Conditional Link Prediction of Category-Implicit Keypoint Detection

Supplementary Materials

Ellen Yi-Ge[1], Rui Fan[2], Zechun Liu[1], Zhiqiang Shen[1]
[1] Carnegie Mellon University
[2] UC San Diego

A. Deep Path Aggregation Detector

Multi-scale feature fusion aims to aggregate features at different resolutions. Formally, given a multi-scale feature P_{li} at layer li, we aim to design an appropriate approach to effectively aggregate different features and update to a deeper layer with renovated features. The conventional Feature Pyramid Networks (FPN) (Lin et al. 2017) aggregate multi-scale features in a top-down manner, but it is inherently limited by the one-way information flow. Thus, PANet (Liu et al. 2018) provides an extra bottom-up path aggregation network. Figure 1(a) illustrates the additional path with red arrows. The neurons in high layers strongly respond to entire objects, while other neurons are more likely to be activated by local texture and patterns. This manifests the necessity of augmenting a top-down path to propagate semantically strong features and enhance all features with reasonable classification capability. The coarser feature map C_{li} at layer li and the generated feature maps N_{li} at layer li with higher resolution can be calculated as:

$$C_{li} = Concat(U(C_{li+1}), g(P_{li})), \quad (1)$$

$$N_{li} = Concat(D(N_{li-1}), g(C_{li})). \quad (2)$$

where g represents the convolutional operations for feature processing, U is usually an upsampling procedure and D is usually a downsampling procedure for resolution alignment, and $Concat$ denotes the concatenate operation.

Inspired by PANet (Liu et al. 2018), we design a deep path aggregation network, DPAD, to enhance the localization capability of the entire feature hierarchy by propagating strong responses of low-level patterns, which is illustrated in Figure 1(b). ResNeXt (Xie et al. 2017) is utilized as the backbone network to generate different levels of feature maps, namely $\{C3, C4, C5, C6, C7\}$. Table 1 demonstrates the the superiority of ResNeXt-101 considering both AP and FLOPs. In addition to these feature maps generated from FPN, two higher-level feature maps, C8 and C9, are created by downsampling from C5. The augmented path starts from the lowest level and gradually approaches to the top. From C3 to C9, the spatial size is gradually downsampled with factor 2. $\{N3, N4, N5, N6, N7, N8, N9\}$ denote newly generated feature maps corresponding to $\{C3, C4, C5, C6, C7, C8, C9\}$. Each building block takes a higher-resolution feature map N_i and a coarser map C_{i+1} through lateral connection to generate a new feature map N_{i+1} as follows:

$$C_{li} = D(C_{li-1})_{i\in(8,9)}, \quad (3)$$

$$N_{li} = Concat(D(N_{li-1}), g(C_{li}))_{i\in(8,9)}, \quad (4)$$

Unlike PANet (Liu et al. 2018), we remove the mask branch and adopt CIoU (Zheng et al. 2020) to penalize the union area over the circumscribed rectangle's area in IoU Loss. CIoU can achieve better convergence speed and accuracy for bounding box (BBox) regression problem.

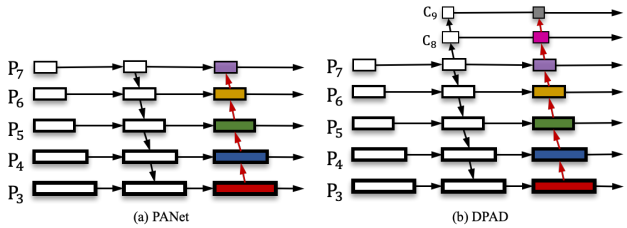

Figure 1: a) The architecture of the PANet. b) The architecture of DPAD. Two higher-level feature maps, C8 and C9, are created by down-sampling from C5. Thus, another two feature maps, N8 and N9, are generated on the neck part. This design enhances the localization capability of the entire feature hierarchy by propagating strong responses of low-level patterns. Additionally, unlike the backbone ResNet utilized in PANet, we use ResNeXt as the backbone in our DPAD.

Table 1: Detection performance (%) comparison of the ResNeXt models with different layers.

Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FLOPs(G)
ResNeXt-50	40.6	59.3	37.2	17.8	41.6	56.4	4.6
ResNeXt-72	41.4	62.1	41.9	18.3	42.4	58.9	6.1
ResNeXt-101	42.7	64.7	45.8	20.1	45.1	64.3	7.3
ResNeXt-124	42.9	65.1	46.3	20.8	45.4	64.9	10.6
ResNeXt-152	43.3	65.3	46.5	21.3	45.9	65.3	12.1

Table 2: Detection performance (%) comparison among different models. DPAD$*$ has two more higher-level feature maps $C10, C11$, and DPAD† has four more higher-level feature maps $C10, C11, C12, C13$.

Backbone	AP	AP_{50}	AP_{75}	FLOPs(G)	#Params
PANet	42.0	65.1	45.7	7.1	78M
DPAD$*$	43.0	64.9	46.3	7.8	94.5M
DPAD†	42.9	64.7	46.1	8.1	99.3M
NAS-FPN	43.1	65.3	46.5	12.1	166.5M
BiFPN	44.4	66.4	48.3	18.7	189.2M
DPAD	42.7	64.7	45.8	7.3	89.8M

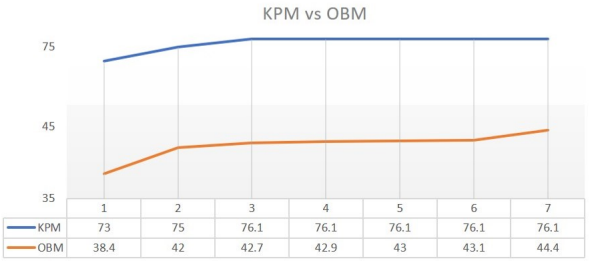

Figure 2: The performance (%) trends of object detection module and keypoint estimation module. In terms of the Keypoint Detection Module (KPM): 1:FPN-based CKLM; 2:PANet-based CKLM; 3.DPAD-based CKLM; 4.DPAD$*$-based CKLM; 5. DPAD†-based CKLM; 6.NAS-FPN-based CKLM; 7.BIFPN-based CKLM. In terms of the Object Detection Module (OBM):1:FPN; 2:PANet; 3.DPAD; 4.DPAD$*$; 5. DPAD†; 6.NAS-FPN; 7.BIFPN.

We have also tried a deeper DPAD and compared the performance with other approaches. Table 2 shows the performance of DPAD$*$ and DPAD†, another two deeper DPADs. Based on DPAD, the former has two more higher-level feature maps $C10$ and $C11$, and the other has four more higher-level feature maps $C10, C11, C12$, and $C13$. From Table 2, we achieve an advance of 0.3% in AP for DPAD$*$, but the FLOPs increases by 0.5G. Even the module is designed deeper, like in DPAD†, the AP will even decrease, which indicates the precision of the module does not increase w.r.t. the deeper feature layers. Another two approaches, NAS-FPN (Ghiasi, Lin, and Le 2019) and BIFPN (Tan, Pang, and Le 2020) can achieve higher precision, but the FLOPs and the number of parameters are also huge. Thus, we discuss what influences object detector on the top-down keypoint detection in Figure 2.

As shown in Figure 2, the orange line is the trend of the object detector performance; and the blue one illustrates the performance of the top-down keypoint detection. When the object detector's precision is low, the top-down keypoint detection performance will be more dependent on the objector detector. However, when the object detector's AP is larger than 42.7%, the precision of the top-down keypoint detection becomes saturated; namely, the top-down keypoint detection performance does not heavily rely on the object detector. Thus, a trade-off between the precision and the cost is the key to design an efficient top-down keypoint detector. Based on this prospect, we adopt DPAD as the final object detector in our system.

Table 3: Comparison of a 3-stage CKLM with different corse-to-fine strategies on COCO minival dataset. For each strategy, the number in the table represents the kernel size. The kernel size controls the fineness of supervision and a smaller value indicates a finer setting.

Setting	1	2	3	4	5	6	7
Stage 1	7	5	7	7	7	5	5
Stage 2	7	5	5	7	5	5	3
Stage 3	7	5	5	5	3	3	3
AP(%)	75.0	74.6	75.1	74.8	75.3	74.4	74.1

From the top row of Figure 4, it is obvious to figure out the differences among the heatmaps with distinguishable sizes of Gaussian kernels, 7, 5, and 3. The strategy is based on the observation that the estimated heatmaps from multi-stages are also in a similar coarse-to-fine manner. The bottom row of Figure 4 shows an illustrative corresponding predictions (yellow lines) and ground truth annotations (blue lines). The pink circles on both left and center images display the prediction errors, which demonstrates that the proposed strategy is able to refine localization accuracy gradually.

B. Cross-Stage Keypoint Localization Module (CKLM)

In the paper, we have discussed the performance of CKLM w.r.t. different number of single stages. The final CKLM module contains three stages to balance the precision and the cost. After a single stage, the single heatmap contains predefined the most probable keypoint locations. However, the heatmap from the first single stage is a coarse prediction with abounding noise, even adequate features have been extracted in the stage. To filter the noise, another two stages are cascaded with a coarse-to-fine surveillance strategy to boost the keypoint localization performance. Since the Gaussian kernel is used to generate the ground truth heat map for each key point, we decide to use distinguishable sizes of kernels, 7, 5, and 3, in these three stages. The distribution of the heatmaps with distinguishable sizes of kernels are demonstrated in Figure 3.

Table 3 demonstrates the performance of the 3-stage CKLM with disparate Gaussian kernel sizes. We employ distinctive settings for the 3-stage CKLM each time. As shown in Table 3, setting 1 achieves an AP of 75.0%, whose kernel sizes are 7 in all three stages. We tried to degrade the kernel size to 5 in all three stages, however, the AP decreases by 0.4%. It indicates that if we adopt the same kernel sizes, the larger one can present a better performance. We conjecture that this is because the smaller region after the first stage would negatively affect the performance of the detector. Thus, setting 3 and setting 4 are proposed to prove our speculation. When the kernel size in the first stage increases from 5 to 7 in setting 3, the AP is escalated by 0.5%; while the kernel size of the second stage further increases to 7 in the setting 4, the AP is depreciated 74.8 $\%$, even better than

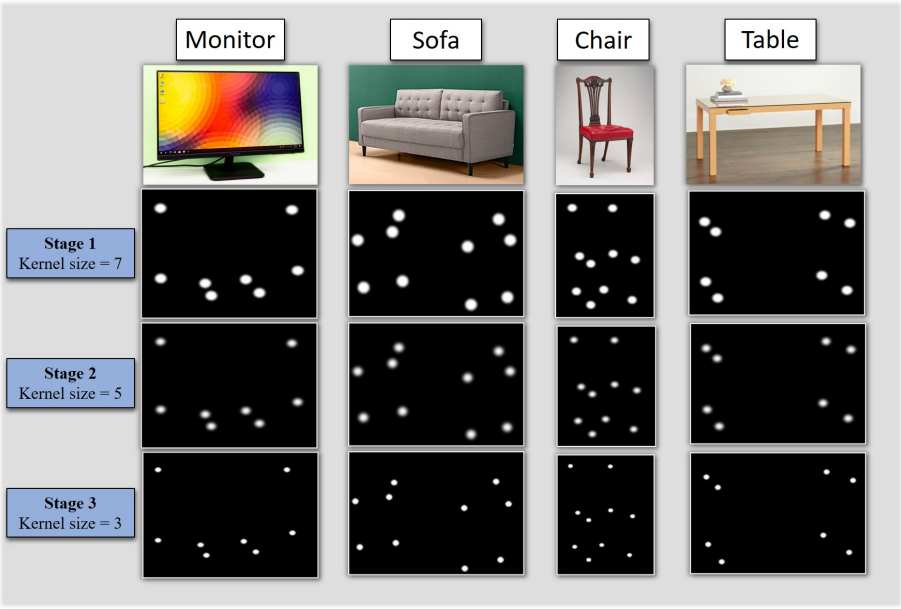

Figure 3: Performance of CKLM with different Gaussian Kernel size.

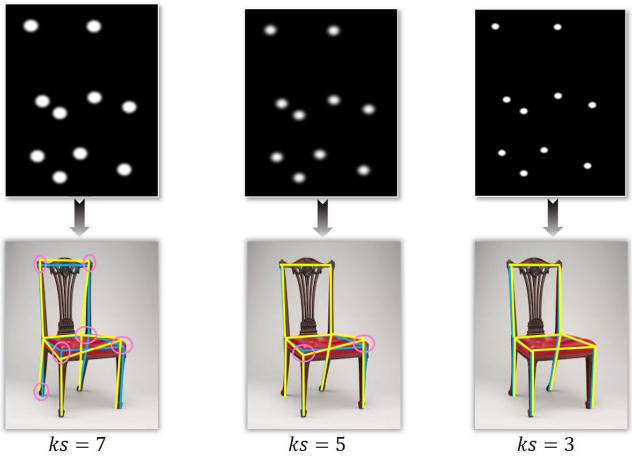

Figure 4: Illustration of a coarse-to-fine strategy. The top row shows the ground-truth heatmaps of the distinctive single stages. From left to right, the Gaussian kernel sizes (ks) of each stage are 7, 5, and 3. The bottom one illustrates the corresponding prediction performance (yellow lines) and the ground-truth annotations (blue lines). The radius of pink circles displays the prediction errors, which are the L2 distances between the ground-truth keypoint position and predicted location.

the performance in setting 2. It shows that the kernel size in the second stage should be smaller than the one in the first stage. Thus, we decide to diminish the kernel size in the third stage to further validate our hypothesis. In setting 5, the kernel sizes in three stages are set as 7, 5 and 3, respectively. The performance in this setting accomplished the best, and the AP is escalated to 75.3 %. Another two further settings, setting 6 and setting 7, are also made to figure out the performance trends with a smaller kernel size in the first stage. In accordance with expectations, the AP is worse than the one in setting 5. Finally, we adopt the best setting of the kernel sizes: 7, 5 and 3 in our 3-stage CKLM.

Table 4 shows the architecture of the single stage of our proposed CKLM. There exist two main branches, the downsampling path and the upsampling path, in each single stage. Each path contains four corresponding layers: DS-1, DS-2, DS-3 and DS-4 for the downsampling path and US-1, US-2, US-3 and US-4 for the upsampling path. The downsampling layer consists of several BottleNeck-4 and bottleneck-3 blocks; while the upsampling layer encompasses several UpUnit-4 and UpUnit-3 blocks. Each BottleNeck and UpUnit block includes distinctive number of convolution layers, batch normalization (Ioffe and Szegedy 2015), and ReLU (Agarap 2018) activation functions.

C. Conditional Link Prediction Graph Module (CLPGM)

In the paper, we construe the details of CLPGM, which includes the Location Instability Strategy (LIS). The LIS is utilized to disentangle occlusion cases under the same category. When multiple targets are occluded, the number of detected nodes may be higher than the predefined number in the overlapped area. If these targets are with the same label, we should design a LIS to infer which nodes are for each overlapped target. Since the details of LIS are already introduced in the paper, we only demonstrate our approach with more samples.

Tabel 5 illustrates the comparison among KLPNet, KLPNet$\star$ and KLPNet† on ObjectNet3D+. From Table 5, KLPNet† achieves the best performance on distinctive categories. Since our approach is the forerunner for link prediction on multi-class rigid bodies, it is hard to compare it with others either quantitatively or qualitatively. Here we vi-

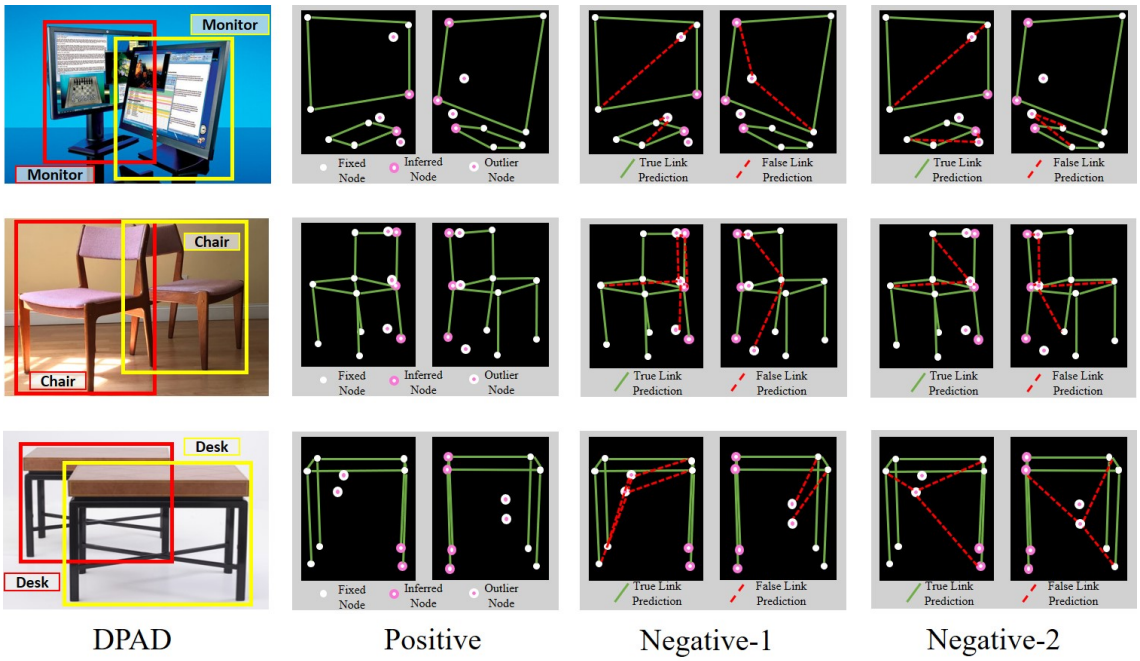

Figure 5: Location Instability Strategy Performance. The left columns indicate the results of the DPAD; The column "Positive" displays the results of LPGM with LIS; The columns "Negative-1" and "Negative-2" illustrate the potential negative samples from the results of the system without LIS.

sualize the conditional connection link to illustrate the qualitative performance. From Figure 5, our KLPNet† provides correct connection links in various cases and the semantic information well manifests themselves.

D. Loss Function

Recall the total loss Keypoint and Link Prediction Network (KLPNet) is formulated as follows:

$$\mathcal{L}_{KLPNet} = \alpha\mathcal{L}_{kd} + \beta\mathcal{L}_{link}, \tag{5}$$

where α and β are the predefined constant parameters.

Table 6 illustrates the performance of the whole module with different settings of the loss. We tried nine settings for the coefficient, α and β. If the proportion of α is large, the precision of the KLPNet is low; while if it is tiny, the precision is not achievable either. Finally, the setting of α and β, 0.3 and 0.7, can achieve the best performance.

E. Other Applications

KLPNet can be utilized for keypoint detection on rigid bodies. In this section, we discuss some other applications based on our KLPNet.

Refined Object Detection

If CLPGM is removed from the KLPNet, the category-implicit keypoints are localized in the images. Without CLPGM, we cannot connect the keypoints correctly due to semantic chaos. In this case, we hope to refine the bounding box as a polygon area, which can encircle the target with a narrow yet more accurate area. The results are shown in Figure 6. However, we concede this approach is only an attempt to refine the object detection. Some recent object detector (Wei et al. 2020) can generate a more accurate and efficient area to encircle the target in the image.

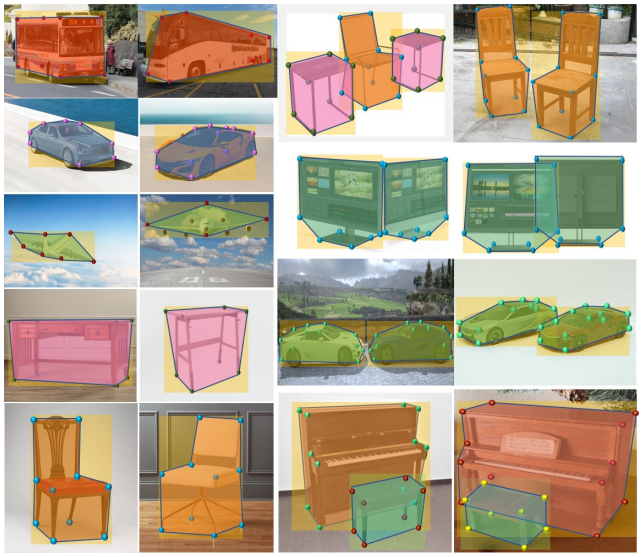

Figure 6: Without CLPGM, the nodes are bounded to get the maximum semantic bounding area. It narrows the interested regions.

Table 4: The architecture of the single stage in CKLM.

DownSampling Path	Type	Number	Type	Number
DS-1	BottleNeck-4	3	BottleNeck-3	4
DS-2	BottleNeck-3	5	BottleNeck-4	3
DS-3	BottleNeck-4	2	BottleNeck-3	5
DS-4	BottleNeck-3	5	BottleNeck-4	3

UpSampling Path	Type	Number	Type	Number
US-1	UpUnit-4	3	UpUnit-3	4
US-2	UpUnit-3	5	UpUnit-4	3
US-3	UpUnit-4	2	UpUnit-3	5
US-4	UpUnit-3	5	UpUnit-4	3

Type	Layer	Kernel	Stride	Padding
BottleNeck-4	Conv	1×1	1×1	–
	BN + ReLU	–	–	–
	Conv	3×3	1×1	1×1
	BN + ReLU	–	–	–
	Conv	3×3	1×1	1×1
	BN + ReLU	–	–	–
	Conv	1×1	1×1	–
	BN + ReLU	–	–	–
BottleNeck-3	Conv	1×1	1×1	–
	BN + ReLU	–	–	–
	Conv	3×3	1×1	1×1
	BN + ReLU	–	–	–
	Conv	1×1	1×1	–
	BN + ReLU	–	–	–
UpUnit-4	Conv	1×1	1×1	–
	BN + ReLU	–	–	–
	Conv	3×3	1×1	1×1
	BN + ReLU	–	–	–
	Conv	1×1	1×1	–
	BN + ReLU	–	–	–
	Conv	3×3	1×1	1×1
	BN + ReLU	–	–	–
UpUnit-3	Conv	1×1	1×1	–
	BN + ReLU	–	–	–
	Conv	1×1	1×1	–
	BN + ReLU	–	–	–
	Conv	3×3	1×1	1×1
	BN + ReLU	–	–	–

Simultaneous Localization and Mapping (SLAM)

SLAM is a computational problem of constructing or updating the map of an unknown environment while simultaneously keeping track of an agent's location within it. SLAM (Durrant-Whyte and Bailey 2006) (Bailey and Durrant-Whyte 2006) contains two subsystems: localization and mapping. Localization is not only the first step but also the key of success to figure out the decision of the whole system. Current approaches (Mur-Artal, Montiel, and Tardos 2015) (Cui and Ma 2019) (Gomez-Ojeda et al. 2019) of localization concentrate to pair the landmarks on two frames. We believe that KLPNet offers a new sight to localize the special objects. In terms of SLAM, the details, such as texture, color, are not essential and could be ignored to localize the target. Based on the accurate detection and localization of semantic keypoints in the real world, it is accessible to discern the robot's location and build the real-world mapping. To accomplish it, we need to consider how to get the coordinates in the 3D space, which is discussed in the next part.

Table 5: Comparison of Keypoint Estimation Results (%) on ObjectNet3D+. Note: KLPNet represents KLPNet with only the basic backbone; KLPNet$\star$ additionally includes the cross-stage (three stages) feature aggregation scheme; KLPNet† contains both cross-stage feature aggregation scheme and Location Instability Strategy (LIS). LIS of CLPGM provides feedback to CKLM to adjust and fine-tweak the keypoint localization.

Method	bed	sofa	bookshelf	chair	monitor
KLPNet	69.6	68.9	71.8	74.1	79.6
KLPNet$\star$	81.3	75.8	76.3	77.6	85.3
KLPNet†	87.4	81.1	83.4	84.8	89.7

Method	car	bus	aircraft	mirror	piano
KLPNet	63.1	76.9	69.0	69.1	63.5
KLPNet$\star$	69.7	80.1	73.3	73.5	69.8
KLPNet†	74.5	85.9	78.9	78.7	72.6

Method	laptop	diningtable	basket	eraser	flashlight
KLPNet	62.9	77.4	71.2	69.3	65.3
KLPNet$\star$	64.3	81.3	74.3	72.8	70.8
KLPNet†	71.8	84.9	78.7	76.5	74.1

Method	microwave	console	guitar	loudspeaker	knife
KLPNet	91.4	63.7	71.6	68.8	64.1
KLPNet$\star$	96.1	68.5	74.8	73.9	69.3
KLPNet†	77.8	71.3	79.9	79.1	72.6

Method	keyboard	extinguisher	camera	hammer	train
KLPNet	71.8	65.8	70.3	64.8	73.2
KLPNet$\star$	74.7	69.4	75.9	69.5	76.8
KLPNet†	80.3	72.3	80.1	74.3	81.1

Method	cabinet	helmet	coffee maker	printer	blackboard
KLPNet	84.3	72.1	73.4	71.3	78.5
KLPNet$\star$	87.9	75.3	76.8	74.8	81.3
KLPNet†	90.1	79.1	81.2	79.7	86.9

Table 6: The performance of the module with different coefficient setting of the loss function

Setting	1	2	3	4	5	6	7	8	9
α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
β	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
AP	55.3	66.4	75.3	73.7	53.8	47.9	39.4	34.3	34.1

3D Reconstruction and Rendering

3D Reconstruction (Park et al. 2020) can determine the object's 3D profile, and 3D Rendering (Johnson et al. 2017) is the final process of creating the actual 2D image or animation from the prepared scene. The first step of both fields is

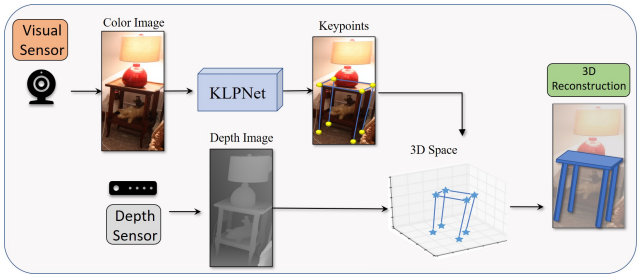

Figure 7: KLPNet with depth map in 3D Applications.

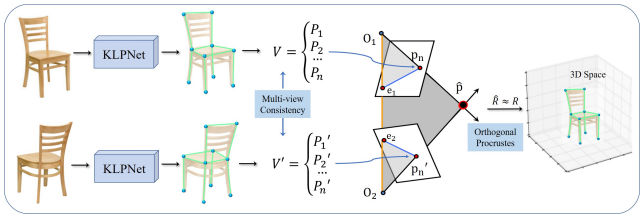

Figure 8: KLPNet with multi-view consistency in 3D Applications. V and V' represents two views; $\{P_1, \cdots, P_i\}$ and $\{P_1\text{'}, \cdots, P_i\text{'}\}$ represent points under the corresponding view; O denotes the original point and e denotes the epipolar.

to project the 2D object into a 3D space. Thus, we propose two latent approaches to implement the projection: KLPNet with multi-view consistency and KLPNet with depth map, which are shown in Figure 7 and Figure 8.

Figure 7 illustrates the first approach for converting 2D targets into a 3D space. Two sensors are utilized to capture useful information, color image and depth map. Using KLPNet, the keypoints can be localized and connected on the 2D heatmap. Depth map affords the space distance of each pixel in the 2D image. After merging the heatmap and depth map, it is conceivable to reconstruct the 3D object in the 3D space.

The second approach to build the 3D object is to utilize the multi-view consistency instead of the depth map. After obtaining the location and connection of keypoints on the 2D neighbour keyframes, the known rigid rotation (**R**) and translation (**T**) between the two views is provided as a supervisory signal. As shown in the 8, V_1 and V_2 are the two views that best match one view to the other. A multi-view consistency loss can be considered to measure the discrepancy between the two sets of keypoints under the ground truth transformation. Once the transformation is corrected, it is conceivable to reconstruct the 3D object in 3D space.

F. Blemish

During the testing, we also find some blemishes as shown in Figure 9. The top-left one is a desk with four additional legs, which lead our model to misjudge the four basic bottom nodes; The bottom-left one is a sofa-bed combination that baffles the model to localize the node accurately since the bed and sofa have a different predefined number of nodes; The top-right contains an unusual desk and a chair who have more legs than normal samples during training; Our model predicts all nodes on the chair correctly but fails to connect

Figure 9: Some unusual cases.

all chair legs. We notice that our model cannot self-adapt the node number. The predefined number of nodes per class limits the performance of the model. In the future, a novel supervised approach can be designed to depict more suitable edges for specific geometrical patterns of the objects.

References

Agarap, A. F. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Bailey, T., and Durrant-Whyte, H. 2006. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine* 13(3):108–117.

Cui, L., and Ma, C. 2019. Sof-slam: A semantic visual slam for dynamic environments. *IEEE Access* 7:166528–166539.

Durrant-Whyte, H., and Bailey, T. 2006. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine* 13(2):99–110.

Ghiasi, G.; Lin, T.-Y.; and Le, Q. V. 2019. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7036–7045.

Gomez-Ojeda, R.; Moreno, F.-A.; Zuñiga-Noël, D.; Scaramuzza, D.; and Gonzalez-Jimenez, J. 2019. Pl-slam: A stereo slam system through the combination of points and line segments. *IEEE Transactions on Robotics* 35(3):734–746.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Johnson, P. T.; Schneider, R.; Lugo-Fagundo, C.; Johnson, M. B.; and Fishman, E. K. 2017. Mdct angiography with 3d rendering: a novel cinematic rendering algorithm for enhanced anatomic detail. *American Journal of Roentgenology* 209(2):309–312.

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

Liu, S.; Qi, L.; Qin, H.; Shi, J.; and Jia, J. 2018. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8759–8768.

Mur-Artal, R.; Montiel, J. M. M.; and Tardos, J. D. 2015. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics* 31(5):1147–1163.

Park, K.; Mousavian, A.; Xiang, Y.; and Fox, D. 2020. Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10710–10719.

Tan, M.; Pang, R.; and Le, Q. V. 2020. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10781–10790.

Wei, F.; Sun, X.; Li, H.; Wang, J.; and Lin, S. 2020. Point-set anchors for object detection, instance segmentation and pose estimation. *arXiv preprint arXiv:2007.02846*.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500.

Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; and Ren, D. 2020. Distance-iou loss: Faster and better learning for bounding box regression. In *AAAI*, 12993–13000.