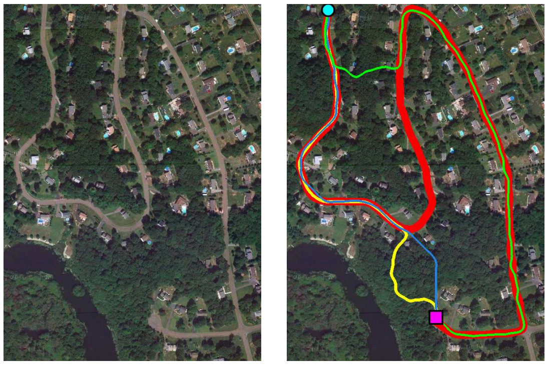
Progressive Minimal Path Method with Embedded CNN

Wei Liao
Independent Researcher
liaowei.post@gmail.com

Abstract

We propose Path-CNN, a method for the segmentation of centerlines of tubular structures by embedding convolutional neural networks (CNNs) into the progressive minimal path method. Minimal path methods are widely used for topology-aware centerline segmentation, but usually these methods rely on weak, hand-tuned image features. In contrast, CNNs use strong image features which are learned automatically from images. But CNNs usually do not take the topology of the results into account, and often require a large amount of annotations for training. We integrate CNNs into the minimal path method, so that both techniques benefit from each other: CNNs employ learned image features to improve the determination of minimal paths, while the minimal path method ensures the correct topology of the segmented centerlines, provides strong geometric priors to increase the performance of CNNs, and reduces the amount of annotations for the training of CNNs significantly. Our method has lower hardware requirements than many recent methods. Qualitative and quantitative comparison with other methods shows that Path-CNN achieves better performance, especially when dealing with tubular structures with complex shapes in challenging environments.



(a) Input image (b) Segmented centerlines

Figure 1. Short cuts due to complex centerline geometry. (a) The input is a satellite image of roads. (b) Start point (magenta box) and end point (cyan circle) of a road are given. Results of a previous method using only hand-tuned tubularity measure (yellow line) and two methods using CNNs (blue and green lines) contain short cuts. Our approach achieves the correct centerline (red line), although it is much longer than the results with short cuts.

1. Introduction

Topology-aware centerline segmentation for tubular structures plays a crucial role in computer vision. One of its most important application areas is the quantitative analysis of roads and rivers in satellite images for measurement, planning, or navigation. These are challenging tasks due to the complex shape of roads and rivers, and the high variability of their environment.

When using common methods for object segmentation for this task, usually a binary mask of the tubular structure is computed in the first step. After that, post-processing, often based on heuristics, is necessary, in order to determine the centerline and to deal with small gaps on the tubular structure due to noise or image clutter. In contrast, *minimal path methods* based on Dijkstra's algorithms [7] or the fast marching method [5] provide a more elegant solution. As in most minimal path methods, we assume that the start point $\mathbf{x}_\mathrm{s}$ and end point $\mathbf{x}_\mathrm{e}$ of the centerline are given, and focus on the determination of the path itself. Often, the start and end points are automatically obtained using application specific methods, such as [19]. Minimal path methods allow finding the best path as the global optimum of a cost function, while inherently enforcing strict line-topology, i.e., the result is always a sequence of coordinates of points on the centerline. Also, small gaps on the path can be completed automatically. However, minimal path methods usually utilize hand-tuned image features, such as differential measures [8, 13], to distinguish between tubular structures and the background. Such features are efficient to compute, but they are relatively weak and may lead to *short cuts* for images containing challenging environments. With convolutional neural networks (CNNs), stronger image features can be extracted automatically from images. Although these features alone cannot ensure the topology of the centerlines, they can be used to better classifying the pixels and thus improve the results of minimal path methods. However, CNNs often require large amounts of annotated training data, which can be expensive to obtain. If only limited annotations are available, often CNN-based methods cannot be fully trained, and thus they also result in short cuts. Examples of short cuts are shown in Fig. 1.

In this work, we propose a novel method, Path-CNN, to embed CNNs into the progressive minimal path method [15], so that these two techniques operate *alternately* and benefit from each other: On the one hand, we use CNNs to learn image features automatically so that a wider variety of short cuts can be detected, and therefore minimal paths can be better determined. Instead of learning features for *isolated pixels* as in most previous approaches, our CNNs learn features for *rectified patches* along *paths*. On the other hand, the progressive minimal path method not only ensures the line-topology of the results, but also provides strong geometric priors, which are used in turn to reduce the number of training samples for CNNs significantly. Although we only use centerlines as training data, our method not only determines centerlines, but also produces binary segmentation masks for tubular structures. To the best of our knowledge, this is the first approach to employ such geometric priors for CNNs to segment centerlines of tubular structures. Compared with most other approaches based on deep learning, our method has lower requirements not only for the amount of annotations and but also for hardware.

2. Related Work

In this section, we review the two main components of minimal path methods: Image features and minimal path computation. We also emphasize the differences between our method and other recent methods for road extraction.

Image features Minimal path methods often use tubularity measures as image features. Such measures can be interpreted as the probability that a pixel belongs to a tubular structure. Widely used features, such as Hessian-based measures [8, 23] or flux-based features [13, 27], are hand-tuned. There are also learning-based features. For example, [24] uses features based on decision trees, and [18] uses a CNN. However, these features are learned for isolated *pixels*. Also, these features are *static*, i.e., they are computed *before* the minimal path computation starts, and remain constant thereafter. In contrast, we use *path-based features*. While being stronger than their pixel-based counterparts, such features can only be computed *dynamically*, i.e., *during* the minimal path computation.

Minimal path computation There are different methods to overcome short cuts when computing minimal paths. With domain-lifting, additional dimensions are introduced into the parameter space to represent more features such as line width or orientation [14,20], but the computational cost increases significantly with the dimensionality. In [3,4], fast marching methods with anisotropic features are proposed, but such features require more complex numerical schemes than commonly used isotropic features. We use Dijkstra's algorithm, in which both anisotropic and isotropic features are used in the same way, without the need for further numerical schemes. In [28], more complex graph structures are used to represent higher-order constraints such as curvature or torsion, but this results in high computation time and limited feature types. In [2, 12], additional keypoints are inserted heuristically, and *progressive minimal path methods* [4,15] employ path-based features computed on-the-fly. But these approaches still rely on hand-tuned features based on appearance or geometry. Also, the path-based features in these approaches are still derived from pixel-based features. In contrast, we use dynamic features which are learned directly using paths. Furthermore, our method can handle a wider variety of short cuts in a uniform way.

Methods for road extraction For road extraction using satellite images, there exist recent approaches (e.g., [1, 16, 19]). However, there are important distinctions between these methods and ours. First, existing methods usually require large amounts of training data. For example, [19] uses one of the largest public datasets of road images [1], and employs U-Net to obtain features for further refinement. In contrast, by combining minimal path methods and CNNs, our method requires much less annotated training data than U-Net and most other architectures, i.e., our method performs better if the available annotation is very sparse. Second, many models, such as [1], have relatively high requirements for hardware, whereas our model can be trained and deployed efficiently using only 2GB GPU memory. Third, previous methods were only applied to extract roads in urban or suburban areas, while our method has been used also for roads in other environments and rivers. Furthermore, although our method is trained only with centerline annotations, it also produces a classification of each pixel in the image, which corresponds to a binary segmentation. In this way, further properties of the tubular structures, such as width or area, can also be determined.

3. Minimal Path Framework

Segmentation of tubular structures can be formulated naturally using the minimal path framework, which relies on Dijkstra's algorithm in discrete cases, or the fast marching method in continuous cases. Our approach focuses on Dijkstra's algorithm. Let image I induce a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of *vertices* and *edges*, respectively. Each vertex corresponds to a pixel in I, and vertices of neighboring pixels are connected by edges. Function $\mathbf{w}$ uses image features to assign positive *weights* to the edges. A *path* γ is a sequence of vertices $\{v_0, v_1, \ldots, v_{|\gamma|}\}$. Given a start point $\mathbf{x}_\mathrm{s}$ and an end point

```
Algorithm 1: Unified Dijkstra's algorithm.
  Input: Start point $\mathbf{x}_\mathrm{s}$, end point $\mathbf{x}_\mathrm{e}$, image $I$, graph
         $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, initial edge weights $\mathbf{w}_\mathrm{ini}$,
  Output: predecessor function $\pi$
  for each $\mathbf{x} \in \mathcal{V}$ do
      $\pi[\mathbf{x}] \leftarrow$ none; $d[\mathbf{x}] \leftarrow \infty$;
  $d[\mathbf{x}_\mathrm{s}] \leftarrow 0$; $\mathcal{Q} \leftarrow \mathcal{V}$; $\mathcal{S} \leftarrow \varnothing$;
  while $\mathbf{x}_\mathrm{e} \notin \mathcal{S}$ do // Main loop
      $u \leftarrow \arg\min_{\mathbf{x} \in \mathcal{Q}} d(\mathbf{x})$;
      $\mathcal{Q} \leftarrow \mathcal{Q} - \{u\}$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{u\}$;
      for each $v \in \mathcal{N}(u)$ do
          $\mathbf{w}[e_{u,v}] = \mathsf{AdaptWeight}(u, v, \mathbf{w}_\mathrm{ini}, \pi, I)$;
          if $d[v] > d[u] + \mathbf{w}[e_{u,v}]$ then
              $d[v] \leftarrow d[u] + \mathbf{w}[e_{u,v}]$;
              $\pi[v] \leftarrow u$;
  return $\pi$
```

$\mathbf{x}_\mathrm{e}$, the minimal path $\hat{\gamma}_{\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{e}}$, which corresponds to the centerline of a tubular structure in the image, can be determined by minimizing the following cost function

$$\hat{\gamma}_{\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{e}} = \underset{\gamma \in \Gamma(\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{e})}{\arg\min} \sum_{i=1}^{|\gamma|} \mathbf{w}[e_{v_{i-1},v_i}], \tag{1}$$

where $\Gamma(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{e})$ is the set of all paths connecting $\mathbf{x}_\mathrm{s}$ and $\mathbf{x}_\mathrm{e}$, and v_i is the i-th vertex on the path γ. Consecutive vertices v_{i-1} and v_i on γ are connected by edge e_{v_{i-1},v_i}.

To minimize (1), we use a unified formulation of Dijkstra's algorithm adapted from [6], as shown in Algorithm 1. For each vertex u, $\pi(u)$ specifies its *predecessor* in the path, while $d(u)$ is the *path weight*, i.e., sum of weights of all edges on the path between u and the start point $\mathbf{x}_\mathrm{s}$. $\mathcal{N}(u)$ denotes the set of neighboring vertices of u. We introduce a new function AdaptWeight to transform $\mathbf{w}_\mathrm{ini}$. In the standard Dijkstra's algorithm, AdaptWeight just returns the initial weight $\mathbf{w}_\mathrm{ini}[u, v]$ for edge $e_{u,v}$, i.e., $\mathbf{w} = \mathbf{w}_\mathrm{ini}$. An improved version of AdaptWeight is proposed in Sec. 4 below. $\mathcal{V}$ is divided into two disjoint sets $\mathcal{Q}$ and $\mathcal{S}$. For each vertex u in $\mathcal{S}$, $d(u)$ and $\pi(u)$ are finalized, while in $\mathcal{Q}$, they may still be updated. In each iteration of the main loop, the vertex u with minimum path weight is moved from $\mathcal{Q}$ to $\mathcal{S}$, and it is checked for each neighbor $v \in \mathcal{N}(u)$ whether $d(v)$ can be reduced by reaching v via $e_{u,v}$. If this is the case, then u becomes predecessor of v. Once the end point $\mathbf{x}_\mathrm{e}$ is added to $\mathcal{S}$, $\hat{\gamma}_{\mathbf{x}_\mathrm{s},\mathbf{x}_\mathrm{e}}$ can be extracted by starting at $\mathbf{x}_\mathrm{e}$, and recursively looking up the predecessors using π, until the start point $\mathbf{x}_\mathrm{s}$ is reached. The centerlines obtained in this way automatically have strict *line-topology*.

Usually, the initial weight $\mathbf{w}_\mathrm{ini}$ for edge e is defined as

$$\mathbf{w}_\mathrm{ini}[e] = \frac{1}{\mathcal{V}(e) + \epsilon} + \lambda \cdot L(e), \tag{2}$$

where $\mathcal{V}(e)$ is a tubularity measure, and ϵ is a small constant to avoid division by zero. The term $\lambda \cdot L(e)$ controls the smoothness of the path, where $L(e)$ is the Euclidean length of e, and λ is a constant. In our method, the tubularity measure [8] is used.

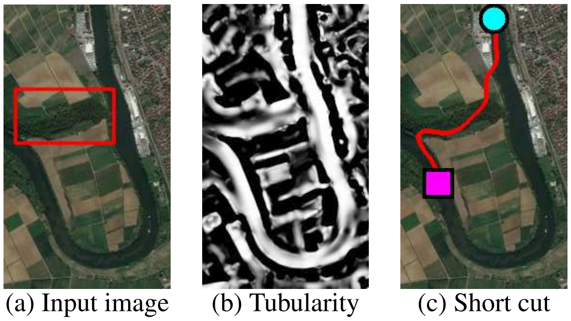

(a) Input image (b) Tubularity (c) Short cut

Figure 2. A short cut due to similar appearance of foreground and background. The region in the red box in (a) has similar tubularity as the foreground in (b), leading to a short cut in (c).

4. Path-CNN Method

In this section, we propose Path-CNN, a new approach to solving a common problem of minimal path methods: The short cut problem. To do so, we embed CNNs into the progressive minimal path method [15] in such a way that these two techniques naturally complement each other to achieve better performance.

4.1. Taxonomy of Short Cuts

Short cuts are incorrect centerlines found by minimal path methods. There are mainly two reasons for short cuts, which we refer to as Type 1 and Type 2, respectively. In cases of Type 1, the correct path may be very long and curved, so that a wrong but shorter connection, despite running through the background, still achieves a lower cost (1), such as the examples in Fig. 1b. In cases of Type 2, the background may appear very similar to the foreground. For example, in Fig. 2a, the red box indicates a background region similar to the river nearby. This region has also high tubularity (Fig. 2b), leading to a short cut (Fig. 2c). Most previous methods only attempt to avoid short cuts of Type 1, and some methods deal with Type 2 under certain assumptions for the geometry or appearance of the tubular structures (e.g., [4]). In contrast, our method takes both types into account in a general and uniform way.

4.2. Path Classification

To avoid short cuts, we add a step into Dijkstra's algorithm: A CNN is applied to classify image patches extracted using local paths, and the classification result is used to detect possible short cuts.

```
Algorithm 2: AdaptWeight for Path-CNN.
   Input: vertices $u$ and $v$, weight $\mathbf{w}_{\mathrm{ini}}$, $\pi$, image $I$
   Output: adapted weight $\mathbf{w}_a$
 1 $\gamma_{\mathrm{L}} \leftarrow$ ComputeLocalPath$(u, \pi)$;
 2 $P \leftarrow$ CropTubularPatch$(\gamma_{\mathrm{L}}, I)$;
 3 $P_{\mathrm{rect}} \leftarrow$ RectifyPatch$(\gamma_{\mathrm{L}}, P)$;
 4 $c \leftarrow$ Classify$(P_{\mathrm{rect}})$;
 5 if $c = Foreground$ then
 6     $\mathbf{w}_a = \mathbf{w}_{\mathrm{ini}}[e_{u,v}]$;
 7 else // $c = Background$, add penalty
 8     $\mathbf{w}_a = \mathbf{w}_{\mathrm{ini}}[e_{u,v}] + \mathbf{w}_{\mathrm{p}}$;
 9 return $\mathbf{w}_a$
```

Local paths Following [15], the *local path* $\gamma_{\mathrm{L}}(u)$ at a vertex $u \in \mathcal{S}$ is defined as the path of a *constant* length L_0. $\gamma_{\mathrm{L}}(u)$ can be determined by starting at u with an empty path, and recursively looking up the predecessors using π, until the path length reaches L_0. Intuitively, we can avoid $\hat{\gamma}_{\mathbf{x}_{\mathrm{s}},\mathbf{x}_{\mathrm{e}}}$ with short cuts by detecting and avoiding its segment γ_{L} which contains such short cuts. Using our method described below, we are able to use local paths to cope with short cuts of Types 1 and 2 by applying a single CNN.

CNN and path-based features To use path-based features, we introduce three new operations into the function AdaptWeight: *Cropping* of tubular patches along local paths, *rectification* of these patches, and *classification* of rectified patches using a CNN. All these steps must be computed on-the-fly, i.e., *during* the minimal path computation, since local paths need to be computed using π in the set $\mathcal{S}$, but $\mathcal{S}$ is non-empty only after the main loop of Algorithm 1 has started in Line 4.

Details of AdaptWeight are shown in Algorithm 2. The initial edge weights $\mathbf{w}_{\mathrm{ini}}$ are computed using tubularity measure. Depending on the classification result of the CNN, the final weight $\mathbf{w}$ is either equal to $\mathbf{w}_{\mathrm{ini}}$, or much higher than $\mathbf{w}_{\mathrm{ini}}$. First, we extract the *local path* γ_{L} at vertex u. Then, the image region along γ_{L} is *cropped* to a tubular image patch P with constant width, so that γ_{L} is the centerline of P. In the subsequent step of *rectification*, the tubular patch P is transformed into a rectangular patch by warping it along its centerline, and rotated into a canonical orientation, resulting in a rectified patch P_{rect}. Then P_{rect} is *classified* by the CNN, which has been trained using rectified image patches in the canonical orientation, instead of using non-rectified image patches of arbitrary orientations. Subsequent steps depend on the classification result of the CNN: If P_{rect} is classified as foreground, we conclude that u (the start point of γ_{L}) is inside certain tubular structures. In this case, the weight of the edge between u and v does not change. This is the same as in the standard Dijkstra's algorithm. On the other hand, if P_{rect} is classified as background, then we conclude that u is not inside a tubular structure, and consequently γ_{L} is more likely to be part of a short cut than part of the final minimal path $\hat{\gamma}_{\mathbf{x}_{\mathrm{s}},\mathbf{x}_{\mathrm{e}}}$, i.e., a possible short cut is detected at u. In this case, we increase the weight of the edge $e_{u,v}$ from its initial value $\mathbf{w}_{\mathrm{ini}}$ by a penalty $\mathbf{w}_{\mathrm{p}}$, which is a large positive number. So, even if v eventually turns out to be on $\hat{\gamma}_{\mathbf{x}_{\mathrm{s}},\mathbf{x}_{\mathrm{e}}}$, the probability that u is predecessor of v on $\hat{\gamma}_{\mathbf{x}_{\mathrm{s}},\mathbf{x}_{\mathrm{e}}}$ is significantly reduced due to $\mathbf{w}_{\mathrm{p}}$, since v might be reached via other neighboring vertices v' with lower path weight $d(v') + e_{v,v'}$.

We have put the new steps for the classification of P_{rect} into Algorithm 2 to better emphasize the difference to the standard Dijkstra's algorithm. In an actual implementation, P_{rect} only needs to be classified once for all neighbors of u.

Illustration One step of the main loop of the complete algorithm is demonstrated in Fig. 3. Suppose u is the element in $\mathcal{Q}$ with minimum path weight $d = 10$, its neighbors in $\mathcal{Q}$ have temporal path weights of $d(v_1) = 20$ and $d(v_2) = 12$, and the edge weights are $\mathbf{w}[e_{u,v_1}] = 5$ and $\mathbf{w}[e_{u,v_2}] = 8$. After computing local path γ_{L} (black line starting at u), the tubular patch P can be extracted (blue stripe around γ_{L}). Magenta and green shapes inside P represent its texture. We then transform and rotate P into a canonical orientation to obtain P_{rect}, and γ_{L} is accordingly transformed into a straight vertical line segment $\gamma_{\mathrm{L,rect}}$ in the middle of P_{rect}. The texture of P undergoes the same transformation and rotation. The CNN-classifier checks whether P_{rect} is in foreground. If this is the case, then $d(v_1)$ and $d(v_2)$ are updated as in the standard Dijkstra's algorithm: The previous value of $d(v_1)$ is larger than $d(u) + \mathbf{w}[e_{u,v_1}] = 15$, so $d(v_1)$ is updated to 15, and $\pi(v_1) = u$. $d(v_2)$ is not updated, since $d(u) + \mathbf{w}[e_{u,v_2}] = 18 > d(v_2)$. In contrast, if P_{rect} is *not* in foreground, then γ_{L} is a possible short cut. Thus the weights $\mathbf{w}[e_{u,v_1}]$ and $\mathbf{w}[e_{u,v_2}]$ are both increased by a high penalty $\mathbf{w}_{\mathrm{p}} = 1000$ to reduce the probability that u becomes a vertex on the final minimal path. In this case, neither $d(v_1)$ nor $d(v_2)$ changes, and π is not updated.

F-maps Our method not only provides centerlines of tubular structures, but also classifies pixels into foreground or background (Line 4 in Algorithm 2). To avoid confusion with centerline segmentation, we use the term *F-map* (foreground map) instead of *segmentation* to refer to the image region classified as foreground. F-maps have a remarkable property: Our CNN-classifiers are trained using only *centerline* annotations, but the resulting F-maps provide *binary segmentation masks* for the foreground.

4.3. Training and Inference

To train the CNN, we use centerlines *without* width information as annotations. Rectified patches along these centerlines are used as positive samples, while negative samples are not rectified. For example, in positive samples for

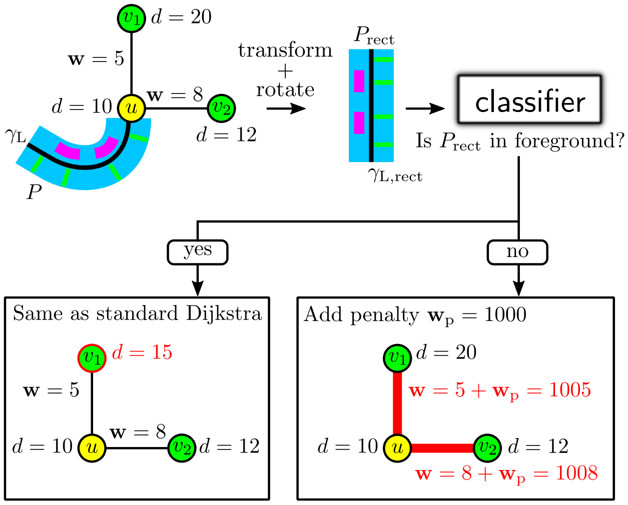


Figure 3. An illustration of one step of the complete algorithm. Indices are omitted because they are obvious in the shown setting. Red color indicates changes compared to the previous step.

road segmentation, there must be a vertical road in the center of the rectified patch (Fig. 4a). In negative samples, either there is no road at all (Fig. 4b), or the road deviates strongly from the vertical position (Fig. 4c, d). For data augmentation, rectified patches are only rotated in a very small range to compensate the possible numerical inaccuracies of centerlines.

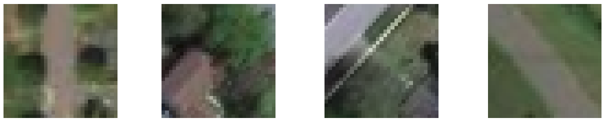

(a) Positive (b) Negative (c) Negative (d) Negative

Figure 4. Samples for the training of our CNN. Positive samples must have a vertical road in the center of the rectified patch.

The inference using CNN is embedded as the function Classify in the minimal path computation in Algorithm 2. The input images of the CNN are rectified patches along local paths, which are provided by the minimal path method.

4.4. Interplay of Minimal Path Method and CNN

The two main components of our method, i.e., the minimal path method and CNNs, benefit from each other to improve the overall centerline segmentation.

Using CNNs, local paths in the minimal path method can be better classified by exploiting strong classifiers to learn features automatically. Also, given sufficient training samples, short cuts of Type 1 and 2 can be detected using a single classifier. However, two difficulties remain: CNNs cannot ensure topology of the centerline. Also, during inference using our method, CNNs need rectified samples as input, instead of axis-aligned samples. These samples must be first generated with other means.

The two difficulties of CNNs are addressed naturally using minimal path methods. Line-topology of the centerlines is ensured as an inherent property of minimal path methods, and rectified patches can be obtained by computing local paths on-the-fly. Rectification provides a strong geometric prior which significantly reduces the degree of freedom of input samples, i.e., the diversity of image patches caused by rotation and curvature is removed, since all paths become straight segments in the canonical orientation. This has two advantages. First, less data is needed to train the CNNs. Second, it is easier for CNNs to classify paths, especially for complex shapes, such as paths with high curvature.

5. Experimental Results

We conducted experiments using four datasets of satellite images, including: EPFL [25] (14 images of roads in suburban areas), ROAD (20 images of roads in rural areas), RIVER (20 images of rivers in the wild), and MRD [17] (a road dataset with over 1000 high-resolution images). Besides centerlines, EPFL contains also width information for roads, so we generated *masks* of roads for this dataset. The images in ROAD and RIVER with centerline annotations were created by ourselves using data from GoogleEarth, since we were not aware of annotated datasets containing roads and rivers in such environments.

Our experiments are divided into two parts. In the first part, we study the classification performance of the CNNs, and illustrate the impact of integrating CNNs into the minimal path framework. In the second part, we apply our method to segment centerlines of tubular structures between user-specified start and end points, and compare our results with results of other minimal path methods and U-Net.

5.1. Classification Performance of CNN

In our method, the CNNs are used in two different ways compared to the usual application of CNNs. First, our CNNs are applied only to *rectified* patches, instead of arbitrary axis-aligned patches. Second, our CNNs are embedded into minimal path methods. In this section, we study the effects caused by these differences.

The EPFL dataset was used for experiments in this section. Samples from 5 images were used as training set, and samples from the other 9 images were used as test set. For each experiment, we used roughly 1000 positive samples and 3000 negative samples for training. The trained CNNs were tested on a test set of about 3000 rectified positive samples and 9000 negative samples. The experiments were repeated for different sizes of the image patches, ranging from 11×11 to 71×71. We tested different CNN architectures as well, including DenseNet121 [11], InceptionV3 [26], MobileNet [10], MobileNetV2 [22], and ResNet50 [9]. A comparison of them is shown in Fig. 5. It turns out that preci-

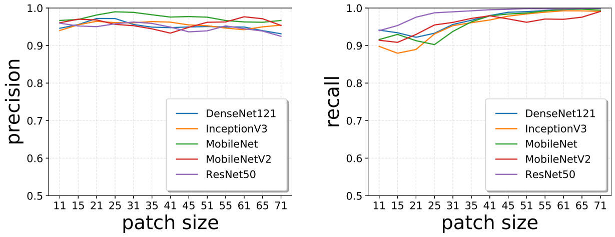
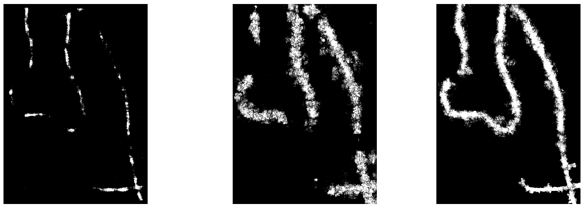


Figure 5. Results of CNNs trained with rectified image patches.



(a) Pixel-based CNN (b) No rectification (c) Rectification

Figure 6. Regions classified as foreground. (a) M_1: Pixel-based classification. (b) M_2: Path classification, trained without rectification. (c) M_3: Path classification, trained using rectified samples.

sion does *not* always improve with increasing patch size, and the classification performance of different architectures does *not* differ significantly. Based on the trade-off between accuracy and speed, we selected MobileNet as our CNN, and consequently the maximum GPU memory consumption during testing is only about 2 GB. We used patches of the size 31 × 31 for all experiments.

We use the image in Fig. 1 from EPFL to compare three methods M_1, M_2, and M_3 for pixel classification. M_1 and M_2 both employ the same classifier C_a trained using axis-aligned patches, while M_3, i.e., our Path-CNN, uses a classifier C_r trained using rectified patches. M_1 applies C_a directly to each pixel individually, while M_2 and M_3 both use our framework and embed the classifier into the Dijkstra's algorithm. With our sparse training data, M_1 results in a large number of gaps, and the tubular structures are difficult to recognize (Fig. 6a). In contrast, by embedding the same CNN C_a into our minimal path method, the F-map of M_2 shows much more geometric details of the roads (Fig. 6b). Trained with rectified patches, the F-map of M_3 contains even less noise and discontinuities in the tubular structures (Fig. 6c). The maps in Fig. 6a, b, and c lead to the blue, green, and red paths in Fig. 1b, respectively.

5.2. Centerline Segmentation

Data The overview of data for our experiments are shown in Tab. 1. For each test path, we set start and end points manually. Our training sets are quite small. For example, for MRD we used patches from 12 images, which in total amount to the size of only 3 full images. In contrast, recent approaches [1, 19] both used 25 images for training and 15 for testing. Our method is trained only using centerlines without width information.

	EPFL	ROAD	RIVER	MRD
Train. images	5	6	6	3 (12)
Test images	9	14	14	49
Test paths	130	64	14	751

Table 1. Overview of data used in experiments.

Baselines We compared our method with five other approaches, including DIJK, DIJK-CNN, PROG, PROG-CNN, and U-Net [21]. DIJK and DIJK-CNN are the standard Dijkstra's algorithm [7] using different tubularity measures: DIJK uses the measure [8], while DIJK-CNN uses MobileNet features (i.e., pixel-wise classification results of a MobileNet like the M_1 method in Sec. 5.1). Similarly, PROG and PROG-CNN are the progressive minimal path method [15] using the measure [8] and MobileNet features, respectively. Like DIJK and PROG, our method also uses the tubularity measure [8] to compute the initial edge weight $\mathbf{w}_{\text{ini}}$. In DIJK and DIJK-CNN, the edge weight remains constant, while PROG and PROG-CNN adapt the edge weight by classifying local paths. But unlike our approach, there is no cropping or rectification in PROG or PROG-CNN. Instead, only mean values of the tubularity measure [8] or MobileNet features on the local paths are used for classification. Similar to the definition of F-map for our method, we define the F-map of PROG and PROG-CNN as regions classified as foreground. DIJK and DIJK-CNN have no classification step, therefore we define the F-map of DIJK and DIJK-CNN as regions for which the distance to the start point will not change anymore (i.e., the region $\mathcal{S}$ in Algorithm 1) when the end point is reached.

Error analysis To measure the errors of the results quantitatively, we used the mean distance between segmented centerlines and the corresponding ground truth, defined as:

$$\bar{e}_d = \frac{1}{N} \sum_{\gamma} \sum_{\mathbf{x}_i \in \gamma} e_d(\mathbf{x}_i) = \frac{1}{N} \sum_{\gamma} \sum_{\mathbf{x}_i \in \gamma} |\mathbf{x}_i - \mathbf{x}_{\text{gt}(i)}|, \quad (3)$$

where N is the total number of points on all segmented centerlines γ, $\mathbf{x}_i$ are the coordinates of points on γ, and $\mathbf{x}_{\text{gt}(i)}$ are the closest points to $\mathbf{x}_i$ on the ground truth. The results are summarized in Tab. 2. For all datasets, our method achieved the lowest mean distances. The errors of DIJK and PROG are higher than our results by a factor of at least 2. DIJK-CNN and PROG-CNN achieve lower errors, which are still significantly higher than those of our results.

In Fig. 7, the errors e_d for points on the segmented centerlines are divided into three intervals, i.e., errors smaller than 5 pixels, between 5 and 10 pixels, and larger than 10 pixels, respectively. The lowest error bound is set to 5 pixels, because this is roughly half the width of the tubular

Dataset	DIJK	PROG	DIJK-CNN	PROG-CNN	Our method
EPFL	4.74 (2.82)	3.36 (2.00)	2.82 (1.68)	2.74 (1.63)	**1.68**
ROAD	6.69 (2.25)	5.98 (2.01)	3.91 (1.32)	3.68 (1.24)	**2.97**
RIVER	19.01 (5.03)	18.43 (4.88)	7.16 (1.89)	6.31 (1.67)	**3.78**
MRD	21.89 (6.10)	18.83 (5.25)	6.17 (1.72)	5.25 (1.46)	**3.59**

Table 2. Errors measured with Eq. (3) in pixels. Numbers in braces indicate ratios between results of the other methods and ours.

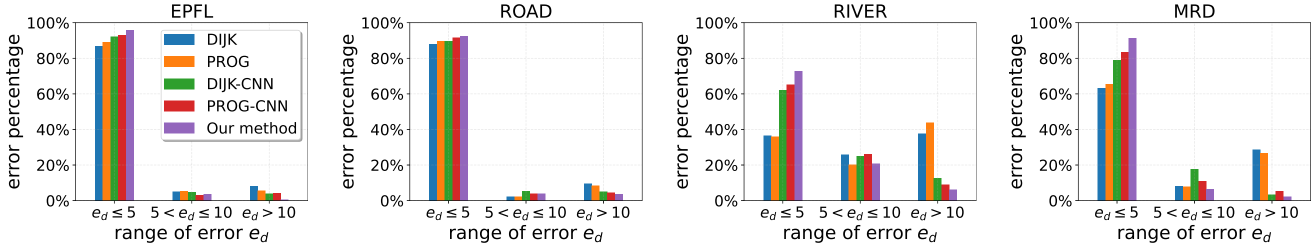


Figure 7. Distribution of errors for centerline segmentation using four datasets.

	U-Net	U-Net	Our method
Annotation type	centerline	mask	centerline
Dice coefficient	0.1163	0.2953	**0.6108**

Table 3. Dice coefficients of U-Net and our method for EPFL

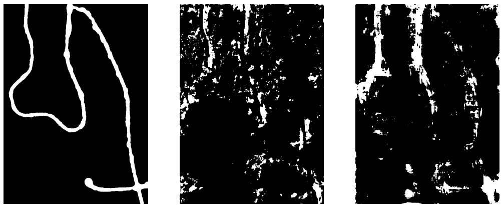
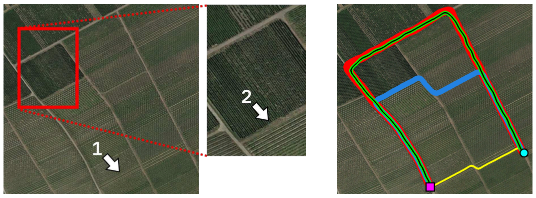


Figure 8. Segmentation results of U-Net for the image in Fig. 1a. Left: Ground truth. Middle: U-Net trained with only centerlines (Dice $= 0.1511$). Right: U-Net trained with masks (Dice $= 0.3521$). Our result is shown in Fig. 6c (Dice $= 0.6658$).



(a) Input image

(b) Results

Figure 9. Results for one image from ROAD. (a) The white arrows show two difficult cases. (b) Results of DIJK, PROG, DIJK-CNN, PROG-CNN and our method are shown in yellow, blue, green, black, and red colors, respectively.

structures in our images. A centerline within this range is considered close enough to the ground truth, since we do not use width information of the tubular structures to train our algorithm. Intuitively, higher percentage of large errors indicate more short cuts. Using our method, most points on the centerlines are within smaller error ranges. For example, among the points on the centerlines segmented by our method in EPFL, 95.78% have errors under 5 pixels, and only 0.52% have errors larger than 10 pixels. In contrast, DIJK, PROG, DIJK-CNN and PROG-CNN have 4% to 8% errors larger than 10 pixels. For RIVER, 72.89% of the errors of our method are under 5 pixels. The percentage of errors over 10 pixels is 6.20% for our method, but 37.71%, 43.88%, 12.71% and 8.92% for DIJK, PROG, DIJK-CNN and PROG-CNN, respectively. Also for MRD, our method has 91.38% errors less than 5 pixels.

As masks for roads are available for EPFL, we computed Dice coefficients for U-Net trained with only centerlines, U-Net trained with masks, and our method. Results for the 9 test images are shown in Tab. 3. Although our method was trained using only centerlines, it achieved the highest Dice coefficient. An example is shown in Fig. 8.

Qualitative results Below we show more results exemplarily. In the figures, the start points and end points are shown as magenta boxes and cyan circles, respectively.

The images in ROAD and RIVER are challenging since there are a lot of tubular structures in the background. Although these structures have slightly different appearance than the actual roads or rivers, it is difficult to use tubularity feature alone to capture the differences. For an image from ROAD, two difficult cases are highlighted in Fig. 9 (white arrows). In case 1, DIJK found a short cut (Type 1), because it is much shorter than the correct path. PROG was able to detect this short cut and avoid it. But in case 2, where the tubularity measure in the background is also high, PROG found a short cut (Type 2). In contrast, DIJK-CNN, PROG-CNN and our approach handled both cases correctly and found the correct centerline. Fig. 10 shows a more challeng-

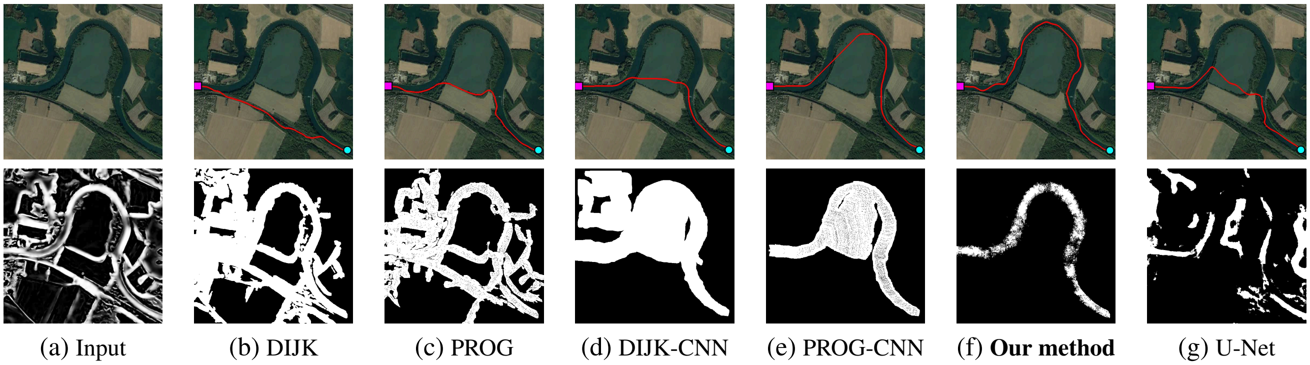


Figure 10. Results for one image from RIVER. (a) Input image and tubularity map. (b) - (f) Results of DIJK, PROG, DIJK-CNN, PROG-CNN, our method (upper row) and the corresponding F-maps (lower row). (g) Results using U-Net.

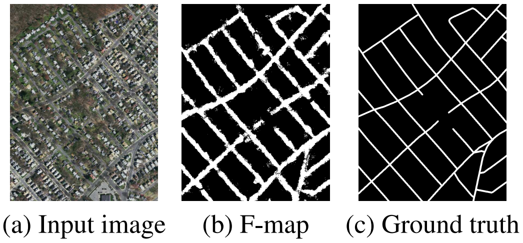


Figure 11. F-map (mask) for a section of an image from MRD. The ground truth contains only centerlines, not masks.

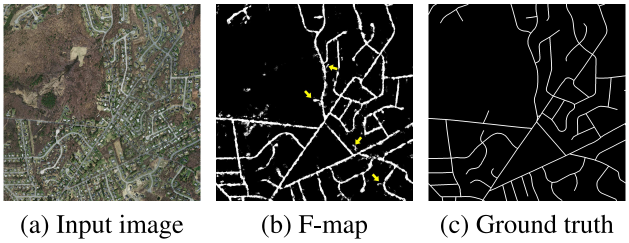


Figure 12. F-map for a full image from MRD obtained using one start point for the entire image. Some roads, which are missing in the ground truth, were detected using our method (yellow arrows).

ing example from RIVER, and only our method successfully segmented the river. The F-maps show that while our method correctly classified most image regions and mostly remained in the foreground, all other approaches explored large portions of background, and resulted in short cuts. We also computed the path by applying DIJK to the mask obtained using U-Net instead of the tubularity map. However, with the small number of samples in our training set (only 9 rivers in the 6 training images), U-Net could not be trained well enough to produce a good mask for foreground. Consequently, the correct path was not found (Fig. 10g).

Images in MRD cover large areas with complex road networks. The roads may appear very differently even within the same image, and they are often occluded by trees or their shadows. Buildings along the roads frequently show certain regular patterns, causing high tubularity outside the roads. There are also tubular structures which are not roads, such as rivers, or narrow spaces between buildings. Fig. 11 shows a section of an image of an urban area. There, our method dealt well with the challenges above, yielding a quite precise F-map (Fig. 11b). *Without re-training*, the same model can be used for roads in other environments. For example, the F-map of a full image of a suburban area is shown in Fig. 12. Our methods even detected several small roads which are missing in the ground truth. To obtain the F-map for all roads in the image, the user only needs to set *one single start point* for the entire image, i.e., no end points or further start points are needed, even if the roads are not connected with each other. F-maps are *not* sensitive to the location of the start point, so similar F-maps can be obtained using different start points.

6. Conclusion

We introduced Path-CNN, a new method for topology-aware centerline segmentation for tubular structures. In our method, a CNN is embedded as an integral component into the progressive minimal path method. The CNN enhances hand-tuned image features to better control the minimal path computation, while the progressive minimal path method provides strong geometric priors to improve the performance of the CNN, and ensures the line-topology of the segmented centerlines. Path-CNN employs path-based classification to avoid different types of short cuts, and consequently centerlines can be better segmented, especially for tubular structures with complex shapes in challenging environments. In addition to centerline segmentation, a binary mask (F-map) is also obtained. Our method only needs sparse and simple annotations for training, and it has lower hardware requirements than many other methods. Its effectiveness is demonstrated using experiments with four datasets of satellite images and comparison with five other methods. Future work includes extension of our method for medical images, especially 3D medical images.

References

[1] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Sam Madden, and David DeWitt. RoadTracer: Automatic Extraction of Road Networks from Aerial Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2, 6

[2] Fethallah Benmansour and Laurent D. Cohen. Fast object segmentation by growing minimal paths from a single point on 2d or 3d images. *Journal of Mathematical Imaging and Vision*, 33(2):209 – 221, 2009. 2

[3] Fethallah Benmansour and Laurent D. Cohen. Tubular structure segmentation based on minimal path method and anisotropic enhancement. *International Journal of Computer Vision*, 92:192–210, 2011. 2

[4] Da Chen, Jiong Zhang, and Laurent D. Cohen. Minimal paths for tubular structure segmentation with coherence penalty and adaptive anisotropy. *IEEE Transactions on Image Processing*, 28(3):1271–1284, Mar. 2019. 2, 3

[5] Laurent D. Cohen and Ron Kimmel. Global minimum for active contour models: A minimal path approach. *International Journal of Computer Vision*, 24(1):57–78, 1997. 1

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (Third Edition)*. MIT Press, 2009. 3

[7] E.W. Dijkstra. A Note on Two Problems in Connection with Graphs. *Numerische Mathematik*, 1:269–271, 1959. 1, 6

[8] Alejandro F. Frangi, Wiro J. Niessen, Koen L. Vincken, and Max A. Viergever. Multiscale vessel enhancement filtering. In *Medical Image Computing and Computer-Assisted Intervention*, 1998. 1, 2, 3, 6

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 5

[10] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *Arxiv*, 2017. 5

[11] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 5

[12] Vivek Kaul, Anthony Yezzi, and Yichang Tsai. Detecting curves with unknown endpoints and arbitrary topology using minimal paths. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 34(10):1952–1965, 2012. 2

[13] Max W.K. Law and Albert C.S. Chung. Three dimensional curvilinear structure detection using optimally oriented flux. In *European Conference on Computer Vision*, 2008. 1, 2

[14] Hua Li and Anthony Yezzi. Vessels as 4-d curves: Global minimal 4-d paths to extract 3-d tubular surfaces and centerlines. *IEEE Transactions on Medical Imaging*, 26(9):1213–1223, 2007. 2

[15] Wei Liao, Stefan Wörz, Chang-Ki Kang, Zang-Hee Cho, and Karl Rohr. Progressive minimal path method for segmentation of 2d and 3d line structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):696–709, Mar. 2018. 2, 3, 4, 6

[16] Gellert Mattyus, Wenjie Luo, and Raquel Urtasun. DeepRoadMapper: Extracting Road Topology from Aerial Images. In *IEEE International Conference on Computer Vision*, 2017. 2

[17] Volodymyr Mnih. *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto, 2013. 5

[18] Volodymyr Mnih and Geoffrey E. Hinton. Learning to detect roads in high-resolution aerial images. In *European Conference on Computer Vision*, 2010. 2

[19] Agata Mosinska, Mateusz Kozinski, and Pascal Fua. Joint segmentation and path classification of curvilinear structures. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 2020. 1, 2, 6

[20] Mickael Pechaud, Renaud Keriven, and Gabriel Peyre. Extraction of tubular structures over an orientation domain. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 2

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241, 2015. 6

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 5

[23] Yoshinobu Sato, Shin Nakajima, Nobuyuki Shiraga, Hideki Atsumi, Shigeyuki Yoshida, Thomas Koller, Guido Gerig, and Ron Kikinis. Three-dimensional multi-scale line filter for segmentation and visualization of curvlinear structures in medical images. *Medical Image Analysis*, 2(2):143–168, 1998. 2

[24] Amos Sironi, Bugra Tekin, Roberto Rigamonti, Vincent Lepetit, and Pascal Fua. Learning separable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):94–106, 2015. 2

[25] Amos Sironi, Engin Tueretken, Vincent Lepetit, and Pascal Fua. Multiscale centerline detection. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 2015. 5

[26] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 5

[27] Engin Tueretken, Carlos Becker, Przemyslaw Glowacki, Fethallah Benmansour, and Pascal Fua. Detecting irregular curvilinear structures in gray scale and color imagery using multi-directional oriented flux. In *IEEE International Conference on Computer Vision*, 2013. 2

[28] Johannes Ulen, Petter Strandmark, and Fredrik Kahl. Shortest paths with higher-order regularization. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 37(12):2588–2600, 2015. 2