Creating Roadmaps in Aerial Images with Generative Adversarial Networks and Smoothing-based Optimization

Dragos Costea[1,2] Alina Marcu[2] Emil Slusanschi[1]
Marius Leordeanu[1,2]
[1] University Politehnica of Bucharest, 313 Splaiul Independentei, Bucharest, Romania
[2] Autonomous Systems, 22 Tudor Vladimirescu Blvd., Bucharest, Romania
{dragos.costea, alina.marcu}@autonomous.ro, {emil.slusanschi, marius.leordeanu}@cs.pub.ro

Abstract

Recognizing roads and intersections in aerial images is a challenging problem in computer vision with many real world applications, such as localization and navigation for unmanned aerial vehicles (UAVs). The problem is currently gaining momentum in computer vision and is still far from being solved. While recent approaches have greatly improved due to the advances in deep learning, they provide only pixel-level semantic segmentations. In this paper, we argue that roads and intersections should be recognized at the higher semantic level of road graphs - with roads being edges that connect nodes. Towards this goal we present a method consisting of two stages. During the first stage, we detect roads and intersections with a novel, dual-hop generative adversarial network (DH-GAN) that segments images at the level of pixels. At the second stage, given the pixelwise road segmentation, we find its best covering road graph by applying a smoothing-based graph optimization procedure. Our approach is able to outperform recent published methods and baselines on a large dataset with European roads.

1. Introduction

Automatic recognition of roads and intersections from aerial images is a challenging problem in computer vision with many applications in UAV localization and navigation. Several approaches have been proposed over the years, but the task is still far from being solved. The presence of occlusions, large paved areas, shadows and the inherent ambiguity of the problem are among the key factors that make the detection of roads and intersections challenging for automated systems. Roads could easily be mistaken for other objects, such as thin constructions or channels of water. One of the main features that distinguishes a road from other objects and regions seen from above, is its structure that covers a larger region with the specific function of connecting one place to another. All roads belong to a network of roads, which could be optimally represented as a graph. Different from many works dedicated to recognition in aerial images that segment images at the level of pixels, we recognize roads and intersections at the level of graphs. The straight road segments become edges, while intersections and high curvature road points become nodes.

Viewing roads as graphs has many practical advantages besides the interesting optimization and theoretical aspects that graphs bring into light. Most of the current methods detect roads at the pixel-level, requiring much more storage than a graph representation. Once the road graphs are recognized, storing them is less costly, while tasks such as navigation become simpler - by efficiently finding paths through graphs that connect the desired locations.

In order to solve road graph extraction we tackle it at two levels. The first is that of pixelwise segmentation of roads, for which we design a deep learning model based on generative adversarial networks (GANs). We propose a novel two-hop GAN architecture – one GAN stacked on top of another – which detects roads at the first hop and intersections at the second one (Section 2). At the second level, a roadmap graph is created by formulating the task as an optimization problem: find the road graph that fits best the pixelwise segmentation under certain constraints. Towards this goal we adapt an efficient smoothing-based optimization (SBO) algorithm [18] (Section 3).

Our main contributions are the following:

1. We introduce a dual-hop generative adversarial net (DH-GAN) for producing pixelwise segmentations of roads and intersection simultaneously at two levels of interpretation. At the first level, roads are extracted. At the second level, roads together with the original input, are passed through the second GAN, which extracts intersections. The full architecture is trained end-to-end, with two discriminators. We show that this architec-

ture outperforms previous work that learn intersections and roads separately.

2. We transform the pixelwise segmentation into roadmap graphs, by extracting nodes and edges, using a smoothing based optimization (SBO) approach, suitable for complex functions that cannot be written explicitly. We conclude, by demonstrating experimentally, that our combined DH-GAN and SBO approach has significant improvements in terms of topology and accuracy over previous segmentation methods and a greedy sampling-based graph extraction baseline.

Related work: Recognizing roads in aerial images has been addressed in the literature mainly by methods that use manually designed features [25, 19, 17, 15, 8]. The success of convolutional neural networks (CNNs) [16, 33, 9] in computer vision has led to greatly improved road detection in recent literature [27, 32]. Other approaches mix aerial images with ground images and LIDAR data to improve accuracy [35]. The lack of good quality aerial images, as well as clutter and occlusion can significantly degrade the recognition performance even for top, state-of-the-art architectures, as shown in [22]. The local, pixel-level results produced often require post-processing in aerial image analysis [25], but generally does not solve the most difficult cases. There are several directions proposed for recognition of roads in aerial imagery, such as following road tracks [10], context modeling using Conditional Random Fields (CRFs) [29], minimum path methods [34] and neural nets [27]. Roads and intersections have also been proven to be successfully used in localization without GPS [5, 2].

Road vectors (graphs) are available for most of the planet. However, they are sometimes misaligned or have a poor level of detail. Some methods attempt to correct the roadmaps by aligning them to real rectified aerial images [24]. Topological road improvement methods trace back to [6]. A more recent approach [29] uses CRFs in conjunction with a minimum cost path algorithm to improve topology, which takes into account cues such as context, presence of vehicles and smoothness between road widths. The same authors previously proposed a metric for topology measurement [36].

Several other methods for road vectorization have been proposed. The approach in [28] starts with multispectral satellite imagery and ends with road vectors. Their method includes texture analysis with manually tuned parameters and genetic guided clustering – used only for creating the road raster map. For generating roadmaps, they employ a genetically guided road vector identification algorithm, followed by a fuzzy shell clustering. They seem to test only on two images. Other work [26] starts from binary road segmentation images and produces road graphs. The number of test images is again small. Finally, the approach in [3] requires raster maps for input and not aerial imagery. Furthermore, the proposed algorithm has no knowledge of high level features such as intersections, and does not have to deal with real problems, such as occlusion, shadows, parking spaces or paved areas.

2. Recognition of roads and intersections

We propose DH-GAN, which consists of two conditional GANs with different roles: one network is capable of predicting pixelwise roadmaps, whilst the other outputs intersection locations. Recently, conditional generative adversarial networks (cGANs) have become a general-purpose solution for image-to-image translation problems [12]. Our task is to translate RGB images into roadmaps and then further translate these predictions into intersection locations. For this purpose, we employ two cGANs: the first GAN learns a road pixelwise segmentation generator G_1 and a discriminator D_1 (that detects G_1's misleading road outputs). Similarly, the second GAN learns an intersections segmentation generator G_2 and discriminator D_2 (that detects G_2's misleading intersections outputs). The second level generator and discriminator have access to both the original RGB as well as the road segmentation output from G_1. The dataflow and overall architecture are illustrated in Figure 1.

2.1. Our Formulation

Similar to [38], we apply adversarial loss for both our mappings ($G_1 : X_1 \rightarrow Y_1$ for roads and $G_2 : X_2 \rightarrow Y_2$ for intersections), each with their discriminators D_1 and D_2. We express the loss for the mapping function $G_1 : X_1 \rightarrow Y_1$ and its discriminator D_1 in 1:

$$\mathcal{L}_{cGAN}(G_1, D_1, X_1, Y_1) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_1(y)] \quad + \\ \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_1(G_1(x))] \quad (1)$$

The same loss function is used for both generators G_1 and G_2, as they share the same objective. Generators are trained to produce indistinguishable outputs from the target samples. The generated samples are verified by an adversary network, the discriminator. While G_1 tries to minimize the objective function, D_1 will try to maximize it in order to detect inconsistencies (i.e. $G^* = arg \min_{G_1} \max_{D_1} \mathcal{L}_{cGAN}(G_1, D_1, X_1, Y_1)$).

The objective of our DH-GAN is to match the distribution of the generated images to the data distribution in the target domain. That is, roads labels for the first hop and intersections for the second. The full adversarial loss used for training our model is expressed in 2:

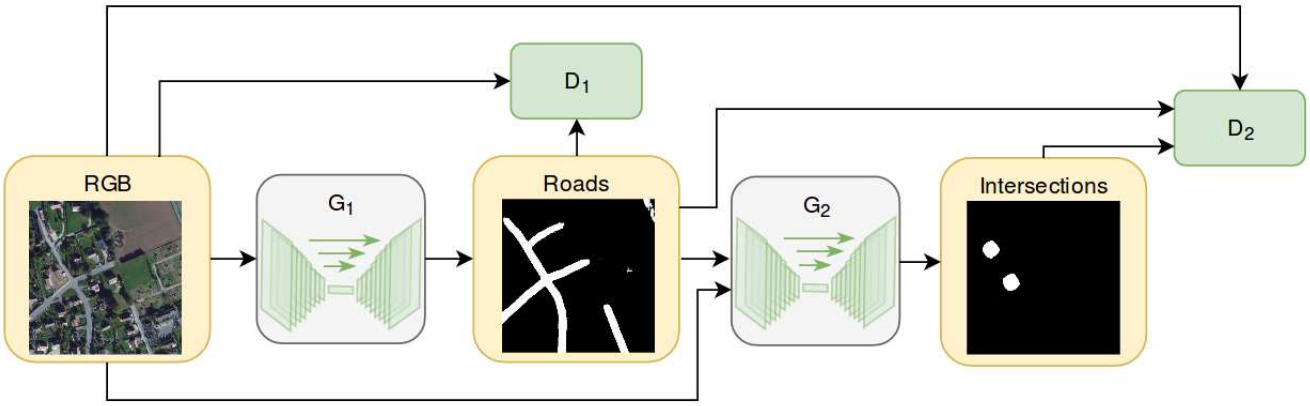

Figure 1. Proposed DH-GAN architecture. The first level conditional GAN learns to produce road segmentations through the generator-discriminator pair (G_1, D_1). The second level GAN, which learns to recognize intersections, is trained simultaneously with the first level in an end-to-end fashion and has access to both RGB input as well as the roads produced by the first level. Our experiments prove that intersections produced by the two level architecture are superior to those learned independently from roads.

$$\mathcal{L}(G_1, G_2, D_1, D_2) = \mathcal{L}_{cGAN}(G_1, D_1, X_1, Y_1) \quad + \\ \mathcal{L}_{cGAN}(G_2, D_2, X_2, Y_2) \quad (2)$$

2.2. Implementation

In our experiments, we used the same network architectures for G_1 and G_2, namely an adapted version of "U-net" [31]. Similarly, for D_1 and D_2, we employed a variant of "PatchGAN" [12]. Finally, we followed the guidelines in [30] for designing our networks.

Generator Architecture. We used a fully convolutional encoder-decoder network. The encoder consists of 9 downsampling modules (stride-2 convolution followed by Batch Normalization [11] and Leaky ReLU [21]). Given an input image of 512x512 pixels and 64 filters (for the first convolution), we reduce the spatial dimension by a factor of two and double the number of filters after each downsampling module up to 512 filters, until a bottleneck layer of size 1x1x512. This representation results in the loss of high frequency information. We solve this problem using skip connections, as previously addressed in [12, 37].

The decoder mirrors the encoder, but uses fractionally-strided convolutions with stride $\frac{1}{2}$ to upsample the feature maps up to the size of the original image. Skip connections concatenate the feature maps of the encoder with the corresponding channels of the decoder.

Discriminator Architecture. We use PatchGAN, a fully convolutional network with 5 downsampling modules, as explored in [12]. We determined that an increase in the number of parameters of the discriminator results in a very small performance gain.

Training details. We optimize our negative log-likelihood objective using mini-batch stochastic gradient descent and apply the Adam solver [13], with a learning rate of 0.0002 and momentum of 0.5. The weights are initialized with values sampled from a normal distribution with zero-mean and standard deviation of 0.02. We trained our models for 200 epochs and determined the 60th to be the best - overfitting was noticeable after this threshold. We used Torch [4] for training and testing our architecture, on a Tesla K40 GPU. Our architecture is trained from scratch end-to-end.

3. Extracting roadmaps

We propose a method of extracting a graph from raster road images. We show that this not only increases road precision, especially at a topological level, but also greatly reduces the storage requirements for a specific image. Before we present our full method, we first present in more detail our graph formulation and then lay down the steps of a simpler and effective baseline for finding such road graphs.

Graph formulation: Let us represent a roadmap by a graph $G = (V, E)$, where V is a set of nodes i, each with an associated position (x_i, y_i) on the road map image and E represents a set of edges $(i, j) \in E$ that have associated a straight line segment l_{ij} in the image, between locations (x_i, y_i) and (x_j, y_j). This formulation, also known as a vector representation, is an approximation that is very useful for efficiently storing, displaying and working with roadmaps. The pixelwise road segmentation M with fixed road width can be immediately recovered (approximately) from this vector representation, by drawing the nodes and the edges and dilating them accordingly, to obtain the desired road width. Since edges in the graph correspond to straight road segments in the image, the nodes correspond either to road points with nonzero curvature or to intersections.

Now we define two functions that are useful for evaluating the quality of a graph representation with respect to the pixelwise representation. It is clear that the two are not identical and that the graph is only an approximation of the pixelwise segmentation. The first measure we propose is to evaluate edges. Let $c(i, j)$ be the cost of edge (i, j) defined as the average distance, along the edge, to the nearest road point in the segmentation map M. This cost is intuitive, it measures how far on average is a given edge from the actual road map M. This measure, which is related to the classic Chamfer distance, can be quickly computed with a standard distance transform. The second measure, denoted $S(V)$, evaluates the quality of the overall graph. It evaluates how well the road vector fits the pixelwise raster map M. We define $S(V)$ to be the standard intersection over union score (IOU), between the graph drawn on the map with dilated edges (according to a given road width) and the pixelwise binary map M.

3.1. Sampling approach

We first present our greedy baseline method for roadmap extraction from the road segmentation map M. It is an efficient approach and a simpler alternative to the smoothing-based optimization approach presented in 3.2. We start by thinning the road segmentation M obtained from the dual-hop GAN architecture. Initial graph nodes are then sampled along the thinned roads at equal intervals, with a fixed length l_s that is found using a small validation set, such that the number of nodes is kept small, while the overall score $S(V)$ remains close to the maximum. As expected, that maximum $S(V)$ is obtained when maximum number of points is used and nodes are neighbors in the image. Therefore, we need to make a trade-off between accuracy and computation and storage costs.

Edges are then established between any two nodes that are closer than a specific threshold distance l_e and have the edge cost $c(i, j) < \theta_e$. Both thresholds l_e and θ_e are chosen to maximize the F-measure w.r.t to the OSM [1] ground truth road map on the validation set. We also make sure that links do not lay over others by a simple procedure which connects node pairs in the increasing order of the distance between them - we do not allow links that overlap with existing ones. We then search for collinear points and remove those nodes and their links, which are replaced by a single line segment. We present the main steps of our method in Algorithm 1.

The resulting method gives good results, with a few drawbacks: since it is a greedy procedure it tends to either over-connect nodes (introducing unnecessary links) or miss connections and brake roads, depending on the fixed threshold parameters chosen. It is clear that for improved results, we need an optimization procedure that searches for the optimal roadmap in an iterative fashion. That is because there is no way to know the optimal number and locations of nodes and their links by reasoning at the local level in a greedy fashion.

Algorithm 1 Roadmap extraction baseline

1) Obtain thinned roads from the initial segmentation M.
2) Sample points at equal distances l_s on thinned roads and initialize road graph vertices V with the sampled points.
3) Initialize edges in the increasing order of their lengths. Accept an edge (i, j) only if its cost $c(i, j) < \theta_e$ and does not overlap with existing ones.
4) Remove collinear points and their edges. Add the necessary edges between the endpoints of the corresponding road segment.
5) Mark as intersections nodes that are close to the intersections found by DH-GAN or those that belong to more than two edges (e.g. $deg(i) > 2$).

3.2. Roadmap optimization

Finding the optimal roadmap by reasoning at a global level requires the optimization of a complex nonlinear function $S(V)$. The graph nodes (number and locations) and edges are not known and the overlap over union measure is a relatively sophisticated function that cannot be expressed in closed form w.r.t to the graph parameters. At the same time, the number of nodes and edges should be kept to a minimum in order to obtain an efficient representation. Therefore, the optimization task, like many in computer vision, is very challenging and computationally expensive.

Two popular algorithms used for such problems are Markov Chain Monte Carlo (MCMC) [7] and Simulated Annealing (and their variants) [14]. While these methods theoretically find the global optimum, in practice they lack efficiency and require a very large number of samples. Here we take a more efficient strategy and adapt to our case the relatively recent smoothing-based optimization (SBO) method [18]. The algorithm is applicable to the maximization of non-negative functions, for which the only requirement is the ability to evaluate the function at a given point. No closed-form formula is required. Moreover, the function does not need to be differentiable or smooth. The key insight behind SBO is that searching for the optimum of a function can be achieved by looking for the maximum through the scale space of that function [20]. We discuss some theoretical aspects here in order to better explain the usage of SBO in our case. We refer the reader to [18] for more theoretical insights.

Smoothing-based optimization: For a given non-negative multi-dimensional function $f : \mathbb{R}^n \rightarrow \mathbb{R}+$ its (smooth) scale space function is defined as $F(\mu, \sigma^2 I) = \int g(x; \mu, \sigma^2 I) f(x) dx$, where g is a multidimensional Gaus-

sian (of dimension n) with mean μ and covariance matrix $\sigma^2 I$. In our case, the function is $S(V)$ and at a given time step t in the optimization process $\mu^{(t)}$ represents the value of the parameter (w.r.t which we optimize). Along with $\sigma^{(t)}$ it also represents the parameters of the Gaussian distribution from which we sample the next values of the parameters. The following update rules for (μ, σ) indicate both how the parameters changes from one step to the next as well as how the Gaussian distribution changes: (i represents dimension indices and $g^{(t)}(x) = g(x; \mu^{(t)}, \sigma^{(t)})$):

1. $\mu^{(t+1)} = \frac{\int x g^{(t)}(x) f(x) dx}{\int g^{(t)}(x) f(x) dx}$.

2. $\sigma^{(t+1)} = \sqrt{\frac{1}{n} \frac{\int (\sum_{i=1}^{n} (x_i - \mu_i)^2) g^{(t)}(x) f(x) dx}{\int g^{(t)}(x) f(x) dx}}$.

These updates ensure that the values of the smoothed function (the scale space function) increases from one iteration to the next. The algorithm converges when σ goes to zero and a maximum of the original function f is found. The integrals in the updates are approximated through sampling from the Gaussian at that step.

We adapt SBO for our use case. We start with an initial graph V_{t0} at time $t0$ over which we apply SBO to optimize $S(V_{t0})$ w.r.t the nodes locations. The initial nodes are given by the intersections found by DH-GAN and edges between them are added only if $c(i, j) < \theta_e$ and are long enough ($l_{ij} > l_e$). At the next iteration we add new nodes (as midpoints of existing edges) and optimize w.r.t their locations. Again valid edges are added only if they pass the same tests and do not overlap with existing edges. We continue by applying SBO again on the nodes locations of the newly updated graph. The method stops when there is no improvement in the $S(V)$ score. It is easy to show that the method improves $S(V)$ from one iteration to the next, since SBO improves it at each iteration and the addition of new nodes and edges is guaranteed to not lower the existing $S(V)$ score. The addition of nodes does not change the scores (since they are added as midpoints of existing edges) and new edges are not added if $S(V)$ decreases. The algorithm for road vectorization is summarized in Algorithm 2 and the representative steps are shown in Figure 2.

It is possible that after the optimization, several road parts (in the segmentation M) remain undiscovered (since they were not connected to any initial intersection). In order to tackle this problem, we apply the baseline algorithm for the road leftovers and create a new subgraph for each one. We then try to join the subgraph to the main graph, if there is at least an edge connecting the main graph that passes the tests ($l_{ij} > l_e$, $c(i, j) < \theta_e$ and $S(V)$ does not decrease). At the end, all nodes locations (including the sampled ones) are optimized with SBO. This method could help the discovery of new intersections, as later confirmed by the experimental results.

Algorithm 2 Graph and intersection extraction with SBO

1) Initialize V with the detected intersections.
2) Initialize E with valid edges: $c(i, j) < \theta_e$, $l_{ij} < l_e$.
3) Remove collinear points and edges. Add the corresponding line segments.
4) Optimize locations $[\mathbf{X}, \mathbf{Y}]$. with SBO:
$[\mathbf{X}(t+1), \mathbf{Y}(t+1)] \leftarrow \operatorname{argmax} S(V(\mathbf{X}(\mathbf{t}), \mathbf{Y}(\mathbf{t})))$.
repeat
  Add a new node q as an edge midpoint.
  Optimize with SBO: $[x_q, y_q] \leftarrow \operatorname{argmax} S(V(x_q, y_q))$.
  Keep midpoint if the score improves.
until Edges lengths $> l_e$
5) If valid edges exist add them and go to Step 4. Otherwise go to Step 6.
6) Use the greedy baseline on isolated roads. Then connect the new sub-graphs to G if valid links exist between them.
7) Final optimization: $[\mathbf{X}, \mathbf{Y}] \leftarrow \operatorname{argmax} S(V(\mathbf{X}(\mathbf{t}), \mathbf{Y}(\mathbf{t})))$.
8) Mark as intersections nodes that are close to the intersections found by DH-GAN or those that belong to more than two edges $deg(i) > 2$.
9) Return intersections and G with its node locations and edges.

4. Experiments

We test our methods and compare to recent approaches on the publicly available European Road Dataset [22]. This image set offers large variations in roads complexity and structure. It covers both rural and urban European areas and contains many difficult cases that have partial occlusions and illumination changes. The dataset contains 200 satellite images (aprox. 276.4 km^2) for training, 20 images (27.7 km^2) for validation and 50 images (70 km^2) for testing. The images were automatically aligned with road maps from OpenStreetMap (OSM) [1] to obtain the ground truth labels for both roads and intersections. From the 200 1600x1550 RGB training images, we cropped 3200 RGB 512x512px patches for training our DH-GAN. The spatial resolution is about 1 m^2 per pixel.

4.1. Road detection

We evaluate the road detection task using two measures: the first is the standard maximum F-measure (F1-score) computed on the pixelwise road segmentation vs. ground truth and the second is the topological accuracy score proposed in [36] (see Tables 1 and 2).

The topological score is meant to measure the degree at which the connectivity of the roads found agrees with the connectivity of the ground truth. It is computed as fol-

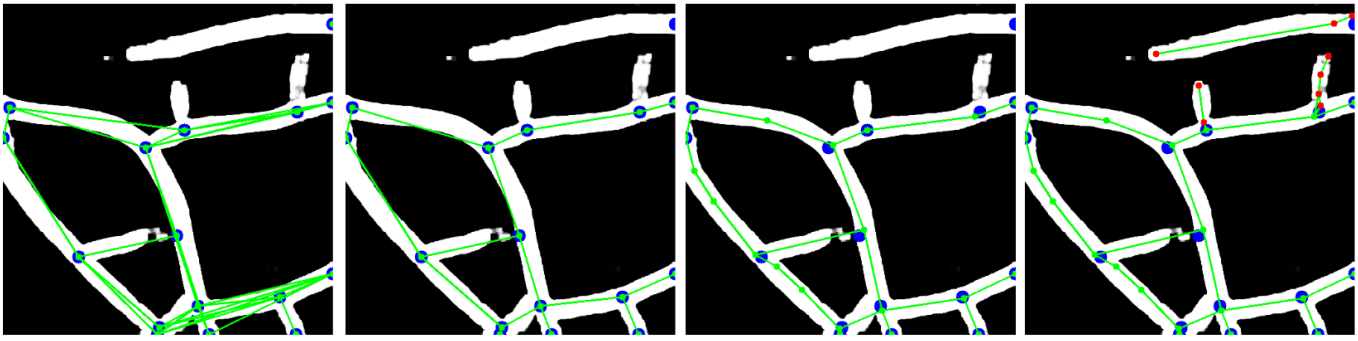

Figure 2. Intermediate results of road extraction with SBO: A - initial intersections (blue) and edges (green), B - pruned edges (after removing collinear and overlapping points), C - edges and intersections optimized with SBO (green edges with green vertices), D - sampled leftover roads (green edges with red vertices) connected to the roads optimized with SBO.

lows (see [36] for details): pairs of random points detected in both ground truth and automatic output are sampled. If the path connecting the points have similar lengths (within 5%) in both automatic output and ground truth, the pair is labeled "Correct". If there is no link in one of the pairs, the label is as "NoConn", otherwise the label is "2long" or "2short" depending on whether the path between the points in the automated output is longer or shorter than their path in the ground truth. These labels capture the cases when the detected roads have been broken (which results in "2long") or have been incorrectly connected ("2short").

In terms of pixelwise detection, both versions of our method – DH-GAN as well as the combination with the roadmap creation – outperform several other popular convolutional neural networks tailored for semantic segmentation (LG-Seg-RestNet-IL [23], U-net [31] and GAN [12]). One advantage of detecting roads as graphs is the low storage cost, with minimum accuracy and topology loss. In fact, roads generated from the created graphs look better in terms of connectivity. They now support any operation over graphs, from finding minimum paths between different locations to finding specific road patterns by graph matching. In terms of accuracy, the method that combines DH-GAN segmentation with SBO for creating roadmaps is superior to the greedy baseline in all categories. We noticed that the baseline tends to break roads. DH-GAN with SBO, in turn, is less accurate than DH-GAN (Tables 1 and 2). This is expected, as DH-GAN is trained to predict the ground truth at the pixel-level, without further simplifications. On the other hand, DH-GAN+SBO is significantly less costly in terms of storage (see Table 4). However, we believe that in future work, interpreting roads as graphs has the potential to improve pixel-level accuracy as well.

4.2. Detecting intersections

We test intersection detection (Table 3) in two cases - first, we use intersections from OSM for both training and testing the models. All OSM maps are made of nodes, edges and relations, thus being efficient for road storage. OSM does not have a node descriptor labeled 'intersection', so we considered the points where two roads meet as an inter-

Method	F-measure
GAN [12]	77.70%
LG-Seg-ResNet-IL [23]	81.06%
U-net [31]	79.79%
DH-GAN	**84.05%**
DH-GAN + Greedy	80.81%
DH-GAN + SBO	81.74%

Table 1. Road detection results. Higher is better.

Method	Correct	2long	2short	NoConn
GAN [12]	85.80 %	2.00 %	2.23 %	9.97 %
LG-Seg-ResNet-IL [23]	39.85 %	0.60 %	5.17 %	54.38 %
U-net [31]	74.03 %	**0.72 %**	3.34 %	21.91 %
DH-GAN	**89.84 %**	0.75 %	**1.80 %**	**7.61 %**
DH-GAN + Greedy	68.82 %	1.16 %	6.13 %	23.89 %
DH-GAN + SBO	76.98 %	1.14 %	7.9 %	13.98 %

Table 2. Topological accuracy score. Higher is better for 'Correct', lower is better for the other columns.

Labeling Type	Method	F-measure
OSM	GAN [12]	54.89%
	DH-GAN	**63.01%**
	DH-GAN + Greedy	31.81%
	DH-GAN + SBO	59.79%
Independent	GAN [12]	64.42%
	DH-GAN	82.65%
	DH-GAN + Greedy	43.42%
	DH-GAN + SBO	**86.00%**

Table 3. Intersection detection results. Higher is better.

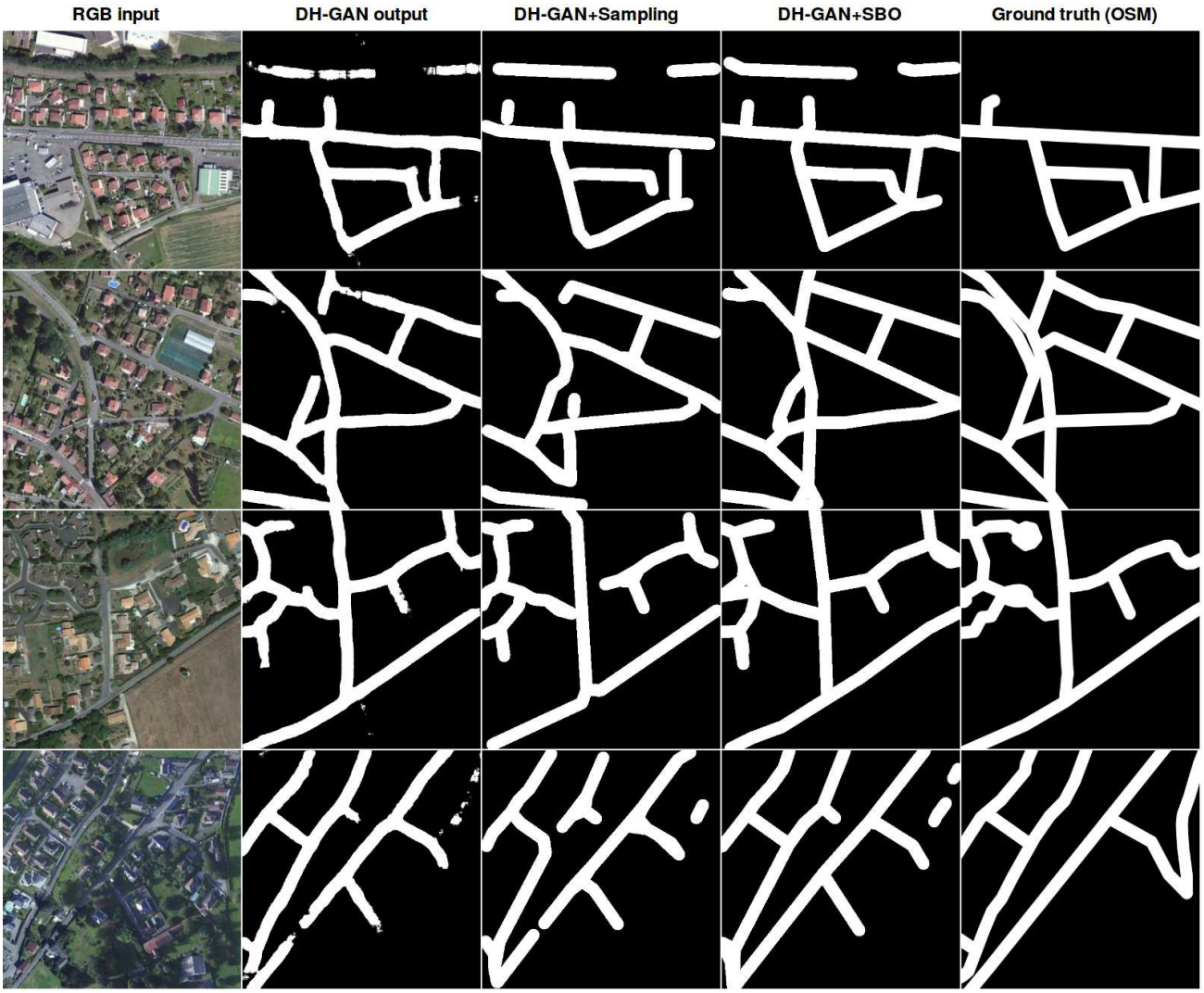

Figure 3. Qualitative results for road detection and map generation using graphs. Sampled and SBO roads are drawn from the graph representation, given a road thickness. Roads extracted with greedy sampling tend to be broken near intersections, while the ones extracted with SBO are topologically correct.

section. We realized that the intersections taken from OSM in this manner were not always correct. This is due to the context needed in order to interpret the presence of an intersection. For example, most failure cases were on intersections close to the image border, where it was impossible to know if it was a road turn or really an intersection. To better evaluate the ability of the models to detect intersections, we hand-labeled a separate ground truth set for testing (the training remained the OSM ground truth in all cases). The performance improved dramatically, even though only the ground truth at test time changes (see results marked with 'Independent' in Table 3). For a better understanding of the differences between the two different ground truth intersections vs. the automatically detected ones with DH-GAN, see Figure 4 .

We also tried to improve intersection detection by using the optimization-based algorithm: we combined (set union) the intersections found by DH-GAN with the graph nodes of degree at least three. Table 3 shows our results. The performance improved on the new ground truth test set, while it slightly decreased on the OSM ground truth. This is again expected, as DH-GAN was trained on the OSM style ground truth. Also note that when a separate single-hop GAN was used for learning intersections (independently of the road detection GAN), the performance was poorer than for the DH-GAN models. This further justifies our choice of the two-hop DH-GAN, at two levels of interpretation, such that the roads become context for the recognition of intersections.

Poor quality labels significantly affect the performance

Method	Vertices	Edges
LG-Seg-ResNet-IL [23]	782*	-
DH-GAN	1345*	-
DH-GAN + Greedy	23	21
DH-GAN + SBO	**19**	**17**
OSM (ground truth)	29	31

Table 4. Average storage cost. Lower is better. The starred vertices are obtained by thinning roads from raster segmentations to a single pixel width and counting all remaining pixels. Since we only have a binary image, no edges are present. Apart from our two graph extraction methods, we also compare with the road vertices and edges from OSM (ground truth).

of all models. We stress out that all training was performed on OSM ground truth. W e expect further boost in performance by training with better quality ground truth.

Qualitative results: In Figure 3, we present the RGB imagery next to the detected roads using our models and the ground truth. We notice that the baseline is good at tracking the roads' center, it has noticeable issues around intersections. It also tends to break roads. DH-GAN+SBO makes a better trade-off between accuracy and efficiency. It uses fewer number of points and edges and it matches at pixel-level, the performance of DH-GAN. In terms of overall structure, it produces roads with a realistic look, which are also well connected. The binary maps are obtained in the cases of DH-GAN+Greedy and DH-GAN+SBO by thickening the edges of the graphs with a fixed width.

Computational cost: The average forward time for a 512x512px image is 110ms for both roads and intersection extraction using a Tesla K40 GPU. We also tested on the Jetson TX2 embedded development platform, which yielded 167ms. We believe it would be manageable to generate raster roads and intersections in near real-time, assuming 1 m^2 per pixel spatial resolution and a reasonable flight speed.

The greedy road graph creation baseline is fast (about 200ms for a 512x512px image) and less accurate, while the SBO-based approach is reasonably fast (a couple of seconds for a 512x512px image on a desktop PC) and more accurate. However, SBO is not optimized at neither logical nor software levels. Future optimization would make our algorithm tractable on current generation embedded hardware.

Storage cost: Compared to storing the road or RGB imagery, vectorization provides a memory-efficient solution, by encoding only a few points and the relations between them. This is often preferred on an UAV, where storage or transmission capabilities are limited. Furthermore, our optimization approach additionally helps to keep meaningful information, by removing the redundant points, while maintaining the accuracy (see Table 4). Stored vertices use an average of 70 times less space compared to the average size of the compressed grayscale road pixel-level segmentations.

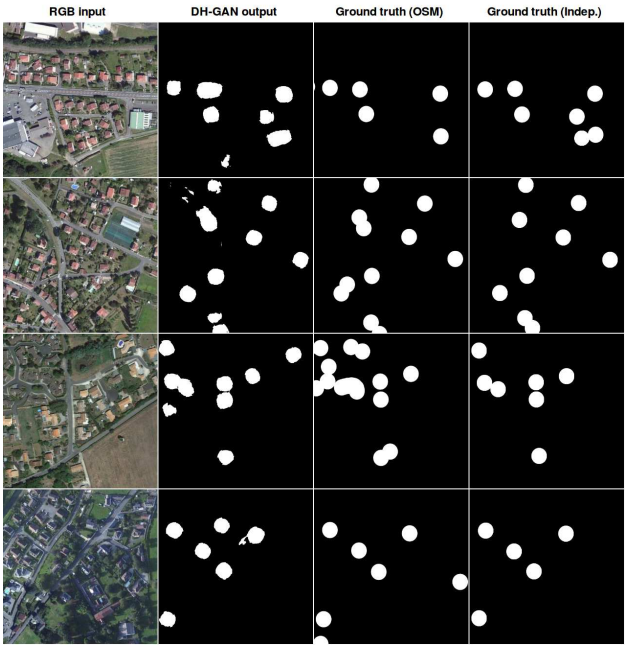

Figure 4. Intersections detection results. OSM intersections near image edges are very hard to detect due to missing context. Independent labeled ones increase the F1-score by a solid 26% in conjunction with SBO.

5. Conclusions and Future Work

We introduced an efficient approach for creating vector roadmaps from aerial images. We first obtain accurate representations by producing pixelwise segmentation of roads and intersections with a novel two-level generative adversarial network, DH-GAN, trained end-to-end. The following stage extracts road graph structure and nodes locations using smoothing-based optimization. We successfully combine aspects from two different research areas in vision, namely deep learning and nonlinear discrete-continuous graph optimization. We validate our method on a large dataset and show improvements over previous methods and baselines. Both our pixel-level model DH-GAN and our graph level method, DH-GAN+SBO, produce top quality results. DH-GAN+SBO is also able to reduce memory costs and provide a useful road graph representation suitable for any graph search or matching algorithm. In the future we plan to automatically improve existing vertex-based maps or create new ones in regions where road information is unavailable. We also plan to improve localization by efficiently using graph representations for matching. We aim to streamline our method for on-board UAV navigation.

Acknowledgements: This work was supported in part by UEFISCDI, project PN-III-P4-ID-ERC-2016-0007 and the Romanian Ministry of European Funds, project IAVPLN POC-A1.2.1D-2015-P39-287.

References

[1] Openstreetmap. https://www.openstreetmap.org. Accessed: 2017-07-01.

[2] T. Ayoul, T. Buckley, and F. Crevier. Uav navigation above roads using convolutional neural networks. Technical report, Stanford University, 2017.

[3] Y.-Y. Chiang and C. A. Knoblock. An approach to automatic road vectorization of raster maps. In *Proc. GREC*, 2009.

[4] R. Collobert, S. Bengio, and J. Mariéthoz. Torch: a modular machine learning software library. Technical report, Idiap, 2002.

[5] D. Costea and M. Leordeanu. Aerial image geolocalization from recognition and matching of roads and intersections. In *BMVC*, 2016.

[6] P. Gamba, F. Dell'Acqua, and G. Lisini. Improving urban road extraction in high-resolution images exploiting directional filtering, perceptual grouping, and simple topological concepts. *Geoscience and Remote Sensing Letters, IEEE*, 3(3):387–391, 2006.

[7] W. R. Gilks, S. Richardson, and D. Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

[8] A. Gruen and H. Li. Road extraction from aerial and satellite images by dynamic programming. *ISPRS Journal of Photogrammetry and Remote Sensing*, 50(4):11–20, 1995.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] J. Hu, A. Razdan, J. C. Femiani, M. Cui, and P. Wonka. Road network extraction and intersection detection from aerial images by tracking road footprints. *Geoscience and Remote Sensing, IEEE Transactions on*, 45(12):4144–4157, 2007.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[13] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[15] D. Klang. Automatic detection of changes in road data bases using satellite imagery. *International Archives of Photogrammetry and Remote Sensing*, 32:293–298, 1998.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] I. Laptev, H. Mayer, T. Lindeberg, W. Eckstein, C. Steger, and A. Baumgartner. Automatic extraction of roads from aerial images based on scale space and snakes. *Machine Vision and Applications*, 12(1):23–31, 2000.

[18] M. Leordeanu and M. Hebert. Smoothing-based optimization. In *CVPR*, 2008.

[19] Y. Lin and S. Saripalli. Road detection and tracking from aerial desert imagery. *Journal of Intelligent & Robotic Systems*, 65(1-4):345–359, 2012.

[20] T. Lindeberg. Scale-space behaviour of local extrema and blobs. *Journal of Mathematical Imaging and Vision*, 1(1):65–99, 1992.

[21] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

[22] A. Marcu and M. Leordeanu. Dual local-global contextual pathways for recognition in aerial imagery. *arXiv preprint arXiv:1605.05462*, 2016.

[23] A. Marcu and M. Leordeanu. Object contra context: Dual local-global semantic segmentation in aerial images. *AAAI Workshops*, 2017.

[24] G. Mattyus, S. Wang, S. Fidler, and R. Urtasun. Enhancing road maps by parsing aerial images around the world. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[25] H. Mayer, S. Hinz, U. Bacher, and E. Baltsavias. A test of automatic road extraction approaches. *International Archives of Photogrammetry, Remote Sensing, and Spatial Information Sciences*, 36(3):209–214, 2006.

[26] J. B. Mena. Automatic vectorization of segmented road networks by geometrical and topological analysis of high resolution binary images. *Knowledge-Based Systems*, 19(8):704–718, 2006.

[27] V. Mnih and G. E. Hinton. Learning to detect roads in high-resolution aerial images. In *Computer Vision–ECCV 2010*, pages 210–223. Springer, 2010.

[28] M. Mokhtarzade, M. V. Zoej, and H. Ebadi. Automatic road extraction from high resolution satellite images using neural networks, texture analysis, fuzzy clustering and genetic algorithms. In *The international archives of the photogrammetry remote sensing and spatial information sciences 2008 Proceedings ISPRS Congress Beijing, B3b*, volume 549, 2008.

[29] J. A. Montoya-Zegarra, J. D. Wegner, L. Ladickỳ, and K. Schindler. Mind the gap: modeling local and global context in (road) networks. In *Pattern Recognition*, pages 212–223. Springer, 2014.

[30] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[31] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597*, 2015.

[32] S. Saito and Y. Aoki. Building and road detection from large aerial imagery. In *IS&T/SPIE Electronic Imaging*, pages 94050K–94050K. International Society for Optics and Photonics, 2015.

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] E. Türetken, F. Benmansour, and P. Fua. Automated reconstruction of tree structures using path classifiers and mixed integer programming. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 566–573. IEEE, 2012.

[35] S. Wang, M. Bai, G. Mattyus, H. Chu, W. Luo, B. Yang, J. Liang, J. Cheverie, S. Fidler, and R. Urtasun. Torontocity: Seeing the world with a million eyes. *arXiv preprint arXiv:1612.00423*, 2016.

[36] J. Wegner, J. Montoya-Zegarra, and K. Schindler. A higher-order crf model for road network extraction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1698–1705, 2013.

[37] Z. Yi, H. Zhang, P. T. Gong, et al. Dualgan: Unsupervised dual learning for image-to-image translation. *arXiv preprint arXiv:1704.02510*, 2017.

[38] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.