International Journal of Computer Vision (2023) 131:717–731
https://doi.org/10.1007/s11263-022-01719-0

Video Region Annotation with Sparse Bounding Boxes

Yuzheng Xu[1] · Yang Wu[1] · Nur Sabrina binti Zuraimi[1] · Shohei Nobuhara[1] · Ko Nishino[1]

Received: 3 September 2021 / Accepted: 9 November 2022 / Published online: 14 December 2022

Abstract
Video analysis has been moving towards more detailed interpretation (e.g., segmentation) with encouraging progress. These tasks, however, increasingly rely on densely annotated training data both in space and time. Since such annotation is labor-intensive, few densely annotated video data with detailed region boundaries exist. This work aims to resolve this dilemma by learning to automatically generate region boundaries for all frames of a video from sparsely annotated bounding boxes of target regions. We achieve this with a Volumetric Graph Convolutional Network (VGCN), which learns to iteratively find keypoints on the region boundaries using the spatio-temporal volume of surrounding appearance and motion. We show that the global optimization of VGCN leads to more accurate annotation that generalizes better. Experimental results using three latest datasets (two real and one synthetic), including ablation studies, demonstrate the effectiveness and superiority of our method.

Keywords Video annotation · Semi-automatic annotation · Graph convolutional network · Automatic boundary finding

1 Introduction

Advances in deep learning techniques have brought about remarkable progress in many computer vision tasks such as detection, segmentation, tracking, and recognition. One major caveat with most deep learning algorithms is that they need to be trained with a huge amount of data that have been carefully labeled with ground truth (Everingham et al., 2015; Lin et al., 2014; Zhou et al., 2019). Furthermore, in many applications such as autonomous driving, visual analysis has to be done on every captured frame for real-time processing or for tasks that require dense spatio-temporal information. For these, dense per-frame region-level annotation becomes essential for training the models.

Manually annotating detailed region boundaries for every video frame is a highly time-consuming, tedious, if not impossible, task. To the best of our knowledge, no publicly available dataset offers per-frame annotation. The lack of densely annotated video data has limited the research on detailed region-level video analysis and has forced researchers to explore image-based models instead. Frame-wise processing, however, misses the spatial-temporal relationships and can lead to inferior results. As such, dense per-frame region annotation with an affordable and efficient means becomes critical. Bounding box is a widely used and rather cheap supervision. What if we only need annotators to provide region bounding boxes for sparsely chosen keyframes and then the annotation tool automatically generates boundaries for the region of interest in every frame, as illustrated in Fig. 1? We introduce a novel dense video annotation method that only requires sparse bounding-box supervision. We fit an iteratively deformed volumetric graph to the video sub-sequence bounded by two chosen keyframes, so that its uniformly initialized graph nodes gradually move to the key points on the sequence of region boundaries. The model consists of a set of deep neural networks, including normal convolutional networks for frame-wise feature map extraction and a volumetric graph convolutional net-

Communicated by Moi Hoon Yap.

✉ Yuzheng Xu
yxu@vision.ist.i.kyoto-u.ac.jp

Yang Wu
wu.yang.8c@kyoto-u.ac.jp

Nur Sabrina binti Zuraimi
sabrina@vision.ist.i.kyoto-u.ac.jp

Shohei Nobuhara
nob@i.kyoto-u.ac.jp

Ko Nishino
kon@i.kyoto-u.ac.jp

[1] Kyoto University, Kyoto, Japan

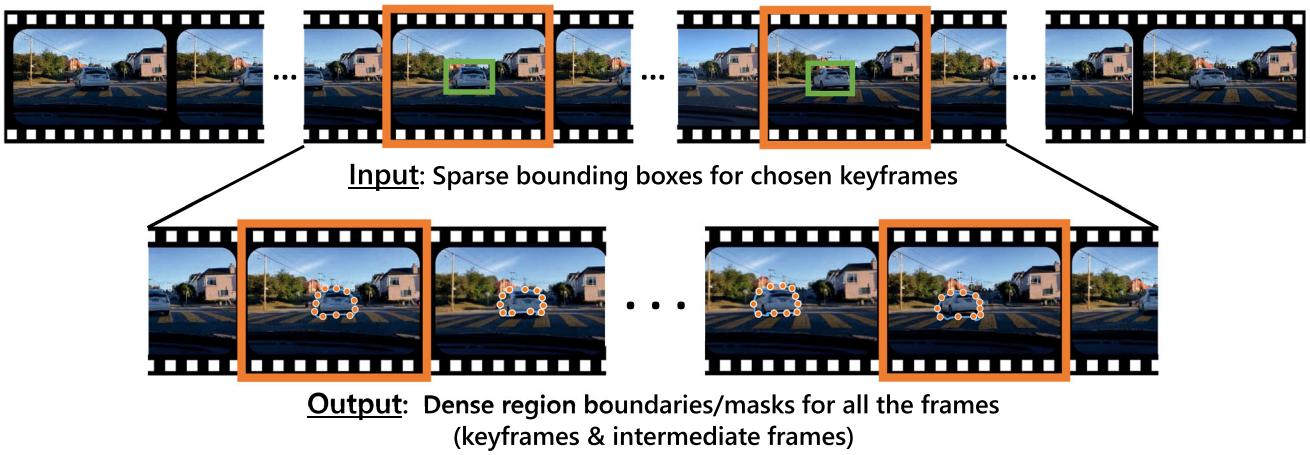

Fig. 1 Our goal is to derive a video region annotation tool that can automatically annotate dense per-frame region boundaries from sparse user-provided bounding boxes given for sparse keyframes

work for iterative boundary point finding. By propagating and integrating node-associated information (sampled from feature maps) over graph edges, a content-agnostic prediction model is learned for estimating graph node location shifts. The effectiveness and superiority of the proposed model and its major components are demonstrated on three latest public datasets: a large synthetic dataset Synthia and two real datasets named KITTI-MOTS and BDD-MOTS capturing natural driving scenes.

This paper extends our preliminary work (Xu et al., 2020) by introducing additional motion features and additional experimental validation that further demonstrates the effectiveness of our method. Brief descriptions of our model implementation are also provided in this version. To summarize, we extend our preliminary work (Xu et al., 2020)

- By introducing temporal difference map as a new motion feature which is more stable than optical flow in real images for contour finding;
- By designing strict rules to generate our training sequences from synthetic and real video data which leads to sequences closer to real annotation scenarios and improves the effectiveness of our learned model;
- And by showing additional ablation studies on the effectiveness of the encoder network and the graph convolutional network which demonstrate different generalization abilities of the two modules and show the fundamental differences between different datasets.

2 Related Work

Region Annotation versus Segmentation Since one can easily get confused by the relationship between our work and large amounts of existing works on segmentation tasks (including semantic segmentation (Li et al., 2019; Ding et al., 2020; Paul et al., 2020; Nabavi et al., 2018), object segmentation (Jain et al., 2017), and instance segmentation (Yang et al., 2019a), we first clarify the difference. Annotation is the process of labeling data to be used for machine learning algorithms, including training and evaluation. Although there have been many studies in learning from unlabeled data (Cho et al., 2015; Wang & Gupta, 2015; Lee et al., 2017), many state-of-the-art algorithms still need some sort of labeled data for training, and in any case quantitative performance evaluation generally requires ground-truth labels. On the other hand, segmentation is the process of predicting pixel-level class labels. The main difference between annotation and segmentation is that *annotation is for building a dataset and, in contrast, segmentation is a vision task model trained on an annotated dataset. As such, any segmentation method would benefit from better annotated data and hence the annotation tool*. This work focuses on region annotation for videos, aiming to alleviate the burden of the annotator to help make the process of creating ground truth data easier, and thus support the development of new video analysis models, including those for image/video based segmentation.

Single-Image Annotation Tools In general, one can still choose to annotate each frame using image annotation tools. Representative works are briefly discussed here. One of the earliest annotation tool that aimed to cut down the time required to annotate was GrabCut (Rother et al., 2004) which does interactive foreground/background segmentation in still images using bounding boxes and foreground and background marking strokes as its inputs. Polygon-RNN (Castrejon et al., 2017) and Polygon-RNN++ (Acuna et al., 2018) use a CNN-RNN architecture to sequentially trace object boundaries given a bounding box. The RNN can only output one vertex at a time which could mean slow inference time depending on the number of vertices to be inferred. In

subsequent work, Curve-GCN (Ling et al., 2019) attempted to get around this limitation by modeling object annotation as a boundary control point regression problem and using graph convolutions to do the joint regression for all the control points (*i.e.*, graph nodes). It was demonstrated to be faster and also more effective than Polygon-RNN. Like Curve-GCN, DeepSnake (Peng et al., 2020) also regresses the boundary starting from an initial contour. It uses a learned snake model and achieves good results in the instance segmentation task. Polytransform (Liang et al., 2020) combines traditional instance segmentation methods with the aforementioned polygon-based methods. It predicts a rough mask with a segmentation network first, then converts the mask into a polygon. Finally, it refines the vertices of the polygon with a deforming network. The deforming network uses the attention mechanism to propagate information within the vertices of the polygon. The contour of an object can also be described by triangular grids. Deformable Grid (Gao et al., 2020) predicts deformed grids whose edges align well with the object boundary.

Since these models and tools do not use temporal information across successive frames, simply extending them for the desired video region annotation task is likely to be inferior than our proposed solution. To demonstrate it, we build two extensions of the state-of-the-art model Curve-GCN (Ling et al., 2019) and compare them with our proposed model in Sect. 4.

Video Annotation Tools Annotating objects in video is not as straight-forward as annotating them in images as it requires observing their motion paths and taking into account the possibility of change in shape over time. One of the earliest video annotation tools publicly available is VATIC (Vondrick et al., 2013) which uses inter-frame interpolation to generate bounding boxes automatically. Bounding boxes, however, are not enough for detailed analyses including pixel-wise segmentation. There have also been efforts on annotating regions in videos using active contours (Wang et al., 2014), approximation of closed boundaries using polygons (Bianco et al., 2015) and partition trees (Giro-i Nieto et al., 2010). Recent work ScribbleBox (Chen et al., 2020) provides an interactive method for video annotation by optimizing both the boxes and the masks. Despite the differences in getting the region boundaries within a frame, all these tools conduct some kind of annotation propagation or interpolation across video frames to achieve video annotation. In contrast, our proposed model jointly optimizes the boundaries in all frames.

Video Instance Segmentation and Multi-Object Tracking and Segmentation Video annotation focus on finding the correlation among all the frames, which is closely related to Video Instance Segmentation (VIS). The goal of VIS is to perform the detection, segmentation, and tracking tasks in a video sequence simultaneously. Yang et al. (2019b) first proposed VIS and by introducing a tracking branch to Mask R-CNN (He et al., 2017). MaskProp (Bertasius & Torresani, 2020) uses a mask propagation branch to propagate predicted instance masks to all the other frames, which improves the quality of segmentation and tracking. VisTR (Wang et al., 2021) proposes an end-to-end framework based on Transformers to generate all the masks of one instance at once. The temporal information integration is also very important for Multi-Object Tracking and Segmentation (MOTS). Voigtlaender et al. (2019a) introduced MOTS and proposed a baseline method called Track R-CNN. Track R-CNN extends Mask R-CNN with two 3D convolutions that exploit the temporal information and uses an association head to handle the tracking task. Annotating MOTS datasets is also time-consuming. Porzi et al. (2020) propose a track mining method to extract more training data. As a sequence-data-related task, Transformer is a very natural choice. TrackFormer (Meinhardt et al., 2021) presents a tracking-by-attention pipeline with Transformers. The tracking task is handled with the Transformer encoder-decoder without using a traditional matching strategy and motion modeling. These works give us many good hints for temporal information modeling.

3 Volumetric Graph Convolutional Network (VGCN)

Our aim is to automatically generate dense per-frame region boundary labels for all the regions of interest in all video frames. We assume that only the bounding boxes of the target regions in sparse keyframes are given by the human annotator(s). The whole task can be decomposed into subtasks each of which focuses on a key step: generating dense per-frame region boundary labels for a sub-sequence of video frames bounded by two keyframes, where a single region inside the target appears across all the frames, as shown in Fig. 1. This is a reasonable setting as human annotators can scan the whole video before annotation and place keyframes to cut the whole sequence of the target region (object, object part, or even stuff) into sub-sequences with reasonably consistent region shapes.

We have three desiderata for a video annotation model. First, besides the raw video data, the model can only assume *sparse bounding boxes* as the input during testing and inference. Second, the method should be applicable to *arbitrary regions* (of different shapes and contents) appearing in sub-sequences of arbitrary lengths. Last, *spatial-temporal inference* should be employed to ensure global joint optimization. In this paper, we propose a novel model which we refer to as Volumetric Graph Convolutional Network (VGCN) that meets all three requirements. It takes as input a pair of bounding boxes from two keyframes for data cropping

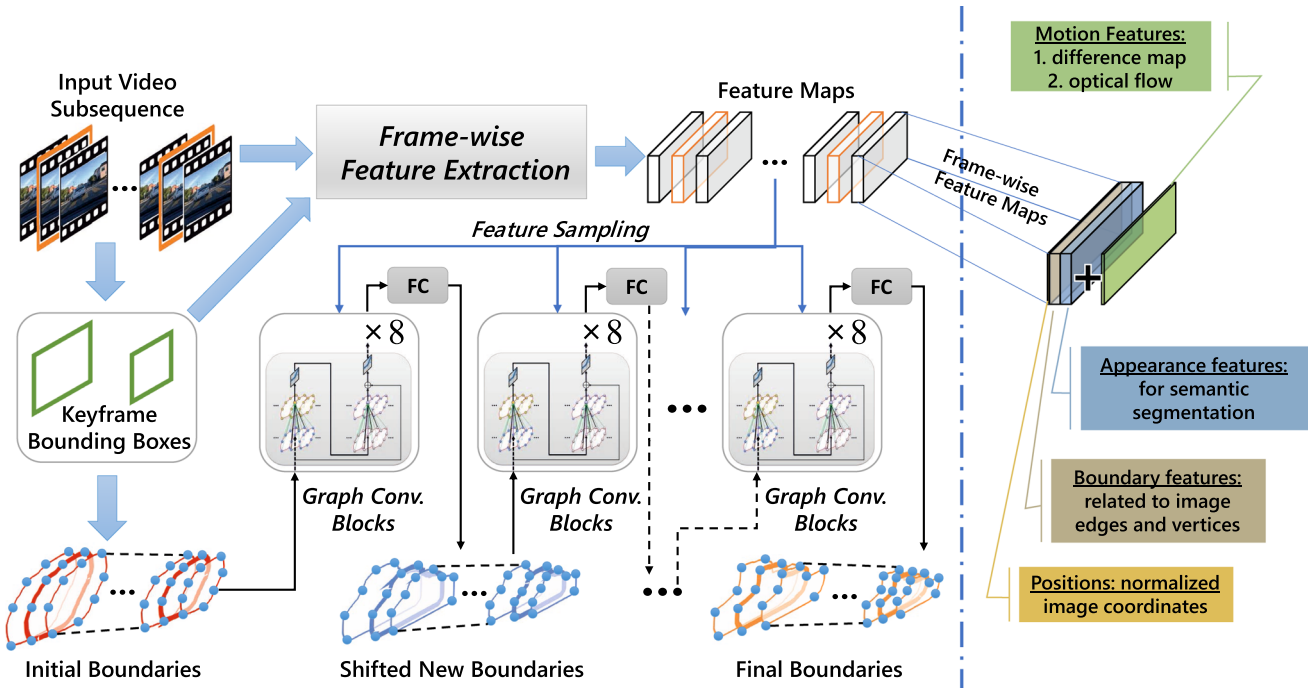

Fig. 2 The overall framework of Volumetric Graph Convolutional Network (VGCN). Per-frame feature maps are illustrated on the right

and normalization, as well as initialization. Its local graph connections (*i.e.*, edges) and weight sharing over the same types of connections allow uniform formulation and robust modeling of arbitrary local shapes. The volumetric graph convolutions integrate and propagate information spatially and temporally, leading to global spatial-temporal inference and the ability to handle arbitrary video of various lengths.

3.1 Overall Framework

As shown in Fig. 2, given an input video sub-sequence bounded by two chosen keyframes, the annotator-provided keyframe bounding boxes are used to crop the video frames, normalize them [following Ling et al. (2019)], and extract frame-wise feature maps whose contents are shown on the right. The bounding boxes also help VGCN initialize the locations of its volumetric nodes that correspond to the keypoints of desired boundaries of all video frames. Then the model samples features from the feature maps according to the node (*i.e.*, boundary keypoint) locations, and such sampled features are fed into a group of graph convolutional blocks (8 of them in our implementation) for information integration and propagation. A fully-connected (FC) layer is adopted to map the updated features of each node to its predicted location shifts. After the actual shifting of node locations, another round of feature sampling and graph convolutions can be applied to predict a new round of location shifts. This process can be iterated several times to ensure an accurate fit to the actual region boundaries.

3.2 Boundary Initialization

We draw a circle at the center of the cropped image. The diameter of this circle is 70% of the image height. We then uniformly sample N points from this circle. In our paper, we set N to 40.

3.3 Graph Structure

Suppose the sub-sequence where a target region exists is bounded by two keyframes with a sparsity factor K indicating the frame ID difference between them. The task is to find the region boundary in each frame. The 1st frame and the "$K+1$"-th frame are the keyframes with bounding-box supervision. Assume the shape of the region boundary in each frame can be well-represented by N control points $\mathcal{V}_k = \{\mathbf{cp}_k^0, \ldots, \mathbf{cp}_k^{N-1}\}$, where k indicates the k-th frame and $\mathbf{cp}_k^i = [x_k^i, y_k^i]^T$ is the location of the i-th control point in this frame, we construct a volumetric graph $G_v = (\mathcal{V}, \mathcal{E})$ covering all the frames of the sub-sequence, for which a three-frame slice is illustrated in Fig. 3. Let $\mathcal{V} = \bigcup_{k=1}^{K+1} \mathcal{V}_k$ denote the graph nodes which are the union of control points from all the frames, we define the edge set $\mathcal{E} = \mathcal{E}_s \cup \mathcal{E}_t$ by introducing two types of connections for each node $\mathbf{cp}_k^i$. The spatial connections $\mathcal{E}_s$ cover both the node's self-connection and the links between the node and its four neighboring nodes (the black lines in Fig. 3), while the temporal connections $\mathcal{E}_t$ link the node to its corresponding nodes in the two neighboring frames (cross-frame green lines) and option-

ally also those nodes' four spatial neighbors (orange lines). Depending on the temporal links, the former case is called "decomposable local connection" as the spatial and temporal links are orthogonal, and the latter is called "full local connection" or simply "full-connection." These spatial-temporal connections enable effective and efficient information integration and propagation among the graph nodes. For a better model-data fit, we add one more frame to each end of the subsequence. As a result, our model deals with $K+3$ frames and has $\mathcal{V} = \bigcup_{k=0}^{K+2} \mathcal{V}_k$.

3.4 Feature Extraction and Representation

With the two keyframe bounding boxes, we get the bounding boxes for other frames with linear interpolation. Then a 15% margin is added to each size of the bounding box to ensure a sufficient coverage of data even when the bounding boxes are tight. The extended bounding boxes are used to crop the frames and normalize the cropped areas to uniform sizes (with spatial coordinates normalized to [0, 1] by [0, 1]). Following Curve-GCN (Ling et al., 2019), we extract three types of features from the normalized data as shown in Fig. 2: appearance features computed using the ResNet-50 (He et al., 2016) model and boundary features computed using additional branches to predict the probability of the existence of an object edge/vertex on a 28×28 grid and position features (normalized pixel coordinates).

We propose two means to add motion information: optical flow and temporal difference map. For optical flow, we integrate it by concatenating the optical flow map [obtained by FlowNet2.0 (Ilg et al., 2017)] to the original image data before feature extraction. It contributes to the boundary features via early fusion, making it motion-aware. For the temporal difference map (Lipton et al., 1998), we also concatenate it with the original images. Given three consecutive images I_1, I_2 and I_3, two temporal difference maps are computed by $|I_1 - I_2|$ and $|I_3 - I_2|$. In the temporal difference maps, boundaries of moving objects are easier to detect and background regions are suppressed. Optical flow is computed for the whole images, while the temporal difference map is computed for cropped patches. Compared to an optical flow map, a difference map is much faster to compute and doesn't require an extra model.

3.5 Graph Convolutional Block

Following Curve-GCN, we also adopt the multi-layer GCN architecture, where each layer has a Graph ResNet structure as shown in Fig. 4. Mathematically, the graph convolution for an arbitrary node $\mathbf{cp}_k^i$ can be formulated as

$$\begin{aligned}\hat{f}_{k,i}^l = W_o^l f_{k,i}^l + \sum_{\mathbf{cp}_k^j \epsilon \mathcal{E}_s(\mathbf{cp}_k^i)} W_s^l f_{k,j}^l \\ + \sum_{\mathbf{cp}_{k-1}^j \epsilon \mathcal{E}_t(\mathbf{cp}_k^i)} W_t^l f_{k-1,j}^l + \sum_{\mathbf{cp}_{k+1}^j \epsilon \mathcal{E}_t(\mathbf{cp}_k^i)} W_t^l f_{k+1,j}^l,\end{aligned} \tag{1}$$

where W_o^l, W_s^l, and W_t^l are the weight matrices at layer l to be learned for transforming the control point's own features, the features of its spatial neighbors (*i.e.*, the nodes connected to $\mathbf{cp}_k^i$ via edges in $\mathcal{E}_s(\mathbf{cp}_k^i)$), and the features of its temporal neighbors [(*i.e.*, the nodes connected to $\mathbf{cp}_k^i$ via edges in $\mathcal{E}_t(\mathbf{cp}_k^i)$], respectively, and $\hat{f}_{k,i}^l$ is the updated feature for $\mathbf{cp}_k^i$ at layer l after one round of information propagation. Note that in the "full-connection" case the two types of temporal edges (the ones connecting corresponding nodes, and the others) have different weight matrices. After that, there is a ReLU activation $g_{k,i}^l = ReLU(\hat{f}_{k,i}^l)$.

Then, there is another round of convolution

$$\begin{aligned}\hat{g}_{k,i}^l = \tilde{W}_o^l g_{k,i}^l + \sum_{\mathbf{cp}_k^j \epsilon \mathcal{E}_s(\mathbf{cp}_k^i)} \tilde{W}_s^l g_{k,j}^l \\ + \sum_{\mathbf{cp}_{k-1}^j \epsilon \mathcal{E}_t(\mathbf{cp}_k^i)} \tilde{W}_t^l g_{k-1,j}^l + \sum_{\mathbf{cp}_{k+1}^j \epsilon \mathcal{E}_t(\mathbf{cp}_k^i)} \tilde{W}_t^l g_{k+1,j}^l,\end{aligned} \tag{2}$$

where $\tilde{W}_o^l$, $\tilde{W}_s^l$, and $\tilde{W}_t^l$ are the corresponding weight matrices. Finally, the residual structure combines the updated feature $\hat{g}_{k,i}^l$ and the original feature $f_{k,i}^l$ to generate the input feature for the next layer, together with a ReLU activation $f_{k,i}^{l+1} = ReLU(\hat{g}_{k,i}^l + f_{k,i}^l)$.

After L layers ($l = 0, \ldots, L-1$), $f_{k,i}^L$ is fed into a single fully connected layer to predict the location shift $(\Delta x_k^i, \Delta y_k^i)$ for $\mathbf{cp}_k^i$. Then its coordinates can be updated as $\tilde{\mathbf{cp}}_k^i = [x_k^i + \Delta x_k^i, y_k^i + \Delta y_k^i]^T$. With the new control point location, we can get the new features $\tilde{f}_{k,i}^{l+1}$ and have it go through the whole VGCN module again to get another location shift. Such shifting of control points can be done several times to nudge the graph nodes to move to the actual region boundaries.

As shown in Fig. 4, the convolution can be viewed as spatio-temporal information integration and propagation. When several blocks are applied, each node can get information from a significantly larger area of the video.

3.6 Loss Function

We use the Normalized Bi-directional Chamfer Distance (NBCD) as the loss to get supervision from the ground-truth boundaries. It directly measures the accuracy of keypoints

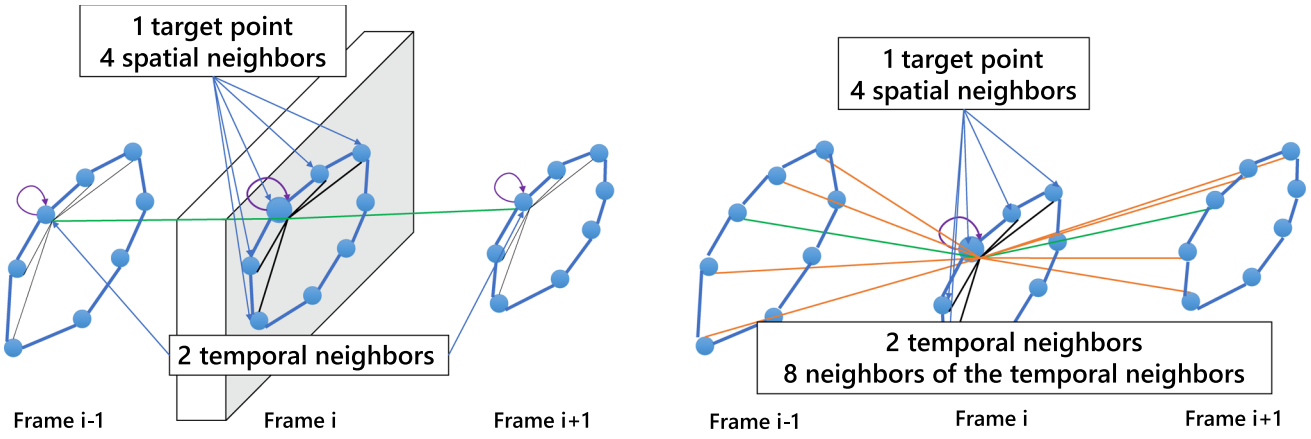

Fig. 3 Graph structures of two variants of VGCN, full local connection (recommended) and decomposable local connection (simplest), illustrated with three adjacent frames. Note that all the nodes of an intermediate frame will have exactly the same number of edges. Some edges for "full local connection" are omitted for better visibility

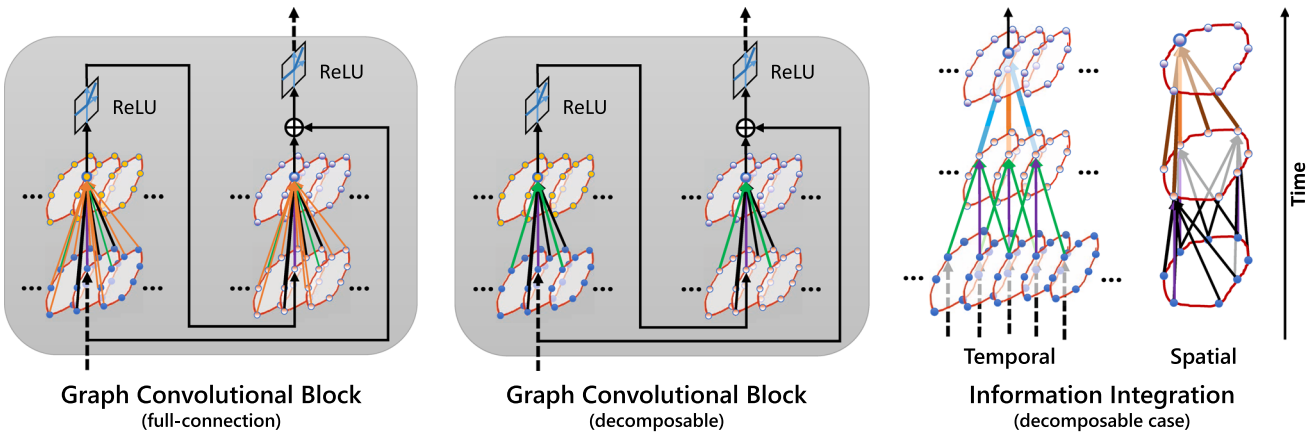

Fig. 4 The graph convolutional block of VGCN. Both full local connection (fullly-connected) and decomposable local connection (basic) are illustrated. The right subfigure shows how the information from graph nodes (boundary keypoints) is integrated spatially and temporally, using the decomposable model as an example

and corresponds to the NBCD metric in our performance evaluation.

3.7 Arbitrary Input Frame Model

If we have sufficient memory for computing, our model can process arbitrary-sized video frames. Following the data format of Pytorch (Paszke et al., 2019), N frames of RGB images with the shape of $H \times W$ are organized in the order of $[N \times C \times H \times W]$. The encoder network takes $[N \times C \times H \times W]$ data as input and outputs an $[N \times D \times H/R \times W/R]$ tensor, where D is the feature dimension. We extract features from this tensor according to the positions of P number of nodes. The extracted feature is $[N \times P \times D]$ and the adjacent matrix in GCN is $[P \times D]$. We extend the adjacent matrix in GCN to $[N \times P \times D]$ to support the information propagation between neighboring frames. Every time we run the GCN, it can receive information from two adjacent frames. By running it multiple times, information can be propagated within a sequence.

4 Experimental Results

4.1 Datasets and Evaluation Metrics

Datasets. *Synthia* (Bengar et al., 2019) is a recently released synthetic driving dataset that has ground truth object bound-

Table 1 Dataset description

Dataset setting		Car	Person (or Pedestrian)	Truck	Bus
Synthia-10	Synthetic dataset, 13 frames	20917/0.97	21320/0.89	5538/0.97	2548/0.99
KITTI-MOTS-4	Real dataset, 7 frames	18774/0.92	5068/0.87	–/–	–/–
KITTI-MOTS-10	Real dataset, 13 frames	14976/0.84	3549/0.78	–/–	–/–
KITTI-MOTS-18	Real dataset, 21 frames	11382/0.75	2394/0.69	–/–	–/–

Dataset setting	Pedestrian	Bicycle	Truck	Car	Bus
BDD-MOTS-4	10969/0.91	840/0.95	11263/0.94	10843/0.94	5222/0.94

Dataset setting	Train	Val	Test
Synthia-10	2127	357	1404
KITTI-MOTS-4	2325	527	554
KITTI-MOTS-10	962	233	230
KITTI-MOTS-18	436	109	111
BDD-MOTS-4	3556	1088	947

Upper part: instance numbers and box mean Intersection over Union (mIoU) for each class in different datasets; lower part: sequence numbers for different splits in the dataset. '–' means that the number is not available due to the category missing in the corresponding dataset

aries for every frame. This dataset contains 178 training video sequences captured at 25fps, with lengths ranging from 15 seconds to 30 seconds. We consider dynamically moving objects relevant to driving scenarios: person, car, truck and bus. *KITTI MOTS* (Voigtlaender et al., 2019b) is a real dataset for Multi-Object Tracking and Segmentation (MOTS). It contains 21 training sequences (12 for training, 9 for validation), and 4 testing sequences are reserved for MOTS Challenge. The dataset only has two object categories: pedestrian and car, with 99 pedestrians and 431 cars for training, 68 pedestrians and 151 cars for validation. We use the 12 training sequences for training and the validation set for testing. Synthia is about 5 times larger in terms of frame/subsequence numbers. *BDD-MOTS* (Yu et al., 2020) contains 60 training videos, 10 validation videos, and 20 testing videos. It is about 2 times larger than KITTI-MOTS. We use categories 'Pedestrian', 'Bicycle', 'Truck', 'Car', and 'Bus' in our experiments.

Evaluation Metrics Besides the widely used mIoU and F1-score measure (Ling et al., 2019) that measures mask and boundary matching accuracy, respectively, we also use the Normalized Bi-directional Chamfer Distance (NBCD) to directly measure the keypoint matching accuracy, so that the performance can be checked from different perspectives.

4.2 Implementation Details

For an N-frame sequence, only two key-frames contain ground truth bounding boxes and polygons for training our VGCN, and the other frames provide no supervision. In testing, the input is just two bounding boxes of the key-frames.

Synthia All the models are trained using Adam (Kingma & Ba, 2015) with weight decay of $1e^{-5}$. We use a learning rate of $3e^{-5}$ for 20 epochs, then use $3e^{-6}$ for 20 epochs. The SGCN is trained with a batch size of 16. The VGCN models are trained with a sequence with two annotated keyframes. For those models with extra motion information in the encoder network, we find it is hard to converge. So we first train an SGCN for 1 epoch, then use the parameters of SGCN to initialize our VGCN for warming up.

KITTI-MOTS We use a learning rate of $3e^{-5}$ for 5 epochs, then continue training with a smaller rate $3e^{-6}$ for 5 epochs. All the models are pre-trained on Synthia.

BDD-MOTS We train the model from scratch. The learning rate and the number of epochs follow the setting of Synthia.

4.3 Data Generation

Four rules are followed to generate sequences of images: there are no overlapping images in different sequences; the target object should appear in every frame in the sequence; in every frame, the area of the target object should always be larger than 100 pixels; all the objects should only contain one component.

In the Synthia dataset, most of the instances are car or person. To balance the ratio of different categories, we randomly remove some of the car and person sequences. Objects like bicycles are annotated by the regions their parts occupy but not their holistic contours in the Synthia dataset, which are hard to represent with a single polygon. In the final data sequences, such objects do not exist. Since new data sequences have better quality for training, the performance

Table 2 Results on Synthia-10, measured by mean Intersection over Union (mIoU)

Model		Car	Truck	Person	Bus	Average
Name	Property	(8.39k)	(0.61k)	(8.56k)	(0.67k)	
SGCN	Frame-wise, based on Ling et al. (2019)	86.26	85.87	74.56	81.46	82.04
SGCN-smoothed	Video-wise (indirect)	86.27	85.81	74.10	81.58	81.94
VGCN-basic	Video-wise	86.86	87.09	**76.53**	83.29	83.44
VGCN-full + Optical-flow	Video-wise, optical flow	87.53	87.98	76.16	83.88	83.89
VGCN-full + Difference-map	Video-wise, difference map	**87.78**	**88.25**	76.36	**85.05**	**84.36**

Since different object categories have different amounts of samples, the number of frames in testing for each category is shown under its name
Bold values indicate the best result in the comparison

of our baseline SGCN is largely improved. Though VGCN can handle sub-sequences of various lengths, we use equal lengths by fixing K (default: 10) to ease model transfer , comparisons across datasets and the ablation study on sparsity. The training split is from the original training sets, while the validation and test splits are from the original test sets.

For the KITTI-MOTS dataset, we also apply the aforementioned four rules. In KITTI-MOTS, objects move much faster, so the mIoU value between an interpolated box and the corresponding ground truth box is generally larger. We set the interval between key-frames K to 4 to keep a similar expected mIoU value with that for Synthia , whose K is set to 10. Note that the actual number of frames for a sequence is $K + 3$, as shown in Table 1.

For BDD-MOTS dataset, K is set to 4. We keep the mIoU of all interpolated boxes higher than 0.7, if not, we drop the sequence. Since the 'Car' category is dominating, we only use $1/11$ of them to get a similar number to other categories.

Table 1 shows the box mIoU, instance numbers, and sequence numbers of the training, validation, and test splits for our experiments. For the KITTI-MOTS experiments, models are trained on KITTI-MOTS-4, and then tested on KITTI-MOTS-4, KITTI-MOTS-10 and KITTI-MOTS-18.

4.4 Results

4.4.1 Models for Comparison

Instead of video-wise joint boundary inference, one may simply apply a frame-wise model (here we choose Curve-GCN (Ling et al., 2019) as it is the state-of-the-art and also the most relevant model) to each video frame with either the provided bounding box (in case of keyframes) or some interpolated bounding box (for a intermediate frame). We refer to this model as Spatial Graph Convolutional Network (SGCN), as it is also based on GCN and only does the graph convolutions spatially. Despite its simplicity, SGCN has a natural limitation of omitting the temporal relationships among successive video frames. To overcome it, one may also think about simply smoothing the results of SGCN on successive video frames using a B-spline function, so that the overall model can be made indirectly video-wise. Such a simple solution is named 'SGCN-smoothed'. However, we believe that a direct modeling of temporal relationships in the model like the proposed VGCN is necessary and superior. To better show the performance difference of direct spatio-temporal modeling and indirect result smoothing, we also test a simplified version of VGCN named 'VGCN-basic', by only keeping the minimal temporal connection (*i.e.*, the decomposable local connection as shown in Fig. 3) and excluding the motion features. Note that for a fair comparison, all the compared models are trained with the same data which only have ground-truth on the sparse key frames.

4.4.2 Effectiveness of VGCN

As shown in Tables 2 and 3, VGCN significantly outperforms SGCN for all metrics on the Synthia dataset. Interestingly, the negligible differences between the results of SGCN-smoothed and SGCN also indicate that simple temporal smoothing is not effective. We conduct the generalization experiments on KITTI-MOTS by fine-tuning the models pre-trained on Synthia. On BDD-MOTS, to leverage the large amount of data, we train both SGCN and VGCN from scratch. As shown in Table 7, VGCN still outperforms SGCN for all metrics. Even SGCN is better in mIoU for the 'Bus,' and VGCN achieves a better score on NBCD.

4.4.3 Generalizability

Since our method makes no assumptions on the data, it can be applied to arbitrary video data. We conduct two types of experiments to validate the generalizability of VGCN and compare its performance with those of relevant past works. One directly applies models trained on Synthia to the test set of KITTI-MOTS. Table 4 shows the results in the upper part. The other is fine-tuning the pre-trained model (trained on Synthia) on the training set of KITTI-MOTS-4 and then testing the fine-tuned models on its test set. If we directly apply the models, VGCN-basic outperforms other models in mIoU.

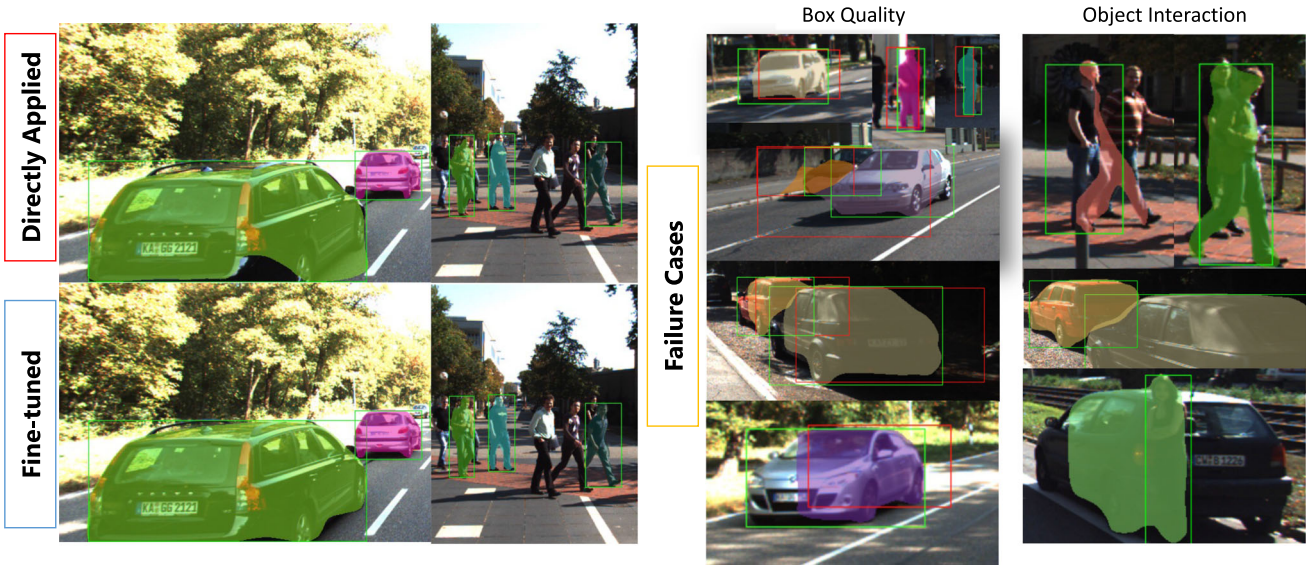

Fig. 5 Comparison of the generalizability of VGCN, and two representative failure cases

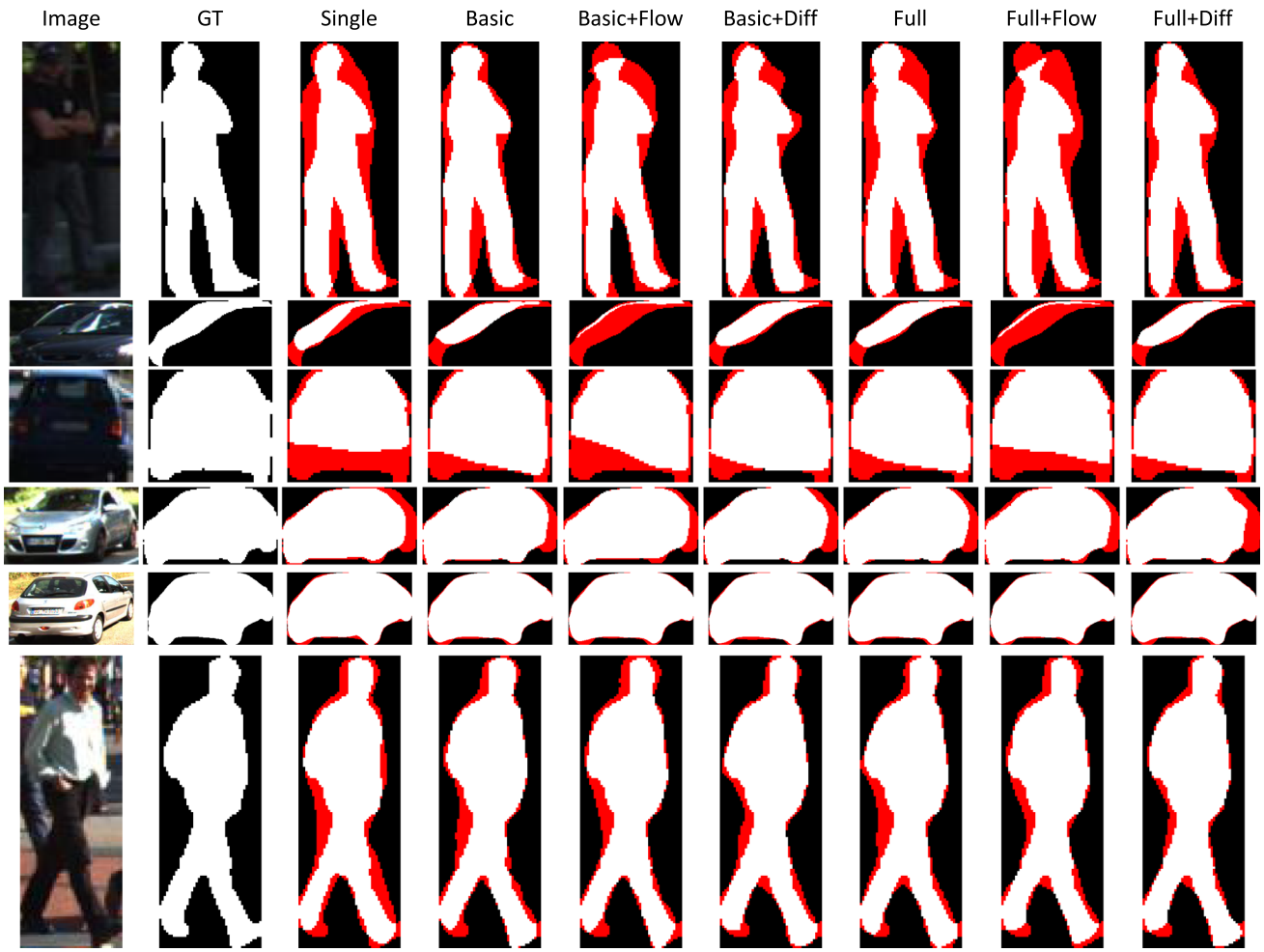

Fig. 6 Qualitative results on KITTI-MOTS

Table 3 Results on Synthia-10 in Normalized Bi-directional Chamfer Distance (NBCD) and F1-score

Model	NBCD				Average
	Car (8.39k)	Truck (0.61k)	Person (8.56k)	Bus (0.67k)	
SGCN	6.71	4.81	2.78	5.15	4.86
SGCN-smoothed	6.74	4.83	2.86	5.16	4.90
VGCN-basic	6.51	4.12	**2.51**	4.73	4.47
VGCN-full + Optical-flow	6.39	**4.01**	2.57	4.78	4.44
VGCN-full + Difference-map	**6.36**	4.04	2.52	**4.48**	**4.35**
Model	F-score (1px/2px)				Average
	Car (8.39k)	Truck (0.61k)	Person (8.56k)	Bus (0.67k)	
SGCN	70.50/83.47	76.60/87.99	81.57/91.32	59.22/79.27	71.97/85.51
SGCN-smoothed	69.98/83.27	76.37/88.00	80.31/ 90.74	59.20/79.44	71.46/85.36
VGCN-basic	74.19/85.93	79.44/90.17	**85.44/93.32**	64.08/81.79	75.79/ 87.81
VGCN-full + Optical-flow	75.32/86.71	**82.27**/91.71	84.57/92.86	69.32/**85.02**	77.87/**89.08**
VGCN-full + Difference-map	**75.76/86.78**	82.01/ **91.81**	84.83/93.07	**71.19**/84.47	**78.45**/89.03

Bold values indicate the best result in the comparison

Table 4 Results on KITTI-MOTS-4

Model	mIoU		NBCD		F1-score (1px/2px)	
	Car	Ped	Car	Ped	Car	Ped
Directly applied						
SGCN	79.97	71.76	6.85	5.28	52.59/73.12	49.68/68.86
VGCN-basic	**80.97**	**72.92**	**6.66**	4.95	54.08/74.67	50.48/71.02
VGCN-full + Optical-flow	79.41	72.52	6.94	5.03	55.12/75.01	**52.00/72.04**
VGCN-full + Difference-map	80.84	72.76	6.80	**4.83**	**58.34/76.99**	51.03/71.49
Fine-tuned						
SGCN	86.81	77.85	5.21	4.27	70.23/86.96	62.42/79.83
VGCN-basic	86.87	79.61	5.09	3.93	72.42/88.81	67.87/83.87
VGCN-full + Optical-flow	86.64	78.97	5.21	4.15	71.89/87.18	66.48/82.73
VGCN-full + Difference-map	**87.42**	**80.39**	**5.02**	**3.83**	**73.80/89.34**	**69.40/85.03**

Upper part: directly applying the models trained on Synthia; lower part: fully fine-tuned models. 'Ped' stands for 'Pedestrian'
Bold values indicate the best result in the comparison

VGCN-full with optical flow and VGCN-full with difference map perform better on Car and Ped in F1-score, respectively. The basic version of VGCN generalizes better without fine-tuning. For fine-tuned models, VGCN-full with difference map outperforms all the other models. VGCN-full is still a little worse than SGCN for Car instances. These findings indicate that a model with optical flow performs comparatively worse in generalization than a model with difference map. Our VGCN can handle more complex objects like Person. Examples of how fine-tuning benefits VGCN are shown in Fig. 5. The figure also shows two representative failure cases: undesirable regions due to badly interpolated boxes and extra part caused by object interaction. The qualitative results are shown in Fig. 6. The regions in red are incorrect predictions.

4.4.4 Running Time

Inference on a 13-frame sequence takes about 1.1s on a NVIDIA RTX2080Ti GPU. Note that interactive correction can be much faster, as features can be pre-computed and refinement is much faster than inference-from-scratch as shown in (Ling et al., 2019). For this, we believe the computational speed of our method would be sufficient for real usage.

4.5 Ablation Studies

We add motion information to VGCN-basic for our ablation studies. Besides fully fine-tuning the pre-trained models on KITTI-MOTS dataset, we also fine-tune the encoder module

Table 5 Ablation studies on VGCN components using Synthia-10, measured by mIoU, NBCD and F-score

Model	mIoU				Average
	Car (8.39k)	Truck (0.61k)	Person (8.56k)	Bus (0.67k)	
VGCN-basic	86.86	87.09	**76.53**	83.29	83.44
VGCN-basic + Optical-flow	87.44	87.69	76.14	**85.58**	84.21
VGCN-basic + Difference-map	87.08	87.78	76.30	85.16	84.08
VGCN-full (*i.e.*, VGCN-basic + Full-connection)	87.00	87.88	75.93	85.49	84.08
VGCN-full + Optical-flow	87.53	87.98	76.16	83.88	83.89
VGCN-full + Difference-map	**87.78**	**88.25**	76.36	85.05	**84.36**
Model	NBCD				Average
	Car (8.39k)	Truck (0.61k)	Person (8.56k)	Bus (0.67k)	
VGCN-basic	6.51	4.12	2.51	4.73	4.47
VGCN-basic + Optical-flow	6.45	4.17	2.56	4.56	4.44
VGCN-basic + Difference-map	6.42	4.08	**2.49**	4.71	4.43
VGCN-full (*i.e.*, VGCN-basic + Full-connection)	6.37	4.09	2.52	**4.36**	**4.33**
VGCN-full + Optical-flow	6.39	**4.01**	2.57	4.78	4.44
VGCN-full + Difference-map	**6.36**	4.04	2.52	4.48	4.35
Model	F-score(1px/2px)				Average
	Car (8.39k)	Truck (0.61k)	Person (8.56k)	Bus (0.67k)	
VGCN-basic	74.19/85.93	79.44/90.17	**85.44**/93.32	64.08/81.79	75.79/87.81
VGCN-basic + Optical-flow	74.79/86.32	80.20/90.38	84.47/92.96	**73.86**/88.60	78.33/89.56
VGCN-basic + Difference-map	74.64/86.06	79.49/90.96	85.28/ **93.39**	67.57/83.46	76.75/88.47
VGCN-full (*i.e.*, VGCN-basic + Full-connection)	74.89/86.53	79.73/90.62	85.10/93.13	73.08/**90.41**	78.20/**90.17**
VGCN-full + Optical-flow	75.32/86.71	**82.27**/91.71	84.57/92.86	69.32/85.02	77.87/89.08
VGCN-full + Difference-map	**75.76/86.78**	82.01/**91.81**	84.83/93.07	71.19/84.47	**78.45**/89.03

Bold values indicate the best result in the comparison

Table 6 Ablation studies on VGCN components using KITTI-MOTS-4

Model	mIoU		NBCD		F1-score (1px/2px)	
	Car	Ped	Car	Ped	Car	Ped
Directly applied						
SGCN	79.97	71.76	6.85	5.28	52.59/73.12	49.68/68.86
VGCN-basic	80.97	**72.92**	6.66	4.95	54.08/74.67	50.48/71.02
VGCN-basic + Optical-flow	**81.40**	71.47	6.74	5.48	56.80/76.49	50.48/70.09
VGCN-basic + Difference-map	79.77	71.32	6.69	5.33	51.86/72.87	49.19/69.01
VGCN-full (*i.e.*, VGCN-basic + Full-connection)	80.47	71.83	**6.44**	4.84	55.73/75.44	49.53/70.25
VGCN-full + Optical-flow	79.41	72.52	6.94	5.03	55.12/75.01	**52.00/72.04**
VGCN-full + Difference-map	80.84	72.76	6.80	**4.83**	**58.34/76.99**	51.03/71.49
Encoder Fixed						
SGCN	85.28	74.26	5.79	4.89	66.73/83.02	54.33/72.90
VGCN-basic	85.38	76.56	5.81	**4.48**	67.92/84.16	59.56/77.37
VGCN-basic + Optical-flow	85.22	74.59	5.86	5.04	66.37/82.37	56.39/74.37
VGCN-basic + Difference-map	**85.97**	76.48	**5.54**	4.59	**69.74/85.13**	59.82/77.78
VGCN-full (*i.e.*, VGCN-basic + Full-connection)	84.98	76.23	5.75	4.52	66.81/83.26	59.23/77.24
VGCN-full + Optical-flow	85.75	75.22	5.74	4.86	67.79/83.75	56.54/74.89
VGCN-full + Difference-map	84.87	**77.11**	5.71	4.48	65.97/82.79	**61.03/78.70**
GCN Fixed						
SGCN	87.05	77.67	5.12	4.28	70.48/87.25	62.22/79.65
VGCN-basic	**87.54**	79.70	4.98	3.94	**74.11**/89.46	68.56/83.95
VGCN-basic + Optical-flow	86.42	78.24	5.51	4.54	72.83/86.10	65.10/80.75
VGCN-basic + Difference-map	87.16	80.23	5.02	**3.85**	73.18/89.20	**69.09/84.71**
VGCN-full (*i.e.*, VGCN-basic + Full-connection)	87.39	79.68	5.02	3.97	73.01/88.82	67.93/83.84
VGCN-full + Optical-flow	87.03	77.69	5.28	4.75	74.08/87.84	64.36/79.66
VGCN-full + Difference-map	87.31	**80.26**	**4.98**	3.90	73.95/**89.55**	69.01/84.68
Fully Fine-tuned						
SGCN	86.81	77.85	5.21	4.27	70.23/86.96	62.42/79.83
VGCN-basic	86.87	79.61	5.09	3.93	72.42/88.81	67.87/83.87
VGCN-basic + Optical-flow	87.08	78.20	5.14	4.24	72.88/88.11	64.72/81.70
VGCN-basic + Difference-map	87.35	80.27	5.07	**3.82**	73.70/89.23	69.08/84.67
VGCN-full (*i.e.*, VGCN-basic + Full-connection)	**87.46**	79.84	**5.01**	3.96	73.02/89.09	68.34/**86.57**
VGCN-full + Optical-flow	86.64	78.97	5.21	4.15	71.89/87.18	66.48/82.73
VGCN-full + Difference-map	87.42	**80.39**	5.02	3.83	**73.80/89.34**	**69.40**/85.03

Directly applied: directly applying the models trained on Synthia; encoder fixed: GCN fine-tuned models (pre-trained on Synthia); GCN fixed: encoder fine-tuned models; fully fine-tuned: fully fine-tuned models. 'Ped' stands for 'Pedestrian'
Bold values indicate the best result in the comparison

and GCN module separately to evaluate their generalization abilities.

VGCN components As shown in Tables 5 and 6, both the motion features and the full connection can help improve the performance. None of the components can benefit all the categories. VGCN-full outperforms VGCN-basic in Car and Truck but gets worse results in Person and Bus in the Synthia dataset. In KITTI-MOTS, fine-tuned VGCN-full with difference map is better than VGCN-full in Pedestrian, but it is a little worse for Car. The diversity of object shapes makes it impossible for a fixed model to achieve the best results in all categories. VGCN-full with optical flow has similar performance to SGCN in mIoU, however, the former model is much better in F1-score. Since mIoU penalizes more for bad cases, we can say VGCN-full with optical flow performs better for good cases.

Effectiveness of encoder and GCN Table 4 shows results for this comparison of fine-tuning strategies. If we fix the encoder module, the performance of every model will be similar. If we fine-tune the encoder module only, our proposed model will outperform the single-frame model. This indicates that for these two datasets, the topological relationships of objects

Table 7 Results on BDD-MOTS, measured by mIoU, NBCD and F-score

Model	mIoU					Average
	Pedestrian (1.45k)	Bicycle (0.07k)	Truck (1.61k)	Car (1.33k)	Bus (0.28k)	
SGCN	68.06	59.23	83.02	80.78	**84.35**	75.09
VGCN-full + Difference-map	**69.71**	**63.63**	**83.14**	**81.08**	83.96	**76.30**
Model	NBCD					Average
	Pedestrian (1.45k)	Bicycle (0.07k)	Truck (1.61k)	Car (1.33k)	Bus (0.28k)	
SGCN	7.88	15.30	16.08	11.74	15.25	13.25
VGCN-full + Difference-map	**7.49**	**14.44**	**15.78**	**11.61**	**14.78**	**12.82**
Model	F-score(1px/2px)					Average
	Pedestrian (1.45k)	Bicycle (0.07k)	Truck (1.61k)	Car (1.33k)	Bus (0.28k)	
SGCN	47.76/65.04	22.97/35.88	48.51/64.11	48.59/65.58	**44.45**/59.48	42.54/58.02
VGCN-full + Difference-map	**50.69/68.01**	**26.48/39.26**	**48.59/64.20**	**50.17/66.70**	43.51/**59.59**	**43.89/59.55**

Bold values indicate the best result in the comparison

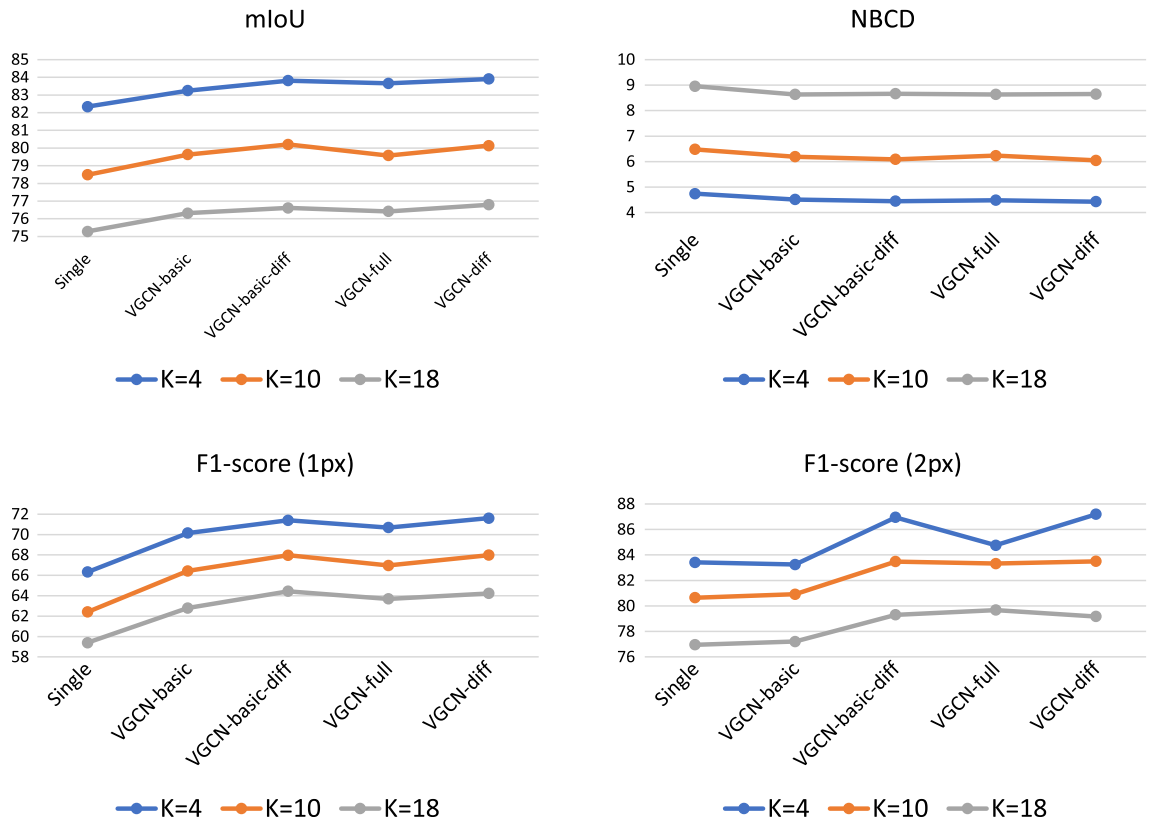

Fig. 7 Performance changes w.r.t. annotation sparsity (keyframe interval K)

Table 8 The performance of different iterations

Iteration number	mIoU	F-score(1px)	F-score(2px)	NBCD
Iter. 1	70.79	28.29	41.98	15.97
Iter. 2	74.97	37.94	53.49	13.71
Iter. 3	**76.30**	**43.89**	**59.55**	**12.82**

Bold values indicate the best result in the comparison

are similar, while the appearance features are different. This is why fine-tuning GCN is worse than re-training the encoder module. VGCN-basic models also perform well in KITTI-MOTS with a fixed encoder, a simple model may be easier to fine-tune than a fully connected model.

Motion information As shown in Table 6, the temporal difference map achieves better accuracy than when using optical flow. This is mainly due to the fact that the optical flow is less accurate in KITTI-MOTS as it consists of real images unlike Synthia. When an object moves fast, an optical flow map may lose the motion information. In contrast, the temporal difference map is computed for cropped images, and it can still catch the object and find the background and non-background regions for our interpolated boxes. Also shown in Table 7, the model with the temporal difference map achieves good performance on the BDD-MOTS dataset.

Sparsity As shown in Fig. 7, we directly apply our model trained on KITTI-MOTS-4 to KITTI-MOTS-10 and 18. All the models perform worse when K is increased. VGCN models always outperform SGCN.

Graph Iteration Our model uses three iterations. As shown in Table 8, all the metrics are improved after each iteration. Note that the parameters are not shared for the model in each iteration. A larger iteration number will lead to a larger final model.

5 Conclusions

This paper presents a novel tool for video region annotation that can generate dense per-frame region boundaries with only bounding boxes on sparse keyframes provided by the annotators. We believe our method opens a new avenue of research for significantly extending video supervision for general deep vision applications. An important future work is to extend the method to allow interactive correction in a human-in-the-loop annotation scheme.

Acknowledgements This work was in part supported by JSPS KAKENHI 17K20143 and SenseTime Japan.

References

Acuna, D., Ling, H., Kar, A., & Fidler, S. (2018). Efficient interactive annotation of segmentation datasets with polygon-rnn++. In: *Proceedings CVPR* (pp. 859–868).

Bengar, J. Z., Gonzalez-Garcia, A., Villalonga, G., Raducanu, B., Aghdam, H. H., Mozerov, M., Lopez, A. M., & van de Weijer, J. (2019). Temporal coherence for active learning in videos. In *IEEE/CVF international conference on computer vision workshop* (ICCVW) (pp. 914–923).

Bertasius, G., & Torresani, L. (2020). Classifying, segmenting, and tracking object instances in video with mask propagation. In *Proceedings CVPR* (pp. 9739–9748).

Bianco, S., Ciocca, G., Napoletano, P., & Schettini, R. (2015). An interactive tool for manual, semi-automatic and automatic video annotation. *CVIU, 131,* 88–99.

Castrejon, L., Kundu, K., Urtasun, & R., Fidler, S. (2017). Annotating object instances with a polygon-rnn. In *Proceedings CVPR* (pp. 5230–5238).

Chen, B., Ling, H., Zeng, X., Gao, J., Xu, Z., & Fidler, S. (2020). Scribblebox: interactive annotation framework for video object segmentation. In *Proceedings* (pp. 293–310). ECCV Springer.

Cho, M., Kwak, S., Schmid, C., Ponce, J. (2015). Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In *Proceedigs CVPR* (pp. 1201–1210).

Ding, M., Wang, Z., Zhou, B., Shi, J., Lu, Z., & Luo, P. (2020) Every frame counts: Joint learning of video segmentation and optical flow. In *Proceedings AAAI* (pp 10713–10720).

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *IJCV, 111*(1), 98–136.

Gao, J., Wang, Z., Xuan, J., & Fidler, S. (2020). Beyond fixed grid: Learning geometric image representation with a deformable grid. *Proc* (pp. 108–125). ECCV: Springer.

Giro-i Nieto, X., Camps, N., & Marques, F. (2010). Gat: A graphical annotation tool for semantic regions. *Multimedia Tools and Applications, 46*(2–3), 155–174.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017) Mask r-cnn. In *Proceedings ICCV* (pp. 2961–2969).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings CVPR* (pp. 770–778).

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., & Brox, T. (2017) Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings CVPR* (pp. 2462–2470).

Jain, SD., Xiong, B., & Grauman, K. (2017). Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In *Proceedings CVPR* (pp. 2117–2126).

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd international conference on learning representations, ICLR 2015* 2015, Conference Track Proceedings. arXiv:1412.6980

Lee, H. Y., Huang, J. B., Singh, M., & Yang, M. H. (2017). Unsupervised representation learning by sorting sequences. In *Proceeding ICCV* (pp. 667–676).

Liang, J., Homayounfar, N., Ma, W. C., Xiong, Y., Hu, R., & Urtasun, R. (2020). Polytransform: Deep polygon transformer for instance segmentation. In *Proceedings CVPR* (pp. 9131–9140).

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Proceedings ECCV* (pp. 740–755).

Ling, H., Gao, J., Kar, A., Chen, W., & Fidler, S. (2019). Fast interactive object annotation with curve-gcn. In *Proceedings CVPR*, (pp. 5257–5266).

Lipton, A., Fujiyoshi, H., & Patil, R. (1998). Moving target classification and tracking from real-time video. In *Proceedings fourth IEEE workshop on applications of computer vision*. WACV'98 (Cat. No.98EX201) (pp. 8–14), https://doi.org/10.1109/ACV.1998.732851.

Li, J., Zhao, Y., Fu, J., Wu, J., & Liu, J. (2019). Attention-guided network for semantic video segmentation. *IEEE Access, 7,* 140680–140689.

Meinhardt, T., Kirillov, A., Leal-Taixe, L., Feichtenhofer, C. (2021). Trackformer: Multi-object tracking with transformers. In *Proc CVPR*.

Nabavi, S. S., Rochan, M., & Wang, Y. (2018). Future semantic segmentation with convolutional LSTM. In *Procceedings BMVC* (p 137).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.) *Advances in neural information processing systems 32* (Curran Associates, Inc., pp. 8024–8035). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Paul, M., Mayer, C., Gool, L. V., Timofte, R. (2020). Efficient video semantic segmentation with labels propagation and refinement. In *Proceedings WACV* (pp. 2873–2882).

Peng, S., Jiang, W., Pi, H., Li, X., Bao, H., & Zhou, X. (2020). Deep snake for real-time instance segmentation. In *Proceedings CVPR* (pp. 8533–8542).

Porzi, L., Hofinger, M., Ruiz, I., Serrat, J., Bulo, S. R., & Kontschieder, P. (2020). Learning multi-object tracking and segmentation from automatic annotations. In *Proceeding CVPR* (pp. 6846–6855).

Rother, C., Kolmogorov, V., & Blake, A. (2004). "grabcut" interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG), 23*(3), 309–314.

Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., & Leibe, B. (2019a) Mots: Multi-object tracking and segmentation. In *Proceeding CVPR* (pp. 7942–7951).

Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., Leibe, B. (2019b) Mots: Multi-object tracking and segmentation. In *Proceeding CVPR* (pp. 7942–7951).

Vondrick, C., Patterson, D., & Ramanan, D. (2013). Efficiently scaling up crowdsourced video annotation. *IJCV, 101*(1), 184–204.

Wang, X., & Gupta, A. (2015). Unsupervised learning of visual representations using videos. In *Proceeding ICCV* (pp. 2794–2802).

Wang, Y., Xu, Z., Wang, X., Shen, C., Cheng, B., Shen, H., Xia, H. (2021). End-to-end video instance segmentation with transformers. In *Proceedings CVPR* (pp. 8741–8750).

Wang, T., Han, B., & Collomosse, J. (2014). Touchcut: Fast image and video segmentation using single-touch interaction. *CVIU, 120,* 14–30.

Xu, Y., Wu, Y., binti Zuraimi, N. S., Nobuhara, S., Nishino, K. (2020). Video region annotation with sparse bounding boxes. In *31st British Machine Vision Conference 2020*, BMVC 2020, Virtual Event 2020, BMVA Press.

Yang, L., Fan, Y., & Xu, N. (2019a). Video instance segmentation. In *Proceeeding ICCV* (pp. 5188–5197).

Yang, L., Fan, Y., & Xu, N. (2019b). Video instance segmentation. In *Proceeding ICCV* (pp. 5188–5197).

Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., Darrell. T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceeding CVPR* (pp. 2636–2645).

Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., & Torralba, A. (2019). Semantic understanding of scenes through the ade20k dataset. *IJCV, 127*(3), 302–321.