# Supplementary Material:
# DARNet: Deep Active Ray Network for Building Segmentation

Dominic Cheng[1, 2] Renjie Liao[1, 2, 3] Sanja Fidler[1, 2, 4] Raquel Urtasun[1, 2, 3]
University of Toronto[1] Vector Institute[2] Uber ATG Toronto[3] NVIDIA[4]
{dominic, rjliao, fidler}@cs.toronto.edu urtasun@uber.com

## 1. Proof of Proposition

**Proposition 1.** *Given a closed convex set $X$, a ray starting from any interior point of $X$ will intersect with the boundary of $X$ once.*

*Proof.* First, it is straightforward that a ray starting from any interior point of $X$ will intersect with its boundary since otherwise $X$ is not closed. Then we prove the intersection can only happen once by contradiction. We assume a ray starts from an interior point $a$ and intersects with the boundary of $X$ twice at $b$ and $c$. Without loss of generality, we assume $b$ is in between $a$ and $c$. Since $a$ is an interior point, we can find an open ball $A$ which centers at $a$ and $A \in X$. Given any point $\tilde{a} \in A$ other than $a$, we can uniquely determine a line $l$ which crosses $\tilde{a}$ and $c$. Then we can uniquely draw an open ball $B$ which centers at $b$ and has $l$ as the tangent line. Since $b$ is a boundary point of $X$, we can always find a point $\tilde{b} \in B$ such that $\tilde{b} \notin X$. Connecting $c$ and $\tilde{b}$, we can uniquely determine a line $\tilde{l}$. Since $|\angle \tilde{a}cb| \geq |\angle \tilde{b}cb|$, line $\tilde{l}$ will intersect with $A$ for at least once. Denoting any intersection point as $\hat{a}$, we know $\hat{a} \in A \in X$. Since $c \in X$, we know any point in between $c$ and $\hat{a}$, *i.e.*, the convex combination of $c$ and $\hat{a}$, should be in $X$ due to the fact that $X$ is convex. Therefore, $\tilde{b} \in X$ which contradicts. We show the schematic of proof in Fig. 1. □

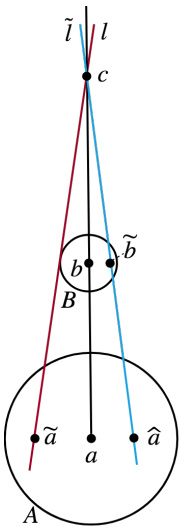


Figure 1: Illustration of proof.

## 2. Contour Inference Details

Our contour inference relies on the following equation,

$$\rho^{(t+1)} = \rho^{(t)} - \Delta t \left(A\rho^{(t)} + f\right) \tag{1}$$

where $\rho^{(t)}$ represents the contour at step $t$, $\Delta t$ is a time step hyper-parameter for solving the system, $A$ and $f$ consist of partial derivatives of the energy w.r.t. $\rho^{(t)}$. Here, we detail the construction of this equation.

**Matrix equation** As mentioned in our paper, the relevant partial derivatives are as follows. For the data term,

$$\frac{\partial E_{\text{data}}(c)}{\partial \rho_i} = \frac{\partial D(c_i)}{\partial x}\cos(i\Delta\theta) + \frac{\partial D(c_i)}{\partial y}\sin(i\Delta\theta) \tag{2}$$

where

$$c_i = \begin{bmatrix} x_c + \rho_i \cos(i\Delta\theta) \\ y_c + \rho_i \sin(i\Delta\theta) \end{bmatrix} \tag{3}$$

For the curvature term,

$$\begin{aligned} \frac{\partial E_{\text{curve}}(c)}{\partial \rho_i} \approx\ & [2\beta(c_{i+1})\cos(2\Delta\theta)]\rho_{i+2} + \\ & [-4(\beta(c_{i+1}) + \beta(c_{i-1}))\cos(\Delta\theta)]\rho_{i+1} + \\ & 2[\beta(c_{i+1}) + 4\beta(c_i) + \beta(c_{i-1})]\rho_i + \\ & [-4(\beta(c_i) + \beta(c_{i-1}))\cos(\Delta\theta)]\rho_{i-1} + \\ & [2\beta(c_{i-1})\cos(2\Delta\theta)]\rho_{i-2} \end{aligned} \tag{4}$$

For the balloon term,

$$\frac{\partial E_{\text{balloon}}(c)}{\partial \rho_i} \approx -\frac{\kappa(c_i)}{\rho_{\max}} \tag{5}$$

Combining these into the overall energy, we obtain

$$\begin{aligned} \frac{\partial E}{\partial \rho_i} \approx\ & [2\beta(c_{i+1})\cos(2\Delta\theta)]\rho_{i+2} + \\ & [-4(\beta(c_{i+1}) + \beta(c_{i-1}))\cos(\Delta\theta)]\rho_{i+1} + \\ & 2[\beta(c_{i+1}) + 4\beta(c_i) + \beta(c_{i-1})]\rho_i + \\ & [-4(\beta(c_i) + \beta(c_{i-1}))\cos(\Delta\theta)]\rho_{i-1} + \\ & [2\beta(c_{i-1})\cos(2\Delta\theta)]\rho_{i-2} + \\ & \frac{\partial D(c_i)}{\partial x}\cos(i\Delta\theta) + \frac{\partial D(c_i)}{\partial y}\sin(i\Delta\theta) - \frac{\kappa(c_i)}{\rho_{\max}} \end{aligned} \tag{6}$$

We have $L$ such equations, one for each $\rho_i$. In each equation, there are dependencies on the four adjacent entries of $\rho_i$, and $\rho_i$ itself (*i.e.* for entries on the borders, the indices wrap around). This summarizes into matrix form,

$$\frac{\partial E}{\partial \rho} \approx \begin{bmatrix} c_1 & b_1 & a_1 & 0 & \cdots & 0 & e_1 & d_1 \\ d_2 & c_2 & b_2 & a_2 & 0 & \cdots & 0 & e_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{L-1} & 0 & \cdots & 0 & e_{L-1} & d_{L-1} & c_{L-1} & b_{L-1} \\ b_L & a_L & 0 & \cdots & 0 & e_L & d_L & c_L \end{bmatrix} \begin{bmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_{L-1} \\ \rho_L \end{bmatrix} + \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{L-1} \\ f_L \end{bmatrix} \tag{7}$$

$$= A\rho + f \tag{8}$$

where

$$a_i = 2\beta(c_{i+1})\cos(2\Delta\theta) \tag{9}$$
$$b_i = -4(\beta(c_{i+1}) + \beta(c_{i-1}))\cos(\Delta\theta) \tag{10}$$
$$c_i = 2(\beta(c_{i+1}) + 4\beta(c_i) + \beta(c_{i-1})) \tag{11}$$
$$d_i = -4(\beta(c_i) + \beta(c_{i-1}))\cos(\Delta\theta) \tag{12}$$
$$e_i = 2\beta(c_{i-1})\cos(2\Delta\theta) \tag{13}$$
$$f_i = \frac{\partial D(c_i)}{\partial x}\cos(i\Delta\theta) + \frac{\partial D(c_i)}{\partial y}\sin(i\Delta\theta) - \frac{\kappa(c_i)}{\rho_{\max}} \tag{14}$$

**Solving the system** As mentioned in [2], to iteratively minimize the energy, we can solve this system of equations by introducing a time variable such that the solution is found at equilibrium. That is,

$$\frac{\rho^{(t+1)} - \rho^{(t)}}{\Delta t} = -(A\rho + f) \tag{15}$$

where we purposefully leave out the time indices on the right hand side of the equation. [2] then adopts an implicit-explicit method by interpreting $A$ implicitly (so it is associated with $\rho^{(t+1)}$) and $f$ explicitly (so it is associated with $\rho^{(t)}$),

$$\frac{\rho^{(t+1)} - \rho^{(t)}}{\Delta t} = -(A\rho^{(t+1)} + f) \tag{16}$$
$$\rho^{(t+1)} = \left(A + \frac{1}{\Delta t}I\right)^{-1}\left(\frac{1}{\Delta t}\rho^{(t)} - f\right) \tag{17}$$

To avoid the need to perform a batched matrix inverse, we instead adopt an explicit method,

$$\frac{\rho^{(t+1)} - \rho^{(t)}}{\Delta t} = -(A\rho^{(t)} + f) \tag{18}$$
$$\rho^{(t+1)} = \rho^{(t)} - \Delta t(A\rho^{(t)} + f) \tag{19}$$

## 3. CNN Architecture

**Our CNN backbone** Our CNN backbone uses Dilated Residual Network [5], specifically DRN-D-22 with its last pooling and fully-connected layers removed, as the main method of feature extraction. We append additional upsampling layers to yield output maps that match the input image size. This is illustrated in Figure 2.

**Adaptation to Deep Structured Active Contours (DSAC)** To implement DSAC [4], we leverage the same architecture in Figure 2, except with 4 outputs corresponding to the energy maps required in that framework. Following the implementation of DSAC, we add a gaussian smoothing layer with kernel size 9 and $\sigma = 2$ to the final output of the data term.

## 4. Hyper-parameters

We train our network using SGD with momentum. We choose a learning rate of $4 \times 10^{-5}$, which halves every $E$ epochs, momentum of 0.3, a weight decay of $1 \times 10^{-5}$, and a batch size of 10. We set $E = 30$ for Vaihingen and Bing Huts, and $E = 1$ for TorontoCity. We train for 100 epochs on Vaihingen and Bing Huts, and 10 epochs on TorontoCity.

To encourage stability in contour inference without using common techniques that are non-differentiable, we pretrain the maps to output values that cause the contour to converge, although not necessarily close to the ground truth rays. Specifically, we leverage the Euclidean distance transform because it possesses some desirable properties. Recall that we wish for $D$ to assign relatively lower values to the building boundaries, such that its gradient near these boundaries can attract contour points towards it. Both of these properties are reflected in the distance transform. For $\beta$, we adopt the distance transform with the building interiors masked out; we wish for contour points to evolve outwards without restriction, and straighten out as it approaches the boundaries. For $\kappa$, we adopt the distance transform with the building exteriors masked out. To pretrain, we take the predictions pr_0 and pr_1 (from Figure 2) and regress to these distance transforms with a smooth $L_1$ loss, using Adam [3] with an initial learning rate of $1 \times 10^{-3}$, which halves every $E$ epochs, and weight decay of $4 \times 10^{-4}$. We set

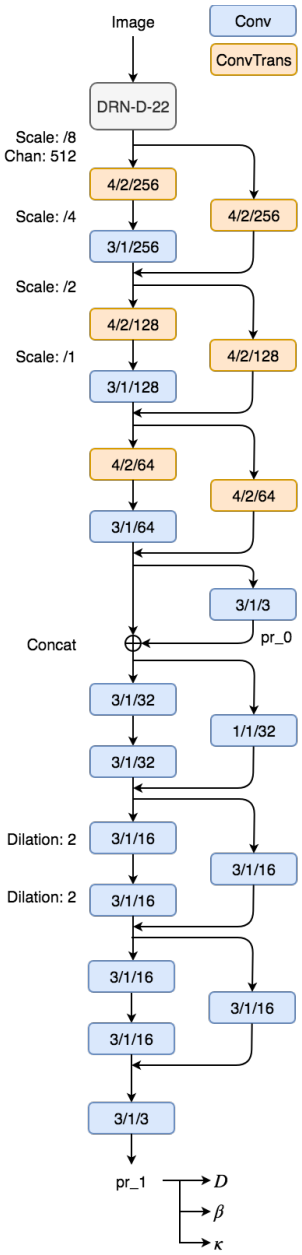


Figure 2: CNN architecture. Convolutions shown in blue; transposed convolutions shown in orange. Layers are annotated as kernel_size/stride/output_channels. Dilation is set to 1 unless stated otherwise. Output sizes are listed as a scale factor from the original input image. All residual blocks are combinations of Convolution-BatchNorm-ReLU in the style of [5], except for pr_0 and pr_1, where BatchNorm [1] is not used.

$E = 50$ for Vaihingen and Bing Huts, and $E = 3$ for TorontoCity. We pretrain for 250 epochs on Vaihingen and Bing Huts, and 10 epochs on TorontoCity. After pretraining, we scale the $\beta$ and $\kappa$ maps by $0.005$ and $0.1$ respectively so that, during the initial stages of training, the contours do not move too far. We found this procedure increases the stability of training.

## 5. More Visual Examples

We show more examples in Figures 3, 4, 5, and 6.

## References


[1] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. 4

[2] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1988. 3

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[4] D. Marcos, D. Tuia, B. Kellenberger, L. Zhang, M. Bai, R. Liao, and R. Urtasun. Learning deep structured active contours end-to-end. In *CVPR*, pages 8877–8885, 2018. 3

[5] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *CVPR*, 2017. 3, 4

(a) (b) (c) (d) (e) (f)

Figure 3: Results on (a-b) Vaihingen, (c-d) Bing Huts, (e-f) TorontoCity. Bottom three rows highlight failure cases. Original image shown in left. On right, our output is shown in cyan; DSAC output in yellow; ground truth is shaded.

(a) (b) (c) (d) (e) (f)

Figure 4: Results on (a-b) Vaihingen, (c-d) Bing Huts, (e-f) TorontoCity. Bottom three rows highlight failure cases. Original image shown in left. On right, our output is shown in cyan; DSAC output in yellow; ground truth is shaded.

(a) (b) (c) (d) (e) (f)

Figure 5: Results on (a-b) Vaihingen, (c-d) Bing Huts, (e-f) TorontoCity. Bottom three rows highlight failure cases. Original image shown in left. On right, our output is shown in cyan; DSAC output in yellow; ground truth is shaded.

Figure 6: Demonstration of our method (segmentations shown in shaded color) on a large area of the TorontoCity dataset.