Computational Visual Media
https://doi.org/10.1007/s41095-018-0128-6 Vol. 4, No. 4, December 2018, 385–397

Research Article

DeepPrimitive: Image decomposition by layered primitive detection

Jiahui Huang[1] (✉), **Jun Gao**[2], **Vignesh Ganapathi-Subramanian**[3], **Hao Su**[4], **Yin Liu**[5], **Chengcheng Tang**[3], **and Leonidas J. Guibas**[3]

Abstract The perception of the visual world through basic building blocks, such as cubes, spheres, and cones, gives human beings a parsimonious understanding of the visual world. Thus, efforts to find primitive-based geometric interpretations of visual data date back to 1970s studies of visual media. However, due to the difficulty of primitive fitting in the pre-deep learning age, this research approach faded from the main stage, and the vision community turned primarily to semantic image understanding. In this paper, we revisit the classical problem of building geometric interpretations of images, using supervised deep learning tools. We build a framework to detect primitives from images in a layered manner by modifying the YOLO network; an RNN with a novel loss function is then used to equip this network with the capability to predict primitives with a variable number of parameters. We compare our pipeline to traditional and other baseline learning methods, demonstrating that our layered detection model has higher accuracy and performs better reconstruction.

Keywords layered image decomposition; primitive detection; biologically inspired vision; deep learning

1 Tsinghua University, Beijing, 100084, China. E-mail: huang-jh18@mails.tsinghua.edu.cn (✉).

2 Computer Science Department, University of Toronto, Toronto, M5S2E4, Canada. E-mail: jungao@cs.toronto.edu.

3 Stanford University, Stanford, 94305, United States. E-mail: V. Ganapathi-Subramanian, vigansub@stanford.edu; C. Tang, chengcheng.tang@cs.stanford.edu; L. J. Guibas, guibas@cs.stanford.edu.

4 University of California San Diego, La Jolla, 92093, United States. E-mail: haosu@eng.ucsd.edu.

5 University of Wisconsin-Madison, Madison, 53715, United States. E-mail: yinl@cs.wisc.edu.

Manuscript received: 2018-11-30; accepted: 2018-12-03

1 Introduction

The computer vision community has been interested in performing detection tasks on images for a long time. The success of object detection techniques has been a shot-in-the-arm for better image understanding. The potent combination of deep learning techniques with traditional techniques [1, 2] has yielded state-of-the-art techniques which focus on detecting objects in an image through bounding box proposals. While this works well for tasks that require strong object localization, other applications in robotics and autonomic systems require a more detailed understanding of the objects in the image. Thus, another well-studied task in visual media processing is that of instance segmentation, where a per-pixel class label is assigned to an input image. Such dense labeling schemes are too redundant, and an intermediate representation needs to be developed.

Understanding images or shapes in terms of basic primitives is a very natural human abstraction. The parsimonious nature of primitive-based descriptions, especially when the task at hand does not require fine-grained knowledge of the image, makes them easy to use and a good choice. This has been explored extensively in the realms of both computer vision and graphics. Various traditional approaches exist for modeling images and objects, such as blocks world [3], generalized cylinders [4], and geons [5]. While primitive-based modeling generally uses classical techniques, using machine learning techniques to extract these primitives can help us to attack more complex images, with multiple layers of information in them. Basic primitive elements such as rectangles, circles, triangles, and spline curves are usually the building blocks of objects in images, and in combination, provide simple, yet extremely informative

representations of complex images. Labeling image pixels with high-level primitive information also aids in vectorizing rasterized images.

Complex images have multiple layers of information embedded in them. It is shown in Ref. [6], that human analysis of an image is always performed in a top–down manner. For example, when given an image of a room, the biggest objects such as desks, beds, chairs, etc., are observed. Then the focus shifts to specific objects, e.g., objects on the desk such as books and monitor; this analysis is performed recursively. When analyzing an image of a window, humans tend to focus on the border of the window first; the inner structure within the window and decorations are considered later. However, original object detection networks neglect this layered search and treat objects from different information layers the same. Layered detection has added value when there are internal occlusions in the image, which make traditional object detection more difficult to perform. In this work, we attempt to generate a deep network that separates multiple information layers as in Fig. 1, and is able to detect the positions of the primitives in each layer as well as estimating their parameters (e.g., the width, height, and orientation of a rectangle or the number and positions of control points of a spline). The proposed method is shown to be more accurate than traditional methods and other learning-based approaches.

This paper is organized as follows. We consider related work in Section 2, and provide an analysis of the novelty of our work. Then, in Section 3, we propose a framework based on the traditional YOLOv2 network [2], to provide parameters that are fully interpretable and high-level. We also tackle the problem of regressing parameters for primitives with a variable number of unknowns. Then, we propose a layered architecture in Section 4, which can learn to separate different information layers of the image and regress parameters in each layer separately. In Section 6, we give experiments used to evaluate the performance of our network against existing traditional state-of-the-art techniques, and in Section 7, we show how this framework could be applied to image editing and recognition by components. We also discuss the limitations of our framework. Finally, in Section 9, we attempt to envisage how the framework provided in this work would help to solve the important problem of primitive-based representations, which has applications that lie at the intersection of vision, AI, and robotics.

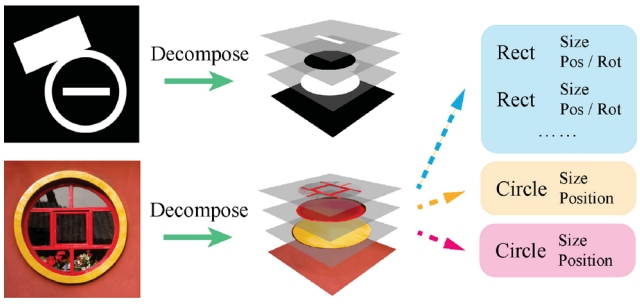


Fig. 1 Motivation: given an image composed of abstract shapes, our framework can decompose overlapping primitives into multiple layers and estimate their parameters.

To sum up, our contributions in this paper include:

- A framework based on the YOLOv2 network that enables class-wise parameter regression for different primitives.
- An RNN model to estimate a sequence of a variable number of control points representing a closed spline curve in a single 2D image.
- A layered primitive detection model to extract relationship information from an image.

2 Related work

Our task of decomposing an input image into layers of correlated and possibly overlapping geometric primitives is inherently linked to three categories of problems, which have been treated and studied independently in the traditional setting. Object detection and high-level vision, regression and reconstruction of geometric components such as splines and primitives, and finally, understanding relationships and layout of objects and entities are problems that provide information at different scales, all of great importance to the computer vision and graphics communities. After considering these three categories of applications, we conclude the discussion of related work with relevant machine learning methodologies, with a focus on recurrent neural networks.

2.1 Object detection and high-level vision

Among the traditional model-driven approaches to object detection, the generalized Hough transform [7] is a classical technique applicable to detecting particular classes of shapes up to rigid

transformations. Variability of shapes as well as input nuances are tackled by deep-learning based techniques; faster-RCNN [8] utilizes region proposal networks (RPN) to locate objects and fast-RCNN to determine the semantic class of each object. Recent works like YOLO [1, 2] and SSD [9] formulate the task of detection as a regression problem and propose end-to-end trainable solutions. We use the detection framework of the efficient YOLOv2 [2] as the backbone of our framework. However, unlike YOLO or YOLOv2, as well as providing bounding boxes and class labels, our framework also regresses geometric parameters and handles the problem of occlusion, in layered fashion.

To construct high-level objects using simple primitives, Biederman [5] introduced the idea of visual composition. Recently, SCAN [10] tries to compose *visual primitives* in a hierarchical way and learn an implicit hierarchy of concepts as well as their logical relations using a β-VAE network. While they build their hierarchy over concepts, our work is based on visual containment relationships for different shapes. Lake et al. [11] proposed a probabilistic program induction scheme to parse hand-writing images into several strokes and sub-strokes using a few images as training data, but their method is limited to the specific domain of hand-written characters.

2.2 Spline fitting and vectorization

Primitives and splines are widely used for representing geometry or images due to their succinctness and precision. Thus, recovering them by fitting input data is a long-standing problem in graphics. The idea of iteratively minimizing a distance metric [12–14], serving as a foundation of many studies, has been improved by either more effective distance metrics [15] or more efficient optimization techniques [16]. However, most previous works fail due to lack of decent initialization, which is overcome by a learning-based algorithm in our case. It is worth noting that vectorizing rasterized images [17, 18] also aims to solve a related problem. However, since previous works do not decompose an image into assemblies of clean primitives, there is a loss of high-level information about shape and layering.

2.3 Layered object detection

Multiple works have of late attempted to introduce composable layers into the process of object detection. Liu et al. [9] attempt to use feature hierarchies and detect objects based on different feature maps. Lin et al. [19] further improve this elegant idea by adding top–down convolutional layers and skip connections. However, these works only focus on how to combine features at different scales regardless of the relationships between objects and the associated layers composing the original image. The work by Bellver et al. [6] formulates detection as a reinforcement learning problem and represents an image as a predefined hierarchical tree, leaving the agent to iteratively select subsequent parts to look at. The work most relevant to ours is CSGNet [20], a recursive neural network model which generates a structured program defining the relationships between a sparse set of primitives. However, the possible positions and sizes of the primitives are limited to the size of a finite action space. In contrast, our work allows more detailed transformations of primitives, and our layered representation is less prone to redundancy.

2.4 Recurrent neural networks

The recurrent neural network (RNN) (and its variants LSTM [21], GRU [22]) is a common model widely used in natural language processing which has recently been applied to computer vision tasks. One key inspiration for our work is polygon-RNN [23], in which a sequence of vertices forming a polygon is predicted in a recurrent manner. One of the key differences in our work is that we aim to abstract the simplest types of representation on different layers, based on general splines instead of polylines, or interpolating cubic Bézier curves as in the polygon-RNN.

The discussion above only samples the studies most relevant to our work. There are many other relevant areas such as image parsing, dense captioning, structure-aware geometry processing, and more. Despite richness of relevant works across a wide range which manifest the importance of the topic, we believe that the problem of understanding images as abstract compositions is underexplored.

3 Basic model

In this section, we propose a framework based on a standard modification of the YOLOv2 model [2], inspired by Ref. [24], to perform parameter regression. The parameters regressed by the model, as opposed

to those in Ref. [24], are fully interpretable and high-level.

3.1 Adapting YOLO for parameter regression

The primary idea of this model is to extend the architecture of the state-of-the-art object detector YOLOv2 to detect primitives in an image, and in addition, to estimate the parameters of each primitive. The deep neural network architecture is capable of extracting more detailed descriptors of detected objects, as well as the bounding box location. Providing additional structural information about the object to the YOLOv2 architecture aids in augmenting the learned features.

The YOLOv2 network in the original paper consumes an entire image and segments it into a grid of size $S \times S$. Each square in the grid can contain multiple primitives. The networks model this multiplicity by containing up to B possible anchors (primitives in this case). Thus, traditional YOLOv2 networks learn $S \times S \times B \times (K+5)$ different parameters; the $K+5$ term arises since, in addition to the class labels for the K different primitive classes, the network also predicts 1 object probability value and 4 bounding-box related values [2]. While regressing parameters for the bounding boxes, the regressor needs to predict M extra variables for each bounding box being predicted. The M variables are the total number of possible parameters from all different primitive categories. This increases the number of parameters predicted by the network to $S \times S \times B \times (5+K+M)$.

To achieve this end, a new loss term is added to the loss function previously proposed in Ref. [24]. The new term, $\mathcal{L}_{\mathrm{p}}$, feeds information about the primitive parameters into the network. This term is defined as

$$\mathcal{L}_{\mathrm{p}} = \sum_{i=0}^{S}\sum_{j=0}^{S}\sum_{k=0}^{B} \mathbb{1}_{i,j}^{(k)} \sum_{l=0}^{K} \mathbb{1}_{(i,j),k}^{(l)} \sum_{m \in X(l)} \mathcal{L}(t_{(i,j),k}^{(m)}, \hat{t}_{(i,j),k}^{(m)}) \tag{1}$$

where $\mathbb{1}_{i,j}^{(k)}$ is an indicator function that determines if grid square (i,j) is assigned a positive object label for bounding box k. The indicator $\mathbb{1}_{(i,j),k}^{(l)}$ is a function that determines if bounding box k of grid square (i,j) belongs to the primitive defined by l. The purpose of introducing this term is to include a weighing for a primitive in the loss only when the primitive is plausible for the image. $X(l)$ is the set of parameters for primitive l. The terms t and $\hat{t}$ denote the target and predicted parameters respectively.

3.2 Definition of primitive parameters

Primitives with fixed number of parameters. Simple primitives like rectangles or circles have fixed numbers of parameters, and so the values of these parameters can be used directly as ground truth for training. For parameters lying within $[0,1]$, we can further increase the network training stability by applying a sigmoid function to the network output to constrain the estimated parameters. Readers are referred to Section S1 in the Electronic Supplementary Material (ESM) for detailed definitions of primitive parameters.

Primitives with variable number of parameters. Some of the primitives discussed in this paper, including closed B-spline curves, have a variable number of control points. This permits primitives to represent different kinds of shapes, but it is not compatible with the previously defined model. This incompatibility is solved by learning a fixed-length embedding of the control point positions. In addition, a recurrent neural network (RNN) is appended to the model, to serve as a decoder to output the control points in a sequential manner. At time step i, the model predicts the position of the ith control point c_i, and a stop probability $p_i \in [0,1]$, that indicates the end of the curve. We apply cross-entropy life loss to the stop probability while training the RNN.

The loss functions for the RNN-based model must be designed with care. Naively, one can use a simple mean-squared error (MSE) loss for control point position prediction and a cross entropy loss for probability prediction. However, this only handles the situation where the sequence of control points is fixed and well-defined. Note that every point in the control point sequence $C = (c_1, \ldots, c_N)$ of a closed spline curve can be viewed as the starting point of the sequence. Thus, in order to predict a control point sequence invariant to the position of starting point, a circular loss similar to that used in Ref. [23] is defined as follows:

$$\mathcal{L}_{\mathrm{circ}} = \min_{k \in [1,N]} (\min(\mathcal{L}(C, G_k), \mathcal{L}(C, G_k'))) \tag{2}$$

where $\mathcal{L}$ is the MSE loss, G_k is the ground truth control point sequence rotated by k places, i.e., if g_i denotes the ith control point in the ground truth, then G_k is the sequence $(g_k, \cdots, g_N, g_1, \cdots, g_{k-1})$ and G_k'

is the inverse sequence of G_k. In this way, the ground truth sequence that leads to minimum MSE loss is considered to be the target sequence, making the loss function rotation-invariant. Also note that the introduction of G'_k guarantees the loss to be invariant to clockwise and anti-clockwise sequencing.

4 Layered detection model

4.1 Layered detection

We use a layered model to capture the nested structure of primitives in an image. The idea is inspired by two observations. Our first observation is from how multiple layers in design tools, such as Adobe Photoshop and Illustrator, can help create a vector graphics image. With layers, artists can plan the arrangement of items in the space in a top–down manner. This fact that all vector icon images can be decomposed into multiple layers, as shown in Fig. 1, serves as inspiration to extend the model proposed in Section 3 to include layered detection. Secondly, for the detection of each layer, it allows one to focus on a specific part of the image, instead of working on the entire image. For example in Fig. 1, the white rectangle in the lower-right of the image is completely inside the black disk: one can focus in the interior of the disk where the only accessible primitive is the rectangle.

However, training separate networks for different levels of detection is a redundant and time-consuming process, since intuitively, the parameters regressed by these networks are likely to be related. Therefore, we propose a layered detection model to perform this regression task, thereby making the training process both faster and cognizant of previous learning. We perform region of interest (RoI) pooling [25] on the intermediate output of our network. This enables us to extract regions in the image to focus on, to perform detection at the next level.

4.2 Architecture

After an image is forwarded through the backbone network, simple post-processing steps including thresholding and non-maximal suppression are performed to obtain the final prediction results. The backbone network is the previously discussed YOLO network with modified loss; the difference lies in that the backbone network is intended to only predict primitives in the top layer, i.e., the outermost primitives in the image. Following this, the coordinates of the bounding boxes of detected primitives are fed into an RoI pooling layer. The RoI pooling layers consume the intermediate output of the network and pool it into a uniform sized feature map for detection following the layering. Figure 2 illustrates this model.

Specifically, the architecture of the backbone network can be treated as multiple consecutive modules, which contain several convolution layers with ReLU activation; each module is combined with pooling layers. We denote the modules by $f_1, \cdots, f_M$ (from shallow layers to deep layers). The deepest layer f_M has output J_1 that is processed by the detection block d_1. Subsequent detection blocks d_i process the output of convolutional layer f_{M-i+1}. We do not use the whole feature map J_i as the input to d_i, but instead, we crop the feature map using the prediction results from d_{i-1} and resize it to a uniform size. In

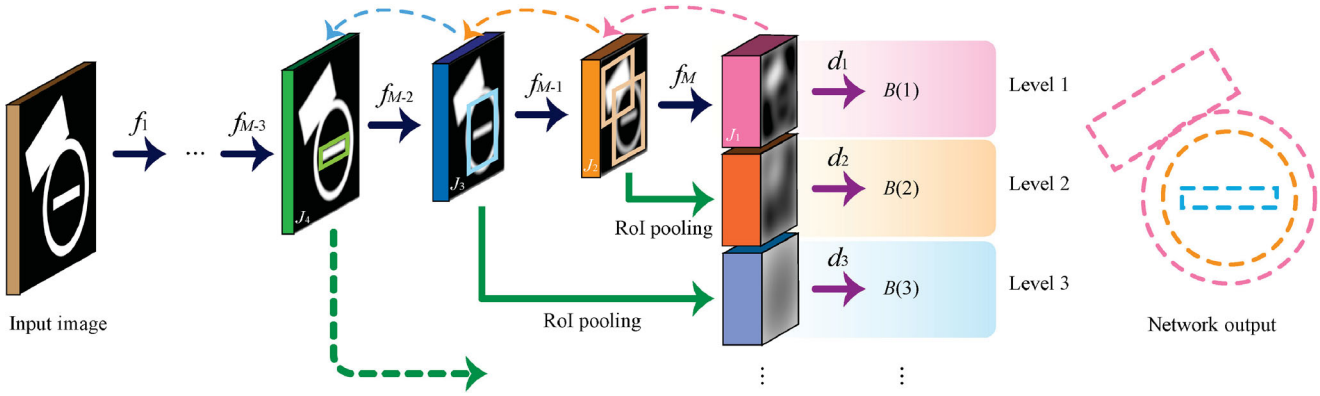


Fig. 2 The detection process in our layered model. Cuboids denote input images or feature maps. Dark blue arrows, dark green arrows, and dark purple arrows represent conv layers, RoI pooling layers, and detection blocks, respectively; notation is consistent with that in the text. The final output of our network is a layered primitive tree containing both shape information and layer information.

this way, the layering is represented explicitly by cropping within the interior of an image. This model can be expressed as

$$B(1) = d_1(J_1) \tag{3}$$

$$B(i) = d_i(R[J_i; B(i-1)]), \quad i \geqslant 2 \tag{4}$$

where $R[J; B(i)]$ represents feature map J cropped using bounding box information from $B(i)$ which is fed to an RoI pooling layer to obtain a uniform size output for future processing.

Lower level feature maps are employed for deeper layer detection since deeper layer primitives are usually smaller in size and thus clearer feature maps are required to perform accurate detection. For consistency within different regions in image, we perform training using local coordinates within the parent bounding box as the ground truth for $B(i)$. For example, consider an image with a rectangle inside a circle. Then, the ground truth coordinates for the rectangle should lie within the local coordinate system with respect to the circle. Therefore, predicted coordinates are transformed before calculating the loss functions. These local coordinates are used for ground truth since RoI pooling is known to capture partial information in the image, as testified by faster-RCNN [8]. Meanwhile, since there are multiple layers of convolutional operations, the feature map can encode some information outside the bounding box, thus providing the model with the capability to correct mistakes made in outer layers, by considering both local and global information while making detections in inner layers.

It is worth noting that the information passed from higher to lower layers is not simply restricted to the explicit bounding box position. The feature map in shallower convolutional layers is used to predict both higher and lower level primitives (e.g., in Fig. 2, J_2 affects both $B(1)$ and $B(2)$). Although we only pass the bounding box information explicitly, knowledge from higher layers can be passed *implicitly* via these related feature maps.

5 Implementation

In this section, we present our implementation details.

5.1 Primitive and parameter selection

Four types of primitives are used in our experiments: rectangles, triangles, ellipses, and closed spline curves. We observed that the predicted bounding box position is usually more accurate than the regressed parameters. Hence, a local parameter with respect to the bounding box is defined for each primitive so as to be able to perform better reconstruction. Readers are referred to Section S1 in the ESM for detailed descriptions of the parameters used.

5.2 Network architecture

Our code is adapted from an open source PyTorch implementation [①]. The backbone network uses the Darknet-19 architecture configured as in Redmon and Farhadi [2]. We set the depth of our layered detection model to 3, using three detection blocks. Detailed configuration of detection block d_i $(i = 1, 2, 3)$ is provided in Section S2 of the ESM.

5.3 Training

The entire hierarchical model can be trained fully end-to-end. Additionally, we adopt a method similar to scheduled sampling [26] to enhance training stability and testing performance. The predicted information $B(i-1)$ from level $i-1$, which is fed into level i, is substituted by the ground truth value for level $i-1$ with probability p. The value of p is set to 0.9 in the first 10 epochs and is subsequently decreased by 0.05 every 2 epochs.

An RNN decoder model is pre-trained separately to regress a fixed length embedding for control point positions. While training this RNN model, the grid number S is set to 1 in the YOLOv2 detection framework and the features of closed spline curve images are extracted with our backbone Darknet-19 network. The pre-trained RNN decoder learns to decode the fixed length embedding and output positions of control points sequentially. When the layered model is being trained, the value of the embedding is used as direct supervision. In the first 5 epochs, the embedding is supervised and in subsequent epochs, the network is trained with the positions of control points instead. Note that the RNNs share the same weights across different levels of the hierarchy.

5.4 Data synthesis

Following previous works [10, 27], we use synthetic datasets due to the lack of annotated datasets. The hierarchical model was trained with 150,000 synthetic

[①] https://github.com/longcw/yolo2-pytorch

pictures of size 416×416. When we generated the training data, we kept the containment relationships across layers; there may be multiple primitives in each layer. The number of primitives in a single image is restricted to 8, the maximum number of layers to 3, and the number of control points of closed spline curves varies from 5 to 7. In order to test the robustness of our method, noise was added to the shapes of the primitives, as well as hatching patterns for primitives and some skewing of the image itself. Selected dataset images are shown in Fig. 3.

6 Experiments and results

6.1 Ablation study for circular loss

During the pretraining process for the RNN decoder to predict control point positions, we compare the training and validation losses using two different loss functions, i.e., the previously defined $\mathcal{L}_{\text{circ}}$ and a simple MSE loss. As shown in Table 1, training with circular loss leads to better convergence loss and thus better prediction results. Figure 4 shows two examples comparing the prediction results given the same curve image as input. We found that using circular loss eliminates the ambiguity of starting point and clock direction in the training data, and leads to more accurate fitting results.

Table 1 Error and accuracy measures during training and testing with two different loss functions. Loss denotes the MSE distance between the ground truth and predicted positions of control points (distances are normalized to lie in the unit interval). # Point Acc. denotes the frequency of predicting the number of control points correctly

	Training		Validation	
	Loss	# Point Acc.	Loss	# Point Acc.
$\mathcal{L}_{\text{MSE}}$	0.12203	74.60	0.12210	74.93
$\mathcal{L}_{\text{circ}}$	0.04365	76.32	0.04369	75.83

6.2 Comparisons to other methods

Although our model detects primitives in a layered manner, simple object detection measurements including precision and recall rate (or mAP for methods with confidence score output) can be applied to test model accuracy. Meanwhile, we define our reconstruction loss as the pixel-wise RMSE between the input picture and the re-rendered picture using the predicted results from the network. There are multiple approaches to shape detection; we set up 5 independent baselines for comparison. The first two baselines are traditional methods while the last three are learning-based approaches:

- *Contour method.* In this method, edge detection is first applied to the input image; each independent contour is separated. A post-processing approximation step is then employed to replace almost collinear segments with a single line segment with a parameter q controlling the strength of approximation. The type of shape is determined by counting the number of line segments (i.e., its number of edges). This method is implemented using `findContours` and `approxPolyDP` functions of OpenCV [28].
- *Hough transform* [29]. This is widely used to find imperfect shape instances in images by a voting procedure in parameter space. For rectangles and triangles, whose edges are straight line segments, we first use Hough line transform to detect all possible lines and then recover the parameters of the primitives by solving a set of linear equations. For ellipses, we use the method described in Ref. [30].
- *CSGNet* [20]. In 2D, this takes a single image as input and generates a program defining the shapes presented. This model allows for more

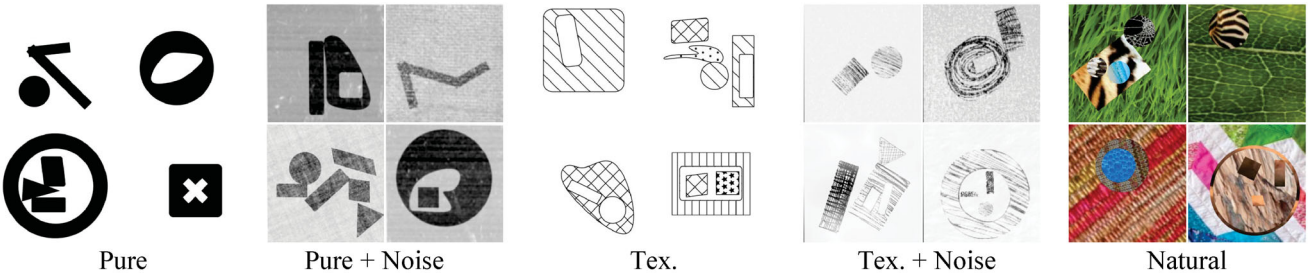


Fig. 3 Examples drawn from our synthetic training dataset. For the *Pure* dataset, we synthesized simple binary images for training. The *Pure+Noise* dataset modified the *Pure* dataset by adding noise and random affine transformations to each image. The *Tex.* (short for "Textured") dataset allows testing of the robustness of shape detection methods by adding hatching patterns to the shapes. The *Textured+Noise* dataset imitates real world hand drawn shape pictures. The *Natural* dataset imitates colored versions of real world images.

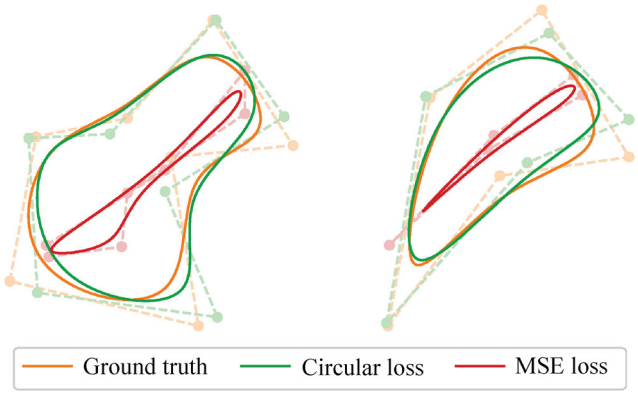


Fig. 4 Two closed spline curve fitting cases using circular loss and MSE loss.

complex Boolean operations between shapes but the sizes and positions of the primitives are highly discretized. We use the post-processed (optimized) top-1 prediction as the output of this algorithm.

- *Flat model*. This method uses a learning approach trained using the YOLOv2 architecture. The ground truth of the detector is directly set to all primitives in the canvas, regardless of their hierarchical information.
- *Recursive model*. We train only one detector to detect the primitive in the first hierarchy (i.e., the outermost primitive at the current level). Once the detector successfully detects some primitives in the current level, we crop the detected region, resize the cropped region to the network input size, and feed the image into the same network again.

Results from these different models are compared in Table 2 (precision–recall–reconstruction comparison) and Table 3 (primitive–reconstruction comparison). Some of the prediction results from different methods are shown in Fig. 5 using the same input in each case.

The contour method with small q value traces the pixels on the contour precisely but ignores the high-level shape information of the shape boundary, leading to a high reconstruction performance but low precision and recall accuracy in shape classification tasks. Using a greater q value simply approximates continuous curves with polygons, leading to poor reconstruction performance. It is also observed that the contour method cannot separate overlapping primitives since it only attempts to detect boundaries in images. The Hough transform-based method for line segment detection and circle detection requires a careful choice of parameters; it generally leads to higher recall values than the contour method. This method partially solves the overlap problem by extending detected line segments and finding intersections, but cannot effectively distinguish extremely short line segments and segments of a circle.

The above problems can be overcome by learning-based models. Learning-based models generally have better performance across all different datasets and the gap in performance widens as we add more noise to our dataset, which is partially due to the fact that the learned features extracted from the image using our data-driven method are more effective and representative in comparison to hand-crafted features of traditional methods. Despite the feature improvement, the absence of effective shape and relationship representations can be fatal to the final detection results. Using CSGNet [20], the possible locations and sizes of primitives are restricted due to the size limitation of the action space. In order to compose the target shape, redundant shapes and expressions are generated.

Table 2 Precision, recall, and reconstruction loss measures using various methods as described in Fig. 3. Prec and Recall denote the precision and recall values as percentages respectively while Recon measures the RMSE loss between the original picture and the reconstructed picture using the layered prediction results

Method	Pure			Pure+Noise		Textured		Textured+Noise		Natural	
	Prec	Recall	Recon	Prec	Recall	Prec	Recall	Prec	Recall	Prec	Recall
Contour ($q = 4 \times 10^{-4}$)	78.8	42.9	1.44	10.1	37.7	10.8	54.6	10.0	47.5	5.9	62.2
Contour ($q = 2 \times 10^{-3}$)	94.0	72.8	1.70	32.5	60.1	16.8	88.0	15.6	73.2	6.4	70.3
Hough transform	32.6	78.6	1.61	5.1	73.7	—	—	—	—	—	—
CSGNet (optimized) [20]	37.1	65.4	28.7	—	—	—	—	—	—	—	—
Flat model	**99.7**	91.0	—	99.5	90.0	**99.6**	91.2	99.4	91.0	57.9	62.2
Recursive model	96.1	72.4	1.64	60.1	61.2	74.0	60.1	95.8	49.9	**98.9**	84.5
Our model	**99.7**	**96.1**	1.61	99.5	**95.0**	**99.6**	**95.8**	**99.5**	**95.4**	97.9	**87.6**
Our model (optimized*)	**99.7**	**96.1**	**1.39**	**99.6**	**95.0**	—	—	—	—	—	—

* It is impossible to measure reconstruction loss for images with texture or noise, making it unclear how to define the optimization target.

Table 3 Average precision (AP) measures of learning-based shape detection methods. Values are presented in percentage

	Mean	Parallelogram	Triangle	Oval	Spline
Flat	87.2	87.2	86.3	84.4	90.9
Recursive	54.3	43.8	53.8	76.0	43.6
Ours	**90.5**	**88.2**	**90.7**	**90.9**	**92.0**

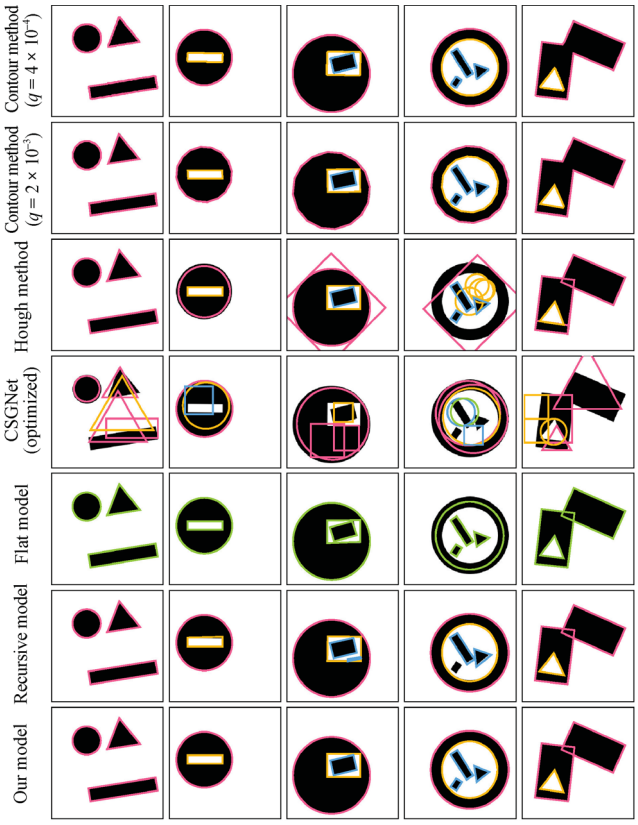


Fig. 5 Detection results examples. Shapes detected at different levels are marked in different colors: level 1, pink; level 2, orange; level 3, blue. For the flat model, there is no predicted layer information, so all shapes are marked in green.

Other learning-based baselines fix this with simple containment representations but problems still occur due to lack of layering or incorrect layering. The flat model detects almost all primitives regardless of their layer. However, in cases where two primitives of the same kind (e.g., concentric circles forming an annulus) overlap, the post-processing step (non-maxima suppression) eliminates one of them and predicts the median result, which is undesirable. It is also difficult to reconstruct the original image using the detected primitives due to the loss of layering information. In the recursive model, the layering information is preserved, but if the detection in an outer layer is not accurate enough, the error snowballs and the inner layer primitives cannot be well-reconstructed. Unlike the baselines, our method can extract high-level shape information as well as containment relationships. Our model outperforms the others both quantitatively and qualitatively, except for the reconstruction loss. However, after appending a simple local optimizer to our model, denoted *Our model (optimized)* in Table 2, the reconstruction loss is further decreased.

The trained model was applied directly to Google Material icons [31] (lines 1–4 of Fig. 6, using *Pure* model) and a small real world dataset containing 150 images selected from the PASCAL VOC2012 dataset [32] and the Internet (lines 5–8 of Fig. 6, using *Natural* model). To the best of our knowledge, no public dataset exists that provides ground truth annotations at geometric primitive level. So we have manually annotated the 150 images from this small real world dataset. Testing using our trained model reached an mAP (the metric used in all experiments) of 54.5%. Readers are referred to Sections S3 and S4 in the ESM for further results.

While DeepPrimitive manages to decompose the real world images into relevant primitives, it is to be remembered that this is not the primary focus of our

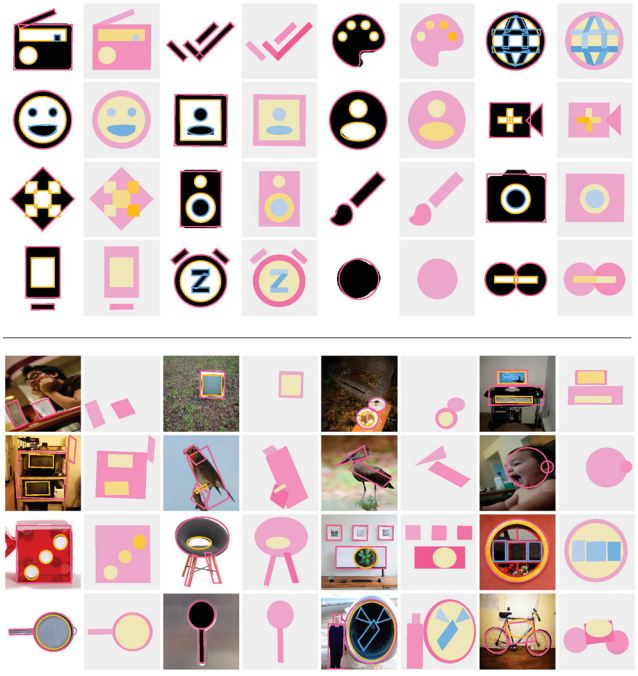


Fig. 6 Selected test results for our layered detection model. In each pair of columns, the left picture shows the original input image as well as the detection result while the right picture reconstructs the input image using the detection result (different instances of primitives within the same hierarchy vary slightly in color for clarity). More test results are available in Sections S3 and S4 in the ESM.

work. Our current model is trained only on synthetic images, but adapting synthetic images to real images with domain adaptation techniques is one trend in the vision community. A few recent vision papers have been trained and tested on purely synthetic datasets (e.g., Ref. [27]).

7 Applications

Once an image has been decomposed into several layers and high-level parameters defining the primitives in the image acquired, one can utilize this information for a variety of applications. In this paper, we demonstrate the use of these parameters in two example applications.

The first application we present is image editing. It is usually very difficult for an artist to modify the shapes in a rasterized image directly. With a low reconstruction loss, our model can decompose an image into several manipulable components with high fidelity and flexibility. For example, in Fig. 7, it is easy for an icon designer to modify parameters of the shapes, changing the angle between the hands of the clock, or tweaking the shape of the paint brush head. For real world images in Fig. 8, we can directly manage the position of the parts in an image using high-level editing tools (e.g., as in Ref. [33]).

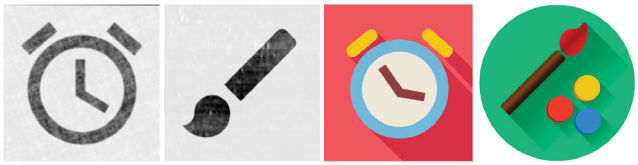


Fig. 7 Image editing on a rasterized image at a primitive level. Primitive detection is performed on the image, followed by editing of the primitives.

Another potential application is recognition-by-components [5]. Usually, state-of-the-art classifiers based on deep networks need very much data for training, and its lack hampers accuracy. Once primitives in an image have been recognized, one can easily define classification rules using the layered information obtained. Additional training data is not needed and only a single shape detection model has to be trained. The idea is illustrated in Fig. 9. Given an image, pre-processing steps such as denoising and thresholding are performed to extract the borders of shapes. The proposed model is then applied to detect the primitives and generate a shape parsing tree (in XML format in the figure for demonstration purposes), with which a handcrafted classifier could easily predict the class of an object in the image by top–down traversal of the tree.

8 Limitations

As an explorative study aiming to understand and reconstruct images as primitives composed layer-wise, there are several limitations left to be resolved in future work. For images with highly-overlapping primitives within the same layer, our model cannot distinguish between them: the output will either be a single primitive or misclassified primitives. Our model discovers only containment relationships: if one higher-level primitive intersects multiple lower-level primitives, duplicate detections of the higher-level primitive are possible. The last two images of line 4 in Fig. 6 demonstrate such failures. These limitations restrict the layer decomposability of our model. Meanwhile, only synthetic images are used for training. Annotated real world data would make the model more generalizable.

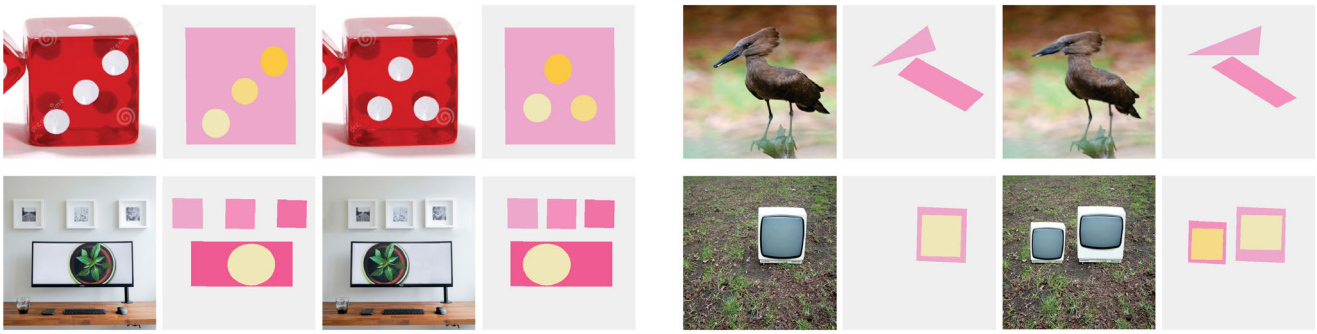


Fig. 8 High-level image editing of real world images based on detected primitives. The first two columns of each group show the original image and its layered decomposition while the last two columns of each group show manipulated results.

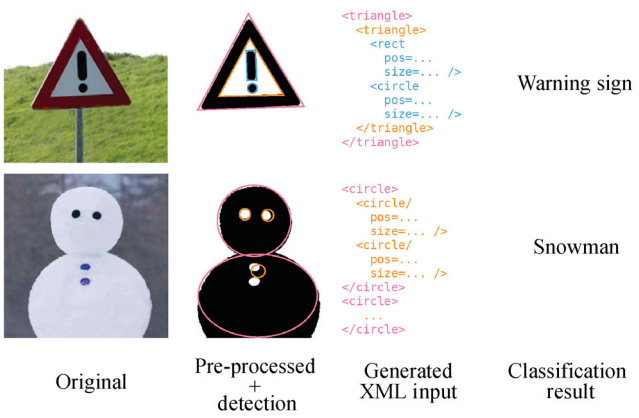


Fig. 9 Recognition-by-components demonstration using our proposed hierarchical primitive detection model.

9 Conclusions

This paper demonstrates a data-driven approach to layered detection of primitives in images, and subsequent 2D reconstruction. As noted, abstraction of objects into primitives is a very natural way for humans to understand objects. As artificial intelligence moves towards performing tasks in human-like fashion, there is value in trying to perform these tasks in the way a human would.

Such tasks often also fall in the intersection of robotics and computer vision, e.g., in the cases of autonomous driving and robotics. In such tasks, building in environment-awareness into cars or robots based on their field of vision is key, and primitive-level reconstruction would be useful. Primitive-level understanding would also help in understanding physical interactions with objects in manipulation tasks. While there are many such avenues where this understanding could be applied, there is a lack of open datasets for training on real world data. A good direction for future study would involve learning tasks of an unsupervised or self-supervised kind.

Acknowledgements

Chengcheng Tang would like to acknowledge NSF grant IIS-1528025, a Google Focused Research award, a gift from the Adobe Corporation, and a gift from the NVIDIA Corporation.

Electronic Supplementary Material Supplementary material with detailed experimental configuration and results is available in the online version of this article at `https://doi.org/10.1007/s41059-018-0128-6`.

References

[1] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 779–788, 2016.

[2] Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 6517–6525, 2017.

[3] Roberts, L. G. Machine perception of three-dimensional solids. Ph.D. Thesis. Massachusetts Institute of Technology, 1963.

[4] Binford, T. O. Visual perception by computer. In: Proceedings of the IEEE Conference on Systems and Control, 1971.

[5] Biederman, I. Recognition-by-components: A theory of human image understanding. *Psychological Review* Vol. 94, No. 2, 115–147, 1987.

[6] Bellver, M.; Giro-i-Nieto, X.; Marques, F.; Torres, J. Hierarchical object detection with deep reinforcement learning. In: Proceedings of the Deep Reinforcement Learning Workshop, NIPS, 2016.

[7] Ballard, D. H. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition* Vol. 13, No. 2, 111–122, 1981.

[8] Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 39, No. 6, 1137–1149, 2017.

[9] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A. C. SSD: Single shot multibox detector. In: *Computer Vision – ECCV 2016. Lecture Notes in Computer Science, Vol. 9905*. Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 21–37, 2016.

[10] Higgins, I.; Sonnerat, N.; Matthey, L.; Pal, A.; Burgess, C.; Botvinick, M.; Hassabis, D.; Lerchner, A. SCAN: Learning abstract hierarchical compositional visual concepts. *arXiv preprint* arXiv:1707.03389, 2017.

[11] Lake, B. M.; Salakhutdinov, R.; Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science* Vol. 350, No. 6266, 1332–1338, 2015.

[12] Rogers, D. F.; Fog, N. Constrained B-spline curve and surface fitting. *Computer-Aided Design* Vol. 21, No. 10, 641–648, 1989.

[13] Besl, P. J.; McKay, N. D. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 14, No. 2, 239–256, 1992.

[14] Chen, Y.; Medioni, G. Object modeling by registration of multiple range images. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2724–2729, 1991.

[15] Wang, W.; Pottmann, H.; Liu, Y. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics* Vol. 25, No. 2, 214–238, 2006.

[16] Zheng, W.; Bo, P.; Liu, Y.; Wang, W. Fast B-spline curve fitting by L-BFGS. *Computer Aided Geometric Design* Vol. 29, No. 7, 448–462, 2012.

[17] Sun, J.; Liang, L.; Wen, F.; Shum, H.-Y. Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics* Vol. 26, No. 3, Article No. 11, 2007.

[18] Lecot, G.; Levy, B. Ardeco: Automatic region detection and conversion. In: Proceedings of the 17th Eurographics Symposium on Rendering Techniques, 349–360, 2006.

[19] Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2117–2125, 2017.

[20] Sharma, G.; Goyal, R.; Liu, D.; Kalogerakis, E.; Maji, S. CSGNet: Neural shape parser for constructive solid geometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5515–5523, 2018.

[21] Gers, F. A.; Schraudolph, N. N.; Schmidhuber, J. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research* Vol. 3, No. 1, 115–143, 2002.

[22] Cho, K.; Merriënboer, B. V.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. *arXiv preprint* arXiv:1406.1078, 2014.

[23] Castrejón, L.; Kundu, K.; Urtasun, R.; Fidler, S. Annotating object instances with a polygon-RNN. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5230–5238, 2017.

[24] Jetley, S.; Sapienza, M.; Golodetz, S.; Torr, P. H. S. Straight to shapes: Real-time detection of encoded shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4207–4216, 2017.

[25] Girshick, R. Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 1440–1448, 2015.

[26] Bengio, S.; Vinyals, O.; Jaitly, N.; Shazeer, N. Scheduled sampling for sequence prediction with recurrent neural networks. In: *Advances in Neural Information Processing Systems 28*. Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; Garnett, R. Eds. Curran Associates, Inc., 1171–1179, 2015.

[27] Wu, J.; Tenenbaum, J. B.; Kohli, P. Neural scene de-rendering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[28] Itseez. Open source computer vision library. 2015. Available at https://github.com/itseez/opencv.

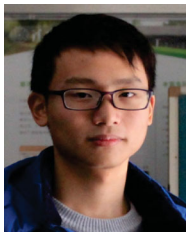
[29] Duda, R. O.; Hart, P. E. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM* Vol. 15, No. 1, 11–15, 1972.

[30] Xie, Y.; Ji, Q. A new efficient ellipse detection method. In: Proceedings of the IEEE International Conference on Pattern Recognition, Vol. 2, 957–960, 2002.

[31] Google. Google material icon. 2017. Available at https://material.io/icons/.

[32] Everingham, M. The PASCAL Visual Object Classes Challenge 2012 (VOC2012). Available at http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[33] Barnes, C.; Shechtman, E.; Finkelstein, A.; Goldman, D. B. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics* Vol. 28, No. 3, Article No. 24, 2009.



Jiahui Huang received his B.S. degree in computer science and technology from Tsinghua University in 2018. He is currently a Ph.D. candidate in computer science in Tsinghua University. His research interests include computer vision and computer graphics.

Jun Gao received his B.S. degree in computer science from Peking University in 2018. He is a graduate student in the Machine Learning Group at the University of Toronto and also affiliates to the Vector Institute. His research interests are in deep learning and computer vision.

Vignesh G. Subramanian is a Ph.D. candidate in the Department of Electrical Engineering, Stanford University. He previously obtained his dual degrees (B.Tech. in EE and M.Tech. in communication engineering) from IIT Madras, India. His research interests include shape correspondences, 3D geometry, graphics, and vision.

Hao Su received his Ph.D. degree from Stanford University, under the supervision from Leonidas Guibas. He joined UC San Diego in 2017 and is currently an assistant professor of computer science and engineering. His research interests include computer vision, computer graphics, machine learning, robotics, and optimization. More details of his research can be found at http://ai.ucsd.edu/haosu.

Yin Liu received his B.S. degree from Department of Automation of Tsinghua University in 2018. He is currently a Ph.D. candidate in computer science at the University of Wisconsin-Madison. His research interest is in machine learning.

Chengcheng Tang received his Ph.D. and M.S. degrees from King Abdullah University of Science and Technology (KAUST) in 2015 and 2011, respectively, and his bachelor degree from Jilin University in 2009. He is currently a postdoctoral scholar in the Computer Science Department at Stanford University. His research interests include computer graphics, geometric computing, computational design, and machine learning.

Leonidas J. Guibas received his Ph.D. degree from Stanford University in 1976, under the supervision of Donald Knuth. His main subsequent employers were Xerox PARC, MIT, and DEC/SRC. Since 1984, he has been at Stanford University, where he is a professor of computer science. His research interests include computational geometry, geometric modeling, computer graphics, computer vision, sensor networks, robotics, and discrete algorithms. He is a senior member of the IEEE and the IEEE Computer Society. More details about his research can be found at http://geometry.stanford.edu/member/guibas/.