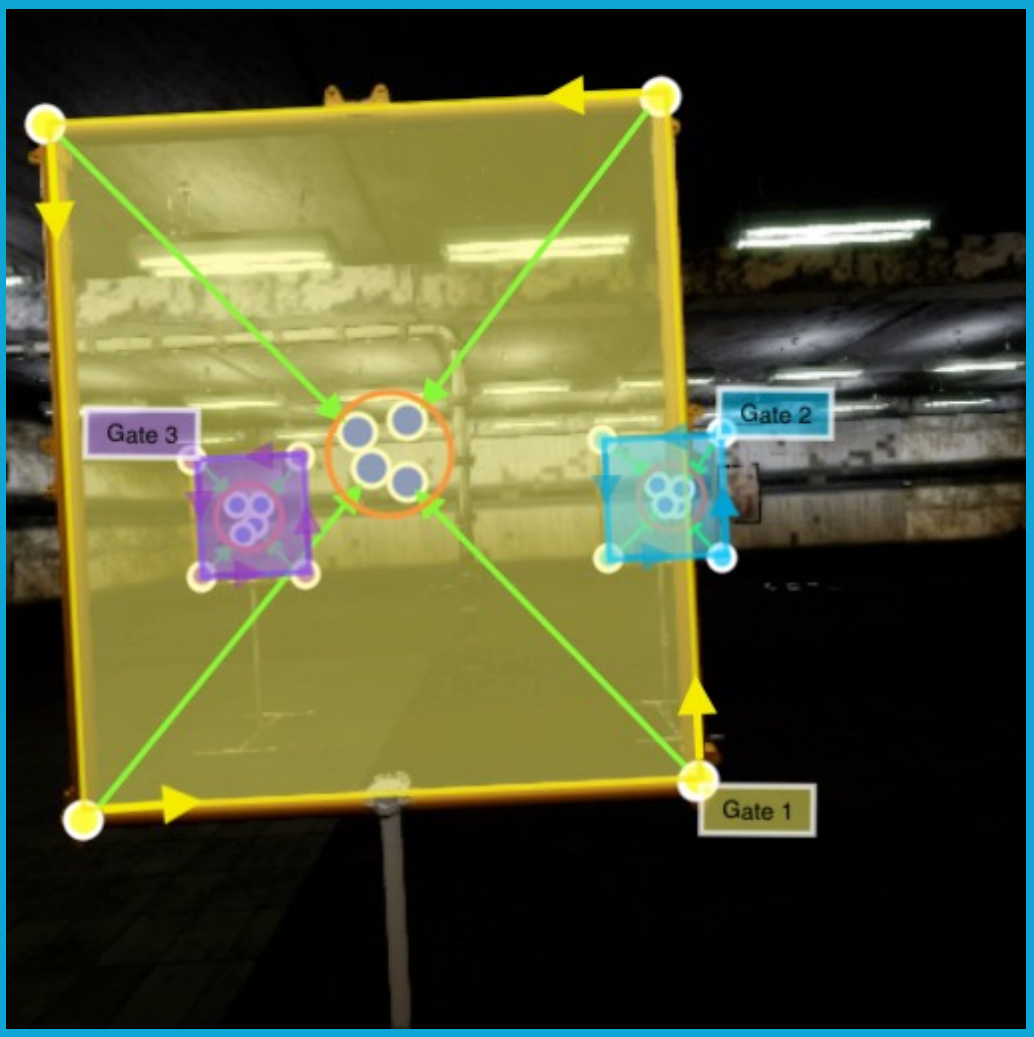
MSc thesis in Computer Science

Vertex-voting-based polygonal object detection

Kang Lang



TU Delft

VERTEX-VOTING-BASED POLYGONAL OBJECT DETECTION

by

Kang Lang

A thesis submitted in fulfillment of the requirements
for the degree of

Master of Science in Computer Science

at the Delft University of Technology,
to be defended publicly on Monday November 2, 2020 at 10:00 AM.

Student number:	4795245	
Thesis committee:	Prof.dr. Jan van Gemert ,	TU Delft
	Prof.dr. Asterios Katsifodimos ,	TU Delft
	Prof.dr. Silvia Pintea ,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

PREFACE

This paper "Vertex-voting-based polygonal object detection" is submitted as my master's graduation thesis. The research was conducted within Computer Vision Lab at TU Delft, under the supervision of Prof.dr. Jan van Gemert and Yancong Lin as my daily supervisor.

First, I would like to thank my thesis supervisor Prof.dr. Jan van Gemert and daily supervisor Yancong Lin for their guidance throughout my thesis. They provided many precious suggestions during the meetings and left me enough freedom to experiment on my own ideas. Without their kindly help, I cannot steer the research direction and get through all the difficulties.

Also, I want to thank Prof.dr. Silvia Pintea and Prof.dr. Asterios Katsifodimos for being interested in my thesis and evaluation of my work. Prof.dr. Silvia Pintea also gave me valuable suggestions during my work.

Finally, I need to say thanks to my parents and friends for their generous support. They made the stressful study life to be also colorful and memorable.

Kang Lang
October, 2020

Vertex-voting-based polygonal object detection

Kang Lang
Computer Vision Lab
Delft University of Technology

Yancong Lin
Computer Vision Lab
Delft University of Technology

Jan C. van Gemert
Computer Vision Lab
Delft University of Technology

Abstract

Although the pixel-wise labelling approaches have been exploited in depth and achieve good results in segmentation tasks, the grouped pixels are not ideal output for many end-users. In this paper, we propose a vertex-voting-based approach that can directly extract the polygon representations of objects. In order to better solve overlapping scenarios, we also propose a novel method that distinguishes objects by learning a virtual depth axis. When compared with the state-of-the-art method, our experiments demonstrate that this voting-based method is more robust to occlusion and shows a potential research direction.

1. Introduction

A fundamental task in computer vision is to extract objects in a refined representation. This task is usually solved by instance segmentation methods[4][19][8][38], which perform object detection and semantic segmentation simultaneously and produces a per-pixel classification result as the output. However, the grouped pixels are not ideal for many applications like urban mapping or sketching and have to be transferred to a more editable vector representation. Inspired by recent works[23][25][22], we investigate the possibility to directly output a vectorized polygon representation for each object in an image.

The idea for extracting a high-level, more abstract representation for objects is particularly researched in application fields like automated map generation. However, we notice that most related researches, such as [25],[46],[47],[34],[33],[11],[18],[48], were conducted on aerial images. We decide to jump out of the box and compete with the state-of-the-art on datasets where occlusion and overlapping scenarios exist. These scenarios were rarely considered by other works before. We propose a method to extract object depth information by importing a triplet loss[43].

We notice that polygon objects usually have distinct corner vertices. It is natural to first detect object vertices and assemble them into polygons. The mainstream polygonal object detection methods like [25][23] also use this idea. However, although they declared to follow a bottom-up pipeline, their underlying methodology is not. These methods follow an instance-first strategy, which means they generate object proposals first, then refine the proposals to extract vertices or lines. Our approach tries to avoid region proposal generation by using a voting method.

We validate our design on the gate detection dataset and compare with the state-of-the-art instance segmentation method. The proposed framework shows great potential in occlusion cases and indicates a future research direction.

2. Related work

2.1. Instance segmentation

Instance segmentation tries to solve object detection and semantic segmentation at the same time. Two major instance segmentation methods are FPN(Fully Convolutional Network[31])-based approaches and region-proposal-based approaches.

Region-proposal-based approaches are usually instance-first strategy, which means detecting instances first followed by segmentation. One representative network from this type is Mask RCNN[17]. By combining Faster R-CNN[40] and an additional branch for binary mask generation, it achieves the state-of-the-art when it was published and is widely used as a baseline method.

FCN-based networks are usually superior in semantic segmentation tasks but have some inherent problems when transferred to instance segmentation due to the lack of translation variant features. To solve this problem, Dai et al.[10] introduced a position-sensitive score map to FCN and used a downstream network for object detection sub-task. Following Dai's work, Li et al.[24] proposed fully convolutional instance-aware semantic segmentation method(FCIS) that shares the underlying convolutional representations and score maps to both object detection and segmentation sub-tasks. To generate box proposals, It combines RPN instead of sliding windows in Dai's work. However, FCIS exhibits errors when it comes to overlapping objects. Another variant in this category is proposal free networks.

Works like [20][30][27] try to avoid proposal prediction. Starting from per-pixel classification, they obtain instances by splitting the pixels within the same category. Liang et al. introduced Proposal-free Network[27], which predicts pixel-wise instance location maps followed by clustering to generate instance-level segmentation results. Similar works like [19][39] also predict object locations and use clustering methods to distinguish instances. Our paper uses the concept of proposal free networks and the voting strategy, but is different at its enhancement on the overlapping scenarios. Moreover, it only uses object vertices for voting so as to get vectorized polygon representation whose components are detected vertices.

2.2. Polygonal object detection

Early works related to polygonal object detection is a combination of contour extraction and subsequent contour simplification. For contour extraction, methods like level set[44], graph-cut[5] and the more advanced GrabCut[42] solve the problem in an energy minimization fashion. Performance of these methods rely on either color distribution [42][5] or condition initialization[44], thus are not accurate and require human intervention. Performance of the alternative methods like superpixel grouping[21] or object saliency detection algorithms[9] are also limited by line evidence or image contrast. As for the simplification step, it can be done on Douglas-Peucker algorithm[50]. These multi-step approaches usually yield high accuracy loss in practice.

Another strategy is based on assembling geometrical primitives, like line segments in the image, into closed polygons. Sun et al.'s work[45] detects polygonal objects by partition a weighted line fragment graph.

Polygonal object detection can also be achieved by grouping the over-segmented polygon cells in an image. Polygonal partition can be achieved by methods like [1][49][12][15][3], while the grouping can be done using aggregation mechanisms [41][26]. This strategy's performance depends on fine polygonal partitions that fit the input image well, which is hard to achieve in practice.

Neural network based polygonal detection is a new research direction with limited works been done so far. Polygon-RNN[7] proposed a semi-automatic approach to annotate object instances. This work, however, requires a human annotator to provide object bounding boxes. It's improved version [2] introduces a graph neural network (GNN) to increase output resolution, but the system remains non-automatic. More recent work like PolyMapper[25] and [23] introduce automatic approaches to directly predict polygon representations. Both works solve polygon detection using a two stage pipeline: keypoint detection and grouping. PolyMapper[25] combines CNNs and RNNs with convolutional LSTM modules, which shows good results on aerial images. While [23] avoids intermediate learning of object boundaries in PolyMapper and uses a fully geometric-based grouping method instead of deep learning modules. Our design falls into this category but is different on both keypoint detection and grouping stages.

3. Method

3.1. Object Representation

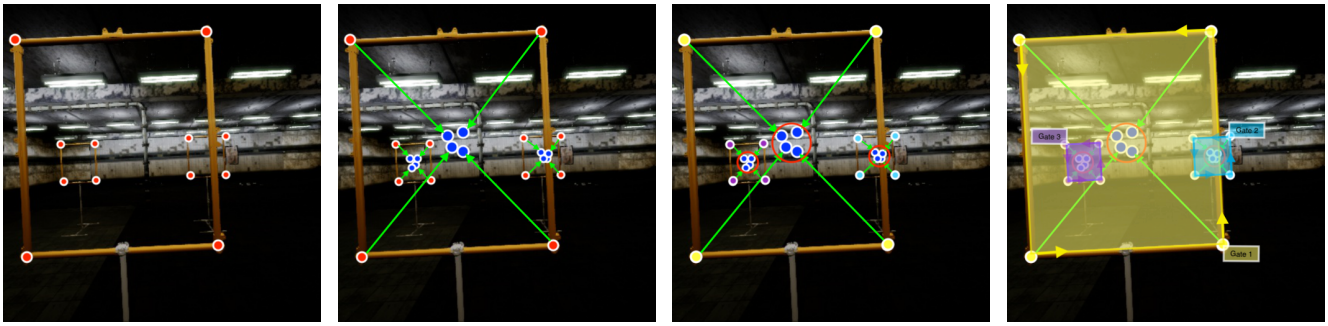
Instead of a group of pixels, we represent objects using closed polygons for two reasons: (1). Compared to pixels, polygons are not resolution-dependent and can be manipulated more easily from the user's point of view. (2). Polygon can approximate man-made objects well as they are usually a combination of polygon shapes. Similar to works like [22][25], we represent the polygons in a vector data structure where the elements are polygon vertices.

3.2. Overall Pipeline

Prior works like [25][23] tried to integrate ideas from Region-proposal-based instance segmentation frameworks [17][29]. They used region proposal network (RPN) to first localize instances, followed by further refinement. Out of the consideration that region proposal generation itself is a challenging task and the data structure of polygon representation, it is natural to avoid instance-first strategy and use a proposal-free design.

Our framework follows a bottom-up pattern, which means the final results are built on early detected geometrical cues. Firstly, object vertices are extracted. For each vertex, a voting vector that points to its corresponding object center will be predicted as well. Then, by adding the vertices' coordinates to the voting vectors, we can obtain a set of predicted object centers. After that, if two vertices have geometrically close predicted object centers, they will be considered from the same object and be grouped together. In the last step, all grouped vertices will be re-arranged in particular orders so as the final vector representations can form closed polygons. Figure 1 provides an overview of the workflow.

We divide all tasks into two consecutive stages: (1). A multi-task learning neural network is responsible for the prediction of a set of feature maps which contains geometrical cues. The feature maps include a junction likelihood map for vertices detection, a voting vector map for object center voting, a remapping map for accuracy compensation during re-scaling image and another feature map which will be used to deal with overlapping scenarios. (2). A post-processing stage which infers polygon representations based on feature maps from the first stage.



(a) Vertices prediction (b) Center voting (c) Vertices grouping (d) Re-arranging vertices sequence

Figure 1: Workflow of the our approach.

3.3. Multi-task learning stage

Considering multiple objectives need to be solved in the first stage, we propose a multi-task CNN architecture based on [6],[28] and [19]. As shown in Figure 2, this architecture is composed of a common feature extractor and four single-task decoder branches. By sharing feature representations, this architecture can reduce computation run-time when compared with training all branches individually.

Backbone network The backbone network functions as a feature extractor for successive modules. As traditional CNN modules use a layer-by-layer hierarchy to extract features, an inbuilt multi-scale pyramid is formed implicitly. Thus, when network depth increases, output feature maps tend to have more abstract global features at the cost of local information. In our case, the prediction of voting vectors requires the receptive field to be large enough to include both object vertices and object centers. On the other hand, the prediction of vertices requires fine low-level information from the smaller receptive field. As a result, we use the Hourglass network as the backbone because it can provide a balanced combination of global and local features due to the use of skip layers. Similar to [37][51], we gradually refine the feature maps by stacking hourglass modules accompanied with intermediate supervision. The loss is the sum of the losses on all stacked modules.

3.4. Decoder branches

The goal for decoder branches is to learn a mapping from shared feature representations to target outputs. All decoders in our network share identical structure and are composed of several fully convolutional layers. The only difference lies in the number of output channels and the attached activation functions.

Object vertices localization branch: To extract the probability for a pixel to be an object vertex from certain category, this branch performs pixel-wise classification. The output has $M + 1$ channels where M represents the number of given classes in a dataset and the additional one channel means background. A Softmax function is used on the output to normalize the probability distribution. We use cross entropy loss as the loss function for this branch.

Voting vectors prediction branch: This branch is used to predict a voting vector for every object vertices. We represent the vectors as a combination of its projection on x-axis and y-axis, which results in a 2-channel output. As the vectors pointing from vertices to object centers can be either positive or negative, we use hyperbolic tangent function(Tanh) as the activation function. During inference phase, the output feature map will be re-scaled from range $[-1, 1]$ to original image space by multiplying the two output channels with image width and image height.

Virtual z-axis prediction branch: One problem bounded with voting based methods is poor handling on overlapping objects. This is because clustering algorithms cannot distinguish objects in 2D image space if their centers are geometrically close to each other. We tackle this problem by importing an additional one-channel feature map for depth information extraction. As no annotated depth information is provided, we use the triplet loss function during training. Its concept is similar to [36], all pixels from the same object should be assigned similar values while values for pixels from different objects will be pushed away from each other. On output feature map *O*, the loss is calculated as follows:

$$L(a, p, n) = max(0, D(O_a, O_p) - D(O_a, O_n) + margin)$$

where a represents the anchor pixel, p is the positive pixel which falls in the same category as the anchor. n is the negative pixel which belongs to a different category. All pixels are randomly selected. Function D represents a distance function. And the margin means the target interval between $D(O_a, O_p)$ and $D(O_a, O_n)$. We adapt the triplet loss by average over all anchor, positive, negative combinations. Although we conjecture that this branch can learn object depth

information, the result is nothing like the traditional z-axis. Thus we name it virtual z-axis.

Remapping offset prediction branch: This branch serves to predict the decimal values that have been rounded when we downsample the image. Values from this branch will be added to the detection result before we rescale to original image space in order to compensate for accuracy loss. Due to the range of the offset along x-axis and y-axis should be limited to $[\frac{-1}{2}, \frac{1}{2}) * [\frac{-1}{2}, \frac{1}{2})$, we apply a sigmoid activation function and add a -0.5 shift value to the result.

According to Liebel et al.'s research[28], a decoder may favor certain feature representations learned by the encoder, while some of them may also be exploited by other decoders. Thus this multi-task joint learning approach is believed to be able to boost the overall performance when compared with training the branches separately.

3.5. Post-processing stage

The post-processing stage is a combination of several unsupervised learning algorithms. It takes all output feature maps from the previous stage to infer the polygon approximations of objects.

The first step for post-processing is vertices extraction. This is done by extracting the top *N* pixels with the highest objectness score on the junction likelihood map predicted in the previous stage.

Secondly, for each candidate vertex i, we sum its coordinates $[X_i, Y_i]$ with its corresponding voting vector $[Vx_i, Vy_i]$ to obtain the predicted object center $[Cx_i, Cy_i]$.

As vertices of the same object tend to vote for geometrically close centers, we can assign all vertices from the same class category to different objects by applying a clustering algorithm. We use DBSCAN[14] as the clustering method in our paper. DBSCAN, which is a non-parametric density-based clustering algorithm, can efficiently group closely packed points. We choose it for two reasons: (1). It requires no prior knowledge of the number of clusters like K-means[32]. (2). It has reasonable time complexity. On average DBSCAN's time complexity is $O(n \log n)$. Thus it will not bring too much computation cost.

As shown by the input image in figure 2, overlapping objects can have very close centers, and their vertices may be incorrectly grouped together. To separate them, we perform DBSCAN the second time along the virtual z-axis.

By far, the vertices from the same object should be grouped together. However, all the vertices need to be arranged in certain order so as to correctly represent the polygon that encloses the target object. To obtain the vectorized polygon representations, we apply Graham scan[16] on the vertices. Graham scan is an algorithm that can find a convex hull of a given set of points with time complexity $O(n \log n)$.

Input image has been downsampled in the first stage to reduce computation pressure, and all post-processing steps by far are applied on the shrunken image. As a result, the last step is to remap detection results back to the original image space. To compensate for the accuracy loss mentioned in the previous section, we add the predicted decimal values to vertices' coordinates before re-scaling them. For the detected polygonal objects, its class label is identical to its vertices' labels. Moreover, its confidence score, which is used to filter out weak predictions, is the mean of all its vertices' objectness scores.

4. Experiments

4.1. Dataset

Testing of polygon detection approaches is usually limited to their application fields. Experiments are conducted mainly on satellite imagery like the crowdAI dataset[35]. However, there is usually no overlapping scenarios in such datasets, which we are also interested in. In this respect, we use the gate detection dataset from P. Duernay et al.'s work[13].

The gate detection dataset is synthesized using Epic's Unreal engine[1] and AirSim-Plug-In[2] by Microsoft. Hollow racing gates are randomly scattered in various environments.

We select four sub-datasets from the gate detection dataset to validate our design. Details of the datasets are shown in table 1 and figure 4.

Dataset name	Number of images	Illumination condition
Basement1	2441	Low
Basement3	1407	Low
Iros1	3163	Medium
Daylight1	3412	High

Table 1: Details of the datasets used in this paper

There are several flaws in this dataset that may degrade the performance of our approach. The first problem is that some annotations of gate vertices are out of image space. Those vertices are not usable in our framework as we use per-pixel classification for vertices extraction. We resolve this problem by recalculating alternative vertices within image space, which results in many pixels around image corners being marked as vertices even though they contain no object information at all. Another issue is that the vertices annotations are not accurate. This issue may reduce the performance of the vertex-voting-based method by a large margin as the network is distracted to learn background instead

[1]https://www.unrealengine.com/en-US/what-is-unreal-engine-4
[2]https://github.com/Microsoft/AirSim

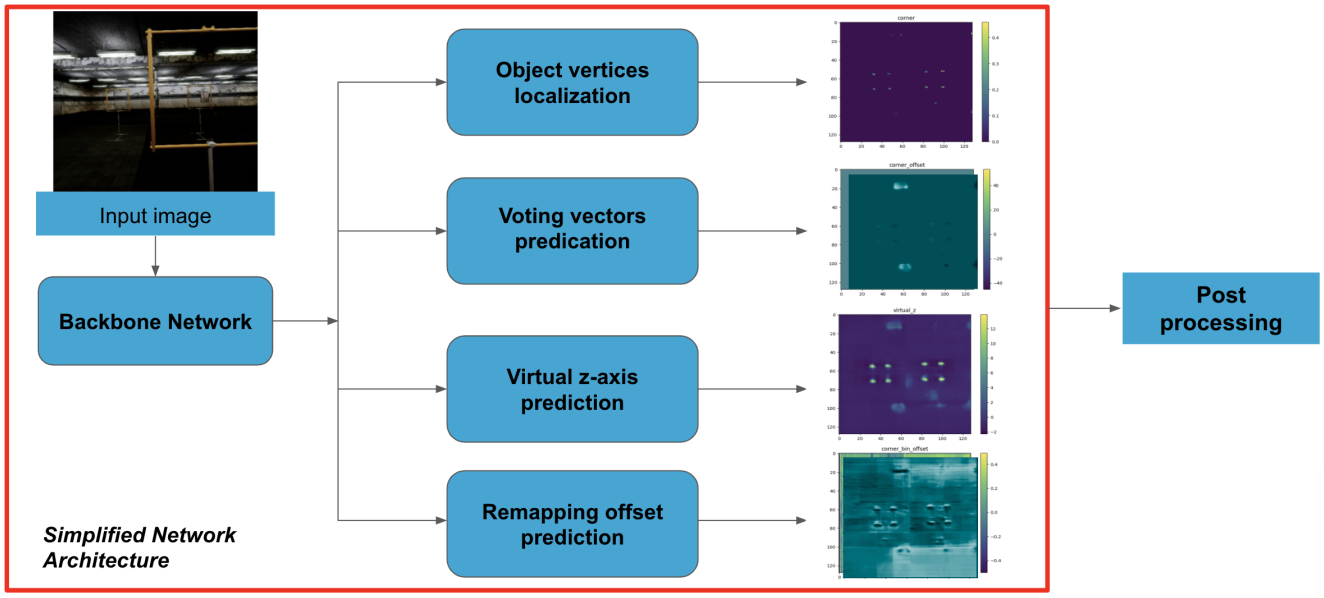


Figure 2: An overview of our network architecture

of object features. Both issues are shown by the gate whose ground truth boundary is marked in red in image 3.

Figure 3: An example of the flaws bounded with this dataset

4.2. Implementation Details

For our multi-task learning network, we implement the encoder following Zhou et al.'s design[51]. The input image, after brightness augmentation and rotation augmentation, will be first processed by a 7×7 stride-2 convolution. Then, three residual blocks accompanied with stride-2 max-pooling will be applied to downsample the input image to 128×128 with $channel = 256$. The intermediate features will be further processed by two stacked hourglass modules for feature extraction and sent to four successive single-task branches. The depth of hourglass is set to be 4. As for the four branches, a 3×3 convolutional layer and a 1×1 convolutional layer are used with ReLU between them as the activation function. After being converted to the target dimension by the 1×1 convolution, each branch applies different activation functions and loss functions as demonstrated in the previous section. The loss function is a combination of the losses from all individual branches and intermediate supervision. The loss contributed by negative background pixels is multiplied by a factor that equals 0.0005 so as to solve the class imbalance problem. Inspired by [28], we introduce four learnable weights to maintain the balance between each branch's loss contribution. We use Adam as the optimizer and initialize learning rate as 4×10^{-4}. Weight decay is set to be 1×10^{-4}. Identical to Zhou et al.'work[51], we reduce the learning rate by 10 after 10 epochs.

In the post-processing stage, we extract 50 candidate vertices with highest objectness score after applying Non-maximum suppression(NMS) with a 3×3 max-pooling. When DBSCAN is used in 2D image space, we set its hyperparameters as epsilon=7 and minPts=1. For the second time when we apply DBSCAN on virtual z-axis, we use epsilon=2 and minPts=1. A confidence threshold that equals 0.5 is used to filter out objects with low confidence.

4.3. Comparison to State-of-the-art

To evaluate the performance of our network, We use average precision(AP, averaged over IoU thresholds). AP_{50}(IOU threshold = 0.5),and AP_{75}(IOU threshold = 0.75). We additionally evaluate average recall(AR) to measure the proportion of gates extracted by our approach with respect to the ground truth. Both AP and AR are calculated based on mask IOU, but we need to mention that in our framework the outputs are polygon representations rather than pixel-wise masks.

(a) Basement1 (b) Basement3 (c) Iros1 (d) Daylight1

Figure 4: Examples of the gate detection dataset. (a) and (b) are from Basement1 and Basement3 respectively, where gates are set in a basement with low artificial illumination. In (c), illumination is a combination of natural light and artificial light. (d) is from the Daylight1 dataset with intense natural light through windows.

We compare our network with the state-of-the-art method Mask R-CNN[17]. The Mask R-CNN uses ResNet50+FPN as backbone with scale jittering and horizontal flipping as augmentation methods. Compared with our design with only 10,475,790 parameters, Mask R-CNN is implemented with 43,918,038 parameters, almost 4 times as many as ours. We train both networks on Basement1 dataset using two NVIDIA GTX 2080Ti GPUs for 100 epochs. Batch size is set to maximize GPU memory usage. The training takes about a day to finish.

We test both networks on three datasets with different illumination conditions. The test results are shown in table 2. During inference, Mask R-CNN uses 0.055 s/image on average while our design uses 0.085 s/image on neural network stage, and 0.006 s/image on post-processing stage. This shows that our post-processing method will not bring too much computation pressure when the number of vertex candidates is low.

After being trained on Basement1 dataset with low illumination condition, both networks experience gradual performance drop when tested on datasets with medium and strong illumination. Mask R-CNN performs better than our framework on all datasets. However, we suspect that Mask R-CNN's advantage is not gained from learning more robust feature representations of gates. As racing gates are hollow objects, the majority pixels within a gate belong to the background image. Compared with the vertex-voting-based method whose result is obtained from grouping vertices, Mask R-CNN may also take background pixels as a hint of objects. Thus, although illumination changes, the similar indoor scenery can still provide a performance boost for Mask R-CNN.

To validate our idea, we pad three textures inside the racing gates. The three textures are sky, lava and stone brick, which are irrelevant to the indoor scenery in gate detection dataset. Examples of texture padding are shown in figure 5. The padding experiment, although may block some overlapping objects, is fair to both networks. We test both networks on the padded datasets and show the results in table 3. In general, the voting-based method is slightly better than Mask R-CNN. One exception is on the padded daylight1 dataset where Mask R-CNN still has advantages. We attribute the failure on daylight1 dataset to missing vertices. As shown in image 4d, some object vertices are hard to recognize even for human due to the intense light. As a gate only consists of four vertices, Losing even one vertex will cause significant IOU reduction between predicted objects and the ground truth. We expect to ease this problem by predicting more vertices, which is one of the future directions.

Based on these two experiments, the vertex-voting-based approach appears to focus more on the learning of object features while Mask R-CNN tends to import more context information during prediction. It reveals that our voting-based approach is more robust to context changes and occlusion scenarios.

4.4. Ablation Study

In this section, we study our proposed network by comparing it with two of its variants. The results are shown in table 4.

Virtual z-axis prediction branch: To validate the efficiency of virtual z-axis, we train another network which shares the identical design with our proposed method except for the virtual z-axis prediction branch. In general, the performance of our design is higher by a small margin. The exception on daylight1 dataset can also be attributed to the loss of vertices due to intense illumination. In daylight1 dataset, many object vertices cannot be extracted. By applying an additional clustering along virtual z-axis, the grouped vertices will be further split into smaller groups, leading to an insufficient number of points to form closed polygons.

(a) Sky texture (b) Lava texture (c) Brick texture

Figure 5: Examples of texture padding on gate detection dataset.

Dataset	Basement3				Daylight1				Iros1			
Metrics	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR
Mask R-CNN	**69.0**	**89.8**	**82.5**	**72.5**	**31.9**	**53.9**	**37.2**	**37.6**	**63.9**	**84.3**	**74.9**	**68.2**
Ours	64.8	82.8	77.2	69.6	21.6	37.8	24.9	32.6	54.8	75.1	66.6	62.7

Table 2: Evaluation results on gate detection dataset

Interpretable object detection approach: We force the network to solve object detection in an interpretable way. The network is trained to first look for potential object centers in 2D space, followed by distinguishing overlapping objects using depth information. We compare our design with a network which is trained to directly vote in 3D space. In the 3D voting network, the virtual z-axis prediction branch and the 2D voting vectors prediction branch are replaced with a branch that can predict three-dimensional voting vectors. The only constraint for this variant is that vertices from the same object should vote for geometrically close location. As shown in table 4, our interpretable approach has much better performance. We conjecture it is related to the constraints imported by the interpretable approach during optimization.

5. Conclusion

In this paper, We have proposed an vertex-voting-based approach which is able to extract vectorized polygon representations for objects. We have shown that by importing a triplet loss, voting based method can improve its ability to distinguish overlapping objects. We have also shown that forcing network to learn an interpretable solution is beneficial to its performance due to the imported constraints during optimization. Although our approach is not better than the state-of-the-art in all cases, it still shows its potential to occlusion and background context changes.

There are still many questions left unanswered. Firstly, as finding optimal hyper-parameters is hard and time-consuming, how to make the voting-based method end-to-end? Secondly, how to further improve the depth learning branch? Is it also possible to extract depth information in a comprehensive way rather than using triplet loss? Last but not least, will the increase of vertex voters improves our design's false tolerance? If so, how to control the complexity of the extracted polygon?

References

[1] R. Achanta and S. Susstrunk. Superpixels and polygons using simple non-iterative clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4660, 2017. 2

[2] D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 859–868, 2018. 2

[3] J.-P. Bauchet and F. Lafarge. Kippi: Kinetic polygonal partitioning of images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2018. 2

[4] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1

[5] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001. 2

[6] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Pro-*

Dataset	Basement3				Daylight1				Iros1			
Metrics	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR
Mask R-CNN	55.2	**77.7**	67.1	58.7	**22.0**	**38.8**	**23.2**	**27.1**	43.9	62.8	52.6	48.0
Ours	**59.6**	75.2	**70.5**	**64.7**	15.8	27.0	17.8	24.3	**44.0**	**58.1**	**53.4**	**49.5**

Table 3: Evaluation results on the paded gate detection dataset

Dataset	Basement3				Daylight1				Iros1			
Metrics	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR
Ours	**64.8**	**82.8**	**77.2**	**69.6**	21.6	37.8	24.9	32.6	**54.8**	**75.1**	**66.6**	**62.7**
No virtual z-axis	64.2	82.1	76.4	69.2	**23.9**	**42.2**	**27.3**	**33.5**	52.4	71.2	63.8	59.1
3D voting	26.2	34.4	32.2	31.6	2.2	3.8	2.5	4.2	19.6	28.4	23.0	25.7

Table 4: Ablation study. We compare our original design with two variants of our network. One of them uses no virtual z-axis and another is trained to directly vote in 3D space.

ceedings of the IEEE conference on computer vision and pattern recognition, pages 7291–7299, 2017. 3

[7] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5230–5238, 2017. 2

[8] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[9] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu. Global contrast based salient region detection. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):569–582, 2014. 2

[10] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016. 1

[11] M. Dalla Mura, J. A. Benediktsson, B. Waske, and L. Bruzzone. Morphological attribute profiles for the analysis of very high resolution images. *IEEE Transactions on Geoscience and Remote Sensing*, 48(10):3747–3762, 2010. 1

[12] L. Duan and F. Lafarge. Image partitioning into convex polygons. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3119–3127, 2015. 2

[13] P. Duernay. Detecting empty wireframe objects on micro-air vehicles. 2018. 4

[14] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. 4

[15] J. Forsythe, V. Kurlin, and A. Fitzgibbon. Resolution-independent superpixels based on convex constrained meshes without small angles. In *International Symposium on Visual Computing*, pages 223–233. Springer, 2016. 2

[16] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Info. Pro. Lett.*, 1:132–133, 1972. 4

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1, 2, 6

[18] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler. Learning aerial image segmentation from online maps. *IEEE Transactions on Geoscience and Remote Sensing*, 55(11):6054–6068, 2017. 1

[19] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 1, 2, 3

[20] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother. Instancecut: from edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5008–5017, 2017. 2

[21] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Optimal contour closure by superpixel grouping. In *European Conference on computer vision*, pages 480–493. Springer, 2010. 2

[22] M. Li, F. Lafarge, and R. Marlet. Approximating shapes in images with low-complexity polygons. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8633–8641, 2020. 1, 2

[23] Q. Li, L. Mou, Y. Hua, Y. Sun, P. Jin, Y. Shi, and X. X. Zhu. Instance segmentation of buildings using keypoints. *arXiv preprint arXiv:2006.03858*, 2020. 1, 2

[24] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2359–2367, 2017. 1

[25] Z. Li, J. D. Wegner, and A. Lucchi. Topological map extraction from overhead images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1715–1724, 2019. 1, 2

[26] Z. Li, X.-M. Wu, and S.-F. Chang. Segmentation using superpixels: A bipartite graph partitioning approach. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 789–796. IEEE, 2012. 2

[27] X. Liang, L. Lin, Y. Wei, X. Shen, J. Yang, and S. Yan. Proposal-free network for instance-level object segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2978–2991, 2017. 2

[28] L. Liebel and M. Körner. Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334*, 2018. 3, 4, 5

[29] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2

[30] S. Liu, J. Jia, S. Fidler, and R. Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3496–3504, 2017. 2

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[32] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. 4

[33] D. Marmanis, K. Schindler, J. D. Wegner, S. Galliani, M. Datcu, and U. Stilla. Classification with an edge: Improving semantic image segmentation with boundary detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, 135:158–172, 2018. 1

[34] D. Marmanis, J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla. Semantic segmentation of aerial images with an ensemble of cnss. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2016*, 3:473–480, 2016. 1

[35] S. P. Mohanty. Crowdai dataset, https://www.crowdai.org/challenges/mapping - challenge/dataset_files. 2018. 4

[36] A. Newell, Z. Huang, and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in neural information processing systems*, pages 2277–2287, 2017. 3

[37] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016. 3

[38] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European conference on computer vision*, pages 75–91. Springer, 2016. 1

[39] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9277–9286, 2019. 2

[40] 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. 1

[41] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2018, 2013. 2

[42] C. Rother, V. Kolmogorov, and A. Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004. 2

[43] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015. 1

[44] J. A. Sethian. Level set methods, evolving interfaces in geometry, fluid mechanics comuputer vision, and materials sciences. *Cambridge Monographs on Applied and Computational Mathematics, 3*, 1996. 2

[45] X. Sun, C. M. Christoudias, and P. Fua. Free-shape polygonal object localization. In *European Conference on Computer Vision*, pages 317–332. Springer, 2014. 2

[46] P. Tokarczyk, J. D. Wegner, S. Walk, and K. Schindler. Beyond hand-crafted features in remote sensing. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:W1, 2013. 1

[47] M. Volpi and V. Ferrari. Semantic segmentation of urban scenes by learning local class interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–9, 2015. 1

[48] M. Volpi and D. Tuia. Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):881–893, 2016. 1

[49] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32(4):722–732, 2008. 2

[50] S.-T. Wu and M. R. G. Marquez. A non-self-intersection douglas-peucker algorithm. In *16th Brazilian symposium on computer graphics and Image Processing (SIBGRAPI 2003)*, pages 60–66. IEEE, 2003. 2

[51] Y. Zhou, H. Qi, and Y. Ma. End-to-end wireframe parsing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 962–971, 2019. 3, 5

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Instance segmentation, as a fundamental task in computer vision, aims at performing object detection task and semantic segmentation task simultaneously. Most existing instance segmentation frameworks produce a per-pixel classification result as the output. However, the grouped pixels are not ideal for many applications like urban mapping or sketching and need to be transferred to a more editable vector representation. Thus, we decide to directly provide the vectorized polygon representations for objects in an image through our work.

We decide to represent objects using polygons in a vector data structure where the components are object vertices. This is due to the following reasons: (1). Compared to pixels, polygons are not resolution-dependent and can be manipulated more easily from the user's point of view. (2). Man-made objects are usually a combination of polygon shapes, thus can be represented well by polygons.

Moving away from pixel-wise labelling to polygon detection is a rather new field. There are limited works in this field[15][8][4][2][3][48][40][37]. Motivated by the satisfactory result from Zuoyue et al.'s work[40], we design the network in a bottom-up fashion, which means detecting geometrical cues(vertices) first and inferring final result based on them. Different from Zuoyue's work, we don't generate region proposals due to the consideration that proposal generation itself is a challenging task. Instead, inspired by [42][32][52], we draw polygon predictions using a voting method. We only use object corners as voters as they have better localization properties than normal texture pixels which locate within object boundary. Our framework has two stages: the first stage is responsible for vertices detection and object center prediction for each vertex, which is done by the neural network. The second stage uses an unsupervised learning algorithm to group detected vertices and provide the final polygon representation for every object. We also enhance our network's ability to deal with overlapping scenarios using a virtual depth axis predicted by the network.

1.1 RESEARCH QUESTIONS

We have following research questions addressed in this paper:

- How to avoid explicit pixel-wise labeling and provide more abstract representations for objects
- How to deal with the overlapping or occlusion scenarios as they are rarely considered in other polygonal detection works

1.2 OUTLINE

This paper is arranged in the following structure. In Chapter 2, we will provide needed background knowledge for our paper. Chapter 3 illustrates the data set we used in our experiments. Chapter 4 introduces the detailed design of our framework. Then in chapter 5, we will describe the experiment setting and results to answer the research questions. Chapter 6 will provide a conclusion and some flaws of our design for future study.

2 BACKGROUND

This chapter will provide the required knowledge for readers to understand other parts of the thesis. First, the vision-task-oriented deep learning technique - Convolutional Neural Network will be introduced. Then, general introduction to several computer vision tasks will be covered. We will provide some knowledge about polygonal detection, multitask learning and some algorithms we used in our design at the end of this chapter.

2.1 CONVOLUTIONAL NEURAL NETWORK

2.1.1 Representation Learning

Generally, whether machine learning algorithms can be effective is highly related to the representations of input data. The large Intra-class and inter-class variation make manually designed feature representations unreliable when it comes to changes like illumination condition, resolution etc. Also, further improvements of AI require it can automatically understand and extract useful information from observed data with less human interfering. As a result, representation learning methods received rapid development and led to the transfer from human-designed representations to CNN-based representations in computer vision field.

2.1.2 Early stage feature learning methods

Based on the idea that gradient distributions and edge directions can have a good description for local objects' appearance and shape, gradient-based descriptors like Scale-invariant feature transform(SIFT)[47], histograms of oriented gradients(HOG)[14], as well as edge-based descriptors are widely used in the early stage of computer vision.

These traditional feature extraction methods received reasonable results in many tasks, including real-world challenges. For example, in Dalal et al[14]'s work, the combination of SVM and gradient-based descriptor shows satisfactory results on pedestrian detection.

However, traditional feature representations have a significant problem that it is not intelligent enough. This period's 'Artificial Intelligence' cannot automatically learn useful hidden information from observed data. Instead, in order to achieve higher performance, most efforts are devoted to designing proper feature representations. This so-called 'feature engineering' uses domain-dependent prior knowledge and thus does not have good generalization ability[5].

2.1.3 Convolutional Neural Network(CNN)

Inspired by the concept of receptive field from the breakthrough research in animal visual cortical system[31], Fukushima et al.[23] proposed 'neocognltron', which is a multilayered network consisting neural-like cells. 'neocognltron' imports a hierarchical structure. shallower layers can only observe a narrow area of the input image and are in charge of extracting simple features while deeper layers extract more complicated features based on extracted information from previous layers. 'Neocognl-

tron' marks the start of convolutional neural network. In 1990, LeCun et al.[35] took one step further by combining the idea of back-propagation learning procedure[58] and multi-layer network and achieving the state-of-the-art in real-world handwritten digit recognition problem. Due to convolutional neural network's potential on large practical tasks, it has become a research hotspot especially in computer vision field and brought about many effective solutions to problems related to image classification, object detection, semantic segmentation and instance segmentation etc. Subsequently, finding efficient network architecture has replaced feature representation and become the core task.

2.1.4 Basic architecture of CNN

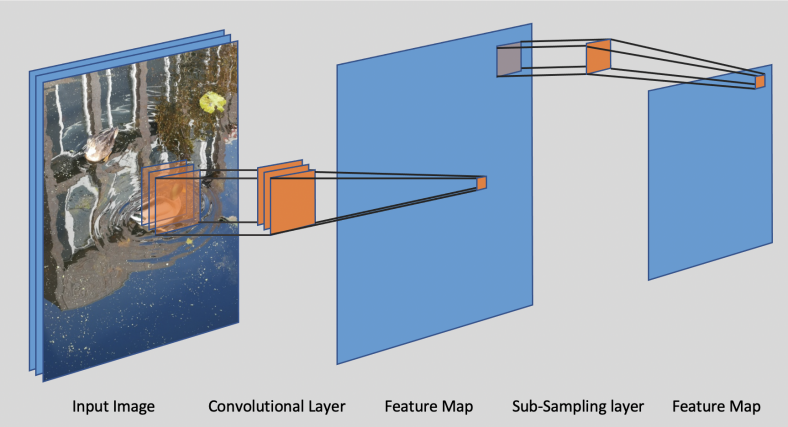


Figure 2.1: Basic CNN architecture

Figure 2.1 illustrates the basic CNN architecture. Convolutional layer and Subsampling layer together form a layer in Convolutional Neural Network. By stacking CNN layers, more abstract features can be extracted at the cost of more computational overhead.

Convolutional Layer: In convolutional layer, a small kernel is slid through the input image. Image patches overlapping with the sliding kernels are weighted summed and added with a bias vector. Certain features will have unique responses after processing. Thus we can see these kernels as feature detectors. To add nonlinearity, results are usually processed by nonlinear functions before being sent to the next layer. There are two advantages of the convolutional layer. The first is that it can reduce the number of parameters during training. Each image patch is processed by detectors with shared weights. When compared with fully connected network, this local connectivity constraint tremendously reduces the number of parameters. Another advantage is that the convolutional layer imports certain levels of distortion and translation invariance. When the input is shifted, the response will also shift with unchanged value. When convolutional layers are stacked after each other, more complex and abstract features can be extracted from the increased receptive field. This abstraction ability brings tolerance to slight distortion and translation.

Figure 2.2 illustrates the convolution action. The kernel marked in orange slide through whole image space and functions as feature detectors. In every step, matrix multiplication is applied on the kernel and the overlapping portion of image. Then the kernel shifts S pixels before being applied the same operation. S is called stride length. To avoid the dimensionality reduction of the output feature map, zero

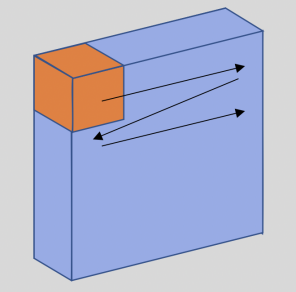


Figure 2.2: Movement of kernel

padding is usually applied before convolution. Assume padding length is P, image size is W*W, kernel size is K*K, then the output feature map will have size O as:

$$O = (W + 2P - K)/S + 1 \tag{2.1}$$

Sub-Sampling layer: Directly feed the outputs of the convolutional layer to the following network will cause heavy computation pressure and will have a higher probability to cause over-fitting problem. Sub-Sampling layers serve as a method to reduce dimensionality and de-noise. On the other hand, the same pooling features will be obtained even if a small translation happened. Max-pooling and Average-pooling are two main types of the pooling methods.

2.2 COMMON TASKS IN COMPUTER VISION

Although this thesis is more closely related to object detection and instance segmentation, four different computer vision tasks, namely image classification, object detection, semantic segmentation and instance segmentation, will be covered due to their close connections. Although goals are different, these four tasks show incremental steps from coarse inference to fine inference[27].

2.2.1 Image classification

Compared to other tasks covered in this section, image classification is the simplest and most basic as it has no further requirements apart from the classification task. Its concept is to extract a set of features which will then be processed and help to label a given image with one or more predefined categories according to confidence scores.

Before the era of deep learning, the most challenging part of this task results from the large amount of intra-class variability. Manually designed features like Gabor features, local binary patterns (LBP), SIFT and HOG can perform well under certain data or tasks[9]. However, the handcrafted features cannot generalize well and be adapted easily to different data sets.

2.2.2 Object Detection

Object Detection goes one step further than image classification. As one of the fundamental problems in computer vision, object detection aims at not only predicting objects' classes but also obtaining their corresponding locations by enclosing objects in rectangular boxes[44]. The common pipeline for object detection, as shown in image 2.3, consists of following three stages:

- **Feature extraction**. It is the most important section because all other tasks are built upon image features. Images are high dimensional signals, and as a

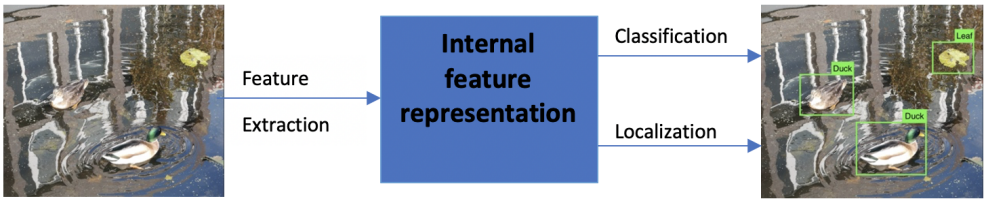


Figure 2.3: Pipeline for object detection. First, feature extractor obtains the appropriate representations for current task. Afterwards, Classification and localization stage take charge of deriving bounding boxes in a compact manner with corresponding class names

result of 'the curse of dimensionality', performance will suffer if other stages are directly run on original image feature space. It is more desirable to extract features that can provide a semantic and robust representation[47]. Thanks to the development of deep learning, CNN extracted features perform better than the traditional ones like Scale-invariant feature transform(SIFT)[47], histograms of oriented gradients(HOG)[14] by a large margin[20][61].

- **Informative region selection**. As only the objects are of value during object detection while a large portion of the image is occupied by background scenery, selecting potential regions can help to filter out redundant information so as to reduce computation stress. Usually, the spatial locations of an object are coarsely represented by rectangle-box-like bounding boxes. The evolution from R-CNN and Fast R-CNN's selective search to Faster R-CNN's Region Proposal Network demonstrates the close connection between detection speed and informative region selection methods.

- **Classification**. Classification is performed based on derived image features from the informative region selection stage. One or more predefined class labels are attached to each region proposals. Like in previous stages, Old school classifiers like the supported vector machine(SVM)[12], AdaBoost[22] and deformable part-based model (DPM)[19] are also outperformed by deep learning based classifiers.

When it comes to the taxonomy of object detection techniques, in general it can be classified into two categories. One is the region proposal-based framework, which generates object proposals at the beginning and then classifies each of them into different categories. Another one tries to obtain object locations and class labels simultaneously with a unified framework. This approach values object detection as a regression or classification problem.

Region proposal-based architectures. This type of methods is also called two-stage object detection as they first perform a coarse scan over images and then focus on the classification of region of interest(ROIs).

- R-CNN: Girshick et al.'s work[25] combines the region proposal generation algorithm with CNN features and thus is named R-CNN. Image 2.4 is the system overview of R-CNN. For every image, about 2000 region proposals are extracted and warped to a common size. CNN feature extractors are applied to each proposal. For the extracted features, several class-specific linear SVMs are used to achieve class labels. On top of the SVM classifiers, a linear regression model is used to re-locate the detection windows' coordinates, which was given by the selective search algorithm, for error reduction purpose. Non-maximum suppression(NMS) is then used to only retain regions with top scores. Several drawbacks exist in R-CNN's design. First of all, although the selective search algorithm has reasonable recall, it can produce redundant region proposals and is time-consuming(2 seconds for 2000 region proposals). Secondly, due to the use of fully connected layer, input image needs to have

fixed size. Moreover, warping proposals may import distortions and incorrect features which would undermine performance. Apart from that, the training procedure is complex as it consists of multiple stages, including Convolutional network, SVM and bounding-box regressors. Last but not least, the whole CNN feature extractor needs to be run individually on each proposal, thus leads to high computation overhead.

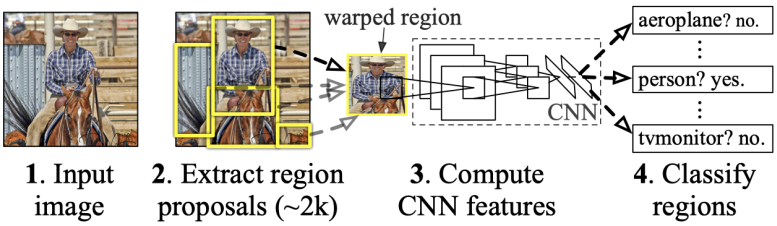


Figure 2.4: System overview of R-CNN, image is from the original paper[25]

- SPP-Net: To solve R-CNN's fixed input size issue and increase scale-invariance, Kaiming et al.[30] proposed Spatial Pyramid Pooling(SPP) network. Image 2.5 illustrates the spatial pyramid pooling method. In contrast to pooling using a sliding window whose output size depends on input image size, spatial pyramid pooling divides the image into several bins and pools from them. As the size of bins is proportional to the image and the number of bins is fixed, the output size has fixed length and can be directly fed to fully connected layer. Another contribution of this architecture is that it only runs convolutional layers once over the entire image and shares the features to all successive layers. When compared with R-CNN, SPP-net yields a speedup of over 100 times.

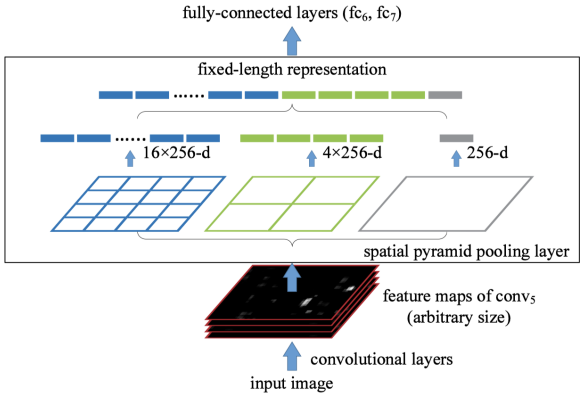


Figure 2.5: Illustration of spatial pyramid pooling layer. Image is from the original paper

- Fast-RCNN: SPPnet didn't improve R-CNN's multi-stage training problem and suffers accuracy loss when the network gets deeper. To solve this issue Fast-RCNN[24] uses a multi-task loss on classification and bounding box regression. The network, except the region proposal section part, has only one unified stage and thus greatly improve training and testing speed. Figure 2.6 gives an overview of Fast-RCNN's architecture. The process of Fast-RCNN starts with passing an entire image to convolutional layers, followed by max pooling to obtain feature maps. Then ROIs generated by algorithms like selective search will be used to extract local features from the output feature map. The extracted features will then be processed by an ROI pooling layers, which serves as a special case of spatial pyramid pooling, to get fixed length feature vectors. After being processed by multiple fully connected layers, the feature vectors will be passed to two different branches. One is used

to produce K+1(number of given categories plus background) softmax probability for class prediction. Another is to regress the bounding boxes. The loss function for Fast-RCNN is as follows:

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v) \tag{2.2}$$

p is a discrete class distribution and u is the ground-truth class distribution. L_{cls} is calculated using a cross entropy loss. t^u is the predicted bounding box for class u and v is the ground truth bounding box. L_{loc} is computed with smooth L1 loss. λ represents a factor to balance the contribution of these classification and regression branches.

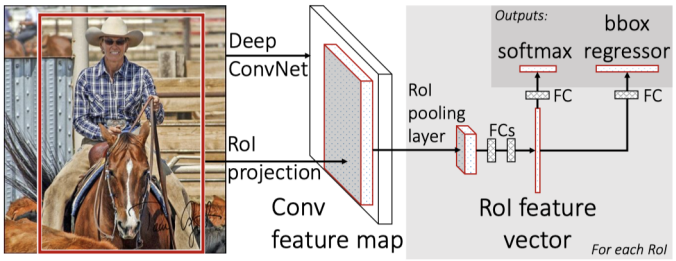


Figure 2.6: Illustration of Fast-RCNN. Image is from the original paper

- Faster R-CNN: SPPnet and Fast-RCNN's success in reducing the running time of detection network leaves region proposal algorithm as the computational bottleneck. To solve this problem, Shaoqing et al.[54] observed that the convolutional features used by region proposal based networks like Fast-RCNN can also be used to generate object proposals. He introduced a region propose network(RPN) that simultaneously predict bounding boxes and objectness score. By merging RPN and Fast-RCNN, the proposed Faster-RCNN imports 'attention' mechanism in the prediction process. Image 2.7 illustrates the process of Region proposal network. This small network acts as a replacement of Selective search algorithm and shares the convolutional features with the detection network. It slides over convolutional features with an intermediate layer and uses the output for two sibling branches. At each sliding window position, multiple proposals with different scale and aspect ratio, which is named 'anchor', are processed simultaneously. Anchor's functionality is similar to image pyramid and serves as references at multiple scales and aspect ratios. RPN uses one branch for anchor box regression and another for anchor box classification. The positive anchor boxes will be further classified and regressed by latter network modules.

Regression/classification-based architectures. Two-stage approaches' training is usually cumbersome due to the combination of multiple correlated stages. Even Faster-RCNN needs to use alternative training so as to share convolutional parameters between RPN and detection network. Regression/classification-based architectures algorithms are usually referred as one-stage object detection as they aim to bring the training of separate stages together as a global regression/classification task.

- YOLO: YOLO[53] is one of the most important breakthroughs among one-stage object detection frameworks. It treats the detection task as a regression problem and runs only one single convolutional network to simultaneously predict bounding boxes and corresponding confidence score. Apart from being fast, unlike sliding window-based techniques, YOLO can encode contextual information about classes and make less false positive predictions on

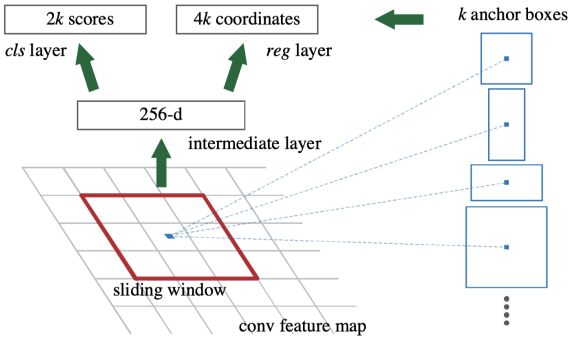


Figure 2.7: Illustration of RPN module from Faster-RCNN paper. Assume k anchors are processed, the classification score is a 2k-dimension vector as at this stage, regions are roughly classified as object or background. the 4k dimension vector from regression branch originates from the coordiantes for object center x, y and width, height. This image is from the original paper.

the background image. Figure 2.8 illustrates the training/testing process of YOLO. It first divides the image into S*S grid cells. A grid cell is responsible for detecting objects whose centers fall into it. Then in each cell, B box locations(x,y,width,height) and their confidence scores are predicted. Last but not least, a set of conditional class probability is predicted regardless of the B bounding boxes. Formula:

$$Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Class_i) * IOU_{pred}^{truth} \quad (2.3)$$

gives the class-specific confidence score for each box. Results are extracted by setting a threshold to filter out low confidence boxes and applying NMS. YOLO shows a trade-off between speed and accuracy. Although being fast, it cannot precisely localize objects, especially small ones.

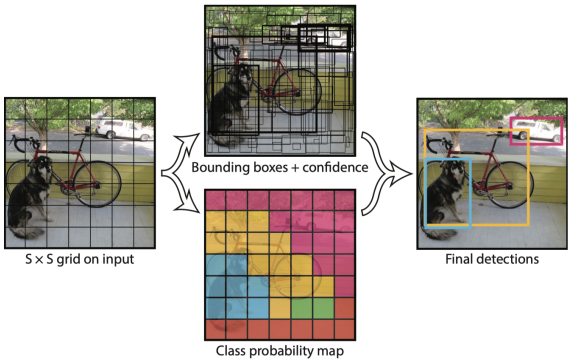


Figure 2.8: Illustration of YOLO network. Prediction is a S*S*(B*5+C) tensor, S,B,C represent number of grids, number of bounding boxes and number of given classes respectively. Image is from the original paper.

2.2.3 Semantic segmentation

Fields like autonomous driving and robotics require to further understand a scene, including object shapes, textures etc. Semantic segmentation is one step to fill the blank. The goal of semantic segmentation is to label every pixel with the class of the object that encloses it. There are mainly two different approaches in this field, namely region-proposal-based approach and FCN(fully convolutional network)-based approach.

- Region-proposal-based approach: The region-proposal-based approach usually starts with region proposal generation. It then classifies the pixels within the region. One representative framework using this approach is RCNN. Different from the object-detection-oriented framework, RCNN for semantic segmentation uses CPMC algorithm to generate proposals. It then encloses the regions in rectangular windows and wraps them to fixed-size square windows. Segmentation task is fulfilled by running a detection network on the square windows and assign the predicted class labels to region proposals enclosed in the windows. However, this method doesn't ignore the convolutional features generated by the background image. So author optimizes the design by feeding both foreground features and full features(background+foreground) to the network. This approach can also be used by RCNN's successor like Fast-RCNN and Faster-RCNN.

- Fully Convolutional Network (FCN): One problem bounded with the region-proposal-based approach is that it can only process fixed-size images due to the use of fully connected layer. Moreover, the wrapping and ROI pooling will cause the loss of spatial details and degrade segmentation accuracy. J.Long et al. [46] proposed a fully convolutional network that can solve this problem. FCN takes an arbitrary size input image and predicts a feature map which is the same size as input. Each pixel within the feature map will be assigned a class label. As it is named, FCN only consists of convolutional layers. To compensate the dimension reduction in output feature maps caused by sub-sampling, author imports deconvolution layers. To further improve the prediction's details, skip layers are introduced so as both low level, fine information and high level, coarse information can be used at the same time. The architecture of FCN is shown in figure 2.9.

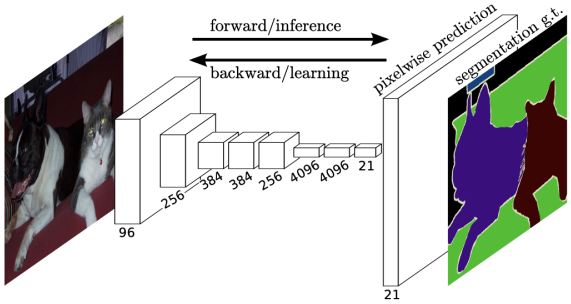


Figure 2.9: Architecture of FCN. Image is from the original paper

Dilated Residual Network[66] is another FCN-based architecture. It tries to avoid the spatial information loss caused by subsampling and uses dilated convolution instead.

Based on FCN's finding, finer predictions can be achieved by passing context information from contracting paths to higher resolution layers in expansive paths. Olaf et al.[56] modified FCN by adding more channels in expansive paths and replace the sum operation with concatenation. The final network structure, which is shown in figure 2.10, is symmetric and looks like a U shape. Thus it is named U-net.

Another variation of FCN-based network is hourglass network[51]. Unlike FCN, which is heavy in bottom-up processing but light in top-down processing, Hourglass Network is more symmetric and similar to U-net. More difference between FCN and hourglass network lies in that hourglass network does not use deconvolution layer, instead it uses nearest-neighbor interpolation to perform upsampling. As for the differences between U-net and hourglass network, U-net uses concatenation in top-down processing while hourglass

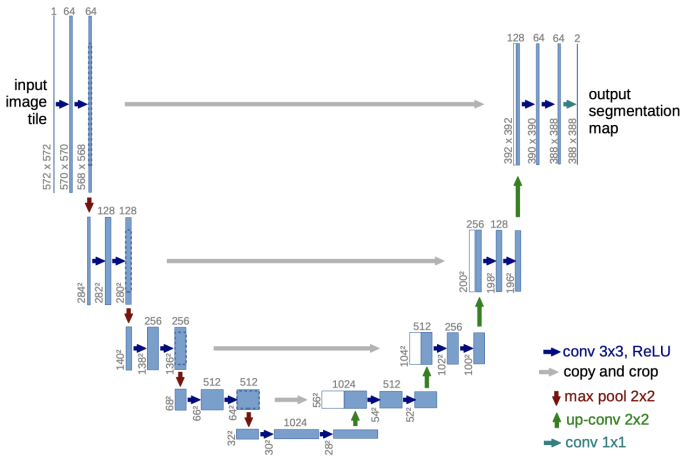


Figure 2.10: Architecture of U-net. Image is from the original paper

uses addition. Hourglass network also uses residual modules instead of simple convolutional layers. More importantly, hourglass network shows great performance increase when stacked together accompanied with intermediate supervision. An hourglass module is shown in figure 2.11

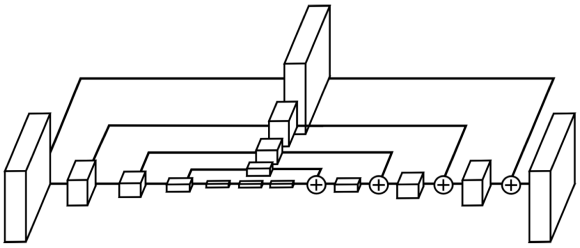


Figure 2.11: Illustration of a single hourglass module. Image is from the original paper

2.2.4 Instance segmentation

Instance segmentation tries to solve object detection and semantic segmentation at the same time. It not only provides the class label for each pixel but can also distinguish the instances. Instance segmentation can also be classified into FCN-based and region-proposal-based frameworks. Region-proposal-based frameworks are usually instance-first strategy, which means detecting instances followed by segmentation. Early work like [28],[10] which utilize the region proposal methods as the requisite all fall in this category.

Another representative network from this type is Mask RCNN. It achieved state-of-the-art performance when it was published. Figure 2.12 illustrates Mask RCNN's architecture. It has an identical first stage(RPN) as Faster RCNN but adds an additional branch in the second stage. This new branch is responsible for binary mask generation. However, different from works like [13], mask prediction branch, bounding box classification and offset prediction branch are run in parallel with no interaction. This design reduces competition among tasks and is proved to have better performance. Mask generation branch produces one binary mask per class, resulting in that the output is Km^2 dimension where K represents the number of given categories and m^2 represents the resolution of ROI. The class label predicted by bounding box classification and offset prediction branch will then be used to filter the masks and obtain the final result. Due to the promising result of RCNN, we choose it as our baseline.

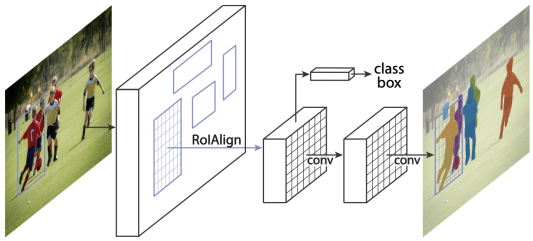


Figure 2.12: Mask RCNN's architecture. Image is from the original paper

While FCN-based networks are usually superior in semantic segmentation tasks, they have some inherent problems when transferred to instance segmentation. The problem of FCN frameworks lies in that convolution is translation invariant. 'Translation invariant' means that the same pixel will get the same response after translation. Thus FCN cannot differentiate two pixels from different objects at different position if they share similar value. One solution is to add translation variant property. Dai et al.[13] introduced a position-sensitive score map to FCN for mask proposal generation, which is followed by a downstream network for detection. Following Dai's work, Li et al.[39] proposed fully convolutional instance-aware semantic segmentation method(FCIS) that shares the underlying convolutional representation and score maps to both object detection and segmentation sub-tasks. FCIS also combines RPN to operate on box proposals instead of sliding window in Dai's work. However, FCIS exhibits errors when it comes to overlapping objects. Another variant in this category is proposal free networks. As generating proposal itself is a challenging task, works like [33][45][42] try to avoid proposal prediction. Starting from per-pixel classification, they obtain instances by cutting the pixels from the same category. Liang et al.[42] proposed Proposal-free Network, which predicts pixel-wise instance location maps followed by clustering technique to generate instance-level segmentation results. Similar work like [32][52] also predict object location(object center) and use clustering methods to obtain instances. Our paper fits in this category but is different at its enhancement on the overlapping scenarios and its robustness to occlusion.

2.3 POLYGONAL OBJECT DETECTION

Early works related to polygonal object detection is a combination of contour extraction and subsequent contour simplification. For contour extraction, methods like level set[60], graph-cut[6] and the more advanced GrabCut[57] solve the problem in a energy minimization fashion. Performance of these methods rely on either color distribution [57][6] or initialization[60], thus are not accurate and require human intervention. Performance of the alternative methods like superpixel grouping[36] or object saliency detection algorithms[11] are also limited by line evidence or image contrast. As for the simplification step, it can be done on Douglas-Peucker algorithm[65]. These multi-step approaches usually yield high accuracy loss in practice.

Another strategy is based on assembling geometrical primitives like line segments in the image into closed polygons. Sun et al.'s work[62] detects polygonal objects by partition a weighted line fragment graph.

Polygonal object detection can also be achieved by grouping the over-segmented polygon cells in an image. Polygonal partition can be achieved by methods like [1][63][15][21][4], while the grouping can be done using aggregation mechanisms [55][41]. This strategy's performance depends on fine polygonal partitions that fit the input image well, which is hard to achieve in practice.

Neural network based polygonal detection is a new research direction with limited works been done so far. Polygon-RNN[8] proposed a semi-automatic approach to annotate object instances. This work, however, requires a human annotator to provide object bounding boxes. It's improved version [2] introduces a graph neural network (GNN) to increase output resolution, but the system remains non-automatic. More recent work like PolyMapper[40] and [38] introduce automatic approaches to directly predict polygon representations. Both works solve polygon detection using a two stage pipeline: keypoint detection and grouping. PolyMapper[40] combines CNNs and RNNs with convolutional LSTM modules, which shows good results on aerial images. While [38] avoids intermediate learning of object boundaries in PolyMapper and uses a fully geometric-based grouping method instead of deep learning modules. Our design falls into this category but is different on both keypoint detection and grouping stages.

2.4 MULTITASK LEARNING

Multitask learning focuses on simultaneously solving multiple tasks from a single input. Usually, multitask learning follows an encoder-decoder structure where the encoder learns the feature representations and share them to successive decoders. The decoders are the task solvers which forms several single-task branches.

It is proved that multitask learning can help to improve individual tasks' performance and reduce the computation time when compared with training tasks separately. The explanation for this benefit is that features learned by one task may help other tasks as well. In Liebel et al.[43]'s work, author verify this concept by importing auxiliary tasks which are irrelevant to the main application. Although unrelated to main tasks, the added auxiliary task does improve the overall performance. Liebel attributes the performance boost to a more robust common representation learned by the encoder. As the network is forced to generalize to more tasks, parameter space is also added with more restrictions during optimization. In other words, tasks act as regularizers to each other, leading to better feature representations.

One problem bounded with multitask learning is the overall loss. As tasks' losses may have different scales, their contributions to the overall loss are not balanced. In early works like [59][34][17], the overall loss function is formed as a weighted sum of all tasks:

$$L_{overall} = \sum_i \omega_i L_i \tag{2.4}$$

Where i represents the i-th task. The weights are set to be uniform or manually tuned using grid search. Tuning weights manually can be time-consuming and may not be able to obtain the optimal combination. In Kendall et al.'s paper[32], the high correlation between performance and weights is discovered. Kendall proposed an idea to use task uncertainty to capture loss weights. The loss function is shown below:

For regression tasks, assume they follow a Gaussian distribution. x,y denote model's input and ground truth label. $f^W(x)$ is the network's output when input is x and weight is W. The likelihood is:

$$p(y|f^W(x)) = N(f^W(x), \sigma^2) \tag{2.5}$$

Scalar σ represents the observation noise. By maximizing likelihood of this model, the loss for a model with two regression tasks is:

$$L(W, \sigma_1, \sigma_2) = \frac{1}{2\sigma_1^2} L_1(W) + \frac{1}{2\sigma_2^2} L_2(W) + log\sigma_1 + log\sigma_2 \tag{2.6}$$

The observation noise σ in loss function 2.6 can be seen as learnable weights for different tasks. A large σ can decrease the loss contribution from a task. In case σ being too large, $log\sigma$ is used as a regularizer to penalize large values. In practice, $log\sigma^2$ is favored more as a regularizer because it is more numerically stable.

Liebel et al.[43] improves loss function 2.6 by replacing the regularizer to $ln(1+\sigma^2)$ so as the regularization values can always be positive. The adapted loss function for multiple tasks is:

$$L_{overall} = \sum_{t\in\tau}[\frac{1}{2\sigma_t^2}L_t + ln(1+\sigma_t^2)], \quad \tau \textit{ is the task set} \tag{2.7}$$

2.5 DBSCAN

Density-based spatial clustering of applications with noise(DBSCAN)[18] is a non-parametric density-based clustering method. It can group geometrically close points and mark points in low-density regions as outliers. The algorithm is described as follows:

hyperparameters for DBSCAN:
1.$minPts$ 2.ϵ
Assume p and q are two different points from the input, DBSCAN classifies all the input points using following rules:

- p is marked as a core point if no less than $minPts$ points are within distance ϵ of it.
- q is directly reachable from p if q is within distance ϵ from core point p. Points are only said to be directly reachable from core points.
- q is reachable from p if the path p_1, ..., p_n exists, where $p_1 = p$ and $p_n = q$ and all p_{i+1} is directly reachable from p_i. The Reachability in DBSCAN is not a symmetric relationship.
- All points that are not reachable from a core point are marked as outliers.

A core point and all other points that are reachable from it together form a cluster.

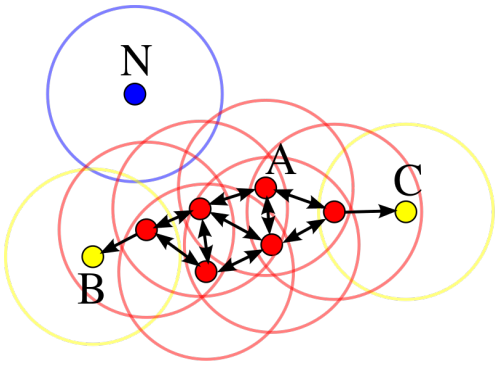


Figure 2.13: Illustration for DBSCAN. Yellow points are reachable from red core points and they together form a cluster. Blue point is an outlier.

An example[1] is shown in image 2.13. We use DBSCAN as it requires no prior knowledge of the number of clusters and its low time complexity. The average time complexity for DBSCAN is $O(n \log n)$.

1 The image is from https://en.wikipedia.org/wiki/DBSCAN

2.6 GRAHAM SCAN

Graham scan[26] is an algorithm that can find a convex hull of a given set of points with time complexity $O(n \log n)$. Given a set of points, Graham scan extracts a convex hull that covers all points using the following Pseudocode:

```
ccw: ccw > 0 if three points make a counter-clockwise turn, clockwise if ccw < 0, and collinear if ccw = 0.

let points be the list of points
let stack = empty_stack()

find the lowest y-coordinate and leftmost point, called P0
sort points by polar angle with P0, if several points have the same polar angle then only keep the farthest

for point in points:
    # pop the last point from the stack if we turn clockwise to reach this point
    while count stack > 1 and ccw(next_to_top(stack), top(stack), point) <= 0:
        pop stack
    push point to stack
end
```

An example of Graham scan is shown in 2.14[2].

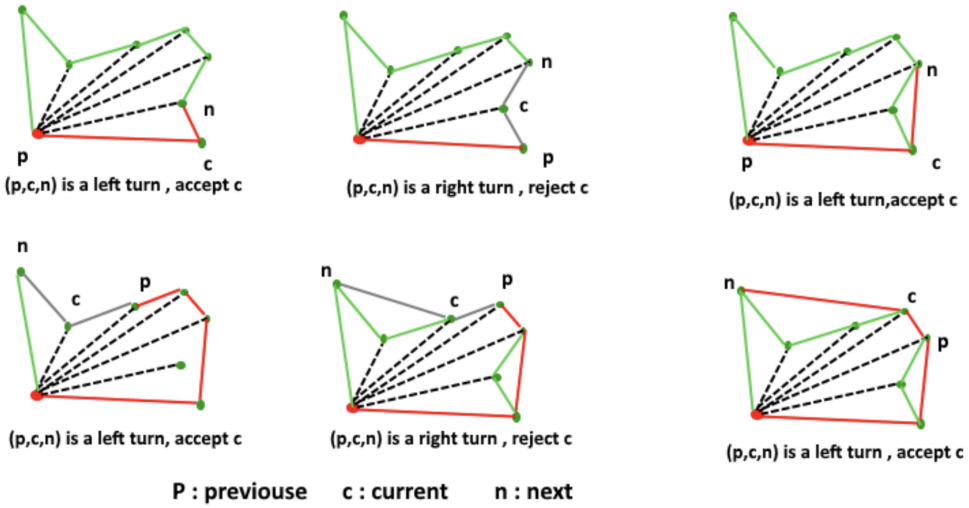


Figure 2.14: Illustration of Graham scan

2 Image is from https://www.geeksforgeeks.org/convex-hull-set-2-graham-scan/

3 DATA

This chapter has three sections. The first section illustrates what dataset we use in our paper and why we use it. Next section provides details of the dataset. The third section provides the pre-processing applied to the dataset. The last section will reveal some flaws of the dataset and pre-processing strategies we applied.

3.1 EXPLANATION FOR USING GATE DETECTION DATASET

Deep neural networks require a large amount of annotated data for training. Due to the polygonal object detection task involves the detection of polygon vertices, we need datasets that accurately annotated object corners. Annotating polygon vertices itself is less cumbersome than annotating masks for regular instance segmentation tasks, but is still time-consuming. Thus we decide to utilize existing datasets. The choices are very limited. One dataset is the crowdAI dataset[49] which consists of 280741 tiles of satellite imagery for training and 60317 images for validation. The dataset is too large to perform experiments considering our limited time. Moreover, satelite dataset does not contain any overlapping objects, thus cannot verify our research question on overlapping scenarios. As a result, we choose another dataset at hand, which is the gate detection dataset. This dataset originates from another TU Delft student's work[16] related to drone racing.

This dataset is synthesized using Epic's Unreal engine[1] and AirSim-Plug-In[2] by Microsoft. As shown in figure 3.1, racing gates are scattered in various environments at different locations with random angles and captured by cameras to create 2D images. All images are of size 416*416 pixels.

As shown in image 3.1c and 3.1d, overlapping occasionally happen in this dataset, which may distract detectors during training if one gate contains another. This overlapping scenario makes gate dataset perfect for us to verify our second research question.

The gates are originally not complex polygons, but different angles of gate directions and the rotation of camera import distortions to target objects(as shown in 3.1b). Another challenge lies in that gates are hollow, which means the majority part of pixels within a gate belongs to the background image. While effective features of a gate are hard to detect due to its thin structure. Thus, training on only one dataset can easily cause overfitting as detectors may take background pixels as a hint for objects.

3.2 DETAILS OF THE DATASET

Based on the consideration that illumination condition can vastly influence object appearance, we select four sub-datasets from gate detection dataset with different illumination. The datasets are exhibits in table 3.1:

1. Basement1: Environment of this dataset is a basement with only weak artificial light source. This dataset is used for training. An example is shown in 3.1a.

1 https://www.unrealengine.com/en-US/what-is-unreal-engine-4
2 https://github.com/Microsoft/AirSim

(a) Basement1 **(b)** Basement3

(c) Daylight **(d)** IROS2018

Figure 3.1: Examples from gate detection dataset. (a) is the training dataset which is set in a basement with low artificial illumination. (b),(c),(d) are used for testing which respectively has low,high,medium illumination condition

2. Basement3: Illumination is similar to Basement1. This dataset is for testing use and is shown in 3.1b.
3. Daylight1: Gates are placed in a room with windows, through which daylight can pass. Windows can cause strong variation between different parts of a gate. As can be seen in 3.1c, the upper parts of gates at the far side can hardly be recognized under intense daylight.
4. Iros1: The environment of this dataset simulates IROS Autonomous Drone Race 2018. Both artificial light source and daylight through windows exist. Illumination condition is in between Basement13 and Daylight1. An example is shown in 3.1d.

We expect the testing results will be high on basement3 due to its similarity with training dataset but can drop sharply when using daylight dataset. The performance on iros1 should lie between daylight and basement3.

3.3 PRE-PROCESSING

First of all, we upsample the image to 512*512 pixels using bilinear interpolation for convenience as in the backbone network the image can be downsampled by 2*2 maxpooling with stride 2 for four times.

For each gate, the annotation consists of the 2D coordinates of 4 gate corners V and the gate center C. However, as shown in image 3.2a, some gate corners are out of image space and thus are not usable for us as we use per-pixel classification for

Dataset details		
Dataset name	number of images	illumination condition
Basement1	2441	Low
Basement3	1407	Low
Daylight1	3412	High
Iros1	3163	Medium

Table 3.1: Details for datasets been used in this paper

vertices detection. We resolve this problem by replacing unusable vertices with the intersections of object boundary and image boundary. Image corners are also used as vertices when necessary. The recalculated ground truth is shown in image 3.2b

(a) Ground truth **(b)** Recalculated ground truth

Figure 3.2: Annotations are recalculated if they are outside image space

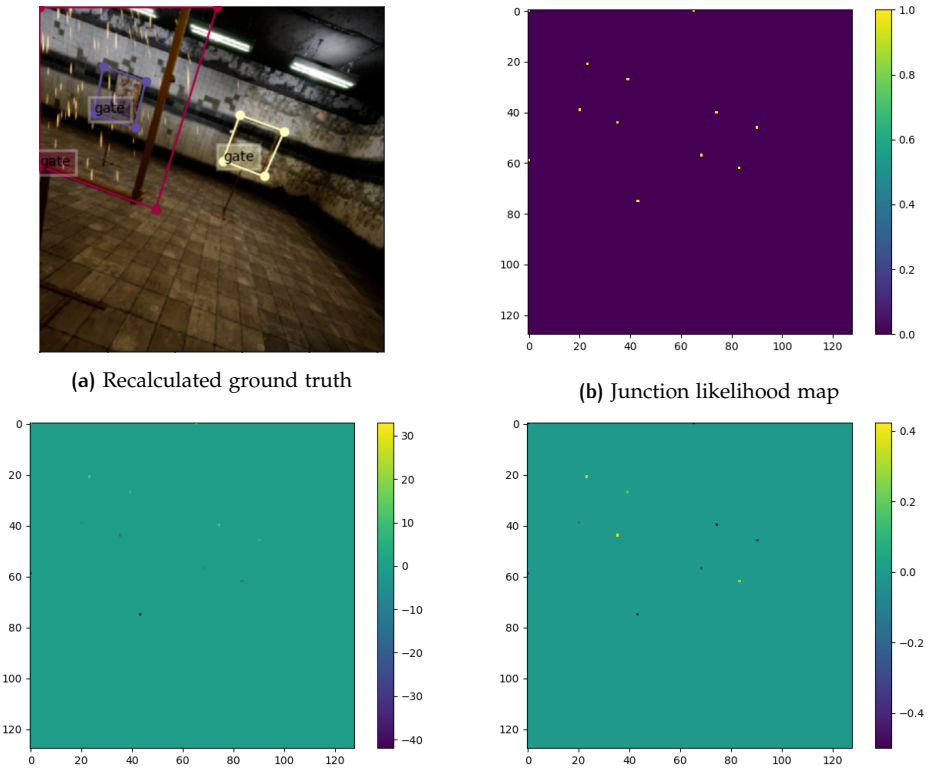
After the recalculation, we generate three maps as target feature maps for different tasks in our network:

1. A **junction likelihood map** for vertices detection use. It is a binary mask where 1 represents a pixel is an object vertex. An example is shown in 3.3b.

2. An **offset map**. In our design, for every vertex, we predict the offset from it to its corresponding object center(centroid). The offset is calculated as $[C_x - V_x, C_y - V_y]$ where the subscript x,y indicates the coordinates on x-axis and y-axis. The offset map has two channels, one is its projection on the x-axis, another is on the y-axis. Offset map on x-axis is shown in 3.3c.

3. A **remapping map**. We predict this map because we perform post-processing on a downsampled image space. Remapping is required to recover detected objects back to the original image space. This remapping process, however, can cause accuracy loss. For example, if one ground truth vertex on a 512*512 input image is $[105, 105]$, after being downsampled to a 128*128 image space, the coordinates of this vertex are rounded to $[26, 26]$. If we detected this vertex on the shrunk image and remap it back, the coordinates will be $[104, 104]$. A one-pixel accuracy loss emerges! Although the accuracy loss is not huge, we still use one branch in our network to compensate it. We use the remapping map for this very task. Similar to the offset map, remapping map also has two channels representing remapping offsets along x-axis and y-axis respectively. A remapping map along the x-axis is shown in 3.3d, values

on remapping maps can be either positive or negative and is bounded by $[1/2, 1/2) \times [1/2, 1/2)$.



(a) Recalculated ground truth

(b) Junction likelihood map

(c) Offset map

(d) Remapping map

Figure 3.3: According to the recalculated object(shown in 3.3a) center and vertices We generate 3 maps as ground truth for different tasks in our network

3.4 FLAWS OF DATASET & PRE-PROCESSING STRATEGY

In image 3.3a or 3.2b, it is noticeable that some vertices are not annotated well. The annotated vertex coordinates can have a shift from where it should be. We decide not to relabel the dataset due to time limitation. The shifts will downplay the performance of the networks trained on the dataset, especially our vertex-voting-based approach.

Another potential problem is related to our pre-processing strategy. As mentioned above, some image corners are used as object vertices so as to form a closed polygon. But as shown in image 3.3a or 3.2b, these image corners contain very little information of the gate. We conjecture that those recalculated vertices can distract our network to use the features from background image for gate detection and undermine the generalization ability. However, we don not have a better solution.

4 METHODOLOGY

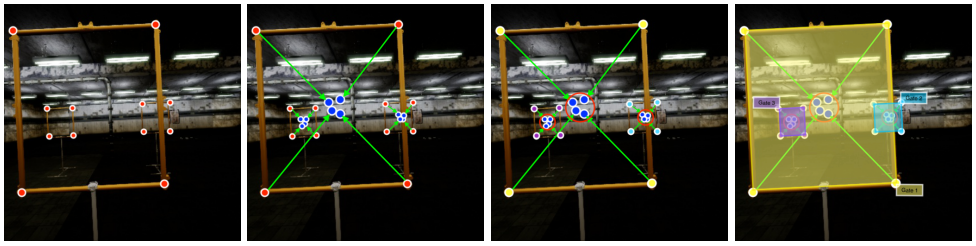
In this section, we will illustrate our framework in detail. In the first section, we will provide a general explanation about how to process object detection in a polygon detection fashion. The second section will show the detailed network design. Then we illustrate the post processing stage in the third section.

4.1 GENERAL PIPELINE

Similar to [42][32][52], we propose a voting based proposal-free network for object detection. Our framework has two stages. The first stage uses a neural network to predict several feature maps which will then be fed to the second stage. The second stage is a post-processing stage that infers target objects based on the input feature maps. The overall pipeline is as follows:

> **Extract object vertices ⇒ Each vertex votes for its object center ⇒ Group vertices that have geometrically close object centers ⇒ Link grouped vertices in certain sequence to form final polygon representation**

Figure 4.1 illustrates this process with an example input image from the gate detection dataset.



(a) 1. vertices prediction **(b)** 2. center voting **(c)** 3. vertices grouping **(d)** 4. final result

Figure 4.1: Illustration of the overall pipeline of our approach.

4.2 NETWORK DESIGN

4.2.1 Multi-task learning

Our post-processing strategy requires a junction likelihood map for vertices detection, an offset map for object center prediction, a remapping map for accuracy compensation during re-scaling image and another feature map which is used to deal with overlapping scenarios. This requires that our neural network can learn multiple objectives with high efficiency. Considering Multi-task learning's ability to reduce computation run-time by sharing representations, and its ability to improve accuracy due to the regularization effect between different tasks, we propose a multi-task CNN architecture based on [43] and [32].

Due to the computation pressure brought by high-resolution images, we first downsample the input from 512*512 pixels to 128*128. Then a backbone network

is used for feature extraction. The feature representations will be shared by four different branches, as shown in figure 4.2. In figure 4.3, we show an example of the network's output after feeding an image from the gate detection dataset.

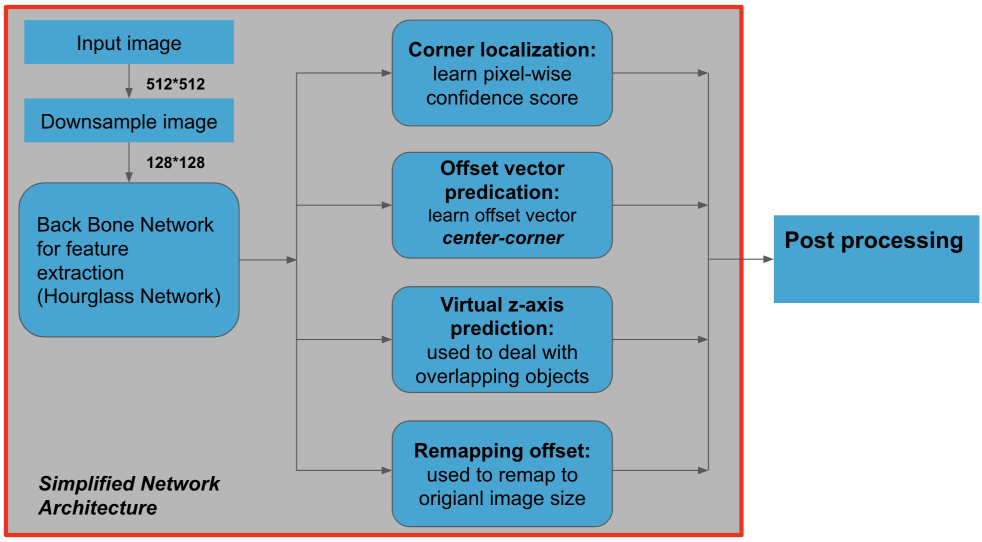


Figure 4.2: An overview of our network's architecture

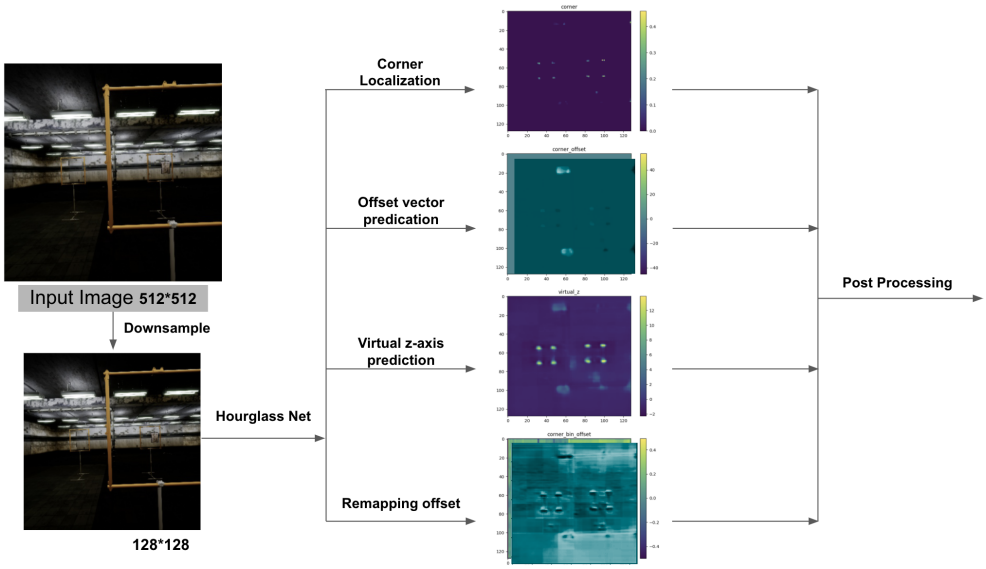


Figure 4.3: An Example of our network's output

4.2.2 Feature extractor

Multi-task learning is usually implemented in an encoder-decoder style. A common encoder extracts feature representations for the successive decoders. The decoders solve their respective tasks individually and form several single-task branches which are attached to the encoder.

We follow this concept and use the hourglass network as our feature extractor. This is due to the issue embedded with CNN. As CNN uses a layer-by-layer hierarchy to extract features, it forms an inbuilt multi-scale pyramid inexplicitly. Thus higher layers tend to have more abstract features that are robust to variation in translation or deformation but lack of low-level information due to resolution reduction. In comparison, lower layers that have smaller receptive fields contain more details

but are ignorant of global information. As we are forcing our network to learn an offset pointing from vertices to object centroids and localize the vertices at the same time, both fine low-level local information and coarse high-level global information are needed. Hourglass Network's skip layer provides a solution to obtain both of them at the same time. The hourglass module we use is identical to [51].

4.2.3 Decoder branches

All of the decoders use identical structure. They share the same feature representations from the backbone hourglass network and predict their respective feature maps with several fully convolutional layers. The only differences between them are the number of output channels and the attached activation functions.

- **Corner localization branch**: In order to obtain the probability that a pixel belongs to a certain category, this branch performs pixel-wise classification. It predicts $M + 1$ channels where M is the number of given classes in a dataset and 1 means background. In our case, M=1 as we only have the gate class in gate detection dataset. We use a Softmax function on the output to normalize the probability distribution. The result will be used to extract vertices in the post-processing stage.

- **Offset vector prediction branch**: For every vertex, we predict its offset to object center along x-axis and y-axis separately. Thus the output contains two channels. As the offset can be either positive or negative, we use hyperbolic tangent function(Tanh) as the activation function. Moreover, due to the ground truth offsets can be very large values while the range of Tanh's outputs is limited within range [-1,1], we rescale the output by multiplying its 2 channels with image width and image height respectively.

- **Virtual z-axis prediction branch**: As illustrated in section 3, overlapping can occur in the gate detection dataset. If two overlapping gates have geometrically close object centers, the voting&grouping strategy cannot distinguish them in 2D space. That is why we import this branch to predict an additional axis. We train this branch to predict different values for pixels if they are from gates with different distances to the camera. Although we conjecture that by using a triplet loss, this branch can learn object depth information, the learning result is not guaranteed to reveal this in a comprehensive way. Thus we call it virtual z-axis. The concept of this branch is similar to Associative Embedding method introduced by Newell et al.[50]. The output of this branch has only one channel.

- **Remapping offset branch**: In this branch, a two-channel feature map is predicted by this decoder. It serves to predict the decimal values that have been rounded and lost when we downsample the image. Values from this branch will be added to detection results before we rescale to original image space. Due to the range of offset should be limited to $[\frac{-1}{2}, \frac{1}{2}) * [\frac{-1}{2}, \frac{1}{2})$, We apply a sigmoid activation function and add a -0.5 shift to the output feature map.

A decoder may favor certain feature representations learned by the encoder, while some of them may also be exploited by other decoders. Thus this multi-task fashion joint learning is believed to be able to boost the overall performance[43].

4.2.4 Intermediate supervision

Inspired by [51],[64] and [7], we use an iterative prediction architecture and import intermediate supervision. The total loss is the sum of the losses at all iterations.

4.2.5 Loss

We use different loss functions for different branches. We use an average cross entropy loss for object vertices prediction. Smooth L1 loss is used for offset vector prediction as it is less sensitive than L2 loss and in some cases can prevent exploding gradients. We also use smooth L1 loss in the remapping offset branch. For virtual z-axis prediction branch, we use a margin-based triplet loss to excavate the underlying depth information of objects. If two pixels are from the same object, they should contain similar depth information and will be projected in the near-by region on the virtual z-axis. In contrast, pixels which are from objects with different depth will be pushed further from each other. The triplet loss function we use is as follows:

$$L(a, p, n) = max(0, D(a, p) - D(a, n) + margin) \tag{4.1}$$

where a means anchor, p represents positive and n is negative. function $D(a, p)$ and $D(a, n)$ calculate the L2 distance between samples. This function tries to make the distance between anchor and positive smaller than the distance between anchor and negative by a margin. For every image, we calculate the loss by summing below losses and taking the average:

- $L(background\ pixel\ a, background\ pixel\ b, gate\ pixel)$
- $L(gate\ pixel\ a, gate\ pixel\ b, Another\ gate's\ pixel)$
- $L(gate\ pixel\ a, gate\ pixel\ b, background\ pixel)$

During training, we discovered huge class imbalance. The majority of pixels belongs to the background while only a few pixels are the positive gate vertices. This imbalance may cause the network to emphasize more on the correct classification on less challenging negatives and lead to sub-optimal results. We address this problem by introducing a weight factor on corner localization branch, offset vector prediction branch and remapping offset branch. The factor is set to be the inversed class frequency.

We noticed that in our experiment, losses from different branches have diverse scale. Instead of balancing all tasks' loss contributions with fixed weight factors, we follow [32] and [43] and use learnable weights to our network. The total loss of our network is as follows:

$$L_{total} = \sum_{s \in S} \sum_{t \in \tau} [\frac{1}{2\sigma_t^2} L_{st} + ln(1 + \sigma_t^2)] \tag{4.2}$$

where S represents the number of stages from intermediate supervision. τ is the task set in our model.

4.3 POST PROCESSING

In this section, we continue to use the input image shown in figure 4.3 to illustrate the post-processing method and show it in figure 4.4. Post-processing stage takes all output feature maps predicted by neural network as input and uses the following procedures to infer objects.

1. **Extract vertices**: As mentioned above, the vertices prediction branch will predict a $(M + 1)$-channel feature map where M represents the number of given categories in the dataset. From the generated feature map, we extract N pixels with the highest objectness score(calculated by *1 - the probability for a corner being background*) as candidate vertices. Then, we label every candidate vertex with the class that has the highest confidence score at this location.

2. **Vote for object center**: Assume the coordinates of the i-th candidate vertex are $[X_i, Y_i]$, and the feature map generated by offset vector prediction branch has O_{x-axis} and O_{y-axis} at location$[X_i, Y_i]$, for every candidate we calculated the voted object center by:

$$[X_i + O_{x-axis}[X_i][Y_i], Y_i + O_{y-axis}[X_i][Y_i]]$$

3. **Cluster the voted object centers**: We first separate candidate vertices according to their class labels. Then we perform a simple clustering algorithm on candidates that belong to the same category. The clustering algorithm we use is DBSCAN, which is a density-based clustering method. DBSCAN will group geometrically close object centers together. By remapping the grouped centers back to their voters, we know which vertices are grouped together.

4. **Cluster along virtual-z axis**: As shown in **??**, overlapping objects' centers can be very close. Thus vertices from different objects may be grouped together in step 3. To separate them, we perform DBSCAN the second time, using the learned depth information to distinguish the vertices.

5. **Infer polygon representations**: By far, we already grouped vertices from the same objects together. However, to provide the vectorized polygon representation, we need to link vertices in the correct order. We use Graham scan algorithm to obtain the vector representation of a convex hull which covers all the grouped vertices. If the target object is also in a convex polygon shape, like the gates in the gate detection dataset, then the convex hull can approximate the target object well. For the detected objects, its class label is assigned the same as its vertices' label. And the confidence, which is used to filter out weak predictions, is the mean of all vertices' objectness scores. In other words, vertices vote for object locations, their classes, and their confidence scores.

6. **Rescale to original image space**: As mentioned in the previous section, we downsampled the image four times. To compensate for the accuracy loss, we add the predicted remapping offsets to candidates' coordinates before rescaling them to original image space.

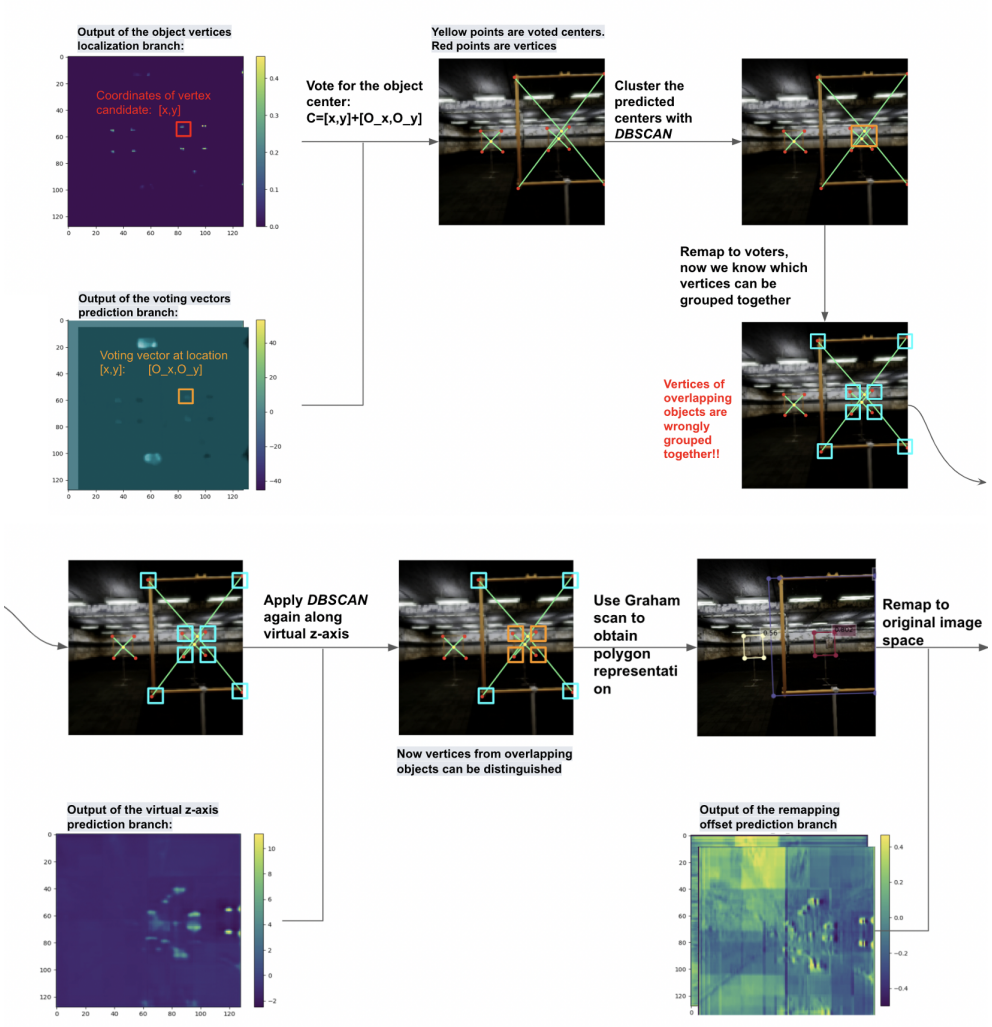


Figure 4.4: Illustration of post-processing strategy

5 EXPERIMENTS

5.1 IMPLEMENTATION DETAILS

For our multitask learning neural network, we implement the encoder following Zhou et al.'s design[67]. We start by applying brightness augmentation and rotation augmentation on the dataset. An input image will be first processed by a 7×7 stride-2 convolution. Then three residual blocks accompanied with stride-2 max-pooling are applied. By far, the image is downsampled to 128×128 with channel=256. Two stacked hourglass modules further process the intermediate features and send the extracted representations to four successive branches. The depth of hourglass is set to be 4. As for the four single-task branches, a 3×3 convolutional layer and a 1×1 convolutional layer are used with ReLU as activation function between them. After being converted to the target dimension by the 1×1 convolution, each branch applies different activation function as demonstrated in the previous section.

The loss function is a combination of losses from all individual branches and the intermediate supervision. A factor that equals 0.0005 is multiplied to the loss generated by negative background pixels to solve the class imbalance problem. Then we introduce four learnable weights to balance each branch's contribution following [43]. We use Adam as the optimizer and initialize the learning rate as 4×10^{-4}. Weight decay is set to be 1×10^{-4}. Identical to Zhou et al.'work[67], we reduce the learning rate by 10 after 10 epochs.

In the post-processing stage, we extract 50 vertex candidates with highest objectness scores after applying Non-maximum suppression(NMS) with a 3×3 max-pooling. When DBSCAN is used in 2D image space, we set its hyperparameters as epsilon=10 and minPts=1. For the second time when we apply DBSCAN on virtual z-axis, we use epsilon=1 and minPts=1. After that, a confidence threshold that equals 0.5 is used to filter out instances with low confidence.

We train the network on two NVIDIA GTX 2080Ti GPUs for 100 epochs. Batch size is set to be 10 to maximize GPU memory usage. The training takes about a day to finish.

5.2 EVALUATION METRICS

We choose a subset of the evaluation metrics used by COCO, namely AP,AR,$AP^{IoU=.50}$, and $AP^{IoU=.75}$. Average precision, also referred as mean average precision(mAP), is considered as the most important metric in COCO. Before demonstrate it, two metrics need to be introduced.

$$\textbf{precision} = \frac{number\ of\ true\ positives}{number\ of\ true\ positives + number\ of\ false\ positives}$$

$$\textbf{recall} = \frac{number\ of\ true\ positives}{number\ of\ true\ positives + number\ of\ false\ negatives}$$

If the Intersection over Union(IOU) between a predicted object and the ground truth object is larger than a threshold, this object is considered to be correctly detected.

The number of true positives in the previous formula means the number of objects correctly detected. False positives are those wrong predictions with no corresponding ground truth. The false negatives are ground truth objects that have not been detected. Thus, precision is a measurement of how many of the detected objects are correct detections. While recall measures how many objects can be extracted among all the objects.

As the number of true positives, false negatives and false positives are dependent on the IOU threshold, In COCO, AP and AR are averaged over multiple IOU thresholds to reward detectors with better localization ability. The AR we use is calculated using ten IOU thresholds ranging from 0.5 to 0.95 at a 0.05 interval. We also use AP^{50} and AP^{75}, which means the AP calculated when threshold equals 0.5 or 0.75. AR^{10}, which means the recall given 10 detections per image, is also used to measure the performance. Both AR and AP are measured using mask IOU.

5.3 COMPARISON TO MASK R-CNN

We compare our network with the state-of-the-art method Mask R-CNN[29], which is implemented by Facebook Research team[1]. The Mask R-CNN uses ResNet50 + FPN as backbone with scale jittering and horizontal flipping as augmentation methods. Compared with our design with only 10,475,790 parameters, Mask R-CNN is implemented with 43,918,038 parameters, almost 4 times as many as ours. We train both networks on Basement1 dataset using two NVIDIA GTX 2080Ti GPUs for 100 epochs. Batch size is set to maximize GPU memory usage. The training takes about a day to finish.

We test both networks on three datasets with different illumination conditions. During inference, Mask R-CNN uses 0.055 s/image on average while our design uses 0.085 s/image in neural network part, and 0.006 s/image in post-processing section. This shows that our post-processing method will not bring too much computation pressure when the number of vertex candidates is low. The test results are shown in table 5.1. After being trained on a dataset with low illumination condition, both networks experience gradual performance drop when tested on datasets with medium and strong illumination. Mask R-CNN performs better than our framework on all datasets. However, we suspect that Mask R-CNN's advantage is not gained from learning more robust feature representations of gates. As racing gates are hollow objects, the majority pixels within a gate belong to the background image. Compared with the vertex-voting-based method whose result is obtained from grouping vertices, Mask R-CNN may also take background pixels as a hint of objects. Thus, although illumination changes, the similar indoor scenery can still provide a performance boost for Mask R-CNN.

(a) Sky texture **(b)** Lava texture **(c)** Brick texture

Figure 5.1: Examples of texture padding on gate detection dataset.

To validate our idea, we pad three textures inside the racing gates. The three textures are sky, lava and stone brick, which are irrelevant to the indoor scenery

1 https://github.com/facebookresearch/detectron2

in gate detection dataset. Examples of texture padding are shown in figure 5.1. The padding experiment, although may block some overlapping objects, is fair to both networks. We test both networks on the padded datasets and show the results in table 5.2. In general, the voting-based method is slightly better than Mask R-CNN. One exception is on the padded daylight1 dataset where Mask R-CNN still has advantages. We attribute the failure on daylight1 dataset to missing vertices. As shown in image 4.1d, some object vertices are hard to recognize even for human due to the intense light. As a gate only consists four vertices, Losing even one vertex will cause significant IOU reduction between predicted objects and the ground truth. We expect to ease this problem by predicting more vertices, which is one of the future directions.

Based on these two experiments, vertex-voting-based approach appears to focus more on the learning of object features while Mask R-CNN tends to import more context information during prediction. It reveals that our voting-based approach is more robust to context changes and occlusion scenarios.

Dataset	Basement3				Daylight1				Iros1			
Metrics	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR
Mask R-CNN	**69.0**	**89.8**	**82.5**	**72.5**	**31.9**	**53.9**	**37.2**	**37.6**	**63.9**	**84.3**	**74.9**	**68.2**
Ours	64.8	82.8	77.2	69.6	21.6	37.8	24.9	32.6	54.8	75.1	66.6	62.7

Table 5.1: Evaluation results on gate detection dataset

Dataset	Basement3				Daylight1				Iros1			
Metrics	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR
Mask R-CNN	55.2	**77.7**	67.1	58.7	**22.0**	**38.8**	**23.2**	**27.1**	43.9	62.8	52.6	48.0
Ours	**59.6**	75.2	**70.5**	**64.7**	15.8	27.0	17.8	24.3	**44.0**	**58.1**	**53.4**	**49.5**

Table 5.2: Evaluation results on the paded gate detection dataset

5.4 ABLATION STUDY

In this section, we study our proposed network by comparing it with two of its variants. The results are shown in 5.3.

Virtual z-axis prediction branch: To validate the efficiency of virtual z-axis, we train another network which shares the identical design with our proposed method except for the virtual z-axis prediction branch. In general, the performance of our design is higher by a small margin. The exception on daylight dataset can also be attributed to the loss of vertices due to intense illumination. In daylight1 dataset, many object vertices are not recognizable. By applying an additional clustering along virtual z-axis, the grouped vertices will be further split into smaller groups, leading to an insufficient number of points to form closed polygons.

Interpretable object detection approach: We force the network to solve object detection in an interpretable way. The network is trained to first look for potential object centers, followed by distinguishing overlapping objects using depth information. We compare our design with a network which is trained to directly vote in 3D space. The only constraint for this variant is that vertices from the same object should vote for geometrically close location. As shown in table 5.3, interpretable approach has much better performance. We conjecture it is related to the constraints imported by interpretable approach during optimization. We haven't tested the directly voting network on higher dimension, because it is proved that if a network can successfully predict high-dimensional embeddings, it should also be possible to project them in lower dimensions as long as it has enough network capacity[50].

Dataset	Basement3				Daylight1				Iros1			
Metrics	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR	AP	AP_{50}	AP_{75}	AR
Ours	**64.8**	**82.8**	**77.2**	**69.6**	21.6	37.8	24.9	32.6	**54.8**	**75.1**	**66.6**	**62.7**
No virtual z-axis	64.2	82.1	76.4	69.2	**23.9**	**42.2**	**27.3**	**33.5**	52.4	71.2	63.8	59.1
3D voting	26.2	34.4	32.2	31.6	2.2	3.8	2.5	4.2	19.6	28.4	23.0	25.7

Table 5.3: Ablation study. We compare our original design with two variants of our network.

6 CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In this paper, We have proposed an vertex-voting-based approach which is able to extract objects' vectorized polygon representations. We have shown that by importing a triplet loss, voting based method can improve its ability to distinguish overlapping objects. We have also shown that forcing network to learn an interpretable solution is beneficial to its performance due to the imported constraints during optimization. Although our approach is not better than the state-of-the-art in all cases, it still shows its potential to occlusion and background context changes.

6.2 ANSWER TO RESEARCH QUESTIONS

- **How to avoid explicit pixel-wise labeling and provide more abstract representations for objects**:

  We found that the polygon representation is an effective approximation for man-made objects. So we transfer the object detection problem to polygon detection.

  Polygonal objects can be seen as a composition of vertices. Thus we see the first step of obtaining polygon representations as extracting object vertices. It is found to be a workable solution to extract vertices using a junction likelihood map.

  We found it is possible not to use region proposal network but use a voting method to localize objects. Moreover, object centers are found to be good indications of objects and can be used to train voting vectors.

  We also found that using traditional machine learning algorithms is able to extract the polygon representation. And the time cost is not high by choosing low time-complexity algorithms.

  In summary, a voting-based approach is proved to be able to solve this problem and still have enough room for improvement.

- **How to deal with the overlapping or occlusion scenarios as they are rarely considered in other polygonal detection works**:

  Overlapping or occlusion are rarely seen in aerial images and was seldom considered in polygonal object detection. We found that occlusion scenarios can be solved by using a voting method. The vertex voters are object parts. By inferring an object based on these object parts, it can in turn force the network to focus more on object features rather than context features. It is proved in our experiment that this method is more robust to environment changes or occlusion scenarios.

  For the overlapping problem, it is not solvable by the voting method in a 2D space if two objects have very close centers. We found that, although the annotation of depth information is not provided, it can still be extracted by

using a triplet loss. The extracted depth information is proved to be useful for dealing with overlapping objects.

6.3 DRAWBACKS AND FUTURE WORK

There are two major drawbacks in our framework which should be improved in future research:

- **Vertices extraction:** Although object vertices are usually object corners and are considered to have better localization property than normal points, we still found that extracting vertices can be problematic when illumination condition changes. This problem is not eased by illumination augmentation as some vertices are even hard to recognize by human. As polygons are abstract representations which are composed of only several points, losing vertices can significantly deform the polygon and cause a huge performance drop. One potential solution to be verified in the future is increasing the polygon representation's complexity. By importing more points, losing a small amount of them will not affect performance too much. We expect the points on object boundaries to be a good choice for this idea.
- **Post-processing:** Although the time cost for post-processing is much lower than the neural network part when the number of candidate vertices is low, it still needs to be improved as the selection of optimal hyperparameters is hard. We currently do not have a workable neural-network-based solution in mind for vertices grouping. As for the Graham scan, which can only provide convex polygons, is more prone to cause performance loss if objects are in non-convex shapes. One potential solution is to use LSTM to replace Graham scan for vertices linking.

BIBLIOGRAPHY

[1] Achanta, R., and Susstrunk, S. Superpixels and polygons using simple non-iterative clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 4651–4660.

[2] Acuna, D., Ling, H., Kar, A., and Fidler, S. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2018), pp. 859–868.

[3] Bastani, F., He, S., Abbar, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Madden, S., and DeWitt, D. Roadtracer: Automatic extraction of road networks from aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 4720–4728.

[4] Bauchet, J.-P., and Lafarge, F. Kippi: Kinetic polygonal partitioning of images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 3146–3154.

[5] Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence 35*, 8 (2013), 1798–1828.

[6] Boykov, Y. Y., and Jolly, M.-P. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001* (2001), vol. 1, IEEE, pp. 105–112.

[7] Cao, Z., Simon, T., Wei, S.-E., and Sheikh, Y. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 7291–7299.

[8] Castrejon, L., Kundu, K., Urtasun, R., and Fidler, S. Annotating object instances with a polygon-rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 5230–5238.

[9] Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., and Ma, Y. Pcanet: A simple deep learning baseline for image classification? *IEEE transactions on image processing 24*, 12 (2015), 5017–5032.

[10] Chen, Y.-T., Liu, X., and Yang, M.-H. Multi-instance object segmentation with occlusion handling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3470–3478.

[11] Cheng, M.-M., Mitra, N. J., Huang, X., Torr, P. H., and Hu, S.-M. Global contrast based salient region detection. *IEEE transactions on pattern analysis and machine intelligence 37*, 3 (2014), 569–582.

[12] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[13] Dai, J., He, K., and Sun, J. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 3150–3158.

[14] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (2005), vol. 1, IEEE, pp. 886–893.

[15] Duan, L., and Lafarge, F. Image partitioning into convex polygons. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3119–3127.

[16] Duernay, P. Detecting empty wireframe objects on micro-air vehicles.

[17] Eigen, D., and Fergus, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 2650–2658.

[18] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (1996), vol. 96, pp. 226–231.

[19] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence 32*, 9 (2009), 1627–1645.

[20] Fischer, P., Dosovitskiy, A., and Brox, T. Descriptor matching with convolutional neural networks: a comparison to sift. *arXiv preprint arXiv:1405.5769* (2014).

[21] Forsythe, J., Kurlin, V., and Fitzgibbon, A. Resolution-independent superpixels based on convex constrained meshes without small angles. In *International Symposium on Visual Computing* (2016), Springer, pp. 223–233.

[22] Freund, Y., and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences 55*, 1 (1997), 119–139.

[23] Fukushima, K., and Miyake, S. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition 15*, 6 (1982), 455–469.

[24] Girshick, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1440–1448.

[25] Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 580–587.

[26] Graham, R. L. An efficient algorithm for determining the convex hull of a finite planar set. *Info. Pro. Lett. 1* (1972), 132–133.

[27] Hafiz, A. M., and Bhat, G. M. A survey on instance segmentation: state of the art. *International Journal of Multimedia Information Retrieval* (2020), 1–19.

[28] Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. Simultaneous detection and segmentation. In *European Conference on Computer Vision* (2014), Springer, pp. 297–312.

[29] He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2961–2969.

[30] He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence 37*, 9 (2015), 1904–1916.

[31] Hubel, D. H., and Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology 160*, 1 (1962), 106.

[32] Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 7482–7491.

[33] Kirillov, A., Levinkov, E., Andres, B., Savchynskyy, B., and Rother, C. Instancecut: from edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 5008–5017.

[34] Kokkinos, I. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 6129–6138.

[35] LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems* (1990), pp. 396–404.

[36] Levinshtein, A., Sminchisescu, C., and Dickinson, S. Optimal contour closure by superpixel grouping. In *European Conference on computer vision* (2010), Springer, pp. 480–493.

[37] Li, M., Lafarge, F., and Marlet, R. Approximating shapes in images with low-complexity polygons. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 8633–8641.

[38] Li, Q., Mou, L., Hua, Y., Sun, Y., Jin, P., Shi, Y., and Zhu, X. X. Instance segmentation of buildings using keypoints. *arXiv preprint arXiv:2006.03858* (2020).

[39] Li, Y., Qi, H., Dai, J., Ji, X., and Wei, Y. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2359–2367.

[40] Li, Z., Wegner, J. D., and Lucchi, A. Topological map extraction from overhead images. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 1715–1724.

[41] Li, Z., Wu, X.-M., and Chang, S.-F. Segmentation using superpixels: A bipartite graph partitioning approach. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), IEEE, pp. 789–796.

[42] Liang, X., Lin, L., Wei, Y., Shen, X., Yang, J., and Yan, S. Proposal-free network for instance-level object segmentation. *IEEE transactions on pattern analysis and machine intelligence 40*, 12 (2017), 2978–2991.

[43] Liebel, L., and Körner, M. Auxiliary tasks in multi-task learning. *arXiv preprint arXiv:1805.06334* (2018).

[44] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. Deep learning for generic object detection: A survey. *International journal of computer vision 128*, 2 (2020), 261–318.

[45] Liu, S., Jia, J., Fidler, S., and Urtasun, R. Sgn: Sequential grouping networks for instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 3496–3504.

[46] Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3431–3440.

[47] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision 60*, 2 (2004), 91–110.

[48] Marcos, D., Tuia, D., Kellenberger, B., Zhang, L., Bai, M., Liao, R., and Urtasun, R. Learning deep structured active contours end-to-end. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 8877–8885.

[49] Mohanty, S. P. Crowdai dataset, https://www.crowdai.org/challenges/mapping-challenge/dataset_files.

[50] Newell, A., Huang, Z., and Deng, J. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in neural information processing systems* (2017), pp. 2277–2287.

[51] Newell, A., Yang, K., and Deng, J. Stacked hourglass networks for human pose estimation. In *European conference on computer vision* (2016), Springer, pp. 483–499.

[52] Qi, C. R., Litany, O., He, K., and Guibas, L. J. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 9277–9286.

[53] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 779–788.

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

[55] Ren, Z., and Shakhnarovich, G. Image segmentation by cascaded region agglomeration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 2011–2018.

[56] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241.

[57] Rother, C., Kolmogorov, V., and Blake, A. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG) 23*, 3 (2004), 309–314.

[58] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature 323*, 6088 (1986), 533–536.

[59] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229* (2013).

[60] Sethian, J. A. Level set methods, evolving interfaces in geometry, fluid mechanics comuputer vision, and materials sciences. *Cambridge Monographs on Applied and Computational Mathematics, 3* (1996).

[61] Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2014), pp. 806–813.

[62] Sun, X., Christoudias, C. M., and Fua, P. Free-shape polygonal object localization. In *European Conference on Computer Vision* (2014), Springer, pp. 317–332.

[63] Von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence 32*, 4 (2008), 722–732.

[64] Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. Convolutional pose machines. computer vision and pattern recognition (cvpr). In *2016 IEEE Conference on* (2016), vol. 2.

[65] Wu, S.-T., and Marquez, M. R. G. A non-self-intersection douglas-peucker algorithm. In *16th Brazilian symposium on computer graphics and Image Processing (SIBGRAPI 2003)* (2003), IEEE, pp. 60–66.

[66] Yu, F., Koltun, V., and Funkhouser, T. Dilated residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 472–480.

[67] Zhou, Y., Qi, H., and Ma, Y. End-to-end wireframe parsing. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 962–971.

A | APPENDIX

A.1 ABBREVIATIONS

AP: Average precision

AR: Average recall

CNN: Convolutional Neural Network

DBSCAN: Density-based spatial clustering of applications with noise

FCN: Fully convolutional network

FCIS: fully convolutional instance-aware semantic segmentation

NMS: Non-maximum suppression

ROI: Region of interest

RPN: Region propose network

SPP: Spatial Pyramid Pooling

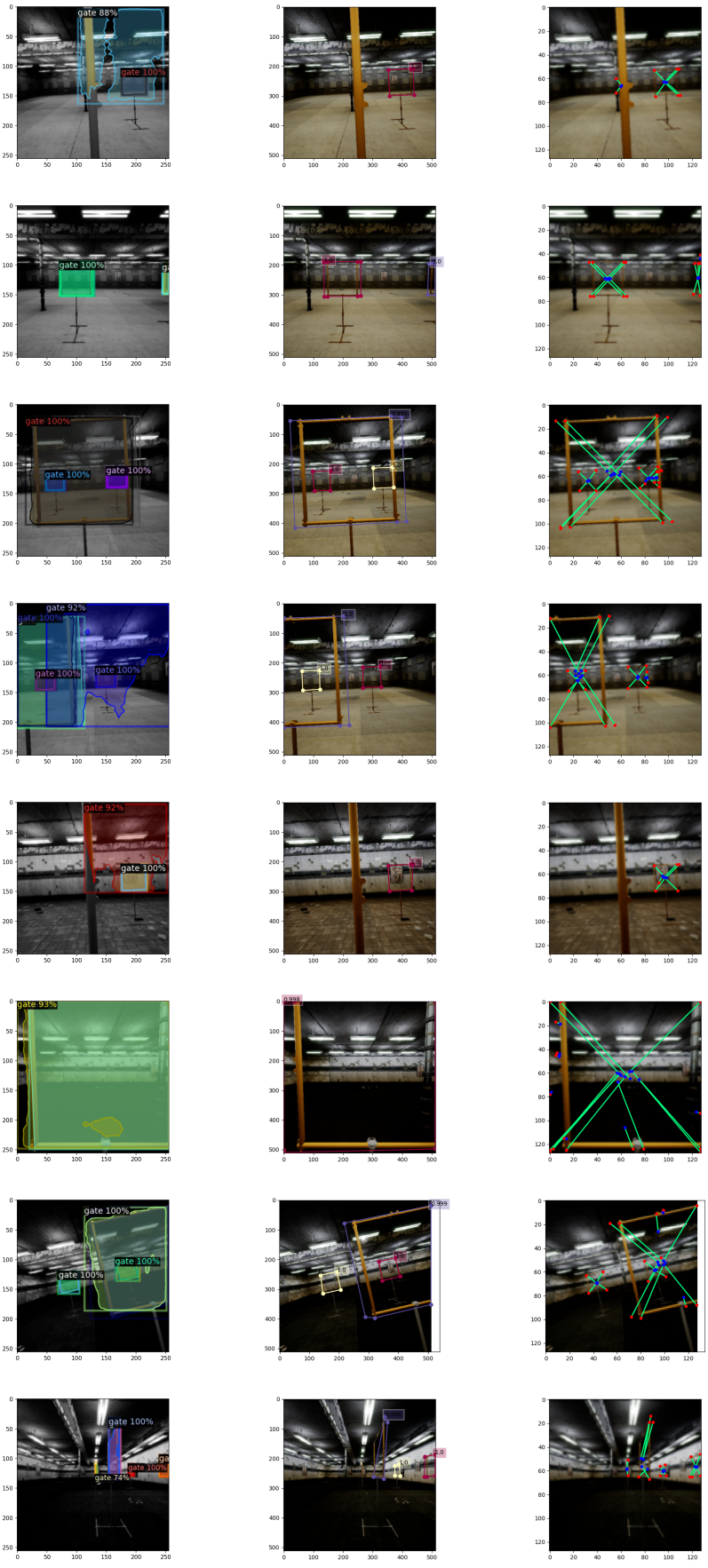
SIFT: Scale Invariant Feature Transform

SVM: Support Vector Machine

Yolo: You only look once

A.2 RESULTS

Some test results on basement3 dataset is shown in this section. First column shows Mask R-CNN's predictions and the ground truth masks. The second column shows our approach's results. The third column shows detected vertices and voted centers.


gate 88%
gate 100%
gate 100%
gate 100%
gate 100%
gate 100%
gate 92%
gate 100%
gate 100%
gate 100%
gate 92%
gate 100%
gate 93%
gate 100%
gate 100%
gate 100%
gate 100%
gate 74%