. Mossinghoff. *Combinatorics and graph theory*, volume 2. Springer, 2008. 9

[16] Z.-K. Huang, Z.-N. Wang, J.-M. Xi, and L.-Y. Hou. Chinese rubbing image binarization based on deep learning for image denoising. In *Proceedings of the 2nd International Conference on Control and Computer Vision*, pages 46–50. ACM, 2019. 2, 3

[17] P. Jain and V. Tyagi. A survey of edge-preserving image denoising methods. *Information Systems Frontiers*, 18(1):159–170, 2016. 2

[18] D. Jyoti, B. Raj, A. Sharma, and K. Kapoor. Document image binarization technique for degraded document images by using morphological operators. *Int. J. Adv. Res. Ideas Innov. Technol.*, 2(3):1–7, 2016. 2

[19] V. V. Khryashchev, A. L. Priorov, I. V. Apalkov, and P. S. Zvonarev. Image denoising using adaptive switching median filter. In *IEEE International Conference on Image Processing 2005*, volume 1, pages I–117. IEEE, 2005. 2, 10

[20] V. Kumar and A. Samadhiya. Comparative performance analysis of image de-noising techniques. *arXiv preprint arXiv:1901.06529*, 2019. 2

[21] J. K. Lakshmi and M. Punithavalli. A survey on skeletons in digital image processing. In *2009 international conference on digital image processing*, pages 260–269. IEEE, 2009. 3

[22] L. Lam, S.-W. Lee, and C. Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 14(9):869–885, 1992. 1

[23] D. Li. Support vector regression based image denoising. *Image and Vision Computing*, 27(6):623–627, 2009. 3

[24] D. G. Lowe et al. Object recognition from local scale-invariant features. In *ICCV*, volume 99, pages 1150–1157, 1999. 1

[25] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi. Nonlocal transform-domain filter for volumetric data denoising and reconstruction. *IEEE transactions on image processing*, 22(1):119–133, 2012. 2

[26] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, et al. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103, 2017. 5

[27] M. Needham and A. E. Hodler. *Graph Algorithms: Practical Examples in Apache Spark and Neo4j*. O'Reilly Media, 2019. 9

[28] N. Ntogas and D. Veintzas. A binarization algorithm for historical manuscripts. In *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering*, volume 12, pages 41–51. World Scientific and Engineering Academy and Society, 2008. 2

[29] P. Patidar, M. Gupta, S. Srivastava, and A. K. Nagawat. Image de-noising by various filters for different noise. *International journal of computer applications*, 9(4):45–50, 2010. 2

[30] C. Patvardhan, A. Verma, and C. Lakshmi. Denoising of document images using discrete curvelet transform for ocr applications. *International Journal of Computer Applications*, 55(10), 2012. 3

[31] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992. 2

[32] T. Saba, A. Rehman, and G. Sulong. An intelligent approach to image denoising. *Journal Of theoretical and Applied information technology*, 17(2):32–36, 2010. 3

[33] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33(2):225–236, 2000. 2

[34] J.-L. Starck, E. J. Candès, and D. L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on image processing*, 11(6):670–684, 2002. 3

[35] D. Strong and T. Chan. Edge-preserving and scale-dependent properties of total variation regularization. *Inverse problems*, 19(6):S165, 2003. 2

[36] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Iccv*, volume 98, page 2, 1998. 2

[37] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014. 5, 10

[38] R. Verma and J. Ali. A comparative study of various types of image noise and efficient noise removal techniques. *International Journal of advanced research in computer science and software engineering*, 3(10), 2013. 2

[39] M. Visvalingam and J. D. Whyatt. Line generalisation by repeated elimination of points. *The cartographic journal*, 30(1):46–51, 1993. 3

[40] B. Wohlberg. SParse Optimization Research COde (SPORCO). Software library available from http://purl.org/brendt/software/sporco, 2016. 5, 10

[41] L. Yang, D. Oyen, and B. Wohlberg. Image classification using topological features automatically extracted from graph representation of images. In *Proceedings of the 15th International Workshop on Mining and Learning with Graphs (MLG)*, 2019. 1, 3, 7, 9

[42] L. Yang, D. Oyen, and B. Wohlberg. A novel algorithm for skeleton extraction from images using topological graph analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 1, 7, 9

[43] L. Yang and M. Worboys. Generation of navigation graphs for indoor space. *International Journal of Geographical Information Science*, 29(10):1737–1756, 2015. 1, 4, 9

[44] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017. 3

[45] T. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984. 3, 4

[46] Y. Zhang and L. Wu. A fast document image denoising method based on packed binary format and source word accumulation. *Journal of Convergence Information Technology*, 6(2):131–137, 2011. 2