Efficient Amodal Segmentation and 3D Detection using Rich 3D Features and Synthetic Augmentation

Jingcheng Yang

Shenzhen College of International Education, Shenzhen, China

Abstract Although classical computer vision tasks such as instance segmentation has reached high accuracy, it does not provide sufficient information for reliable decision making in more complex downstream tasks that requires further spatial context, e.g. autonomous driving. Amodal perception tasks such as amodal instance segmentation and 3D object detection, can provide adequate information for making difficult spatial decisions.

Amodal segmentation is the pixel-accurate segmentation of object shapes, including the occluded region. We tackle this problem using the principle that depth is an inherent part of occlusion. We propose a novel depth-aware amodal instance segmentation network, DAISnet, to fully utilize depth information for amodal instance segmentation. Given the network is already using 3D contextual information, we extend our network to make 3D bounding box predictions using point clouds and 3D convolution. By using 2D proposals, we significantly reduce the complexity of finding 3D RoIs, which enable us to make fine-tuned predictions at these attentive regions. To address the lack of training data, we propose a synthetic augmentation dataset to enhance network performance on existing datasets e.g. Kitti. Our experiments demonstrate that synthetic data can significantly improve accuracy in the most ambiguous cases.

Keywords: Instance Segmentation, Amodal Perception, 3D Object Detection, Deep Learning, Synthetic Data Collection

1 Introduction

1.1 Amodal Instance Segmentation

Amodal instance segmentation is the segmentation of the visible and occluded region of all objects from an image, as shown in Fig 1. Visible objects can be segmented with a high degree of accuracy thanks to progress made in the last few years. Segmenting occluded regions of an object, is much more difficult to achieve. There is no single probable solution for some occluded regions as Fig 2 demonstrates, and it requires a high level understanding of the image semantics to achieve reasonable predictions. The annotation of amodal instance segmentation is also expensive, as it requires annotators to thoroughly examine all possibilities. Although the task appears ill-posed, past works have demonstrated that high quality amodal instance segmentation is achievable and helpful. Amodal instance segmentation is also a necessary stepping stone for tasks that require greater amodal awareness, such as 3D amodal segmentation and amodal reconstruction. Amodal instance segmentation is also useful in numerous down-stream tasks. Self-driving vehicles can make better decisions on road using occlusion and spacial information. Occlusion information offers robust information on the order of vehicles for better navigation, while spacial information can help deduce viable parking regions. Amodal instance segmentation is also important in robotic vision. Navigation towards an occluded object, efficient retrieval of objects in complex environments, and preemptive measures all benefit from accurate amodal instance segmentation predictions.

Fig. 1: Amodal instance segmentation segments the entire object while modal (visible) instance segmentation only segments visible regions

Past works regarding amodal instance segmentation involves the use of Mask R-CNN architecture which have achieved state of the art performance in visible instance segmentation. [1–5] Direct implementation of such architecture have yielded reasonable results. Most amodal instance segmentation networks improve upon this architecture. Notably, Xiao et al.[6] introduced a code-book mechanism that conducts feature mapping based on pretrained 2D shape awareness. Amodal mask are refined through this

Fig. 2: There can be multiple amodal predictions given the nature of uncertainty. P3 is the closest to ground truth, but both P1 and P2 are possible predictions. This shows that it's difficult to obtain accurate results in amodal instance segmentation

process, improving the accuracy of such predictions. However, there are limitations in representing 3D objects in a 2D context, which neglects actual spacial relationship between objects, as well as it's 3D features. We believe depth is inherent to occlusion, as it provides information on possible occluded regions and additional features of visible regions. More information about the visible regions of an object can help make better predictions on amodal predictions.

We improve upon previous solutions by creating a depth-aware amodal instance segmentation network (DAISnet) through a monocular depth prediction layer pretrained on synthetic and real-life data. Depth information is incorporated into the backbone feature extraction pipeline to make our network fully depth-aware. Depth information is regionally normalized and computed into a normal tangent map through a normal tangent encoder. Normal maps help describe the change in 3D surface while being distance invariant. To best utilize tangent and depth information, we introduce the Edge Occlusion and Tangent Occlusion modules which relies on normalized depth and tangent information of objects to predict possible regions of occlusion, which is essentially a rough prediction of amodal masks. We base this from the principle that areas directly touching visible objects with smaller depth (closer to the observer) have a very high possibility of being the occluder. We refine amodal mask predictions using depth information for 3D shape prior mapping, which provides rich features even under circumstances with obscure shapes. We used a 3D real-time graphics engine to generate synthetic data of objects with occlusion, photo-realistic colors, depth and amodal segmentation masks. This workflow allows us to generate large quantities of data with accurate pixel-level ground truth based on various scenarios for training.

1.2 3D Object Detection

3D Object Detection is the detection of objects with their 3D bounding boxes. This is much more difficult

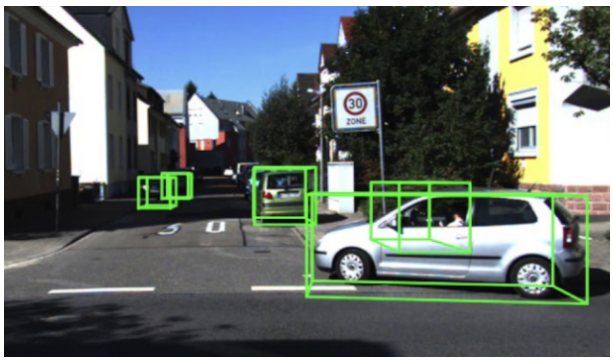

Fig. 3: 3D object detection is the prediction of 3D bounding boxes using 2D or 3D data.

as there needs to be extensive spatial understanding of the object for an accurate 3D boundary to be predicted. Advancements in computational power has enabled developments towards this area. By using using LIDAR point clouds and depth maps, partial or full 3D features can be extracted to acheive 3D Object Detection. 3D Object Detection is crucial towards autonomous navigation in vehicles and beyond. No reliable predictions can be made in our 3D world using purely information from 2D contexts, such as Instance Segmentation.

3D Object Detection suffers from longer training and inference times, data deficiency, and loss of accuracy in encoding point clouds and other 3D data. We propose methods to address these issues by augmenting a trained network with additional synthetic data, use existing 2D network components and decrease prediction times by improving attentiveness of our network. Notably, we used the Mask-RCNN network for feature extraction and 2D object detection. By projecting these information to a 3D representation, we can minimize complexity in locating regions of interest and use rich 2D features from backbones trained on ImageNet for 3D object detection.

1.3 Contributions

Our contributions can be summarized as:

- A novel depth-aware network for amodal instance segmentation.
- A synthetic dataset that can be used for depth prediction, amodal instance segmentation, and 3D object detection training that enhances performance and accuracy on existing datasets.
- A post refinement RoI mask head for amodal completion using depth and tangent information.
- A 3D codebook mechanism for amodal mask refinement based on 3D shape prior.
- A 2D-3D hybrid network using Mask-RCNN for efficient 3D object detection.

2 Related Work

2.1 Instance Segmentation

Instance Segmentation segment and classify instances' visible part separately from the image. Amodal instance segmentation extends directly from research in instance segmentation, which itself extended from research regarding object detection. Ross at el. [7] introduced the R-CNN network, which selects region of interests from an image and conduct convolutions on these regions. Regional CNNs have supersede all previous implementations in performance and accuracy. Ross at el. [8] introduced the Fast R-CNN network that significantly faster than R-CNN, which was later improved again into Faster R-CNN [1], which improved proposal quality and performance by using a Region Proposal Network for proposal generations.

He et al. extends Mask R-CNN [2] to instance segmentation by introducing a Mask Head, which conducts segmentation on detected objects using work derived from Long at el. [9] on semantic segmentation using Fully Convolutional Networks(FCN). He also proposed RoI Align solution in place of RoI pooling which achieved pixel accurate region of interests. Chen et al. [3] incorporated Cascade techniques [4] to Instance Segmentation and improved accuracy through stage specific refinement. Ding et al. [5] extends cascade instance segmentation by taking advantage of separate stages to establish a bidirectional relation between the mask and the bounding box. By introducing a mask guided RoI align method, the bounding box and mask prediction continues to refine each other for accurate instance segmentation. Mask Scoring R-CNN [10] aims to reevaluate predicted masks by giving lower scores for masks with bad quality, and thus improve mask prediction.

2.2 Monocular Depth Estimation

Monocular depth estimation use one image to inference depth of each pixel to achieve a 2D-3D projection. Monodepth [11] proposed a self-supervised training pipeline for monocular depth estimation. They used a FCN to predict disparities for rectified stereo images and supervise it via an image reconstruction loss. Godard et al.[12] extends the monodepth model to a video-based monocular depth estimator. Performance was improved by introducing a Per-Pixel Minimum Reprojection Loss, Auto-Masking Stationary Pixels, and Multi-scale Estimation. Fu et al. [13] directly regress the depth through a CNN and introduced a method for monocular depth estimation that relies on space-increasing discretization, differing from past implementation by its smaller architecture and faster convergence, achieved very accurate results. Lee at al. [14] utilizes planar guidance layers in multiple stages of decoding, which utilizes light and color changes to generate surface information for depth prediction.

2.3 Amodal Instance Segmentation

As work regarding modal instance segmentation begins to yield state-of-the-art results, research have been done to extend instance segmentation to occluded region of objects. Some early work involves directly extending and modifying existing R-CNN architecture to support amodal instance segmentation [15–17] Li at al. [18] used an iterative bounding box method to filter amodal instance segmentation results from a CNN. Xiao et al. [6] used a 2D-shape prior codebook that conducts feature matching to enhance amodal instance segmentation predictions. Zhang et al. [19] introduces a semantics-aware distance map that allows for pixel level amodal segmentation and occlusion order prediction. Others have tried to include abstract feature awareness to achieve more accurate amodal segmentation predictions [20] Yang et al. [21] extended amodal instance segmentation using multi-perspective images that are taken at positions determined by the network. Deng [22] used depth information to predict amodal bounding box of 3D objects with large success. This demonstrates that 3D visual features can enhance amodal predictions overall and the 2.5D visual features are correlated to 3D object sizes, locations, and orientations.

2.4 3D Object Detection

Song [23] proposed a method of projecting depth data into 3D point clouds, then conduct 3D convolution similar to that of Faster-RCNN called sliding shapes to find 3D regions of interest that may locate the object in its 3D boundaries. This method however, is expensive as convolutions in 3D are much larger and memory-consuming. For larger open scenes like open roads, directly conducting 3D convolutions will take too much time. Charles [24] proposed PointNet which avoids conducting convolutions on voxel grid data by directly operating on point clouds. Direct operation on raw point clouds are more accurate than convolutions, as converting point clouds to voxels results in degradation of features. Chen [25] proposed a BEV (Birds Eye View) representation of 3D data, which could then be processed in 2D. This is still employed by many high performing networks [26–29]. The use of raw point clouds and voxels still remains a debate, Voxel R-CNN [30] demonstrated that by using voxel information appropriately, similar levels of performance to raw point clouds can be achieved with significantly less processing time. PV-RCNN [29] points out that both Voxel and Raw Point Clouds have their advantages, and proposed a method that combines both formats for best results. Our 3D Object Detection model is based upon PV-RCNN.

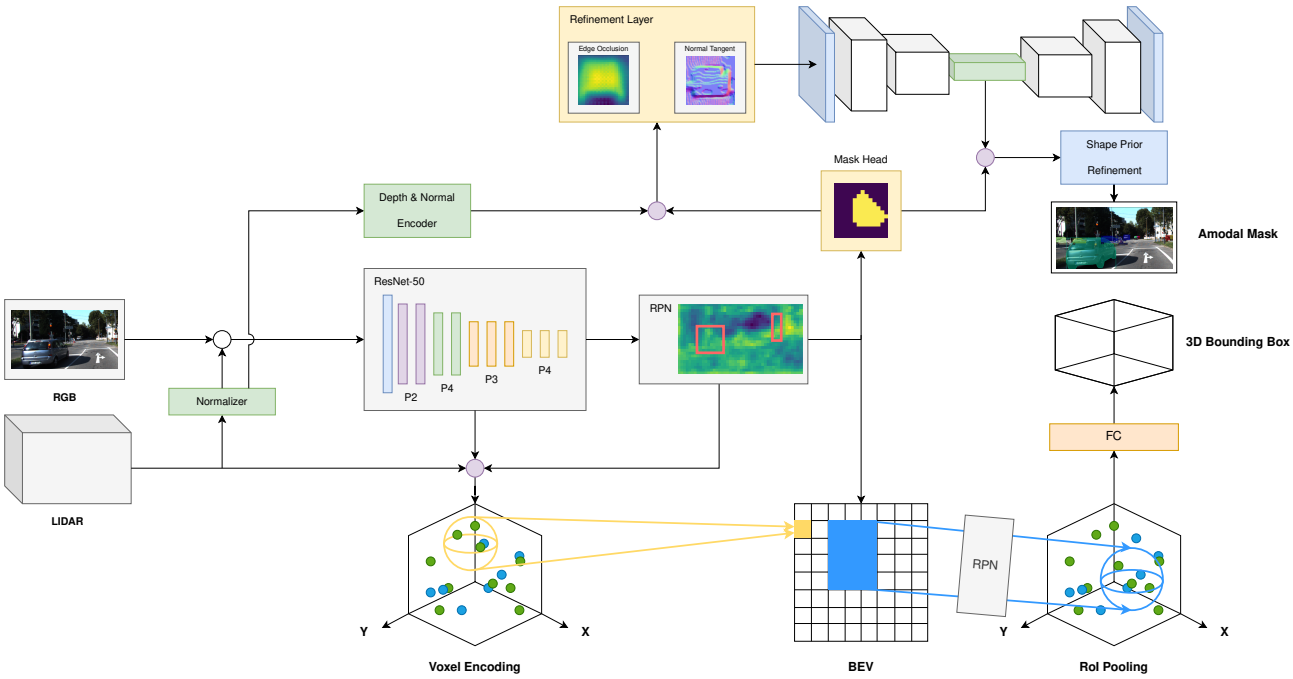

Fig. 4: Network overview

3 Method

3.1 Overview

Our network improves upon the Mask-RCNN network that was adapted for amodal instance segmentation, with the 3D object detection section serving as an integrated extension. We initially proposed a depth-based network for amodal instance segmentation only, this will be referred to as DAISnet (Depth-Aware Amodal Instance Segmentation Network). We first obtain depth information from LIDAR data, though a monocular depth prediction network [14] can be used instead to conduct this task in monocularly. Fig 4 demonstrates our complete network. The extracted/predicted depth map is first passed to the feature extraction backbone alongside RGB data. The backbone extracts feature maps and generate proposals using the Regional Proposal Network; Since depth information was included in feature extraction, this entire process is depth-aware.

Depth information is also explicitly given to the RoI pooling layer to generate pooled depth regions with the same dimension as the object proposals. Box proposals are then used to generate visible mask predictions through the visible mask head. The visible mask and depth information are both passed to the 3D-aware refinement layers, which consists of occlusion edge and 3D shape prior refinements.

The occlusion edge refinement layer uses localized depth values to deduce possible regions of occlusion, which is then given to an occlusion refinement head that refines the amodal mask using occlusion edges. The 3D shape prior is based upon 2D shape prior works done by Xiao et al. [6]. The shape prior mechanism uses deep learning to effectively encode and decode shapes. Statistical operations such as clustering can be done on encoded shapes, which is used for shape matching and refinement based on prior seen shapes. 3D shape prior uses localized depth information during encoding, which means that the stored codebook contains 3D representations of objects. After refinement through both modules, the network returns the final amodal instance segmentation results.

The integrated network, which includes Amodal Instance Segmentation and 3D Object Detection is called ESAAN (Efficient Synthetic Augmented Amodal Network). The 2D proposal network yields high recalls on object detection, we exploit this advantage by mapping 2D proposals directly onto the 3D scene, including the features extracted by the ResNet50 backbone. By locating regions of interest prior to the actual convolution, our network only need to encode regions around the 2D proposals, which is much faster. We transform this grid based aggregation of point clouds to a BEV (Birds Eye View) representation, which is the scene viewed from top down. Another 2D regional proposal network operates over this BEV to obtain bounding boxes, where 3D bounding boxes can be deduced, as most objects are horizontal to ground.

We noticed that most networks suffer from low recall in sparse point cloud situations, on a dataset that has very few samples of. Since LIDAR point clouds could be easily produced, we created a dataset similar to Kitti which our model will train on first. Then we use transfer learning to train the model on the actual Kitti dataset.

Figure 4 shows the ESAAN architecture; However, DAISnet, which is the none 3D part of the net-

work, can be trained and executed separately for pure amodal instance segmentation. Our albation studies on amodal instance segmentation will only consider the DAISnet portion of ESAAN.

3.2 Synthetic Data Collection

Fig. 5: High-fidelity street scene created within Unity Engine using premade assets. The camera will move through the street, and objects will be placed around it. This mimics many of the images in KITTI which was taken in a narrow street with vehicles.

To create a dataset that contains both segmentation and depth data, we decided to develop a synthetic dataset using Unity Engine©. Similar to the KINS dataset used in past research, our synthetic dataset focuses on amodal instance segmentation of Pedestrians and Vehicles in street and road scenarios. For our street scenario, we setup a 3D street that spans for 100 meters long. Buildings and other items are placed for realism and to ensure our network knows how to ignore distractions. The camera moves randomly through the street within a fixed boundary.

The range which a vehicle could be placed is denoted below:

$$-5 \leq x \leq 5 \tag{1}$$
$$1 \leq y \leq 2 \tag{2}$$
$$0 \leq z \leq 50 \tag{3}$$

The additional random parameters for a vehicle is:

$$V \begin{cases} V_n & = r(2,3) \\ V_p & = [\ r(-10,10),\ \gamma,\ z + r(10,50)\ ] \\ V_r & = [\ 0,\ r(0,360),\ 0\ ] \end{cases} \tag{4}$$

Where for vehicle V: V_n denotes number of vehicles in the scene, V_p denotes position vector of a specific vehicle , γ denotes a constant y value for vehicle height, V_r denotes rotation of V euler angles, $r(min, max)$ specifies a random value between two inclusive values.

We assigned bounding boxes for each of the object and use it to detect for collisions. Overlapping objects will continue to move until it found a suitable location. For pedestrians, we opted for 2D cutouts placed among the street using a very thin collision box. The ratio of vehicles and pedestrians is roughly 1:3. The directional light in the scene is also randomly changed per image, so to make sure our dataset is not prone to shadow and light variances. Post processing image effects such as bloom, which blurs bright regions of an image to mimic bright light is also used for photo realism.

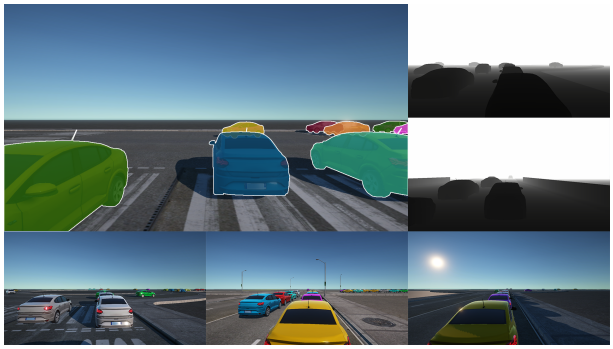

Fig. 6: The Openroads subset of RGSD, which contains RGB, depths and annotations of amodal and visible masks.

After the scene is set up, the camera will render four passes. The RGB pass - objects with proper lighting and albedo textures. The Depth pass, where a fragment shader will compute linear depth between 0 and 50 meters using geometric data. Unlike other depth datasets like KITTI which uses LiDAR, our ground truth depth does not contain gaps and is pixel accurate, this significantly improves the quality of our ground truth data. The final two passes are modal and amodal masks of each generated object, which is generated by assigning an unlit color to the target geometry, while other objects only effect it through occlusion for visible mask capture.

While the street scenario focuses on Pedestrians and few cars in a complex environment, the open roads scenario generates data with lots of vehicles. A simple road network in the form of a graph is established. Vehicles will move through this road network, simulating traffic and road scenarios. Our camera follows one of these vehicles, for every random interval between 1 and 2 seconds, the camera will initiate a scene check to determine is enough vehicles are infront of it for valid data capture.

For 3D object detection, we also created a RGSD-LIDAR dataset to simulate point clouds, especially scenarios with low point cloud density. Using Blender and Open3D, we created low-poly triangulated meshes of cars, pedestrians and cyclists (Figure 7). We randomly place objects, similar to the methods above, then scatter points on the surface of the mesh to simulate LIDAR collisions. The scene is made to mimic the Kitti dataset, where the observer is in the middle of the road, with vehicles and other

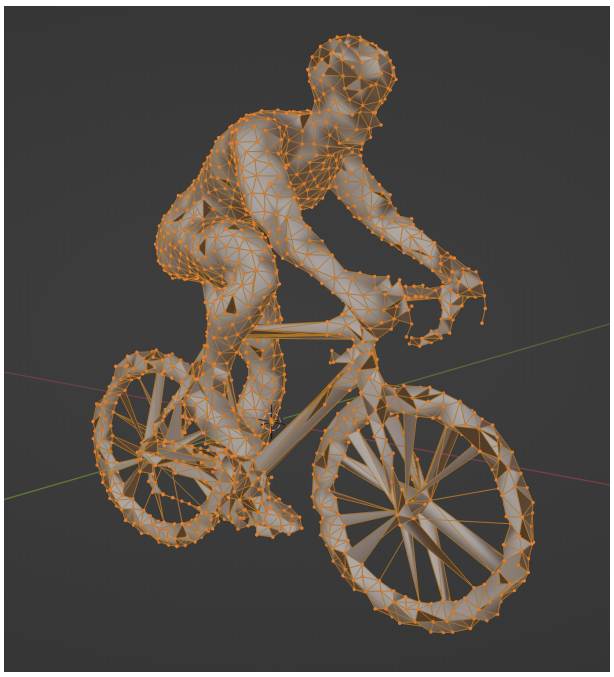

Fig. 7: Triangulated low-poly model of a cyclist.

objects distributed on the side. We deliberately reduce the point clouds to simulate scenarios at the edge of a LIDAR sensor's range, as seen in Figure 8.

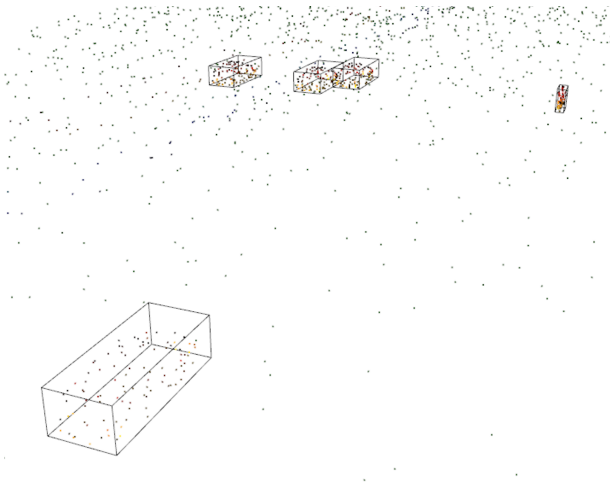

Fig. 8: Synthetic point clouds with extremely low density

3.3 Depth

Monocular Depth Prediction To predict amodal instance segmentation on datasets like KINS which do not provide depth data, we rely on a pretrained monocular depth prediction module. Our model supports monocular inference by using a monocular depth prediction module. We used the BTS network [14] with a backbone trained using ResNet50 on the KITTI Eigen Split depth dataset. We also trained the BTS network [14] on our own DAIS-RGSD dataset.

This monocular prediction module is inserted before the actual backbone. By doing this, our network is able to predict a depth map of our RGB input, and send it to the module. The depth information is also passed to the RoI Head module for occlusion edge and amodal proposal calculations.

For input RGB image $\mathbf{I}$, we define monocular depth prediction as f_d^m. For raw depth map $\mathbf{D}^r$:

$$\mathbf{D}^r = f_d^m(\mathbf{I}) \tag{5}$$

LIDAR Depth Projection The Kitti dataset provides depth maps that were projected from LIDAR point cloud. However, the point clouds are sparsely distributed. We created a dense depth map using a method that samples neighboring points [31] to create a dense depth map. We project and fill point clouds $\mathbf{\Psi}$ to raw depth map $\mathbf{D}^r$.

Depth Normalization Depth values are in an entire different representation than RGB values within the range of 0-255. Our model normalizes $\mathbf{I}$ to $\mathbf{I}^n$ using standard deviation and mean. Notably:

$$\mathbf{I}_{ij}^n = \frac{\mathbf{I}_{ij} - \mathbf{I}_\mu}{\mathbf{I}_\sigma} \tag{6}$$

For $i = 1, \ldots, \mathbf{I}_H$, $j = 1, \ldots, \mathbf{I}_W$. Where $\mathbf{I}_\mu$ and $\mathbf{I}_\sigma$ denotes the mean and standard devitation of $\mathbf{I}$ respectively.

The same normalization is applied over depth $\mathbf{D}$, but its exponential depth value is first linearized to achieve linear depth:

$$\mathbf{D}_i j = log_{10}(\mathbf{D}_i^r j) \tag{7}$$

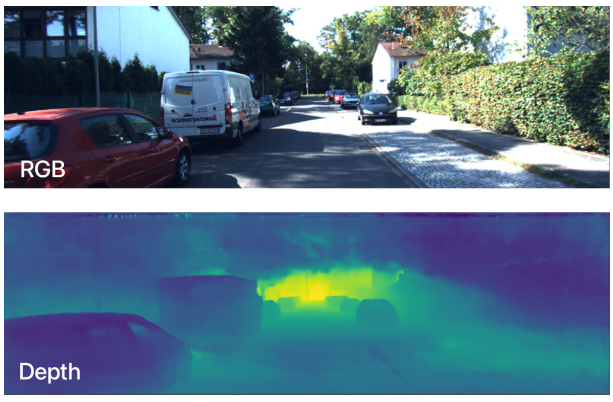

Fig. 9: Monocular depth prediction from a single RGB image from the KITTI dataset, the model used the BTS Network with a DenseNet161 backbone.

It is not possible for depths to be normalized on a per object basis, as that would require the complete segmentation of objects within the image accurately. Since we want the network to be entirely

depth-aware from the beginning, other normalization methods have to be used. The approach using visible regions to normalize depth per object however, is used in the Edge Occlusion refinement head.

Although it is not possible to completely remove object distance variance entirely without object data, we can significantly reduce its impact using a Regional Depth Normalization scheme. The scene is being split into subregions occupying a slice of the full distance, the size of a subregion is called the step size or $\mathbf{D}_s$. We use the floor function to acquire the subregion of any given depth:

$$\mathbf{D}_{ij}^{n} = \mathbf{D}_{ij} - \left\lfloor \frac{\mathbf{D}_{ij}}{\mathbf{D}_s} \right\rfloor \times \mathbf{D}_s \tag{8}$$

Where D_i represents the depth value of a single pixel, D_ir represents the base reference distance / subregion this pixel belongs to. We obtain the results shown in Fig 10 B, although the floor function divides the image into subregions, there exists a hard edge between regions. For this, we introduce a soft floor function Ξ:

$$\Xi(x) = x - \frac{\sin(2\pi x)}{2\pi} \tag{9}$$

Where x is rounded down to the greatest integer less than x with a smooth interpolation between integers. We use this function to pass in $\frac{\mathbf{D}_{ij}}{\mathbf{D}_s}$ as x, obtaining the result in Fig 10 C.

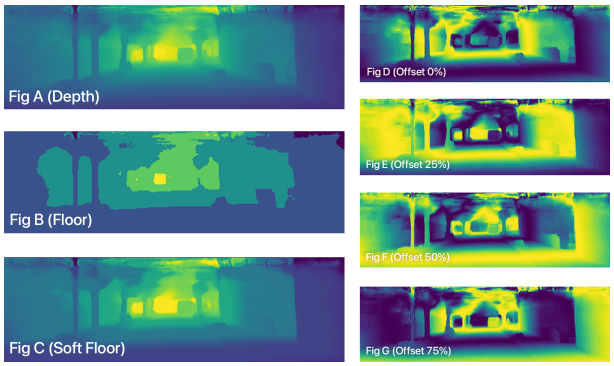

Fig. 10: Regional depth normalization pass.

Although this method greatly decreases variance caused by difference in distances, regional normalization still means that objects will have some variance in distance as seen in Fig 10 D. This could be minimized by decreasing the step size, but it will make the data less perceivable to the backbone and feature extraction. To address this, we introduce an offset value, which creates multiple regional depth maps at different global offsets. We specify the number of offset maps as $\mathbf{D}_m$, which in our setup is 4 by default. Each offset map should have a different offset value, which is obtained simply with $\frac{\mathbf{D}_s}{\mathbf{D}_m}$. This means we

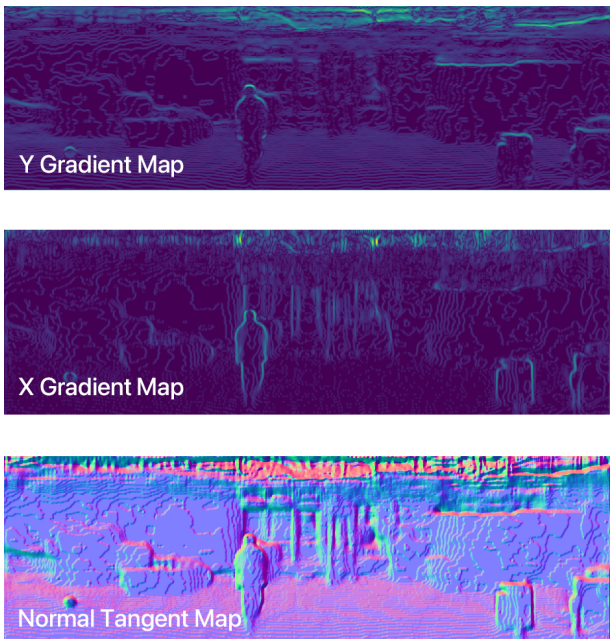

Fig. 11: Normal tangent information obtained using the same depth map in the previous figure. The gradient maps show the raw outputs of a 5x5 Sobel filter over $\frac{d\mathbf{D}}{dx}$ and $\frac{d\mathbf{D}}{dy}$. The final normal tangent map uses a hyper-parameter $\frac{d\mathbf{D}}{dz}$ of 0.7.

are creating $\mathbf{D}_m$ different regional depth maps over equal intervals between the step size $\mathbf{D}_s$. This produces multiple regional depth maps as shown from Fig 10 D to Fig 10 G. A $\mathbf{D}_m$ of 4 is suitable as each object now has a representation of its regional depth in roughly all possible ranges. The remaining variance in distance can be ignored due to its insignificance. This makes individual objects almost invariant to distance, as it includes a normalized depth representation of itself in ranges [(0, 0.25), (0.25, 0.5), (0.5, 0.75), (0.75, 1)] approximately. This means the the backbone will receive 4 additional channels as depth inputs, for each regional depth map. Experiments on the effect of various $\mathbf{D}_m$ and $\mathbf{D}_s$ on the quality of object detection and mask prediction will be conducted.

3.4 Normalized Tangent Map

Depth information however, has extreme variance shifts by nature. Fundamentally, RGB values represent the intensity of each primary color; Depth values represents distance from the camera, this poses a problem, as two objects of the same orientation, shape and feature at two distances will produce completely different sets of depth values. Thus the problem of normalizing global depth to match the RGB input is ill-posed, as there is no perfect way to normalize the depth of every object prior to object extraction and segmentation. Which poses a problem in using depth information for backbone feature extraction.

A way to remove distance variance of depth information is to compute normal tangent maps. Normal tangent maps represent the tangent of a surface at a specific pixel in the form of a normalized vector, where its xyz component translates to RGB respectively, creating an image of 3 channels. Tangent normal maps have been used extensively in computer graphics as it is an easy way of representing 3D surfaces without actual geometry. In our case, we essentially obtained 3D features of an image with depth invariance. Tangent normal maps are also represented in RGB, which our network backbone is already designed and tested for.

To compute tangent normal maps, we must first obtain the partial derivatives of every pixel. Because depth data is discrete, we can only estimate its gradient at a given pixel. We use the Sobel–Feldman operator, a discrete differentiation operator that produces approximations of gradients. For an arbitrary kernel size of N, we define Sobel filter $\mathbf{S}^N$ as:

$$\begin{bmatrix} \mathrm{S}_{11} & \mathrm{S}_{12} & \mathrm{S}_{13} & \dots & \mathrm{S}_{1N} \\ \mathrm{S}_{21} & \mathrm{S}_{22} & \mathrm{S}_{23} & \dots & \mathrm{S}_{2N} \\ \mathrm{S}_{31} & \mathrm{S}_{32} & \mathrm{S}_{33} & \dots & \mathrm{S}_{3N} \\ \mathrm{S}_{41} & \mathrm{S}_{42} & \mathrm{S}_{43} & \dots & \mathrm{S}_{4N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{S}_{N1} & \mathrm{S}_{N2} & \mathrm{S}_{N3} & \dots & \mathrm{S}_{NN} \end{bmatrix} \tag{10}$$

Where for S_{yx} we denote j and i as its vector from the center o:

$$j = y - \frac{N-1}{2} - 1 \tag{11}$$

$$i = x - \frac{N-1}{2} - 1 \tag{12}$$

Using i and j, we can formulate a general equation for sobel filters of any size N:

$$\mathbf{S}^N_{x_{ij}} = \frac{i}{i^2 + j^2} \tag{13}$$

$$\mathbf{S}^N_{y_{ij}} = \frac{j}{i^2 + j^2} \tag{14}$$

Using this equation, we obtain for instance the default 5x5 Sobel filter in the X direction, $\mathbf{S}^5_x$, used for normal tangent calculations:

$$\begin{bmatrix} -5 & -4 & 0 & 4 & 5 \\ -8 & -10 & 0 & 10 & 8 \\ -10 & -20 & 0 & 20 & 10 \\ -8 & -10 & 0 & 10 & 8 \\ -5 & -4 & 0 & 4 & 5 \end{bmatrix} \tag{15}$$

It is multiplied by a constant factor of 20 to obtain integer values.

By conducting convolution using the Sobel filter on the depth map, we obtain the following vector:

$$\mathbf{T}_i = \begin{bmatrix} -\mathbf{D}_{xi} \\ -\mathbf{D}_{yi} \\ \mathbf{D}_z \end{bmatrix} \tag{16}$$

Where i represents a single pixel, $\mathbf{T}_i$ represents the tangent at pixel i, $\mathbf{D}_{xi}$ and $\mathbf{D}_{yi}$ represent the approximate gradient in x and y direction at pixel i. $\mathbf{D}_z$ is a hyper-parameter that represents the magnitude of the Z component, this effects how sensitive the final tangent is with respect to change in $\mathbf{D}_z$ and $\mathbf{D}_x$. There is no specific value for $\mathbf{D}_z$ because the strength of both the normal map and the depth map which it derives from is arbitrary, thus $\mathbf{D}_z$ is also arbitrary. The higher $\mathbf{D}_z$ is, the less sensitive our tangent is to change and vice versa. For our model, we used a $\mathbf{D}_z$ of 0.8.

$\mathbf{T}_i$ can be considered the raw tangent of a given pixel, the final normal tangent map is obtained by calculating the unit vector of $\mathbf{T}_i$. This is done by dividing each vector by its Euclidean norm:

$$\hat{\mathbf{T}}_i = \frac{\mathbf{T}_i}{|\mathbf{T}_i|} = \frac{\mathbf{T}_i}{\sqrt{{\mathbf{D}_{xi}}^2 + {\mathbf{D}_{yi}}^2 + {\mathbf{D}_z}^2}} \tag{17}$$

Where $\hat{\mathbf{T}}_i$ is the normalized tangent vector of pixel i.

3.5 Depth-Aware Feature Backbone

Our backbone is based on ResNet50. It received with RGB input in the dimension of (N, C_{in}, H, W), or (1, 3, 2656, 800) by default. The monocular depth prediction module produces a tensor of size (1, 4, 2656, 800) when using a subregion $\mathbf{D}_s$ of 4. We define the feature extractor operation as f_r, obtaining feature map $\mathbf{F}$:

$$\mathbf{F} = f_r(cat(\mathbf{I}^n, \mathbf{D})) \tag{18}$$

We simply concatenate depth $\mathbf{D}$ as addition channels using the matrix operator *cat*, which results in a tensor of size (1, 7, 2800, 400) for $\mathbf{I}$+$\mathbf{D}$. To accommodate for this increase in channel count, the first layer of our network is also modified. To best preserve the pre-trained weights, we preserve the weights for the original three channels, and repeat it for the subsequent channels. The same applies for the normal tangent input, which 3 channels.

Because objects can occur at various different sizes in instance segmentation, feature maps that are either too complex or simple may have negative effects on classification and segmentation The backbone thus generates feature maps at different dimensions as implemented in Mask R-CNN. Since we require depth as an explicit information in proceeding layers, a separate set of depth maps is generated alongside the feature maps, it is scaled using Bilinear interpolation.

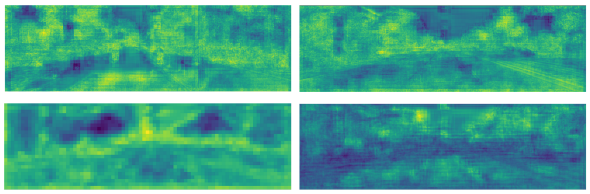

Fig. 12: The backbone extracts features from an image, usually hundreds of channels. Each channel is sensitive to one or multiple visual features. Multiple feature maps of various resolutions (extracted at different layers of the backbone) and channels are shown below.

3.6 RPN and ROI Pooling

For the regional proposal network, we directly target training towards predicting amodal bounding boxes. We define the generation of valid 2D proposals as f_α, valid 2D proposals as $\{\mathbf{P}_i^\alpha\}_{i=1}^N$:

$$f_\alpha(\mathbf{F}) = \{\mathbf{P}_i^\alpha\}_{i=1}^N \tag{19}$$

Where loss $\mathcal{L}_{cls}(\mathbf{P}, \hat{\mathbf{P}})$ and $\mathcal{L}_{reg}(\mathbf{P}, \hat{\mathbf{P}})$ calculates classification and box regression loss respectively, which is the same as Mask-RCNN[2]. $\hat{\mathbf{P}}$ is the set of ground truth amodal proposals

Region of Interest Pooling is the extraction of the bounding box portion of the feature map for evaluation. Mask R-CNN used an heuristically defined equation to specify which feature map to use:

$$f_r^i(\epsilon) = \left\lfloor 4 + log_2\left(\frac{\sqrt{\epsilon}}{224}\right)\right\rfloor \tag{20}$$

Where ϵ is the area of the proposal box, $f_r^i(\epsilon)$ is the feature map index.

We define f_α^p as the pooling operator, $\mathbf{D}^p$ as the pooled depth map, and $\mathbf{F}^p$ as the pooled feature. For proposal $\mathbf{P}_i$:

$$\mathbf{D}^p, \mathbf{F}^p = f_\alpha^p(\mathbf{F}, f_r^i(\mathbf{P}_i^\epsilon), \mathbf{P}_i) \tag{21}$$

Since depth information is required for occlusion edge calculation on a pixel to pixel accuracy, depth maps are also scaled to the same resolutions as the feature maps, and pooled alongside feature maps.

3.7 Visible and Amodal Coarse Mask

After obtaining pooled feature map $\mathbf{F}^p$ and $\mathbf{D}^p$, we directly estimate visible mask V and amodal coarse mask A_c, using the same mask head as Mask-RCNN. We obtain visible mask loss and amodal coarse mask loss $\mathcal{L}_v$ and $\mathcal{L}_{ac}$ respectively.

3.8 Occlusion Edge Refinement[1]

If the visible region of an object is touching another region that is in front of it, it is highly likely that the object is being occluded in this direction. To best represent occlusion orders relative to the object, we calculate the pooled normalized depth $\mathbf{D}_n^p$ as:

$$\mathbf{D}_s^p = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}\mathbf{D}_i^p & \mathrm{V}_i = 1\\ 0 & \mathrm{V}_i = 0\end{cases} \tag{22}$$

$$\mathbf{D}_n^p = \mathbf{D}^p - \mathbf{D}_s^p \tag{23}$$

For $N = H \times W$

The occlusion edge module predicts segments of a visible mask that is possibly being occluded instead of being its natural boundary. We use the intersecting edge between amodal and visible ground truths masks to generate the ground truth for occlusion edge. A Gaussian blur is applied over this edge for relative tolerance. Loss is calculated as:

We define the occlusion edge refinement as f_o, which receives coarse amodal mask A_c, pooled normalized depth $\mathbf{D}_n^p$, and outputs refined amodal mask A_r. The occlusion edge refinement loss, $\mathcal{L}_{oe}$, is:

$$\mathcal{L}_{oe} = \sum_{i=1}^{H\times W} \mathcal{L}_{BCE}(f_o(\mathrm{A}_{ci}, \mathbf{D}_{ni}^p),\ \hat{\mathrm{A}}_i) \tag{24}$$

$\hat{\mathrm{A}}$ denotes the ground truth amodal mask.

Notably, $\mathcal{L}_{BCE}$ denotes the Binary Cross Entropy loss:

$$\mathcal{L}_{BCE}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\ \hat{y}_i + (1-y_i) \cdot \log\ (1-\hat{y}_i) \tag{25}$$

We also had different implementations of the occlusion edge module as seen in Fig 14. The first implementation is an micro FCN network, this network deals normalized depth information as a classification and segmentation problem. The second implementation copies the structure of the amodal mask head, with deep convolutions to extract and use features from the normalized depth map for occlusion edge prediction. The third implementation is a simple 1x1 convolution that simply infers potential areas of occlusion based on the depth value on that pixel. The setup can be selected based on training configurations, the best will be chosen during the experimentation phase.

[1] Occlusion Edge and Edge Occlusion can be used interchangeably, OE and EO stands for the same module.

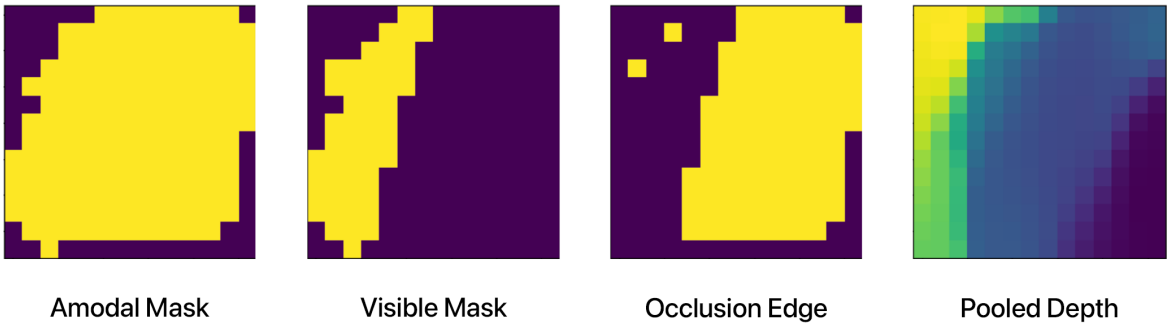

Fig. 13: Ground truth (ideal) masks at occlusion edge. Occlusion edge is focused on inferring the unseen based on visible and depth information. The pooled depth data shows strong feature resemblance to the occlusion edge, it is easy to infer these regions are potentially occluded.

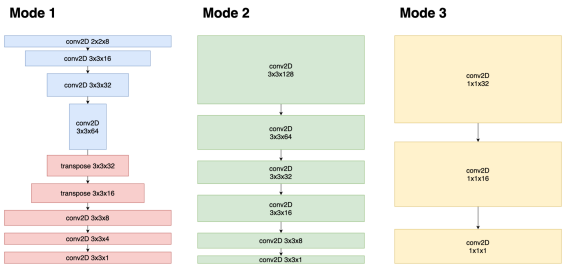

Fig. 14: Three different setups for the Edge Occlusion module.

3.9 3D Shape Prior Refinement

The fundamentals of predicting the unseen region is based o n the seen regions and features to look for similar seen shapes, or shape prior for amodal completion. This is already implicitly achieved by the amodal prediction mask head. Weights are already trained to correspond visible features to amodal shapes, this is why the Mask-RCNN network can make reasonable amodal predictions without any modification - shape memorization is implicitly achieved during training. The shape prior refinement process, first introduced by Xiao at el. [6] further exploit this principle by explicitly storing and matching seen shapes for refinement.

The shape prior refinement process can be simplified into a problem of finding and matching different seen shapes. It is much easier to find similarities between shapes if it could be quantified. This is achieved through embeddings, which converts complex and high-dimensional data into simple, low-dimensional representations. This method is widely used in machine learning, because simpler data - like numbers and vectors, can be easily compared. It is very difficult conversely, to directly compare similarities between different shapes. A simple network consisting of multiple convolutional and deconvolutional (transposed convolutional) layers. The embedding is obtained in the middle between the convolutions (down-sampling) and the deconvolutions (up-sampling). By formulating a training method where given an input shape should result in the exact same shape on output, we are essentially training a network that is capable of encoding a low-dimensional representation of shapes, and conversely decode it back into the shapes. We can simply split the model into decoding and encoding functions for us to encode and decode embeddings. We define the encoder as f_ϕ, the encoder loss as $\mathcal{L}_\phi$:

$$\mathcal{L}_\phi = \mathcal{L}_{BCE}(cat(\mathrm{V}_r, \mathbf{D}_n^p), f_\phi(cat(\mathrm{V}_r, \mathbf{D}_n^p))) \quad (26)$$

We improve upon the shape prior refinement process by adding localized depth information as an additional input channel. By adding localized depth values into the encoding process, embeddings generated by the encoder effectively contains 3D surface features of the shape, which means that it now retains 3D memories of shapes. This is useful as it means that shape refinement based on shape prior can be significantly more specific, as a higher dimension allows for clearer distinction between different seen shapes. Shapes that appear to be similar in 2D might be vastly different when it is represented in 3D, which is something the original 2D shape refinement process is insensitive to.

We define the output embedding of the encoder f_ϕ as Φ. We define the codebook which stores previous embeddings as Δ, the search function finding n nearest embeddings as f_Δ. The nearest embeddings as a collection $\left\{\Phi_i^\Delta\right\}_{i=1}^n$.

$$\Phi^\Delta = f_\Delta(\Phi, n) \in \Delta \quad (27)$$

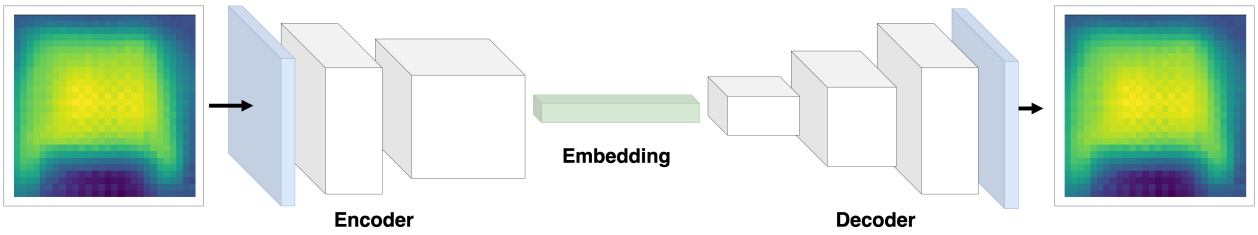

Fig. 15: The encoder-decoder network setup, embedding is obtained at the middle of the network. Training conducted by setting the loss as the difference between the output and the input, in which case they should be the same.

After obtaining n nearest shapes, we define 3D shape-prior refinement loss $\mathcal{L}_{sp}$, shape-prior refinement function f_{sp} and shape-prior refinement amodal mask A_{sp} as:

$$\mathcal{L}_{sp} = \mathcal{L}_{BCE}(f_{\phi}(\Phi^{\Delta}, \mathrm{A}_r), \hat{\mathrm{A}}) \tag{28}$$

It is expensive to compare similarity against every single embedding, many shapes are also extremely similar. We employ K-means clustering to cluster similar embeddings. This way, we only have to compare an embedding against every cluster in the codebook to obtain the closest matching shape prior.

3.10 Point Cloud Aggregation

Most 3D networks conduct convolution, point cloud aggregation or other feature extraction methods for the entire scene. However, because the 2D RPN has already produced highly accurate 2D proposals, it can be projected onto the point cloud. Then, only the proposal related regions will be aggregated and extracted.

Because we have obtained a depth map that has pixel to pixel correlation to our image $\mathbf{I} \rightarrow \mathbf{D}$, we can simply project our proposals onto point clouds, and filter the neighboring points.

To project depth images to point clouds, we would need to obtain calibration matrix M of the used camera.

$$\mathrm{M} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{29}$$

Datasets such as Kitti contains calibration matrices, we could also calculate one using OpenCV, by taking a picture of a checkerboard using the camera to be calibrated, as shown in Figure 17. For $i = 1, \cdots, H$; $j = 1, \cdots, W$. We calculate the point cloud (relative 3D) coordinate of each proposal pixel $\mathbf{\Psi}^{\mathbf{P}}$ of our depth map as:

$$\mathbf{\Psi}^{\mathbf{P}}_{ij} = \begin{cases} x &= \mathbf{D}^r_i j \\ y &= \dfrac{(j - c_x) \times z}{f_x} \\ z &= \dfrac{(i - c_y) \times z}{f_y} \end{cases} \tag{30}$$

We can then select the attentive point clouds $\mathbf{\Psi}^{\alpha}$ from $\mathbf{\Psi}$ with a hyper-parameter distance of λ.

$$\begin{aligned} &\mathbf{\Psi}^{\alpha} = \{\alpha \in \mathbf{\Psi} \mid \\ &\sqrt{(\alpha_x - \mathbf{\Psi}^{\mathbf{P}}_{c\ x})^2 + (\alpha_y - \mathbf{\Psi}^{\mathbf{P}}_{c\ y})^2 + (\alpha_z - \mathbf{\Psi}^{\mathbf{P}}_{c\ z})^2} \\ &< \alpha\} \end{aligned} \tag{31}$$

Where $\mathbf{\Psi}^{\mathbf{P}}_c$ is the closest proposal point to $\alpha \in \mathbf{\Psi}$.

For aggregation of filtered point clouds, we employ a similar method to that of PV-RCNN [29].

$$\Theta^v_i = \max\{G(\mathbf{\Psi}^{\alpha}_{s(i,v)})\} \tag{32}$$

Where v denotes the extraction scale (how large the receptive field is), i denotes a voxel in voxel grid Θ^v, $\mathbf{\Psi}^{\alpha}_{s(i,v)}$ denotes the points within range of voxel Θ^v_i, G denotes a simple perceptron network for encoding the point clouds, while max stands for a standard max-pooling operator. The voxels extracted at different scales are similar to that of the ResNet backbone used for 2D Mask-RCNN, the final pooling operation will select the voxel size that best fits the RoI size.

3.11 3D Detection based on BEV

We convert the final layer of the voxel extraction grids to a BEV map, Υ, which can be seen simply as a multi-channeled image. Thus, R-CNN proposal methods over 2D maps can be used. For which Υ creates proposals $\{\mathbf{P}^{\beta}_i\}^N_{i=1}$. We denote f_{β} as the regional proposal network for Υ, which is identical to f_{α}. We define BEV classification loss and 2D BEV box regression loss as $\mathcal{L}_{\beta cls}(\mathbf{P}^{\beta}_{cls}, \hat{\mathbf{P}}^{\beta}_{cls})$ and $\mathcal{L}_{\beta reg}(\mathbf{P}^{\beta}_{reg}, \hat{\mathbf{P}}^{\beta}_{reg})$.

Fig. 16: Embeddings of a batch of 3D shapes. Each embedding vector has a dimension of 1x392. The image shows a batch of embeddings with dimensions 32 x 392, which means it consists of 32 individual embeddings.

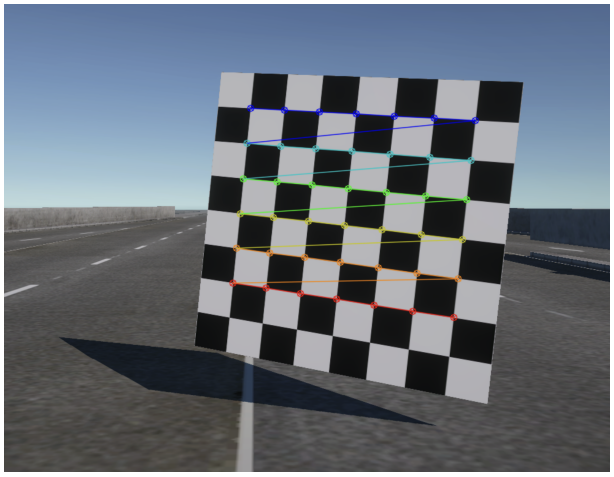

Fig. 17: Calibration of camera using checkerboard on the RGSD Openroads dataset.

Because the 2D proposal projection and attentive point cloud filtering has already limited voxels to regions where objects are very likely to exist, the bird eye view can be seen as refining existing proposals rather than proposing new ones. We are also able to significantly increase the amount of proposals generated near objects than uniformly distributed.

Because objects are horizontal to the ground in the scenario of autonomous driving, only one axis of rotation need to be accounted for. While the BEV representation already gives width and length of bounding boxes, the height can also be easily deduced using the pooled voxels. Thus, 2D proposals on BEV representations are essentially 3D. To further fine tune our predictions, we use the 3D proposals to pool voxel features from the appropriate voxel size. Traditional methods for box regression with residuals is used for 3D proposal fine tuning using pooled features. A 3D box refinement loss is defined as $\mathcal{L}_{\gamma reg}(\mathbf{P}^{\beta}_{reg}, \hat{\mathbf{P}}^{\beta}_{reg})$

3.12 Loss

The total loss of DAISnet $\mathcal{L}_{\text{dais}}$ (amodal instance segmentation only) training is:

$$\mathcal{L}_{\text{dais}} = \mathcal{L}_{cls} + \mathcal{L}_{reg} + \mathcal{L}_{v} + \mathcal{L}_{ac} + \mathcal{L}_{oe} + \mathcal{L}_{\phi} + \mathcal{L}_{sp} \quad (33)$$

For ESAAN, the loss $\mathcal{L}$ is:

$$\mathcal{L} = \mathcal{L}_{\text{dais}} + \mathcal{L}_{\beta cls} + \mathcal{L}_{\beta reg} + \mathcal{L}_{\gamma reg} \quad (34)$$

4 Experiment

The aim of our experiment is to evaluate and prove that incorporating depth information into amodal instance segmentation tasks can yield better results. Because we have introduced a multitude of ways in which depth information can be used, an ablation study will be carried out that evaluates the impacts of each module on performance and different per module configurations.

4.1 Datasets

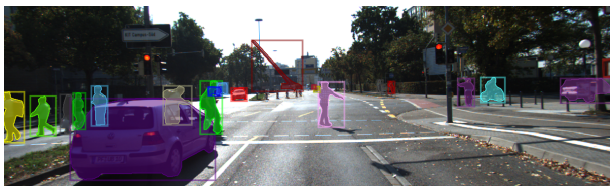

Fig. 18: A single image from the KINS dataset, complete with bounding boxes and amodal masks.

KINS Dataset For training and evaluation of our model, we used the KINS dataset [32] which provides amodal and visible annotations of objects. The KINS dataset is based upon the KITTI dataset, which provides the KITTI Eigen Split depth dataset for which our monocular depth prediction module is trained upon. This ensure the quality of depth predictions while maintaining RGB as the only ground truth input for inference. There is a total of 7474 images for training and 7517 images for evaluation. Annotations in the KINS Dataset are stored in vector masks, it is rasterized into bitmasks prior to training. The KINS Dataset is widely used in amodal instance segmentation research, many models were trained and evaluated using KINS. This provides a common benchmark for comparasion between different methods.

RGSD Dataset All synthetic data are captured in 1280 x 720 pixel resolution on an M1 Max using Unity Engine. The synthetic training dataset comprises of 10000 city scenes, 10000 street scenes and 20000 open road scenes. The validation dataset contains 1000 sets, where 400 are open road and 300 for both street and city. It includes the RGB image, the ground truth depth, amodal and visible masks of objects stored in a bitmask format.

4.2 Model Set-up

Depth Prediction Module Training of the Monocular Depth Prediction Module for KITTI is done using a single RTX3070ti with 46375 iterations with a batch size of 2 over 14 epochs. The synthetic dataset is trained over 19000 iterations with a batch size of 2 over 14 epochs. Both models used a learning rate of 0.0001 with 0.01 linear decay. To optimize training speed and memory usage, the depth prediction module is loaded in inference mode during amodal instance segmentation, it does not take part in training. Although depth information is inferred from RGB inputs, a single image always produces the same result. A cache mechanism is used to reload depth data from disk if it has been predicted before. This reduces training time from 10 hours to 3 hours, more than 3 times quicker.

Amodal Instance Segmentation The project is setup and implemented using a modified version of Detectron2 [33], which has a Mask-RCNN implemented using Pytorch and CUDA. Our project was trained on Ubuntu 20 using Python 3.6, Pytorch 1.4 and CUDA 10.1. Training is conducted on single Nvidia Tesla V100 instances. We used the ResNet-50 weights trained on ImageNet Classification as our pretrained weight for our feature extraction backbone, we also used weights from monocular depth prediction, as well as random weights. Depending on the configuration, data like depth and normal tangent will effect the input channel count for feature extraction. To preserve the pretrained weights, the first convolution layer will expand and repeat its three channel weights. For an input channel count of 9, the pretrained weights will be repeated three times. There are a total of 47979 iterations, the KINS dataset is further post-processed with random crop, rotation and color filters to enlarge the dataset.

Learning Rate A base learning rate of 0.0025 was chosen for training, this works particularly well with the pretrained ImageNet backbone. However, for randomly initialized weights, the loss becomes infinite or not defined (due to division by zero) after a few hundred to thousand iterations depending on the configured complexity. A lower learning rate solves this problem, but it also means the model might not be trained to its best performance. Learning rate warmup is a typical approach for solving high loss training. Constant warmup specifies a lower learning rate for a given number of iterations. Linear warmup increases the learning rate linearly from zero for a given number of iterations. We observe that during training, both constant and linear warmup methods are not suitable for our scenario; Loss either converge too slow, or the initial learning rate remains too high. We introduce a smooth warmup method, which increases the learning rate at a given power:

$$lr_w = \left(\frac{i}{t}\right)^p \times f \times lr_f \tag{35}$$

Where lr_w is the warmup learning rate, lr_f is the final learning rate, i is the current iteration, t is the warmup iteration count, p is the warmup power, and f is the constant warmup factor.

For random weights, we used $lr_f = 1.0$, $p = 1.3$, $t = 15000$.

Post Processing Post processing of training data is commonly used to increase volume of the training set without actually acquiring additional data. The KINS dataset provides 7000 images for training, which is significantly lower than datasets used for instance segmentation and object detection for a task that is overall more difficult. A post processing layer creates variants of the same image, such as flipping the image horizontally, changing the contrast, and randomly crop the image. Post Processing is however computationally expensive, it drastically increases training time. For the ablation study, post processing will be disabled for all tests. We notice a 15% drop in performance with post processing disabled.

4.3 Synthetic Augmentation

For ESAAN with 3D object detection, our main concern is finding out if appending additional synthetic data improves the accuracy of our model. Using our RGSD-LIDAR subset, we will train ESAAN on 8000 unique iterations with 50 epochs. Then, we will train the model using the standard Kitti dataset, but keeping the initial weights optained through synthetic training. We will compare our model with other networks to see if it enhanced 3D detection performance.

4.4 Metrics

Average Precision, or AP, is a method for evaluating performance and accuracy of object detection and segmentation problems. For our evaluation, we used Average Precision, Mean Average Precision and Average Recall to evaluate performance of different models.

The confusion matrix maps out all possible combinations of results as shown in Fig 19. If a result is predicted, it is considered positive; If a result is not predicted, it is considered negative. If the prediction is correct, it is considered true, otherwise it is considered false. Thus, we have True Positive TP, False Positive FP, True Negative TN and False Negative FN.

An important metric in determining Bounding Box precision is the Intersection over Union, also known as IoU:

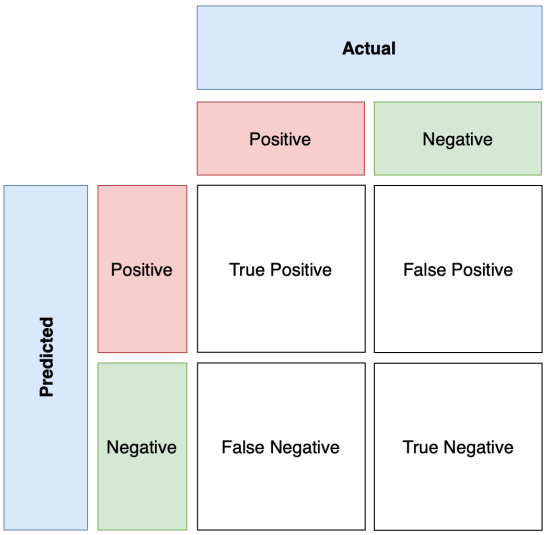

Fig. 19: Confusion Matrix

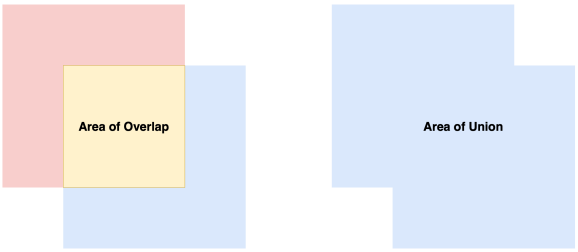

Fig. 20: Area of Overlap vs Area of Union

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{36}$$

Where A, B describes the predicted and ground truth bounding box.

IoU is used to determine whether a prediction is True or False. This depends on an IoU threshold. If the predicted IoU is above this threshold, the result is considered true. Using the confusion matrix, we can determine Precision and Recall:

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN} \tag{37}$$

Precision depicts how many predictions are correct across all predictions. It evaluates how many correct predictions are made among all predictions. Recall depicts how many correct predictions are made. It evaluates how many correct predictions are made among all correct results.

We use the area under the precision recall curve to calculate Average Precision. However, the precision recall curve is not a continuous function, so we can

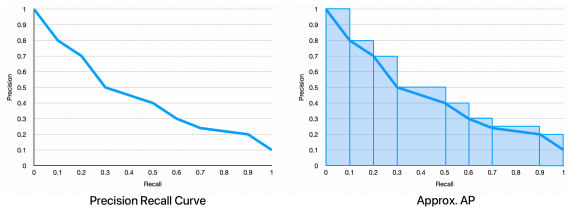

Fig. 21: Average Precision is the area under a Precision Recall Curve, which is precision over recall.

only obtain approximations of the area. One method is to use rectangles to estimate area under the curve as shown in Fig 21.

Calculation of Average Precision is usually for a single class of objects. For the representation of all classes, we use Mean Average Precision (mAP):

$$mAP = \frac{1}{N} \sum_{i}^{N} AP_i \tag{38}$$

Where N is the number of classes, AP_i is the average precision of a single class.

Similarly, Average Recall (AR) is twice the area under a Recall over IoU curve. Similar approximation methods are used to calculate the area.

For both AR and AP, the number behind states the IoU threshold for determining true and false results. AR_{50} represents an average recall with 0.5 IoU threshold or 50%.

Comparisons To further evaluate our network's performance, we will compare our best setup with other commonly used methods such as Mask R-CNN and PANet. All networks used the KINS dataset for evaluation. Post processing will be applied to our final setup during training.

5 Evaluation and Ablation Study

We conducted an ablation study on our network by testing out different setups and combinations of features. The aim of our study is to evaluate how depth information improves amodal instance segmentation performance, and what are the most effective methods of using the depth information. The complete table of results is shown in Table 2. R, D, N O and S stands for RGB, Depth, Normalm Occlusion Edge and 3D shape prior respectively. A checkmark ✓symbol denote this component is enabled for the given row, other special characters represent the use of a variation of this component. Depth with N for example, means depth without normalization. Abbreviations will also be used to describe configurations, the order of components follows that of

Fig. 22: Amodal instance segmentation results from ImageNet weights using RGB and occlusion edge refinement.

the table. For instance, random weights with RGB, depth without normalization, and normal tangent with medium sized filter is described as Random R+DN+NM.

5.1 Backbone

Our feature extraction backbone used the ResNet50 architecture, by using a standard backbone, we can use transfer learning to improve our network's performance. A separate and complete ablation study is done for both weights. We are using random weights because we want to evaluate the pure performance improvements from using depth information. A backbone with pretrained weights that are obtained from other tasks may effect the results of our study negatively. Conversely, most amodal instance segmentation networks utilize a pretrained backbone feature extractor, which results in significant performance boost from using random weights. We will evaluate how well depth information copes with pretrained weights that is not depth aware.

As seen in Table 2, there is around a 30% improvement in accuracy from random weights to ImageNet weights. Imagenet R+N has an AP of 22.83, while Random R+N has an AP of 16.52 only. This demonstrates that even for weights that were not trained on depth information still yields better results than random weights.

5.2 Depth

The most basic approach towards incorporating depth in amodal instance segmentation is by feeding it through the feature extraction backbone. By doing this, the entire network will conduct instance segmentation and amodal segmentation tasks using depth-aware feature maps.

Usefulness By concatenating normalized depth information consisting of 4 channels by default to the RGB input, there is a 25% increase in AP from Random R to Random R+D. This demonstrates that using depth information does result in significant improvements for amodal instance segmentation. To further examine whether depth information can actually provide features for amodal instance segmentation, the network is trained on depth without using any RGB information. Random D is 18% lower in AP than Random R, and 35% lower in AP than Random R+D. An AP of 10.84 means that the network is still able to infer reasonable results soley from depth data. We can also conclude that depth information cannot substitute RGB data, which is logical as RGB information provides surface features that depth is insensitive to.

Impact of Regional Depth Normalization As mentioned in the methods section, a regional depth normalization pass was done over the depth information before feeding it to the feature extractor. To evaluate the impact of said normalization, we tested the model without normalization, denoted by N. For both Random R+DN and ImageNet R+DN, there was only marginal improvements in AP when using regional normalization, still this means that the model performed worse without regional normalization. Evaluation on the impact of step size and subregion intervals were also conducted. ImageNet R+D1, which represents regional depth normalization with a subregion count of 1, essentially not creating subregions. This results in a small 1% decrease in AP and other metrics. ImageNet R+DS1 represents normalization with a step size of 1, which is 2/3 less than the default step size of 3. This means depth information is far more localized and distance specific. As hypothesized in the method section, a smaller step size does lead to a drop in AP of 2%, likely caused by over-fitted local variations.

5.3 Normal Tangent

Depth information can be further processed into normal tangent maps. Normal tangent maps describe the surface tangent at a given pixel. This is different from depth information as normal tangent maps provide an explicit representation of 3D surface features, while depth only provide it implicitly to the network.

Usefulness We have to first evaluate if normal tangent information can be perceived well by the feature extraction backbone. Normal tangent information is different from depth as depth information can be represented well within in one channel. Normal tangents are represented in 3 channels, representing the x, y and z components of the unit vector at a given pixel. There is no implicit separation between RGB and xyz channels, which might also negatively effect results. ImageNet R+N yields an AP of 24.37, which is 6% higher than that of ImageNet R+D. This shows that normal information is more useful than depth information for amodal instance segmentation.

Table 1: Results on Amodal Instance Segmentation
R D N O S stands for RGB, Depth, Normal, Occlusion Edge and 3D Shape Prior
N stands for raw depth without normalization; S1 stands for regional normalization with subregion of 1; O1 stands for regional normalization with offset interval of 1; M stands for medium sized sobel filter; L stands for large sized sobel filter; V stands for visible only depth

Weights	R	D	N	O	S	AP	$AP50$	$AP75$	AP_s	AP_m	AP_l	$AR1$	$AR10$	$AR100$
Random	✓					13.20	28.02	10.20	18.21	17.86	14.48	10.51	25.10	28.76
Random	✓	✓				16.52	31.49	15.60	22.26	21.28	16.87	11.91	27.72	30.14
Random	✓	N				16.42	31.64	15.50	21.68	20.52	17.22	11.96	27.72	30.14
Random		✓				10.84	20.71	10.52	14.00	13.09	11.68	7.98	18.32	20.05
Random	✓		✓			9.55	20.10	8.02	12.20	11.74	10.08	6.72	16.98	19.03
Random			✓			11.26	21.38	10.83	14.66	13.45	11.74	8.39	18.89	20.52
Random	✓	✓	✓			16.13	30.71	15.26	21.56	20.52	17.59	11.16	26.77	29.21
Random	✓			✓		**17.19**	32.62	16.38	23.02	21.94	18.09	12.38	28.29	30.61
Random	✓	✓		✓		17.17	23.57	16.30	23.58	22.44	18.15	12.68	28.59	30.91
Random	✓		✓	✓		9.74	20.33	8.38	12.53	12.25	10.41	6.65	17.50	19.66
Random	✓	✓	✓	✓		10.11	21.34	8.50	12.99	12.37	10.61	7.30	18.46	20.59
ImageNet	✓	✓				22.83	40.77	22.97	28.54	26.53	22.17	15.94	33.41	35.36
ImageNet	✓	O1				22.19	39.90	22.26	27.97	26.22	22.36	15.76	32.67	34.64
ImageNet	✓	S1				22.35	39.93	22.49	28.20	26.39	22.42	16.00	33.04	33.04
ImageNet	✓	N				22.57	40.47	22.51	28.28	26.52	21.82	15.69	32.79	34.86
ImageNet	✓		✓			24.37	44.02	24.48	30.05	29.08	24.45	16.77	35.56	37.98
ImageNet	✓		M			17.35	32.55	16.32	22.32	20.82	16.96	12.84	26.76	28.50
ImageNet	✓		L			16.25	31.04	15.12	21.19	19.79	16.27	12.11	25.30	27.05
ImageNet	✓				✓	24.62	45.63	24.13	30.41	27.67	23.83	16.84	32.60	36.63
ImageNet	✓				V	24.01	45.58	22.92	30.14	28.49	23.58	16.88	32.55	36.62
ImageNet	✓	✓	✓			23.11	41.22	23.44	29.11	27.36	22.74	16.39	33.90	35.93
ImageNet	✓			✓		**25.41**	44.98	25.64	30.73	29.27	24.59	17.45	35.81	37.89
ImageNet	✓	✓		✓		23.00	40.80	23.48	28.59	27.02	22.60	16.23	33.65	28.15
ImageNet	✓		✓	✓		17.13	32.32	16.05	22.26	20.67	16.93	12.46	26.33	28.15
ImageNet	✓	✓	✓	✓		16.93	31.72	16.16	22.27	2.23	17.53	12.73	26.64	28.43

Accuracy and Sensitivity The normal map deduced from the depth information is only an estimate in surface tangents, it is not the physical geometrical tangents. Since the normal map information is not completely accurate, how well do different parameters effect estimations is also important. As described in the method section, different kernel sizes for the sobel filter effect how large of a gradient the normal map is sensitive to. Larger kernel sizes reflect a broader change in gradient, at the cost of less details. We experimented with ImageNet R+NM, ImageNet R+NL, which stands for medium and large kernel sizes respectively. The medium kernel has a size of 21x21, while the large kernel has a size of 71x71. Results show that there is a 29% drop in AP for the medium filter, and a further 34% drop in AP for the large filter. This means that the improvements in overall tangent accuracy does not compensate for the loss in detail. Furthermore, the backbone network ResNet50 uses convolutional kernels larger than 1x1, which increases the perceptive field of a given pixel. This may implicitly create a representation of more comprehensive gradient changes without sacrificing detail.

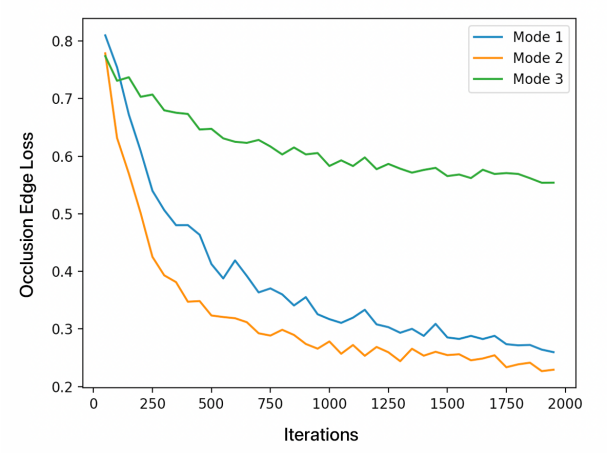

Fig. 23: Loss of different network setups for EO Refinement Layer

5.4 Occlusion Edge Refinement

Occlusion edge refinement differs from other components in that it does not affect the overall network in

feature extraction, object detection and visible segmentation. As described in the method section, there were multiple network setups for the occlusion edge refinement module. The networks were first evaluated with how well loss decreased before using the best network for amodal instance segmentation.

Network Setup Results indicate that Mode 2, which mimics the network setup of the amodal mask head, reaches the lowest loss using the least time. The eo loss of different modes over iteration is shown in Fig.23. Mode 3 yields the highest loss. Although it is hypothesized that Mode 1 which uses a micro FCN implementation should perform best, this is now the case. We believe this is due to either insufficient information, over-complicated network or improper configurations. Mode 3 performs the worst because the network is too simple, it only infers occlusion edge based on local depth values, and do not take in account of more general depth patterns.

Usefulness The EO refinement layer has the single most improvement to amodal instance segmentation compared to other configurations. Random R+O is 30% higher in AP than Random R. ImageNet R+O is 11% higher in AP than ImageNet R+D. We believe this is because the EO refinement layer has the most explicit use of depth information. Since EO refinement occurs after visible mask segmentation, it can use the visible regions of an object to conduct per object depth normalization. This is extremely useful, as any region with depth value above that of the visible maximum is guaranteed to not to be occluded, and any region with depth value lower than that of the visible minimum is likely to be occluded. Even an algorithm can infer plausible results using this information.

5.5 3D Shape Prior Refinement

The 3D shape prior refinement is denoted by S in Table 2. 3D Shape Prior Refinement combines depth information during embedding encoding and decoding. The original shape prior implementation [6] used coarse amodal masks from Mask-RCNN to conduct shape prior matching. Depth information however, is not entirely reflective of the actual object because it contains occluded regions. Depth of occluded regions of an object are not the actual depth of these objects. We differentiate these depths by denoting depth in visible regions as object-only depth. ImageNet R+S results in the second highest AP after Occlusion Edge Refinement. There is a 7% improvement in AP from ImageNet R+D baseline. ImageNet R+SV, which only uses the object only depth, yields an AP of 24.01, which is 2% less than using all the depth. We can conclude that depth in occluded regions does in fact contribute to 3D shape prior refinement. This might be because the shape prior refinement also implicitly memorize occlusion scenarios. If the same region was occluded and has a higher depth value, it is reasonable to assume the same for similar cases.

Table 2: Results on KINS dataset, average precision of amodal segmentation.
We compared our model's results against other methods. The DAISnet result was trained using ImageNet + RGB + Edge Refinement + Post Processing

Model	*AP*
Mask R-CNN [34]	24.93
PANet [32]	27.39
BCNet [35]	28.87
DAISnet (Ours)	**31.48**

5.6 Combinations

We also tested different combinations of components to see if there is additional performance improvements, such as using depth and normal information together. It is logical to assume that if both components improved performance, then the combination of said components should further improve performance. But according to our experimental results, this is not the case. We first evaluate the performance of combining depth and normal information. Random R+D+N is 3% lower in AP than Random R+D. ImageNet R+D+N is 1% better than ImageNet R+D, but 5% lower than ImageNet R+N. This shows that there is no guaranteed improvement when adding depth and normal data together. We conclude that this is because normal tangents already provide the same 3D features that depth provide, and in a more explicit way without the issues distance variation poses.

We also experimented on combining depth-aware feature extraction and occlusion edge refinement. There was marginal difference in adding depth features to occlusion edge, Random R+D+O is only 0.2% less in AP than Random R+O, while ImageNet R+D+O is 10% less in AP than ImageNet R+O. We believe this is due to potentially similar roles both components play. The depth features extracted from R+D is implicitly used for amodal attention. Since the occlusion edge refinement layer uses a similar architecture to the amodal mask head, it is likely that both modules have used depth in a similar way to predict the amodal mask. However, since the occlusion edge refinement layer uses per object normalization, it should perform better, which explains why the occlusion edge component outperforms depth feature extraction. The use of 3D shape prior and occlusion edge proves that explicit 3D features at region of interest levels improve amodal segmentation results. Implicit uses of 3D features may require further investigation for improvements in results.

Table 3: Results on 3D Object Detection

Model	Car Easy	Car Mod.	Car Hard	Ped. Easy	Ped. Mod.	Ped. Hard	Cyc. Easy	Cyc. Mod.	Cyc. Hard
PointPillar [36]	95.66	92.24	91.32	66.55	62.50	59.33	85.27	73.00	69.02
SECOND [37]	95.63	94.17	91.77	68.73	66.33	63.26	87.56	77.09	74.38
Point-RCNN [28]	96.59	92.93	90.55	**73.70**	65.81	62.09	89.36	76.91	75.17
PV-RCNN [29]	95.90	93.82	91.74	72.23	66.02	63.43	**95.27**	80.96	76.18
ESAAN (Ours)	**98.90**	**94.57**	**94.16**	72.86	**67.99**	**65.28**	88.04	**82.05**	**76.24**

5.7 Synthetic Augmentation

We trained our synthetic ESAAN with 30 epochs, then trained it on Kitti with 50 epochs. We compared this model with other implementations which has all been trained over 8000 epochs. The results can be seen in Table 3. All of the metrics used Average Precision over Bounding Box Prediction. For Cars, the evaluation was done using AP-R40@70; For Pedestrians (Ped.), the evaluation was done using AP@50; For Cyclists (Cyc.), the evaluation was done using AP@50. It is evident that our model outperforms in accuracy in almost every metric compared to other implementations, with significantly less epochs and training time. Especially in hard scenarios where our synthetic dataset was prepared for. This proves that using synthetic datasets for training 3D detection tasks can greatly enhance network accuracy.

5.8 Limitation and Future Works

Evaluation results show that although DAISnet has utilized depth-aware 3D features to improve amodal instance segmentation performance, there are still some limitations. Computed tangent normals are not the actual geometric normals of objects, a method of predicting geometric normals using neural network should be made. Depth-based 3D features can be further incorporates into the model by combining the monocular depth prediction module with DAISnet, which provides additional 3D-aware features during monocular depth prediction that might be useful. Current region of interest and mask logit resolutions remain low at 14x14, optimization techniques on increasing RoI resolution is worth investigating. Synthetic augmentation can be further improved with larger datasets, longer training epochs, and more rigorous scenarios. Longer training epochs may yield even higher levels of accuracy compared to existing models.

6 Conclusion

We proposed a novel depth-aware amodal instance segmentation network (DAISnet) that used depth information to infer 3D features and occlusion information. This model contains regional depth normalization, normal tangent computation and occlusion edge refinement methods that implicitly and explicitly utilize the depth information. We also developed a method of obtaining a synthetic dataset that contains depth and amodal mask annotations to address the difficulty of acquiring depth and amodal ground truths. Our experiments demonstrated that depth information and subsequent 3D features extracted implicitly and explicitly brings notable improvements to amodal instance segmentation performance. We have proven that using synthetic data to augment training on existing datasets, results in significantly higher accuracy.

7 References

[1] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Proc. NIPS*, 2015.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *Proc. ICCV*, 2017.

[3] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. In *Proc. CVPR*, 2019.

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. In *Proc. CVPR*, 2018.

[5] Hao Ding, Siyuan Qiao, Alan L. Yuille, and Wei Shen. Deeply shape-guided cascade for instance segmentation. In *Proc. CVPR*, 2021.

[6] Yuting Xiao, Yanyu Xu, Ziming Zhong, Weixin Luo, Jiawei Li, and Shenghua Gao. Amodal segmentation based on visible region segmentation and shape prior. In *Proc. AAAI*, 2021.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. 2013.

[8] Ross Girshick. Fast r-cnn. 2015.

[9] Fully convolutional networks for semantic segmentation. 2014.

[10] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring R-CNN. In *Proc. CVPR*, 2019.

[11] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. CVPR*, 2017.

[12] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. 2019.

[13] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proc. CVPR 2018*, 2018.

[14] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[15] Patrick Follmann, Rebecca König, Philipp Härtinger, Michael Klostermann, and Tobias Böttger. Learning to see the invisible: End-to-end trainable amodal instance segmentation. In *Proc. WACV*, 2019.

[16] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with KINS dataset. In *Proc. CVPR*, 2019.

[17] Xunli Zeng and Jianqin Yin. Amodal segmentation just like doing a jigsaw. *CoRR*, abs/2107.07464, 2021.

[18] Ke Li and Jitendra Malik. Amodal instance segmentation. In *Proc. ECCV*, 2016.

[19] Ziheng Zhang, Anpei Chen, Ling Xie, Jingyi Yu, and Shenghua Gao. Learning semantics-aware distance map with semantics layering network for amodal instance segmentation. In *Proceedings of the 27th ACM International Conference on Multimedia*, 2019.

[20] Yanfeng Liu, Eric Psota, and Lance C. Pérez. Layered embeddings for amodal instance segmentation. *CoRR*, abs/2002.06264, 2020.

[21] Jianwei Yang, Zhile Ren, Mingze Xu, Xinlei Chen, David J. Crandall, Devi Parikh, and Dhruv Batra. Embodied amodal recognition: Learning to move to perceive objects. In *Proc. ICCV*, 2019.

[22] Zhuo Deng and Longin Jan Latecki. Amodal detection of 3d objects: Inferring 3d bounding boxes from 2d ones in rgb-depth images. In *Proc. CVPR*, 2017.

[23] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. arXiv, 2015.

[24] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2016.

[25] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving, 2016.

[26] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds, 2018.

[27] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network, 2019.

[28] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud, 2018.

[29] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection, 2019.

[30] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection, 2020.

[31] balcilar. Densedepthmap. https://github.com/balcilar/DenseDepthMap, 2018.

[32] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with kins dataset. 2019.

[33] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[34] Patrick Follmann, Rebecca König, Philipp Härtinger, Michael Klostermann, and Tobias Böttger. Learning to see the invisible: End-to-end trainable amodal instance segmentation. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2019, Waikoloa Village, HI, USA, January 7-11, 2019*, pages 1328–1336. IEEE, 2019.

[35] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Deep occlusion-aware instance segmentation with overlapping bilayers. In *CVPR*, 2021.

[36] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CoRR*, abs/1812.05784, 2018.

[37] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018.

8 Acknowledgement

The utilization of depth information in amodal segmentation tasks drew a lot of inspirations and ideas from my past experiences in computer graphics. I developed a hobby in many areas of computer science from web development to robotics. The area that interested me the most was game development. I have been developing video games since I was Grade 7. It was fascinating because you could simulate and create an entire world of color and interactions. Throughout this time, I learnt a lot of spacial calculations in 3D and computer graphics. I taught myself how to write shaders, process mesh data, and produce stunning visuals for the final product. It was because of this experience that inspired me to incorporate depth information to amodal segmentation.

I have achieved a lot throughout my exploration in computer science, from being on USACO Gold Division to running my own CS club in school that developed applications used by thousands of students. The only field I that never attempted for was machine learning beyond following what sporadic news of new advancements. Whether it is because of my lack of guidance or belief that it is too difficult for a high school student. When I set out my mind to conduct a research project, I was finally determined to research into AI and step out of my comfort zone. Under the guidance and help provided by Mr. Ding, I finally understood how deep learning worked. When I saw the gradual decrease in loss during training for one of my starter networks, I felt very accomplished because not only did I manage to make it but also understood how and why it worked.

Amodal instance segmentation is a task that has not yet being explored much, and it is also a computer vision task - which interests me due to my past experiences in computer graphics. It was not long after I began my research did I think about using depths in amodal instance segmentation. Depth information is vital to 3D computer graphics, it provides the fundamental information to calculation occlusion (z-buffer and z-write) and distance based visual effects. Thus I began developing the RGSD synthetic database, which used the same 3D engine that I developed games using. DAISnet uses normal tangent maps, which is also an important component in 3D graphics. Normal maps represent the surface tangents of geometries using pixels without actually defining mesh vertex. It is much easier to create normal tangent maps than create actual geometry, it also significantly more efficient. I am very proud that I was able to utilize past experiences in game development to help with my research project on a relatively new field to me - and achieve great improvements in results.

I want to thank Dr. Hao Ding from Johns Hopkins University for providing voluntary guidance and feedback on validation of my ideas and its implementations. I am grateful to finally gain a thorough knowledge in deep learning and neural network.

I want to thank the support provided by the computer science department in my school, namely Mr. Shawn Wu who has supported my research on behalf of my school. I want to thank my parents who has made it possible for me to access a dozen Nvidia Tesla V100 instances through Google Cloud Platform, I would not have been able to iterate and complete such a thorough ablation study in time without it.