Beyond Fixed Grid: Learning Geometric Image Representation with a Deformable Grid

Jun Gao[1,2,3], Zian Wang[1,2], Jinchen Xuan[4], and Sanja Fidler[1,2,3]

[1]University of Toronto, [2]Vector Institute, [3]NVIDIA, [4]Peking University
{jungao, zianwang, fidler}@cs.toronto.edu, 1600012865@pku.edu.cn

Abstract. In modern computer vision, images are typically represented as a fixed uniform grid with some stride and processed via a deep convolutional neural network. We argue that deforming the grid to better align with the high-frequency image content is a more effective strategy. We introduce *Deformable Grid* (DEFGRID), a learnable neural network module that predicts location offsets of vertices of a 2-dimensional triangular grid, such that the edges of the deformed grid align with image boundaries. We showcase our DEFGRID in a variety of use cases, i.e., by inserting it as a module at various levels of processing. We utilize DEFGRID as an end-to-end *learnable geometric downsampling* layer that replaces standard pooling methods for reducing feature resolution when feeding images into a deep CNN. We show significantly improved results at the same grid resolution compared to using CNNs on uniform grids for the task of semantic segmentation. We also utilize DEFGRID at the output layers for the task of object mask annotation, and show that reasoning about object boundaries on our predicted polygonal grid leads to more accurate results over existing pixel-wise and curve-based approaches. We finally showcase DEFGRID as a standalone module for unsupervised image partitioning, showing superior performance over existing approaches. Project website: http://www.cs.toronto.edu/~jungao/def-grid.

1 Introduction

In modern computer vision approaches, an image is treated as a fixed uniform grid with a stride and processed through a deep convolutional neural network. Very high resolution images are typically processed at a lower resolution for increased efficiency, whereby the image is essentially blurred and subsampled. When fed to a neural network, each pixel thus contains a blurry version of the original signal mixing information from both the foreground and background, possibly causing higher sensitivity and reliance of the network to objects and their context. In contrast, in many of the traditional computer vision pipelines the high resolution image was instead partitioned into a smaller set of superpixels that conform to image boundaries, leading to more effective reasoning in downstream tasks. We follow this line of thought and argue that deforming the grid to better align with the high-frequency information content in the input is a more effective representation strategy. This is conceptually akin to superpixels
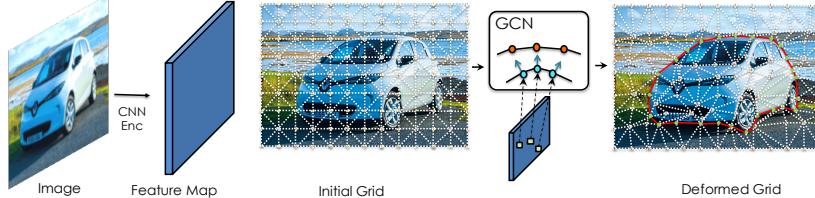
Fig. 1: **DefGrid** is a neural module that represents an image with a triangular grid. Initialized with an uniform grid, DEFGRID deforms grid's vertices, such that grid's edges align with image boundaries, while keeping topology fixed.

but conforming to a regular topology with geometric constraints thus still easily amenable for use with deep convolutional networks for downstream tasks.

Furthermore, tasks such as object mask annotation naturally require the output to be in the form of polygons with a manageable number of control points that a human annotator can edit. Previous work either parametrized the output as a closed curve with a fixed number of control points [27] or performed pixel-wise labeling followed by a (non-differentiable) polygonization step [26,39,29]. In the former, the predicted curves typically better utilize shape priors leading to "well behaved" predictions, however, the output is inherently limited in the genus and complexity of the shape it is able to represent. In contrast, pixel-wise methods can represent shapes of arbitrary genus, however, typically large input/output resolutions are required to produce accurate labeling around object boundaries. We argue that reasoning on a low-resolution polygonal grid that well aligns with image boundaries combines the advantages of the two approaches.

We introduce *Deformable Grid* (DEFGRID), a neural network module that represents an image with a 2-dimensional triangular grid. The basic element of the grid is a triangular cell with vertices that place the triangle in the image plane. DEFGRID is initialized with a uniform grid and utilizes a neural network that predicts location offsets of the triangle vertices such that the edges and vertices of the deformed grid align with image boundaries (Fig 1). We propose several carefully designed loss functions that encourage this behaviour. Due to the differentiability of the deformation operations, DEFGRID can be trained end-to-end with downstream neural networks as a plug-and-play module at various levels of deep processing. We showcase DEFGRID in various use cases: as a learn-able geometric image downsampling layer that affords high accuracy semantic segmentation at significantly reduced grid resolutions. Furthermore, when used to parametrize the output, we show that it leads to more effective and accurate results for the tasks of interactive object mask annotation. Our DEFGRID can also be used a standalone module for unsupervised image partitioning, and we show superior performance over existing superpixel-based approaches.

2   Related Works

We focus on the most relevant work in several related categories.

Deformable Structure: Deformable convolutions [10] predict position offsets of each cell in the convolutional kernel's grid with the aim to better capture object deformation. This is in contrast to our approach which deforms a triangular grid which is then exploited in downstream processing. Note that our approach does not imply any particular downstream neural architecture and would further benefit by employing deformable convolutions. Related to our work, [31,23] learn to deform an image such that the corresponding warped image, when fed to a neural network, leads to improved downstream tasks. Our DEFGRID, which is trained with both unsupervised and supervised loss functions to explicitly align with image boundaries, allows downstream tasks such as object segmentation to perform reasoning directly on the low-dimensional grid.

Image Partitioning: Polygonal image partitioning plays an important role in certain applications such as multi-view 3D object reconstruction [6] and graphics-based image manipulation [22]. Existing works tried to polygonize an image with triangles [6] or convex polygons [11]. [6] used Constrained Delaunay triangulation to get the triangular mesh. However, their method heavily relies either on having good key point locations or good edges in order to get boundary aligned triangles. [11] also relies on the initial line segment detection. In [2], a non-iterative method was proposed to obtain superpixels, followed by a polygonization method using a contour tracing algorithm. Our DEFGRID produces a triangular grid that conforms to image boundaries and is end-to-end trainable.

Superpixels: Superpixel methods aim to partition the image into regions of homogenous color while regularizing their shape and preserving image boundaries [12,32,28,17,5,1,2,25]. Many algorithms have been proposed mainly differing in the energy function they optimize and the optimization technique they utilize [32,28,37,12,17,1,2,40,41]. Most of these approaches produce superpixels with irregular topology, and the final segmentation map is often disconnected and needs postprocessing. Note also that energy is often hand designed, and inference is optimization based. Recently, SSN [24] made clustering-based approaches differentiable by softly assigning pixels to superpixels with the exponential function. SEAL [36] learns superpixels by exploiting segmentation affinities. Both of these methods produce superpixels with highly irregular boundaries and region topology, and thus they may not be trivially embedded in existing convolutional neural architectures [13]. To produce regular grid-like topology, superpixel lattices [30] partition the image recursively, finding horizontal and vertical paths with minimal boundary cost at each iteration. Unlike their approach, DEFGRID utilizes differentiable operations to predict boundary aligned triangular grids and is end-to-end trainable with both unsupervised and supervised loss functions.

3   Deformable Grid

Our DEFGRID is a 2-dimensional triangular grid defined on an image plane. The basic cell in the grid is a triangle with three vertices, each with a location that position the triangle in the image. Edges of the triangle thus represent line segments and are expected not to self-intersect across triangles. The topology of
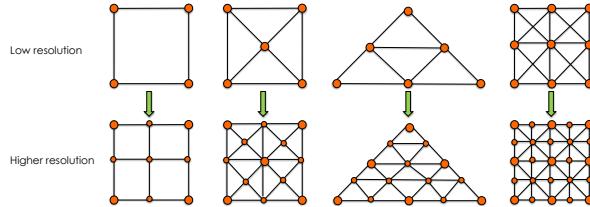
Fig. 2: **Different grid topologies**. We choose the last column for its flexibility in representing a variety of different edge orientations.

the grid is fixed and does not depend on the input image. The geometric grid thus naturally partitions an image into regular segments, as shown in Fig. 1.

We formulate our approach as deforming a triangular grid with uniformly initialized vertex positions to better align with image boundaries. The grid is deformed via a neural network that a predicts position offset for each vertex while ensuring the topology does not change (no self-intersections occur).

Our main intuition is that when the edges of the grid align with image boundaries, the pixels inside each grid cell have minimal variance of RGB values (or one-hot masks when we have supervision), and vice versa. We aim to minimize this variance in a differentiable way with respect to the positions of the vertices, to make it amenable to deep learning. We describe our DEFGRID formulation along with its training method in detail next. In Section 4, we show applications to different downstream tasks.

3.1   Grid Parameterization

Grid Topology: Choosing the right topology of the grid is an important aspect of our work. Since objects (and their parts) can appear at different scales in images, we ideally want a topology that can easily be subdivided to accommodate for this diversity. Furthermore, boundaries can be found in any orientation and thus the grid edges should be flexible enough to well align to any real edge. We experimentally tried four different topologies which are visualized in Fig. 2. We found the topology in the last column to outperform alternatives for its flexibility in representing different edge orientations. Note that our method is agnostic to the choice of topology and we provide a detailed comparison in the appendix.

Grid Representation: Let I be an input image. We denote each vertex of the grid in the image plane as $v_i = [x_i, y_i]^T$, where $i \in \{1, \cdots, n\}$ and n is the total number of vertices in the grid. Since the grid topology is fixed, the grid in the image is entirely specified by the positions of its vertices $\mathbf{v}$. We denote each triangular cell in the grid with its three vertices as $C_k = [v_{a_k}, v_{b_k}, v_{c_k}]$, with $k \in \{1, \cdots, K\}$ indexing the grid cells. We uniformly initialize the vertices on the 2D image plane, and define DEFGRID as a neural network h that predicts the relative offset for each vertex:

$$\{\Delta x_i, \Delta y_i\}_{i=1}^n = h(\mathbf{v}, I). \tag{1}$$

We discuss the choice of h in Section 4. The deformed vertices are thus:

$$v_i = [x_i + \Delta x_i, y_i + \Delta y_i]^T, \quad i = 1, \ldots, n. \tag{2}$$

3.2 Training of DefGrid

We now discuss training of the grid deforming network h using a variety of unsupervised loss functions. We want all our losses to be differentiable with respect to vertex positions to allow for the gradient to be backpropagated analytically.

Differentiable Variance: As the grid deforms (its vertices move), the grid cells will cover different pixel regions in the image. Our first loss aims to minimize the variance of pixel features in each grid cell. Each pixel p_i has a feature vector $\mathbf{f}_i$, which in our case is chosen to contain RGB values. When supervision is available in the form of segmentation masks, we can optionally append a one hot vector representing the class of the mask. Pixel's position in the image is denoted with $p_i = [p_i^x, p_i^y]^T$, $i \in \{1, \cdots, N\}$, with N indicating the total number of pixels in the image. Variance of a cell C_k is defined as:

$$V_k = \sum\nolimits_{p_i \in S_k} ||\mathbf{f}_i - \bar{\mathbf{f}}_k||_2^2, \tag{3}$$

where S_k denotes the set of pixels inside C_k, and $\bar{\mathbf{f}}_k$ is the mean feature of C_k: $\bar{\mathbf{f}}_k = \frac{\sum_{p_i \in S_k} \mathbf{f}_i}{\sum_{p_i \in S_k} 1}$. Note that this definition of variance is not naturally differentiable with respect to the vertex positions. We thus reformulate the variance function by softly assigning every pixel p_i to each grid cell C_k with a probability $P_{i \to k}(\mathbf{v})$:

$$P_{i \to k}(\mathbf{v}) = \frac{\exp(\text{SignDis}(p_i, C_k)/\delta)}{\sum_{j=1}^K \exp(\text{SignDis}(p_i, C_j)/\delta)}, \tag{4}$$

$$\text{SignDis}(p_i, C_k) = \begin{cases} -\text{Dis}(p_i, C_k), & \text{if } p_i \text{ is outside } C_k, \\ \text{Dis}(p_i, C_k), & \text{if } p_i \text{ is inside } C_k. \end{cases} \tag{5}$$

$$\text{Dis}(p_i, C_k) = \min(D(p_i, v_{a_k} v_{b_k}), D(p_i, v_{b_k} v_{c_k}), D(p_i, v_{c_k} v_{a_k})), \tag{6}$$

where $D(p_i, v_i v_j)$ is the L1 distance between a pixel and a line segment $v_i v_j$, and δ is a hyperparameter to control the slackness. We use $P_{i \to k}(\mathbf{v})$ to indicate that the probability of assignment depends on the grid's vertex positions, and is in our case a differentiable function. Intuitively, if the pixel is very close or inside a cell, then $P_{i \to k}(\mathbf{v})$ is close to 1, and close to 0 otherwise. To check whether the pixel is inside a cell, we calculate the barycentric weight of the pixel with respect to three vertices of the cell. If all the barycentric weights are between 0 and 1, then the pixel is inside, otherwise it falls outside the triangle.

We now re-define the cell's variance as follows:

$$\tilde{V}_k(\mathbf{v}) = \sum\nolimits_{i=1}^N P_{i \to k}(\mathbf{v}) \cdot ||\mathbf{f}_i - \bar{\mathbf{f}}_k||_2^2, \tag{7}$$

which is therefore a differentiable function of grid's vertex positions. Our variance-based loss function aims to minimize the sum of variances across all grid cells:

$$L_{\text{var}}(\mathbf{v}) = \sum\nolimits_{k=1}^K \tilde{V}_k(\mathbf{v}) \tag{8}$$

Differentiable Reconstruction: Inspired by SSN [24], we further differentiably reconstruct an image using the deformed grid by taking into account the probability of assignments: $P_{i \to k}(\mathbf{v})$. Intuitively, we represent each cell using its mean feature $\bar{\mathbf{f}}_k$, and "paste" it back into the image plane according to the positions of the cell's deformed vertices. Specifically, we reconstruct each pixel in an image by softly gathering information from each grid cell using $P_{i \to k}(\mathbf{v})$:

$$\hat{\mathbf{f}}_i(\mathbf{v}) = \sum_{k=1}^{K} P_{i \to k}(\mathbf{v}) \cdot \bar{\mathbf{f}}_k, \tag{9}$$

The reconstruction loss is the distance between the reconstructed pixel feature and original pixel feature:

$$L_{\mathrm{recons}}(\mathbf{v}) = \sum_{i=1}^{N} ||\hat{\mathbf{f}}_i(\mathbf{v}) - \mathbf{f}_i||_1. \tag{10}$$

We experimentally found that L1 distance works better than L2.

Regularization: To regularize the shape of the grid and prevent self-intersections, we introduce two regularizers. We employ an **Area balancing** loss function that encourages the areas of the cells to be similar, and thus, avoids self-intersections by minimizing the variance of the areas:

$$L_{\mathrm{area}}(\mathbf{v}) = \sum_{j=1}^{K} ||a_k(\mathbf{v}) - \bar{a}(\mathbf{v})||_2^2, \tag{11}$$

where $\bar{a}$ is the mean area and a_k is the area of cell C_k. We also utilize **Laplacian regularization** following works on 3D mesh prediction [38,8]. In particular, this loss encourages the neighboring vertices to move along similar directions with respect to the center vertex:

$$L_{\mathrm{lap}}(\mathbf{v}) = \sum_{i=1}^{n} ||\Delta_i - \frac{1}{||\mathcal{N}(i)||} \sum_{j \in \mathcal{N}(i)} \Delta_j||_2^2, \tag{12}$$

where $\Delta_i = [\Delta x_i, \Delta y_i]^T$ is the predicted offset of vertex v_i and $\mathcal{N}(i)$ is the set of neighboring vertices of vertex v_i.

The final loss to train our network h is a weighted sum of all the above terms:

$$L_{\mathrm{def}} = L_{\mathrm{var}} + \lambda_{\mathrm{recons}} L_{\mathrm{recons}} + \lambda_{\mathrm{area}} L_{\mathrm{area}} + \lambda_{\mathrm{lap}} L_{\mathrm{lap}}, \tag{13}$$

where $\lambda_{\mathrm{recons}}, \lambda_{\mathrm{area}}, \lambda_{\mathrm{lap}}$ are hyperparameters that balance different terms.

4   Applications

Our DEFGRID supports many computer vision tasks that are done on fixed image grids today. We discuss three possible use cases in this section. DEFGRID can be inserted as a plug-and-play module at several levels of processing. By inserting it at the input level we utilize DEFGRID as a learnable geometric downsampling layer to replace standard pooling methods. We showcase its effectiveness through an application to semantic segmentation in Section 4.1. We further show an application to object mask annotation in Section 4.2 where we propose a model that reasons on the boundary-aligned grid output by a deep DEFGRID to produce object polygons. Lastly, we showcase DEFGRID as a standalone module for unsupervised image partitioning in Section 4.3.
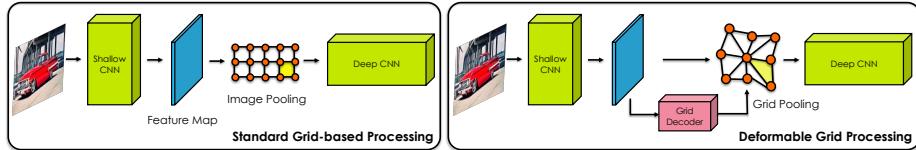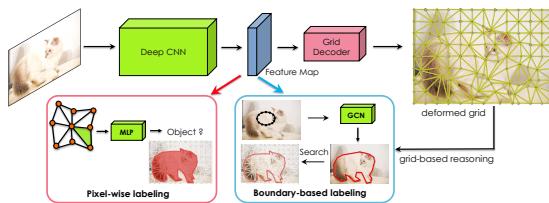
Fig. 3: Feature pooling in fixed grid versus DEFGRID. DEFGRID can easily be used in existing deep CNNs and perform learnable downsampling.

Fig. 4: Object mask annotation by reasoning on a DEFGRID's boundary-aligned grid. We support both pixel-wise labeling and curve-based tracing.

4.1   Learnable Geometric Downsampling

Semantic segmentation of complex scenes typically requires high resolution images as input and thus produces high resolution feature maps which are computationally intensive. Existing deep CNNs often take downsampled images as input and use feature pooling and bottleneck structures to relieve the memory usage [18,19,42]. We argue that downsampling the features with our DEFGRID can preserve finer geometric information. Given an arbitrary deep CNN architecture, we propose to insert a DEFGRID using a shallow CNN encoder to predict a deformed grid. The predicted boundary-preserving grid can be used for geometry-aware feature pooling. Specifically, to represent each cell we can apply mean or max pooling by averaging or selecting maximum feature values within each triangular cell. Due to the regular grid topology, these features can be directly passed to a standard CNN. Note that the grid pooling operation warps the original feature map from the image coordinates to grid coordinates. Thus the final output (predicted semantic segmentation) is pasted back into the image plane by checking in which grid's cell the pixel lies. The full pipeline is end-to-end differentiable. We can jointly train the model in a multi-task manner with a cross-entropy loss for the semantic segmentation branch and grid deformation loss in Eq. 13. The DEFGRID module is lightweight and thus bears minimal computational overhead. The architecture is illustrated in Fig 3.

4.2   Object Mask Annotation

Object mask annotation is the problem of outlining a foreground object given a user-provided bounding box [7,3,27,29,39]. Two dominant approaches have been proposed to tackle this task. The first approach utilizes a deep neural network to predict a pixel-wise mask [29,39,26]. The second approach tries to outline the

boundary with a polygon/spline [20,14,7,3,27]. Our DEFGRID supports both approaches, and improves upon them via a polygonal grid-based reasoning (Fig. 4).

Boundary-based segmentation: We formulate the boundary-based segmentation as a minimal energy path searching problem. We search for a closed path along the grid's edges that has minimal Distance Transform energy[1]:

$$Q = \underset{Q \in \mathcal{Q}}{\arg\min} \sum\nolimits_{i=1}^{M} DT(v_{Q_i}, v_{Q_{(i+1)\%M}}), \tag{14}$$

where $\mathcal{Q}$ denotes the set of all possible paths on our grid, and M is the length of path Q. We first predict a distance transform energy map for an object using a deep network trained with the L2 loss. We then compute the energy in each grid vertex via bilinear sampling. We obtain the energy for each grid edge by averaging the energy values for the points along the line defined by two vertices. Note that directly searching on the grid may result in many local minima. We employ Curve-GCN [27] to predict 40 seed points and snap each of these points to the grid vertex that has the minimal energy among its top-k closest vertices. Then for each neighboring seed points pair, we use Dijkstra algorithm to find the minimal energy path between them. We provide algorithm details in the appendix. Our approach improves over Curve-GCN in two aspects: 1) it better aligns with image boundaries as it explicitly reasons on our boundary-aligned grid, 2) since we search for a minimal energy path between neighboring points output by Curve-GCN, our approach can handle objects with more complex boundaries that cannot be represented with only 40 points.

Pixel-wise segmentation: Rather than producing a pixel-wise mask, we instead predict the class label for each grid cell. Specifically, we first use a deep neural network to obtain a feature map from the image. Then, for every grid cell, we average pool the feature of all pixels that are inside the cell, and use a MLP network to predict the class label for each cell. The model is trained with the cross-entropy loss. As the grid boundary aligns well with the object boundary, pooling the feature inside the grid is more efficient and effective for learning.

4.3 Unsupervised Image Partitioning

We can already view our deformed triangular cells as "superpixels", trained with unsupervised loss functions. We can go further and cluster cells by using the affinity between them. In particular, we view the deformed grid as an undirected weighted graph where each grid cell is a node and an edge connects two nodes if they share an edge in the grid. The weight for each edge is the affinity between two cells, which can be calculated using RGB values of pixels inside the cells.

Different clustering techniques can be used and exploring all is out of scope for this paper. To show the effectiveness of DEFGRID as an unsupervised image partitioning method, we here utilize simple greedy agglomerative clustering. We average the affinity to represent a new node after merging. Clustering stops when we reach the desired number of superpixels or the affinity is lower than a

---

[1] The DT energy for each pixel is its distance to the nearest boundary.

Downsampling Ratio	1/4			1/8			1/16			1/32		
Metric	mIoU	F (4px)	F (16px)	mIoU	F (4px)	F (16px)	mIoU	F (4px)	F (16px)	mIoU	F (4px)	F (16px)
Strided Convolution	65.76	60.59	73.97	59.18	53.80	70.14	51.02	47.33	60.41	41.03	44.12	51.05
Max Pooling	66.32	60.92	74.18	59.83	54.22	70.71	52.93	49.00	63.59	42.44	45.23	52.47
Average Pooling	66.53	61.09	74.39	59.45	0.5361	70.67	52.01	47.58	61.33	43.54	45.37	53.39
Grid Max Pooling	67.87	64.37	75.10	64.75	61.02	72.95	55.87	**53.98**	**66.62**	47.20	**48.74**	**60.73**
Grid Average Pooling	**67.91**	**64.99**	**75.43**	**65.36**	**61.12**	**73.03**	**56.94**	53.77	66.38	**48.30**	48.67	60.62

Table 1: **Learnable downsampling** on Cityscapes Semantic Segm benchmark.

threshold. Note that our superpixels are, by design, polygons. Note that supervised loss functions are naturally supported in our framework, however we do not explore them in this paper.

5 Experiments

We extensively evaluate DEFGRID on downstream tasks. We first show application to learnable downsampling for semantic segmentation. We then evaluate on the object annotation task with boundary-based and pixel-wise methods. We finally show the effectiveness of DEFGRID for unsupervised image partitioning.

5.1 Learnable Geometric Downsampling

To verify the effectiveness of our DEFGRID as an effective downsampling method, we compare it with (fixed) image grid feature pooling methods as baselines, namely max/average pooling and stride convolution, on the Cityscapes [9] semantic segmentation benchmark. The baseline methods perform max/average pooling, or stride convolution on the shallow feature map, while our grid pooling methods apply the max/average pooling on deformed triangle cells. We compare our grid pooling with baselines when the height and width of the feature map is downsampled to 1/4, 1/8, 1/16 and 1/32 of the original image size. We use a modified ResNet50 [18] which is more lightweight than SOTA models [35].

Evaluation Metrics: Following [39,26,27], we evaluate the performance using mean Intersection-over-Union (mIoU), and the boundary F score, with 4 and 16 pixels threshold on the full image. All metrics are averaged across all classes.

Results: Performances (mIoU and boundary F scores) are reported in Table 1. Our DEFGRID pooling methods consistently outperform the baselines, especially on the boundary score. We benefit from the edge-aligned property of the DEFGRID coordinates. At 1/8 with 1/4 downsampling ratios, the baseline performance drops significantly due to missing the tiny instances, while our DEFGRID pooling methods cope with this issue more gracefully. We also outperform baselines when the downsampling ratio is small, showing an efficient usage of limited spatial capacity. We visualize qualitative results for the predicted grids in Fig. 5. Our DEFGRID better aligns with boundaries and thus what the downstream network "sees" is more informative than the fixed uniform grid.

Fig. 5: **Illustration of learnable downsampling:** We show DEFGRID and its reconstructed image (left), comparing it to FIXEDGRID (right). [Please zoom in]

Model	Bicycle	Bus	Person	Train	Truck	Mcycle	Car	Rider	mIoU	F1	F2
Curve-GCN [27]	75.40	86.02	79.87	82.89	86.44	75.69	90.21	76.61	81.64	59.45	75.43
Pixel-wise	74.95	86.19	80.35	81.10	86.10	75.82	89.78	77.14	81.43	60.25	74.49
FIXEDGRID	75.10	85.73	79.84	84.35	86.02	75.97	89.76	76.56	81.67	59.01	74.77
DEFGRID	**75.46**	**86.29**	**80.40**	**84.91**	**86.58**	**76.13**	**90.42**	**77.20**	**82.17**	**61.94**	**77.04**

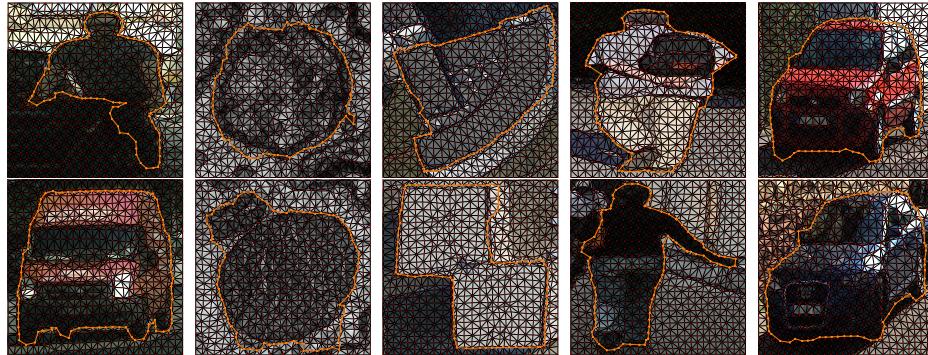Table 2: **Boundary-based object annotation** on Cityscapes-Multicomp.

Fig. 6: **Deformed Grid:** We show examples of predicted grids both on in-domain (Cityscapes) and cross-domain images (Medical, Rooftop, ADE, KITTI). Orange line is the obtained minimal energy path along the grid's edges.

5.2   Object Annotation

Dataset: Following [7,3,27,39,26], we train and test both of our instance segmentation models on the Cityscapes dataset [9]. We assume the bounding box of each object is provided by the annotator and the task is to trace the boundary of the object. We evaluate under two different settings, depending on the model. To compare with pixel-wise methods, we follow the setting proposed in DELSE [39], where an image is first cropped with a 10 pixel expansion around the ground truth bounding box, and resized to the size of 224×224. This setting is

Method	Metrics	KITTI	ADE	Rooftop	Card.MR	ssTEM
Curve-GCN [27]	mIoU	87.43	76.71	81.11	86.18	68.97
	F1	64.90	39.37	30.99	62.73	44.74
	F2	78.40	52.86	45.08	78.22	59.85
Pixel-wise	mIoU	86.99	78.23	80.81	88.00	69.37
	F1	62.73	42.88	28.44	62.63	46.07
	F2	77.17	56.15	42.01	77.94	59.77
DEFGRID	mIoU	**88.05**	**78.54**	**83.10**	**89.01**	**71.82**
	F1	**66.70**	**43.54**	**34.39**	**65.31**	**50.31**
	F2	**80.23**	**57.20**	**49.79**	**81.92**	**65.14**

Table 3: **Cross-Domain Results** for boundary-based methods. DEFGRID significantly outperforms baselines, particularly evident in F-scores.

referred to as Cityscapes-Stretch. Our boundary-based annotation model builds on top of Curve-GCN which predicts a polygon that is topologically equivalent to a sphere. To compare with the baseline, we thus assume that the annotator creates a box around each connected component of the object individually. We process each box in the same way as above, and evaluate performance on all component boxes. We refer to this setting as Cityscapes-Multicomp.

Boundary-based Object Annotation

Network Architecture: We use the image encoder from DELSE [39], and further add three branches to predict grid deformation, Curve-GCN points and Distance Transform energy map. For each branch, we first separately apply one $3{\times}3$ conv filter to the feature map, followed by batch normalization [21] and ReLU activation. For grid deformation, we extract the feature for each vertex with bilinear interpolation, and use a GCN to predict the offset for each vertex. For predicting spoints, we follow Curve-GCN [27]. For the DT energy, we apply two $3{\times}3$ conv filters with batch normalization and ReLU activation.

Baselines: For Curve-GCN [27], we compare with Spline-GCN, as it gives better performance than Polygon-GCN in the original paper [27]. We use the official codebase but instead use our image encoder to get the feature map (with negligible performance gap) for a fair comparison. We also compare with pixel-wise methods, where we add two conv filters after the image encoder, which is similar to DELSE [39] and DEXTR [29] but without extreme points. We further compare to our version of the model where the grid is fixed, referred to as FIXEDGRID.

Results: Table 2 reports results. Our method outperforms baselines in all metrics. Performance gains are particularly significant in terms of F-scores, even when compared with pixel-wise methods. Since network details are the same across the methods, these results signify the importance of reasoning on the grid. Compared with FIXEDGRID at different grid sizes in Fig. 7, DEFGRID achieves superior performance. We attribute the performance gains due to better alignment with boundaries and more flexibility in tracing a longer contour. We show qualitative results in Fig. 6 and 8.
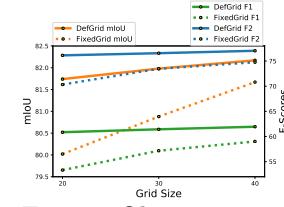

Fig. 7: Object annotation for Cityscapes: FIXEDGRID vs DEFGRID.

Dataset	Model	Round 0 (Automatic)			Round 1			Round 2			Round 3			Round 4			Round 5		
		mIoU	F1	F2	mIoU	F1	F2	mIoU	F1	F2	mIoU	F1	F2	mIoU	F1	F2	mIoU	F1	F2
CityScapes [9]	CurveGCN [27]	80.74	58.01	73.67	83.04	60.47	76.35	84.58	64.42	79.42	85.56	67.41	81.40	86.27	69.87	82.94	86.79	71.91	84.12
	DefGrid	81.98	61.43	76.56	84.48	66.59	80.96	85.73	69.98	83.41	86.41	72.05	84.71	86.81	73.60	85.51	87.12	74.84	86.20
ADE [43]	CurveGCN [27]	76.71	39.37	52.86	79.26	42.43	56.62	81.09	46.04	60.51	82.47	49.61	64.00	83.44	52.56	66.61	84.38	55.42	69.00
	DefGrid	78.54	43.54	57.20	80.67	48.57	62.76	82.27	52.33	66.45	83.43	55.60	69.37	84.23	58.22	71.79	84.69	60.03	73.29
KITTI [15]	CurveGCN [27]	87.43	64.90	78.40	89.41	69.20	82.84	90.50	73.11	85.84	91.22	75.94	87.73	91.67	78.16	89.20	92.00	79.90	90.21
	DefGrid	88.05	66.70	80.23	89.94	70.80	84.16	90.90	74.65	87.12	91.45	77.08	88.81	91.70	78.67	89.78	91.87	79.72	90.33
Rooftop [34]	CurveGCN [27]	81.11	30.99	45.08	83.94	34.00	49.24	85.53	37.42	53.39	86.72	41.39	57.85	87.68	45.29	61.88	88.28	48.96	65.60
	DefGrid	83.10	34.39	49.79	85.37	39.60	56.04	86.77	43.74	60.73	87.53	47.30	64.51	88.19	49.95	67.50	88.55	52.20	69.66
Card.MR [33]	CurveGCN [27]	86.18	62.73	78.22	89.22	68.26	84.91	90.42	73.10	89.16	91.25	76.94	91.94	92.00	80.67	94.45	92.45	83.45	95.99
	DefGrid	89.01	65.31	81.92	91.01	72.28	88.83	91.90	77.64	92.74	92.55	81.03	94.89	92.96	83.24	95.98	93.31	85.47	96.96
ssTEM [16]	CurveGCN [27]	68.97	44.74	59.85	71.66	47.81	63.21	74.75	52.78	68.82	76.57	56.88	72.99	78.06	60.67	76.50	79.22	63.84	79.17
	DefGrid	71.82	50.31	65.14	73.51	55.43	70.44	75.25	60.64	75.69	77.28	65.18	79.72	78.46	68.44	82.24	78.89	70.59	83.62

Table 4: **Interactive annotation results** on both in-domain (first two lines) and cross domain datasets. Our DefGrid starts with a higher automatic performance and keeps this gap across (simulated) annotation rounds.

Model	Bicycle	Bus	Person	Train	Truck	Mcycle	Car	Rider	mIoU	F1	F2
DELSE* [39]	74.32	**88.85**	80.14	**80.35**	86.05	74.10	86.35	76.74	80.86	60.29	74.40
PolyTransform [26]	74.22	88.78	80.73	77.91	86.45	**74.42**	86.82	**77.85**	80.90	62.33	76.55
OurBack + SLIC [1]	73.88	85.47	79.80	77.97	86.32	72.62	87.85	76.14	80.01	57.95	72.17
DefGrid	**74.82**	87.09	**80.87**	**81.05**	**87.52**	73.44	**89.19**	77.36	**81.42**	**63.38**	**76.89**

Table 5: **Pixel-based methods** on Cityscapes-Stretch. Note that PolyTransform [26] uses 512x512 resolution, while other methods use 224x224 resolution.

Cross Domain Results: Following [27], we evaluate models' ability in generalizing to other datasets out of the box. Quantitative and qualitative results are reported in Table 3 and Fig. 6, Fig 8, respectively. We outperform all baselines on all datasets, in terms of all metrics, and the performance gains are particularly evident for the F-scores. Qualitatively, in all cross-domain examples the predicted grid's edges align well with real object boundaries (without any finetuning), demonstrating superior generalization capability for DefGrid.

Interactive Instance Annotation: We follow Curve-GCN [27] and also report performance for the interactive setting in which an annotator corrects errors by moving the predicted polygon vertices. We follow the original setting but restrict our reasoning on the deformed grid. Results for different simulated rounds of annotation are reported in Table 4 with evident performance gains.

Pixel-wise Object Instance Annotation

Network Architecture: To predict the class label for each deformed grid's cell, we first average the feature of all pixels inside each cell, and use a 4-layer MLP to predict the probability of foreground/background. For the hyperparameters and architecture details, we refer to the appendix.

Results: Table 5 provides quantitative results. We show qualitative results in the appendix. Predicting the (binary) class label over the deformed grid's cells

Fig. 8: **Qualitative results:** Note that the method employs ground-truth boxes. The first row is from Cityscapes. In the bottom two rows, from left to right: Medical, KITTI, Rooftop, ADE.
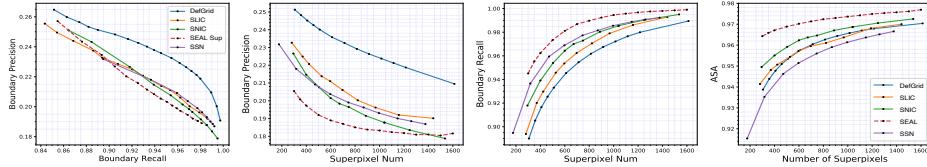

Fig. 9: **Unsupervised Image Partitioning**. From left to right: BP-BR, BP, BR and ASA. We use dotted line to represent supervised method.

achieves higher performance than carefully designed pixel-wise baselines, demonstrating the effectiveness of reasoning on our deformed grid.

5.3   Unsupervised Image Partitioning

Dataset: Following SSN [24], we train the model on 200 training images in the BSDS500 [4] and evaluate on 200 test images. Details are provided in appendix.

Evaluation Metric: Following the SSN and SEAL [24,36], we use Achievable Segmentation Accuracy (ASA), Boundary Precision (BP) and Boundary Recall (BR) to evaluate the performance of superpixels. For BP and BR, we set the tolerance to be 3 pixels. The evaluation scripts are from SEAL[2].

Baselines: We compare our method with both traditional superpixel methods, SLIC [1], SNIC [2], and deep learning based method SSN [24], SEAL [36]. Note that SEAL not only utilizes ground-truth annotation for training, but also is

---

[2] https://github.com/wctu/SEAL

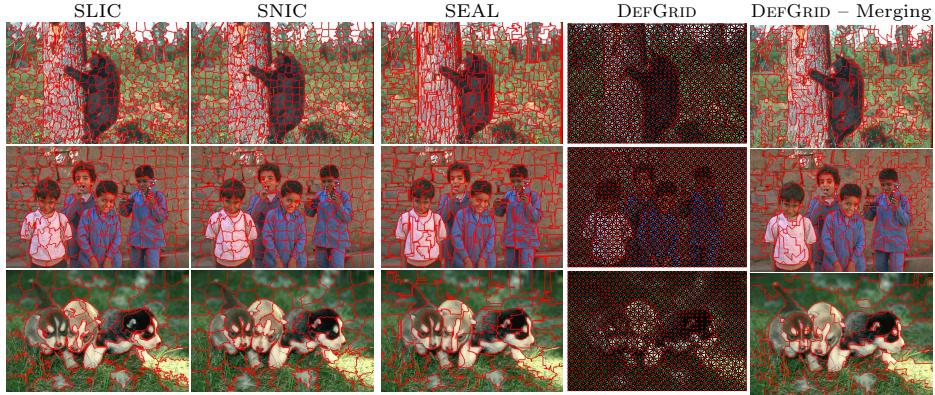| SLIC	SNIC	SEAL	DEFGRID	DEFGRID – Merging

Fig. 10: **Unsupervised Image Partitioning.** We compare the DEFGRID and results after clustering with existing superpixel baselines. [Please zoom in]

trained on validation set. We use the official codebase and trained model provided by the authors. We perform all comparisons at different numbers of superpixels.

Results: Quantitative and qualitative results are presented in Fig 9, Fig 10, respectively. The deformed grid aligns well with object boundary and outperforms all unsupervised baselines in terms of BP and BP-BR curve, with comparable performance with counterparts in terms of ASA and BR. Our intuition is that, since the edge between two vertices in our grid is constrained to be a straight line, while the ground truth annotation is labelled pixel-by-pixel, our grid sacrifices a little boundary recall while achieving higher boundary precision. It is worth to note that our method outperforms the supervised method SEAL [36]. This reflects the fact that an appearance feature provides a useful signal for training our DEFGRID , and our method effectively utilizes this signal.

6   Conclusion

In this paper, we proposed to deform a regular grid to better align with image boundaries as a more efficient way to process images. Our DEFGRID is a neural network that predicts offsets for vertices in a grid to perform the alignment, and can be trained entirely with unsupervised losses. We showcase our approach in several downstream tasks with significant performance gains. Our method produces accurate superpixel segmentations, is significantly more precise in outlining object boundaries particularly on out of domain datasets, and leads to a large improvements for semantic segmentation. We hope that our DEFGRID benefits other computer vision tasks when combined with deep networks.

References

1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels. Tech. rep. (2010)
2. Achanta, R., Susstrunk, S.: Superpixels and polygons using simple non-iterative clustering. In: CVPR. pp. 4651–4660 (2017)
3. Acuna, D., Ling, H., Kar, A., Fidler, S.: Efficient interactive annotation of segmentation datasets with polygon-rnn++. In: CVPR (2018)
4. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence **33**(5), 898–916 (2010)
5. den Bergh, M.V., Roig, G., Boix, X., Manen, S., Gool, L.V.: Online video superpixels for temporal window objectness. In: ICCV (2013)
6. Bódis-Szomorú, A., Riemenschneider, H., Van Gool, L.: Superpixel meshes for fast edge-preserving surface reconstruction. In: CVPR. pp. 2011–2020 (2015)
7. Castrejon, L., Kundu, K., Urtasun, R., Fidler, S.: Annotating object instances with a polygon-rnn. In: CVPR (2017)
8. Chen, W., Gao, J., Ling, H., Smith, E., Lehtinen, J., Jacobson, A., Fidler, S.: Learning to predict 3d objects with an interpolation-based differentiable renderer. In: Advances In Neural Information Processing Systems (2019)
9. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3213–3223 (2016)
10. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 764–773 (2017)
11. Duan, L., Lafarge, F.: Image partitioning into convex polygons. In: CVPR. pp. 3119–3127 (2015)
12. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision **59**(2), 167–181 (2004)
13. Gadde, R., Jampani, V., Kiefel, M., Kappler, D., Gehler, P.V.: Superpixel convolutional networks using bilateral inceptions. In: ECCV. pp. 597–613. Springer (2016)
14. Gao, J., Tang, C., Ganapathi-Subramanian, V., Huang, J., Su, H., Guibas, L.J.: Deepspline: Data-driven reconstruction of parametric curves and surfaces. arXiv preprint arXiv:1901.03781 (2019)
15. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: CVPR (2012)
16. Gerhard, S., Funke, J., Martel, J., Cardona, A., Fetter, R.: Segmented anisotropic ssTEM dataset of neural tissue (11 2013). https://doi.org/10.6084/m9.figshare.856713.v1, https://figshare.com/articles/Segmented_anisotropic_ssTEM_dataset_of_neural_tissue/856713
17. Grundmann, M., Kwatra, V., Han, M., Essa, I.: Efficient hierarchical graph-based video segmentation. In: CVPR (2010)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
19. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)

20. Huang, J., Gao, J., Ganapathi-Subramanian, V., Su, H., Liu, Y., Tang, C., Guibas, L.J.: Deepprimitive: Image decomposition by layered primitive detection. Computational Visual Media **4**(4), 385–397 (2018)
21. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. pp. 448–456 (2015)
22. Jacobson, A., Baran, I., Popović, J., Sorkine, O.: Bounded biharmonic weights for real-time deformation. SIGGRAPH **30**(4), 78:1–78:8 (2011)
23. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in neural information processing systems. pp. 2017–2025 (2015)
24. Jampani, V., Sun, D., Liu, M.Y., Yang, M.H., Kautz, J.: Superpixel samping networks. In: ECCV (2018)
25. Levinshtein, A., Stere, A., Kutulakos, K., Fleet, D., Dickinson, S., Siddiqi, K.: Turbopixels: Fast superpixels using geometric flows. PAMI **31**(12), 2290 – 2297 (2009)
26. Liang, J., Homayounfar, N., Ma, W.C., Xiong, Y., Hu, R., Urtasun, R.: Polytransform: Deep polygon transformer for instance segmentation. arXiv preprint arXiv:1912.02801 (2019)
27. Ling, H., Gao, J., Kar, A., Chen, W., Fidler, S.: Fast interactive object annotation with curve-gcn. In: CVPR. pp. 5257–5266 (2019)
28. Liu, M.Y., Tuzel, O., Ramalingam, S., Chellappa, R.: Entropy rate superpixel segmentation. In: CVPR 2011. pp. 2097–2104. IEEE (2011)
29. Maninis, K.K., Caelles, S., Pont-Tuset, J., Van Gool, L.: Deep extreme cut: From extreme points to object segmentation. In: CVPR (2018)
30. Moore, A., Prince, S., Warrell, J., Mohammed, U., Jones, G.: Superpixel lattices. In: CVPR (2008)
31. Recasens, A., Kellnhofer, P., Stent, S., Matusik, W., Torralba, A.: Learning to zoom: a saliency-based sampling layer for neural networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 51–66 (2018)
32. Shi, J., Malik, J.: Normalized cuts and image segmentation. Departmental Papers (CIS) p. 107 (2000)
33. Suinesiaputra, A., Cowan, B.R., Al-Agamy, A.O., Elattar, M.A., Ayache, N., Fahmy, A.S., Khalifa, A.M., Medrano-Gracia, P., Jolly, M.P., Kadish, A.H., et al.: A collaborative resource to build consensus for automated left ventricular segmentation of cardiac mr images. Medical image analysis **18**(1), 50–62 (2014)
34. Sun, X., Christoudias, C.M., Fua, P.: Free-shape polygonal object localization. In: European Conference on Computer Vision. pp. 317–332. Springer (2014)
35. Takikawa, T., Acuna, D., Jampani, V., Fidler, S.: Gated-scnn: Gated shape cnns for semantic segmentation. ArXiv **abs/1907.05740** (2019)
36. Tu, W.C., Liu, M.Y., Jampani, V., Sun, D., Chien, S.Y., Yang, M.H., Kautz, J.: Learning superpixels with segmentation-aware affinity loss. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 568–576 (2018)
37. Vincent, L., Soille, P.: Watersheds in digital spaces: An efficient algorithm based on immersion simulations. PAMI **13**(6), 583–598 (1991)
38. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.: Pixel2mesh: Generating 3d mesh models from single RGB images. In: ECCV. vol. 11215, pp. 55–71. Springer (2018)
39. Wang, Z., Acuna, D., Ling, H., Kar, A., Fidler, S.: Object instance annotation with deep extreme level set evolution. In: CVPR (2019)
40. Yamaguchi, K., McAllester, D., Urtasun, R.: Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In: ECCV (2014)

41. Yao, J., Boben, M., Fidler, S., Urtasun, R.: Real-time coarse-to-fine topologically preserving segmentation. In: CVPR (2015)
42. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017)
43. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: CVPR (2017)