# Self-supervised Hypergraphs for Learning Multiple World Interpretations

Alina Marcu[1,2]    Mihai Pirvu[1,2]    Dragos Costea[1,2]    Emanuela Haller[3]

Emil Slusanschi[1]    Ahmed Nabil Belbachir[4]    Rahul Sukthankar[5]    Marius Leordeanu[1,2,4] *

[1]UPB    [2]IMAR    [3]Bitdefender    [4]NORCE    [5]Google Research

## Abstract

*We present a method for learning multiple scene representations given a small labeled set, by exploiting the relationships between such representations in the form of a multi-task hypergraph. We also show how we can use the hypergraph to improve a powerful pretrained VisTransformer model without any additional labeled data. In our hypergraph, each node is an interpretation layer (e.g., depth or segmentation) of the scene. Within each hyperedge, one or several input nodes predict the layer at the output node. Thus, each node could be an input node in some hyperedges and an output node in others. In this way, multiple paths can reach the same node, to form ensembles from which we obtain robust pseudolabels, which allow self-supervised learning in the hypergraph. We test different ensemble models and different types of hyperedges and show superior performance to other multi-task graph models in the field. We also introduce Dronescapes, a large video dataset captured with UAVs in different complex real-world scenes, with multiple representations, suitable for multi-task learning.*

## 1. Introduction

Learning robustly multiple interpretations of the complex world, such as segmentation, depth, and surface information, with minimal human supervision is one of the great challenges in vision today. In this work, we exploit the consensus that naturally appears between such interpretation layers in order to learn them self-supervised. We construct a multi-task hypergraph, in which nodes represent the layers, while the hyperedges (or edges) group them together and capture their relationships. Each hyperedge has one or multiple input nodes that are transformed, through a neural network, into a single output one. Thus, the set of hyperedges forms multiple pathways through the hypergraph that could reach a given node. This gives the possibility to form ensembles from which robust pseudolabels can be extracted for the output node when supervised data is not available. Such pseudolabels, along with the available

ground truth, are then used as a supervisory signal to distill self-supervised single hyperedges for the next learning cycle. With each cycle, we add novel unlabelled data in order to follow a scenario which is often met in practice, when more data is easily available but annotations are not.

In our extensive experiments, we demonstrate the effectiveness of our contributions: we show that more complex hyperedges bring a strong boost over simple pairwise edges. We also show that by adding an unsupervised learning cycle on unlabeled data from new scenes, we improve generalization and performance on those scenes. Moreover, we demonstrate that the multi-task consensus, which improves during the unsupervised learning stages, also brings the desired temporal stability and consistency, even though no explicit temporal information is used in our framework and all processing is done at the level of single frames. Last but not least, we show that our self-supervised hypergraph learning can be used to improve in both accuracy and temporal consistency over a heavily trained transformer expert when such an expert is used to initialize our hyperedge (edge) nets. This result is even more surprising when we consider the fact that our neural nets are lightweight U-Nets with two orders of magnitude fewer parameters.

Our **main contributions** are:

- We introduce, to the best of our knowledge, the first multi-interpretation hypergraph model that considers higher-order relationships between multiple world views and learns to find consensual pseudolabels from multiple pathways in the hypergraph. In extensive experiments, we show that our model is superior to related multi-task self-supervised graph models, which consider only simple edges and do not learn the ensembles that form pseudolabels.

- We introduce Dronescapes, a large video dataset with annotations for semantic segmentation, odometry, and 3D information. This represents a very complex real-world test bed, with a wider variety of scenes, which distinguishes our work with respect to the self-supervised multi-task literature, which is limited to synthetic or simpler real-world contexts (e.g., indoors).

---
*Primary contact: Marius Leordeanu at leordeanu@gmail.com
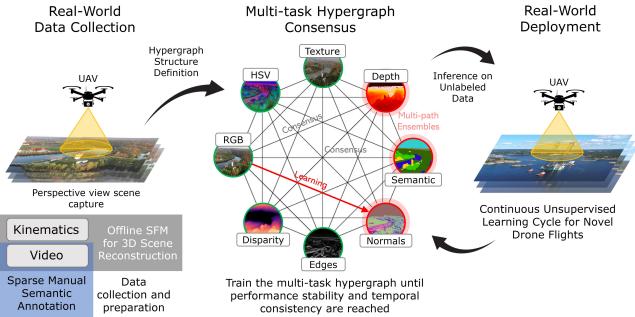
Figure 1: Our self-supervised multi-layer hypergraph in the context of real-world UAVs. We use kinematics and video drone data for offline 3D scene reconstruction and sparse manual labeling as an initial data collection and annotation for three dense prediction tasks: semantic segmentation, depth and surface normals estimation. We define the hypergraph structure in terms of input nodes (from sensors) and output nodes (those that will be predicted). We train in a self-supervised manner over multiple cycles, which improves both accuracy and temporal consistency. The hypergraph is also able to adapt to novel scenes and improve over initial powerful experts used for initialization.

- We demonstrate a solid generalization capability, by learning multiple tasks, with human supervision available only for the segmentation task for approximately 1% of all training data. Moreover, our hypergraph uses lightweight neural nets which can significantly improve both the accuracy and the temporal consistency over powerful, heavily pretrained experts, even though no temporal information is used.

## 2. Related work

**Consensus-based learning:** Recent works [27, 12] exploit clustering-based consensus from different transformations of the same image, but our goal is to consider meaningful scene representations (tasks), not just representation clusters. Also, focusing on a single task degrades the performance of others. It is difficult to achieve all-around improvements, as we aim here. In order to better balance the tasks, some approaches assign them weights [21]. Other works [31] take a step further towards holistic scene understanding by learning together, and self-supervised, depth and motion alongside semantics. Our model is more general and can accommodate in principle any number of tasks.

**Graph-based learning:** Work on hypergraph neural networks is scarce and uses a single label for a node [8, 32]. Iterative graph-based semantic segmentation is also related, but it considers only one or two representations [23, 42]. Other semi-supervised multi-task approaches do not consider higher-order relations between tasks and do not learn the ensembles used for generating pseudolabels [17, 14].

Other works do not have a semi-supervised learning component, but focus on robustness instead, with two-hop graphs [36] or larger multi-task ones [37]. Other related work [38] shows there is a strong relationship between multiple complementary tasks and that simultaneously exploiting the common and distinctive features between these tasks is effective for generalizing to out-of-distribution scenes.

**Unsupervised multi-task learning:** There are several important advances in unsupervised learning that are based on constraining together several specific tasks such as relative pose, depth and even semantic segmentation [6, 43, 26, 2, 10, 35, 31, 13, 4, 29]. There are also approaches that combine inputs from multiple senses for cross-modal prediction learning [16, 19, 39, 25, 15, 41].

**Knowledge Distillation:** The Teacher-Student paradigm is one of the most popular approaches in which smaller networks have been shown to be very effective in learning from large networks [22]. However, it is very rare that the smaller Student outperforms the Teacher [33]. A few methods perform knowledge distillation from multiple representations but without the graph structure [3]. Feeding pseudolabels for retraining the Student or Teacher is known as self-distillation [11]. Some methods perform distillation to yield compact models suitable for real-time inference [34]. Others attempt to use network architecture search (NAS) to design better models that achieve the same objective [5]. Nevertheless, the best-performing models are trained from scratch, either by developing a lighter version of top-performing architectures [40] or by optimizing older architectures suitable for parallel operations [9].

**Our work in context:** we put together and advance over many previous ideas, with demonstrated benefits. We do self-distillation, as we train single hyperedges at the next iteration by ensemble teachers from the previous one. We propose a hypergraph structure, as the only way to consider many tasks and capture their complex relationships, and also form supervisory ensembles for each, within a single system. We do self-supervised learning, by letting the graph teach itself, through such unsupervised ensembles, after an initial stage of learning either from experts or strictly supervised with limited labeled data.

## 3. Multi-task Self-supervised Hypergraphs

We present an overview of the proposed hypergraph model in Fig. 1. Hypergraph nodes represent different interpretation layers of the scene (e.g. RGB, semantic segmentation, depth, camera normals, etc.). Hyperedges capture relations between two or more nodes. Thus, in a k-th order hyperedge, a neural net takes k-1 node layers as input and learns to output the k-th node layer. In this manner we obtain, in the same hypergraph, multiple paths (by going through intermediate hyperedges) from the sensor input nodes (e.g. RGB images) to any output node. In principle,

any node could play simultaneously the role of an output or an input, in different hyperedges. Thus, for a given output node, we have several candidate layers, coming from different paths, which are used to form an ensemble. We use these ensembles, at the current learning cycle, to produce pseudolabels to distill (retrain) the edges and hyperedges in the next iteration. In experiments, we observed that when more unlabeled data is added, each self-supervised learning cycle improves not only accuracy but also temporal consistency, which, as discussed in Section 4, could be interpreted as a measure of trustworthiness.

### 3.1. Structure of Hyperedges

In our hypergraph, each hyperedge (often referred to as "edge" in the 2nd order case) has a set of one or multiple input nodes and a single output one. Input nodes could be either known (from sensors) or pseudolabels estimated by ensembles as explained in Sec. 3.2. Hyperedges are modeled by lightweight U-Nets [28], with the same structure of 1.1M parameters, but independently learned. Below we present the different types of hyperedges that we introduce (also see Fig. 2):

- **Edges (E)** - they learn a transformation between an input node to an output node through a neural net, similar to recent work on self-supervised multi-task graph models [17, 14, 37, 38]. We name each edge based on its input and output node layers (e.g., $rgb \rightarrow sseg$ refers to the edges that have the RGB image as input and the segmentation layer as output). Such direct edges from RGB input are the main ones we aim to improve during hypergraph learning. They are light and can be deployed at a small cost in practice (e.g., on UAVs).

- **Dual-hop Edges (DH-E)** - they use intermediate predicted representations. For example, the dual-hop edge $rgb \rightarrow depth \rightarrow sseg$ refers to the path that takes RGB as input, it produces the depth layer, and from that depth layer, it outputs the segmentation layer.

- **Aggregation Hyperedges (AH)** - they concatenate all input representations and learn the mapping from the aggregated volume of input nodes to each output node. We denote AH-ufo as the hyperedge that includes at input the prediction of UFODepth, an off-the-shelf purely unsupervised depth estimation method [20], to better understand the impact of such methods, concatenated with the RGB image.

- **Cycle Hyperedges (CH)** - they start from the concatenation of all input nodes together with the outputs of all AH hyperedges (including the AH output for the current node) and predict the output at the current node.
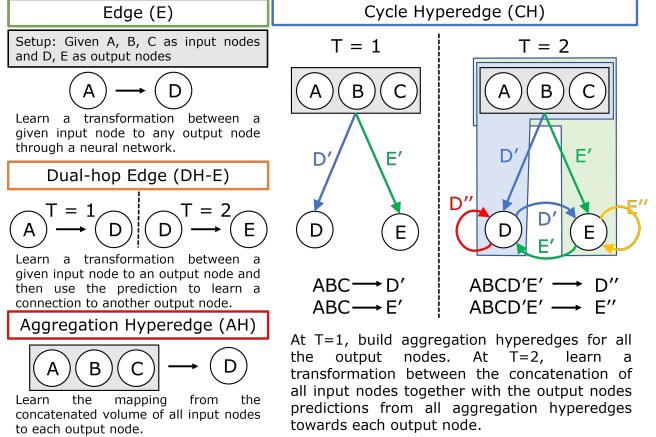


Figure 2: Types of edges and hyperedges in the hypergraph. Previous works use only edges, while we introduce two types of hyperedges to capture more complex relations between different layers.

### 3.2. Learning Hyperedge Ensembles

As data passes from all input nodes ($N_i$) to all output nodes ($N_o$) through the hypergraph $G(N_i, N_o)$ via the edge/hyperedge neural nets, various paths are created. At the end of the message passing process (*MP*), each output node has a list of messages, one for each such path. These messages can be seen as candidates in an ensemble learning process to compute a final response. Traditionally, ensembles are formed by simply averaging the candidates, in the regression case, or taking the majority, in the classification case. We study whether this ensemble mechanism can be improved by adding a secondary learning process, one at each output node, on top of the pathway-independent candidates. This can be seen as a special case of the *aggregate* function in Graph Neural Networks (GNNs): $\mathbf{Y}_{agg}(n_o) = f_{agg}(\mathbf{Y}_{n_o})$, where $\mathbf{Y}_{n_o} = MP(n_i \rightarrow ... \rightarrow n_o)$, for each pathway defined in the graph structure, with data $\mathbf{X}_{N_i}$ provided only at input nodes.

We introduce 2 types of parameterized ensemble models: *Linear Ensembles* and *Neural Network Ensembles*. Note that all individual edges are pretrained and frozen. The second optimization is done only on the results of each pathway, on the labeled training set, for which we have access to ground truth (denoted with **gt**). The tensor shape of each output node before aggregation is $\mathbf{Y}_{\mathbf{n_o}} :: (p_{n_o}, n, c_{n_o}, h, w)$, where $p_{n_o}$ represents the number of pathways, $n$ is the train set size and $(c_{n_o}, h, w)$ represents the shape of each output representation.

**Linear Ensembles:** We want to learn a vector of weights $\mathbf{w}$, with one $w_i$ per pathway $i$, using the train set, which is going to be used later on, to produce pseudolabels on the semi-supervised set. The weights are learned using linear regression. For the semantic segmentation node, we optimize using logistic regression. The aggregation function be-
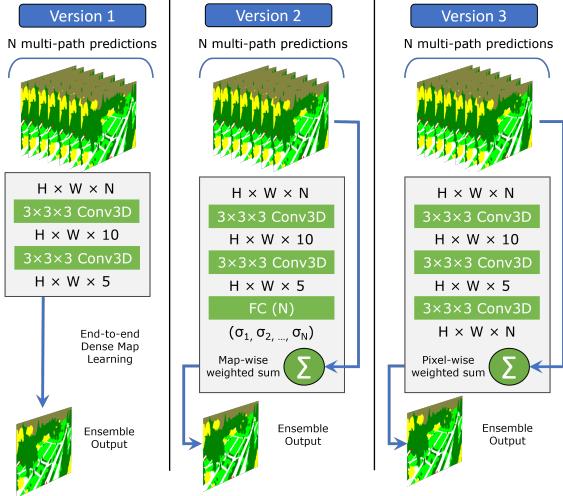
Figure 3: Proposed Neural Network Ensembles: **Version 1** produces directly a final output map. **Version 2** outputs one weight per each candidate input, then adds the weighted inputs. **Version 3** outputs one weight per each pixel for each candidate input, then adds the weighted inputs.

comes the weighted sum: $\mathbf{Y}_{agg} = \mathbf{w}^\top \mathbf{Y}_{\mathbf{n_o}}$. We repeat this process for each output node, resulting in a fixed weights matrix $\mathbf{W}$ which is then used for generating pseudolabels.

**Neural Network Ensembles:** Instead of learning a set of fixed weights, as before, in this case we train a separate neural net (*NN*) for each output node. Thus, each new input will dynamically produce, through this neural net, a new way of combining the set of candidate outputs $\mathbf{Y}$ (defined as before). The loss function is $\min |(NN(\mathbf{Y}) - \mathbf{gt})|$ and the *NN* model is optimized using iterative gradient descent. We propose three types of neural net ensembles (as also shown in Fig. 3): **Version 1).** *NN* produces directly a final output map: $Y_{agg} = NN(\mathbf{Y})$. **Version 2).** Dynamic weights, one for each candidate layer, aggregated by weighted mean: $\mathbf{y_{NN}} :: (p,) = NN(\mathbf{Y}); Y_{agg} = \mathbf{y_{NN}}^T \mathbf{Y}$. **Version 3).** Dynamic weights, a separate one for each pixel in each candidate layer, followed by point-wise multiplication: $\mathbf{Y_{NN}} :: (p, c, h, w) = NN(\mathbf{Y}); Y_{agg} = \mathbf{Y_{NN}} \odot \mathbf{Y}$.

### 3.3. Self-supervised iterative hypergraph learning

Learning in the hypergraph requires an initial set of annotations, which is either given by humans, automatically generated (by an offline analytical method) or provided by a pretrained expert. These annotations are used to initialize the individual edges and hyperedges and then learn the ensemble functions. Once the initialization stage is completed, we can proceed with the semi-supervised learning stages (iterations), in which we first produce pseudolabels on newly added data, then retrain the individual edges and hyperedges on the new pseudolabels (including all the other labels and pseudolabels available from previous iterations).

Succinctly, each self-supervised learning iteration consists of 1) adding new unlabeled data; 2) producing pseudolabels for the new data; 3) retraining the hyperedges by including the new pseudolabeled data in the training set.

### 3.4. Dronescapes Dataset

We introduce a large-scale UAV video dataset with automatic odometry and 3D information for all frames and semi-automatic semantic segmentation annotation for a subset of frames, as explained later in this Section. All video sequences include GPS information, linear and angular velocities, and absolute camera angles. The total length is about 50 minutes. Videos have $3840 \times 2160$ 30 FPS images, while the odometry is provided at 10 Hz. We collect a total of 10 widely varied scenes that we split into 7 training and 3 test scenes. A visual representation for each scene from our dataset is shown in Figure 4. We highlight the variety in landscapes, altitudes, and object scales between each of the rural (Atanasie, Gradistei, Petrova, Barsana, Comana), urban (Olanesti, Herculane, Slanic) and seaside scenes (Jupiter, Norway). For the test scenes we chose three different kinds: one (Barsana) with a more similar semantic class distribution to at least one of the training scenes, one (Comana) that is less similar to the training scenes and a third one (Norway), that is very different from any of the training scenes.

**Dataset split.** Exclusively for the task of semantic segmentation, we sparsely manually annotate frames. Based on their number we divide the scenes in *weakly-labeled scenes* (Gradistei, Herculane, Jupiter, Petrova, Olanesti, Barsana and Comana) and *strongly-labeled scenes* (Atanasie, Slanic and Norway). For the strongly labeled scenes, we sample frames every 2 seconds covering the whole video. For the weakly-labeled scenes, we uniformly sample triplets of frames from each scene, such that the frames in each triplet are 2 seconds apart (or 60 frames between the triplet limits), while the triplets are at least 26 seconds apart and have significant changes in pose (viewpoint). We divided our dataset into 4 sets, in a suitable manner for multiple training iterations with the addition of novel data every iteration. In Tab. 1 we present these splits and the total number of frames per scene, alongside the number of manually labeled frames. Half of the triplets/frames from the training scenes are included in *Train Unlabeled (iter 1)*, whilst the other half is in *Train Unlabeled (iter 2)*. The frames from all the test scenes are included in *Train Unlabeled (iter 3)*. We note that the only manually-labeled frames used in learning the hypergraph are the frames from *Train Labeled* set, whilst the rest are used for evaluation purposes only.

**Semantic segmentation annotation.** Every pixel in a frame is labeled with one of the 8 classes - *land*, *forest*, *residential*, *road*, *little-objects*, *water*, *sky* and *hill*. The annotations process was difficult, especially due to the fact that

Figure 4: Sample frames from each of the 10 scenes from the Dronescapes dataset. The scenes framed with **green** borders represent training scenes for which we have access to a small fraction of manual annotations during training. The others depict unseen, test scenes with semantic distributions that are closer to the training set (in **blue**) or out-of-distribution (**red**). There is a large variation in spatial distributions of classes among the different Dronescapes scenes, which range from rural (Atanasie, Gradistei, Petrova, Barsana, Comana), to urban (Olanesti, Herculane, Slanic) and seaside (Jupiter, Norway), while also being geographically far apart.

Table 1 Dronescapes dataset split. For each training scene we report the number of frames used for purely supervised training (Train Labeled set), and also for iterative self-supervised training (Train Unlabeled iterations 1, 2 and 3 sets). In parenthesis we show the number of frames for which we have manual segmentation annotations. Except for the Train Labeled set on the training scenes, all the other annotations are used strictly for evaluation.

| Scene Name | Train Labeled | Train Unlabeled (iter 1) | Train Unlabeled (iter 2) | Train Unlabeled (iter 3) |
|---|---|---|---|---|
| Atanasie | 76 | 4501 (76) | 4500 (75) | - |
| Gradistei | 18 | 726 (18) | 484 (12) | - |
| Herculane | 12 | 484 (12) | 363 (9) | - |
| Jupiter | 21 | 847 (21) | 605 (15) | - |
| Olanesti | 18 | 726 (18) | 484 (12) | - |
| Petrova | 12 | 484 (12) | 363 (9) | - |
| Slanic | 76 | 4501 (76) | 4500 (75) | - |
| Barsana | - | - | - | 1452 (36) |
| Comana | - | - | - | 1210 (30) |
| Norway | - | - | - | 2941 (50) |
| TOTAL | 233 | 12269 (233) | 11299 (207) | 5603 (116) |

some objects are too small to be seen clearly, while other larger regions can fall into multiple categories (e.g., an area can be labeled as hill, land, and forest at the same time).

**Depth annotation.** The drone trajectory computed with structure from motion (SfM) [1] is aligned automatically with the trajectory from GPS, by a similarity transformation (translation, rotation and scale), which is then applied to the SfM 3D model. By combining it with the known 6D pose (from GPS and odometry) and camera intrinsic parameters, we obtain accurate metric depth maps (less than $2\%$ error in our extensive offline tests). The generated metric depth maps are used for training only during Iteration 1, otherwise used for evaluation purposes.

**Surface normals annotation.** We automatically processed the SfM 3D meshes in Blender and obtained surface normals at every pixel w.r.t world coordinates, which, multiplied with the inverse of the camera rotation matrix give normals w.r.t camera (aka. "camera normals").

**Hypergraph learning on Dronescapes:** We focus on scenarios where exact ground truth or human annotations are very scarce or simply not available, which is a very common case in practice. Such an example is the case of learning from UAV videos, which is representative for our scenario and also extremely difficult due to the wide variety and complexity of scenes and camera viewpoints. As explained in the theoretical section, in each self-supervised learning cycle we add new unlabeled data. We first produce pseudolabels for each output task by using ensembles of edges and hyperedges and then retrain (distill) the next generation of edges and hyperedges by including the pseudolabeled data during training. For clarity, we detail further how we use the current dataset split in our iterative learning procedure: **Iteration 1)** Using the Train Labeled dataset, we employ a semi-automatic label propagation method [23] to annotate intermediate frames from adjacent manually labeled ones and obtain Train Unlabeled (iter 1). For depth and normals, we use the automatically generated labels as described before. For **Iteration 2)** we used the fully-trained hypergraph from Iteration 1 and generate pseudolabels for frames in Train Unlabeled (iter 2) set, using the hypergraph ensembles for all the predicted tasks. We join the two sets Train Unlabeled (iter 1 + 2) and retrain the hypergraph. For **Iteration 3)** we repeat the steps from Iteration 2 and generate pseudolabels on the Train Unlabeled (iter 3) set, to

expand the set to Train Unlabeled (iter 1 + 2 + 3).

## 4. Experimental Analysis

We focus on learning three complementary tasks (output nodes): semantic segmentation, depth estimation and surface normals prediction. First, we study the impact made by each of our contributions w.r.t previous self-supervised graph multi-task methods, such as the addition hyperedges vs. simple edges (Sec. 4.1) and learning parameterized multi-path ensemble models (linear and deep neural nets) vs. non-parametric ones (Sec. 4.2). We also test the ability of the hypergraph to improve self-supervised over an initial state-of-the-art expert for semantic segmentation, which is used for distillation at different stages (Sec. 4.4).

**Performance metrics.** We report mean IoU (% - higher is better ($\uparrow$)) for semantic segmentation and L1 error * 100 (lower is better ($\downarrow$)) for depth and normals estimation.

**Temporal consistency metrics.** Even though we process single frames without any temporal information, we observed that the hypergraph self-supervised learning process significantly improves the prediction consistency between nearby frames (for pixels that belong to the same physical point), which is more than just improving average accuracy per frame. We designed a special *consistency metric*, which uses optical flow [30] to establish temporal chains that connect corresponding pixels across frames. For segmentation, the consistency for a given pixel measures the percentage of votes received by the winning class along the 5-frames temporal chain centered at that pixel. For depth and camera normals, we measure consistency using the variance $var$ of predictions along the same chain, as $e^{-var}$ (to map it between 0 and 1 and increase with quality). Note that if the optical flow works well, there is no occlusion (which is often the case) and the predictions are correct, then the consistencies for all three tasks should be 1. We found this metric to be highly valuable, as it is complementary to the average accuracy: a predictor could have good accuracy on average per frame (low bias), but lack sufficient temporal consistency (high variance). Temporally inconsistent, highly fluctuating predictions, generally indicate higher levels of uncertainty and lower reliability.

**Input representations.** We consider, besides the main RGB input node, other input nodes that are mathematically derived from RGB, such as HSV color, soft edges [18] and soft segmentation [18]), which are effective by expanding the number of edges and indirectly improving the power of the ensembles that they form at the output nodes. We also tested the possibility of adding to the pool of input nodes an unsupervised metric depth map, such as UFODepth [20], but found it did not bring a significant boost in performance. We show examples of each of the input layer types in Fig. 1.

Table 2 Evaluation of edges and hyperedges for multiple tasks: 1 - semantic segmentation (sseg); 2 - depth estimation (depth); 3 - surface normals (norm). We report mean IoU (% - higher values are better ($\uparrow$) for the task of semantic segmentation and L1 error * 100 (lower is better ($\downarrow$)) for depth and normals estimation. The layers in the Type column, denote the input node (for edge type E) and the intermediate node (for edge type DH-E, for which RGB is always the input node). We evaluate exclusively on the manually annotated frames and report mean performance over all scenes from the Train Unlabeled (iter 2) set and also on Barsana and Comana test scenes from Train Unlabeled (iter 3) (see 1 for more details). Bolded results highlight the mean performance gain of training hyperedges over edges.

|  | Type | Train Unlabeled (iter 2) | | | Train Unlabeled (iter 3) | | |
|---|---|---|---|---|---|---|---|
|  |  | (1) | (2) | (3) | (1) | (2) | (3) |
| Edges | E: rgb | 42.85 | 5.04 | 10.37 | 32.79 | 21.66 | 12.40 |
|  | E: hsv | 41.70 | 4.69 | 10.54 | 33.51 | 19.90 | 12.48 |
|  | E: softedges | 32.47 | 6.26 | 11.56 | 27.28 | 18.61 | 13.53 |
|  | E: softseg | 30.71 | 5.97 | 11.14 | 24.68 | 22.70 | 12.76 |
|  | E: ufo | 20.77 | 7.19 | 11.69 | 16.93 | 17.55 | 12.89 |
|  | DH-E: sseg | - | 6.25 | 11.39 | - | 19.00 | 12.93 |
|  | DH-E: depth | 29.24 | - | 12.22 | 24.11 | - | 13.79 |
|  | DH-E: norm | 30.56 | 6.17 | - | 26.35 | 21.15 | - |
|  | mean | 32.61 | 5.94 | 11.27 | 26.52 | **20.08** | 12.97 |
| Hyperedges | AH | 41.80 | 5.33 | 10.37 | 33.63 | 23.96 | 12.24 |
|  | AH-ufo | 41.96 | 5.16 | 10.78 | 33.82 | 21.10 | 12.72 |
|  | CH | 44.63 | 4.93 | 10.32 | 36.92 | 20.36 | 12.23 |
|  | mean | **42.80** | **5.14** | **10.49** | **34.79** | 21.81 | **12.40** |

### 4.1. Impact of Hyperedge Complexity

Related work tackled multi-task learning through multi-path consensus [14, 17] but did not exploit higher-order relations between tasks. We test the performance of each individual hyperedge type on all three tasks (in Tab. 2) and show that the higher-order ones are on average significantly stronger than the pairwise edges. With one exception: the case of metric depth, which is much more scene dependent than the other tasks and for which the more complex hyperedges overfit more easily. However, for the other two tasks, hyperedges bring a significant advantage over the simpler edges, while the dual-hop edges (DH-E), heavily used in [17], have poor performance (since errors accumulate along the two-hops path).

Table 3 Comparison to previous multi-task graph-based methods. We show considerable improvements by adding the proposed hyperedges (denoted with HE in the table) on top of existing work that uses only edges within their graph structure. We further report performance improvements by learning the proposed types of ensembles on top of both edges and hyperedges predictions (denoted by Ours). For this experiment, we considered solely the task of semantic segmentation. We report mean IoU (% - higher values are better and bolded). The evaluation was done on each scene from the testing set and overall. LR stands for Logistic Regression (see Section 3.2 for details).

| Method | IoU($\uparrow$) | | | |
| --- | --- | --- | --- | --- |
| | Barsana | Comana | Norway | Mean |
| NGC [17] (Mean) | 41.53 | 40.75 | 27.38 | 36.55 |
| NGC (Mean) + HE | 42.61 | 42.17 | 27.96 | 37.58 |
| NGC [17] (Median) | 39.25 | 37.41 | 27.01 | 34.56 |
| NGC (Median) + HE | 44.34 | 38.99 | 22.63 | 35.32 |
| CShift [14] (Mean) | 43.91 | 42.13 | 29.68 | 38.57 |
| CShift (Mean) + HE | 44.71 | 43.88 | 30.09 | 39.56 |
| CShift [14] (Median) | 43.30 | 40.62 | 29.51 | 37.81 |
| CShift (Median) + HE | 46.27 | 43.67 | 29.09 | 39.68 |
| LR (Ours) | 46.51 | **45.59** | **30.17** | **40.76** |
| NN (v1) (Ours) | 45.53 | 42.92 | 28.37 | 38.94 |
| NN (v2) (Ours) | 45.48 | 43.25 | 26.36 | 38.36 |
| NN (v3) (Ours) | **48.21** | 44.85 | 28.94 | 40.67 |

## 4.2. Impact of Different Ensemble Models

We train all edges and hyperedges on the Train Unlabeled (iter 1) set and test different ensemble models for the task of semantic segmentation. Being the only task for which we have manually annotated ground truth, we focused on it for validating different ensembles. Our results are showcased in Table 3.

Both NGC [17] and CShift [14] models use only edges and relatively simple non-parametric ensemble models at nodes (NGC - simple average and CShift - non-parametric pixel-wise kernel weighted average). We bring a performance boost by adding hyperedges and also by allowing the ensembles to learn. As discussed in Sec. 3.2, we propose one linear and 3 types of non-linear (neural nets) ensembles (Fig 3). Our experiments show that learning parametric ensemble models, even a simple linear one, improves significantly (above 2% on average) over previously published work. Performance is reported on the Train Unlabeled (iter 3), which includes only the test scenes.

Table 4 Iterative learning performance on the single task links. The evaluation was done on the test scenes. The reported performance is averaged.

| Type | Semantic | | Depth | | Normals | |
| --- | --- | --- | --- | --- | --- | --- |
| | IoU ($\uparrow$) | Cons. ($\uparrow$) | L1 ($\downarrow$) | Cons. ($\uparrow$) | L1 ($\downarrow$) | Cons. ($\uparrow$) |
| rgb-sup. | 25.04 | 88.85 | - | - | - | - |
| rgb-iter1 | 32.79 | 94.04 | 21.66 | 5.89 | 12.40 | 98.32 |
| rgb-iter2 | 37.26 | 95.72 | 17.34 | 7.06 | 11.93 | 98.87 |
| rgb-iter3 | **40.31** | **98.13** | **16.64** | **30.26** | **11.71** | **99.30** |

## 4.3. Impact of Iterative Self-supervised Learning

To further highlight the benefits of learning through multiple iterations in a self-supervised manner, we test by progressively adding novel unlabeled data (learning iterations that are explained in Sec. 3.4 and Sec. 3.3). After training each link in the hypergraph (Tab. 2) for each of the output tasks, we form multi-path consensus ensembles (Tab. 3), to produce pseudolabels for the second, self-supervised learning iteration, when we retrain all the edges and hyperedges on an expanded set of labels, which includes the supervised labels from Train Labeled, and the automatically generated labels from Train Unlabeled (iter 1) and Train Unlabeled (iter 2). We use again the newly retrained edges and hyperedges to form ensembles and pseudolabels for the additional Train Unlabeled (iter 3) set and continue retraining the single edges ($rgb \rightarrow task$) for a third self-supervised learning iteration. The results show considerable and consistent improvement at the level of the distilled edge $rgb \rightarrow task$ in both accuracy and temporal consistency, on all 3 tasks (Tab. 4). We denote by $rgb-sup.$ the fully-supervised edge trained exclusively on the Train Labeled set.

## 4.4. Adapting to Novel Scenes

We want to test the ability to improve over an initial state-of-the-art Expert Teacher and also adapt to novel test scenes. We focus on semantic segmentation and use the recent Mask2Former [7] as the SoTA Expert, pretrained on Mapillary Vistas [24]. Being a similar scenario, we easily map the Mask2Former set of classes to ours.

We consider three phases of learning in this setup, depending on the moment we use our self-supervised hypergraph learning. **Phase 1** denotes training the single $rgb \rightarrow seg$ edge, as presented so far, in two iterations of learning (before seeing any data from the test scenes). **Phase 2** refers to distilling $rgb \rightarrow seg$ edge (fine-tuning after Phase 1 or from scratch when Phase 1 is missing) on the output of Mask2Former on the test scenes. **Phase 3** refers to training $rgb \rightarrow seg$ during one iteration of self-supervised hypergraph learning, only on unlabeled data from test scenes, after Phase 2. In essence, Phase 1 and Phase 3 represent

Table 5 Distilling a VisTransformer-based Expert's knowledge into the hypergraph. Experiments were done on novel scenes (test) for the task of semantic segmentation. Experiments show the advantages of using our hypergraph procedures in multiple learning phases and using an off-the-shelf method for the task of semantic segmentation. Performance evaluation was done at the level of the distilled direct edges. Bolded numbers are best.

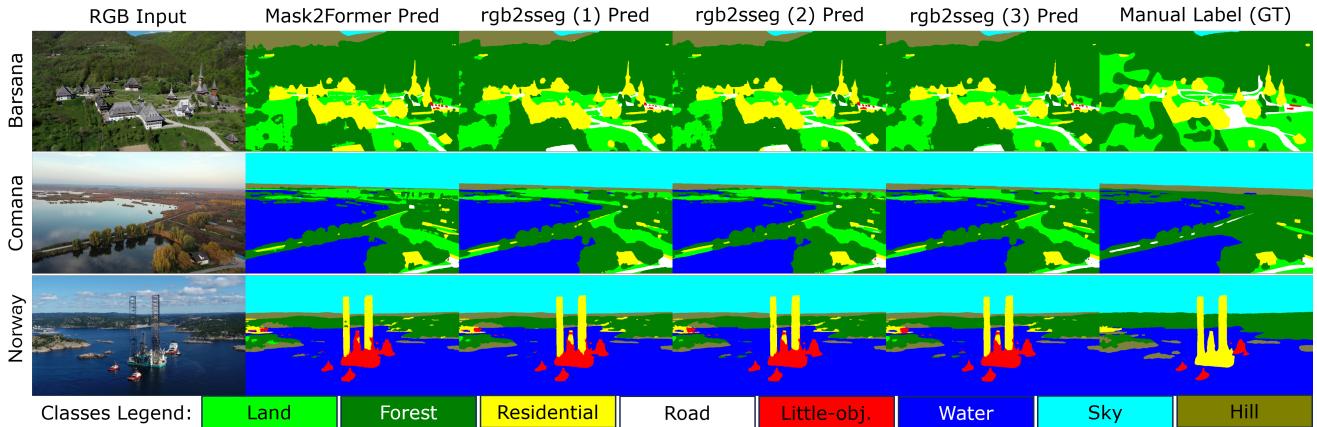| Method | Phase 1 | Phase 2 | Phase 3 | Barsana | | Comana | | Norway | | Mean | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | IoU ($\uparrow$) | Cons. ($\uparrow$) | IoU ($\uparrow$) | Cons. ($\uparrow$) | IoU ($\uparrow$) | Cons. ($\uparrow$) | IoU ($\uparrow$) | Cons. ($\uparrow$) |
| Mask2Former [7] | - | - | - | 56.77 | 95.48 | 59.84 | 97.26 | 41.71 | 98.23 | 52.77 | 96.99 |
| $rgb \rightarrow sseg$ (1) (Ours) | ✗ | ✓ | ✓ | 57.86 | 98.46 | 57.82 | 98.88 | 42.95 | 99.51 | 52.88 | 98.95 |
| $rgb \rightarrow sseg$ (2) (Ours) | ✓ | ✓ | ✗ | 58.02 | 98.22 | 60.14 | 98.70 | 42.55 | 99.42 | 53.57 | 98.78 |
| $rgb \rightarrow sseg$ (3) (Ours) | ✓ | ✓ | ✓ | **59.16** | **98.58** | **60.49** | **99.06** | **43.14** | **99.55** | **54.26** | **99.06** |



Figure 5: Qualitative results on images from the testing set using Mask2Former labels as starting point for distillation. Even on a very dissimilar test scene w.r.t all other scenes (Norway), we obtain improved results over the baseline, and overall our method yields favorable numbers on all scenes, for both accuracy and temporal consistency, as shown in Tab. 5

the same type of hypergraph learning with different starting points (initialization) for $rgb \rightarrow seg$, separated by Phase 2 of fine-tuning on Mask2Former output. Our goal is to evaluate the impact of the hypergraph model when applied at different stages, before and after Expert distillation. The results, presented in Tab. 5 and Fig. 5, demonstrate the added value of the hypergraph in each case, with maximum effect when applied in both Phases 1 and 3. Interestingly, experiment 2 (hypergraph pretraining) is more effective than experiment 1 (hypergraph post-training), probably due to the additional data from the other scenes available in experiment 2. Also note that the added benefit of the hypergraph is significant at the level of temporal consistency (more results in the appendix), which suggests that the self-supervised consensus among multiple tasks strongly reduces classifier variance, potentially increasing reliability and trustworthiness. Moreover, the improvements are obtained by an edge that is two orders of magnitude smaller, in terms of the number of parameters, than the SoTA Mask2Former.

## 5. Conclusions

We introduced a novel multi-task self-supervised hypergraph model for learning in the case of very limited training data. Our theoretical contributions include the addition of hyperedges and parameterized ensemble learning, with proven experimental benefits. We also introduce Dronescapes, a large-scale video dataset for UAVs, which brings our experiments into the real world, different from the synthetic datasets used by prior works. With almost all annotations being automatically generated (except for a very small set of manually annotated frames for semantic segmentation) we show that by using hyperedges and learning ensembles of such hyperedges, we improve both accuracy and temporal consistency even though no temporal information is given. Our model is also effective when it uses as initial annotations the output of a state-of-the-art expert, as demonstrated by experiments on three different novel scenes, with no ground truth available for training.

# References

[1] AliceVision. Meshroom: A 3D reconstruction software., 2018. 5

[2] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In *Advances in Neural Information Processing Systems*, pages 35–45, 2019. 2

[3] Hong Cai, Janarbek Matai, Shubhankar Borse, Yizhe Zhang, Amin Ansari, and Fatih Porikli. X-distill: Improving self-supervised monocular depth via cross-task distillation. *arXiv preprint arXiv:2110.12516*, 2021. 2

[4] Po-Yi Chen, Alexander H Liu, Yen-Cheng Liu, and Yu-Chiang Frank Wang. Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2624–2632, 2019. 2

[5] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. *arXiv preprint arXiv:1912.10917*, 2019. 2

[6] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7063–7072, 2019. 2

[7] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1290–1299, 2022. 7, 8

[8] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3558–3565, 2019. 2

[9] Guangwei Gao, Guoan Xu, Juncheng Li, Yi Yu, Huimin Lu, and Jian Yang. Fbsnet: A fast bilateral symmetrical network for real-time semantic segmentation. *IEEE Transactions on Multimedia*, 2022. 2

[10] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8977–8986, 2019. 2

[11] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021. 2

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020. 2

[13] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. Semantically-guided representation learning for self-supervised monocular depth. *arXiv preprint arXiv:2002.12319*, 2020. 2

[14] Emanuela Haller, Elena Burceanu, and Marius Leordeanu. Self-supervised learning in multi-task graphs through iterative consensus shift. *BMVC*, 2021. 2, 3, 6, 7

[15] Li He, Xing Xu, Huimin Lu, Yang Yang, Fumin Shen, and Heng Tao Shen. Unsupervised cross-modal retrieval through adversarial learning. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1153–1158. IEEE, 2017. 2

[16] Di Hu, Feiping Nie, and Xuelong Li. Deep multimodal clustering for unsupervised audiovisual learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[17] Marius Leordeanu, Mihai Cristian Pîrvu, Dragos Costea, Alina E Marcu, Emil Slusanschi, and Rahul Sukthankar. Semi-supervised learning for multi-task scene understanding by neural graph consensus. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1882–1892, 2021. 2, 3, 6, 7

[18] Marius Leordeanu, Rahul Sukthankar, and Cristian Sminchisescu. Generalized boundaries from multiple image interpretations. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1312–1324, 2014. 6

[19] Yunzhu Li, Jun-Yan Zhu, Russ Tedrake, and Antonio Torralba. Connecting touch and vision via cross-modal prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[20] Vlad Licăret, Victor Robu, Alina Marcu, Dragoş Costea, Emil Sluşanschi, Rahul Sukthankar, and Marius Leordeanu. Ufo depth: Unsupervised learning with flow-based odometry optimization for metric depth estimation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6526–6532. IEEE, 2022. 3, 6

[21] Shikun Liu, Stephen James, Andrew Davison, and Edward Johns. Auto-lambda: Disentangling dynamic task relationships. *Transactions on Machine Learning Research*, 2022. 2

[22] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2604–2613, 2019. 2

[23] Alina Marcu, Vlad Licaret, Dragos Costea, and Marius Leordeanu. Semantics through time: Semi-supervised segmentation of aerial videos with iterative label propagation. In *Asian Conference on Computer Vision (ACCV)*, pages 2881–2890, 2020. 2, 5

[24] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999, 2017. 7

[25] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 653–658. ACM, 2004. 2

[26] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12240–12249, 2019. 2

[27] Jayanth Reddy Regatti, Aniket Anand Deshmukh, Eren Manavoglu, and Urun Dogan. Consensus clustering with unsupervised representation learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2021. 2

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[29] Sinisa Stekovic, Friedrich Fraundorfer, and Vincent Lepetit. Casting geometric constraints in semantic segmentation as semi-supervised learning. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1854–1863, 2020. 2

[30] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, pages 402–419. Springer, 2020. 6

[31] Fabio Tosi, Filippo Aleotti, Pierluigi Zama Ramirez, Matteo Poggi, Samuele Salti, Luigi Di Stefano, and Stefano Mattoccia. Distilled semantics for comprehensive scene understanding from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4654–4665, 2020. 2

[32] Francesco Tudisco, Konstantin Prokopchik, and Austin R Benson. A nonlinear diffusion method for semi-supervised learning on hypergraphs. *arXiv preprint arXiv:2103.14867*, 2021. 2

[33] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[34] Jipeng Wu, Rongrong Ji, Jianzhuang Liu, Mingliang Xu, Jiawen Zheng, Ling Shao, and Qi Tian. Real-time semantic segmentation via sequential knowledge distillation. *Neurocomputing*, 439:134–145, 2021. 2

[35] Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised learning of geometry from videos with edge-aware depth-normal consistency. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[36] Teresa Yeo, Oğuzhan Fatih Kar, and Amir Zamir. Robustness via cross-domain ensembles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12189–12199, October 2021. 2

[37] Amir R Zamir, Alexander Sax, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas J Guibas. Robust learning through cross-task consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11197–11206, 2020. 2, 3

[38] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 2, 3

[39] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1058–1067, 2017. 2

[40] Wenqiang Zhang, Zilong Huang, Guozhong Luo, Tao Chen, Xinggang Wang, Wenyu Liu, Gang Yu, and Chunhua. Shen. Topformer: Token pyramid transformer for mobile semantic segmentation. *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[41] Hang Zhao, Chuang Gan, Andrew Rouditchenko, Carl Vondrick, Josh McDermott, and Antonio Torralba. The sound of pixels. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 570–586, 2018. 2

[42] Mingmin Zhen, Jinglu Wang, Lei Zhou, Shiwei Li, Tianwei Shen, Jiaxiang Shang, Tian Fang, and Long Quan. Joint semantic segmentation and boundary detection using iterative pyramid contexts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13666–13675, 2020. 2

[43] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017. 2