Weakly supervised 3D segmentation from extreme points

NVIDIA

HAL-MICCAI 2019

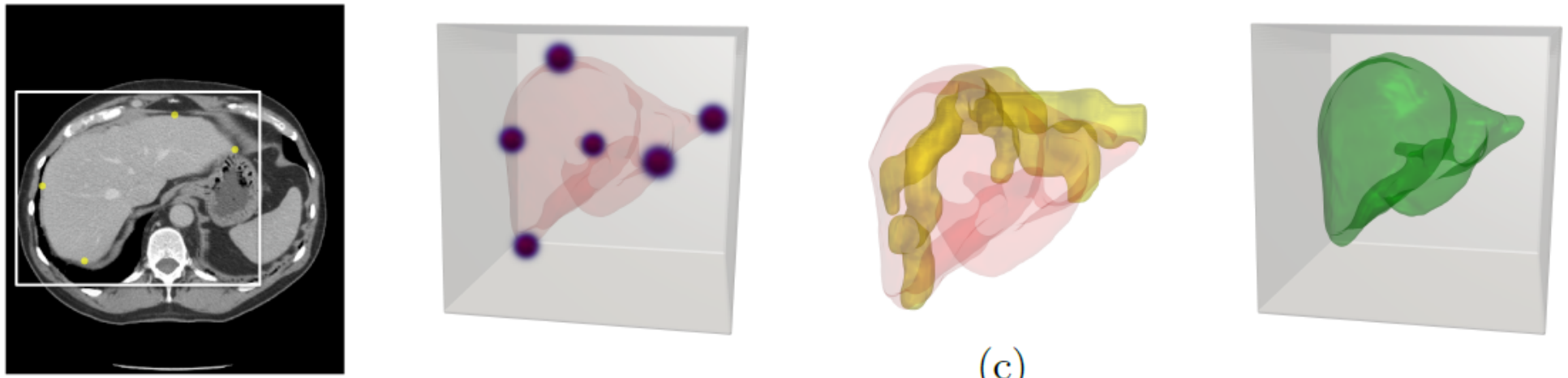
Motivation

- Previous 3D segmentation work is in a slice-by-slice fashion, which is limited to certain organs has good image contrast.
- Bounding Box has corners that lie outside the object of interest.
- It is tricky to select bounding box for 3D objects where the user typically has to navigate three multi-planar reformatted views.

- Completely unsupervised techniques might fail to generalize to organs where the boundary information is not as clear.

- Solution: This work propose a method with minimal user interaction in form of extreme point clicks.

Method



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

Fig. 1: Our weakly supervised segmentation framework. (a) The user selects extreme points that define the organ of interest (here the liver) in 3D space. (b) Extreme points are modeled as Gaussians in an extra image channel which is fed to a 3D segmentation model. (c) Foreground scribbles are generated automatically to initialize random walker (the ground truth surface is shown in red for reference). (d) Model returns the segmentation results.

Results

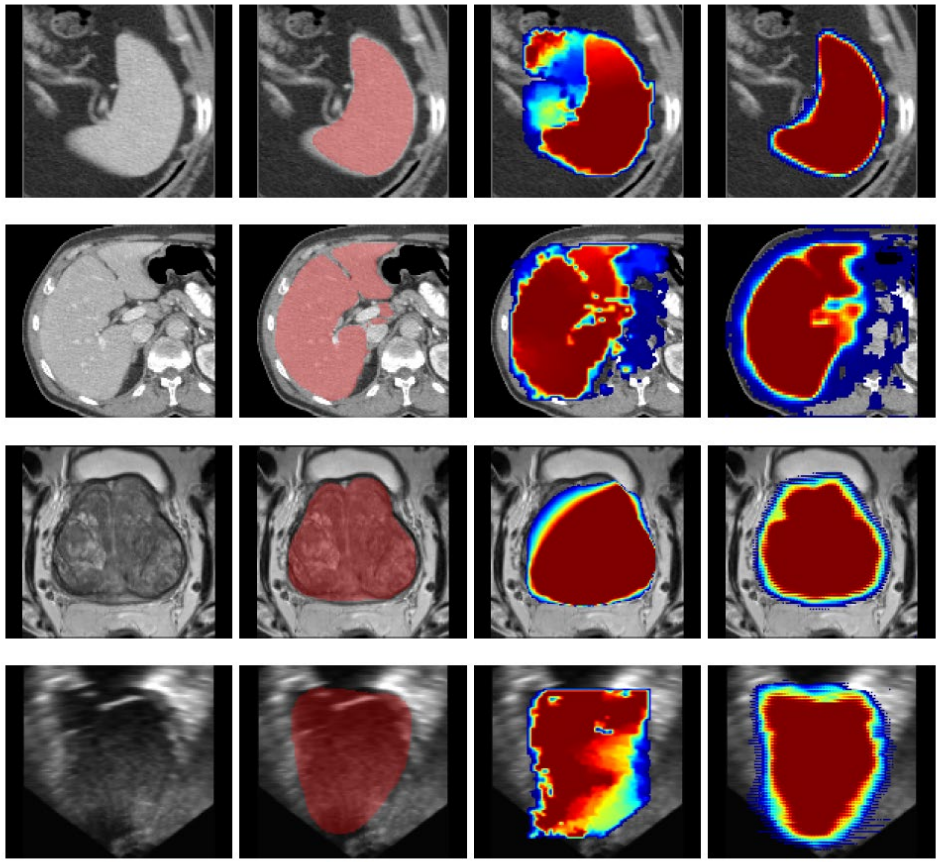


Fig. 2: Our results. We show (a) the image, (b) overlaid (full) ground truth (used for evaluation only), (c) initial random walker prediction, and (d) our final segmentation result produced by the weakly supervised FCN. We show qualitative results for top to bottom: spleen (CT), liver (CT), prostate (MRI), and left ventricle (US) segmentation.

Table 1: Summary of our weakly supervised segmentation results. This table compares the random walker initialization with weakly supervised training from extreme points with (w) and without (w/o) random walker (RW) regularization, and with RW regularization but without the extra extreme points channel as input to the network (w RW; no extr.). For reference, the performance on the same task under fully supervised training is shown.

Dice	**Spleen (CT)**	**Liver (CT)**	**Prostate (MRI)**	**LV (US)**
Rnd. walk. init.	0.852	0.822	0.709	0.808
Weak. sup. (w/o RW)	0.905	0.918	0.758	0.876
Weak. sup. (w RW; no extr.)	0.924	0.935	0.779	0.860
Weak. sup. (w RW)	**0.926**	**0.936**	**0.830**	**0.880**
fully supervised	*0.963*	*0.958*	*0.923*	*0.903*

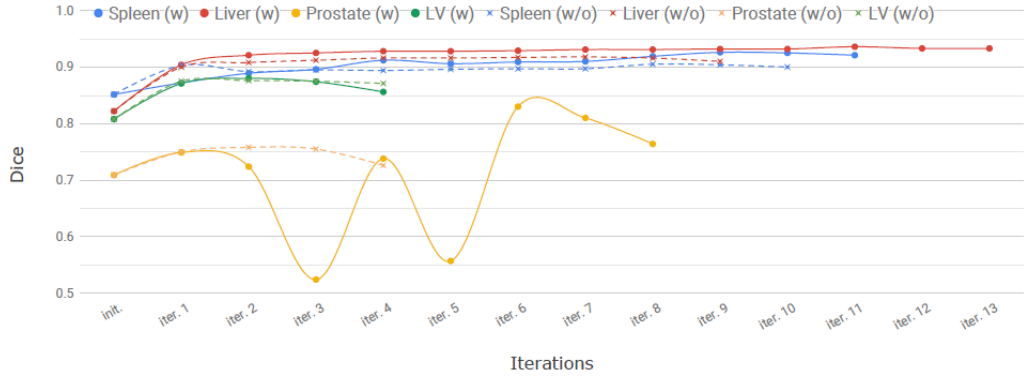


Fig. 3: Weakly supervised training from scribble based initialization. Each segmentation task is shown with (w) and without (w/o) random walker regularization after each round of FCN training.

Fast Interactive Object Annotation with Curve-GCN

Huan Ling[1,2*] Jun Gao[1,2*] Amlan Kar[1,2] Wenzheng Chen [1,2] Sanja Fidler[1,2,3]
[1]University of Toronto [2]Vector Institute [3]NVIDIA
{linghuan, jungao, amlan, wenzheng, fidler}@cs.toronto.edu

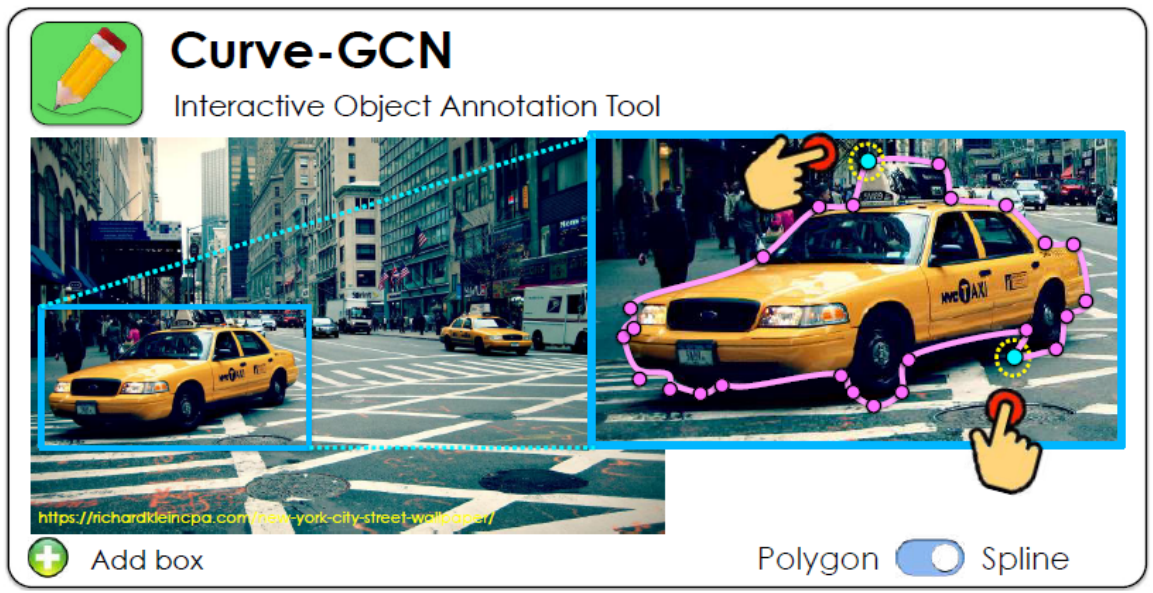


Figure 1: We propose Curve-GCN for interactive object annotation. In contrast to Polygon-RNN [7, 2], our model parametrizes objects with either polygons or splines and is trained end-to-end at a high output resolution.

Comparison

- Outperforms all existing approaches in automatic mode.
- Runs at 29.3ms in automatic, and 2.6ms in interactive mode, making it 10x and 100x faster than Polygon-RNN++.
- Outperforms the baselines in cross-domain annotation, that is, a model trained on Cityscapes is used to annotate general scenes, aerial, and medical imagery.
- DEXTR (Deep Extreme Cut) is pixel-wise, the worst scenario still requires many clicks. Differs from this method in that it directly predicts a polygon or spline around the object.
- Polygon-RNN. The recurrent nature of the model limits scalability to more complex shapes.

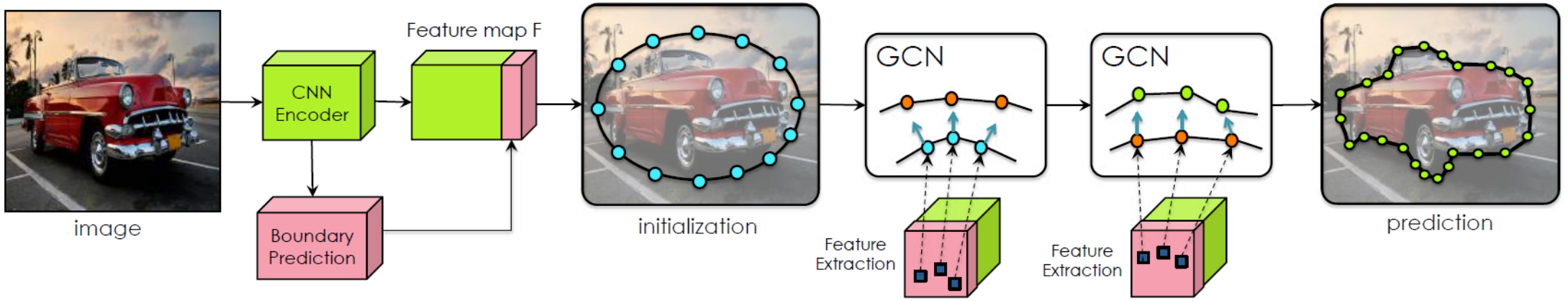


Figure 2: **Curve-GCN**: We initialize N control points (that form a closed curve) along a circle centered in the image crop with a diameter of 70% of image height. We form a graph and propagate messages via a Graph Convolutional Network (GCN) to predict a location shift for each node. This is done iteratively (3 times in our work). At each iteration we extract a feature vector for each node from the CNN's features F, using a bilinear interpolation kernel.

Point Matching Loss

$$L_{\text{match}}(\mathbf{p}, \mathbf{p}') = \min_{j \in [0 \cdots, K-1]} \sum_{i=0}^{K-1} \|p_i - p'_{(j+i)\%K}\|_1$$

Algorithm 1 Learning to Incorporate Human-in-the-Loop

```
while not converged do
    (rawImage, gtCurve) = Sample(Dataset)
    (predCurve, F) = Predict(rawImage)
    data = []
    for i in range(c) do
        corrPoint = Annotator(predictedCurve)
        data += (predCurve, corrPoint, gtCurve, F)
        predCurve = InteractiveGCN(predCurve, corrPoint)
                                          ▷ Do not stop gradients
    TrainInteractiveGCN(data)
```

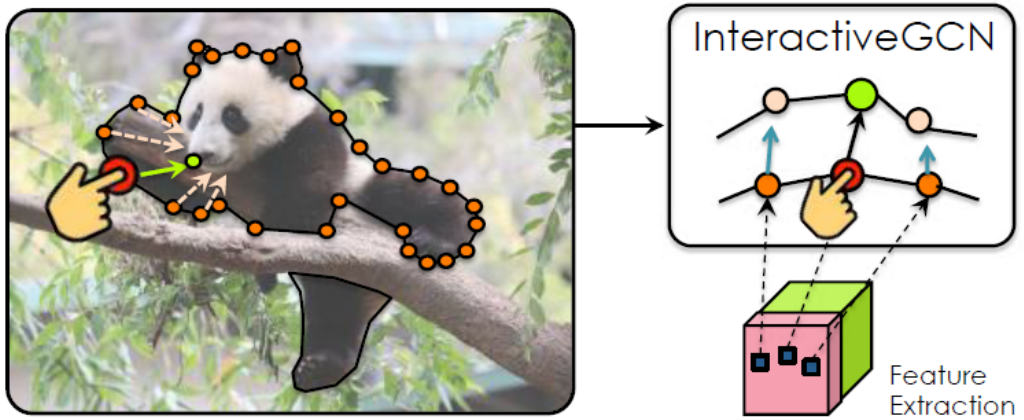


Figure 4: **Human-in-the-Loop**: An annotator can choose any wrong control point and move it onto the boundary. Only its immediate neighbors ($k = 2$ in our experiments) will be re-predicted based on this interaction.

Model	Bicycle	Bus	Person	Train	Truck	Motorcycle	Car	Rider	Mean
Polygon-RNN++	57.38	75.99	68.45	59.65	76.31	58.26	75.68	65.65	67.17
Polygon-RNN++ (with BS)	63.06	81.38	72.41	64.28	78.90	62.01	79.08	69.95	71.38
PSP-DeepLab	67.18	83.81	72.62	**68.76**	**80.48**	**65.94**	80.45	70.00	**73.66**
Polygon-GCN (MLoss)	63.68	81.42	72.25	61.45	79.88	60.86	79.84	70.17	71.19
+ DiffAcc	66.55	85.01	72.94	60.99	79.78	63.87	81.09	71.00	72.66
Spline-GCN (MLoss)	64.75	81.71	72.53	65.87	79.14	62.00	80.16	70.57	72.09
+ DiffAcc	**67.36**	**85.43**	**73.72**	64.40	80.22	64.86	**81.88**	**71.73**	**73.70**

Table 1: **Automatic Mode on Cityscapes.** We compare our Polygon and Spline-GCN to Polygon-RNN++ and PSP-DeepLab. Here, *BS* indicates that the model uses beam search, which we do not employ.

Model	mIOU	F at 1px	F at 2px
Polyrnn++ (BS)	71.38	46.57	62.26
PSP-DeepLab	73.66	47.10	62.82
Spline-GCN	**73.70**	**47.72**	**63.64**
DEXTR	79.40	55.38	69.84
Spline-GCN-EXTR	**79.88**	**57.56**	**71.89**

Table 2: **Different Metrics**. We report IoU & F boundary score. We favorably cross-validate PSP-DeepLab and DEXTR *for each metric* on val. *Spline-GCN-EXTR* uses extreme points as additional input as in DEXTR.

Model	Spline	Polygon
GCN	68.55	67.79
+ Iterative Inference	70.00	70.78
+ Boundary Pred.	72.09	71.19
+ DiffAcc	**73.70**	72.66

Table 3: **Ablation study** on Cityscapes. We use 3 steps when performing iterative inference. *Boundary Pred* adds the boundary prediction branch to our CNN.

Model	Time(ms)
Polygon-RNN++	298.0
Polygon-RNN++ (Corr.)	270.0
PSP-Deeplab	71.3
Polygon-GCN	28.7
Spline-GCN	29.3
Polygon-GCN (Corr.)	2.0
Spline-GCN (Corr.)	2.6

Table 4: **Avg. Inference Time** per object. We are 10× faster than Polygon-RNN++ in forward pass, and 100× for every human correction.

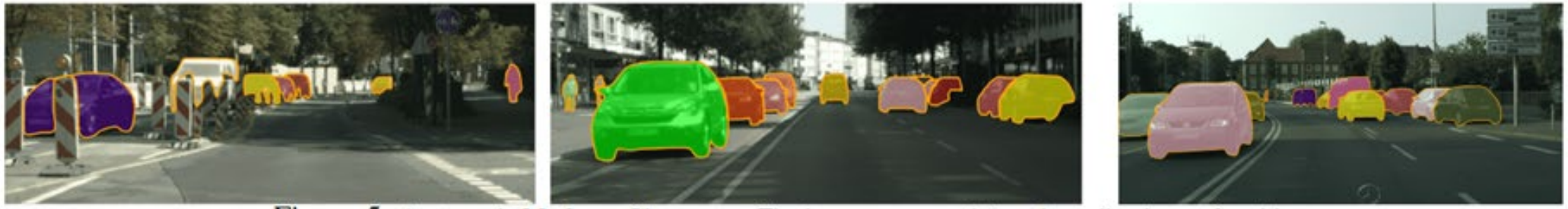


Figure 5: **Automatic Mode on Cityscapes**. The input to our model are bounding boxes for objects.

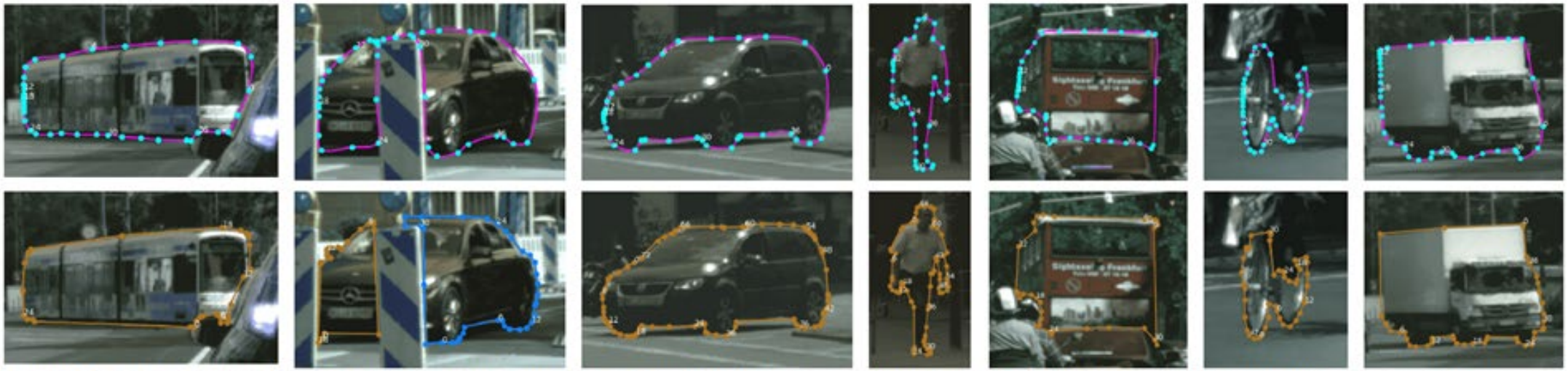


Figure 6: **Automatic mode on Cityscapes**. We show results for individual instances. **(top)** Spline-GCN, **(bottom)** ground-truth. We can observe that our model fits object boundaries accurately, and surprisingly finds a way to "cheat" in order to annotate multi-component instances.

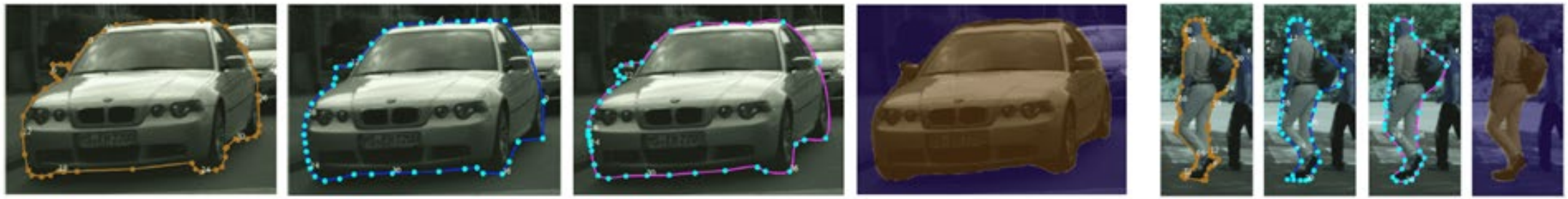


Figure 7: **Comparison in Automatic Mode**. From left to right: ground-truth, Polygon-GCN, Spline-GCN, PSP-DeepLab.

Model	Bicycle	Bus	Person	Train	Truck	Mcycle	Car	Rider	Mean	# clicks
Spline-GCN-BOX	69.53	84.40	76.33	69.05	85.08	68.75	83.80	73.38	76.29	2
PSP-DEXTR	74.42	87.30	79.30	73.51	85.42	73.69	85.57	76.24	79.40	4
Spline-GCN-EXTR	**75.09**	87.40	**79.88**	72.78	**86.76**	73.93	**86.13**	**77.12**	79.88	4
Spline-GCN-MBOX	70.45	88.02	75.87	76.35	82.73	70.76	83.32	73.49	77.62	2.4
+ One click	73.28	**89.18**	78.45	**79.89**	85.02	**74.33**	85.15	76.22	**80.19**	3.6

Table 5: **Additional Human Input.** We follow DEXTR [23] and provide a budget of 4 clicks to the models. Please see text for details.

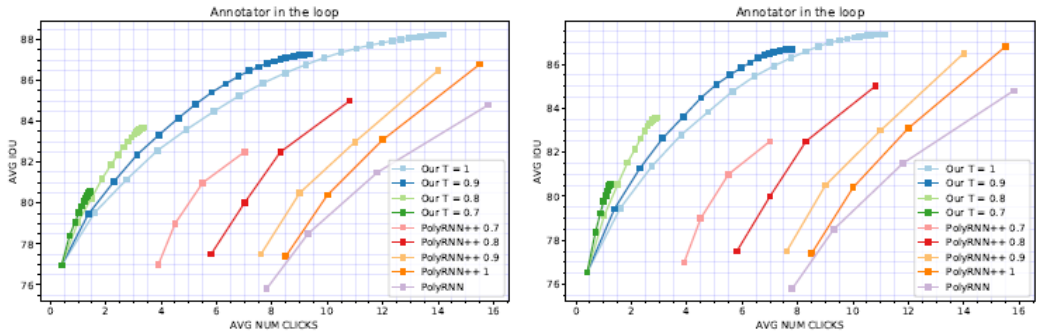


Figure 8: **Interactive Mode on Cityscapes**: (**left**) 40 control points, (**right**) 20 control points.

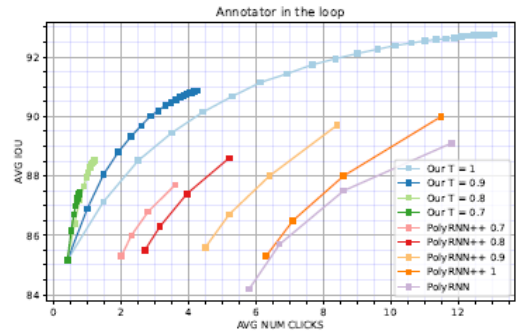


Figure 9: **Inter. Mode on KITTI**: 40 cps

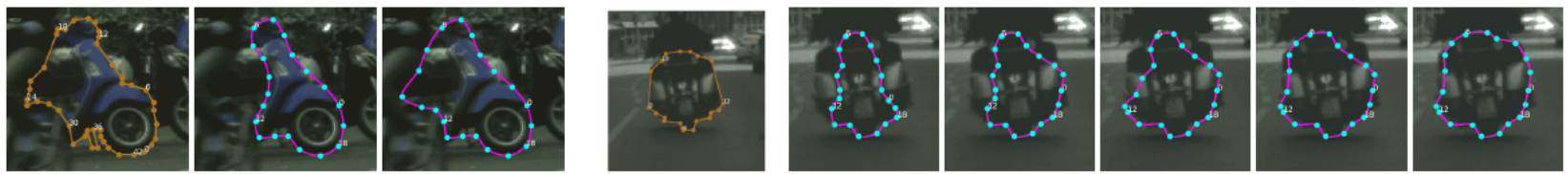


Figure 10: **Annotator in the Loop**: GT, 2nd column is the initial prediction from Spline-GCN, and the following columns show results after (simulated) annotator's corrections. Our corrections are local, and thus give more control to the annotator. However, they sometimes require more clicks (right).

Model	KITTI	ADE	Rooftop	Card.MR	ssTEM
Square Box (Perfect)	-	69.35	62.11	79.11	66.53
Ellipse (Perfect)	-	69.53	66.82	92.44	71.32
Polygon-RNN++ (BS)	83.14	71.82	65.67	80.63	53.12
PSP-DeepLab	83.35	72.70	57.91	74.11	47.65
Spline-GCN	**84.09**	**72.94**	**68.33**	78.54	58.46
+ finetune	**84.81**	77.35	**78.21**	**91.33**	-
Polygon-GCN	83.66	72.31	66.78	**81.55**	**60.91**
+ finetune	84.71	**77.41**	75.56	90.91	-

Table 6: **Automatic Mode on Cross-Domain**. We outperform PSP-DeepLab out-of-the-box. Fine-tuning on 10% is effective.

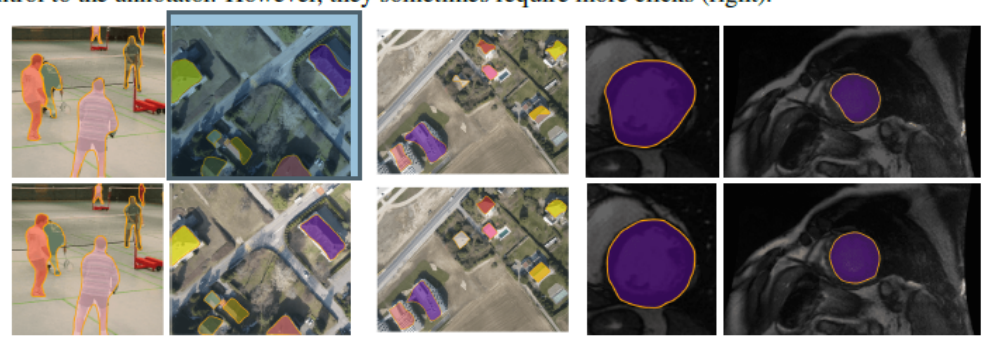


Table 7: **Automatic Mode for Cross-Domain**. (**top**) Out-of-the-box output of Cityscapes-trained models, (**bottom**) fine-tuned with 10% of data from new domain.