An Improved Regression-Based Method for Centerlines and Boundary Detection

Amos Sironi[1], Vincent Lepetit[2], and Pascal Fua[1]

[1]Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne (EPFL)
[2]Institute for Computer Graphics and Vision, Graz University of Technology
{firstname.lastname}@epfl.ch, lepetit@icg.tugraz.at

1. Introduction

In this work we present an extension of the regression-based method presented in [27] for centerline and boundary detection.

As shown in [27], classification based approaches are inaccurate when applied to centerline detection of linear structures. The reason is that centerline points and locations immediately next to them are extremely difficult to distinguish,

The solution proposed in [27] is to reformulate centerline detection as a regression problem. Instead of the binary function used by classification-based approach, the authors approximate a function designed to respond maximally along the centerlines and with smaller values on other pixels, depending on their distance to the centerlines.

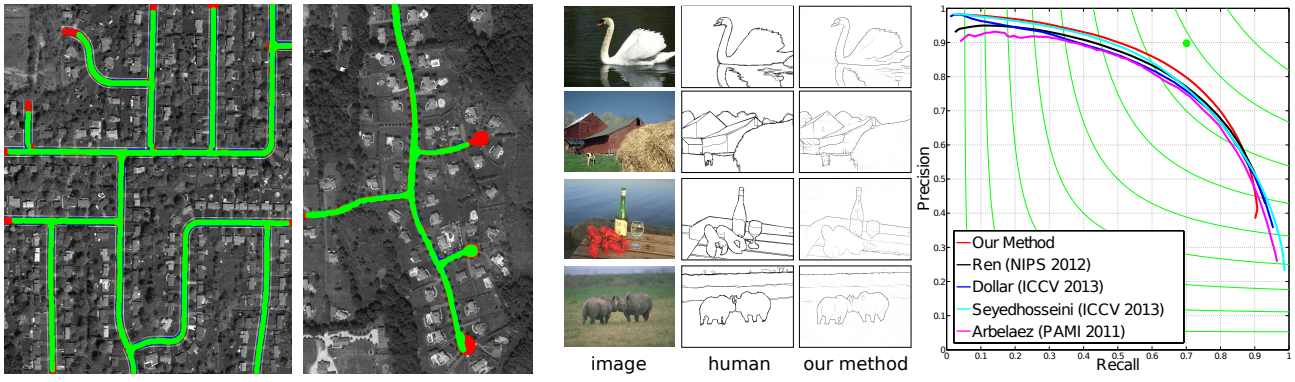
The idea of using regression instead of classification is general and can be applied to other problems with similar properties. Contour detection is one of these.

Consider Fig. 2, where are shown the boundaries marked by 5 human subjects on a sample image. Even if the 5 humans most of the time agree on the detection of a boundary, the exact localization of this boundary is not precise and the annotations are distant by few pixels.

This uncertainty can be caused by several factors, like for example lack of local information, low resolution, blurring and other image artifacts.

Because of the ambiguity of pixels in a neighborhood of a boundary and their extremely similar aspect, it is not clear if pixels close to boundaries should be considered as positive or negative samples. As for centerline detection, training a classifier to distinguish boundary points from the rest may produce multiple responses on the boundaries and low localization accuracy.

In this work we use an auto-context like approach to improve the performance of [27]. We test our method on the following two problems: 1. detecting road's centerline in 2D aerial images; 2. contour detection in natural images. We



(a) (b)

Figure 1. Centerline and boundary detection results. (a) Results obtained using our method as input to the algorithm of **??** to obatin reconstruction of road images; (b) Qualitative results and Precision-Recall curves obtained on BSDS500 dataset, by training our method to detect boundaries. Our method outperform state-of-the-art techniques.

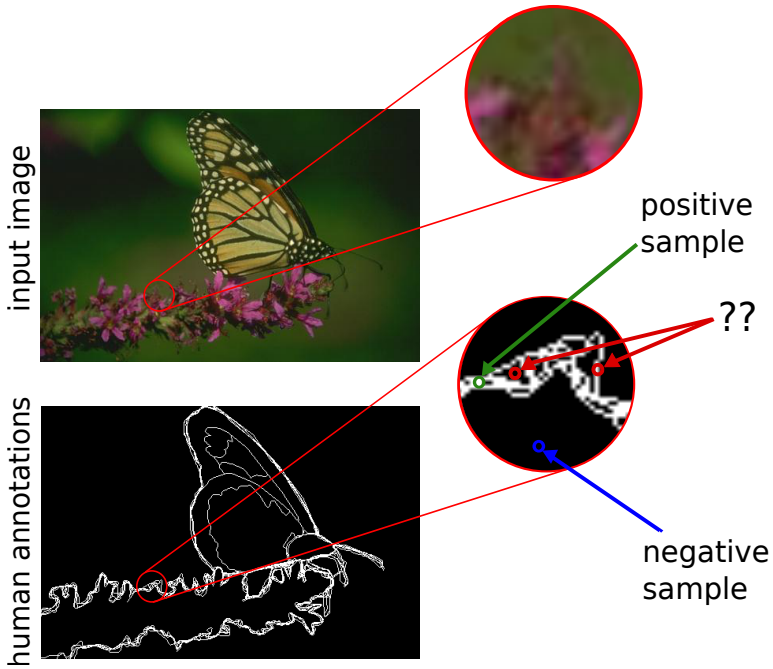


Figure 2. As for centerline detection, classification is not the right formalism to use to solve the boundary detection problem. The exact localization of a boundary is not well defined because of low resolution, blurring and low contrast. Applying classification to distinguish boundary pixels from the background do not take into account localization uncertainty and errors in the ground truth. In the detail of this sample image, the blue pixel lyes on the background and is clearly a negative sample. The green pixel instead can be considere ad positive, since it is marked by all human subjects as boundary point. However, if we consider the red pixels, it is not clear if they are positive or negative samples. Their aspect is really similar to boundary points but only one response for each boundary should be returned. In these situations even the human subjects are not sure about the exact localization of the boundaries.

evaluate performance on a challanging aerial dataset and on the Berkeley BSDS500 [1] dataset and show state-of-the-art results in both cases.

Our approach is more robust in detecting centerlines and boundaries and less sensitive to errors or small displacement in the ground truth. It also ensures a single and well localized detection for each centerline or boundary point. As a consequence, the centerlines can be then easily extracted by post processing operation, typically non-maxima suppression.

Moreover, we show that our approach in combination with a reconstruction algorithm [30] can correctly recover roads in aerial images improving the performance of existing methods. Fig. 1(a) shows the results of road tracking on two test images, while in Fig. 1(b) are shown the results on the BSDS500 dataset and the comparison of the performance with state-of-the-art techniques.

2. Related work

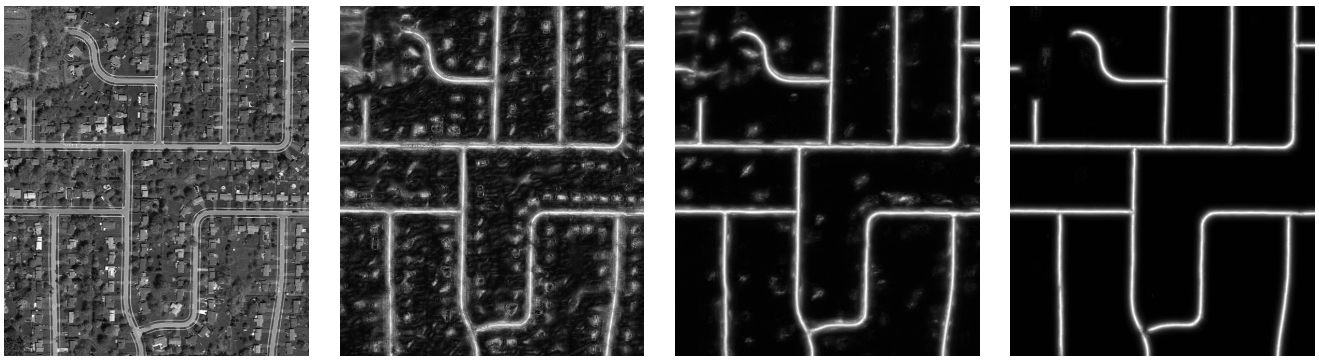
Road detection and edge detection are two of the first problems addressed in Computer Vision [22, 7, 8, 4]. In this section we discuss the main methods that have been proposed.

2.1. Road detection

The automatic extraction of roads from aerial images is an important and challenging problem. Many services rely on an updated road map. However, because of occlusions, illumination changes, and the variability of roads shapes, an automatic method able to extract roads in all situations still does not exist.

Many methods have been proposed in the literature. Early approaches rely on handcrafted features to distinguish roads pixels from other objects [2, 10, 15, 13]. However, these methods are prone to fail if applied to large databases, where the aspect of roads and non-roads objects is difficult to model and predict.

Recently, machine learning algorithms have been applied to the problem of roads and centerline detection [3, 12, 5, 14, 20, 21]. Typically, these methods try to solve centerline detection as a classification method to classify roads or centerlines pixels from the rest. The authors of [20] train a convolutional neuran network to classify road pixels and add a post processing



(a) (b) (c) (d)

Figure 3. Comparison of our method with [27] on a sample road image. (a) Input Image $I(\mathbf{x})$; (b) Score map obtained with [27]: $\varphi^{(0)}(\mathbf{x})$. (c) Score map obtained using our method: $\varphi^{(N)}(\mathbf{x})$, with $N = 2$; (d) Score map obtained adding the multiscale post-processing step: $\Phi(\mathbf{x})$. In (b),(c) and (d) we show the maximum projection along the radial dimension for visualization purposes.

step to include structure information. In [12] Conditional Random Fields are applied to road labeling, [14] instead uses spectral-structural features and SVMs to find road centerlines.

In contrast with these approaches, we reformulate centerline detection as a regression problem and show that this approach is more accurate and robust than classification based methods. Extending our approach to multiscal detection we can also estimate the width of the roads and generate a segmentation of the roads or use the output of our method in combination with tracking algorithms [15, 13, 29]. Results corresponding to this approach are also shown in Sec. 4.

2.2. Contour detection

Edge detection is one of the most studied problem in computer vision. The first attempt to solve this problem are based on filters designed to respond to specific intensity profiles in the image. Among these, the most known is the Canny edge detector [4], while a more recent approach is given by [1].

As for road detection, attention recently shifted to classification based methods [19, 5, 23, 18, 17, 6, 26].

In [19] gradients on different image channels are combined with a logistic regression classifier to predict contours in natural images. In [23] instead, gradient of sparse codes are used to train SVM's to classify contours at different orientations, before recombining them in a final classifier.

Sketch tokens and random forests are used in [17], while [5] uses a boosting algorithm. Recently, in [6] structured random forests have been used to generate an extremely efficient contour detector.

Finally, the approach of [26] uses a cascade of classifiers at different resolutions to predict the boundary map. This last approach is similar in spirit to ours, however it still relies on classification. In this work we show that regression is more appropriate and has several advantages compared to classification also when applied to contour detection.

3. Method

We start from the same framework of [27]. Given an image $I = I(\mathbf{x})$, with $\mathbf{x} \in \mathbb{R}^2$ and the set of centerlines (or contour) points $C = \{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} \text{ is on a centerline}\}$, we want to learn a regressor to approximate a function $y(\cdot)$ that associates to a feature vector $f(\mathbf{x}, I)$ extracted form a local neighborhood of a pixel $\mathbf{x}$, the value of the function $d(\mathbf{x})$ defined by:

$$d(\mathbf{x}) = \begin{cases} e^{a(1-\frac{\mathcal{D}_C(\mathbf{x})}{d_M})} - 1 & \text{if } \mathcal{D}_C(\mathbf{x}) < d_M \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathcal{D}_C$ is the Euclidean distance transform to the centerlines and a, d_M are parameters defining the shape of the function d. We use the same parameters as in [27].

Given a training set $\{(f_i, y_i)\}_i$, with $f_i = f(\mathbf{x}_i, I_i) \in \mathbb{R}^M$ the feature vector corresponding to a point $\mathbf{x}_i$ in image I_i and $y_i = d(\mathbf{x}_i)$, $y(\cdot)$ is approximated in [27] using the Gradient Boost algorithm [11] by a function of the form:

$$\varphi(f(\mathbf{x}, I)) = \sum_{k=1}^{K} \alpha_k h_k(f(\mathbf{x}, I)) \,, \tag{2}$$

where the weak learners $h_k : \mathbb{R}^M \to \mathbb{R}$ are regression trees and $\alpha_k \in \mathbb{R}$ are weights. For simplicity we will write $\varphi(\mathbf{x})$ instead of $\varphi(f(\mathbf{x}, I))$ and $h_k(\mathbf{x})$ instead of $h_k(f(\mathbf{x}, I))$.

Learning the function $\varphi(\mathbf{x})$ is the first step in our algorithm.

In our method we then iteratively train a sequence of regressors and use the output of one iteration as input to the next one. More precisely, as it is done in the auto-context algorithm [28] and in [26] for classification, we use the score map $\varphi(\mathbf{x}) = \varphi^{(0)}(\mathbf{x})$ to extract a new set of features able to capture more contextual information. These new features will be added to the original feature vector to train a new regressor.

Let $g(\mathbf{x}, \varphi^{(0)})$ be the feature vector extracted from the score map $\varphi^{(0)}(\mathbf{x})$ and let $\{(f_i, g_i, y_i)\}_i$ be the new training set, with $g_i = g(\mathbf{x}_i, \varphi_i^{(0)}) \in \mathbb{R}^{M'}$ and $\varphi_i^{(0)} = \varphi^{(0)}(f(\mathbf{x}_i, I_i))$. We apply again the Gradient Boost algorithm to learn a better approximation of the function $y(\cdot)$:

$$\varphi^{(1)}(f(\mathbf{x}, I), g(\mathbf{x}, \varphi^{(0)})) = \sum_{k=1}^{K} \alpha_k^{(1)} h_k^{(1)}(f(\mathbf{x}, I), g(\mathbf{x}_i, \varphi_i^{(0)})) \,. \tag{3}$$

We iterate this process N times learning a sequence $\{\varphi^{(n)} = \varphi^{(n)}(f(\mathbf{x}, I), g(\mathbf{x}, \varphi^{(n-1)}))\}_{n=1,\dots,N}$. The idea is that at each iteration we correct the mistakes done at the previous one and we include more context information. The final output $\varphi^{(N)}(\cdot)$ will be used as approximation of $y(\cdot)$. Again, for the sake of brevity, we will write $\varphi^{(N)}(\mathbf{x})$ instead of $\varphi^{(n)}(f(\mathbf{x}, I), g(\mathbf{x}, \varphi^{(n-1)}))$.

In practice, the method converges fast and, as shown in Sec. 4, for $N = 2$ or $N = 3$ the performance improvement is saturated.

To limit the problem of overfitting, typical in an auto-context framework, we adopt several strategies. First, at the beginning of each auto-context iteration new pixel locations are extracted from the train images to build a new training set. Second, at each boosting iteration k we learn the weak learner h_k using only a subset of the whole training set, this variant of boosting is also known as Stochastic Gradient Boost [9]. Finally, as discussed in Sec. 3.2.1, also the features used to learn a weak learner are subsampled at each boosting iteration, this further increases robustness.

3.1. Multiscale Detection

In the case of the centerline detection problem, many applications also require an estimation of the width of the linear structures. As it is done in [27], we extend our approach to multiscale centerline detection by moving our discussion from the spatial domain $\mathbf{x} \in \mathbb{R}^2$ to the scale-space domain $(\mathbf{x}, r) \in \mathbb{R}^2 \times \mathbb{R}_{>0}$.

Given the set of centerlines with associated radial value $C = \{(\mathbf{x}, r) : \mathbf{x} \text{ is on a centerline of a linear structure of radius } r\}$, we now want to learn a function $y : f^r(\mathbf{x}, I) \to d(\mathbf{x}, r)$, where $d(\mathbf{x}, r)$ is the scale-space extension of (1):

$$d(\mathbf{x}, r) = \begin{cases} e^{a \cdot (1 - \frac{\mathcal{D}_C(\mathbf{x}, r)}{d_M})} - 1 & \text{if } \mathcal{D}_C(\mathbf{x}, r) < d_M, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The feature vector $f^r(\mathbf{x}, I)$ depends now on the radius of the structure we would like to detect.

We approximate $y(\cdot, r)$ by a set of regressors $\{\varphi_{r_1}(\cdot), \varphi_{r_2}(\cdot), \dots, \varphi_{r_R}(\cdot)\}$, for some predefined set of radii $\{r_i\}_{i=1,\dots,R}$. Each regressor $\varphi_{r_i}(\cdot)$ is trained to have maximal response for centerlines at radius r_i. The values of $\varphi_{r_i}(\mathbf{x})$ decrease exponentially moving away from centerline points both in the spatial direction and in the radial component.

Notice that, if at test time we want to detect centerline points at a new scale $r_0 \notin \{r_i\}_{i=1,\dots,R}$, we can use scale space theory techniques to approximate $y(\cdot, r_0)$ starting from $\{\varphi_{r_i}(\cdot)\}_i$. For example a simple way to do this is to select the closest value to r_0 in $\{r_i\}_i$: $\bar{r} = \operatorname{argmin}_i |r_0 - r_i|$, rescale the input image by a factor $\bar{r}/r_0$, apply the learned regressor $\varphi_{\bar{r}}$ to the rescaled image and finally rescale the output at the original size.

In our method, as for the single-scale case, we use auto-context and iterate the learning procedure. We create therefore a sequence $\{\varphi_{r_i}^{(n)}(\cdot)\}_{r_i,n}$ of scale-space regressors.

The score functions at iteration $n-1$ for the different radius: $\{\varphi_{r_i}^{(n-1)}(\mathbf{x})\}_{r_i}$, could all be used to extract features to fed to the regressor for the next iteration. However, this would increase the learning time and the computational cost considerably. For this reason we adopt a simpler but effective solution: for the first N iterations the regressors at different scales are learned independently: $\varphi_{r_{i_0}}^{(n)} = \varphi_{r_{i_0}}^{(n)}(f^{r_{i_0}}(\mathbf{x}, I), g(\mathbf{x}, \varphi_{r_{i_0}}^{(n-1)}))$. Then, as a final step, we take the score maps obtained at all scales $\{\varphi_{r_i}^{(N)}(\mathbf{x})\}_{r_i}$ and use them to train a multivariate regressor $\Phi(\mathbf{x}, r_1, \dots, r_R) \approx y(\mathbf{x}, r)$. For this last stage we don't use the

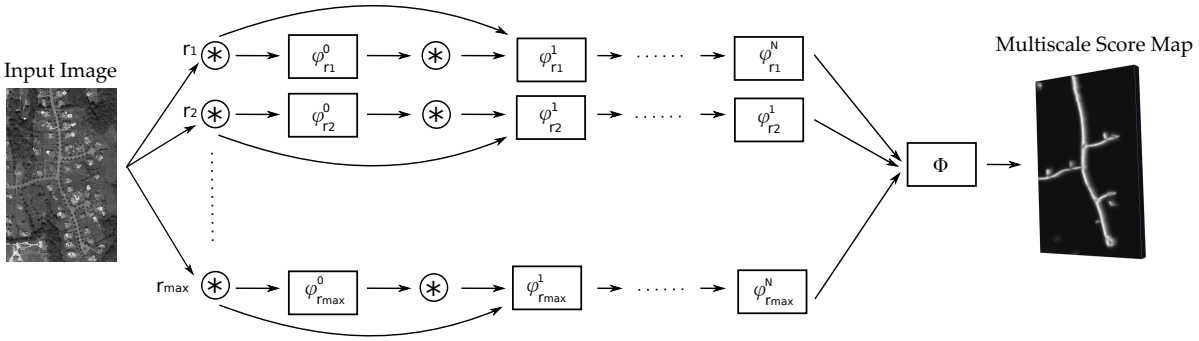


Figure 4. Multiscale centerline detection. For each value of the radius r, the input image is convolved with a bank of filters to extract features. The features are used as input to different regressors $\varphi_r^{(0)}$, one for each scale. The output scores of the regressors are then convolved with other filter banks to extract new features that are fed to a new layer if regressors $\varphi_r^{(1)}$. The process is iterated N times. In the final step, the output of the regressors is combined by the multiscale regressor Φ to obtain the final score map.

image information anymore. $\Phi(\cdot,\cdot)$ is built again with a boosting algorithm and has the form:

$$\Phi(\mathbf{x}, r_1, \ldots, r_R) = \sum_{k=1}^{K} \alpha_k h_k(\{\varphi_{r_i}^{(N)}(\mathbf{x})\}_{r_i}) \,, \tag{5}$$

where now the weak learners $h_k(\{\varphi_{r_i}^{(N)}(\mathbf{x})\}_{r_i}) \in \mathbb{R}^R$. $\Phi(\mathbf{x}, r_1, \ldots, r_R)$ is a scale space representation of the input image learned to be maximal at location $\mathbf{x}$ and scale r_i, where $\mathbf{x}$ is a centerline point or a linear structure of radius r_i. This last step has the effect to give consistency and smoothness to the values returned by the different regressors, that have been trained independently. Fig. 3 shows the advantages of this last step on a sample road image, while Fig. 4 gives a schematic representation of our method.

3.2. Feature vectors

In this section we discuss the features used to learn the regressors introduced in the previous section.

3.2.1 Image features

To compute the feature vector $f(\mathbf{x}, I)$, we start from a bank of convolutional filters $\{f_1, \ldots, f_J\}$, learned via convolutional sparse coding [24] on the training data. In [27], the features at a point $\mathbf{x}$ are given by the convolutions $f(\mathbf{x}, I) = \{f_j * I(\mathbf{x})\}_j$ for all $j = 1, \ldots, J$. In our framework we do not consider only the central pixel $\mathbf{x}$, but also locations within a certain distance to it. In this way we obtain a much larger pool of possible features $f(\mathbf{x}, I) = \{f_j * I(\mathbf{x} + \rho)\}_{j,\rho}$, with $\rho \in \mathbb{R}^2$ and $\|\rho\| \leq \|\rho_{\max}\|$ for some fixed $\rho_{\max}$.

For example, for a filter bank of 121 filters, as the one used on the task of centerline detection, and setting $\|\rho_{\max}\| = 13$, we obtain a total number of possible features is ≈ 64200. At each boosting iteration only a subset of this big pool of features is selected. In this way we increase efficiency and also robustness to overfitting.

In the case of color or multichannel input images, a different filter bank is learned for each channel.

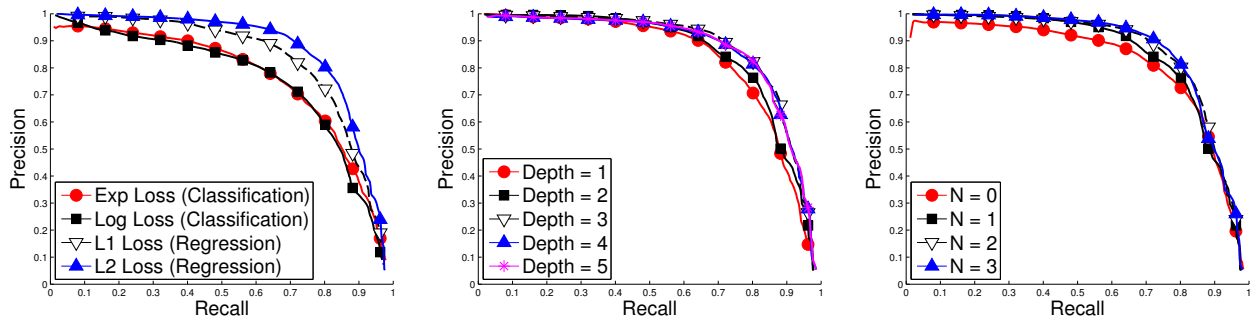
To speed up the feature extraction phase, the computation of the convolutions is approximated by linear combination of separable filters, as described in [25].

3.2.2 Auto-context features

After the first auto-context iteration the score map is used to extract a new set of features that will be used in the next iteration. Also in this case we consider convolutions with a learned filter bank.

Ideally we should learn a set of filters on the score maps at each auto-context iteration, however, this approach is too expensive. We therefore learn a filter bank on the ground truth function $d(\mathbf{x})$ computed on the train images and use the obtained filters on the score maps obtained at each iteration.

More precisely, let $\{g_1, \ldots, g_{J'}\}$ the set of filters learned on $d(\mathbf{x})$, then, the auto-context features $g(\mathbf{x}, \varphi^{(n)})$ used at iteration n are given by $\{g_j * \varphi^{(n)}(\mathbf{x} + \rho)\}_{j,\rho}$, again with $\rho \in \mathbb{R}^2$, $\|\rho\| \leq \|\rho'_{\max}\|$.



(a) (b) (c)

Figure 5. Parameter tuning on the Aerial dataset using a tollerance of 1 pixel on the localization of the centerlines. (a) Loss function; (b) Depth of trees used as weak learners; (c) number N of auto-context iterations.

For the task of road's centerline detection, 36 filters are used. Setting again $\|\rho'_{\max}\|$, the total number of features is now $\approx 64200 + 19100 = 83300$.

In the case of multiscale detection, at the last iteration, a multiscale function Φ is learned starting from $\{\varphi_r^{(N)}\}_r$. Since this step is used only to obtain consistency between the outputs of the different regressors and to obtain a smoother output, we simply use values $\{\varphi_r^{(N)}(\mathbf{x}+\rho)\}_{r,\rho}$ as features.

4. Results

In this section we first introduce the datasets used in our experiments, then we discuss the choice of the parameters and finally we discuss the results. The code and the parameters used in our experiments are publicly available at XXX.

4.1. Datasets

We considered one aerial dataset to test our method on the centerline detection problem and the BSDS500 [1] dataset for boundary detection:

- **Aerial**. Aerial images containing roads. 7 images are used for training and 7 for testing. The roads aspect changes from image to image and several non-road objects on the background have local aspect similar to the roads. The ground truth is annotated by a human and it is also provides radial information. For these experiments we only considered the grayscale intensities images.
- **BSDS500**. This is a standard dataset used to test boundary detection algorithms. It is composed by 500 color images, 200 are used for training, 100 for validation and the remaining 200 for testing. Several human annotations are given in the binary groundtruth, before computing the distance transform used to build function d, we merge them together.

4.2. Parameters Selection

In this section we study the influence of the parameters on the performance of our method.

We fix the number of samples in the training set to 40000, half of them are taken from locations close to centerline points, the second half are selected from the background. The number of boosting iterations used is $K = 150$.

On the roads images we use the same filters as in [27], but we train only 2 regressors at scales 6 and 8 to predict all scales in the range 5 to14.

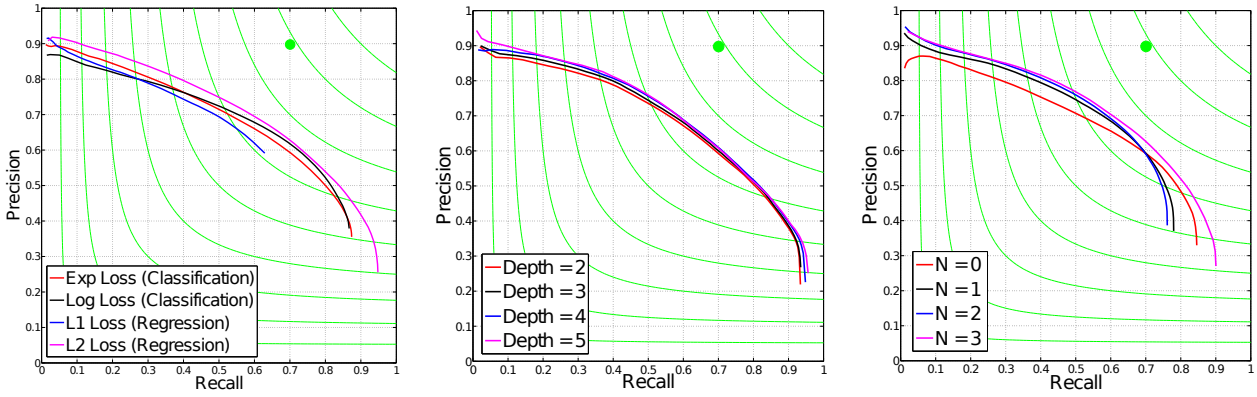
For the BSDS500 dataset, we first converted the images to the Luv color space and then learned learn different filter banks on the three channel at different scales.

For the auto-context iterations we learn another filter bank on the ground truth images $d(\mathbf{x})$, one for each dataset.

To evaluate the results on the Aerial dataset we used the metric defined in [29], using a tollarance parameter of 1 pixel on the localization of the centerlines. For the BSDS500 dataset we used the standard Berkeley evaluation framework [1].

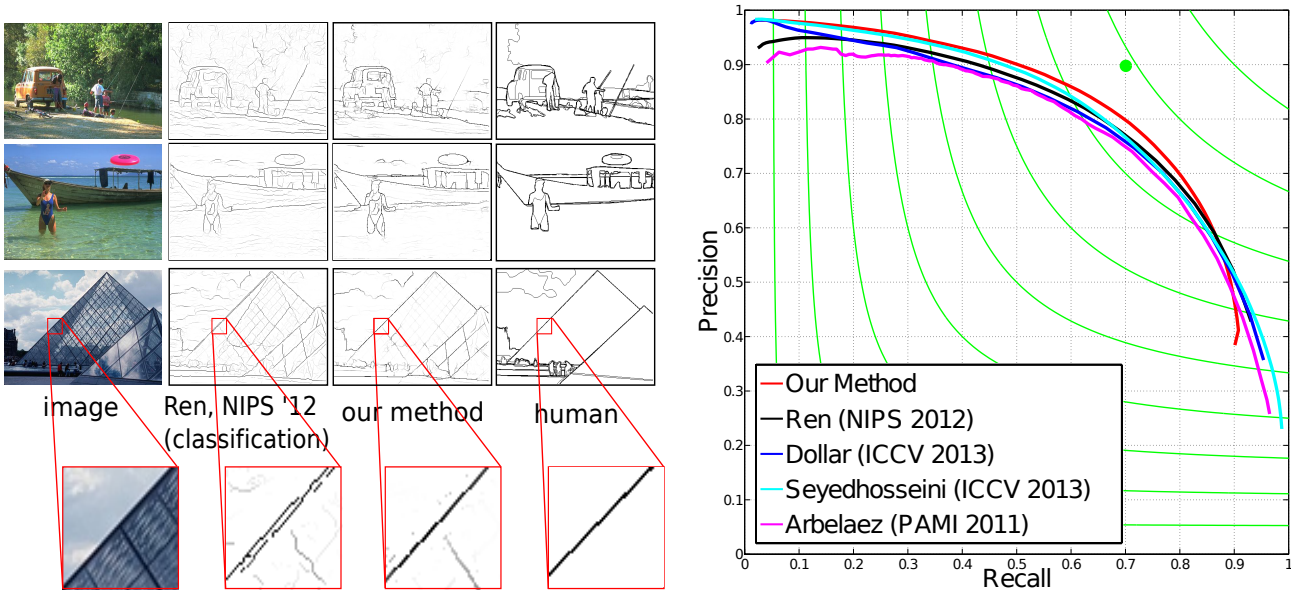
Fig. 5 and 6 show the performance of the methods as a function on the number of iterations N, the maximum depth of the trees used as weak learners, and the loss used by the gradient boost algorithm.

From this curves we can see that, for both datasets, the performance increases considerably by using our iterative approach. The best number of iterations is $N = 3$ for the BSDS500 dataset, while $N = 2$ is enough for the simpler Aerial dataset.



(a) (b) (c)

Figure 6. Parameter tuning on BSDS500 validation set. (a) Loss function; (b) Depth of trees used as weak learners; (c) number N of auto-context iterations.



(a) (b)

Figure 7. Boundary detection results. (a) Comparison between our method and [23] on the BSDS500 dateset. (b) Presicion recall curves.

Also the choice of the loss has a big influence on the performance. Using the proposed regression based approach with the squared L_2 norm gives the best results. On the roads, both the L_2 and the L_1 losses used for regression are better than the exponential loss and logaritmic loss used for classification. On the BSDS500 dataset the L_1 norm gives worse results, the reason for this is that the solution obtained with the L_1 norm is more sparse and many boundaries are not detected. The best performance is again obtained by regression, using the squared loss.

Concerning the tree depth, deeper trees increase performance, in particular for the roads, however, this parameter has less influence on the performance than the others.

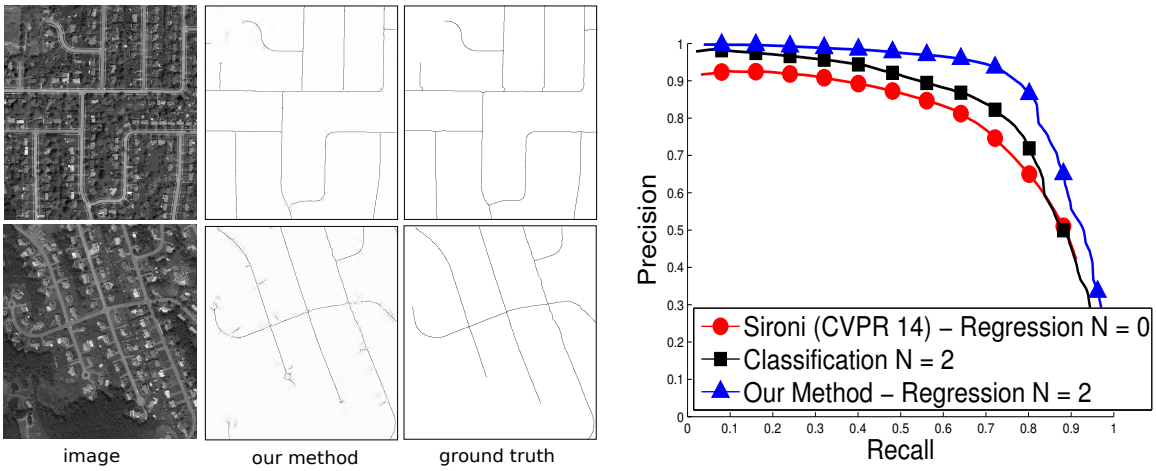
4.3. Discussion

In this section we run a new set of experiments using the best parameters found in the previous section. Moreover, we train our model using more training samples: 400000 for the Aerial dataset and 10^6 on the BSDS500 dataset, again half of the samples are taken from the background and the other half close to the centerlines. We use $K = 250$ and the squared loss.

As baseline we use the results of [27] for on the Aerial dataset, that are state-of-the-art for this dataset. To compare against classification we train a second time our method with the same parameters, but using the binary ground truth and the exponential loss.

Table 1. BSDS dataset results.

Method	ODS	OIS	AP
Our Method	**0.75**	**0.77**	0.78
Ren [23]	0.74	0.76	0.77
Dollar [6]	0.73	0.75	0.78
Seyedhosseini [26]	0.73	0.75	**0.80**
Arbelaez [1]	0.73	0.76	0.73



(a) (b)

Figure 8. Centerline detection results. (a) Centerlines extracted with our method (b) Presicion recall curves.

Figure 9. Qualitative results on road tracking. Gree is true positive, blue false positive, red false negative.

Table 2. Tracking results.

Method	DIADEM score
Our Method	**0.82**
Turetken [30]	0.75

For the BSDS500 we compare against several state-of-the-art methods [23, 6, 26, 1], where [23, 26] are classiffication based approaches. The corresponding precision recall curves are shown in Fig. **??**. In Table 1 we also report the ODS and ODF $F-$values on the BSDS500 dataset.

Finally, as a last experiment, we used our method in combiantion with the tracking algorithm [30] to obtain a reconstruction of the roads network. We compare our results against those obtained using an handcrafted measure [16]. We used the DIADEM metric [] to aveluate the accuracy of the reconstruction. The results are reported in Table 2. With our method more accurate results are obtained. Fig. 9 shows instead some qualitative results.

5. Conclusion

In this work we presented an extension of the regression-based method [27] for centerlines and boundary detection. We show that a regression is more accurate and robust than classification in these tasks, where little displacements of the ground truth are negligible and can generate confusion in a binary classification framework.

We increase the accuracy of the original method by using auto-context iterations. This help to eliminate false responses on the background and to incorporate more contextual information to increase detection accuracy.

Our method outperform state-of-the-art techniques for boundary detection on the BSDS500 dataset and on road tracking, both for centerline detection and road tracking, when used in combination with a reconstruction algorithm.

For future work we want to test our method on larger databases and test its the generalization capability. Moreover, as for the case of roads tracking, the boundaries detected with our method could be used as input to a global tracing algorithm to generate segmentation of natural images.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour Detection and Hierarchical Image Segmentation. *PAMI*, 33(5):898–916, 2011. 2, 3, 6, 8

[2] R. Bajcsy and M. Tavakoli. Computer recognition of roads from satellite pictures. *IEEE Transactions on Systems, Man, and Cybernetics*, 1976. 2

[3] U. Bhattacharya and S. Parui. An Improved Backpropagation Neural Network for Detection of Road-Like Features in Satellite Imagery. *International Journal of Remote Sensing*, 18:3379–3394, 1997. 2

[4] J. Canny. A Computational Approach to Edge Detection. *PAMI*, 8(6), 1986. 2, 3

[5] P. Dollar, Z. Tu, and S. Belongie. Supervised Learning of Edges and Object Boundaries. In *CVPR*, 2006. 2, 3

[6] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 3, 8

[7] M. Fischler and A. Heller. Automated Techniques for Road Network Modeling. In *DARPA IUW*, pages 501–516, 1998. 2

[8] J. R. Fram and E. S. Deutsch. On the quantitative evaluation of edge detection schemes and their comparison with human performance. *IEEE Trans. Computers*, 1975. 2

[9] J. Friedman. Stochastic Gradient Boosting. *Computational Statistics & Data Analysis*, 2002. 4

[10] D. Geman and B. Jedynak. An Active Testing Model for Tracking Roads in Satellite Images. *PAMI*, 18(1):1–14, January 1996. 2

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001. 3

[12] X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale Conditional Random Fields for Image Labeling. In *CVPR*, pages 695–702, 2004. 2, 3

[13] J. Hu, A. Razdan, J. Femiani, M. Cui, and P. Wonka. Road network extraction and intersection detection from aerial images by tracking road footprints. *IEEE T. Geoscience and Remote Sensing*, 2007. 2, 3

[14] X. Huang and L. Zhang. Road Centreline Extraction from High-Resolution Imagery Based on Multiscale Structural Features and Support Vector Machines. *International Journal of Remote Sensing*, 30:1977–1987, 2009. 2, 3

[15] 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. *Mach. Vis. Appl.*, 2000. 2, 3

[16] M. Law and A. Chung. Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In *ECCV*, 2008. 8

[17] J. J. Lim, C. L. Zitnick, and P. Dollr. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013. 3

[18] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative Learned Dictionaries for Local Image Analysis. In *CVPR*, 2008. 3

[19] D. Martin, C. Fowlkes, and J. Malik. Learning to Detect Natural Image Boundaries Using Local Brightness, Color and Texture Cues. *PAMI*, 26(5), 2004. 3

[20] V. Mnih and G. Hinton. Learning to Detect Roads in High-Resolution Aerial Images. In *ECCV*, 2010. 2

[21] V. Mnih and G. Hinton. Learning to Label Aerial Images from Noisy Data. In *ICML*, 2012. 2

[22] L. Quam. Road Tracking and Anomaly Detection. In *DARPA IUW*, pages 51–55, May 1978. 2

[23] X. Ren and L. Bo. Discriminatively Trained Sparse Code Gradients for Contour Detection. In *Advances in Neural Information Processing Systems*, December 2012. 3, 7, 8

[24] R. Rigamonti and V. Lepetit. Accurate and Efficient Linear Structure Segmentation by Leveraging Ad Hoc Features with Learned Filters. In *MICCAI*, 2012. 5

[25] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua. Learning Separable Filters. In *CVPR*, 2013. 5

[26] M. Seyedhosseini, M. Sajjadi, and T. Tasdizen. Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks. In *Proceedings of the IEEE International Conference on Computer Vison (ICCV 2013)*, 2013. 3, 4, 8

[27] A. Sironi, V. Lepetit, and P. Fua. Multiscale Centerline Detection by Learning a Scale-Space Distance Transform. In *CVPR*, 2014. 1, 3, 4, 5, 6, 7, 9

[28] Z. Tu and X. Bai. Auto-Context and Its Applications to High-Level Vision Tasks and 3D Brain Image Segmentation. *PAMI*, 2009. 4

[29] E. Turetken, C. Becker, P. Glowacki, F. Benmansour, and P. Fua. Detecting Irregular Curvilinear Structures in Gray Scale and Color Imagery Using Multi-Directional Oriented Flux. In *ICCV*, December 2013. 3, 6

[30] E. Turetken, F. Benmansour, B. Andres, H. Pfister, and P. Fua. Reconstructing Loopy Curvilinear Structures Using Integer Programming. In *CVPR*, June 2013. 2, 8