DeepPrimitive: Image Decomposition by Layered Primitive Detection

Anonymous CVPR submission

Paper ID 934

Abstract

Human perception of the world is through basic building blocks, such as circle or rectangle primitives. While traditional object detection tasks focus on assigning semantic labels and bounding boxes to each image, decomposing an image into simple primitives to get layering information can be used for image reconstruction. In this work, a standard modification of the YOLO network is proposed to detect primitives and regress their parameters; an RNN with a novel loss function is then presented to equip this network with the capability to predict primitives with a variable number of parameters. Using this modification, we build a framework to detect primitives in a layered manner. We also compare to traditional and other naive learning methods, to demonstrate that our layered detection model has higher accuracy and performs better reconstruction.

1. Introduction

The computer vision community has been interested in trying to perform detection tasks on images for a long time. The success of object detection techniques has been a shot-in-the-arm to obtain better image understanding. The potent combination of deep learning techniques with traditional techniques [18, 19] has managed to provide state-of-the-art techniques which focus on detection objects in an image through bounding box proposals. While this is great for tasks that require strong object localization, certain other applications in robotics and autonomous would require a more detailed understanding of the objects in the image. On the contrary, another well-studied task in computer vision is that of instance segmentation, where a per-pixel class label is assigned to an input image. This dense labeling scheme carries too much redundancy, and an intermediate representation would need to be developed.

Understanding images or shapes by means of basic primitives is a very natural human abstraction. The parsimonious nature of primitive-based descriptions, especially when the task at hand does not involve requirement of fine-grained knowledge of the image, makes it easy and favorable to use. This has been explored extensively in the realms of both computer vision and graphics. There have been different traditional approaches for modeling images and objects, such as blocks world [22], generalized cylinders [26] and geons [6]. While primitive-based modeling is generally done using classical techniques, using machine learning techniques to be able to extract these primitives would help us 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. Modeling image pixels with high-level primitive information also aids in vectorizing rasterized images.

Complex images also have multiple layers of information embedded in them. It is shown in [3], that human analysis of an image is always performed in a top-down manner. For example, when given an image of a room, the bigest objects such like desks, beds, chairs, etc. are observed. Then the focus shifts on to specific objects, such as objects on the desk, like books, monitor, etc.; and this analysis is performed recursively. 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, in which case, traditional object detection is more difficult to perform. In this work, we attempt to generate a deep network that separates multiple information layers as in Figure 1, and is able to detect the position of the primitives in each layer as well as getting their parameters (e.g., the width, height, rotation angle of a rectangle or the number and position of control points of a spline). The proposed method is shown to be more accurate compared to traditional methods and other learning-based approaches.

This paper is organized as follows: We provide a sampling of related work to this paper in Section 2, along with an analysis of the novelty in our work in comparison to these different works. Then, in Section 3, we propose a framework based on the traditional YOLOv2 network [19] to be able to regress parameters that are fully interpretable

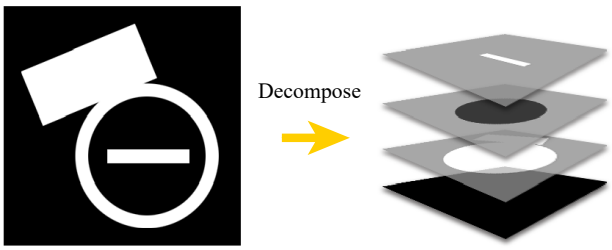

Figure 1. **Motivation** Given a binary image composed of abstract shapes, our framework could decompose overlapping primitives into multiple layers and predict their parameters.

and high-level. Here, 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 every layer separately. In Section 5, we show experiments to evaluate the performance of our network against existing traditional state-of-the-art techniques, and in Section 6, we show how this framework could be applied in the realms of image editing and recognition by components. Here, we also discuss the limitations of our framework. Finally, in Section 7, 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 in the intersection of vision, AI, and robotics.

To sum up, our contributions in this paper include:

- A framework based on the YOLO network that enables class-wise parameter regression for different primitives.
- An RNN model to predict the sequence of a variable number of control points of a closed spline curve from 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 classification, regressing and reconstructing geometric components such as splines and primitives and finally, understanding relationship and layout of objects or entities are problems that provide information of different resolutions, all very impactful to the computer vision and graphics communities. Beyond those three categories of applications, we conclude the discussion of related work with relevant machine learning methodologies, with a focus on recurrent neural networks.

Object Detection and Classification Among the traditional model-driven approaches to perform object detection and classification, the generalized Hough transform [2] is a classical technique applicable for 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 [21] utilizes Region Proposal Network (RPN) to propose the position of objects and Fast-RCNN to determine the semantic class for each object. Recent works like YOLO [18, 19] and SSD [17] 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 [19] as the backbone of our framework. However, unlike YOLO or YOLOv2, besides predicting bounding boxes and class labels, our framework also regresses geometric parameters and handles the problem of occlusion, in layered fashion.

Spline Fitting and Vectorization Primitives and splines are widely used for representing geometry or images due to their succinctness and precision. Thus, recovering them through fitting different input is a long-standing problem in graphics. The idea of iteratively minimizing a distance metric [23, 5, 8], serving as a foundation of many studies, have been improved by either more effective distance metrics [27] or more efficient optimization techniques [30]. However, most of the previous works fail due to the lack of a decent initialization, which, is handled by a learning-based algorithm in our case. It is worth noting that vectorizing rasterized images [25, 15] 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.

Layered Object Detection Multiple works have, of late, attempted to introduce composable layers into the process of object detection. Liu et al. [17] attempt to use feature hierarchies and detect objects based on different feature maps. Lin et al. [16] 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 relationship among objects and the associated layers composing the original image. Approaching from a different perspective, Zhu et al. [31] represent an object with a hierarchical tree model. Redmon et al. [19] consider the hierarchy among different classes and classify an object by class-hierarchy trees. The work most relevant to ours is the one by Bellver et al. [3], which represents an image as a hierarchical tree; the hierarchy is predefined, leaving the agent to iteratively select the next parts. In contrast, in our work, we do not assume a predefined decomposition of an image and demonstrate layers emerging from the process of learning automatically.

Recurrent Neural Network (RNN) The RNN (and its variants LSTM[11], GRU[9]) is a common model widely used in natural language processing and is recently applied to more computer vision tasks. Xu et al. [29] use the LSTM model to predict the sequence of words describing a given image. RNN is also employed to predict concrete attention zones to work for tasks of instance segmentation [24, 20]. One work that is most similar to ours is the Polygon-RNN [7], in which a sequence of vertices forming a polygon is predicted in a recurrent manner. One of the key differences with respect to this 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 Bezier curves as in the Polygon-RNN.

The discussion above is only a sampling of the studies most relevant to our work. It should also be mentioned that 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 manifests the importance of the topic, we believe that problem of understanding images abstractively and compositively, is underexplored.

3. Basic Model for Parameter Regression

In this section, we propose a framework based on a standard modification of the YOLOv2 model [19] that is inspired by [14], to perform parameter regression. The parameters regressed by the model, as opposed to [14], are fully interpretable and high-level.

3.1. Adapting YOLO for Parameter Regression

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

The proposed YOLOv2 network in the original paper consumes an entire image and segments it into a grid of dimensions $S \times S$. Each square in the grid could contain multiple primitives, the networks model this multiplicity as 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. There is a $K+5$ term, 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 [19]. While regressing parameters for the bounding boxes, the regressor would need to predict M extra variables for each bounding box that is being predicted. The M variables would be the total number of possible parameters from all the different primitive categories. This increases the number of predicted paramters of the network to $S \times S \times B \times (5 + K + M)$.

To achieve this end, a new loss term is added to the previously proposed loss function in [14]. The new term, $\mathcal{L}_\text{p}$, feeds information about the primitive parameters into the network. This term is defined as follows:

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

Here, $\mathbb{1}_{i,j}^{(k)}$ 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 the grid square (i,j) belongs to the primitive defined by l. The purpose of introducing this term is to provide a weighing of a primitive to the loss only when the primitive is plausible for the image. $X(l)$ is the set of parameters which belong to the 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. Thus the values of these parameters can directly be used as ground truth for training. For parameters within range $[0, 1]$, we can further increase the network training stability by applying a sigmoid function to the network output to constrain the estimated parameters.

Primitives with variable number of parameters Some of the primitives discussed in this paper, which include closed b-spline curves, have a variable number of control points. This enables primitives with the capability to represent different kinds of shapes. This 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, an 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 i^{th} control point c_i, and a stop probability $p_i \in [0, 1]$, that indicates the end of the curve.

The design of loss functions for the RNN-based model has to be dealt with care. Naively, one can use a simple mean-squared error (MSE) loss for control point position prediction and a cross entropy loss for the probability prediction. However, this only handles the situation where the sequence of control points is fixed and well-defined. It is to be noted that every point in the control point sequence $C = (c_1, c_2, \ldots, c_N)$ 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 [7] is defined as follows:

$$\mathcal{L}_{\text{circ}} = \min_{k\in[1,N]} \left(\min(\mathcal{L}(C, G_k), \mathcal{L}(C, G'_k)) \right) \quad (2)$$

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 i^{th} control point in ground truth, then G_k is the sequence $(g_k, g_{k+1}, \ldots, g_{N-1}, g_N, g_1, g_2, \ldots, g_{k-2}, g_{k-1})$ 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 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

Layer Detection Human designers typically use multiple layers to help with the design process while creating a vector graphics image. For example, when artists try to draw a ring, they usually draw the outer circle first, then draw the inner circle and deduct it from the outer one. This fact that all vector icon images can be decomposed into multiple layers as shown in Figure 1 serves as inspiration to extend the model proposed in Section 3 to include layered detection. Meanwhile, for the detection of each layer, it is not necessary to work on the entire image; instead, one could focus on a specific part of the image. For example in Figure 1, the white rectangle in the lower-right of image is completely inside the black disk, one can focus in the interior of the disk and thus 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 expected to be related. Therefore, we propose the 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 [12] on the intermediate output of our network. This enables us to extract the region in the image, where one could focus on, so as to perform detection at the next level.

Architecture After an image is forwarded through the backbone network, simple post-processing steps including thresholding and NMS are performed to obtain the final prediction results. The difference between the backbone network in this model and the previously discussed YOLO network, is that the backbone network is expected 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 detections 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, and each module is combined with pooling layers. We denote each module by $f_1, f_2, ..., f_M$ (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, instead, we crop the feature map using the the prediction results from d_{i-1} and resize it into a uniform size. In this way, the layering is represented explicitly by cropping within the interior of an image. The model could be expressed as:

$$B(1) = d_1(J_1) \quad (3)$$

$$B(i) = d_i(R[J_i; B(i-1)]), \quad i \geq 2 \quad (4)$$

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. To make it consistent within different regions in image, we perform training using the local coordinates within the parent bounding box as the groundtruth for $B(i)$. For example, consider an image with a rectangle inside a circle. Then, the groundtruth coordinates for the rectangle should lie within the local coordinate system w.r.t 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 the partial information in the image, as testified in Faster-RCNN [21]. Meanwhile, since there are multiple layers of covolutional operations, the feature map could encode some information outside the bounding box, thus enabling the model with the capability to correct mistakes made in the outer layers, considering both local and global information while making detections in inner layers

5. Experiments and Results

In this section, we present our implementation details, experiments and results.

5.1. Implementation Details

Primitive and Parameter Selection Four types of primitives are used in our experiments: rectangles, triangles, ellipses and closed spline curves. We observed that the pre-

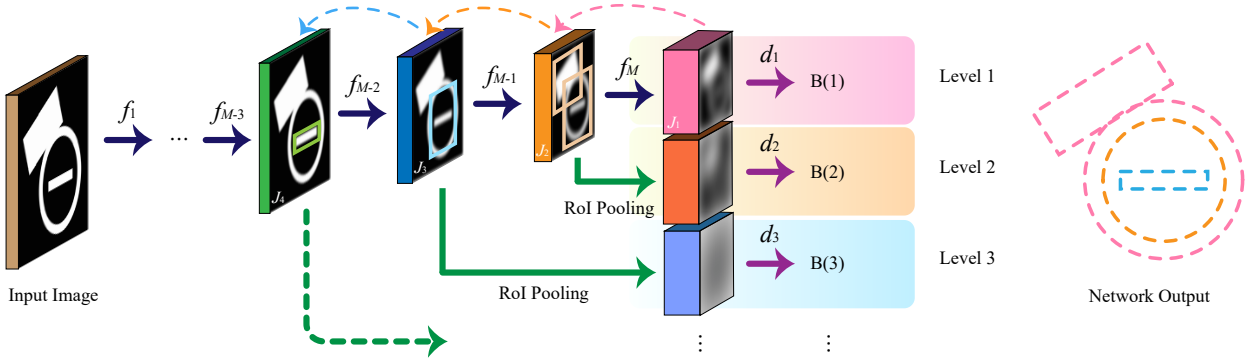

Figure 2. Illustration of the detection process in our layered model. Cuboids denote input images or feature maps. Dark blue arrows, dark green arrows and dark purple arrows represent conv layers, RoI pooling layers and detection blocks, respectively. The letters are consistent with the ones used in text. Final output of our network is a layered primitive tree containing both the shape information and the layer information.

dicted 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 to be able to perform better reconstruction.

Network Architecture Our code is adapted from an open source PyTorch implementation[1]. The backbone network used is the Darknet-19 architecture and the configuration is as in [19]. We set the deepest level of our layered detection model to 3, and hence a total of three detection blocks are used. Detailed configuration of the detection block $d_i (i = 1, 2, 3)$ is shown in Table 1.

Type	Channel	Filter size
Conv	$512 \rightarrow 1024$	3×3
Conv	$1024 \rightarrow 512$	1×1
Conv	$512 \rightarrow 1024$	3×3
Conv	$1024 \rightarrow 512$	1×1
Conv	$512 \rightarrow 1024$	3×3
Conv	$1024 \rightarrow C_{out}$	1×1

Table 1. The Detection Block d_i used in our network. Final output channels $C_{out} = B \times (5 + K + M)$.

Training The entire hierarchical model can be trained fully end-to-end. Additionally, we adopt a method similar to Scheduled Sampling [4] to enhance training stability and testing performance: The predicted $B(i-1)$ information from level $i-1$, which is fed into the level i, is substituted with the groundtruth value of level $i-1$ with a probability p. The value of p is set to 0.9 in the first 10 epochs and is decreased by 0.05 every 2 epochs after that.

An RNN decoder model is pre-trained separately to regress a fixed length embedding for control point positions.

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

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 image with our backbone Darknet-19 network are extracted. 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 position of control points instead. Note that the RNNs across different levels of hierarchies share the same weights.

Data Synthesis The hierarchical model was trained with 150,000 pictures with size 416×416 that were synthesized, and the hierarchical information during the generation process is recorded for training. The number of primitives in a single image are 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 is added to the shape of the primitives, the hatching pattern of the primitives and some skewing of the image itself is also done. Selected dataset images are shown in Figure 3.

5.2. Results and Analysis

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 $\mathcal{L}_{circ}$ and a simple MSE loss. As shown in the Table 2, training with circular loss leads to better convergence loss and thus reach better prediction results. Figure 4 shows two examples comparing the prediction results given the same curve image as input. We found that using the circular loss eliminates the ambiguity of starting point and clock direc-

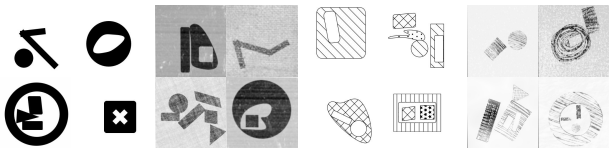

Figure 3. Examples drawn from our synthesized training dataset. For *Pure* dataset, we synthesized simple binary images for training. *Pure+Noise* dataset is a modified version of *Pure* dataset by adding noise and random affine transformations to each image. The design of *Textured* dataset tests the robustness of various shape detection methods by adding hatch pattern to the shapes. *Textured+Noise* dataset imitates real world hand drawn shape pictures.

tion in the training data and leads to more accurate fitting results.

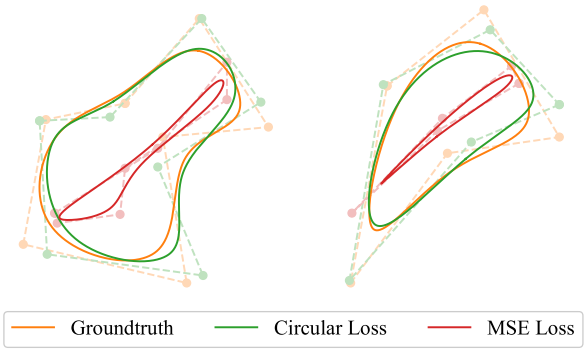

Figure 4. Two closed spline curve fitting cases using Circular loss and MSE loss.

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

Table 2. Error and accuracy measures during training and testing with two different loss functions. The **Loss** columns is the MSE distance between the groudtruth and predicted positions of control points (distances are all normalized to lie in the unit interval). The **# Point Acc.** columns is the accuracy of predicting the number of control points correctly.

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 and we set up 4 independent baselines for comparison. The first two baselines are traditional methods and the last two 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 several colinear segments with one single line segment with a parameter q controlling the strength of appoximation. The type of the shape is determined by counting the number of line segments (i.e., the number of edges of the shape). This method is implemented using `findContours` and `approxPolyDP` function in OpenCV [13].

- Hough Transform: Hough Transform [10] 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 [28].

- Flat model: This method uses a learning approach and is trained using YOLOv2 architecture. The groundtruth 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 in the current level). Once the detector successfully detects some primitives in the current level, we crop the detected region and resize the cropped region to the network input size and feed the image into the same network again.

Comparison results of these different models are shown in Table 3 (precision-recall-reconstruction comparison) and Table 4 (primitive-reconstruction comparison). Some of the prediction results of different methods using the same input are shown in Figure 5.

The contour method with smaller 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 (Recon) but low accuracy in shape classification tasks (Prec & Recall). On the contrary, contour method with a greater q value simply approximates continous curves with polygons, which explains why a bad reconstruction performance is observed. It is also observed that the contour method cannot separate overlapping primitives since this method 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 and we can see the Hough method generally

Method	Pure			Pure + Noise		Textured		Textured + Noise	
	Prec	Recall	Recon	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
Contour ($q = 2 \times 10^{-3}$)	94.0	72.8	1.70	32.5	60.1	16.8	88.0	15.6	73.2
Hough Transform	32.6	78.6	1.61	5.1	73.7	-	-	-	-
Flat Model	**99.7**	91.0	-	99.5	90.0	99.6	91.2	99.4	91.0
Recursive Model	96.1	72.4	1.64	60.1	61.2	74.0	60.1	95.8	49.9
Our Model	**99.7**	**96.1**	1.61	99.5	**95.0**	99.6	95.8	99.5	95.4
Our Model (Optimized)	**99.7**	**96.1**	**1.39**	**99.6**	**95.0**	-	-	-	-

Table 3. Precision, Recall and Reconstruction loss measures using various methods. **Prec** and **Recall** column show the precision and recall values in percentage respectively while **Recon** column measures the RMSE loss between the original picture and the reconstructed picture using the layered prediction results. The **Pure**, **Pure+Noise**, **Textured** and **Textured+Noise** paradigms are as described in Figure 3.

	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**

Table 4. Average precision (AP) measures of learning-based shape detection methods. The values are presented in percentage.

lead to higher recall value than the contour method. This method partially solves the overlapping problem by extending detected line segments and finding intersections, but cannot effectively distinguish between extremely short line segments and segments from a circle.

The above problems can be fixed in 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 handcrafted features of traditional methods. However, they have problems which occur either due to lack of or due to incorrect layering. The flat model detects almost all primitives regardless of their layer hierarchy. However, in cases where two primitives of the same kind (e.g., concentric circles forming an annulus) overlap with each other, the post-processing step eliminates one of them and predicts the median result, which is not desirable. It is also difficult to reconstruct the original image with 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. Compared to all the other baselines, our method can extract the high level shape information as well as their containing relationships. It should also be noted that 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 by Our Model (optimized) in Table 3, the reconstruction loss can be futher decreased.

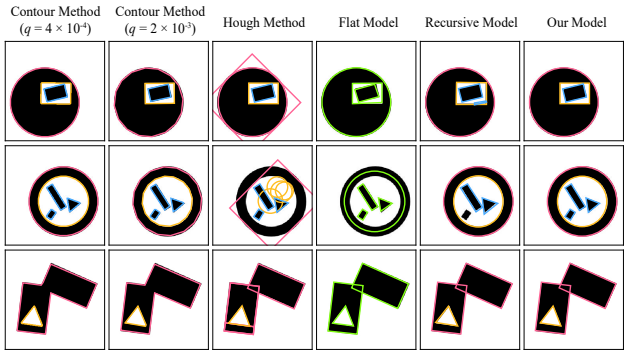

Figure 5. Detection results examples. Shapes detected at different levels are marked as different colors (level 1: pink; level 2: orange; level 3: blue). For the flat model, there is no predicted layer information, so all the shapes are marked as green.

The trained model was applied directly to Google Material icons[1] and selected results are shown in Figure 6. The first 14 images show optimized prediction results on Google Material icons using our trained model and the last 2 images show failure cases which are to be discussed later. More results are shown in the supplementary materials.

6. Applications

Once an image is decomposed into several layers and high-level parameters defining the primitives in the image are acquired, one can utilize this information and apply it to a variety of applications. In this paper, we demonstrate the use of these parameters in two applications as examples.

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 could decompose the image into several manipulatable components with high fidelity and flexibility. For example, in Figure 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.

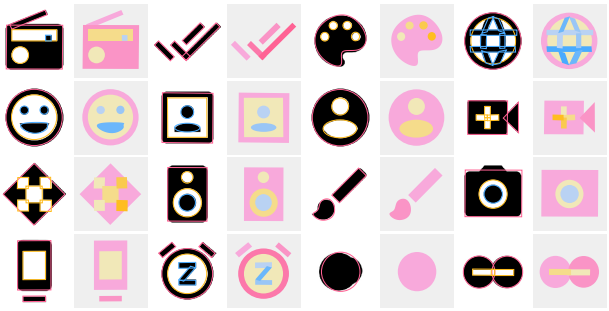

Figure 6. Selected test results of our layered detection model. The left picture in each column shows the original input image as well as detection result while the right picture in each column reconstructs the input image with the detection result (different instances of primitives within the same hierarchy vary a little bit in colors for clarity).

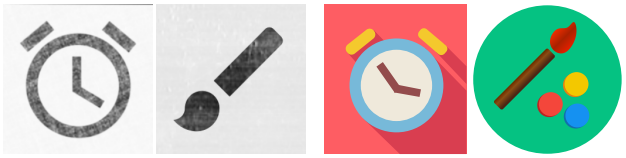

Figure 7. Image editing on a rasterized image at a primitive level. Primitive detection is performed on the image, followed by rearrangement and colorization of the primitives. The angle between dials of the clock is modified (*case 1*), and the shape of paint-brush head, along with the orientation, is modified (*case 2*).

Another potential application is in Recognition-by-components [6]. Usually, state-of-the-art classifiers based on deep networks need tons of data for training and the lack of training data has become one of the bottlenecks for accuracy. Once the primitives in an image are recognized, one could easily define certain classification rules using the layered information obtained. Additional training data is not needed and a single shape detection model has to be trained. The idea is illustrated in Figure 8. Given an image, pre-processing steps such as denoising or thresholding are performed to extract the border of shapes. The proposed model is then applied to detect the primitives and generate a shape parsing tree (written in XML format in the figure for demonstration purposes), with which a handcrafted classifier could easily predict the class of the object in the image by a top-down traversal of the tree.

Limitations As an explorative study aiming towards understanding and reconstructing images on a primitive level compositively layer-wise, there are several limitations left to be resolved in future works. 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 contained relationships: if one higher-level primitive intersects multiple lower-level primitives, then duplicate detections of the higher-level primitive are possible. The last two images in Figure 6 demonstrate the failure cases. These limitations restrict the layer decomposability of our model. Meanwhile, only synthesized black-and-white images are used for training. More annotated real-world data would make the model more generalizabe.

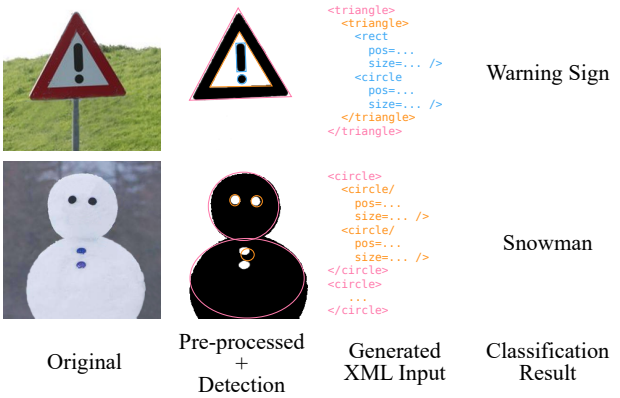

Figure 8. Recognition-by-components demonstration using our proposed hierarchical primitive detection model.

7. Conclusion

In this paper, a data-driven approach to layered detection of primitives in images and the subsequent 2D reconstruction has been demonstrated. As discussed earlier in Section 1, abstraction of objects into primitives is a very natural way of understanding objects for humans. As artificial intelligence moves towards performing tasks in human-like fashion, there is value in trying to understand these tasks in the way a human would.

Such tasks often also fall in the intersection of robotics and computer vision, such as in the case of autonomous driving or autonomous robotics. In these tasks, building in an environment-awareness into the robots based on its field of vision is key, where a 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 come in handy, due to the lack of open datasets to aid perform training on real-world data, good directions for future study would involve learning tasks of an unsupervised or self-supervised kind.

References

[1] Google material icon. https://material.io/icons/. Accessed: 2017-08-01. 7

[2] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981. 2

[3] M. Bellver, X. Giro-i Nieto, F. Marques, and J. Torres. Hierarchical object detection with deep reinforcement learning. In *Deep Reinforcement Learning Workshop, NIPS*, December 2016. 1, 2

[4] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1171–1179. Curran Associates, Inc., 2015. 5

[5] P. J. Besl, N. D. McKay, et al. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992. 2

[6] I. Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987. 1, 8

[7] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. *arXiv preprint arXiv:1704.05548*, 2017. 3, 4

[8] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2724–2729. IEEE, 1991. 2

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 3

[10] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the Acm*, 15(1):11–15, 1972. 6

[11] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002. 3

[12] R. Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015. 4

[13] Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2015. 6

[14] S. Jetley, M. Sapienza, S. Golodetz, and P. H. S. Torr. Straight to shapes: Real-time detection of encoded shapes. 2017. 3

[15] G. Lecot and B. Levy. Ardeco: automatic region detection and conversion. In *17th Eurographics Symposium on Rendering-EGSR'06*, pages 349–360, 2006. 2

[16] T.-Y. Lin, P. Dollr, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017. 2

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 1, 2

[19] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016. 1, 2, 3, 5

[20] M. Ren and R. S. Zemel. End-to-end instance segmentation with recurrent attention. 3

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2, 4

[22] L. G. Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963. 1

[23] D. F. Rogers and N. Fog. Constrained b-spline curve and surface fitting. *Computer-Aided Design*, 21(10):641–648, 1989. 2

[24] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *European Conference on Computer Vision*, pages 312–329. Springer, 2016. 3

[25] J. Sun, L. Liang, F. Wen, and H.-Y. Shum. Image vectorization using optimized gradient meshes. In *ACM Transactions on Graphics (TOG)*, volume 26, page 11. ACM, 2007. 2

[26] T.O.Binford. Visual perception by computer. In *IEEE Conference on Systems and Control*, 1971. 1

[27] W. Wang, H. Pottmann, and Y. Liu. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics (ToG)*, 25(2):214–238, 2006. 2

[28] Y. Xie and Q. Ji. A new efficient ellipse detection method. In *International Conference on Pattern Recognition, 2002. Proceedings*, pages 957–960 vol.2, 2002. 6

[29] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015. 3

[30] W. Zheng, P. Bo, Y. Liu, and W. Wang. Fast b-spline curve fitting by l-bfgs. *Computer Aided Geometric Design*, 29(7):448–462, 2012. 2

[31] L. Zhu, Y. Chen, A. L. Yuille, and W. T. Freeman. Latent hierarchical structural learning for object detection. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010. 2