Learning deep structured active contours end-to-end

Diego Marcos, Devis Tuia, Benjamin Kellenberger
University of Wageningen, Netherlands
name.surname@wur.nl

Lisa Zhang, Min Bai, Renjie Liao, Raquel Urtasun
University of Toronto, Canada
{lczhang,mbai,rjliao,urtasun}@cs.toronto.edu

Abstract

The world is covered with millions of buildings, and precisely knowing each instance's position and extents is vital to a multitude of applications. Recently, automated building footprint segmentation models have shown superior detection accuracy thanks to the usage of Convolutional Neural Networks (CNN). However, even the latest evolutions struggle to precisely delineating borders, which often leads to geometric distortions and inadvertent fusion of adjacent building instances. We propose to overcome this issue by exploiting the distinct geometric properties of buildings. To this end, we present Deep Structured Active Contours (DSAC), a novel framework that integrates priors and constraints into the segmentation process, such as continuous boundaries, smooth edges, and sharp corners. To do so, DSAC employs Active Contour Models (ACM), a family of constraint- and prior-based polygonal models. We learn ACM parameterizations per instance using a CNN, and show how to incorporate all components in a structured output model, making DSAC trainable end-to-end. We evaluate DSAC on three challenging building instance segmentation datasets, where it compares favorably against state-of-the-art. Code will be made available on https://github.com/dmarcosg/DSAC.

1. Introduction

Accurate footprints of individual buildings are of paramount importance for a wide range of applications, such as census studies [33], disaster response after earthquakes [25] and developmental assistances like malaria control [11]. Automating large-scale building footprint segmentation has thus been an active research field, and the emergence of high-capacity models like fully convolutional networks (FCNs) [13], together with vast training data [32], has led to promising improvements in this field.

Most studies address semantic segmentation of buildings, which consists of inferring a class label (e.g. "building") densely for each pixel over the overhead image of interest [16, 20, 21, 30]. While this approach may provide global statistics such as building area coverage estimation, it comes short at yielding estimations at the instance level. In computer vision, this problem is known as *instance segmentation*, where models provide a segmentation mask on a per-object instance basis. Solving this task is far more challenging than semantic segmentation, since the model has to understand whether any two building pixels belong to the same building or not. Precise delineation of object borders, with sharp corners and straight walls in the case of buildings, is a task that CNNs generally perform poorly at [9]: as a result, building segmentations from CNNs commonly have a high detection rate, but fail in terms of spatial coverage and geometric correctness.

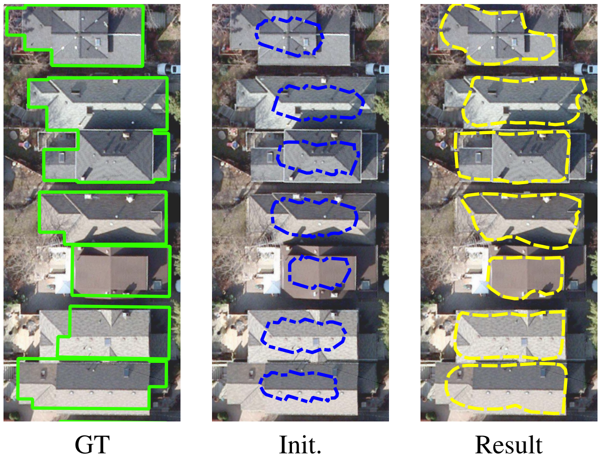


Figure 1. DSAC uses a CNN to predict the energy function used by an Active Contour Model (ACM) to modify an initial instance polygon using learned geometric priors. Left: image from the TorontoCity validation dataset with ground truth polygons, center: initial polygons provided by [2], right: results of DSAC.

Active Contour Models (ACM [17]), also called *snakes*, may be considered to address this issue. ACMs augment bottom-up boundary detectors with high-level geometric constraints and priors. They work by constraining the possible outputs to a family of curves (e.g. closed polygons with a fixed number of vertices), and optimizing them by means of energy minimization based on both the image features and a set of shape priors such as boundary continuity and smoothness. Additional terms have been proposed, among which the balloon term [7] is of particular interest: it mimics the inflation of a balloon by continuously pushing the snakes' vertices outwards, thus preventing it to collapse to a single point. By expressing object detection as a polygon fitting problem with prior knowledge, ACMs have the potential of approaching object edges precisely and without the need for additional post-processing. However, the original formulation lacked flexibility, since it relied on low-level image features and a global parameterization of priors, when a more useful approach would be to penalize strongly the curvature in the regions of the boundary known to be straight or smooth and reduce the penalization in the regions that are more likely to form a corner. Moreover, the balloon term has so far only been included as a post-energy global minimization force and does not take part in the energy minimization defining the snake.

In this paper, we propose to combine the expressiveness of deep CNNs with the versatility of ACMs in a unified framework, which we term Deep Structured Active Contours (DSAC). In essence, we employ a CNN to learn the energy function that would allow an ACM to generate polygons close to a set of ground truth instances. To do so, DSAC leverages the original ACM formulation by learning high-level features and prior parameterizations, including the balloon term, in one model and *on a local basis, i.e.* penalizing each term differently at each image location. We cast the optimization of the ACM as a structured prediction problem and find optimal features and parameters using a Structured Support Vector Machine (SSVM [1, 29]) loss. As a consequence, DSAC is trainable end-to-end and able to learn and adapt to a particular family of object instances. We test DSAC in three building instance segmentation datasets, where it outperforms state-of-the-art models.

Contributions This work's contributions are as follows:

- We formulate the learning of the energy function of an ACM as a structured prediction problem;
- We include the balloon term of the ACM into the energy formulation;
- We propose an end-to-end framework to learn the guiding features and local priors with a CNN.

2. Related work

Building footprint extraction Most current automated approaches make use of 3D information extracted from ground or aerial LIDAR [31], or employ humans in the loop [4]. The use of a polygonal shape prior has been shown to substantially improve the results [27] of systems based on color imagery and low level features. Recent efforts employ deep CNNs for semantic segmentation and allowed a great leap towards full automation of building segmentation [16]. Works considering building instance segmentation are scarcer and the task has been recently defined as far-from-being solved [32], despite the interest shown by the participation to numerous contests aiming at automatic vectorization of building footprints from overhead imagery: SpaceNet[1], DSTL[2] or OpenAI Challenge[3]. Our proposed DSAC aims at making high-level geometric information available to CNN based methods as a step towards bridging this gap.

Instance segmentation in Computer Vision Since instance segmentation combines object detection and dense segmentation, many proposed pipelines attempt at fusing both tasks in either separate or end-to-end trainable models. For example, [8] employ a multi-task CNN to detect candidate objects and infer segmentation masks and class labels per detection. [10] train a CNN on pairs of locations and predicts the likelihood for the pair to belong to the same object. [22] apply an attention-based RNN sequentially on deep image features to trace object instances in propagation order. [2] refine an existing semantic segmentation map by predicting a distance transform to the nearest boundary. High level relationships are accounted for in [23, 34] by means of an instance MRF applied to the CNN's output.

All these methods employ pixel-wise CNNs and are thus not apt to integrating output shape priors directly, as polygonal output models would be. Only a few works deal with CNNs that explicitly produce a polygonal output. In [5], a recursive neural network is used to generate a segmentation polygon node by node, while in [24] a CNN predicts the direction of the nearest object boundary for each node in a polygon and uses it as a data term in an ACM. However, the first model is tailored towards a different problem (interactive segmentation and correction) and does not allow the inclusion of strong priors, and the second decouples the CNN training from ACM inference, thus lacking the end-to-end training capabilities of the proposed DSAC.

Active contours The first ACMs were introduced by Kass *et al.* in 1988 under the name of snakes [17]. Variants of this original try to overcome some of its limitations, such as the need for precise initializations, or the dependence on user interaction. In [12] the authors propose to use two coupled snakes that better capture the information in the image. The above mentioned balloon force was introduced by [7].

Although some modifications [18] have been proposed to improve the data term of the original paper, they rely on simple assumptions about the appearance of the objects and on global parameters for weighting the different terms in the

[1] https://wwwtc.wpengine.com/spacenet

[2] https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection

[3] https://werobotics.org/blog/2018/01/10/open-ai-challenge/

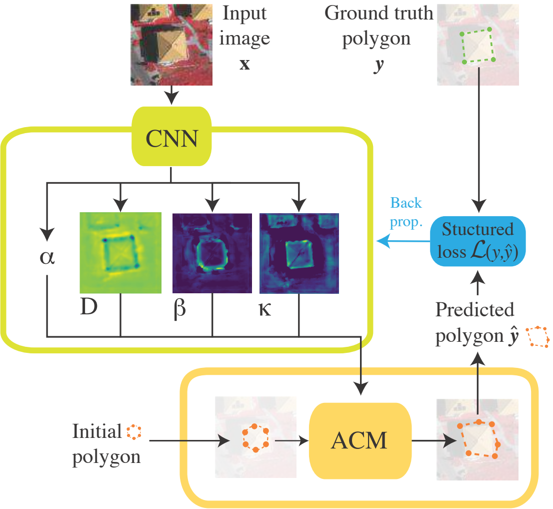


Figure 2. DSAC idea. The CNN predicts the values of the energy terms to be used by the active contour model (ACM): a global α for the length penalization and maps for local D, the data term, β, the curvature penalization and κ, the balloon term. After ACM inference, a structured loss is computed and given to the CNN, whose parameters can then be updated using backpropagation.

energy function. The proposed DSAC leverages the original formulation by including local prior information, i.e. values weighting the snakes' energy function terms on a per-pixel basis, and learns them using a CNN. Although this work focuses on curvature priors useful for segmenting objects of polygonal shape, other priors can be enforced with ACMs, such as convexity for biomedical imaging [23].

Structured learning with CNNs Structured prediction [28] allows to model dependencies between multiple output variables and hence offers an elegant way to incorporate prior rule sets on output configurations. End-to-end trainable structured models exceed traditional two-step solutions by enriching the learning signal with relations at the output level. Although these models have been applied to a variety of problems [3, 6, 26], we are not aware of any work dealing with instance level segmentation.

We use a structured loss as a learning signal to a CNN such that it learns to coordinate the different ACM energy terms, which are heavily interdependent.

3. Method

We present the details of a modified ACM inference algorithm with image-dependent and local penalization terms as well as the structured loss that is used to train a CNN to generate these penalization maps. A diagram of the proposed method is shown in Fig. 2. The proposed training algorithm proceeds as exposed in Algorithm 1.

```
Data: $\mathcal{X}, \mathcal{Y}$: image/polygon pairs in the training set.
  $\mathcal{Y}^0$: corresponding polygon initializations.
for $\mathbf{x}_i, \mathbf{y}_i \in \mathcal{X}, \mathcal{Y}$ do
    CNN inference: $D, \alpha, \beta, \kappa \leftarrow CNN_\omega(\mathbf{x}_i)$
    ACM inference: $\hat{\mathbf{y}}_i \leftarrow ACM(D, \alpha, \beta, \kappa, \mathbf{y}_i^0)$
    $\frac{\partial \mathcal{L}}{\partial D}, \frac{\partial \mathcal{L}}{\partial \alpha}, \frac{\partial \mathcal{L}}{\partial \beta}, \frac{\partial \mathcal{L}}{\partial \kappa} \leftarrow \hat{\mathbf{y}}^i, \mathbf{y}_i$ and Eqs. 18-21
    Compute $\frac{\partial \mathcal{L}}{\partial \omega}$ using backpropagation
    Update CNN: $\omega \leftarrow \omega - \eta \frac{\partial \mathcal{L}}{\partial \omega}$
end
```

Algorithm 1: The DSAC training algorithm. At every iteration, the CNN forward pass is followed by ACM inference, which yields a contour that is used to compute the structured loss.

Note that i) DSAC does not depend on any particular ACM inference algorithm, and ii) the chosen ACM algorithm does not need to be differentiable.

3.1. Locally penalized active contours

An active contour [17] can be represented as a polygon $\mathbf{y} = (\mathbf{u}, \mathbf{v})$ with L nodes $\mathbf{y}_s = (u_s, v_s) \in \mathbb{R}^2$, with $s \in 1 \ldots L$, where each s represents one of the nodes of the discretized contour. The polygon $\mathbf{y}$ is then deformed such that the following energy function is minimized:

$$E(\mathbf{y},\mathbf{x}) = \sum_{s=1}^{L}\Big[D\big(\mathbf{x},(\mathbf{y}_s)\big)+\alpha\big(\mathbf{x},(\mathbf{y}_s)\big)\Big|\frac{\partial \mathbf{y}}{\partial s}\Big|^2 + \beta\big(\mathbf{x},(\mathbf{y}_s)\big)\Big|\frac{\partial^2 \mathbf{y}}{\partial s^2}\Big|^2\Big] + \sum_{u,v\in\Omega(\mathbf{y})}\kappa\big(\mathbf{x},(u,v)\big), \quad (1)$$

where $D(\mathbf{x}) \in \mathbb{R}^{U\times V}$ is the data term, depending on input image, of size $U \times V$, $\mathbf{x} \in \mathbb{R}^{U\times V\times d}$, $\alpha(\mathbf{x}), \beta(\mathbf{x}) \in \mathbb{R}^{U\times V}$ are the terms encouraging short and smooth polygons respectively, $\kappa(\mathbf{x})$ is the balloon term and $\Omega(\mathbf{y})$ is the region enclosed by $\mathbf{y}$. The notation $D\big(\mathbf{x},(\mathbf{y}_s)\big)$ means the value in $D(\mathbf{x})$ indexed by the position $\mathbf{y}_s = (u_s, v_s)$.

Due to their local nature, D,β and κ are $U \times V$ maps in our experiments while α is treated as a single scalar.

3.1.1 Data term

This term identifies areas of the image where the nodes of the polygon should lie. In the literature, $D(\mathbf{x})$ is usually some predefined function on the image, typically related to the image gradients. $D(\mathbf{x})$ should learn to provide relatively low values along the boundary of the object of interest and high values elsewhere. During ACM inference, the direction of steepest descent $-\nabla D(\mathbf{x}) = -\big[\frac{\partial D(\mathbf{x})}{\partial u}, \frac{\partial D(\mathbf{x})}{\partial v}\big]$ is used as the data force term, moving the contour towards regions where D is low.

3.1.2 Internal terms

In the literature, the values of α and β are generally a single scalar, meaning that the penalization has the same strength in all parts of the object. This leads to a trade-off between over-smoothing corner regions and under-smoothing others. We avoid this trade-off by assigning different β penalizations to each pixel, depending on which part of the object lies underneath.

The internal energy $E_{int} = \alpha\big(\mathbf{x}, (\mathbf{y}_s)\big)|\mathbf{y}'|^2 + \beta\big(\mathbf{x}, (\mathbf{y}_s)\big)|\mathbf{y}''|^2$ penalizes the length (membrane term) and curvature (thin plate term) of the polygon. In order to obtain the direction of steepest descent, we can express the internal energy as a function of finite differences:

$$E_{int} = \sum_{s=0}^{L} \alpha(\mathbf{y}_s)\Big|\frac{\mathbf{y}_{s+1} - \mathbf{y}_s}{\Delta s}\Big|^2 + \beta(\mathbf{y}_s)\Big|\frac{\mathbf{y}_{s+1} - 2\mathbf{y}_s + \mathbf{y}_{s-1}}{\Delta s^2}\Big|^2, \tag{2}$$

and compute the derivative of E_{int} w.r.t. the coordinates of node s, $\mathbf{y}_s$, expressed as a sum of scalar products:

$$\begin{aligned}\frac{\partial E_{int}}{\partial \mathbf{y}_s} &= \frac{2}{\Delta s}[-\alpha_{s-1}, \alpha_{s-1}+\alpha_s, -\alpha_s]\cdot[\mathbf{y}_{s-1}, \mathbf{y}_s, \mathbf{y}_{s+1}]^\top \\ &+ \frac{2}{\Delta s^2}[\beta_{s-1}, -2\beta_s - 2\beta_{s-1}, \beta_{s-1} + 4\beta_s + \beta_{s+1}, \\ &- 2\beta_{s+1} - 2\beta_s, \beta_{s+1}] \cdot [\mathbf{y}_{s-2}, \mathbf{y}_{s-1}, \mathbf{y}_s, \mathbf{y}_{s+1}, \mathbf{y}_{s+2}]^\top.\end{aligned} \tag{3}$$

The Jacobian matrix (in this case with two column vectors) can then be expressed as a matrix multiplication:

$$\frac{\partial E_{int}}{\partial \mathbf{y}} = (A + B)\mathbf{y} \tag{4}$$

where $A(\alpha)$ is a tri-diagonal matrix and $B(\beta)$ is a penta-diagonal matrix.

3.1.3 Balloon term

The original balloon term [7] consists of adding an outwards force of constant magnitude in the normal direction of each node, thus inflating the contour. As with the β term, we propose to increase its flexibility by allowing it to take a different value at each image location.

In [7], the balloon term is only considered as a force added after the direction of steepest descent for the other energy terms has been computed. In DSAC, the SSVM formulation requires to express it in the form an energy.

The normal direction to the contour at $\mathbf{y}_s$ follows the vector:

$$\mathbf{n}_s = \big[\mathbf{y}_{s+1} - \mathbf{y}_{s-1}\big]_{+90^o} = \big[v_{n+1} - v_{n-1}, u_{n-1} - u_{n+1}\big]. \tag{5}$$

This can be rewritten such that the whole set of L normal vectors is expressed as:

$$\mathbf{n} = \Big[C\mathbf{v}, \mathbf{u}^\top C\Big] \tag{6}$$

where C is a tri-diagonal matrix with 0 in the main diagonal, 1 in the upper diagonal and -1 in the lower diagonal.

Integrating this expression with respect to $\mathbf{u}$ and $\mathbf{v}$, we obtain the scalar E_b, corresponding to the polygon's area (by the shoelace formula to compute the area of a polygon):

$$E_b = \mathbf{u}^\top C\mathbf{v} = \int\int_{u,v\in\Omega(\mathbf{y})} du dv \tag{7}$$

Instead of maximizing the area of the polygon, which would be the result of pushing nodes in the normal direction, we propose to use a more flexible term that maximizes the integral of the values of a map $\kappa(\mathbf{x}) \in \mathbb{R}^{M\times N}$ over the area enclosed by the contour, $\Omega(\mathbf{y})$. If we discretize the integral to the pixel values that conform κ, we obtain:

$$E_k = \sum_{u,v\in\Omega(\mathbf{y})} \kappa(u, v) \tag{8}$$

After this modification we need to recompute the force form of this term by finding the $L \times 2$ Jacobian matrix $[\frac{\partial E_k}{\partial u_s}, \frac{\partial E_k}{\partial v_s}]$, $s \in [1, L]$.

This corresponds to how a perturbation in u_s and v_s would affect E_k. Since the perturbations are considered to be very small, we assume that the distribution of the $\kappa(u, v)$ values along the segments $[\mathbf{y}_s, \mathbf{y}_{s+1}]$ and $[\mathbf{y}_{s-1}, \mathbf{y}_s]$ will be identical to the one in $[\mathbf{y}_s + \Delta\mathbf{y}, \mathbf{y}_{s+1}]$ and $[\mathbf{y}_{s-1}, \mathbf{y}_s + \Delta\mathbf{y}]$, respectively. As shown in Fig. 3, this boils down to summing a series of trapezoid areas, forming the two depicted triangles, each one weighted by its assigned κ value.

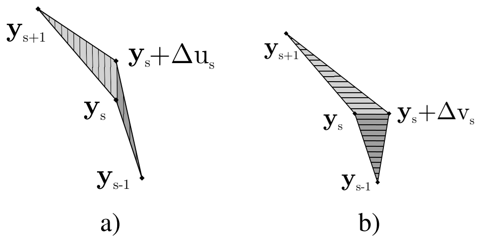


Figure 3. A perturbation of $\mathbf{y}_s$ in either the u or v direction would result in a change in area highlighted as two shaded triangles sharing the same base.

In Fig. 3a, both triangles have bases of length Δu_s and heights $v_{s-1} - v_s$ and $v_{s+1} - v_s$, while in Fig. 3b the bases are Δv_s and the heights $u_{s-1} - u_s$ and $u_{s+1} - u_s$.

To obtain the κ weighted areas in Fig. 3a, we compute:

$$\begin{aligned}\Delta E_k = \frac{\Delta u_s}{v_{s-1} - v_s}\int_{h=0}^{v_{s-1}-v_s} h\kappa(h)dh+ \\ \frac{\Delta u_s}{v_{s+1} - v_s}\int_{h=0}^{v_{s+1}-v_s} h\kappa(h)dh,\end{aligned} \tag{9}$$

and therefore the force term we need for inference is:

$$\frac{\partial E_k}{\partial u_s} = \frac{1}{v_{s-1} - v_s} \int_{h=0}^{v_{s-1}-v_s} h\kappa(h)dh + \frac{1}{v_{s+1} - v_s} \int_{h=0}^{v_{s+1}-v_s} h\kappa(h)dh \quad (10)$$

The same for Fig. 3b can be obtained by swapping u and v.

These derivatives point in the normal direction when the values of κ are equal in all locations.

3.2. Active contour inference and implementation

When solving the active contour inference, Eq. (1), the four energy terms can be split into external terms E_{ext}: the data (D) and balloon energies (E_k); and internal terms E_{int}: the energies penalizing length (α) and curvature (β). Since E_{int} depends only on the contour $\mathbf{y}$, we can find an update rule that minimizes it on the new time time step:

$$\mathbf{y}^{t+1} = \mathbf{y}^t - \frac{dE_{ext}}{d\mathbf{y}^t} - (A+B)\mathbf{y}^{t+1}. \quad (11)$$

If we solve this expression for $\mathbf{y}^{t+1}$, we obtain:

$$\mathbf{y}^{t+1} = (I + A + B)^{-1}\Big(\mathbf{y}^t - \frac{dE_{ext}}{d\mathbf{y}^t}\Big). \quad (12)$$

With I being the identity matrix. An efficient implementation of the ACM inference is critical for the usability of the method, since thousands of iterations are typically required by CNNs to be trained, and the ACM inference has to be performed at each iteration. We have implemented the described locally penalized ACM using a Tensorflow graph. The typical inference time is under 50 ms on a single CPU for the settings used in this paper.

3.3. Structured SVM loss

Since no ground truth is available for the penalization terms, we frame the problem as structured prediction, in which loss augmented inference is used to generate negative examples to complement the positive examples of the ground truth polygons. The weights of the energy terms can then be modified such that the energy corresponding to the ground truth is lowered, while the one of the loss augmented results, which are presumed to be wrong, is increased.

Given a collection of ground truth pairs $(\mathbf{y^i}, \mathbf{x^i}) \in \mathcal{Y} \times \mathcal{X}, i = 1 \ldots N$, and a task loss function $\Delta(\mathbf{y}, \hat{\mathbf{y}})$, we would like to find the CNN parameters ω such that, by optimizing Eq. (1) and thus obtaining the inference result:

$$\hat{\mathbf{y}}^i = \arg\min_{\mathbf{y}\in\mathcal{Y}} E(\mathbf{y}, \mathbf{x}, \omega) \quad (13)$$

one could expect a small $\Delta(\mathbf{y}^i, \hat{\mathbf{y}}^i)$. The problem becomes:

$$\hat{\omega} = \arg\min_{\omega} \sum_i \Delta(\mathbf{y}^i, \arg\min_{\mathbf{y}\in\mathcal{Y}} E(\mathbf{y}, \mathbf{x}, \omega)) \quad (14)$$

Since $\Delta(\mathbf{y}^i, \hat{\mathbf{y}}^i)$ could be a discontinuous function, we can substitute it by a continuous and convex upper bound, such as the hinge loss. By adding an ℓ_2 regularization and summing for all training samples, this becomes the max-margin formulation:

$$\mathcal{L}(\mathcal{Y}, \mathcal{X}, \omega) = \frac{1}{2}\|\omega\|^2 + C \sum_i \Bigg(\max_{\mathbf{y}\in\mathcal{Y}} \left[0, \Delta(\mathbf{y}, \mathbf{y}^i) - E(\mathbf{y}, \mathbf{x}^i; \omega) + E(\mathbf{y}^i, \mathbf{x}^i; \omega)\right] \Bigg). \quad (15)$$

Since $\mathcal{L}(\mathcal{Y}, \mathcal{X}, \omega)$ is convex but not differentiable, we compute the subgradient, which requires to find the most penalized constraint with the current ω:

$$\hat{\mathbf{y}}^i = \arg\max_{\mathbf{y}\in\mathcal{Y}} \left[\Delta(\mathbf{y}, \mathbf{y}^i) - E(\mathbf{y}, \mathbf{x}^i; \omega)\right] \quad (16)$$

This means to first run the ACM using the current ω and an extra term corresponding to the loss $\Delta(\mathbf{y}, \mathbf{y}^i)$. Once we obtain $\hat{\mathbf{y}}^i$, we can then compute the subgradient as:

$$\frac{\partial \mathcal{L}(\mathcal{Y}, \mathcal{X}, \omega)}{\partial \omega} = \omega + C \sum_i \big(\frac{\partial E(\mathbf{y}^i, \mathbf{x}^i; \omega)}{\partial \omega} - \frac{\partial E(\hat{\mathbf{y}}^i, \mathbf{x}^i; \omega)}{\partial \omega} \big) \quad (17)$$

We compute the subgradients of the loss with respect to each of the four outputs as

$$\frac{\partial \mathcal{L}(\mathbf{y}^i, \mathbf{x}^i, \omega)}{\partial D_\omega(\mathbf{x}^i)} = [(u,v) \in \mathbf{y}^i] - [(u,v) \in \hat{\mathbf{y}}^i] \quad (18)$$

$$\frac{\partial \mathcal{L}(\mathbf{y}^i, \mathbf{x}^i, \omega)}{\partial \alpha_\omega(\mathbf{x}^i)} = \left|\frac{\partial \mathbf{y}^i(u,v)}{\partial s}\right|^2 [(u,v) \in \mathbf{y}^i] - \left|\frac{\partial \hat{\mathbf{y}}^i(u,v)}{\partial s}\right|^2 [(u,v) \in \hat{\mathbf{y}}^i] \quad (19)$$

$$\frac{\partial \mathcal{L}(\mathbf{y}^i, \mathbf{x}^i, \omega)}{\partial \beta_\omega(\mathbf{x}^i)} = \left|\frac{\partial^2 \mathbf{y}^i(u,v)}{\partial s^2}\right|^2 [(u,v) \in \mathbf{y}^i] - \left|\frac{\partial^2 \hat{\mathbf{y}}^i(u,v)}{\partial s^2}\right|^2 [(u,v) \in \hat{\mathbf{y}}^i] \quad (20)$$

$$\frac{\partial \mathcal{L}(\mathbf{y}^i, \mathbf{x}^i, \omega)}{\partial \kappa_\omega(\mathbf{x}^i)} = [(u,v) \in \Omega(\mathbf{y}^i)] - [(u,v) \in \Omega(\hat{\mathbf{y}}^i)]. \quad (21)$$

In the above equations, $[\cdot]$ represents the Iverson bracket. Finally, we can get $\frac{\partial \mathcal{L}(\mathcal{Y}, \mathcal{X}, \omega)}{\partial \omega}$ using the chain rule and modifying each CNN parameter ω applying:

$$\omega_{t+1} = \omega_t - \eta \frac{\partial \mathcal{L}(\mathcal{Y}, \mathcal{X}, \omega)}{\partial \omega}, \tag{22}$$

which will simultaneously decrease $E(\mathbf{y}^i, \mathbf{x}^i; \omega)$ and increase $E(\hat{\mathbf{y}}^i, \mathbf{x}^i; \omega)$, thus making a better solution more likely when performing inference anew.

Task loss The task loss $\Delta(\mathbf{y}, \mathbf{y}^i)$ defines the actual objective we want to solve with the SSVM loss. Since it's the most common metric in instance segmentation, we employ the Intersection-over-Union (IoU) between the prediction $\mathbf{y}$ and the ground truth $\mathbf{y}^i$. Note that optimizing for IoU can be split into maximizing the intersection while minimizing the union. During training, this allows us to simply add a negative value during training to the κ map at the locations within the ground truth and a positive outside to obtain a loss-augmented inference (see Fig. 4).

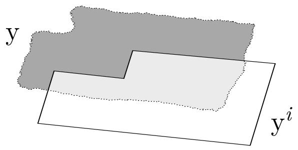


Figure 4. When training we encourage a high task loss (IoU) by modifying the balloon term E_κ, adding a negative constant to κ at the nodes of the prediction $\mathbf{y}$ inside the ground truth $\mathbf{y}^\mathbf{i}$ (light gray), and a positive constant to those outside (dark gray).

4. Experiments

We test the proposed DSAC method for building footprint extraction from overhead images. We consider two settings: *manual initialization*, where the user provides a single click near the center of the building and *automatic initialization*, where an instance segmentation algorithm is used to generate the initial polygons. The first setting is tested in two datasets, *Vaihingen* and *Bing Huts*, while the second is tested in the *TorontoCity* dataset [32]. The three datasets are detailed in the respective sections.

4.1. CNN architecture and general setup

To learn the ACM energy terms, we use a CNN architecture similar to the Hypercolumn model in [14]. The input consists of a patch cropped around each initialization polygon and resized an image of fixed size for each dataset. The first layer consists of 7×7 convolutions, the second of 5×5 and all subsequent layers are of size 3×3. All the convolutional layers are followed by ReLu, batch normalization and 2×2 max-pooling. The number of filters is increased with the depth: 32, 64, 128 ,128, 256 and 256 for the six blocks. The output tensors of all the layers are then upsampled to the output size and concatenated. After this, a two-layer MLP with 256 and 64 hidden units is used to predict the four output maps: $D(\mathbf{x})$, $\alpha(\mathbf{x})$, $\beta(\mathbf{x})$ and $\kappa(\mathbf{x})$. We use this architecture for all datasets, with the exception of the *Bing huts* dataset, for which we skip the last two convolutional layers. In all cases, we use the Adam optimizer with a learning rate of 10^{-4}. We augment the data with random rotations. The number of ACM iterations is set to 50 in all the experiments, and the number of nodes is set to $L = 60$ in *Vaihingen* and *TorontoCity* and $L = 20$ in *Bing huts*.

4.2. Manual initialization

In this setting, the detection step is done manually by visual inspection. The only input required from the user is a single click to indicate the approximate center of the building. Two datasets are considered:

Vaihingen buildings The dataset consists of 168 buildings extracted from the training set of the ISPRS "2D semantic labeling contest"[4]. The images have three bands, corresponding to near infrared, red and green wavelengths, and a resolution of 9 cm. We used 100 buildings to train the models and the remaining 68 as a test set.

Bing huts The dataset consists of 605 individual huts visible on Bing maps aerial imagery at a resolution of 30 cm, over a rural area in Tanzania. See Fig. 5 for an overview of the study area and Fig. 7 for a full resolution subset. The ground truth building footprints have been obtained from OpenStreetMap[5]. A total of 335 images of size 80×80 pixels are used to train the models and the remaining 270 to test. The lower spatial resolution, low contrast between the buildings and the surrounding soil, as well as the high level of label noise make *Bing huts* a very challenging dataset.

We compare DSAC against a baseline where we train a CNN with the same architecture used by DSAC, but with a 3-class cross entropy loss with classes: building, building boundary, background. The boundary class is added to help the model focus on learning the shapes of the buildings. In this case, the click from the user is used to select the nearest connected region that has been labeled as building and treat it as the instance prediction.

4.3. Automatic initialization

Although the manual initialization only requires a single click from the user, it can still be a tedious task for large scale datasets. Existing instance segmentation algorithms,

[4] http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html

[5] http://www.openstreetmap.org

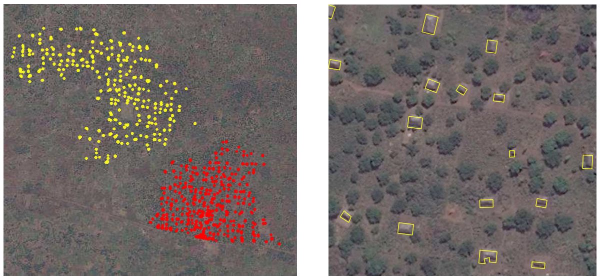


Figure 5. Left: Overview of the 4 km^2 area covered by the Bing huts dataset. The training instances are higlighted in red and the test ones in yellow. Right: detail of the test set.

	Average IoU		RMSE
	Vaihingen	Bing huts	Bing huts
CNN Baseline	0.78	0.56	23.9
DSAC (ours)	**0.84**	**0.65**	**13.4**
DSAC (scalar κ, β)	0.64	0.60	19.1
DSAC (no κ)	0.63	0.42	31.2
DSAC (local α)	0.83	**0.65**	**13.4**

Table 1. Results on the test set for the manual initialization experiments, reported as average intersection over union (IoU, left) and area estimation (*Bing huts* only), with RMSE in m^2 (right).

such as the recently proposed Deep Watershed Transform (DWT) [2], can be used instead to initialize the active contours. These methods have a good recall, but tend to undersegment the objects and to lose detail near to the boundaries. To compensate for this effect, the authors of [2] apply a morphology-based post-processing step. We test the possibility of initializing the ACM within DSAC with the results obtained by [2] on the TorontoCity building instance segmentation dataset [32], with around 28000 instances for training and 12000 for testing. The ACM contours are initialized with the output of the Deep Watershed Transform (DWT) [2], the current state-of-the-art in terms of IoU. Two initialization polygon types are considered: the raw DWT output and the post-processed versions used in [32]. We also consider a third variant, where the raw DWT is used at train time and the post-processed one for inference at test time: this variant is based on the intuition that making the problem harder at train time, in addition to using the loss augmentation, helps learning a better energy function.

5. Results and discussion

Manual initialization Table 1 reports the average Intersection over Union (IoU) for the two datasets. Since the ground truth shift noise in the Bing huts dataset makes the IoU assessment untrustworthy, the root mean square error (RMSE in m^2) committed when estimating the area of the building footprints is also reported. DSAC significantly improves the baseline in terms of IoU for both datasets. This ablation study confirms the need to allow κ and β to vary locally (as opposed to having a single value for the whole image), while α can be treated as a single value without loss of performance. It also highlights the importance of the balloon term for the convergence of the contour.

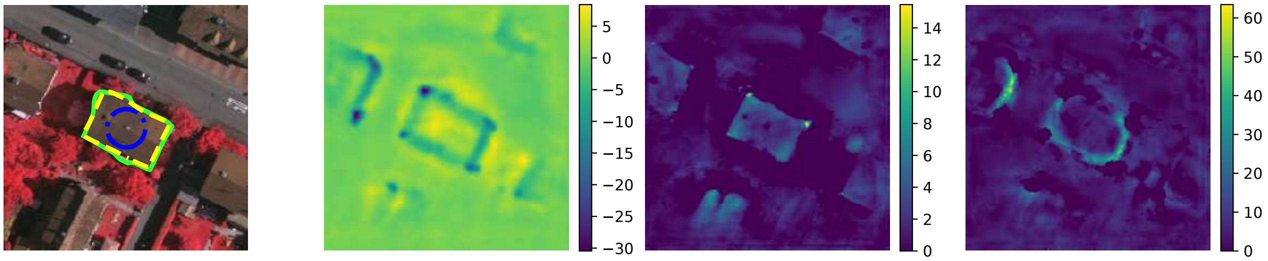
Examples of segmentation results for the Vaihingen dataset (Fig. 7, top row) show that the learned priors do indeed promote smooth, straight edges while often allowing for sharp corners. By looking at the predicted energy terms in Fig. 6 we observe that the model focuses on the corners by producing very low D values close to them, while predicting high κ inside the building next to the corners and a sharp drop to 0 on the outside. Moreover, the smoothness term β is close to 0 at the corners and high along the edges.

In the Bing huts dataset results (Fig. 7, bottom row), the biggest jump in performance can be seen in the area estimation metric. DSAC still tends to oversmooth the shapes, probably since it is unable to learn the location of corners due to the ground truth shift noise inherent to OpenStreetMap data, but manages to converge to polygons of the correct size, most probably because it learns to balance the balloon (κ, promoting large areas) and the membrane (α, promoting short contours) terms.

Automatic initialization Table 2 reports the results obtained on the TorontoCity dataset using two metrics: the IoU-based weighted coverage (“WeighCov”) and the shape similarity PolySim [32]. Besides DWT, we also compare DSAC against the results of building footprint segmentation with FCN and ResNet, as reported in [32]. We observe an improvement with respect to DWT of both metrics. DSAC obtains the best weighted coverage scores irrespectively of the initialization strategy. Interestingly, the best results are obtained by the hybrid initialization using raw DWT at training time and post-processed DWT polygons at test time. This suggests that our intuition about making the model work harder at train time is correct and seems to complement the use of a task loss in the SSVM loss. Finally, segmentation examples are shown in the last row of Fig. 7: DSAC (in yellow) consistently returns a more desirable segmentation with respect to DWT (in blue), closer to the ground truth polygon (in green). Although we can still see oversmoothing in our results, note how an important amount of shift noise is also present in some instances, making the DSAC result more plausible than the ground truth in a few cases (red arrows).

6. Conclusion

We have shown the potential of embedding high-level geometric processes into a deep learning framework for the segmentation of object instances with strong shape priors, such as buildings in overhead images. The proposed Deep Structured Active Contours (DSAC) uses a CNN to pre-



a) image **x** b) data term $D(\mathbf{x})$ c) balloon term $\kappa(\mathbf{x})$ d) thin plate term $\beta(\mathbf{x})$

Figure 6. a) Image from the Vaihingen test set. The initial contour is in blue and the result in yellow, with the ground truth in green. b) Data term $D(\mathbf{x})$, where we can observe regions of lower energy along the boundary of the building. c) The balloon term $\kappa(\mathbf{x})$ has learned to produce positive values only inside the building, especially next to corners. d) In the thin plate term $\beta(\mathbf{x})$, we see that the curvature tends to be less penalized close to the building's corners. The membrane term provided by the model in this example was $\alpha(\mathbf{x}) = 0.74$

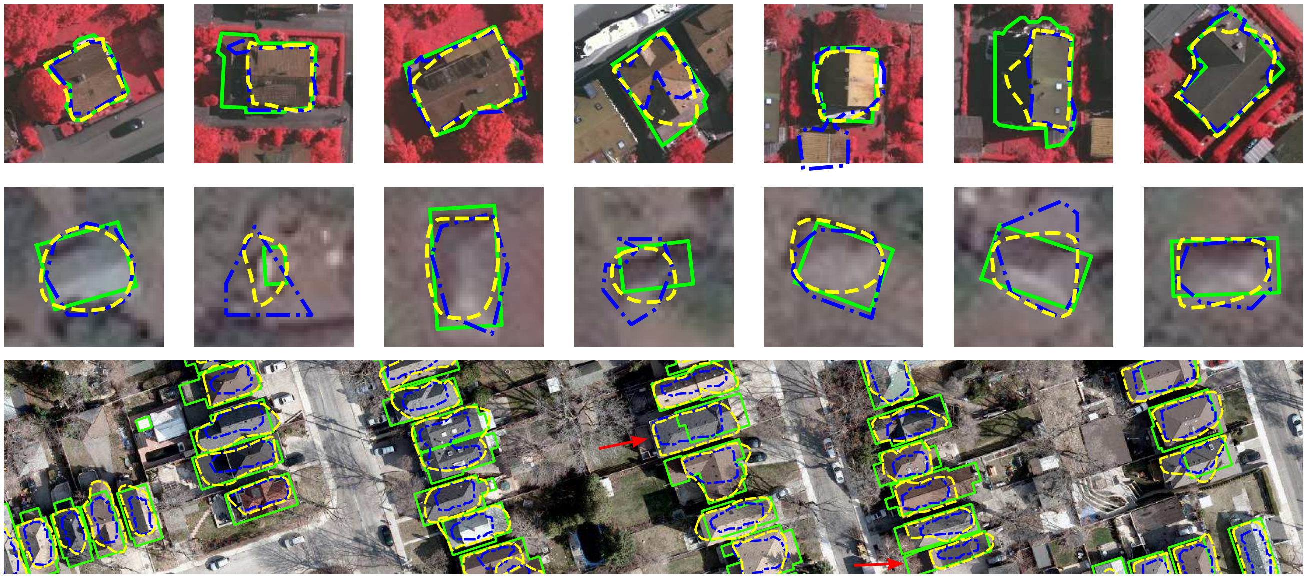


Figure 7. Examples of test set buildings in the Vaihingen (top row), Bing huts (middle row) and TorontoCity (bottom row) datasets. Ground truth in solid green line, baseline result in dash-dot blue and our active contour result in dashed yellow. Note that some of the ground truth polygons in the TorontoCity dataset are shifted (red arrows).

	WeighCov	PolySim
FCN [19]	0.46	**0.32**
ResNet [15]	0.40	0.29
DWT, raw [2] (`RW`)	0.42	0.20
DWT, postproc. (`PP`)	0.52	0.24
DSAC (init.: train `RW` / test `RW`)	0.55	0.26
DSAC (init.: train `PP` / test `PP`)	0.57	0.26
DSAC (init.: train `RW` / test `PP`)	**0.58**	0.27

Table 2. Results of the proposed DSAC and the methods reported in [32] on the validation set of the TorontoCity dataset, containing over 12000 detected building instances. Two ACM initializations, `RW` ([2]) and `PP` ([2] post-processed), are compared.

dict the energy function parameters for an Active Contour Model (ACM) such as to make its output close to a ground truth set of polygonal footprints. The model is trained end-to-end by bringing the ACM inference into the CNN training schedule and using the ACM's output and the ground truth polygon to assess a structured loss that can be used to update the CNN's parameters using back-propagation. DSAC opens up the possibility of using a large collection of energy terms encoding for different priors, since an adequate balance between them is learned automatically. The main limitation of our model is that the initialization is assumed to be given by some external method and is therefore not included in the learning process.

Results in three different datasets, which include a 10% relative improvement over the state-of-the-art on the *TorontoCity* dataset, show that combining the bottom-up feature extraction capabilities of CNNs with the high-level constraints provided by ACMs is a promising path for instance segmentation when strong geometric priors exist.

References

[1] Y. Altun, T. Hofmann, and I. Tsochantaridis. Support vector learning for interdependent and structured output spaces. In G. Bakir, T. Hofmann, B. Schlkopf, A. J. Smola, and S. Vishwanathan, editors, *Predicting Structured Data*, pages 85–105. MIT press, 2007. 2

[2] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017. 1, 2, 7, 8

[3] D. Belanger and A. McCallum. Structured prediction energy networks. In *ICML*, pages 983–992, 2016. 3

[4] R. Brooks, T. Nelson, K. Amolins, and G. B. Hall. Semi-automated building footprint extraction from orthophotos. *Geomatica*, 69(2):231–244, 2015. 2

[5] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-RNN. In *CVPR*, 2017. 2

[6] L.-C. Chen, A. Schwing, A. Yuille, and R. Urtasun. Learning deep structured models. In *ICML*, pages 1785–1794, 2015. 3

[7] L. D. Cohen. On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218, 1991. 2, 4

[8] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, pages 3150–3158, 2016. 2

[9] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 1

[10] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017. 2

[11] J. Franke, M. Gebreslasie, I. Bauwens, J. Deleu, and F. Siegert. Earth observation in support of malaria control and epidemiology: MALAREO monitoring approaches. *Geospatial health*, 10(1), 2015. 1

[12] S. R. Gunn and M. S. Nixon. A robust snake implementation; a dual active contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):63–68, 1997. 2

[13] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, pages 345–360. Springer, 2014. 1

[14] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456, 2015. 6

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 8

[16] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler. Learning aerial image segmentation from online maps. *IEEE Transactions on Geoscience and Remote Sensing*, 2017. 1, 2

[17] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988. 1, 2, 3

[18] S. Kichenassamy, A. Kumar, P. Olver, A. Tannenbaum, and A. Yezzi. Gradient flows and geometric active contour models. In *ICCV*, pages 810–815. IEEE, 1995. 2

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 8

[20] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):645–657, 2017. 1

[21] J. A. Montoya-Zegarra, J. D. Wegner, L. Ladickỳ, and K. Schindler. Semantic segmentation of aerial images in urban areas with class-specific higher-order cliques. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3):127, 2015. 1

[22] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *ECCV*, pages 312–329. Springer, 2016. 2

[23] L. A. Royer, D. L. Richmond, C. Rother, B. Andres, and D. Kainmueller. Convexity shape constraints for image segmentation. In *CVPR*, 2016. 2, 3

[24] C. Rupprecht, E. Huaroc, M. Baust, and N. Navab. Deep active contours. *arXiv preprint arXiv:1607.05074*, 2016. 2

[25] L. Sahar, S. Muthukumar, and S. P. French. Using aerial imagery and GIS in automated building footprint extraction and shape recognition for earthquake risk assessment of urban inventories. *IEEE Transactions on Geoscience and Remote Sensing*, 48(9):3511–3520, 2010. 1

[26] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015. 3

[27] X. Sun, C. M. Christoudias, and P. Fua. Free-shape polygonal object localization. In *ECCV*, pages 317–332. Springer, 2014. 2

[28] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *ICML*, pages 896–903. ACM, 2005. 3

[29] I. Tsochantaridis, T. Finley, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005. 2

[30] M. Volpi and D. Tuia. Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):881–893, 2017. 1

[31] O. Wang, S. K. Lodha, and D. P. Helmbold. A bayesian approach to building footprint extraction from aerial lidar data. In *International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 192–199. IEEE, 2006. 2

[32] 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. 1, 2, 6, 7, 8

[33] Y. Xie, A. Weng, and Q. Weng. Population estimation of urban residential communities using remotely sensed morphologic data. *IEEE Geoscience and Remote Sensing Letters*, 12(5):1111–1115, 2015. 1

[34] Z. Zhang, S. Fidler, and R. Urtasun. Instance-level segmentation for autonomous driving with deep densely connected mrfs. In *CVPR*, pages 669–677, 2016. 2