Exploiting Scene Depth for Object Detection with Multimodal Transformers (Supplementary Material)

Hwanjun Song[1,*], Eunyoung Kim[2]
Varun Jampani[2], Deqing Sun[2]
Jae-Gil Lee[3], Ming-Hsuan Yang[2,4,5]

[1]NAVER AI Lab, [2]Google Research
[3]KAIST, [4]University of California Merced
[5]Yonsei University

1 Overview

In this supplementary material, we first provide qualitative analysis of each component in MEDUSA (Section 2). Second, we describe the training setup used in our experiments for reproducibility (Section 3). Finally, we review the standard form of the transformer architecture including its multi-head attention and multi-layer structure (Section 4).

2 Qualitative Analysis of MEDUSA

The pipeline of MEDUSA consists of *three* components designed for **(i)** feature extraction from RGB and depth inputs, **(ii)** feature refinement of RGB and depth features, and **(iii)** feature fusion between them through the multimodal transformers. This section provides a qualitative analysis of each component to understand the detailed benefits of MEDUSA.

2.1 Global Saliency of Depth Features (Figure 1)

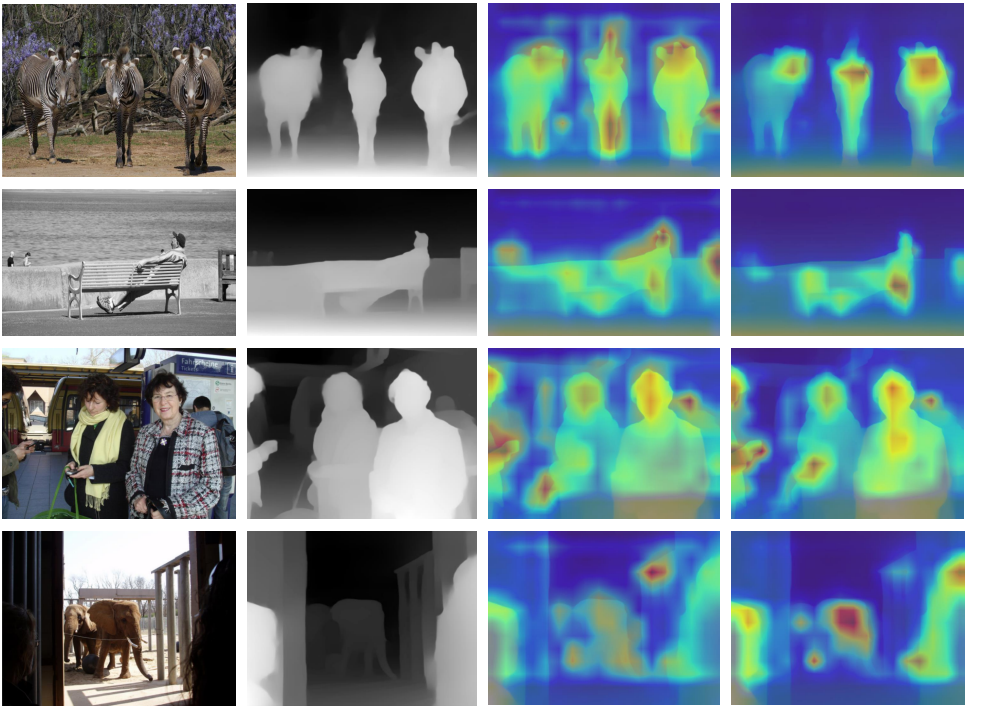
The contribution of the feature refiner stage in MEDUSA is to attenuate noise in the inferred features by the region- and channel-wise attentions. The improvement of the global saliency map [1] of depth features by the refiner is displayed in Figure 1. In general, the initial saliency map in Figure 1(c) does not clearly focus on the salient objects in the scene. However, the initial feature map is polished with a guide of the clean RGB input by the feature refiner and then becomes a more reliable one, leading to a more clear global saliency map in Figure 1(d). For example, in the first row of Figure 1(d), the front heads of the three zebras become more clearly identifiable.

2.2 Attention Weights for Transformers (Figures 2 and 3)

Encoder Attention.: We visualize the attention map of each modality derived by the self-attention block in the encoder. As illustrated in Figure 2, the encoder of MEDUSA explores all pairwise pixel interactions for each modality by maintaining the modality-wise self-attention module. Given a reference point (i.e., colored dots in Figure 2), the attention weights are emphasized if their positions are closely related to the reference point.

As already confirmed by Carion et al. [1], the self-attention in the encoder tends to separate objects, which helps simplify object extraction and localization for the decoder. This process is performed independently for each modality using two self-attention streams

[1]The global saliency map was computed by adding all the values in the feature map along the channel line.



(a) RGB Image. (b) Estimated Depth Map. (c) Depth Saliency. (d) Refined Depth Saliency.

Figure 1: **Visualization of the global saliency map of depth features**: (a) the source image; (b) the estimated depth map by MiDaS; and (c)–(d) the global saliency map of depth features before and after applying the feature refiner, respectively.

in the encoder of MEDUSA. By doing so, prior to RGB-D fusion in the decoder, MEDUSA captures the areas where RGB and depth features are respectively more effective to localize the object near the reference point.

Decoder Attention. MEDUSA performs attention-based RGB-D fusion using the transformer. The multimodal attention block in the decoder derives the attention weights for the stack sequence of RGB and depth representations, determining which representation should be highlighted to maximize their complementary information. Figure 3 shows the attention weights of each modality for RGB-D fusion along with its query identifiers and predicted labels. Unlike the previous work [6, 8, 17] that simply concatenates or adds the entire RGB and depth feature maps, MEDUSA determines a different fusion strategy object-wisely.

The RGB and depth information are complementary to each other for object detection. For the objects framed by cyan and yellow boxes in Figure 3(b), the RGB information is much dominant in classifying the bench and the person because depth values are very similar for the two objects. In contrast, the depth information serves as a decisive clue for the objects framed by cyan and green boxes in Figures 3(c) and 3(d).

3 Detailed Experimental Setup

3.1 Datasets

Three benchmark datasets are used for evaluation: VOC, COCO, and SUN-RGBD – an instance of data contains 2.35, 6.53, and 5.66 objects on average, respectively. We apply the

state-of-the-art monocular depth estimator, MiDaS [7], for VOC and COCO because ground-truth depth maps are not available in both datasets.

- VOC [2]: Two trainval sets of 17K images in VOC2007 and VOC2012 were combined for training data, and the 5K test set in VOC2007 was used for test data. Each instance contains 2.35 objects on average.
- COCO [4]: We used the widely-used trainval set (80K train set + 35K subset of validation set) in COCO2017 for training data, while the remaining 5K subset of the validation set was used for test data. COCO is more challenging than VOC since each instance contains 6.53 objects on average.
- SUN-RGBD [8]: We used standard splits: 50K training and 50K testing images with ground-truth depth maps captured by several devices. 19 major categories were used for detection following the literature [3, 9]. Each instance contains 5.66 objects on average.

3.2 Training Configuration

We describe the training hyperparameters used in our experiments. All the compared methods were trained using AdamW [5] for 150 epochs regardless of datasets. Except for the backbone, all the architectures were trained with an initial learning rate of 10^{-4} decayed by 10 at the 100-th epoch. We also applied a weight decay of 10^{-4}, a dropout of 0.1, and a gradient clipping with a maximal norm 0.1. For reproducibility, the random seed was always set to be 42, and the detailed hyperparameter setup is as follows:

Backbone. The ResNet-50 pretrained on ImageNet was used as the feature extractor, discarding the last classification layer. However, since the model was not fully compatible with the three extensions of DETR, the first convolutional layer in ResNet-50 was replaced to support the four-channel RGB-D input or one-channel depth input for the early and late fusion approaches, respectively. In addition, all the batch normalization weights and statistics of the backbone were frozen during training following the common practice in object detection [1].

Transformers. DETR introduces multiple hyperparameters for the transformer. We used the transformer with six encoder and decoder layers of width $d = 256$ for all methods. For each encoder (or decoder) layer, the multi-head attention mechanism with eight heads was applied, followed by the point-wise FFN of 2048 hidden units. Furthermore, an additive dropout of 0.1 was applied before the layer normalization. All the transformer architectures were trained from scratch using the wights initialized with Xavier initialization.

Feature Refiner. An additional component in MEDUSA is the feature refiner. For the region-wise attention, the number of binary mask maps b was set to be 50. Regarding the channel-wise attention, we used the standard form introduced in CBAM [11]; the average pooling was used to obtain the aggregated descriptor of each channel, and then the size of each descriptor was reduced with a reducti on ratio of 16 to alleviate parameter overhead.

Detection Head. The output of the decoder is fed to a 3-layer FFN for bounding box regression and linear projection for classification,

$$\hat{B} = \mathrm{FFN}_{\text{3-layer}}(O^{obj}) \;\text{ and }\; \hat{P} = \mathrm{Linear}(O^{obj}). \tag{1}$$

For box regression, the FFN produces the bounding box coordinates, $\hat{B} \in [0,1]^{n\times 4}$, that encodes the normalized box center coordinates along with its width and height. For classification, the linear projection uses a softmax function to produce the classification probabilities for all possible classes including the background class.

4 Preliminaries: Transformers

A transformer is a deep model that entirely relies on the self-attention mechanism for machine translation [10]. In this section, we briefly revisit the standard form of the transformer. For consistency with the manuscript, we assume that the input Z is a d-dimensional embedding sequence of length hw.

Single-Head Attention. The basic building block of the transformer is a self-attention module, which generates a weighted sum of the values, where the weight assigned to each value is the attention score computed by the scaled dot-product between its query and key. Let W_Q, W_K, and W_V be the learned projection matrices of the attention module, and then the output is generated by

$$\begin{gathered}\text{Attention}(Z) = \text{softmax}\big(\frac{(ZW_Q)(ZW_K)^\top}{\sqrt{d}}\big)(ZW_V) \in \mathbb{R}^{hw\times d}, \\ \text{where } \; W_Q, W_K, W_V \in \mathbb{R}^{d\times d}.\end{gathered} \tag{2}$$

Multi-Head Attention. It is beneficial to maintain multiple heads such that they repeat the linear projection process k times with different learned projection matrices. Let W_{Q_i}, W_{K_i}, and W_{V_i} be the learned projection matrices of the i-th attention head. Then, the output is generated by the concatenation of the results from all heads,

$$\begin{gathered}\text{Multi-Head}(Z) = [\text{Attention}_1(Z), \text{Attention}_2(Z), \ldots, \text{Attention}_k(Z)] \in \mathbb{R}^{hw\times d}, \\ \text{where } \; \forall_i \, W_{Q_i}, W_{K_i}, W_{V_i} \in \mathbb{R}^{d\times(d/k)}.\end{gathered} \tag{3}$$

Typically, the dimension of each head is divided by the total number of heads due to the computational overhead.

Feed-Forward Network (FFN). The output of the multi-head attention is fed to the point-wise FFN, which performs the linear transformation for each position separately and identically to allow the model focusing on the contents of different representation subspaces. Here, the residual connection and layer normalization are applied before and after the FFN. Hence, the final output H is generated by

$$\begin{gathered}H = \text{LayerNorm}(\text{Dropout}(H') + H''), \\ \text{where } \; H' = \text{FFN}(H'') \; \text{ and } \; H'' = \text{LayerNorm}(\text{Dropout}(\text{Multi-Head}(Z)) + Z).\end{gathered} \tag{4}$$

Encoder and Decoder. The encoder only has two sub-layers: a multi-head self-attention module and a point-wise FFN. That is, the output of the encoder is produced as described in Eq. (4). On the other hand, the decoder includes another third layer in addition to the two sub-layers in the encoder. The third layer receives the output of the encoder (i.e., memory) as the key and value, and the output of its multi-head self-attention module as the query. Similar to the encoder, the decoder employs the residual connection and layer normalization around each of the sub-layers.

Multi-Layer Transformers. For both an encoder and a decoder, the output of a previous layer is fed directly to the input of the next layer. Regarding the positional encoding, the same value is added to the input of each attention module for all layers. Please note that MEDUSA adds the sinusoidal-based spatial positional encoding to supplement the flattened input sequence following the recent work [1, 13].

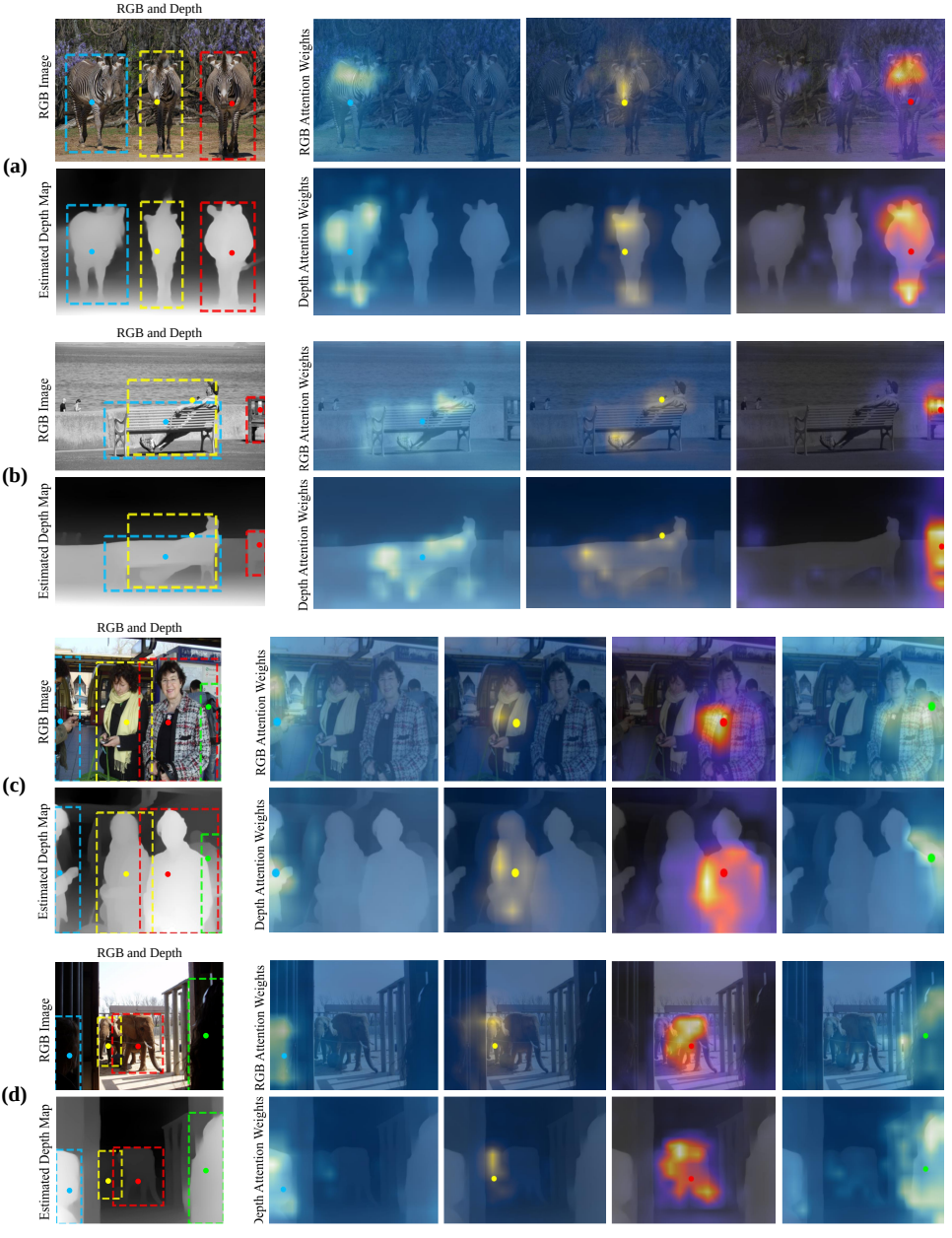


Figure 2: **Visualization of encoder attention weights for self-attention**. All the attention weights are derived from the "self-attention block" in the encoder. As the reference point for encoder attention maps, we provide a set of suitable positions inside the predicted bounding boxes. Using the average value of multiple heads in the final encoder layer, the attention weights are coded with different colors for each reference point.

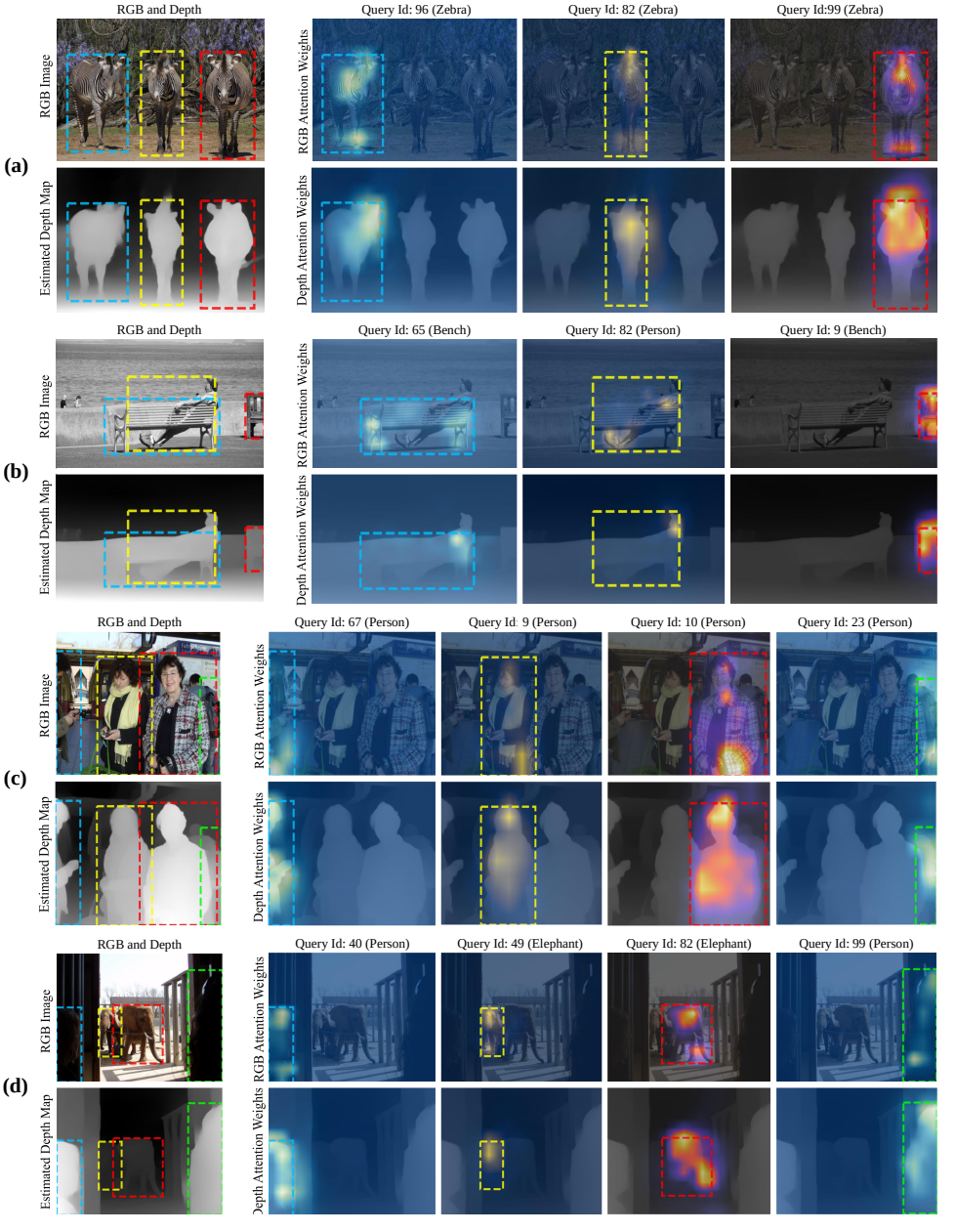


Figure 3: **Visualization of decoder attention weights for RGB-D fusion**. All the attention weights are derived from the "multimodal attention block" in the decoder. All predicted objects with their softmax probability > 0.8 are plotted at their respective location from left to right. The first and second rows of each example in (a)–(d) are the attention weights for RGB and depth representations. Using the average value of multiple heads in the final decoder layer, the attention weights are coded with different colors for each object.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229, 2020.

[2] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[3] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, pages 345–360, 2014.

[4] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, pages 740–755, 2014.

[5] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[6] Tanguy Ophoff, Kristof Van Beeck, and Toon Goedemé. Exploring RGB+ depth fusion for real-time object detection. *Sensors*, 19(4):866, 2019.

[7] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[8] Max Schwarz, Anton Milan, Arul Selvam Periyasamy, and Sven Behnke. RGB-D object detection and semantic segmentation for autonomous manipulation in clutter. *The International Journal of Robotics Research*, 37(4-5):437–451, 2018.

[9] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR*, pages 567–576, 2015.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[11] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In *ECCV*, pages 3–19, 2018.

[12] Jiachen Yang, Chenguang Wang, Huihui Wang, and Qiang Li. A RGB-D based real-time multiple object detection and ranging system for autonomous driving. *IEEE Sensors Journal*, 2020.

[13] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.