MMMS: Multi-Modal Multi-Surface Interactive Segmentation

Robin Schön
Fakultät für Angewandte Informatik
University of Augsburg
Augsburg, Germany
robin.schoen@uni-a.de

Julian Lorenz
Fakultät für Angewandte Informatik
University of Augsburg
Augsburg, Germany
julian.lorenz@uni-a.de

Katja Ludwig
Fakultät für Angewandte Informatik
University of Augsburg
Augsburg, Germany
katja.ludwig@uni-a.de

Daniel Kienle
Fakultät für Angewandte Informatik
University of Augsburg
Augsburg, Germany
daniel.kienzle@uni-a.de

Rainer Lienhart
Fakultät für Angewandte Informatik
University of Augsburg
Augsburg, Germany
rainer.lienhart@uni-a.de

***Abstract*—In this paper, we present a method to interactively create segmentation masks on the basis of user clicks. We pay particular attention to the segmentation of multiple surfaces that are simultaneously present in the same image. Since these surfaces may be heavily entangled and adjacent, we also present a novel extended evaluation metric that accounts for the challenges of this scenario. Additionally, the presented method is able to use multi-modal inputs to facilitate the segmentation task. At the center of this method is a network architecture which takes as input an RGB image, a number of non-RGB modalities, an erroneous mask, and encoded clicks. Based on this input, the network predicts an improved segmentation mask. We design our architecture such that it adheres to two conditions: (1) The RGB backbone is only available as a black-box. (2) To reduce the response time, we want our model to integrate the interaction-specific information after the image feature extraction and the multi-modal fusion. We refer to the overall task as multi-modal multi-surface interactive segmentation (MMMS). We are able to show the effectiveness of our multi-modal fusion strategy. Using additional modalities, our system reduces the NoC@90 by up to 1.28 clicks per surface on average on DeLiVER and up to 1.19 on MFNet. On top of this, we are able to show that our RGB-only baseline achieves competitive, and in some cases even superior performance when tested in a classical, single-mask interactive segmentation scenario.**

***Index Terms*—segmentation, interactive, multi-modal, surface, multi-surface, black-box**

I. Introduction

Segmentation tasks constitute some of the most important tasks in computer vision. The most prominent are instance segmentation [1] and semantic segmentation [2], [3]. To train a segmentation model, we need large amounts of annotated segmentation data.

However, ground truth segmentation masks are hard to obtain. This led to the development of click-based interactive segmentation systems [4]–[6]. Therein, the user places clicks on the image to indicate which object surface they want to segment. The system then combines these clicks with the image to produce a high-quality segmentation mask.

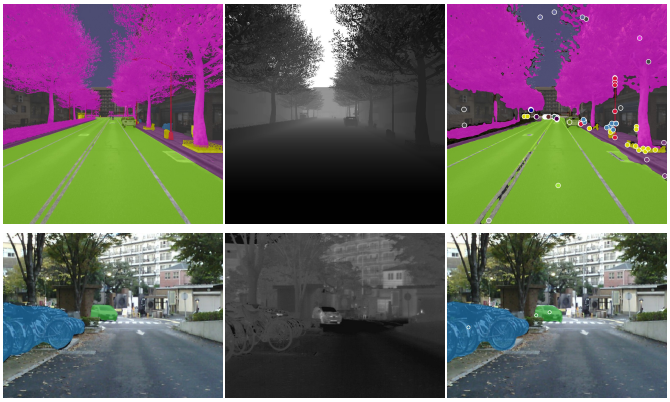


Fig. 1: Qualitative examples from DeLiVER (upper images) and MFNet (lower images). The left column shows the ground truth, the middle column shows the additional modality, and the right column shows the prediction. We only plot positive points for easier identification. More qualitative examples can be found in the supplementary material section J.

Most of the interactive segmentation literature only considers the segmentation process of each mask in isolation. Nevertheless, there are situations in which we have multiple adjacent surfaces in the same image (e.g. Fig. 1).

In such cases, false positive pixels in the annotations can cause conflicts between masks (see Fig. 2). A user then has to revisit some masks for correction. Previous evaluation procedures for interactive segmentation ignore this problem. For this reason, we propose a novel evaluation procedure that takes this problem into account.

When it comes to segmentation tasks, we often have the opportunity to benefit from additional modalities instead of just RGB images. For example, we might have depth maps or thermal images. These additional modalities are usually collected by a specific sensor at the same time as the images.

Thus, by the time we annotate our images with segmentation masks, we have access to the non-RGB modalities. For this reason, we may leverage the other modalities to ease the annotation of our data. In this paper, we propose a network architecture that can leverage an arbitrary number of modalities for interactive segmentation. We are able to show that that our multi-modal fusion strategy supports the interactive segmentation process.

On top of this, we design our multi-modal segmentation strategy in a way that allows for the RGB backbone to be a black-box. In particular, this allows us to use RGB foundation models which are outsourced to an external foundation model provider. This scenario is discussed in greater detail in section B of the supplementary material.

In summary, we tackle the problem of multi-modal multi-surface interactive segmentation (MMMS). [1] Our contributions can be summarized as follows:

- We present an **asymmetric multi-modal fusion strategy** which assumes the RGB backbone to be an untrainable black-box while all other parts of the network are trainable. We show that our fusion strategy successfully leverages multi-modal data for interactive segmentation.
- We introduce **new metrics for interactive segmentation of multiple adjacent surfaces** in the same image. We adapt the evaluation mechanism to account for challenges in this scenario that have previously been ignored by evaluation mechanisms for classical interactive segmentation.
- We provide an experimental evaluation that demonstrates the **performances improvements caused by our strategy of multi-modal fusion** on various datasets with a wide range of non-RGB modalities.

II. Related Work

A. Interactive Segmentation

While there are other modes of interaction [6]–[8], this paper only focuses on click-based interactive segmentation [4], [9]. A detailed introduction to this problem can be found in the supplementary material section A. Most modern interactive segmentation systems are based on a neural network that takes the encoded clicks and the image as input, and tries to predict a high-quality mask [4]–[6], [10], [11]. As the task formulation revolves around iteratively improving a mask, recent systems [4], [10] use the previous, imperfect mask as additional input. The bulk of these works require rerunning the entire network after each click, incurring slow response times. Similar to previous works [5], [12]–[14], we design our network such that only a small part of the architecture has to be rerun after each click. Notably, there is also previous work on interactive segmentation on images with multiple surfaces [15]–[18]. However, come to find their interaction modes rather arbitrary and constraining with respect to architectural choices. In contrast to previous methods, we offer an extension of the standard interactive segmentation problem.

[1] We will release the code upon publication: https://github.com/Schorob/mmms

B. Multi-Modal Segmentation

Earlier work on multi-modal segmentation has mostly dealt with a single additional input modality. Examples for such architectures are FuseNet [19], MFNet [20], RTFNet [21], SegMiF [22] and CMX [23]. More recent methods use multiple additional modalities at once. TokenFusion [24] combines tokens from a variable number of modalities. MCubeSNet [25] concatenates the features from different modalities before feeding them to the decoder. CMNeXt [26] introduces a mechanism to select tokens from multiple non-RGB modalities, which are afterwards fused with the main RGB modality. Both, HRFuser [27] and CAFuser [28] use windowed cross-attention to combine multiple modalities. GeminiFusion [29] applies a pixel-wise fusion mechanism between the attention mechanism and the MLP of the backbone. StitchFusion [30] uses adapter modules which are inserted into a frozen backbone. All of the aforementioned methods assume access to the RGB backbone during training. However, our method will allow for a black-box RGB backbone. We are not the first to use non-RGB information for the purpose of click-guided interactive segmentation. The methods presented in [31] and [32] are both based on using pseudo-depth maps generated by a pretrained monocular depth estimation (MDE) network. In contrast to these methods, we want to use real and arbitrary additional modalities.

III. Multi-Surface Interactive Segmentation

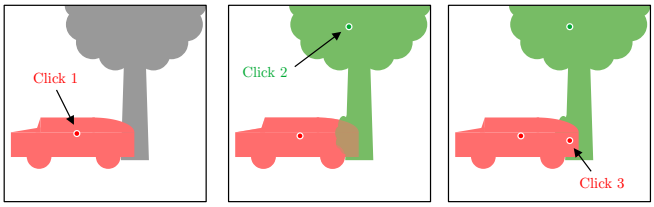
In this section, we discuss an extension to the classical interactive segmentation problem. We adress challenges that occur when segmenting multiple masks in the same image. We recommend that any reader unfamiliar with the standard interactive segmentation problem and the NoC metric may first read section A in the supplementary material. The standard NoC metric considers each mask in isolation.

Suppose we want to interactively create segmentation masks for multiple surfaces in the same image using a regular interactive segmentation system. The most straightforward way of doing this would be to annotate these masks independently. Let L be the number of surfaces / masks we want to annotate. We want to create a set of segmentation masks

$$\mathcal{S}_{\mathbf{m}} = \left\{ \mathbf{m}^{(1)}, ..., \mathbf{m}^{(i)}, ..., \mathbf{m}^{(j)}, ..., \mathbf{m}^{(L)} \right\}, \tag{1}$$

where each of those masks is annotated using the standard interactive segmentation procedure. We generally do not assume the user to annotate each of these masks to absolute perfection. We just expect a sufficiently high degree of quality. This is also reflected in the automatic evaluation of interactive segmentations systems. We only continue the annotation clicks until an IoU with a pre-existing grouth truth mask of at least Θ_{IoU} is reached. These imperfections allow for conflicts by creating overlap between adjacent masks.

The standard version of the interactive segmentation problem ignores the issue of overlap between imperfect masks. It deals with each of these masks in isolation, despite each describing a different surface on the same image. Given the



a. Click 1 guides the prediction of mask 1.

b. Click 2 causes false positive pixels in mask 2 which harm the quality of mask 1.

c. Mask 1 has to be revisited with an additional click 3.

Fig. 2: We first place click 1 to annotate mask 1 (a.). Afterwards, we place click 2 to annotate mask 2 (b.). False positive pixels in mask 2 harm the quality of mask 1. We then have to revisit mask 1 by placing click 3 (c.).

fact that these surfaces are sometimes semantically defined (e.g. *grass*, *sky* or *water*), they may actually be heavily entangled.

In the remainder of this section, we will describe an extended version of the interactive segmentation evaluation procedure that takes this problem into account. We will also describe corresponding metrics.

Let $\mathbf{m}^{(i)}, \mathbf{m}^{(j)} \in \mathcal{S}_\mathbf{m}$ be two masks from Eq. 1. Here, i and j indicate different target surfaces; not different iterations for the same target surface. Assume that $\mathbf{m}^{(j)}$ has been annotated after $\mathbf{m}^{(i)}$. In case of an overlap, the later annotated mask $\mathbf{m}^{(j)}$ overrides the earlier annotated mask $\mathbf{m}^{(i)}$. An example of such a situation can be found in Fig. 2. Some pixels in the overlapping area may have been false positives of $\mathbf{m}^{(j)}$ that actually belong to $\mathbf{m}^{(i)}$ (see Fig. 2 b.). In such a case, the false positive pixels in $\mathbf{m}^{(j)}$ have harmed the quality of $\mathbf{m}^{(i)}$. We would need to revisit $\mathbf{m}^{(i)}$ (see in Fig. 2 c.).

In order to account for this challenge, we introduce two new metrics: *NoCMS* and *FRMS*. These metrics are based on two IoU-thresholds $\Theta_{\text{Average-IoU}}$ and Θ_{IoU}. When computing these metrics, our goal is to reach a sufficiently high average IoU of all masks with their associated ground truth masks. This is the case if

$$\frac{1}{L}\sum_{l=1}^{L} \text{IoU}(\mathbf{m}^{(l)}, \mathbf{m}^{(l)}_{\text{GT}}) \geq \Theta_{\text{Average-IoU}}. \tag{2}$$

If this holds true, we are done with annotating the image. If not, we pick the mask $\mathbf{m}^{(k)} \in \mathcal{S}_\mathbf{m}$ which has the smallest IoU with its corresponding ground truth. This $\mathbf{m}^{(k)}$ is then improved with the regular interactive segmentation mechanism for single surfaces. We add clicks until an IoU of at least Θ_{IoU} is reached. Afterwards, we test again whether the condition in Eq. 2 is true. We repeat this cycle of *selection and improvement* of an $\mathbf{m}^{(k)}$ until an average IoU of at least $\Theta_{\text{Average-IoU}}$ is reached. It should be noted that this evaluation requires $\Theta_{\text{IoU}} \geq \Theta_{\text{Average-IoU}}$. Otherwise, the improvements of $\mathbf{m}^{(k)}$ might not be significant enough for the average IoU to rise above $\Theta_{\text{Average-IoU}}$. The count of already executed clicks for each particular surface is **not** reset at each revisiting attempt. Instead, we accumulate the click count over different attempts with the same surface. Since the network might not be capable of annotating some masks, we use a maximum number of clicks $n_{\text{max}} = 20$ for each surface mask. We introduce two new metrics for multi-surface interactive segmentation. The average number of clicks per surface in this extended scenario is called NoCMS@$(\Theta_{\text{IoU}}, \Theta_{\text{Average-IoU}})$. We can also measure the percentage of surfaces that could not be segmented successfully within $n_{\text{max}} = 20$ clicks. We refer to this percentage as the failure rate FRMS@$(\Theta_{\text{IoU}}, \Theta_{\text{Average-IoU}})$. Additionally, section E of our supplementary material discusses the creation of a joint mask for all surfaces in an image.

IV. Method

In this section, we describe the architecture that we use to tackle our problem. We first explain the general architecture and its overall functionality. Afterwards, we explain the *MMFuser* and the *CSNet* in greater detail. An overview of our architecture is given in Fig. 3.

a) Overall architecture : We will describe the data processing in our architecture in a step-by-step fashion. First, the RGB image $\mathbf{x}_{\text{img}} \in \mathbb{R}^{H\times W\times 3}$ is processed by the RGB backbone FM_{RGB}. We restrict ourselves to cases where FM_{RGB} has a ViT architecture. Let P_{FM} and d_{FM} be the patch size and the embedding dimension of the model, respectively. We obtain the image feature tensors

$$\mathcal{F}_{\text{img}} = \left(\mathcal{F}^{(1)}_{\text{img}}, \mathcal{F}^{(2)}_{\text{img}}, \mathcal{F}^{(3)}_{\text{img}}, \mathcal{F}^{(4)}_{\text{img}}\right) = \text{FM}_{\text{RGB}}(\mathbf{x}_{\text{img}}). \tag{3}$$

These image feature tensors $\mathcal{F}^{(i)}_{\text{img}} \in \mathbb{R}^{\frac{H}{P_{\text{FM}}}\times\frac{W}{P_{\text{FM}}}\times d_{\text{FM}}}$ for $i = 1, 2, 3, 4$ are extracted from various intermediate layers in the network. In our case, the features are extracted after the blocks 3, 6, 9, and 12. As FM_{RGB} is a black-box, we still need a learnable adapter that gives our model the ability to change the representation of $\mathcal{F}_{\text{img}}$. We follow common practice and operate on feature pyramids to better represent multi-scale features. In order to obtain an adapted feature pyramid, we use a *ParallelFPN* to transform $\mathcal{F}_{\text{img}}$ into

$$f_{\text{img}} = \left(f^4_{\text{img}}, f^8_{\text{img}}, f^{16}_{\text{img}}, f^{32}_{\text{img}}\right) = \text{ParallelFPN}(\mathcal{F}_{\text{img}}). \tag{4}$$

We have $f^i_{\text{img}} \in \mathbb{R}^{\frac{H}{i}\times\frac{W}{i}\times C_i}$ for $i \in \{4, 8, 16, 32\}$. The C_i are the feature channel numbers of the respective tensors in our feature pyramid. Unless stated otherwise, all feature pyramids in the following text will have this shape. Our ParallelFPN is inspired by [10]. A detailed description can be found in the supplementary material in section I.

So far, we have not integrated any non-RGB information into our features. This will be done by the *MMFuser*. The MMFuser receives M non-RGB images $\mathbf{x}_{\text{mod},m} \in \mathbb{R}^{H\times W\times C_m}$ for $m = 1, ..., M$ as well as the RGB features f_{img} as input. This results in the mixed feature pyramid

$$\begin{gathered} f_{\text{mix}} = \left(f^4_{\text{mix}}, f^8_{\text{mix}}, f^{16}_{\text{mix}}, f^{32}_{\text{mix}}\right) = \\ \text{MMFuser}(f_{\text{img}}, \mathbf{x}_{\text{mod},1}, ..., , \mathbf{x}_{\text{mod},M}). \end{gathered} \tag{5}$$

f_{mix} has the same tensor shapes as f_{img}. It should be noted that *all computation up until this point has to be carried out only once per image*. The feature extraction is not dependent

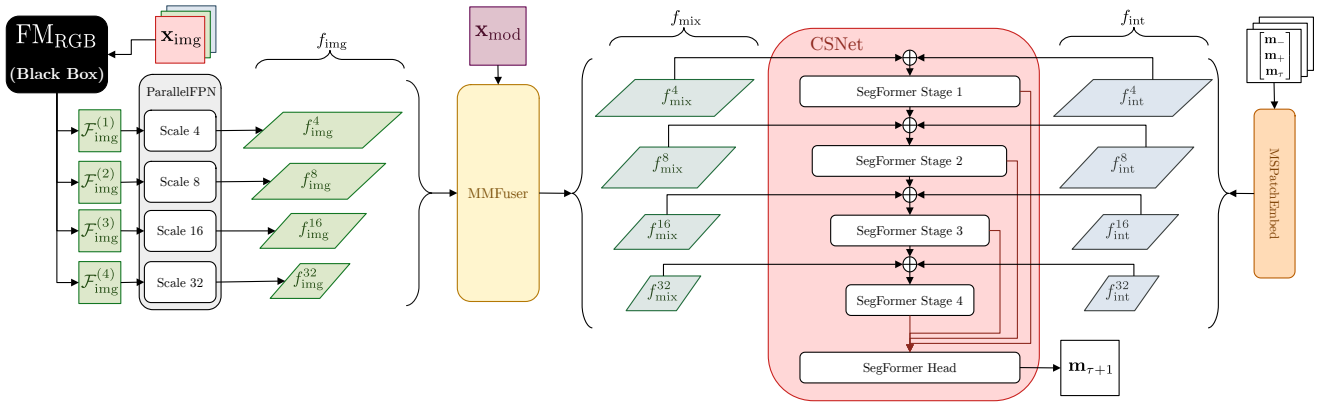


Fig. 3: Overall architecture of our multi-modal interactive segmentation system. The RGB features $\mathcal{F}_{\text{img}}$ from the backbone FM_{RGB} are processed by a ParallelFPN to obtain a feature pyramid f_{img}. The MMFuser integrates information from non-RGB modalities $\mathbf{x}_{\text{mod}}$ into the feature pyramid resulting in f_{mix}. MSPatchEmbed creates a feature pyramid f_{int} from the interaction tensor $[\mathbf{m}_+; \mathbf{m}_-; \mathbf{m}_\tau]$. CSNet uses f_{mix} and f_{int} to predict the mask $\mathbf{m}_{\tau+1}$. Apart from CSNet and the mutli-scale patch embedding (MSPatchEmbed) every part of our architecture only has to be executed once per image. $\oplus$ denotes element-wise addition.

on information for interactive mask annotation. Due to this design choice, we do not have to rerun FM_{RGB} and MMFuser again after each click, which improves the response time. The positive and negative clicks are encoded as small disks in their respective binary maps $\mathbf{m}_+$ and $\mathbf{m}_-$ as described in [4]. The clicks are then concatenated with the previous mask $\mathbf{m}_\tau$, resulting in the *interaction tensor* $[\mathbf{m}_+; \mathbf{m}_-; \mathbf{m}_\tau] \in \mathbb{R}^{H \times W \times 3}$. We want to give our model access to the interaction information at every stage of the mask prediction. For this reason, we make use of a multi-scale patch embedding *MSPatchEmbed*. MSPatchEmbed converts the interaction tensor into an interaction feature pyramid

$$\begin{aligned} f_{\text{int}} &= \left(f^4_{\text{int}}, f^8_{\text{int}}, f^{16}_{\text{int}}, f^{32}_{\text{int}}\right) \\ &= \text{MSPatchEmbed}([\mathbf{m}_+; \mathbf{m}_-; \mathbf{m}_\tau]). \end{aligned} \tag{6}$$

The tensors in f_{int} have the same shapes as those in f_{mix}. MSPatchEmbed is implemented as different convolutions that produce the desired shapes in our feature pyramid.

The last part of our architecture is our click segmentation network *CSNet*, which can be seen on the right side of Fig. 3. CSNet combines the features in f_{mix} and f_{int}, and produces the improved mask $\mathbf{m}_{\tau+1}$:

$$\mathbf{m}_{\tau+1} = \text{CSNet}(f_{\text{mix}}, f_{\text{int}}) \tag{7}$$

We note that the two modules MSPatchEmbed and CSNet are the only two components of our network that would have to be rerun after each click.

b) MMFuser : We now describe our MMFuser. The MMFuser is used to integrate multi-modal information into the feature pyramid (Fig. 4, *left*).

First, the non-RGB modality $\mathbf{x}_{\text{mod}}$ is processed by a SegFormer encoder [2]. This encoder yields a feature pyramid

$$\begin{aligned} f_{\text{mod}} &= \left(f^4_{\text{mod}}, f^8_{\text{mod}}, f^{16}_{\text{mod}}, f^{32}_{\text{mod}}\right) \\ &= \text{SegFormerEncoder}(\mathbf{x}_{\text{mod}}). \end{aligned} \tag{8}$$

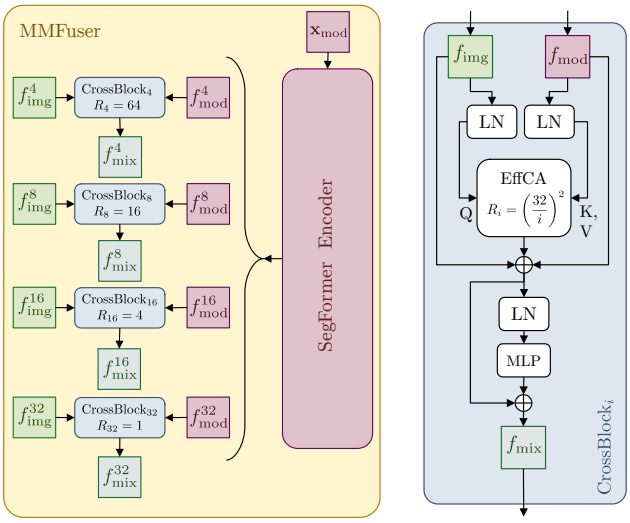


Fig. 4: The MMFuser (*left*) uses a SegFormer encoder to compute the features for the non-RGB modality. The feature fusion happens in the CrossBlock (*right*). Similar to [2], the efficient cross-attention (EffCA) works with a number of keys and values that are reduced by a reduction rate $R \in \{1, 4, 16, 64\}$. In case we use multiple non-RGB modalities, we have one SegFormer encoder and one CrossBlock per modality. $\oplus$ denotes element-wise addition.

The tensors in f_{mod} have the same shape as those in f_{img}. In order to mix f_{img} and f_{mod}, we will employ a cross-attention based *CrossBlock* (Fig. 4, *right*) at each stage. For each $i \in \{4, 8, 16, 32\}$, we compute

$$f^i_{\text{mix}} = \text{CrossBlock}_i(f^i_{\text{img}}, f^i_{\text{mod}}). \tag{9}$$

We want to equip the model with the capacity to dynamically integrate useful parts from other modalities. For this reason,

the first part of the CrossBlock is a cross-attention module. To reduce the execution time, we extend the efficient self-attention of [2] to cross-attention. We reduce the number of keys and values by a reduction rate R_i. The number of queries stays the same. This results in what we call *efficient cross-attention EffCA*. Therein, the image features f^i_{img} are used to compute the queries Q, while we compute the keys K and values V on the basis of f^i_{mod}. For a particular $i \in \{4, 8, 16, 32\}$ we will use a reduction rate $R_i = \left(\frac{32}{i}\right)^2$. Formally we have

$$\hat{f}^i_{\text{mix}} = \text{EffCA}(\text{LN}(f^i_{\text{img}}), \text{LN}(f^i_{\text{mod}})) + f^i_{\text{img}} + f^i_{\text{mod}} \quad (10)$$

$$f^i_{\text{mix}} = \text{MLP}(\text{LN}(\hat{f}^i_{\text{mix}})) + \hat{f}^i_{\text{mix}}, \quad (11)$$

where LN represents layer normalizations. Each layer normalization has its own learnable parameters. In cases where we have more than one non-RGB modality, we use a separate SegFormer encoder [2] for each of them. In addition to this, we have multiple cross-attentions, one for each modality.

c) CSNet : The click segmentation network *CSNet* uses the interaction information f_{int} and our feature pyramid f_{mix} to predict a mask $\mathbf{m}_{\tau+1}$. Both, f_{int} and f_{mix} are encoded in a dense fashion. We thus deem it natural to use existing insights about processing dense features, and employ an existing segmentation architecture. In our case, we will use the different stages and the prediction head of a SegFormer [2]. The internal mechanism is depicted on the right side of Fig. 3.

We use the stages of the SegFormer sequentially, and make sure that the network has access to both, image features and interaction features, at each stage of the network. The input to the first stage only consists of $f^0_{\text{mix}} + f^0_{\text{int}}$. The input to later stages is the sum of f^i_{mix}, f^i_{int} and the output of the previous stage. The output of all stages is then fed to the SegFormer prediction head which will predict the subsequent mask $\mathbf{m}_{\tau+1}$.

V. Experiments

To show the efficacy of our our multi-modal fusion strategy, we test our system on DeLiVER [23], [26] and MFNet [20]. DeLiVER offers depth maps, lidar and event camera images as additional modalities, and MFNet offers thermal images. We use four different types of RGB foundation models as FM_{RGB}: DINOv2-B14 [33], [34], MAE-B16 [35], IJEPA-H16 [36], and the RGB encoder from the SAM model (SAM-B16) [5]. Further implementation details and qualitative examples can be found in sections F and J of our supplementary material, respectively. Section G of the supp. material contains further results on MCubeS [25] and FMB [22].

We use an NVIDIA V100 GPU. If we amortize the duration of the feature extraction over all clicks, our model takes 42 ms per click on average when tested on the MFNet dataset. If we only consider the response time in isolation, we measure 12 ms when averaged over 30 clicks. On the CPU (Intel®Xeon®CPU E5-2697 v4) we arrive at 206 ms per click. We thus consider our model to be real-time-capable.

A. Results on DeLiVER

The results on the DeLiVER dataset [26] are presented in Table I.

The first thing to be noted is that we encounter very complex non-contiguous shapes. For this reason, we also look at lower IoU thresholds with $\Theta_{\text{IoU}} \in \{60, 70, 80, 90\}$. For the DINOv2-B14 we test all possible combinations of the available modalities. Using only a single non-RGB modality our fusion method works best with depth maps, causing a reduction in the NoC@80 of 1.3 clicks on average.

The DeLiVER dataset [26] contains intentionally placed perturbations in lidar and event data. These modalities are thus less reliable. This is reflected in our method performance, such as a NoC@90 of 16.07 for depth vs. 16.10 for depth and lidar. However, our method still causes improvements over the RGB-only case.

For the new multi-surface interactive segmentation metrics (see Sec. III) we use $\Theta_{\text{IoU}} = 80$ and $\Theta_{\text{Average-Iou}} = 70$. The multi-modal fusion works best with a combination of depth and event data with a NoCMS@(80, 70) of 14.50 and a FRMS@(80, 70) of 60.19.

We use depth to demonstrate the effectiveness of our fusion strategy for other backbones. Fig. 1 shows qualitative examples.

B. Results on MFNet

MFNet [20] offers thermal images as an additional modality. The general performance results can be seen in Table II.

The results in Table II corroborate our observations on DeLiVER. When looking at RGB-only results, DINOv2-B14 delivers the best performance with a NoCMS@(80,70) of 11.30. This points to the model's great capability to perform geometrically intricate segmentation tasks. In all cases, our model successfully integrates information from thermal images. The most significant improvements can be observed when using IJEPA, reducing the NoC@90 by 1.19 and the NoCMS@(80, 70) by 2.10. However, it should be noted that thermal images are not sufficient to bridge the pre-exising performance gap between DINOv2 and IJEPA. DINOv2 using only RGB images (NoC@90 of 15.37) still outperforms the IJEPA-based model with thermal images (NoC@90 of 15.62).

C. RGB-Only Baseline Performance

We demonstrate the efficacy of our RGB-only baseline (see Table III). The baseline results from the architecture described in Section IV if we remove the MMFuser. We only compare ourselves to models with similar conditions for a fair comparison. This especially means using a ViT-B as the backbone and the COCO+LVIS dataset [4], [37], [38] as training data. The only exception are the SAM-based models in Table IIIa and IIIb. Most importantly, we only compare ourselves to late fusion models. We do so, because late fusion generally imposes a more difficult condition by sacrificing performance for the goal of improved efficiency, as described by [14]. We test on DAVIS [39], HQSeg-44k [11], GrabCut [40], SBD [41], Berkeley [42], [43] and SHSeg [14].

The results can be seen in Tables IIIa and IIIb. We only compare ourselves to the other methods on datasets for which results are available. Our model generally produces

TABLE I: Results on DeLiVER [26]. The leftmost column (FM_{RGB}) indicates the backbone we used. The usage of non-RGB modalities is indicated by a checkmark. In RGB-only cases the MMFuser is not present.

FM_{RGB}	Depth	Lidar	Event	NoC@60 ↓	NoC@70 ↓	NoC@80 ↓	NoC@90 ↓	NoCMS@(80, 70) ↓	FRMS@(80, 70) ↓
DINOv2-B14				10.92	13.05	15.10	16.99	15.58	66.43
	✓			9.83	**11.68**	**13.80**	**16.07**	14.55	60.89
		✓		10.85	12.96	15.03	16.92	15.48	64.97
			✓	10.86	12.97	15.01	16.91	15.47	65.33
		✓	✓	10.77	12.88	14.99	16.90	15.46	65.18
	✓		✓	**9.78**	**11.68**	**13.80**	16.08	**14.50**	**60.19**
	✓	✓		9.82	**11.68**	13.82	16.10	14.59	61.24
	✓	✓	✓	9.79	11.73	13.82	16.09	14.59	61.06
MAE-B16				11.88	13.87	15.68	17.33	16.11	69.50
	✓			10.16	11.97	13.96	16.13	14.72	61.85
SAM-B16				11.63	13.59	15.47	17.19	15.93	68.39
	✓			10.11	11.89	13.94	16.14	14.67	61.61
IJEPA-H16				11.97	14.05	15.89	17.48	16.34	68.86
	✓			10.31	12.07	14.01	16.20	14.85	62.15

TABLE II: Results on MFNet [20]. The leftmost column (FM_{RGB}) indicates the backbone we used. The usage of thermal images is indicated by a checkmark. In RGB-only cases the MMFuser is not present.

FM_{RGB}	Thermal	NoC@80 ↓	NoC@90 ↓	NoCMS@(80, 70) ↓
DINOv2-B14		10.05	15.37	11.30
	✓	**8.66**	**14.88**	**9.88**
MAE-B16		11.28	16.14	12.40
	✓	9.39	15.38	10.58
SAM-B16		11.01	15.93	12.30
	✓	9.26	15.31	10.91
IJEPA-H16		12.06	16.81	13.45
	✓	9.79	15.62	11.35

TABLE III: A comparison with other methods.

(a) Comparison of our baseline with other models on DAVIS and HQSeg-44k. The metric is always the NoC@90 (lower is better). Results for other methods taken from [14] and [44].

Architecture	DAVIS	HQSeg-44k
SAM [5]	5.14	7.46
MobileSAM [45]	5.83	8.70
HQ-SAM [11]	5.26	6.49
SegNext [13]	5.34	7.18
Interformer [12]	5.45	7.17
HR-SAM [44]	5.48	7.66
HR-SAM++ [44]	5.41	7.47
SkipClick [14]	**4.94**	**6.00**
Our Baseline	5.14	6.30

(b) Comparison of our baseline on further datasets with the NoC@90 metric (lower is better). Our model outperforms others on SBD and SHSeg. Results for other methods taken from [12] and [14].

Architecture	GrabCut	SBD	Berkeley	SHSeg
SAM [5]	-	-	-	7.46
HQ-SAM [11]	-	-	-	14.29
Interformer [12]	1.50	6.34	3.14	-
SkipClick [14]	**1.44**	6.18	**2.45**	2.52
Our Baseline	1.54	**6.10**	2.75	**2.51**

(c) Comparison of our method with SkipClick on DeLiVER. In order to use depth in SkipClick, we integrated the MMFuser module as described in section H in the supplementary material.

Architecture	NoC@80 ↓	NoC@90 ↓	NoCMS@(80, 70) ↓
SkipClick + Depth	14.54	16.53	14.90
Ours + Depth	**13.80**	**16.07**	**14.55**

competitive results and even outperforms most of the SAM-based methods. We are on-par with SAM on DAVIS. This is remarkable, since all of these methods made use of the large SA-1B dataset [5] with 1.1 billion masks at some point in their training. On SBD and SHSeg our model even outperforms all other methods. In general, however, SkipClick delivers the best performance and is thus the only method with better results on most of the datasets.

We also want to see how our method performs in comparison to SkipClick if both are given multi-modal information. For this reason, we apply our multi-modal fusion strategy to SkipClick by integrating the MMFuser module. A description on how we do this can be found in section H in the supplementary material. In Table IIIc we carry out the comparison on the DeLiVER dataset with depth maps as the additional modality. Our model outperforms SkipClick in the multi-modal setting. We thus assume our model to give us the best performance in a multi-modal setting.

VI. Conclusion

In this paper, we introduced a system for interactive segmentation with multiple image-like input modalities. Our system is constructed to adhere to certain constraints. We want our model to run fast and we want the image feature computation to be completely decoupled from the integration of interactions. For this reason we opt for a late fusion model.

On top of this, we introduce a metric that considers interactive segmentation with multiple, possibly adjacent surfaces per image. We propose a multi-modal feature fusion strategy that is capable of dealing with a black-box RGB backbone. We are able to show that this fusion strategy incurs improvements for multiple RGB backbones on multiple datasets for different non-RGB modalities. Finally, we are able to show that our RGB-only baseline offers competitive, and in some cases even superior performance for regular interactive segmentation tasks. We even outperform the currently best model in a multi-modal setting, when integrating our multi-modal fusion strategy into its architecture.

REFERENCES

[1] Feng Li, Hao Zhang, Huaizhe Xu, Shilong Liu, Lei Zhang, Lionel M Ni, and Heung-Yeung Shum. Mask dino: Towards a unified transformer-based framework for object detection and segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3041–3050, 2023.

[2] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Neural Information Processing Systems (NeurIPS)*, 2021.

[3] Daniel Kienzle, Marco Kantonis, Robin Schön, and Rainer Lienhart. Segformer++: Efficient token-merging strategies for high-resolution semantic segmentation. *IEEE International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2024.

[4] Konstantin Sofiiuk, Ilya A Petrov, and Anton Konushin. Reviving iterative training with mask guidance for interactive segmentation. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3141–3145. IEEE, 2022.

[5] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[6] Xi Chen, Zhiyan Zhao, Yilei Zhang, Manni Duan, Donglian Qi, and Hengshuang Zhao. Focalclick: Towards practical interactive image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1300–1309, 2022.

[7] Mathias M. Lowes, Jakob L. Christensen, Bjørn Schreblowski Hansen, Morten Rieger Hannemose, Anders B. Dahl, and Vedrana Dahl. Interactive scribble segmentation. In *Proceedings of the Northern Lights Deep Learning Workshop*, volume 4, 2023.

[8] Shiyin Zhang, Jun Hao Liew, Yunchao Wei, Shikui Wei, and Yao Zhao. Interactive object segmentation with inside-outside guidance. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12234–12244, 2020.

[9] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8623–8632, 2020.

[10] Qin Liu, Zhenlin Xu, Gedas Bertasius, and Marc Niethammer. Simpleclick: Interactive image segmentation with simple vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22290–22300, October 2023.

[11] Lei Ke, Mingqiao Ye, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, Fisher Yu, et al. Segment anything in high quality. *Advances in Neural Information Processing Systems*, 36, 2024.

[12] You Huang, Hao Yang, Ke Sun, Shengchuan Zhang, Liujuan Cao, Guannan Jiang, and Rongrong Ji. Interformer: Real-time interactive image segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22301–22311, 2023.

[13] Qin Liu, Jaemin Cho, Mohit Bansal, and Marc Niethammer. Rethinking interactive image segmentation with low latency high quality and diverse prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3773–3782, 2024.

[14] Robin Schön, Julian Lorenz, Daniel Kienzle, and Rainer Lienhart. Skipclick: Combining quick responses and low-level features for interactive segmentation in winter sports contexts. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV) Workshops*, pages 1247–1256, February 2025.

[15] Eirikur Agustsson, Jasper RR Uijlings, and Vittorio Ferrari. Interactive full image segmentation by considering all regions jointly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11622–11631, 2019.

[16] Mykhaylo Andriluka, Stefano Pellegrini, Stefan Popov, and Vittorio Ferrari. Efficient full image interactive segmentation by leveraging within-image appearance similarity. *arXiv preprint arXiv:2007.08173*, 2020.

[17] Amit Kumar Rana, Sabarinath Mahadevan, Alexander Hermans, and Bastian Leibe. Dynamite: Dynamic query bootstrapping for multi-object interactive segmentation transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1043–1052, 2023.

[18] Kun Li, Hao Cheng, George Vosselman, and Michael Ying Yang. Learning from exemplars for interactive image segmentation. *arXiv preprint arXiv:2406.11472*, 2024.

[19] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers. Fusenet: incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Asian Conference on Computer Vision*, November 2016.

[20] Qishen Ha, Kohei Watanabe, Takumi Karasawa, Yoshitaka Ushiku, and Tatsuya Harada. Mfnet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5108–5115. IEEE, 2017.

[21] Yuxiang Sun, Weixun Zuo, and Ming Liu. RTFNet: RGB-Thermal Fusion Network for Semantic Segmentation of Urban Scenes. *IEEE Robotics and Automation Letters*, 4(3):2576–2583, July 2019.

[22] Jinyuan Liu, Zhu Liu, Guanyao Wu, Long Ma, Risheng Liu, Wei Zhong, Zhongxuan Luo, and Xin Fan. Multi-interactive feature learning and a full-time multi-modality benchmark for image fusion and segmentation. In *International Conference on Computer Vision*, 2023.

[23] Jiaming Zhang, Huayao Liu, Kailun Yang, Xinxin Hu, Ruiping Liu, and Rainer Stiefelhagen. Cmx: Cross-modal fusion for rgb-x semantic segmentation with transformers. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[24] Yikai Wang, Xinghao Chen, Lele Cao, Wenbing Huang, Fuchun Sun, and Yunhe Wang. Multimodal token fusion for vision transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[25] Yupeng Liang, Ryosuke Wakaki, Shohei Nobuhara, and Ko Nishino. Multimodal material segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19800–19808, June 2022.

[26] Jiaming Zhang, Ruiping Liu, Hao Shi, Kailun Yang, Simon Reiß, Kunyu Peng, Haodong Fu, Kaiwei Wang, and Rainer Stiefelhagen. Delivering arbitrary-modal semantic segmentation. In *CVPR*, 2023.

[27] Tim Broedermann, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Hrfuser: A multi-resolution sensor fusion architecture for 2d object detection. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2023.

[28] Tim Brödermann, Christos Sakaridis, Yuqian Fu, and Luc Van Gool. Cafuser: Condition-aware multimodal fusion for robust semantic perception of driving scenes. *IEEE Robotics and Automation Letters*, 2025.

[29] Ding Jia, Jianyuan Guo, Kai Han, Han Wu, Chao Zhang, Chang Xu, and Xinghao Chen. GeminiFusion: Efficient pixel-wise multimodal fusion for vision transformer. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 21753–21767. PMLR, 21–27 Jul 2024.

[30] Bingyu Li, Da Zhang, Zhiyuan Zhao, Junyu Gao, and Xuelong Li. Stitchfusion: Weaving any visual modalities to enhance multimodal semantic segmentation. *arXiv preprint arXiv:2408.01343*, 2024.

[31] Robin Schön, Katja Ludwig, and Rainer Lienhart. Impact of pseudo depth on open world object segmentation with minimal user guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4809–4819, June 2023.

[32] Bin Wang, Anwesa Choudhuri, Meng Zheng, Zhongpai Gao, Benjamin Planche, Andong Deng, Qin Liu, Terrence Chen, Ulas Bagci, and Ziyan Wu. Order-aware interactive segmentation, 2025.

[33] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, pages 1–31, 2024.

[34] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers, 2023.

[35] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollar, and Ross Girshick. Masked autoencoders are scalable vision learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988. IEEE Computer Society, 2022.

[36] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive

architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023.

[37] 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 *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014.

[38] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[39] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[40] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.

[41] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, pages 991–998, 2011.

[42] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423 vol.2, 2001.

[43] Kevin McGuinness and Noel E. O'Connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recogn.*, 43(2):434–444, February 2010.

[44] You Huang, Wenbin Lai, Jiayi Ji, Liujuan Cao, Shengchuan Zhang, and Rongrong Ji. Hrsam: Efficient interactive segmentation in high-resolution images, 2024.

[45] Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.

[46] Robin Schön, Julian Lorenz, Katja Ludwig, and Rainer Lienhart. Adapting the segment anything model during usage in novel situations. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3616–3626, 2024.

[47] Wensheng Gan, Shicheng Wan, and Philip S. Yu. Model-as-a-Service (MaaS): A Survey . In *2023 IEEE International Conference on Big Data (BigData)*, pages 4636–4645, Los Alamitos, CA, USA, December 2023. IEEE Computer Society.

[48] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[49] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

[50] Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In *Proceedings of ICML*, 2022.

[51] Amazon. User guide for amazon nova. Accessed: 09-07-2025, https://docs.aws.amazon.com/pdfs/nova/latest/userguide/nova-ug.pdf.

[52] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[53] Kangpeng Hu, Quansen Sun, Yinghui Sun, and Tao Wang. Interactive segmentation by considering first-click intentional ambiguity. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 4823–4831, 2024.

[54] Cilin Yan, Haochen Wang, Jie Liu, Xiaolong Jiang, Yao Hu, Xu Tang, Guoliang Kang, and Efstratios Gavves. Piclick: Picking the desired mask from multiple candidates in click-based interactive segmentation. *Neurocomputing*, 599:128083, 2024.

[55] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[56] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[57] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

APPENDIX

A. Introduction to the Interactive Segmentation Problem

In this section, we will introduce the interactive segmentation problem by describing the inputs and outputs of our interactive segmentation model. Afterwards we will describe the NoC (Number of Clicks) metric. The NoC is generally used to measure the performance of interactive segmentation systems based on clicks.

1) Interactive Segmentation Task: In the standard interactive segmentation scenario, we assume to have an RGB image $\mathbf{x}_{\text{img}} \in \mathbb{R}^{H \times W \times 3}$. This image contains an object whose surface we want to segment. More precisely, we want to create a mask $\mathbf{m} \in \{0,1\}^{H \times W}$ with $m_{i,j} = 1$ if and only if the pixel (i,j) belongs to the desired surface, while $m_{i,j} = 0$ otherwise. In order to create such a mask, we will leverage a neural network N_{IntSeg}. N_{IntSeg} has been trained to predict high-quality masks by using a certain form of user guidance. In this paper, the only form of user interaction are iterative clicks. The user will repeatedly carry out the following steps:

1) The user is shown the currently estimated mask $\mathbf{m}_\tau$, with τ being the index of the current round. In the initial round, we will not have a current estimate of the mask. Therefore we define $\mathbf{m}_0$ to only consist of 0s (background only).
2) In case the user judges the mask to be of insufficient quality, they place a corrective click $\mathbf{p}_\tau = (i_\tau, j_\tau, l_\tau)$ on a falsely annotated position. Such a click consists of a coordinate $(i_\tau, j_\tau) \in \{1, ..., H\} \times \{1, ..., W\}$ and a label $l_\tau \in \{+, -\}$. If the user places the click with the left mouse button, they indicate that the position belongs to the foreground ($l_\tau = +$), while the right mouse button indicates a background position ($l_\tau = -$).
3) The network predicts a new mask

$$\mathbf{m}_{\tau+1} = N_{\text{IntSeg}}(\mathbf{x}_{\text{img}}, \mathbf{p}_{0:\tau}, \mathbf{m}_\tau), \tag{12}$$

where $\mathbf{p}_{0:\tau}$ are all so far accumulated clicks and $\mathbf{m}_\tau$ is the previous faulty mask.

These steps are carried out repeatedly until the user arrives at a resulting mask $\mathbf{m}_{\text{Res}}$ of sufficient quality.

2) NoC Metric: The mode of interaction from the previous paragraph strongly insinuates that the quality of the mask can be easily increased by adding more click annotations. If the user were to just continue clicking long enough, the mask could likely reach an arbitrary level of quality. The most commonly used metric to measure the performance of interactive segmentation systems follows this intuition. The NoC@Θ_{IoU} metric measures the minimal Number of Clicks that is necessary to create a mask of sufficient quality. We consider a predicted mask $\mathbf{m}_\tau$ to be of sufficient quality if its IoU with an existing ground truth mask $\mathbf{m}_{\text{GT}}$ reaches or surpasses a given threshold $\Theta_{\textbf{IoU}}$. Formally, this is the case if

$$\text{IoU}(\mathbf{m}_\tau, \mathbf{m}_{\text{GT}}) = \frac{|\mathbf{m}_\tau \cap \mathbf{m}_{\text{GT}}|}{|\mathbf{m}_\tau \cup \mathbf{m}_{\text{GT}}|} \cdot 100 \geq \Theta_{\text{IoU}}. \tag{13}$$

It should be noted that we follow common practice [4], [5], [14], [44] and express the IoU in percentage points in a range of $[0, 100] \subset \mathbb{R}$ instead of a ratio in the range of $[0, 1] \subset \mathbb{R}$.

It remains possible that the network is not capable of segmenting a surface with a satisfying degree of quality at all. In order to account for such cases, the number of clicks is usually capped to a value of $n_{\text{max}} = 20$. If the number of clicks exceeds n_{max}, we consider the attempt to segment the object as a failure and use n_{max} as a surrogate value in computing the average NoC on a dataset. In some works the authors decide to either measure the number of failures [4], [13] or the percentage of failures (i.e. the failure rate) [46]. As we do not have an actual human user at our disposal during the testing process, the selection of the click placement is simulated automatically. We follow the simulation strategy described by [4].

B. Oursourcing Subtasks for Interactive Segmentation

1) Scenario Discussion: We now discuss the possibility to outsource subtasks of interactive segmentation and why it is beneficial to be able to deal with a black-box RGB backbone. Recently, a number of companies have trained large foundation models that are deliberately kept private [47]–[50]. This phenomenon also extends to image processing backbones being offered as a service. A real-world example of this is Amazon Nova [51]. We assume this trend to continue, leading to private foundation models with increasing performance being offered as a service. As we have no direct access to them, these models can only be used as a black-box. Training or altering such a model is not possible.

In our scenario, the RGB backbone FM_{RGB} for image feature extraction is such an external foundation model. The external party providing the foundation model service will be called *foundation model provider*. The scenario is depicted in Figure 5. Therein, we are the *operating institution*, and will act on behalf of our interest. We can send RGB images $\mathbf{x}_{\text{img}}$ to the foundation model provider and receive a set of image features $\mathcal{F}_{\text{img}}$. Most multi-modal fusion strategies for segmentation assume the backbone to be either trainable or at least allow backpropagation through the network as a precondition. As this is not possible in our scenario, we will devise an asymmetric multi-modal fusion strategy that does not depend on the trainability and internal workings of the RGB backbone. We are thus able to effectively treat the RGB backbone as a black-box whilst still profiting from the additional knowledge from other modalities.

Since our method is designed to be capable of dealing with a black-box, using an external foundation model as an RGB backbone is no obstacle during training or during usage:

- **During training**, we send all RGB images in our training dataset to the foundation model provider for feature extraction. The *foundation model provider* responds with the extracted features $\mathcal{F}_{\text{img}}$ for our entire dataset. This feature extraction can be executed once in bulk before the training even starts. We thus do not have to attend it or wait for a response during training. On our side, we

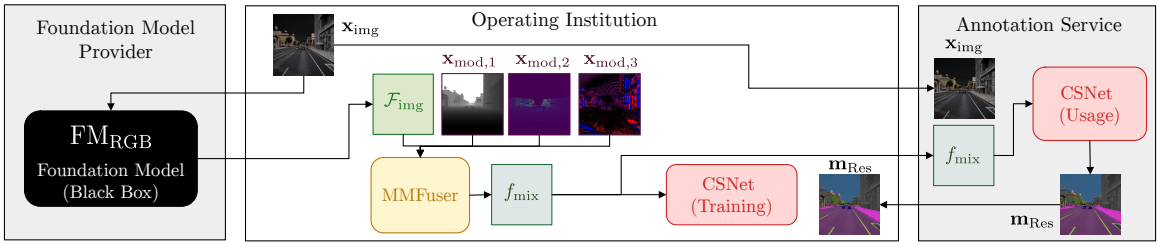


Fig. 5: A scenario for interactive segmentation where the RGB foundation model and the the annotation task have been outsourced. In this scenario, we are the *operating institution* and will act on behalf of our interests. The RGB foundation model FM_{RGB} has been outsourced to an external *foundation model provider*. This makes FM_{RGB} a black-box. We first use FM_{RGB} to extract features $\mathcal{F}_{\text{img}}$ from RGB images $\mathbf{x}_{\text{img}}$. Afterwards, we integrate the information from non-RGB modalities $\mathbf{x}_{\text{mod},i}$ using our MMFuser. From MMFuser, we obtain a mixed feature tensor f_{mix}. f_{mix} is the input to CSNet, along with the clicks to generate the masks. The annotation task itself is outsourced to an external *annotation service*, which only receives $\mathbf{x}_{\text{img}}$ and f_{mix}, and gives us the high-quality result masks $\mathbf{m}_{\text{Res}}$. As f_{mix} already contains all non-RGB information, we can avoid giving the non-RGB modalities $\mathbf{x}_{\text{mod},i}$ to the annotation service. This is beneficial, as we consider $\mathbf{x}_{\text{mod},i}$ valuable data. A detailed discussion of our network and modules such as MMFuser and CSNet can be found in the main paper's method section.

can effectively train all other components of the network on the extracted features $\mathcal{F}_{\text{img}}$.

- **During usage**, we usually use interactive segmentation systems to annotate ground truth masks for entirely new datasets. We can send the images of this new dataset to the foundation model provider, which will in turn send the extracted features $\mathcal{F}_{\text{img}}$ to us. Similar to the training, this feature extraction can be carried out in bulk without any user attending to it. In case we also have non-RGB modalities, we then use the MMFuser module to generate the mixed feature tensor f_{mix}. Both, $\mathcal{F}_{\text{img}}$ and f_{mix}, only need to be created once per image. The interactive annotation itself can then be repeatedly carried out using CSNet.

It should be noted that our framework based on a black-box RGB backbone does not actually necessitate the use of an external foundation model service. In only offers the possibility to do so. The RGB backbone may just as well be present locally. In fact, during our experiments, we only use local RGB backbones FM_{RGB}. We will simulate FM_{RGB} being an external black-box by freezing the model and refraining from any architectural alterations.

Since we want to annotate data in an interactive segmentation setting, we also assume the existence of a third party, the *annotation service*. Once we receive the extracted RGB features from the foundation model provider, we will enrich those features with additional modalities using a multi-modal fusion strategy. These enriched features are then sent to the *annotation service*. The external *annotation service* will only receive the RGB image, the enriched features, and a small interactive segmentation network that operates on the features and user clicks (see Fig. 5).

2) Notes on used Network Bandwidth: First, it should be mentioned that the size of tensors fabricated by foundation model providers, such as AWS Nova or LeewayHertz, strongly depends on what the user demands. In fact, the handbook for AWS Nova, explicitly mentions this [51] and lists examples for image resolution and resulting token numbers. For example, an image with a resolution of 900×900 results in approximately 1300 tokens according to the handbook. From this, we can infer their approximate patch size as

$$\frac{900}{\sqrt{1300}} \approx 25. \tag{14}$$

Assuming that the images have a resolution of 448×448, we would arrive at a spatial token resolution of 18×18. With an embedding dimension of 1280, a float32 encoding, and 4 extracted tensor per image, this implies the transfer of

$$18 \cdot 18 \cdot 1280 \cdot 4 \cdot 32 \text{ bits} = 53084160 \text{ bits} \tag{15}$$

per image. With a consumer connection of 500 mbit per second, a single image only needs 0.1061 seconds for transfer, which is much faster than any user could annotate an image. On top of this, the operation can be carried out in bulk without the user attending to it. We thus do not assume network bandwidth to be a problem.

C. Differences to Other Work Related to Interactive Segmentation for Multiple Surfaces in an Image

The main paper does not present the very first work on interactive segmentation with multiple surface in the same image. However, we deem all previous work to be too different from our work to allow for a direct comparison. On top of this, we come to find their interaction mechanisms quite arbitrary. In [15], the authors propose a mechanism where an initial segmentation is generated using specifically chosen border points. Corrections can only be carried out as an extension mechanism for existing regions, making corrections on non-exising regions impossible. The authors of MagicPaint [16] provide a vast variety of input tools such as freeform scribbles, variable stroke thickness and filling mechanisms.

TABLE IV: A comparison of our baseline model with DynaMITe for the single-surface interactive segmentation task. In all cases the metric is the NoC@90, for which a lower value indicates better performance. It should be noted that DynaMITe uses a different backbone than us.

Model	GrabCut	Berkeley	SBD	DAVIS
DynaMITe (Swin-L)	1.72	1.90	5.64	5.09
DynaMITe (Swin-T)	1.78	1.96	6.32	5.23
MMMS (ours, ViT-B)	1.54	2.75	6.10	5.14

In contrast these methods our mechanism only works with clicks. DynaMITe [17] is the only purely click-based system for multiple surfaces in the same image. While our system allows for a sequential creation of masks, DynaMITe enforces a direct manipulation of the multi-surface mask. This direct manipulation requires the user to reconsider their class selection after every click. Additionally, the multi-surface aspect and the accompanying interaction mechanism are deeply connected to the architecture of DynaMITe. Thus, the interaction mechanism and the metrics within the multi-surface case of the DynaMITe system are too different from our scenario to allow for a meaningful performance comparison. However, as the DynaMITE provides results for single-surface interactive segmentation, we can make a comparison for the single-surface case. The comparison can be found in Table IV. Nonetheless, it should be noted that DynaMITe uses the considerably different Swin backbone [52].

In addition to the aforementioned methods, there is also work that is marginally related to our paper. iCMFormer++ [18] is specialized for cases in which multiple object instances of the same class are present in the same image. The segmentation is carried out in two rounds. In the first round, a segmentation mask for a single object instance of the targeted class is created. Using the created mask, this object is then cropped from the image and given to the network in the second round. In the second round, the network is used to create a semantic mask covering all objects of that class as a single surface.

There is also work dealing with target ambiguity in interactive segmentation, such as DISNet [53] and PiClick [54]. Target ambiguity deals with the problem that a single click does not clearly tell the network which surface to segment. *E.g. if I click on the tire of a car, do I want to to only segment the tire, or the complete car?* In contrast to this, multi-surface segmentation deals with the overlap of multiple annotated masks per image. Although somewhat related, these papers tackle different challenges within interactive segmentation.

D. Further Notes on Computational Efficiency

In the main paper, we already include some remarks on the response time. We use an Nvidia V100 as the GPU and a Intel®Xeon®CPU E5-2697 v4 as the CPU. If we amortize the duration of the feature extraction over all clicks, our model takes 42 ms per click on average when tested on the MFNet dataset using the GPU. If we only consider the response time in isolation, we measure 12 ms when averaged over 30 clicks. On the CPU, we arrive at 206 ms per click. We thus consider our model to be real-time-capable.

TABLE V: We compare the response time or our model with measurements from other models from [12].

Model	Device	Response Time
Interformer-Light (ViT-B)	CPU	0.19
Interformer-Tiny (ViT-L)	CPU	0.32
SimpleClick (ViT-B)	CPU	1.51
SimpleClick (ViT-L)	CPU	3.33
SimpleClick (ViT-H)	CPU	7.76
MMMS (ours, ViT-B)	CPU	0.206
MMMS (ours, ViT-B)	GPU	0.012

TABLE VI: We measure the occupied VRAM during evaluation and count the number of parameters. The first row only uses RGB as input. In the rows beneath, we progressively add Depth, Lidar, and Event camera images.

Modalities	VRAM (MiB)	# params (mio.)
RGB	1474	125.2
RGB+D	1582	144.1
RGB+D+L	1686	159.7
RGB+D+L+E	1772	175.3

Here, we extend our discussion regarding the computational efficiency of our model. We compare our response time with response times of existing models using the measurements from [12]. It should be noted that the authors of [12] did not disclose which CPU they used. The comparison can be found in Table V.

A second aspect of computational efficiency lies in the memory usage. We measure the occupied VRAM during evaluation on our NVIDIA V100 and count the number of parameters in Table VI. We do this for the RGB-only case and progressively add depth maps, lidar and event camera images. We choose the DINOv2-ViT-B based model as it delivered the best performance.

E. Combining the Surface-Specific Masks into a Joint Mask

Whenever we segment multiple surfaces in the same image, we are able to combine them to a joint mask. This joint mask assigns each pixel as either belonging to one of the surfaces or to non of them (background). In this section, we will describe how the multi-surface interactive segmentation problem from the main paper relates to the construction of such a joint mask. We will first describe how the joint mask is constructed in context of the classical interactive segmentation problem. Afterwards, we extend the procedure of constructing a joint mask to the multi-surface evaluation framework where some masks may be revisited. We will also discuss how the surface-specific masks are extracted from the joint mask.

1) Joint Masks in Classical Interactive Segmentation: We now discuss how a joint mask is constructed, when we work with the assumptions of classical interactive segmentation. Let

L be the number of surfaces that we want to segment. As previously stated in the main paper, we want to create a set

$$\mathcal{S}_{\mathbf{m}} = \left\{ \mathbf{m}^{(1)}, ..., \mathbf{m}^{(L)} \right\}. \tag{16}$$

of segmentation masks. In the setting of classical interactive segmentation, each of these masks is considered in isolation. To create a joint mask $\mathbf{m}^{\text{joint}} \in \{0, 1, ..., L\}^{H \times W}$, the surface-specific masks are created sequentially and pasted onto the joint mask. We start out with an initial joint mask $\mathbf{m}^{\text{joint},0}$ that is completely filled with 0s. In our case a pixel-value of 0 indicates that the pixel belongs to the background, whereas a pixel-value $l = 1, ..., L$ indicates the pixel belongs to the l-th surface. As soon as the l-th mask is created, we update the pixels (i, j) in the previous joint mask with the rule

$$m_{i,j}^{\text{joint},l} = \begin{cases} l & , \text{ if } m_{i,j}^{(l)} = 1. \\ m_{i,j}^{\text{joint},l-1} & , \text{ if } m_{i,j}^{(l)} = 0. \end{cases} \tag{17}$$

In this way, the masks are each constructed and added to the joint mask. As each mask is considered in isolation, no mask is revisited. We eventually end up with the final joint mask $\mathbf{m}^{\text{joint}} = \mathbf{m}^{\text{joint},L}$.

2) Joint Masks in Multi-Surface Interactive Segmentation: In our extended multi-surface evaluation mechanism, the masks for some surfaces may be revisited for correction. Instead of simply adding masks in a linear order $l = 1, ..., L$ we continuously pick the worst mask beloning to surface k and improve it. Afterwards, the improved mask can be used to update the joint mask $\mathbf{m}^{\text{joint},\lambda-1}$ to $\mathbf{m}^{\text{joint},\lambda}$. Note that we change the index notation from l to λ to avoid confusion. We also have to extend our update rule. Once we consider that some surfaces may be revisited, we also have the possibility of decreasing the size of a surface. Assume we currently correct the mask for surface k and by doing so remove a few pixels that previously belonged to that surface in the joint mask. In this case, the removed pixels are set to the background class in the updated joint mask. Our extended update rule for pixels (i, j) is

$$m_{i,j}^{\text{joint},\lambda+1} = \begin{cases} k & , \text{ if } m_{i,j}^{(k)} = 1. \\ 0 & , \text{ if } (m_{i,j}^{(k)} = 0) \wedge (m_{i,j}^{\text{joint},\gamma} = k). \\ m_{i,j}^{\text{joint},\lambda} & , \text{ if } (m_{i,j}^{(k)} = 0) \wedge (m_{i,j}^{\text{joint},\gamma} \neq k). \end{cases} \tag{18}$$

Finally, we mention how we extract a binary mask $\mathbf{m}^{(k)}$ for the k-th surface of the joint mask. We take all pixel belonging to that surface as foreground pixels and the rest as background pixels. We extract pixels (i, j) according to the rule.

$$m_{i,j}^{(k)} = \begin{cases} 1 & , \text{ if } m_{i,j}^{\text{joint}} = k. \\ 0 & , \text{ otherwise.} \end{cases} \tag{19}$$

F. Further Implementation Details

As we want to show the general efficacy of our multi-modal fusion strategy, we use datasets that offer different non-RGB modalities. The first dataset we use in the main paper is the synthetic DeLiVER dataset [23], [26] with the split from [26]. In addition to the RGB images, DeLiVER contains depth maps, lidar and event camera images as additional modalities. We use the lidar representation from [26]. The second dataset in the main paper is MFNet [20], which provides thermal images as an additional modality. For MFNet we use the splits provided with the dataset. Furthermore, we test our method on FMB [22] and MCubeS [25] for which the results can be found in the supplementary material.

We train for 100 epochs on DeLiVER and MFNet, while training for 400 epochs on FMB and MCubeS. We found these training durations promising in preliminary experiments. We use the Adam optimizer [55] with a batch size of 8, a learning rate $\mu = 5 \cdot 10^{-5}$ and $\beta_1 = 0.9, \beta_2 = 0.999$. If we train for 100 epochs, a scheduler reduces the learning by $\frac{1}{10}$ at epochs 95 and 100. If we train for 400 epochs, this reduction happens at epochs 390 and 400. During training and testing, we use a resolution of 448×448 unless said otherwise. During training, this resolution is imposed by random crops. During testing, we follow common practice [4], [10] and rescale to this size.

Our multi-modal fusion strategy works on a variety of vision foundation models. We test it on four different backbones as our RGB foundation model FM_{RGB}. All of these models are vision transformers [56]. The first model is a ViT-B that has been pretrained with DINOv2 [33], [34], and has a patch size of 14. The second model is a ViT-B pretrained with the masked auto-encoder (MAE) [35] framework. We also use the ViT-B encoder that has been trained as part of the Segment Anything Model (SAM) [5], since we assume this backbone to be capable of producing features that are beneficial for segmentation tasks. The SAM backbone only operates on a resolution of 1024×1024. For this reason, we upscale the input images before feeding them to the backbone, and downscale the resulting feature tensors afterwards. On top of this, we use a ViT-H pretrained with IJEPA [36]. Here, we use the *huge (H)* model as no smaller size was available. In order to simulate FM_{RGB} being a black-box, we do not train or backpropagate through the RGB backbones at any time. Our RGB-only baseline is almost identical to our multi-modal model, apart from the MMFuser being removed completely. In all cases, the models receive the RGB images as input. We use a SegFormer-B1 encoder inside the MMFuser and a SegFormer-B0 [2] as the basis for CSNet.

G. Further Results

1) Ablation of the CrossBlock: In Table VII, we can see the results of our ablation study of our CrossBlock on the MFNet dataset. In the first line, neither the efficient cross-attention (EffCA) nor the subsequent MLP are used. In this case, the feature tensors from different modalities are just added. Only adding a single one of the two submodules has different effects depending on how demanding the metric is. Using the efficient cross-attention or the MLP on its own, slightly degrades the performance on the easier NoC@80 metric (8.81 vs 8.93 and 8.88, respectively). Once we use the NoC@90 metric, which demands a higher degree of detail, we also see improvements using a single submodule. This phenomenon can be attributed

TABLE VII: Ablation study of the CrossBlock on MFNet. In all cases, we use the thermal images and the backbone is a DINOv2-B14.

EffCA	MLP	NoC@80 ↓	NoC@90 ↓	NoCMS@(80, 70) ↓
		8.81	15.05	10.11
✓		8.93	15.04	10.23
	✓	8.88	14.93	10.19
✓	✓	**8.66**	**14.88**	**9.88**

TABLE VIII: Results on the FMB dataset [22]. The leftmost column (FM_{RGB}) indicates the backbone we used. The usage of infrared images is indicated by a checkmark. In RGB-only cases the MMFuser is not present.

FM_{RGB}	Infrared	NoC@80 ↓	NoC@90 ↓	NoCMS@(80, 70) ↓
DINOv2-B14		7.86	12.97	8.28
	✓	**7.07**	**11.92**	**7.37**
MAE-B16		8.46	14.00	8.98
	✓	7.56	12.64	8.12
SAM-B16		8.35	13.80	8.89
	✓	7.55	12.53	8.08
IJEPA-H16		9.02	14.52	9.80
	✓	7.72	12.95	8.30

to the fact that the NoC metric is likely to not increase linearly over different IoU thresholds, even when measured on the same model. Using both, EffCA and the MLP, incurs improvements in all metrics.

2) Results on FMBNet: We also evaluate the effectiveness of our multi-modal fusion strategy on the FMB dataset [22]. In addition to RGB images, FMB offers infrared images as a supplementary modality. The results can be found in Table VIII. If we compare different backbones, we see similar results as with other datasets. When taking a look at the NoC@90 metric in the RBG-only setting the ranking is as follows: DINOv2-B14 is best with 12.97, followed by SAM-B16 with 13.80, MAE-B16 with 14.00 and IJEPA-H16 with 14.52. Using the infrared images leads to improvements in all cases. The strongest improvement can be observed on the NoC@90 when using the IJEPA-based backbone. There, the number of clicks is reduced by 1.57 on average. We attribute this to IJEPA as it is generally worse than the other models in its role as FM_{RGB}.

As the DINOv2-based model already produces the best RGB features, the effects of including the information from infrared images are comparatively smaller. On the NoC@90 metric we see an average improvement of 1.05 clicks. The NoCMS@(80, 70) metric is reduced by 0.91. Another aspect that we can observe is that the NoCMS@(80, 70) is always higher than the regular NoC@80. We attribute this to the NoCMS metric accounting for competition between multiple surfaces in the same image. As the NoCMS@(80, 70) metric requires the revisiting of previous surfaces, we assume this metric to be more demanding.

3) Results on MCubeS: The results on the MCubeS dataset [25] can be found in Table IX. In addition to RGB images, this dataset offers three non-RGB modalities: The *Angle of Linear Polarization (AoLP)* and the *Degree of Linear Polarization (DoLP)* as measured by a camera with a polarizaion filter, and *Near Infrared (NIR)* images. RGB images are given to the model in all cases.

For the model with DINOv2-B14 in the role of FM_{RGB}, we first test the effect of using each modality on its own. Our multi-modal fusion mechanism performs best when using the DoLP, which allows the model to reduce the NoC@90 by 0.42, the NoC@80 by 0.63 and the NoCMS@(80, 70) by 0.46. Our model successfully employs all non-RGB modalities when used in isolation. Using all of them at once improves the performance even further, reducing the NoCMS@(80, 70) from 16.77 to 16.13. The failure rate FRMS@(80, 70) is reduced from 65.15 to 61.98. As using all modalities at the same time is the most promising strategy, we also test this for other versions of FM_{RGB}. For MAE the NoC@90 improves from 18.66 to 17.83. When looking at the multi-surface segmentation metric NoCMS@(80, 70), the largest improvement can be observed for IJEPA with a reduction from 17.68 to 16.29. Here, we also see the largest improvement in the FRMS@(80, 70) from 70.95 to 61.90.

H. Integrating MMFuser into SkipClick

In this section, we briefly describe how we integrate MMFuser into the SkipClick architecture. As a complete description of SkipClick would be far beyond the scope of this text, we refer the interested reader to [14] for further details. However, we do illustrate the altered version of SkipClick in Figure 6.

The MMFuser is integrated after the backbone of the SkipClick architecture. As SkipClick uses a DINOv2-B14 as a backbone, the patch size is 14×14 and $d_{\text{model}} = 768$. Let

$$f_1, f_2, f_3, f_4 \in \mathbb{R}^{\frac{H}{14} \times \frac{W}{14} \times 768} \tag{20}$$

be the feature tensors extracted from the intermediate layers of the backbone. We first use a ParallelFPN to convert the features (f_1, f_2, f_3, f_4) to a feature pyramid

$$\left(f^4_{\text{img}}, f^8_{\text{img}}, f^{16}_{\text{img}}, f^{32}_{\text{img}}\right) = \text{ParallelFPN}(f_1, f_2, f_3, f_4). \tag{21}$$

We now have a feature pyramid to which we can apply the MMFuser. We give $\mathbf{x}_{\text{mod}}$ and the feature pyramid to MMFuser and obtain

$$\begin{aligned}\left(f^4_{\text{mm, P}}, f^8_{\text{mm, P}}, f^{16}_{\text{mm, P}}, f^{32}_{\text{mm, P}}\right) = \\ \text{MMFuser}\left(\left(f^4_{\text{img}}, f^8_{\text{img}}, f^{16}_{\text{img}}, f^{32}_{\text{img}}\right), \mathbf{x}_{\text{mod}}\right).\end{aligned} \tag{22}$$

The remaining parts of the SkipClick architecture are designed to operate on tensors in $\mathbb{R}^{\frac{H}{14} \times \frac{W}{14} \times 768}$. Thus, we need to reverse the changes in the tensor shape again. To do so, we apply a second ParallelFPN to which we refer to as *InverseParallelFPN*. The network does not actually constitute the mathematically inverse operation of the regular ParallelFPN. Instead, it consists of multiple scaling modules that each change the shape of a particular stage in our feature pyramid to the original shape of the tensor that has been extracted from the backbone. Similar to the ParallelFPN, each scaling module is based on convolutions, up-/downsampling mechanisms, layer

TABLE IX: Results on the MCubeS dataset [25]. The leftmost column (FM_{RGB}) indicates the backbone we used. The usage of non-RGB modalities is indicated by a checkmark. In RGB-only cases the MMFuser is not present.

FM_{RGB}	AoLP	DoLP	NIR	NoC@60 ↓	NoC@70 ↓	NoC@80 ↓	NoC@90 ↓	NoCMS@(80, 70) ↓	FRMS@(80, 70) ↓
DINOv2-B14				11.94	14.09	16.29	18.35	16.77	65.15
	✓			11.55	13.75	15.93	18.07	16.50	64.36
		✓		11.14	13.38	15.66	17.93	16.31	62.93
			✓	11.10	13.42	15.72	18.01	16.42	64.28
	✓	✓	✓	**10.80**	**13.12**	**15.47**	**17.79**	**16.13**	61.98
MAE-B16				12.84	15.02	17.01	18.66	17.36	69.28
	✓	✓	✓	11.14	13.27	15.63	17.83	16.18	63.57
SAM-B16				12.60	14.72	16.87	18.65	17.21	68.01
	✓	✓	✓	11.05	13.38	15.61	17.89	16.30	62.93
IJEPA-H16				13.33	15.57	17.41	18.84	17.68	70.95
	✓	✓	✓	11.24	13.39	15.65	17.90	16.29	**61.90**

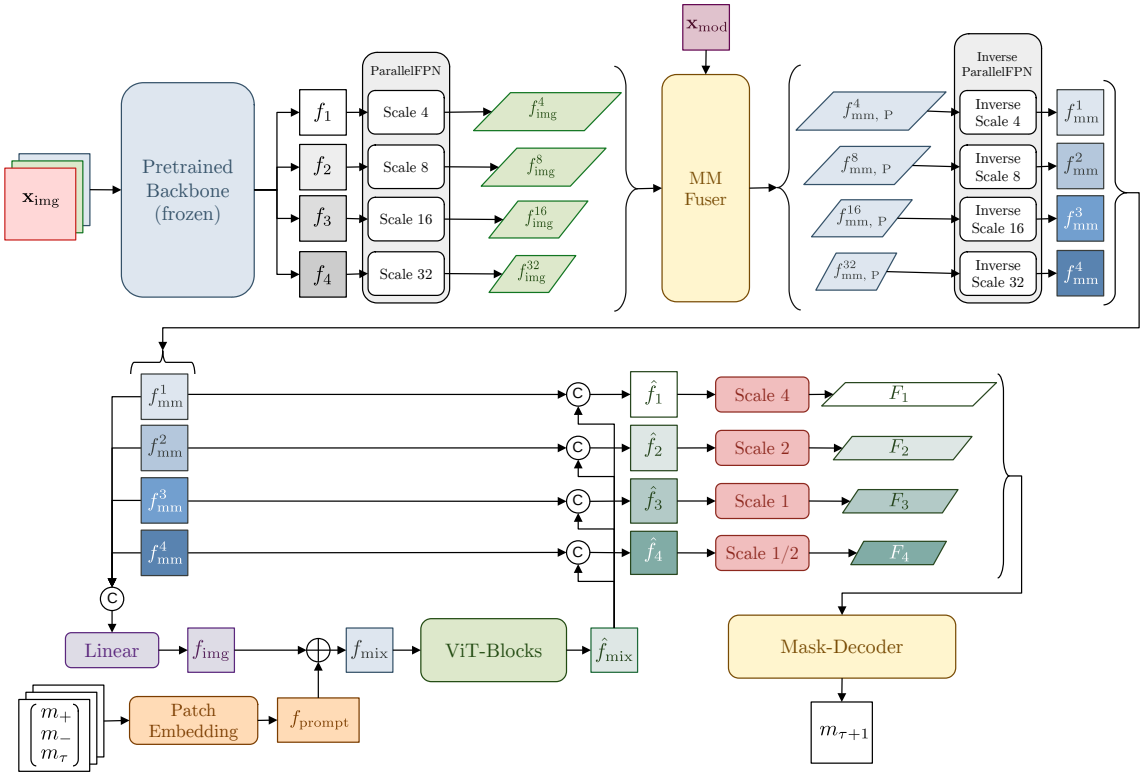


Fig. 6: An extension of the SkipClick architecture to also make use of the MMFuser module. This figure is an extended version of Figure 2b from [14].

normalizations and GELU activations. A detailed description can be found in the supplementary material in section I. We have

$$\left(f^1_{\text{mm}}, f^2_{\text{mm}}, f^3_{\text{mm}}, f^4_{\text{mm}}\right) = \text{InverseParallelFPN}\left(f^4_{\text{mm, P}}, f^8_{\text{mm, P}}, f^{16}_{\text{mm, P}}, f^{32}_{\text{mm, P}}\right). \tag{23}$$

As the shape has been restored to that of the backbones output, we have

$$f^1_{\text{mm}}, f^2_{\text{mm}}, f^3_{\text{mm}}, f^4_{\text{mm}} \in \mathbb{R}^{\frac{H}{14}\times\frac{W}{14}\times 768}. \tag{24}$$

From this stage onward, the remaining part of the architecture stays identical to SkipClick. While Figure 6 provides an overview of the overall architecture, we refrain from a detailed description and refer the interested reader to [14]. Finally, it might be interesting to note that the reason we do not have an extra SkipClick-based backbone in our experiments is that SkipClick uses an unaltered DINOv2-B14 backbone as well.

I. ParallelFPN and InverseParallelFPN

In this section, we describe the *ParallelFPN* used in the main paper. We also describe the *InverseParallelFPN* that is exclusively used when integrating our MMFuser module into SkipClick. It should be noted that our ParallelFPN is inspired by the SimpleFPN used in [10]. Each of the two models consists of four different scaling modules, which are called *Scale* i and *Inverse Scale* i for $i = 4, 8, 16, 32$. Let d_{FM} and P_{FM} be the internal dimensionality of the ViT-based foundation model and its patch size, respectively. If the

original images have the shape $H \times W$, the various tensors of our feature pyramid will have the shape $\frac{H}{i} \times \frac{W}{i}$, and d^i_{embed} will be the respective channel dimension. In our case $d^4_{\text{embed}} = 64, d^8_{\text{embed}} = 128, d^{16}_{\text{embed}} = 320, d^{32}_{\text{embed}} = 512$.

The shape of the scaling modules' input and output tensors changes in the following way:

- The module *Scale i* transforms a feature tensor from the ViT-like representation shape $\mathbb{R}^{\frac{H}{P_{\text{FM}}} \times \frac{W}{P_{\text{FM}}} \times d_{\text{FM}}}$ to a feature pyramid shape $\mathbb{R}^{\frac{H}{i} \times \frac{W}{i} \times d^i_{\text{embed}}}$.
- The module *Inverse Scale i* has the inverse effect on the shape. It transforms a feature tensor from $\mathbb{R}^{\frac{H}{i} \times \frac{W}{i} \times d^i_{\text{embed}}}$ to $\mathbb{R}^{\frac{H}{P_{\text{FM}}} \times \frac{W}{P_{\text{FM}}} \times d_{\text{FM}}}$.

Each of the modules is a feed-forward network, i.e. all of its layers are executed in succession. The architecture of the used modules can be found in Table X. Each of the table cells contains the submodules of a particular scaling module. Within each scaling module the submodules are executed in the order going from top of the table cell to its bottom. It should be noted that the LayerNormalizations have been implemented as GroupNormalizations with a single group to allow for convenient usage of the PyTorch library. The description in the table also makes use of hidden dimensions, which can be computed as follows:

$$d^4_{\text{hidden}} = \max\left\{2 \cdot d^4_{\text{embed}}, \frac{d_{\text{FM}}}{2}\right\} \tag{25}$$

$$d^8_{\text{hidden}} = \max\left\{d^8_{\text{embed}}, \frac{d_{\text{FM}}}{2}\right\} \tag{26}$$

$$d^{16}_{\text{hidden}} = \max\left\{d^{16}_{\text{embed}}, d_{\text{FM}}\right\} \tag{27}$$

$$d^{32}_{\text{hidden}} = \max\left\{d^{32}_{\text{embed}}, 2 \cdot d_{\text{FM}}\right\} \tag{28}$$

The hyperparameters given in Table X are to be interpreted as follows:

- A convolutional layer $\text{Conv}(d_{\text{in}}, d_{\text{out}}, k, s)$ receives d_{in} input channels, produces d_{out} output channels, has a kernel size of $k \times k$ and a stride of s.
- An interpolation layer $\text{Interpolate}(h, w)$ resizes the height and width axes of a tensor to (h, w) by using bilinear interpolation.
- $\text{LayerNormalization}(d)$ carries out a layer normalization on a tensor with d feature channels.
- The activation function GELU [57] has no hyperparameters.

J. Qualitative Results

We also display some qualitative results for DeLiVER [26] in Figure 7, MCubeS [25] in Figure 8, the FMB dataset [22] in Figure 9, and MFNet [20] in Figure 10. In all cases, the first column of each figure always shows the RGB input image with the ground truth plotted onto the image. The last column always shows the RGB image with the prediction and the positive clicks for each surface plotted onto it. We refrain from plotting the negative clicks to avoid clutter.

For each of the figures, the model used to generate the prediction is based on DINOv2-B14 as the RGB foundation backbone. We always choose the version with the maximum number of available modalities. Figure 11 displays examples for conflicts between overlapping masks.

TABLE X: The submodules of the scaling modules in the ParallelFPN and the InverseParallelFPN. Each of the table cells contains the submodules of a particular scaling module. Within each scaling module the submodules are executed consecutively in the order going from top of the cell to its bottom. Any hyperparameters a submodule might need are written in the parentheses next to it.

Submodule	**ParallelFPN**	**InverseParallelFPN**
Scale 4	Interpolate $\left(\frac{H}{8}, \frac{W}{8}\right)$ Conv $\left(d_{\text{FM}}, d^4_{\text{hidden}}, k=3, s=1\right)$ LayerNormalization $\left(d^4_{\text{hidden}}\right)$ GELU Interpolate $\left(\frac{H}{4}, \frac{W}{4}\right)$ Conv $\left(d^4_{\text{hidden}}, \frac{d^4_{\text{hidden}}}{2}, k=3, s=1\right)$ LayerNormalization $\left(\frac{d^4_{\text{hidden}}}{2}\right)$ Conv $\left(d^4_{\text{hidden}}, d^1_{\text{embed}}, k=1, s=1\right)$ LayerNormalization $\left(d^4_{\text{embed}}\right)$ GELU	Interpolate $\left(\frac{H}{8}, \frac{W}{8}\right)$ Conv $\left(d^1_{\text{embed}}, \frac{d^4_{\text{hidden}}}{2}, k=3, s=1\right)$ LayerNormalization $\left(\frac{d^4_{\text{hidden}}}{2}\right)$ GELU Interpolate $\left(\frac{H}{P_{\text{FM}}}, \frac{W}{P_{\text{FM}}}\right)$ Conv $\left(\frac{d^4_{\text{hidden}}}{2}, d^4_{\text{hidden}}, k=3, s=1\right)$ LayerNormalization $\left(d^4_{\text{hidden}}\right)$ Conv $\left(d^4_{\text{hidden}}, d_{\text{FM}}, k=1, s=1\right)$ LayerNormalization $\left(d_{\text{FM}}\right)$ GELU
Scale 8	Interpolate $\left(\frac{H}{8}, \frac{W}{8}\right)$ Conv $\left(d_{\text{FM}}, d^8_{\text{hidden}}, k=3, s=1\right)$ LayerNormalization $\left(d^8_{\text{hidden}}\right)$ Conv $\left(d^8_{\text{hidden}}, d^8_{\text{embed}}, k=1, s=1\right)$ LayerNormalization $\left(d^8_{\text{embed}}\right)$ GELU	Interpolate $\left(\frac{H}{P_{\text{FM}}}, \frac{W}{P_{\text{FM}}}\right)$ Conv $\left(d^8_{\text{embed}}, d^8_{\text{hidden}}, k=3, s=1\right)$ LayerNormalization $\left(d^8_{\text{hidden}}\right)$ Conv $\left(d^8_{\text{hidden}}, d_{\text{FM}}, k=1, s=1\right)$ LayerNormalization $\left(d_{\text{FM}}\right)$ GELU
Scale 16	Conv $\left(d_{\text{FM}}, d^{16}_{\text{hidden}}, k=3, s=1\right)$ Interpolate $\left(\frac{H}{16}, \frac{W}{16}\right)$ LayerNormalization $\left(d^{16}_{\text{hidden}}\right)$ Conv $\left(d^{16}_{\text{hidden}}, d^{16}_{\text{embed}}, k=1, s=1\right)$ LayerNormalization $\left(d^{16}_{\text{embed}}\right)$ GELU	Conv $\left(d^{16}_{\text{embed}}, d^{16}_{\text{hidden}}, k=3, s=1\right)$ Interpolate $\left(\frac{H}{P_{\text{FM}}}, \frac{W}{P_{\text{FM}}}\right)$ LayerNormalization $\left(d^{16}_{\text{hidden}}\right)$ Conv $\left(d^{16}_{\text{hidden}}, d_{\text{FM}}, k=1, s=1\right)$ LayerNormalization $\left(d_{\text{FM}}\right)$ GELU
Scale 32	Conv $\left(d_{\text{FM}}, d^{32}_{\text{hidden}}, k=3, s=1\right)$ Interpolate $\left(\frac{H}{32}, \frac{W}{32}\right)$ LayerNormalization $\left(d^{32}_{\text{hidden}}\right)$ Conv $\left(d^{32}_{\text{hidden}}, d^{32}_{\text{embed}}, k=1, s=1\right)$ LayerNormalization $\left(d^{32}_{\text{embed}}\right)$ GELU	Interpolate $\left(\frac{H}{P_{\text{FM}}}, \frac{W}{P_{\text{FM}}}\right)$ Conv $\left(d^{32}_{\text{embed}}, d^{32}_{\text{hidden}}, k=3, s=1\right)$ LayerNormalization $\left(d^{32}_{\text{hidden}}\right)$ Conv $\left(d^{32}_{\text{hidden}}, d_{\text{FM}}, k=1, s=1\right)$ LayerNormalization $\left(d_{\text{FM}}\right)$ GELU

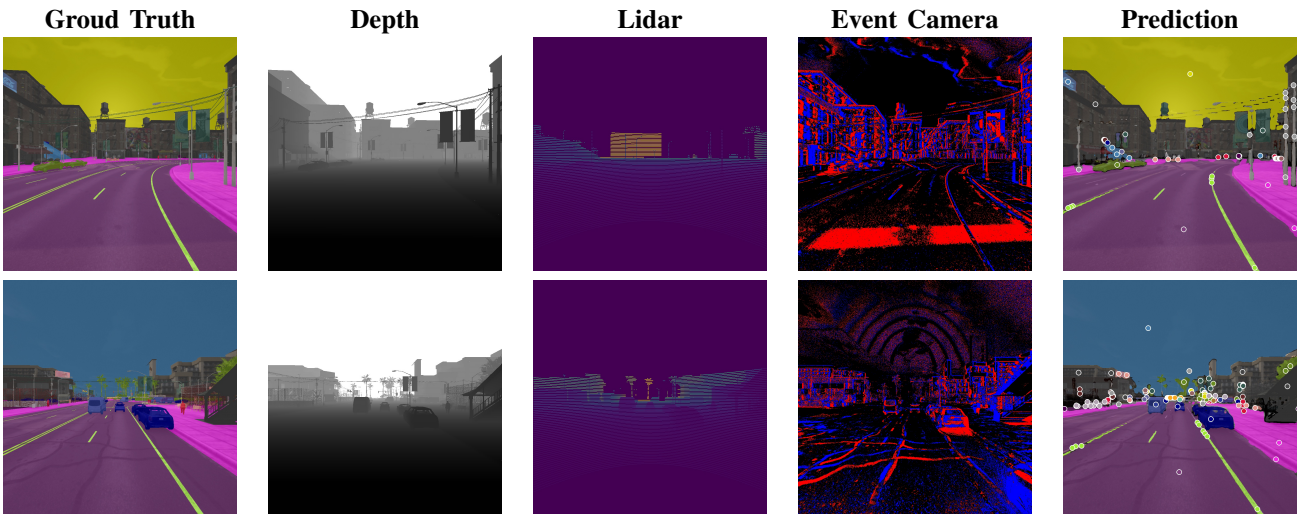


Fig. 7: Qualitative results on the DeLiVER dataset [26]. The model that generated the results is based DINOv2-B14 and has been given all three additional modalities: Depth, lidar, and event camera images. The lidar has been normalized and converted to the viridis color map to allow for better visualization. The first and last image in each row are the RGB image with the ground truth and the prediction plotted onto the image, respectively.

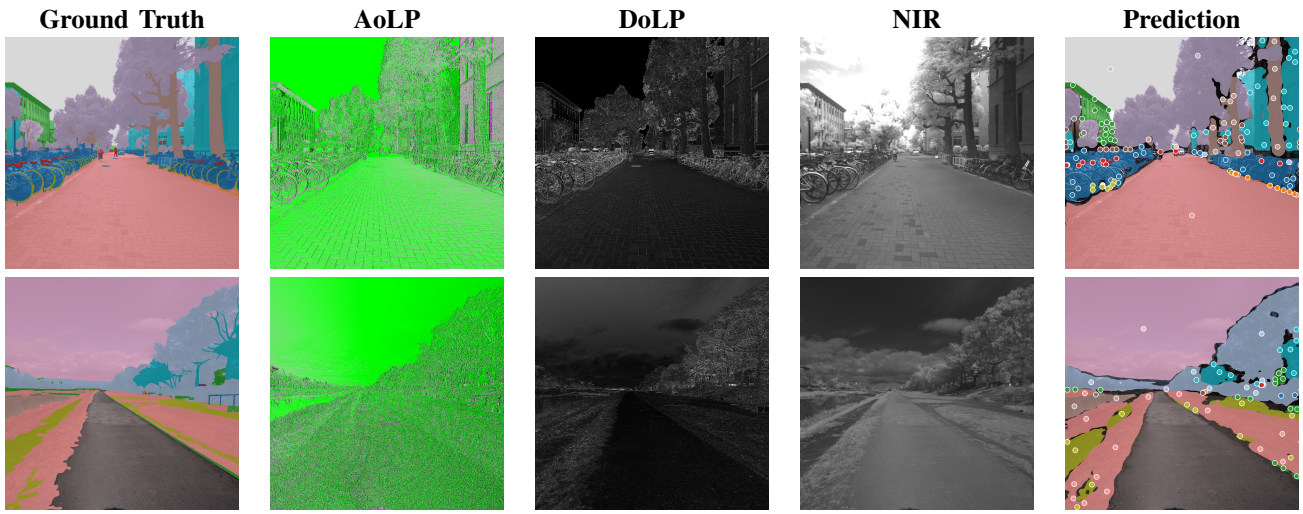


Fig. 8: Qualitative results on the MCubeS dataset [25]. The model that generated the results is based DINOv2-B14. We use *Angle of Linear Polarization (AoLP)*, *Degree of Linear Polarization (DoLP)* and *Near Infrared (NIR)* images as additional modalities. The first and last image in each row are the RGB image with the ground truth and the prediction plotted onto the image, respectively.

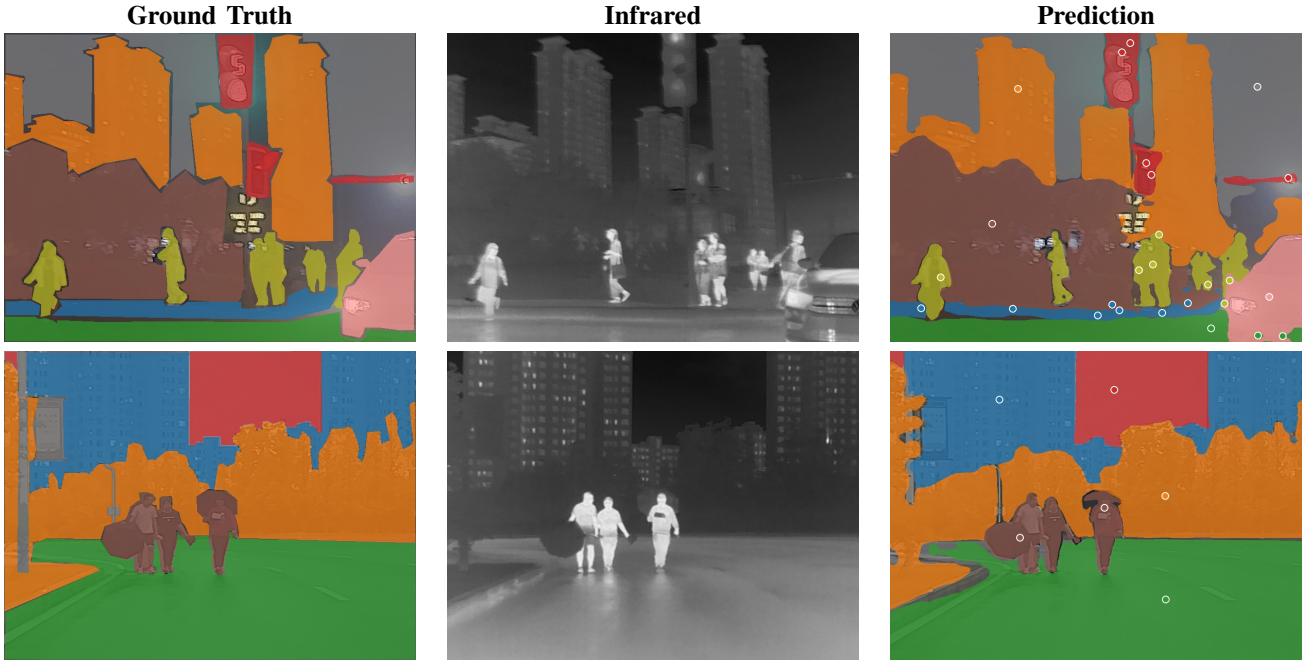


Fig. 9: Qualitative results on the FMB dataset [22]. The model that generated the results is based DINOv2-B14. This dataset offers infrared images as an additional modality. The first and third image in each row are the RGB image with the ground truth and the prediction plotted onto the image, respectively.

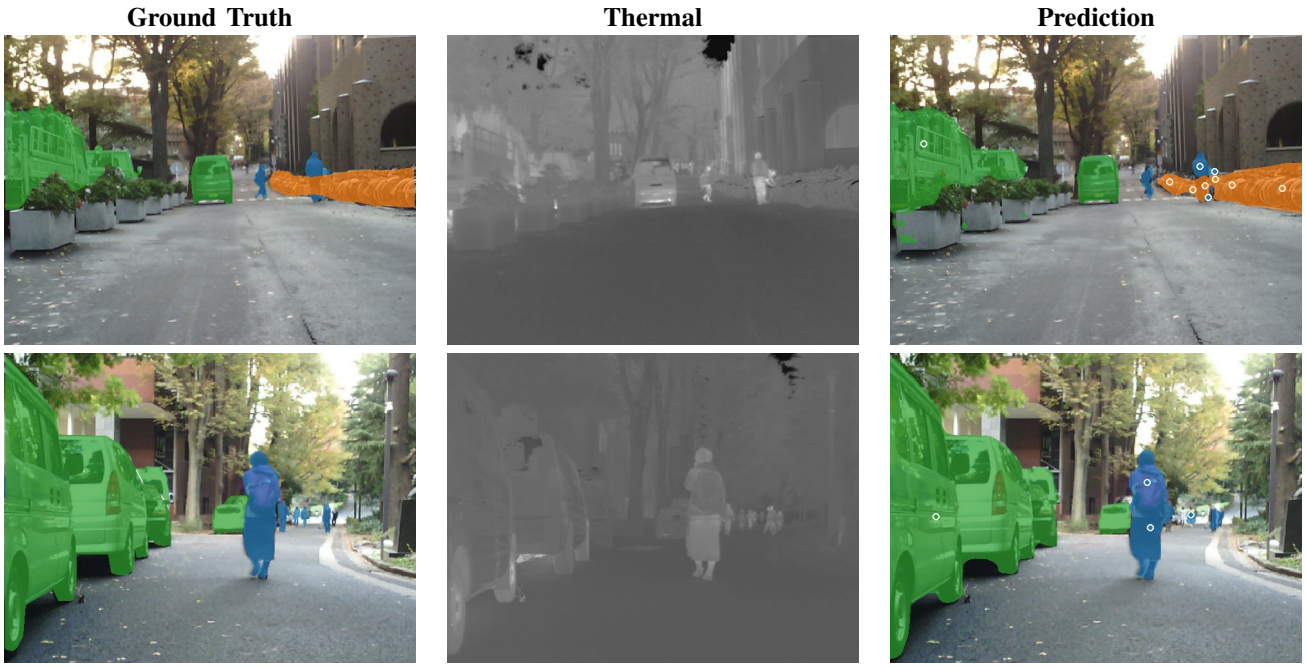


Fig. 10: Qualitative results on the MFNet dataset [20]. The model that generated the results is based DINOv2-B14. This dataset offers thermal images as an additional modality. The first and third image in each row are the RGB image with the ground truth and the prediction plotted onto the image, respectively.

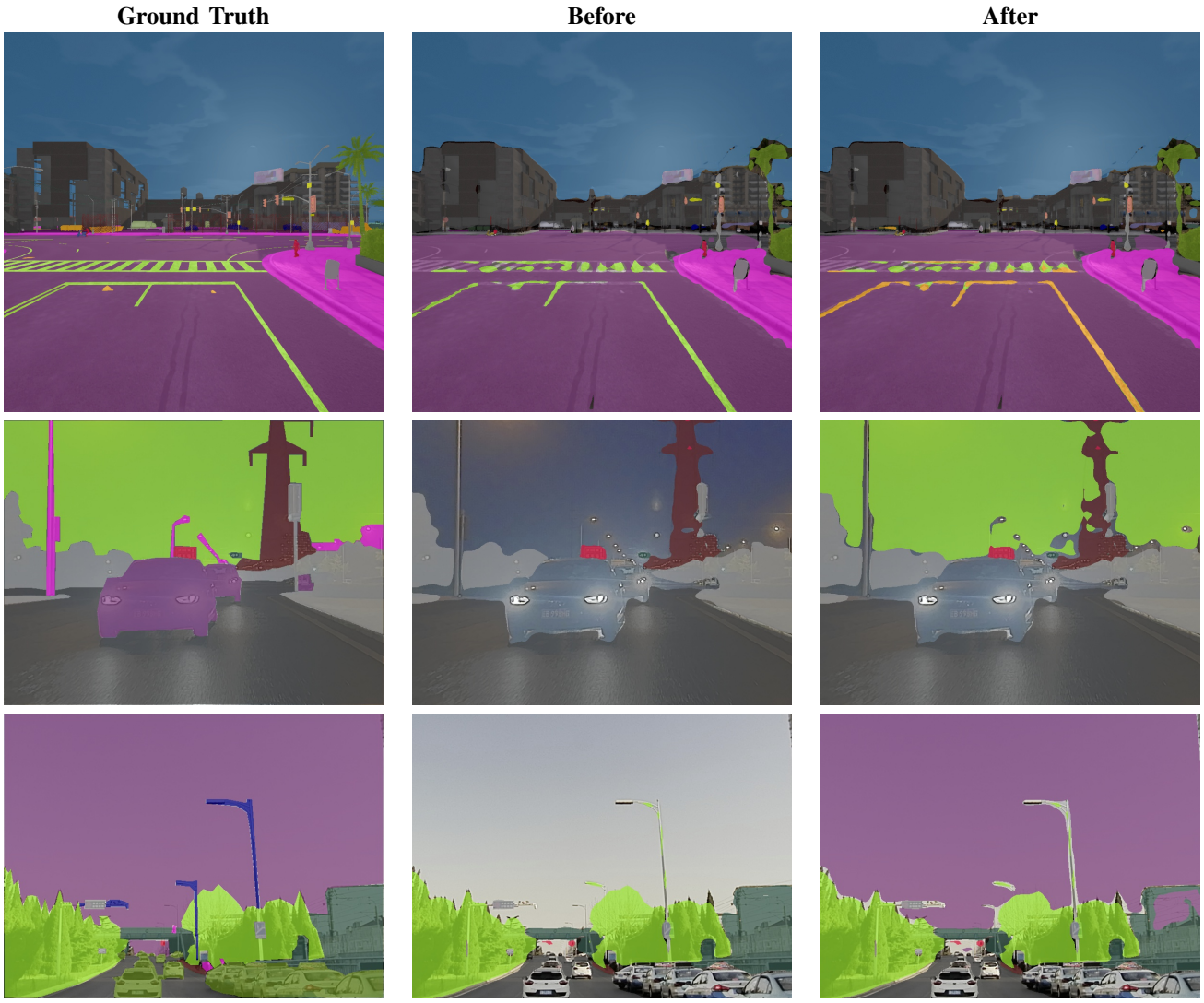


Fig. 11: Examples for conflicts between masks which are caused by false positives in the most recently edited mask. The first row contains examples from DeLiVER [26], while the second and third row are from FMB [22]. The left column displays the ground truth. The middle column displays the joint mask before the overlapping mask has been inserted, and the right column after the overlapping mask has been inserted