2023

Generalized framework for image and video object segmentation using affinity learning and message passing GNNS

Sundaram Muthu

Ruwan Tennakoon

Tharindu Rathnayake

Reza Hoseinnezhad

David Suter
Edith Cowan University

See next page for additional authors

10.1016/j.cviu.2023.103812
Muthu, S., Tennakoon, R., Rathnayake, T., Hoseinnezhad, R., Suter, D., & Bab-Hadiashar, A. (2023). Generalized framework for image and video object segmentation using affinity learning and message passing GNNS. Computer Vision and Image Understanding, 236, article 103812. https://doi.org/10.1016/j.cviu.2023.103812

Computer Vision and Image Understanding 236 (2023) 103812

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Generalized framework for image and video object segmentation using affinity learning and message passing GNNS

Sundaram Muthu [a,*], Ruwan Tennakoon [b], Tharindu Rathnayake [a], Reza Hoseinnezhad [a], David Suter [c], Alireza Bab-Hadiashar [a]

[a] School of Engineering, RMIT University, Melbourne 3000, Australia
[b] School of Science, RMIT University, Melbourne 3000, Australia
[c] School of Science, Edith Cowan University, Perth 6027, Australia

ARTICLE INFO

Communicated by Nikos Paragios

Keywords:
Unsupervised video object segmentation
Image segmentation
Lifted multi-cuts
Graph neural networks
Affinity learning

ABSTRACT

Despite significant amount of work reported in the computer vision literature, segmenting images or videos based on multiple cues such as objectness, texture and motion, is still a challenge. This is particularly true when the number of objects to be segmented is not known or there are objects that are not classified in the training data (unknown objects). A possible remedy to this problem is to utilize graph-based clustering techniques such as Correlation Clustering. It is known that using long range affinities (Lifted multicut), makes correlation clustering more accurate than using only adjacent affinities (Multicut). However, the former is computationally expensive and hard to use. In this paper, we introduce a new framework to perform image/motion segmentation using an affinity learning module and a Message Passing Graph Neural Network (MPGNN). The affinity learning module uses a permutation invariant affinity representation to overcome the multi-object problem. The paper shows, both theoretically and empirically, that the proposed MPGNN aggregates higher order information and thereby converts the Lifted Multicut Problem (LMP) to a Multicut Problem (MP), which is easier and faster to solve. Importantly, the proposed method can be generalized to deal with different clustering problems with the same MPGNN architecture. For instance, our method produces competitive results for single image segmentation (on BSDS dataset) as well as unsupervised video object segmentation (on DAVIS17 dataset), by only changing the feature extraction part. In addition, using an ablation study on the proposed MPGNN architecture, we show that the way we update the parameterized affinities directly contributes to the accuracy of the results.

1. Introduction

Appearance-based image segmentation as well as motion-based video object segmentation (VOS) are ubiquitous problems in computer vision. Single image segmentation involves grouping of pixels with similar appearances. Image segmentation is used in techniques such as object detection (Juneja et al., 2013) and semantic segmentation (Farabet et al., 2012). Meanwhile, video object segmentation involves partitioning a video into distinct segments based on different underlying motions and to cluster and track objects in a dynamic scene. Applications of VOS include autonomous driving (Lu et al., 2018), surveillance and tracking, virtual reality, activity recognition (Shao et al., 2018) and robotic navigation.

In both image and video segmentation problems, dealing with multiple objects still remains a challenge particularly when the number of objects are not known in advance and there are new and unknown objects at the time of inference (objects not classified during the training phase). Performing VOS is even more difficult because it has to deal with complex motions, deforming shapes, camouflaged objects, background clutter, occlusions and objects moving together. Some of these challenges have been demonstrated in Fig. 1(a-b). The VOS problem is particularly challenging when objects are not annotated (not even at the first frame). This is called zero shot or unsupervised VOS (Luiten et al., 2020).

Existing segmentation methods can be classified as (1) supervised object-based detection and tracking methods (Zulfikar et al., 2019; Luiten et al., 2018a; Perazzi et al., 2015) and (2) unsupervised methods (Shi and Malik, 2000; Arbelaez et al., 2010; Pont-Tuset et al., 2016). The supervised methods use state-of-the-art deep learning-based object detection on static images, and track those objects. These methods are often trained using ImageNet, and perform reasonably well if they are only applied to images with trained object classes. These models tend to overfit to a limited range of foreseen objects. For instance, the

* Corresponding author.
E-mail address: sundaram.muthu@data61.csiro.au (S. Muthu).

https://doi.org/10.1016/j.cviu.2023.103812
Received 28 April 2022; Received in revised form 24 March 2023; Accepted 18 August 2023
Available online 19 August 2023

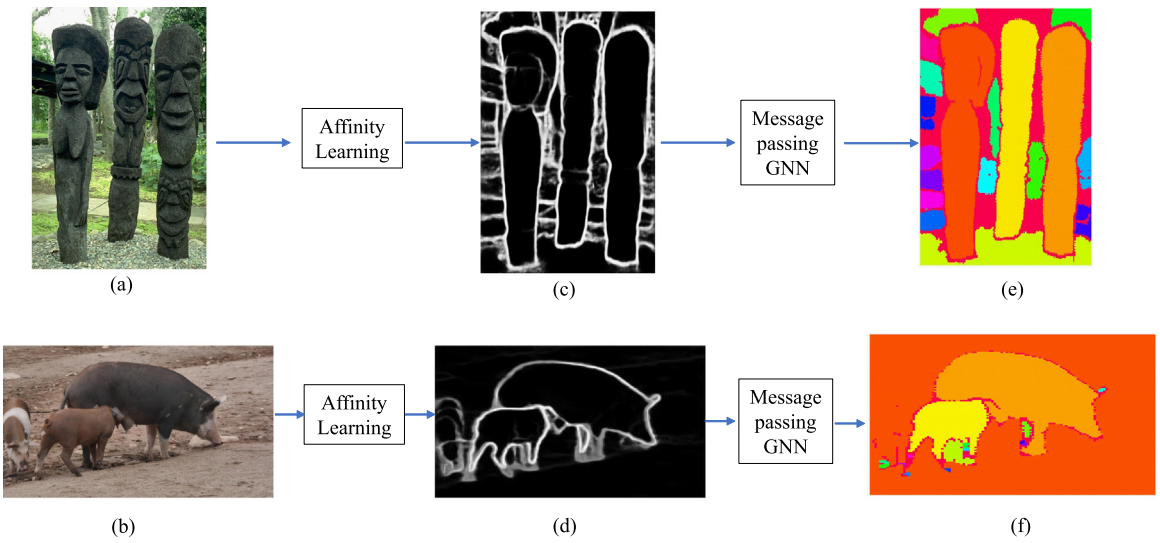

Fig. 1. Challenges involved in segmenting: (a) Unknown class of objects (Image segmentation task-BSDS dataset) (b) Multiple objects (UVOS task-DAVIS dataset) (c,d) One channel of the predicted graph or learnt affinities (Solves the *multiple previously unclassified objects* problem generally associated with supervised methods by predicting the affinities of neighbouring pixels belonging to the same object, instead of the predicting the object segmentation directly.) (e,f) Our segmentation results using our Message passing GNN (Solves the *time consuming process of obtaining the higher order affinities* generally associated with unsupervised methods to obtain accurate clustering by generating the higher order affinities in an efficient manner leading to faster solutions.).

supervised methods can create problems when applied to mobile robots and autonomous driving as such applications require solutions that can handle unknown objects and obstacles (not classified during training).

On the other hand, unsupervised methods generally group perceptually similar regions and use heuristic criteria to handle any appearances of unknown (not classified previously) objects. But there exists one approach that does not require any heuristics to decide the number of clusters (number of moving objects in the case of VOS), which is called correlation clustering or Multicut problem (MP) (Deza and Laurent, 2009). Keuper et al. (2015b) designed primal feasible heuristic algorithms to solve the NP-hard MP on a pixel grid graph. They also extend the MP to the Lifted Multicut problem (LMP) by adding long-range terms to increase the accuracy of clustering. Although the inclusion of long-range terms improves the performance, the approach is intractable for real time applications as the LMP 'lifting' (obtaining long range higher order information) and 'solving' steps do not scale up well when the problem size increases.

Our proposed method attempts to solve the problems associated with both supervised (i.e. existence of multiple unknown objects) and unsupervised (i.e. time-consuming process of obtaining long range higher order information for accurate clustering) methods. To overcome the problem of segmenting multiple objects, we propose an affinity learning module to predict a graph consisting of image pixels as nodes, and neighbourhood affinities as edge weights. The affinities describe the probability of neighbouring pixels belonging to the same segment. Edges connect every node to its four neighbouring nodes: Left, right, top and bottom nodes. As such, the edge weights form the four channels of the affinity matrix. Each channel individually looks like an edge map, which is visualized in Fig. 1(c-d). All four channels together form a graph that is clustered to obtain the segmentation results in challenging scenarios as shown in Fig. 1(e-f). This graph representation of the segmentation offers the following advantages:

- Permutation invariance: The representation is independent of the ordering of detected objects.
- Fixed-size: The size of the affinity matrix is fixed and does not vary with the number of objects in the scene. Constancy of the matrix size facilitates and improves the training capability of the method.

The learnt affinities are then clustered to obtain the segmentation using an MP, defined on a pixel grid graph. To resolve the issue of obtaining higher order long-range information (to improve clustering accuracy), we have designed an MPGNN module inspired by the graph neural network method presented in Gilmer et al. (2017). Similar to LMP (Keuper et al., 2015b), our MPGNN module also obtains higher order information. But our method generates the higher order information in an efficient manner as it converts the LMP to an MP, which is easier and faster to solve.

The significance of the proposed method stems from its ability to perform image/video segmentation of multiple unknown objects (using an affinity learning module) and aggregates higher order information for efficient clustering (using the MPGNN module). This is achieved by the following contributions:

- Introduction of a fixed-size permutation invariant representation for segmentation, using a neighbourhood affinity matrix that can handle multiple objects.
- Design of the MPGNN module that can aggregate the higher order information required for obtaining better segmentation accuracy. It also speeds up the clustering process by compressing the higher order information in an immediate neighbourhood affinity matrix, thereby converting an LMP to an MP.
- Achieving competitive performance on different segmentation problems with the same MPGNN module: Image segmentation (on BSDS dataset) and unsupervised video object segmentation (on DAVIS17 dataset).

The rest of this paper is organized as follows: Section 2 introduces the related work. Section 3 defines and motivates the problem and reasons for the usage of the GNN for obtaining higher order affinities. This section also describes the network architecture of our proposed affinity learning, and the MPGNN modules. Evaluation results on DAVIS17 and BSDS datasets are presented in Section 4: showing the effectiveness of the method to handle multiple moving objects as a result of using the neighbourhood segmentation affinities for both image and motion segmentation. The results also show the effect of using the MPGNN module to capture higher order affinities in an efficient manner leading to faster clustering. Section 5 concludes the paper and discusses future work. Appendix shows a mathematical proof that our MPGNN module is equivalent to simplifying the clustering problem without losing higher order information.

2. Related works

2.1. Image segmentation

Image segmentation methods can be categorized into unsupervised and supervised methods. Unsupervised methods include clustering methods: such as region merging (Zhu and Yuille, 1996), mean shift (Comaniciu and Meer, 2002), watershed segmentation (Najman and Schmitt, 1996), energy minimization and superpixel segmentation methods like SLIC superpixels (Achanta et al., 2012) that over-segment the image into consistent regions. In general, most unsupervised clustering methods are very efficient but require problem specific tuning, to calculate the number of segments.

Supervised contour learning methods, such as Holistically nested Edge Detection (HED) (Xie and Tu, 2015), learn multi-scale and multi-level hierarchical features for producing successively refined edges. The Richer Convolutional Features (RCF) method (Liu et al., 2017) extends the HED by combining all hierarchical levels of convolutional features, at all scales, to capture low level fine details as well as high level semantic details. While the above methods estimate only contours, the Convolutional Oriented Boundaries (COB) method (Maninis et al., 2017) estimates both contours and their orientations, which improves the segmentation accuracy. It should be noted that all these supervised methods detect edge maps and use complicated post-processing steps for converting edge maps to segmentation. Such operations include morphological operations (edge completion, thinning, applying snake movements), Multiscale Combinatorial Grouping (Pont-Tuset et al., 2016), or watershed segmentation (Najman and Schmitt, 1996). In contrast, our proposed method does not require a problem-specific tuning for deciding the number of clusters, and involves no post-processing operation.

2.2. Unsupervised video object segmentation

Existing VOS methods can be classified as semi-supervised, interactive, unsupervised, and self-supervised methods: based on the amount of human involvement. The semi-supervised methods or one-shot VOS methods (Caelles et al., 2017; Lai et al., 2020; Huang et al., 2020; Luiten et al., 2018b; Oh et al., 2019; Yang et al., 2020; Khoreva et al., 2019) require annotations of objects of interest in the first frame. The interactive methods (Miao et al., 2020) requires user interactions like scribbles, to guide and correct the segmentation. Scribble supervised VOS (Huang et al., 2021) uses scribble annotation to increase speed and reduce human labour during the annotation process. SPTFN (Zhang et al., 2018) jointly learn object localizing and segmentation in videos under weak supervision. Zhao et al. (2021) perform weakly supervised salient object detection on videos through fixation guided scribble annotations. The unsupervised methods (UVOS) (Garg and Goel, 2021; Lin et al., 2021; Zhou et al., 2021a; Song et al., 2018; Gowda et al., 2020; Wang et al., 2020, 2019a; Liu et al., 2018; Guo et al., 2017; Tokmakov et al., 2019; Chen et al., 2020) identify all moving objects in the scene, with no prior information about the number of objects or manual annotation of the first frame. The unsupervised terminology in this context refers to no human involvement during inference time but still requires supervision during the training. The recently introduced self-supervised methods (Yang et al., 2021) does not require supervision even during the training phase.

UVOS methods are generally flow-based and object-based. Flow-based methods use optic flow to generate long term point trajectories, and cluster them to obtain temporally a consistent segmentation (Brox and Malik, 2010). Tokmakov et al. (2017a) refined optic flow-based motion features, iteratively, in a fully convolutional network. Flow-based methods are commonly inaccurate at finding object boundaries (Tsai et al., 2016), and often fail when there are texture-less, camouflaged or fast-moving objects (Sun et al., 2014) or degenerate motions.

Advances in object recognition using deep learning techniques have enabled the object-based detection and tracking VOS methods to excel. Perazzi et al. (2015) used graphs to model connectivity between object proposals to perform tracking. Similarly, PReMVOS (Luiten et al., 2018a) was proposed to track and refine segmentations using Mask-RCNN (He et al., 2017). Yang et al. (2019) proposed a method that instantiates single-object trackers on detected object proposals, and merges the tracks to obtain accurate segmentation. In contrast to the above two groups, the methods presented in Tsai et al. (2016), Ranjan et al. (2019) and Cheng et al. (2017) utilize the concept of joint estimation of optic flow and VOS. AGNN (Wang et al., 2019a) uses message passing graph neural networks on a video graph to understand underlying pair-wise relationships to help the segmentation.

Recent high performance approaches in this area are mostly object-based methods. For instance, MaskRNN (Hu et al., 2017) and RVOS (Ventura et al., 2019) both use recurrent neural networks to learn spatial and temporal structure for tracking multiple objects. UnOVOST (Luiten et al., 2020) segments multiple objects from initial object proposals by applying Forest Path Cutting algorithm for merging short-term spatio-temporally similar tracklets to long-term object tracks. STEM (Athar et al., 2020) avoids the tracking step to associate objects temporally, by modelling videos as 3D spatio temporal volumes, using mixing functions. MATNet (Zhou et al., 2020) learns motion attentive appearance features, and captures deep interactions between both appearance and motion features.

These methods are however limited to the objects seen during training, and are time-consuming (due to multiple object proposal generation, association and tracking of these proposals (Cheng et al., 2017)). Different from AGNN (Wang et al., 2019a) that uses a video graph with each frame as a node, our proposed system uses an image graph with each pixel as a node to understand the pair-wise relationships between pixels to help the segmentation. Different from the methods described in Ranjan et al. (2019), Zhou et al. (2020) and Cheng et al. (2017), (which perform binary foreground/background segmentation), our proposed network segments all probable object(s), that are moving relative to the background, by clustering the learnt affinities.

2.3. Clustering methods

Clustering is an important part of unsupervised segmentation methods and we outline the three most common clustering methods that are used for image segmentation namely: Multiscale Combinatorial Grouping (MCG Pont-Tuset et al., 2016), Super-BPD (Wan et al., 2020) and Correlation Clustering (Deza and Laurent, 2009). MCG (Pont-Tuset et al., 2016) proposes a faster way to solve the normalized cuts algorithm by using a fast approximation of eigenvectors, and grouping multi-scale regions of the image by combinatorial searching. The Super-BPD method (Wan et al., 2020) has comparable performance to MCG (Pont-Tuset et al., 2016) in terms of accuracy, but it runs much faster. The Super-BPD uses robust direction features to aid in grouping of adjacent pixels, and to overcome the problem of weak boundaries. The limitation of this method relates to its poor performance for segmenting small regions.

Unlike MCG and Super-BPD, the Correlation clustering method (Deza and Laurent, 2009) finds the optimal number of clusters automatically. Image segmentation is initially posed as a Multicut Problem (MP) on a superpixel graph (Andres et al., 2013). Given the pixel grid graph of an image, and an affinity of incident vertices belonging to the same component(for every edge): MP outputs a simple binary labelling of the edges as either 'join or cut' to produce the final segmentation. However, the method is an NP-hard problem. Keuper et al. (2015b) use two heuristic algorithms called Greedy Additive Edge Contraction (GAEC), and an extension of the Kernighan–Lin (KLj) algorithm (Kernighan and Lin, 1970), to solve the MP, efficiently. The GAEC algorithm performs iterative agglomerative clustering, joining pairs of neighbouring clusters if doing so decreases the cost function.

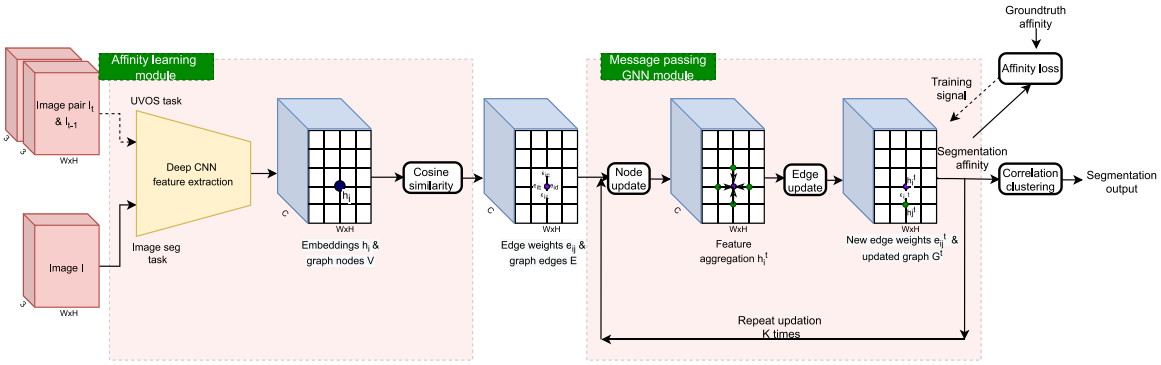

Fig. 2. Overview of the network architecture of our method. The first part is the affinity learning network that uses features extracted to generate segmentation affinity. The second part is the GNN network that uses message passing to aggregate higher order information and update the segmentation affinity learnt.

KLj performs iterative transformations to improve the current clustering by exchange of nodes between clusters or joining clusters. By including additional long-range terms using probabilistic geodesic lifting, the MP is then extended to the Lifted Multicut Problem(LMP) (Keuper et al., 2015b). The LMP performs better than the MP at the cost of increased computation. We will elaborate on this issue in the next section.

2.4. MP vs LMP correlation clustering

Existing correlation clustering methods for solving image segmentation can be categorized into two methods namely: The Multicut Problem (MP) that uses only local information, and the Lifted Multicut Problem (LMP) that uses higher order global information (Keuper et al., 2015b). In this section, we describe the relative advantages of solving the MP compared to solving the LMP, and explain the trade-off between accuracy and efficiency observed by increasing the amount of global information usage.

The graph Multicut Problem (MP) is defined as the problem of finding a unique decomposition of the graph which solves the constrained optimization defined in Keuper et al. (2015b). MP is defined on the pixel grid graph $G = (V, E, W)$. Vertices V are pixels and edges E connect the pixel to the four immediate neighbours. Edge weights W represents the cost at each edge, c_e, the probability of the two vertices of that edge belonging to distinct components. The output decomposition of the graph is to assign $0/1$ labels to all the edges. The edges are labelled such that the assignment minimizes the cost, and follows the Multicut cycle intersection constraint defined in Keuper et al. (2015b). Edges labelled 1 straddle distinct clusters. The NP-hard MP on a pixel grid graph can be solved efficiently using primal feasible heuristic algorithms designed in Keuper et al. (2015b), but the accuracy of clustering is poor due to the use of local immediate neighbourhood information only.

The Lifted Multicut Problem (LMP) is defined on a lifted graph $G' = (V, E \cup F, W)$ that includes additional edges F for all edges uv connecting non-neighbouring vertices in G within a specified search radius $1 < d_{uv} < d^*$. The additional edges add global higher order information to increase the accuracy of the clustering. The cost is the same as MP, and it is the probability of two vertices of an edge belonging to distinct components. The LMP introduces a set of additional linear inequality constraints defined in Keuper et al. (2015b) as a result of the higher order edges.

The cost for each edge in the LMP is calculated by a probabilistic geodesic lifting model (Keuper et al., 2015b). Including additional edges in the LMP improves the performance of clustering by including long range data: but the calculation of the cost is time consuming, as it involves searching for all paths in the graph between the two nodes (choosing a path with maximum probability). Our experiments, explained in Section 4, show that there is a trade-off between accuracy and efficiency in increasing the range of using additional data. This trade-off makes the LMP intractable for real time applications. Inspired by Perona et al. (1994) that makes use of a non-linear diffusion to aggregate neighbourhood information for better accuracy, our method effectively obtains the higher order affinities using Graph Neural Networks (GNNs). GNNs (Scarselli et al., 2008) are suited for non-Euclidean data (Sun et al., 2021), can handle multiple objects (with no particular order) and scale well for large problems. These networks are extended by Message Passing Neural Networks (MPNNs) (Gilmer et al., 2017) that use neural message passing to update nodes and edges jointly for a graph input. We utilize MPNNs as they can learn the underlying higher order grouping information that is required for efficient clustering (Ying et al., 2018; Lee et al., 2019).

To take advantage of this, we have designed a new MPNN module to aggregate long range higher order edge weights in a pixel grid graph. The aggregation process converts the LMP to an MP, which can be solved very efficiently while still maintaining the higher accuracy of the LMP. The effectiveness of our design in speeding up the clustering process is explained in Section 4.

3. Proposed method

Given an image I for solving the image segmentation task (or I_t and I_{t-1} for solving the VOS task as explained earlier), the proposed method produces a pixel-wise labelling S (or S_t for VOS) as the output. The overall structure of the proposed method is shown in Fig. 2. Our framework consists of three main steps: (1) Affinity learning module, (2) Message passing GNN module, and (3) Correlation Clustering.

3.1. Graph formulation

The pixel grid graph $G = (V, E, W)$ is created from an image I with size $w \times h$ as follows:

- Nodes V: Set of $N = w \times h$ vertices where each vertex i is represented by a high dimensional feature vector $h_i \in \mathbb{R}^c$.
- Edges E: Edges connecting four neighbouring nodes (left, right, top and bottom nodes) of every node that form the pixel grid. The edges e_{uv} connect neighbouring vertices in G that are within a specified search radius of $d_{uv} = 1$, noting that this radius represents MP.
- Weights W (also known as the initial segmentation affinity $A_0 \in \mathbb{R}^{N \times 4}$): consists of w_{uv} similarity scores for every edge defined in the graph. It is the cost associated with assigning the two nodes u and v of the edge e_{uv} to distinct components. The size of the segmentation affinity A_0 is fixed. As discussed earlier in Section 1, one of the main contributions of this work is the introduction of this fixed-size permutation invariant representation for segmentation. The size does not vary with the number of objects in the scene which enables the method to handle multiple objects. This also improves the generalizability of our model to handle multiple

tasks by just changing the feature extraction part. Even though we only use the fixed four channels for the four neighbouring nodes, there is no degradation of the performance due to the fact that our proposed MPGNN module captures the long range neighbourhood information via multiple node and edge updates as shown in Section 3.3.

The affinity learning module described in 3.2.1 is used to extract pixel features and assign edge weights A_0 for the pixel grid graph. The MPGNN module described in 3.2.2 is used to update the segmentation affinity A_k and generates the higher-order information required for segmentation. Finally, segmentation is performed by solving an optimization problem on a pixel grid graph that uses the updated segmentation affinities A_k as edge weights. This optimization is solved by the MP clustering algorithm described in 4.

3.2. Network architecture

In this section, we present the network architecture that defines the two main components of the network: The Affinity learning module and the MPGNN module.

3.2.1. Affinity learning module

The affinity learning network is designed to extract pixel features $h_i \in \mathbb{R}^c$, and to use those to generate an initial segmentation affinity $A_0 \in \mathbb{R}^{N\times 4}$ (a similarity score matrix for all edges between pixels that are adjacent to each other). The pixel features (as nodes) and initial segmentation affinity (as edge weights) form the pixel grid graph.

The proposed affinity learning module is depicted in Fig. 2. Since our method is a generalized framework that can perform both image segmentation and video object segmentation, the Deep CNN feature extraction block shown in Fig. 2 is task dependent. For image segmentation task, the feature extraction backbone is a VGG model that is pretrained on ImageNet (Liu et al., 2017). For the video object segmentation task, the feature extraction backbone consists of the convolutional blocks of the ResNet-101 in a two-stream architecture to capture both spatial and temporal information (Zhou et al., 2020)".

For the feature extraction task, we use the VGG network-based encoder of the RCF method (Liu et al., 2017) to generate features for image segmentation task. Initial segmentation affinity A_0 is obtained by calculating the correlations between the extracted features. At every pixel location i, the feature vector at that pixel h_i is correlated with the feature vectors h_j from the four immediate neighbouring pixels, using cosine similarity as the correlation function. The cosine similarity quantifies the orientation similarity between two high dimensional vectors. It also outputs a value between 0 and 1. Hence it best quantifies the similarity between the feature vectors of the neighbours generated by the feature extractor. The calculation of initial segmentation affinity A_0 does not increase the computational complexity as we only calculate the correlation between immediate neighbours at this stage. The proposed MPGNN module mentioned in the next Section 3.2.2 helps us to update the initial affinities and obtain the higher order long range information.

It is important to note that our method is applicable to different segmentation or clustering tasks by only replacing the feature extraction module. We use the MATnet's (Zhou et al., 2020) encoder to generate features for video object segmentation task. We should also mention that MATnet uses both the current image and the optic flow to extract features. In essence, it uses the motion information as an attention function for selecting and enhancing appearance features in specific regions of interest depending on the optic flow.

3.2.2. Message passing graph neural network module (MPGNN)

The proposed MPGNN module is depicted in Fig. 2. For the graph G, MPGNN module refines the initial segmentation affinity A_0 (containing only first order immediate proximity information), performs k repeated node-edge updates, and predicts an updated segmentation affinity A_k (containing higher order proximity information). It contains several iterations of successive node and edge update layers to update the node features and the edge weights, respectively.

During every iteration, the message that is passed inside the MPGNN module between neighbouring pixels consists of two important terms. The first term is the high dimensional feature vector $h_j \in \mathbb{R}^c$ of the neighbouring pixels. h_j is used in the node update layer in order to aggregate adjacent features through the *message function* $M(\cdot,\cdot)$ as described in Eq. (1). This aggregated message represents a global higher order proximity information of the neighbourhood as a result of the non-linear diffusion process occurring during the node update step.

The second term is the similarity scores w_{ij} between neighbouring pixels. w_{ij} indicates the importance of different neighbours. It is used as the self-attention component of the *message function* as described in Eq. (2). Since node update step modifies the features of all nodes due to message passing, the similarity scores w_{ij} should also be refined according to the updated node features. This refinement happens in the edge update layer as described in Eq. (6).

- **Node update layer:** For updating every node, the adjacent features are aggregated, embedded into a larger dimension for increased model capacity and stored as the updated hidden vector. Consider the features at vertices i and j at iteration t, denoted by h_i^t and h_j^t, respectively. The aggregated message at vertex i is then given by:

$$m_i^t = \sum_{j\in N(i)} M(h_i^t, h_j^t) \tag{1}$$

where $N(i)$ denotes all immediate neighbours of vertex i and $M(\cdot,\cdot)$ is the *message function*. The function can be implemented by using a linear or a non-linear mapping. Ablation studies discussed in Section 4.3, and Table 6, show and compare the results obtained using both types of node update functions including:

$$M(h_i^t, h_j^t) = h_j^t\, \alpha_i\, W_m$$

where W_m is a linear mapping that embeds the features into a larger dimension or nonlinear mapping:

$$M(h_i^t, h_j^t) = h_j^t\ \mathrm{MLP}(h_j^t \alpha_i; \Omega_M)$$

where

$$\alpha_i = \textsc{SoftMax}\left(\{w_{ij}^t\}_{j\in N(i)}, i\right)$$

in which the edge weights w_{ij} are elements of the initial affinity matrix A^{t-1} that defines the importance of node j for updating node i:

$$w_{ij}^t = A(h_i^t, h_j^t) \tag{2}$$

and the well-known $\textsc{SoftMax}(\mathbf{w}, i)$ function inputs a vector $\mathbf{w}$ and returns a soft maximum for its ith element, $w(i)$, as defined below (which is also normalized):

$$\textsc{SoftMax}(\mathbf{w}, i) \triangleq \frac{\exp(w(i))}{\sum_{w\in\mathbf{w}} \exp(w)}. \tag{3}$$

In addition, the function $\mathrm{MLP}(\cdot, \Omega_M)$ is a 1-to −1 nonlinear mapping implemented via training a multi-layer perceptron neural network with weights Ω_M.

Self attention in the message passing step is achieved by the message function M, which generates a weighted sum of the neighbouring messages. The attention weights are calculated by the function $\textsc{SoftMax}\left(\{w_{ij}^t\}_{j\in N(i)}, i\right)$ that assigns different importance for different neighbours, depending on the feature similarity obtained from the initial affinity matrix. It also normalizes the affinity matrix along the row dimension to ensure that the sum of the contributions of all neighbours is 1. During the aggregation

step, residual connection (own message) is also included as a self loop for every node with edge weights $w_{ij} = 1$.
Aggregate messages are directly used as the updated node feature by implementing an identity node update function:

$$h_i^{t+1} = m_i^t. \tag{4}$$

- **Edge update layer:** To update the initial affinity matrix A_0, updated node features are fed to the edge update layer. It transforms the affinity matrix A_0 to A_k by applying an edge update function for every pair of nodes i and j belonging to the edge e_{ij} of the graph G.
The edge update function can be implemented by either using a simple (cosine similarity) or a sophisticated edge update (MLP on concatenated features). The cosine similarity does not depend on the magnitude of the vectors. It always outputs a value between 0 and 1 which can be easily interpreted as the similarity between the neighbours. The edge update through the cosine similarity also has low computational complexity. Ablation studies discussed in Section 4.3, and Table 6, show the comparative results obtained using both types of edge update functions:

$$A^t\left(h_i^t, h_j^t\right) = \textsc{Cosine\_Sim}(h_i^t, h_j^t) \tag{5}$$

or

$$A^t\left(h_i^t, h_j^t\right) = \textsc{Sigmoid}\left[\mathrm{FC}_1\left(\textsc{ReLU}\left(\mathrm{FC}_2\left(\textsc{ReLU}\left(\mathrm{FC}_3(\textsc{Concat}[h_i^t, h_j^t])\right)\right)\right)\right)\right] \tag{6}$$

where Cosine_Sim is cosine similarity function, and FC_1, FC_2 and FC_3 are nonlinear mappings implemented by fully connected networks (their trained weights are omitted for the sake of notation simplicity), and ReLU and Sigmoid are nonlinear operations returned by the well-know ReLU and Sigmoid activation functions.
Multiple iterations of node and edge updates lead to aggregation of information from distant nodes as information propagates across the edges of graphs. The segmentation accuracy increases with the increase in the number of iterations of node and edge updates. The number of iterations is chosen such that a correct balance between accuracy and efficiency is achieved as described in Section 4. Different from the usual MPNN, the outcome here is the final updated edge segmentation affinities (that form the edge weights of an image grid graph and are clustered to obtain the segmentation by solving an instance of MP as described in Keuper et al., 2015b).

3.3. Why GNN is used to learn higher order affinities?

As it was explained earlier, the MP has only the first order proximity information (pixel grid graph) whereas the LMP has higher order proximity information (pixel grid graph with additional lifted edges). In Appendix, we show that using our MPGNN module (described above) is equivalent to converting the LMP to MP without losing its higher order proximity information.

Let us consider a 1D simplified version of our pixel grid graph as shown in Fig. 3(a). The adjacency matrix A^0 in Fig. 3(b) contains the one hop neighbourhood information of the pixel grid graph. The powers of adjacency matrix $(A^0)^k$ contains k hop neighbourhood information as shown in Fig. 3(c). Similarly, the Graph Laplacian $L = D - A$ in Fig. 3(d) shows how smoothly an energy potential kept on a specific node diffuses to the immediate neighbours, but L^2 in Fig. 3(e) shows the two hop energy diffusion. Fig. 3(c,e) captures the long range information. Similarly, in Appendix, we show that our proposed MPGNN also captures the long range neighbourhood information via multiple node and edge updates. To prove that the updated affinity of our MPGNN module captures the same long range information, we show that both the kth power of adjacency matrix $(A^0)^k$ as well as the output of our MPGNN module A^k contain the k hop higher order neighbourhood information.

3.4. Loss term

For the task of image segmentation and unsupervised video object segmentation, we use labelled ground-truth segmentation for calculating loss during the supervised training phase. As discussed in Section 2.2, it is to be noted the term 'unsupervised' in the 'unsupervised video object segmentation' task refers to having no human supervision during inference (annotation of objects of interest in the first frame of every video). It still requires supervision during training (per frame labelled ground-truth information).

To define a loss term, we first need to convert the labelled ground-truth segmentation to our segmentation affinity representation A, for every edge connecting only adjacent nodes in the pixel grid graph. The ground-truth affinity is 1 if the label of the pixel and its neighbour are the same. Consider an entry in the training dataset, composed of a training image I, and its ground-truth label matrix of the same size, L. The ground-truth affinity matrix for this entry is then defined for each node u in the graph (corresponding to a pixel in the image, i.e. $u \in I$) and one if its neighbouring pixels $v \in N(u)$, as follows:

$$A_{uv} = \begin{cases} 1, & \text{if } L(u) = L(v), \\ 0. & \text{otherwise,} \end{cases} \tag{7}$$

If the affinity matrix returned by our solution is $\hat{A}$, then the associate loss is defined as the following mean square error (MSE):

$$L(A, \hat{A}) = \frac{1}{\alpha_e} \sum_{u \in I} \sum_{v \in N(u)} \left(1 - A_{uv}\right) \times \left(A_{uv} - \hat{A}_{uv}\right)^2 + \frac{\beta}{\alpha_{ne}} \sum_{u \in I} \sum_{v \in N(u)} A_{uv} \times \left(A_{uv} - \hat{A}_{uv}\right)^2. \tag{8}$$

The first term is the segmentation loss for true edges in the segmentation affinity matrix (if the labels of the compared pixels do not match each other) and the second term is the segmentation loss for non-edges in the segmentation affinity matrix (if labels of the compared pixels match each other). The terms α_e and α_{ne} are included for the sake of normalization; they are the number of the true edges and non-edges in A, respectively.

$$\alpha_e = \sum_{u \in I} \sum_{v \in N(u)} \left(1 - A_{uv}\right) \quad \text{and} \quad \alpha_{ne} = \sum_{u \in I} \sum_{v \in N(u)} A_{uv}$$

and the coefficient β is the ratio of importance of non-edge pixels to edge pixels chosen as discussed in Section 4.3.

The purpose of splitting the loss into two parts is to overcome the imbalance problem in the ground-truth as the number of true edges is commonly far less than the number of non-edges in the ground-truth segmentation affinity matrix (number of 0's $\ll$ number of 1's in A).

To speed up the training convergence, a multi-stage affinity loss term is used to penalize the updated affinities errors after every update of the segmentation affinity matrix.

4. Experimental results

In this section, we evaluate the proposed method on the two segmentation tasks using openly available datasets: (1) The BSDS dataset (Arbelaez et al., 2010) for the static image segmentation, and (2) The DAVIS-2017 (Pont-Tuset et al., 2017) dataset and FBMS dataset (Brox and Malik, 2010) for the Unsupervised Video Object Segmentation (UVOS). We show that the MPGNN module speeds up the clustering, as we hypothesized earlier. Then, we also compare with the state-of-the-art methods, and also perform ablation studies (to understand the main advantages of specific components of our model). We implemented our method in Pytorch and trained the network with a batch size of 2, learning rate of 10^{-5} using the ADAM optimizer. The images are scaled to 384×512 pixels and no data augmentation was performed.

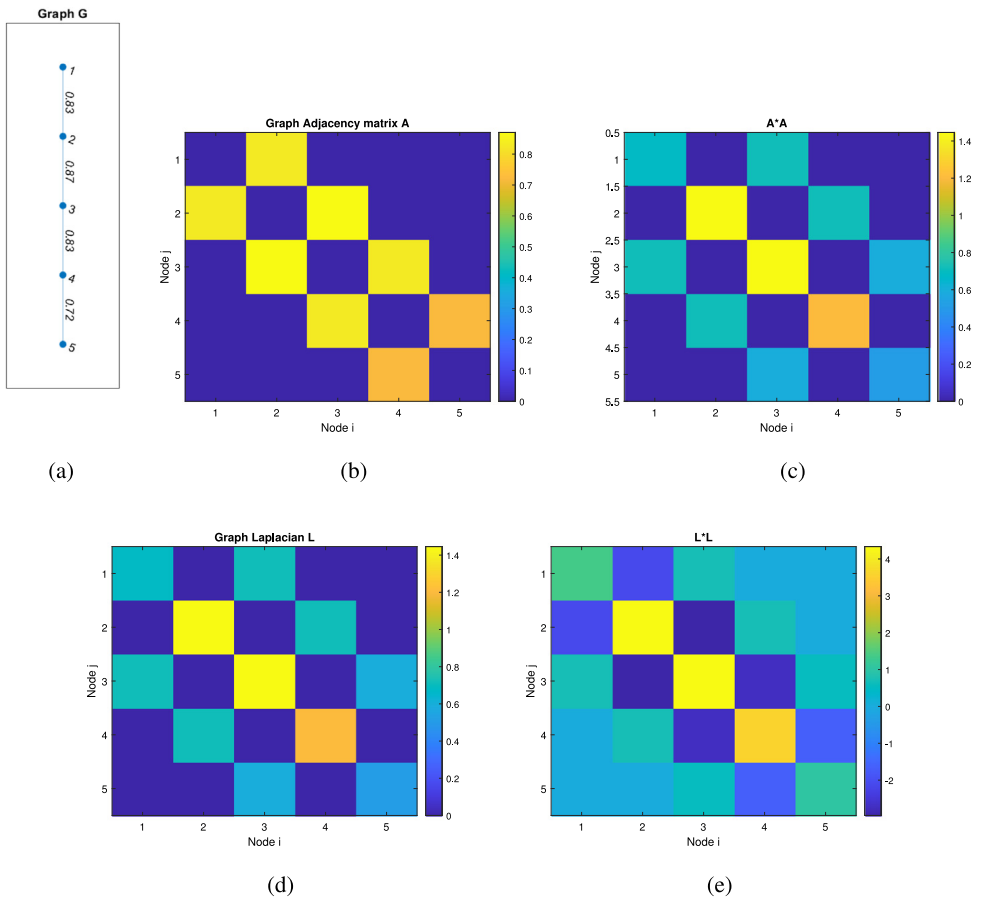

Fig. 3. Higher order affinities : (a) 1D grid graph, (b) Adjacency matrix A, (c) A^2, (d) Graph Laplacian L, (e) L^2.

4.1. Evaluation results: Image segmentation

BSDS-500 benchmark (Arbelaez et al., 2010) for the image segmentation and boundary detection tasks contains 200 training, 100 validation, and 200 test images. Each image has 4 to 10 ground-truth segmentations annotated by humans. We use the following measures provided by Arbelaez et al. (2010) for evaluating our method:

- F-measure (F_b for boundaries)
- Variation of Information (VI)
- Probabilistic Rand Index (RI)
- Segmentation covering ($Covering$).

4.1.1. Hypothesis testing (MP/LMP/proposed) for image segmentation

In order to study the time-accuracy trade off between the MP and the LMP instances for every image in the BSDS-500 benchmark, we initially defined MP and LMP instances (with different neighbourhood sizes) on the output of our affinity learning module without using the MPGNN module. We then solved the MP and LMP with GAEC and KLj heuristic solvers described in Keuper et al. (2015b). The MP works on a pixel grid graph (edges connected to adjacent nodes only) whereas, the LMP includes additional edges between non-neighbouring nodes of the pixel grid graph. The LMP had better performance due to the objective function containing the additional cost associated with wrongly associating distant nodes (assignments to either same or distinct segments). The LMP was also more accurate as the affinity estimation for distant pixels were also more reliable. However, the LMP took several minutes per image to converge (Kardoost and Keuper, 2018). The overall time accuracy trade off between the MP and the LMP, while increasing neighbourhood size, is reported in Table 1.

To examine the effect of adding the MPGNN, we defined an instance of the MP on updated learnt affinity prediction (the output of the MPGNN module) and observed that solving this MP performed better in terms of both time and accuracy compared to the LMP instance without the MPGNN step. Table 1 demonstrates the quantitative results of our model. The results, inline with our theoretical analysis presented in Appendix, show that the MPGNN with node and edge updates is capable of successfully aggregating long range information. In fact, this shows that solving a simple graph by the MP with MPGNN module produces similar or better results compared to time consuming solution of the LMP: Speeding up the clustering without loss of accuracy. In our experiments, we used $k = 5$ but the approximation is valid for any number of passes (more iterations) leading to better performances with increased model complexity.

Table 1
Image segmentation task on BSDS dataset (Arbelaez et al., 2010): Time accuracy trade off - MP vs LMP and approximating LMP as MP by including our MPGNN module.

Method	Covering↑	RI↑	VI↓	Bdy F↑	Timing (ms)↓
MP	0.36	0.57	3.43	0.45	**306**
LMP3	0.49	0.79	4.35	0.41	690
LMP5	0.51	0.8	4.88	0.39	1038
ours	**0.61**	**0.81**	**1.83**	**0.75**	373

4.1.2. Comparative results

The performance of the proposed method is also compared with several related state-of-the-art approaches in Table 2 including: (1) Multiscale Combinatorial Grouping- MCG (Pont-Tuset et al., 2016), (2) Convolutional Oriented Boundaries- COB (Maninis et al., 2017), (3) Super-BPD (Wan et al., 2020), (4) Probabilistic geodesic lifting LMP (Keuper et al., 2015b) methods that have also been discussed in Section 2.

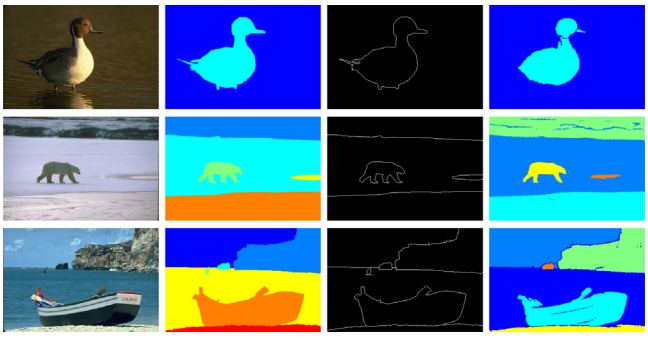

Fig. 4. Image segmentation task qualitative results on BSDS dataset (Arbelaez et al., 2010). From left to right: input, one of the ground-truth segmentation, ground-truth edge map, and segmentation results from our method.

Table 2
Volumetric and Boundary evaluation of Image segmentation results on BSDS dataset (Arbelaez et al., 2010).

Method	Covering↑	RI↑	VI↓	Bdy F↑
Humans	0.69	0.86	1.18	0.73
SLIC	0.37	0.74	2.56	0.52
MCG (Pont-Tuset et al., 2016)	0.61	0.83	1.5	0.74
COB (Maninis et al., 2017)	**0.66**	**0.85**	**1.3**	**0.78**
Super-BPD (Wan et al., 2020)	0.58	0.82	1.74	0.69
LMP5	0.51	0.8	4.88	0.39
ours	0.61	0.81	1.83	0.75

Table 2 shows that, while our method has comparable performance to other competing methods, it is not the highest performing method. There are two main advantages of our method compared to the supervised contour learning methods (Pont-Tuset et al., 2016; Maninis et al., 2017) that performs better than our proposed method. Firstly, the proposed method does not rely on prior knowledge of the segmented objects (as we only use affinities to perform segmentation). This makes our method to be applicable to unseen test videos (videos that contain multiple objects not classified during the training phase as shown in Fig. 5). Next, our method has more generalization capability both across different tasks (Segmentation in images and videos), and across different datasets or domains for the same task (segmentation in videos: DAVIS and FBMS datasets). This makes our method transferable across different tasks by only changing the feature extraction part. Our method also does not require any problem-specific tuning for deciding the number of clusters and involves no post-processing operation. Fig. 4 shows some qualitative results of our method on some challenging single image segmentation tasks on the BSDS dataset (Arbelaez et al., 2010).

4.2. Evaluation results: Unsupervised video object segmentation

We use the DAVIS benchmark (Pont-Tuset et al., 2017) to evaluate the performance of our method for Video Object Segmentation. We also evaluate the performance of our method on the FBMS benchmark (Brox and Malik, 2010) to show that our method generalized across other datasets. We perform the task of *unsupervised* video object segmentation while focusing on the segmentation of *multiple* moving objects in a scene. Unsupervised in this context refers to having no level of human involvement at inference time, which is different from semisupervised methods that require annotation of objects of interest in the first frame of every video. DAVIS contains 60 videos for training and validation, and 30 challenging test videos. FBMS dataset contains 59 sequences with sparse annotations of 720 frames and contains 30 challenging test sequences. We use the following measures provided by DAVIS challenge (Pont-Tuset et al., 2017) to evaluate our method:

Table 3
UVOS task on DAVIS dataset (Pont-Tuset et al., 2017): Time accuracy trade off - MP vs LMP and approximating LMP as MP by including our MPGNN module.

Method	J&F↑	J mean↑	F mean↑	Fdecay ↓	Timing (ms)↓
MP	0.229	0.217	0.241	0.022	143
LMP3	0.254	0.169	0.338	**0.02**	331
ours	**0.436**	**0.416**	**0.456**	0.032	**74**

- Region similarity J: It is the intersection-over-union (IoU) of the predicted segmentation $\hat{S}$ and ground-truth segmentation S.

$$J = \frac{S \cap \hat{S}}{S \cup \hat{S}} \tag{9}$$

- Contour accuracy F: It is the F-measure for the contour points of the predicted segmentation $\hat{S}$ compared to the contour points of the ground-truth S.

$$F = \frac{2PR}{P + R} \tag{10}$$

where P is the contour precision and R is the contour recall.
- Temporal instability F decay: It is a measure of the label inconsistencies across the frames in a video. The exact definition and how to calculate that is explained in Pont-Tuset et al. (2017).

4.2.1. Hypothesis testing (MP/LMP/proposed) for UVOS

Similar to Section 4.1.1, we observed the same time accuracy trade off when solving MP and LMP instances (different neighbourhood sizes) using (Keuper et al., 2015b) for segmenting all moving objects in DAVIS dataset (Pont-Tuset et al., 2017). For implementing this, we only needed to replace the image features by MATNet encoder features that encapsulate both appearance and motion. We applied cosine similarity between adjacent nodes to obtain edge weights for defining the MP while computing long range affinities to obtain additional edge weights for defining the LMP.

Table 3 demonstrates quantitatively that our method, by just solving the MP with MPGNN module, achieves faster clustering with better accuracy compared to the LMP. In terms of time, the prediction of segmentation affinities took only 173 ms/frame on a TitanX GPU with 12 GB memory. Solving the MP clustering took 72 ms/frame and has better accuracy than the LMP that took almost five times as much time (331 ms/frame).

4.2.2. Comparative results: DAVIS17

The performance of the proposed method is also compared with several related state-of-the-art approaches including: (1) UnOVOST (Luiten et al., 2020), (2) RVOS (Ventura et al., 2019), (3) MATnet (Zhou et al., 2020) and (4) Probabilistic geodesic lifting LMP (Keuper et al., 2015b) methods. These methods are selected as they are the top performing methods that (similar to our method) do not use additional training data. We had earlier discussed the operation of methods (in Section 2) and their performance results are shown in Table 4. Fig. 5 shows some qualitative results of our method on some challenging single and multiple object sequences of the DAVIS17 dataset (Pont-Tuset et al., 2017).

As shown in Table 4, our method does not require object mask proposals at the starting stage of the method, and heuristic post-processing at the last stage of the method. Unlike MATnet (Zhou et al., 2020), which uses CRF based problem specific heuristic post-processing to convert the binary foreground/background segmentation, to multi-object segmentation, and UnOVOST (Luiten et al., 2020) and RVOS (Ventura et al., 2019) that use object based mask proposals, which only work well for limited number of categories in the training data: our method can segment all probable moving object(s).

As shown in Fig. 5, the proposed method can segment objects with accurate and sharp boundaries. The higher boundary accuracy can be attributed to the iterative refinement of edge affinities. This is different

Table 4
J&F metrics for UVOS task on DAVIS dataset (Pont-Tuset et al., 2017). Our method does not require problem specific heuristic post-processing. Our method also does not use any object mask proposals for initialization of the method.

Method	J&F↑	Jmean↑	F mean↑	Fdecay ↓	Heuristics	Object masks
UnOVOST (Luiten et al., 2020)	**0.679**	**0.664**	**0.693**	**0.01**	Yes	Yes
MATNet (Zhou et al., 2020)	0.586	0.567	0.604	1.8	Yes	No
RVOS (Ventura et al., 2019)	0.412	0.368	0.457	1.7	No	Yes
LMP3	0.254	0.169	0.338	0.02	No	No
ours	0.436	0.416	0.456	0.032	No	No

Table 5
Region similarity metric J for UVOS task on FBMS dataset (Brox and Malik, 2010).

Method	SFL(Keuper et al., 2015a)	FSEG(Jain et al., 2017)	LVO(Tokmakov et al., 2017b)	ARP(Koh and Kim, 2017)	PDB(Song et al., 2018)
J mean↑	56.0	68.4	65.1	59.8	74.0
Method	MATNet(Zhou et al., 2020)	AGS(Wang et al., 2019b)	COSNet(Lu et al., 2019)	FEMNet(Zhou et al., 2021b)	Ours
J mean↑	76.1	76.0	75.6	**78.5**	74.8

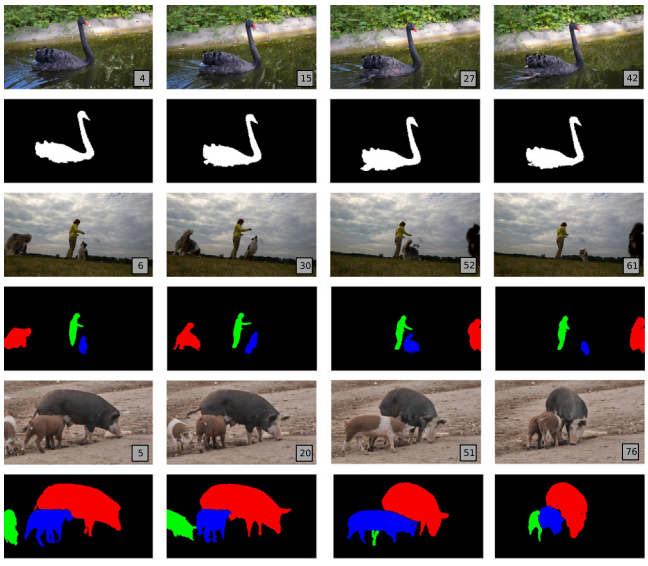

Fig. 5. UVOS task qualitative results on DAVIS17 dataset (Pont-Tuset et al., 2017). From top to bottom: *blackswan* single object sequence input and visual segmentation results, *dogs-jump* multiple object sequence input and visual segmentation results, *pigs* multiple object sequence input and visual segmentation results.

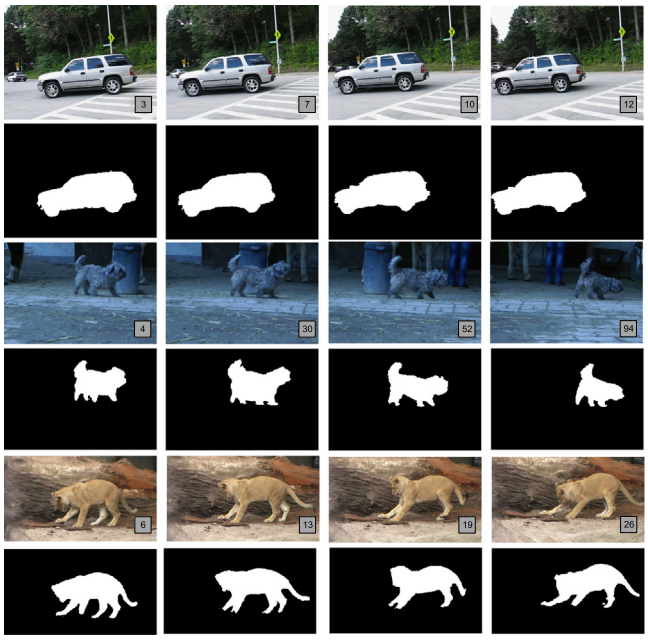

Fig. 6. UVOS task qualitative results on FBMS dataset (Brox and Malik, 2010). From top to bottom: *cars1*, object sequence input and visual segmentation results, *dogs01* object sequence input and visual segmentation results, *lion01* object sequence input and visual segmentation results.

from the conventional CNNs that can also have large receptive fields to capture long range information but tend to produce inaccurate object boundaries due to smooth gradual change of extracted features. Also, the proposed method can be applied to a much broader set of problems, like image segmentation, as was discussed in the previous section.

4.2.3. Comparative results: FBMS

The performance of the proposed method on the FBMS test sequences is compared with several related state-of-the-art approaches in Table 5. As shown in Table 5, we produce competitive results on the region similarity metric. Our method does not suffer from the sparseness of the ground-truth annotations compared to methods like LVO (Tokmakov et al., 2017b) & ARP (Koh and Kim, 2017). Also, the proposed method generalizes well to multiple tasks (Image and video object segmentation), and several datasets for the same task (DAVIS17 and FBMS dataset). Fig. 6 shows some qualitative results of our method on some challenging sequences. As shown in Fig. 6, the proposed method can segment objects with accurate and sharp boundaries due to the iterative refinement of edge affinities as discussed in the previous section.

Table 6
Results of increasing complexity of node and edge update components of our proposed MPGNN module on BSDS dataset (Arbelaez et al., 2010).

Node Update	Edge Update	Cov.↑	RI↑	VI↓	Bdy F↑
Linear	Cosine similarity	0.48	0.78	4.35	0.44
Non-Linear	Cosine similarity	0.48	0.78	4.08	0.48
Linear	Learned similarity	0.56	0.81	3.91	0.47
Non-Linear	Learned similarity	**0.59**	**0.81**	**3.04**	**0.56**

4.3. Ablation study

4.3.1. Complexity of GNN layers

As shown in Table 6, we perform an ablation study to measure the contribution of different types of node and edge updates (described in Section 3) towards improving the segmentation accuracy. We start with a simple model for our MPGNN module by using a linear feature transformation for node aggregation and cosine similarity for the edge updates, to obtain the edge weights of the pixel grid graph.

We then use a MLP as a non-linear node aggregation function and observe small improvement of around (9%) in the boundary metric

Table 7
Results of number of iterations of node and edge updates on BSDS dataset (Arbelaez et al., 2010).

Method	Covering↑	RI↑	VI↓	Bdy F↑
$k = 3$	0.55	0.76	**2.29**	**0.69**
$k = 5$	0.58	0.80	2.49	0.65
$k = 7$	**0.59**	**0.83**	3.32	0.54

Table 8
Results of ensembling models with better volumetric segmentation accuracy and models with better boundary segmentation on BSDS dataset (Arbelaez et al., 2010).

Method	Covering↑	RI↑	VI↓	Bdy F↑
$\beta = 1$	0.58	0.80	3.04	0.56
$\beta = 3$	0.52	0.71	2.53	0.63
$\beta = 5$	0.45	0.61	**2.48**	**0.65**
Ensembled Model	**0.60**	**0.81**	2.78	0.57

only (and not in the volumetric accuracy). Adding a more sophisticated edge update with learnable parameters leads to improvements in boundary (7%) as well as volumetric accuracy (15%) measures. Finally, combining the non-linear node update and learnable edge update yields significant improvements in both boundary (31%) and volumetric accuracy (26%) measures.

4.3.2. Number of iterations

The message passing graph neural network module explained in Section 3.2.2 includes a hyperparameter k, which is the number of node-edge updates happening within the MPGNN module. The role of this parameter is to perform multiple iterations in order to aggregate distant node information. As shown in Table 7, we observed that increasing the value of k leads to improved performance in terms of volumetric accuracy measures at the cost of increased model complexity. Also, the boundary metrics decreases as we include more distant node information. We choose $k = 5$ to obtain the correct balance between accuracy across all metrics.

4.3.3. Model ensembling

The loss term in Eq. (8) includes a hyperparameter β, which is the ratio of importance of non-edge pixels to edge pixels. The role of this parameter is to overcome imbalance: as there are more edges connecting pixels of the same object compared to edges connecting pixels of different objects. As shown in Table 8, we observed that increasing the value of β leads to over segmentation (objects are split into multiple parts), which improves the boundary measure but detracts from the volumetric accuracy measure.

To obtain the best of both worlds and improving both measures simultaneously, we trained three different models with different β values (1, 3 and 5) and used the average of their segmentation affinities for defining the edge weights of the pixel grid graph during inference.

5. Conclusions

In this paper, we introduced a novel framework that addresses two important data segmentation problems by using the affinity representation and a message passing graph neural network. The proposed use of affinity representation solves the multi-object segmentation problem. The MPGNN module solves the time-consuming LMP clustering problem by speeding up the clustering process. As shown by theoretical analysis as well as experimentation, this speed-up is achieved by aggregating long range information, capturing non-linear higher order proximity information, and compressing the information into simple pixel grid graph Multicut problem, which can be solved efficiently.

Our experiments on DAVIS17 dataset for motion segmentation and BSDS dataset for image segmentation show that our method is transferable across different tasks by only changing the feature extraction part. Additionally, our method improves the generalization capability by leveraging the benefits of both learning and unsupervised methods. A drawback of our method is that it fails to segment objects temporarily stopping for a few frames in the video. This occurs since we process each frame independently losing valuable temporal information. So, we plan to extend the work further by using tracking as a prior to update the segmentation affinity prediction of our network. Another area for improvement is to make use of GNN's to incorporate the clustering and make our method to predict the segmentation directly in an end-to-end manner.

CRediT authorship contribution statement

Sundaram Muthu: Conceptualization, Methodology, Validation, Writing – original draft. **Ruwan Tennakoon:** Conceptualization, Investigation, Writing – review & editing, Supervision. **Tharindu Rathnayake:** Software, Data curation, Writing – review & editing. **Reza Hoseinnezhad:** Supervision. **David Suter:** Writing – review & editing. **Alireza Bab-Hadiashar:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

This work was supported by the Australian Research Council through an ARC Linkage Project grant (LP160100662).

Appendix. Our MPGNN module captures higher order affinities: Proof

Let us start by the initial affinity matrix given by Eq. (A.1) in Box I.

For node i, the ith row of the adjacency matrix is:

$$A^0_{\text{node } i} = \begin{bmatrix} \mathbf{0}_{i-2} & a_{i,i-1} & 1 & a_{i,i+1} & \mathbf{0}_{n-i-1} \end{bmatrix} \tag{A.2}$$

where $\mathbf{0}_n$ means an n-dimensional row vector of all zeros.

The powers of adjacency matrix $(A^0)^k$ contains k hop neighbourhood information as shown in Fig. 3. Similarly, the Graph Laplacian $L = D - A$ shows how smoothly an energy potential kept on a specific node diffuses to the immediate neighbours but L^2 shows the 2 hop energy diffusion. For node i, the ith row of $(A^0)^2$ is given by:

$$\begin{aligned} \left(A^0\right)^2_{\text{node } i} \quad = \quad & a_{i,i-1} \begin{bmatrix} \mathbf{0}_{i-3} & a_{i-1,i-2} & 1 & a_{i-1,i} & \mathbf{0}_{n-i} \end{bmatrix} + \\ & \begin{bmatrix} \mathbf{0}_{i-2} & a_{i,i-1} & 1 & a_{i,i+1} & \mathbf{0}_{n-i-1} \end{bmatrix} + \\ & a_{i,i+1} \begin{bmatrix} \mathbf{0}_{i-1} & a_{i+1,i} & 1 & a_{i+1,i+2} & \mathbf{0}_{n-i-2} \end{bmatrix}. \end{aligned} \tag{A.3}$$

Our proposed MPGNN, described in Section 3, captures the long range neighbourhood information via multiple node and edge updates. To prove that the updated affinity of our MPGNN module captures the same long range information, we show that A^1 has the same coefficients as in $(A^0)^2$.

The node update of MPGNN module converts node features F^0 to aggregated features F^1 using A^0 as self attention weights. The MPGNN module is described in detail in Section 3. The above conversion leads to

$$\begin{aligned} \underbrace{F_1}_{n\times c} \quad &= \quad \underbrace{\textsc{SoftMax}\left(A^0\right)}_{n\times n} \times \underbrace{F^0}_{n\times c} \times \underbrace{W^1}_{c\times c} \\ &= \quad \underbrace{\bar{A}^0}_{n\times n} \times \underbrace{\bar{F}^0}_{n\times c}. \end{aligned} \tag{A.4}$$

$$A^0_{(n\times n)} = \begin{pmatrix} 1 & a_{1,2} & 0 & & & & & & & & \\ a_{2,1} & 1 & a_{2,3} & & & & \cdots & & & & \\ & & & a_{i-1,i-2} & 1 & a_{i-1,i} & 0 & 0 & & & \\ & \vdots & & \ddots & 0 & 1 & a_{i,i+1} & 0 & \ddots & & \vdots \\ & & & & 0 & a_{i+1,i} & 1 & a_{i+1,i+2} & & & \\ & & & & & & \cdots & & & 1 & a_{n-1,n} \\ & & & & & & & & & a_{n,n-1} & 1 \end{pmatrix} \tag{A.1}$$

Box I.

For the ith node we have:

$$\begin{aligned} \underbrace{F^1_{\text{node}\,i}}_{1\times c} &= \underbrace{\left[\mathbf{0}_{i-2} \;\; \bar{a}_{i,i-1} \;\; 1 \;\; \bar{a}_{i,i+1}\mathbf{0}_{n-i-1}\right]}_{1\times n} \times \underbrace{\bar{F}^0}_{n\times c} \\ &= \bar{a}_{i,i-1}\,\bar{F}^0_{i-1} + \bar{F}^0_i + \bar{a}_{i,i+1}\,\bar{F}^0_{i+1} \end{aligned} \tag{A.5}$$

The edge update, described in Section 3, uses the updated node features F^1 to calculate A^1 by using the cosine similarity function:

$$\underbrace{A^1}_{n\times n} = \textsc{Cosine\_Sim}\left(\underbrace{F^1}_{n\times c}, \underbrace{F^1_{\text{shifted}}}_{n\times c}\right) \tag{A.6}$$

and for the ith node, it is derived as:

$$A^1_{\text{node}\,i} = \left[\mathbf{0}_{i-3} \;\; \eta_{i,i-2} \;\; \eta_{i,i-1} \;\; 1 \;\; \eta_{i,i+1} \;\; \eta_{i,i+2} \;\; \mathbf{0}_{n-i-2}\right] \tag{A.7}$$

where

$$\eta_{i,j} \triangleq \textsc{Cosine\_Sim}\left(F^1_i, F^1_j\right).$$

To show that the first non-zero cosine similarity term of the updated affinity $A^1_{\text{node}\,i}$ has the same coefficient as the first term of $(A^0)^2$ in Eq. (A.3), we expand the nonzero cosine similarity terms and show the coefficients of Eqs. (A.3) and (A.8) are the same:

$$\begin{aligned} \eta_{i,i-2} &= \frac{\langle F^1_i, F^1_{i-2}\rangle}{\|F^1_i\|\|F^1_{i-2}\|} \\ &\propto \langle F^1_i, F^1_{i-2}\rangle \\ &\propto \Big\langle \left(\bar{a}_{i,i-1}\,\bar{F}^0_{i-1} + \bar{F}^0_i + \bar{a}_{i,i+1}\,\bar{F}^0_{i+1}\right), \\ &\qquad \left(\bar{a}_{i-2,i-3}\,\bar{F}^0_{i-3} + \bar{F}^0_{i-2} + \bar{a}_{i-2,i-1}\,\bar{F}^0_{i-1}\right)\Big\rangle \\ &\propto \left(\bar{a}_{i,i-1}\,\bar{a}_{i-2,i-1}\,\bar{F}^0_{i-1}\right) \end{aligned} \tag{A.8}$$

where $\langle\cdot,\cdot\rangle$ denotes inner product. The above analysis shows that the use of GNN provides an efficient. Indeed the resulting computational complexity will $O(|V|)$, where $|V|$ is the total number of nodes in the graph, while the traditional depth first search that has a substantially larger computational complexity of $O(|V|+|E|)$ where $|E|$ denotes the number of edges in the graph. ■

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., 2012. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans. Pattern Anal. Mach. Intell. 34 (11), 2274–2282.

Andres, B., Yarkony, J., Manjunath, B., Kirchhoff, S., Turetken, E., Fowlkes, C.C., Pfister, H., 2013. Segmenting planar superpixel adjacency graphs wrt non-planar superpixel affinity graphs. In: International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition. Springer, pp. 266–279.

Arbelaez, P., Maire, M., Fowlkes, C., Malik, J., 2010. Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 33 (5), 898–916.

Athar, A., Mahadevan, S., Ošep, A., Leal-Taixé, L., Leibe, B., 2020. STEm-Seg: Spatio-temporal embeddings for instance segmentation in videos. arXiv preprint arXiv:2003.08429.

Brox, T., Malik, J., 2010. Object segmentation by long term analysis of point trajectories. In: European Conference on Computer Vision. pp. 282–295.

Caelles, S., Maninis, K.-K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., Van Gool, L., 2017. One-shot video object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 221–230.

Chen, Y.-W., Tsai, Y.-H., Lin, Y.-Y., Yang, M.-H., 2020. Vostr: Video object segmentation via transferable representations. Int. J. Comput. Vis. 128 (4), 931–949.

Cheng, J., Tsai, Y.-H., Wang, S., Yang, M.-H., 2017. Segflow: Joint learning for video object segmentation and optical flow. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 686–695.

Comaniciu, D., Meer, P., 2002. Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. 24 (5), 603–619.

Deza, M.M., Laurent, M., 2009. Geometry of Cuts and Metrics. Vol. 15.

Farabet, C., Couprie, C., Najman, L., LeCun, Y., 2012. Learning hierarchical features for scene labeling. IEEE Trans. Pattern Anal. Mach. Intell. 35 (8), 1915–1929.

Garg, S., Goel, V., 2021. Mask selection and propagation for unsupervised video object segmentation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1680–1690.

Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E., 2017. Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212.

Gowda, S.N., Eustratiadis, P., Hospedales, T., Sevilla-Lara, L., 2020. ALBA: Reinforcement learning for video object segmentation. arXiv preprint arXiv:2005.13039.

Guo, F., Wang, W., Shen, J., Shao, L., Yang, J., Tao, D., Tang, Y.Y., 2017. Video saliency detection using object proposals. IEEE Trans. Cybern. 48 (11), 3159–3170.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2961–2969.

Hu, Y.-T., Huang, J.-B., Schwing, A., 2017. Maskrnn: Instance level video object segmentation. In: Advances in Neural Information Processing Systems. pp. 325–334.

Huang, P., Han, J., Liu, N., Ren, J., Zhang, D., 2021. Scribble-supervised video object segmentation. IEEE/CAA J. Autom. Sin. 9 (2), 339–353.

Huang, X., Xu, J., Tai, Y.-W., Tang, C.-K., 2020. Fast video object segmentation with temporal aggregation network and dynamic template matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8879–8889.

Jain, S.D., Xiong, B., Grauman, K., 2017. Fusionseg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 2117–2126.

Juneja, M., Vedaldi, A., Jawahar, C., Zisserman, A., 2013. Blocks that shout: Distinctive parts for scene classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 923–930.

Kardoost, A., Keuper, M., 2018. Solving minimum cost lifted multicut problems by node agglomeration. In: Asian Conference on Computer Vision. Springer, pp. 74–89.

Kernighan, B.W., Lin, S., 1970. An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. 49 (2), 291–307.

Keuper, M., Andres, B., Brox, T., 2015a. Motion trajectory segmentation via minimum cost multicuts. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3271–3279.

Keuper, M., Levinkov, E., Bonneel, N., Lavoué, G., Brox, T., Andres, B., 2015b. Efficient decomposition of image and mesh graphs by lifted multicuts. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1751–1759.

Khoreva, A., Benenson, R., Ilg, E., Brox, T., Schiele, B., 2019. Lucid data dreaming for video object segmentation. Int. J. Comput. Vis. 127 (9), 1175–1197.

Koh, Y.J., Kim, C.-S., 2017. Primary object segmentation in videos based on region augmentation and reduction. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7417–7425.

Lai, Z., Lu, E., Xie, W., 2020. MAST: A memory-augmented self-supervised tracker. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6479–6488.

Lee, J., Lee, I., Kang, J., 2019. Self-attention graph pooling. arXiv preprint arXiv:1904.08082.

Lin, H., Wu, R., Liu, S., Lu, J., Jia, J., 2021. Video instance segmentation with a propose-reduce paradigm. arXiv preprint arXiv:2103.13746.

Liu, Y., Cheng, M.-M., Hu, X., Wang, K., Bai, X., 2017. Richer convolutional features for edge detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3000–3009.

Liu, C., Wang, W., Shen, J., Shao, L., 2018. Stereo video object segmentation using stereoscopic foreground trajectories. IEEE Trans. Cybern. 49 (10), 3665–3676.

Lu, X., Ma, C., Ni, B., Yang, X., Reid, I., Yang, M.-H., 2018. Deep regression tracking with shrinkage loss. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 353–369.

Lu, X., Wang, W., Ma, C., Shen, J., Shao, L., Porikli, F., 2019. See more, know more: Unsupervised video object segmentation with co-attention siamese networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3623–3632.

Luiten, J., Voigtlaender, P., Leibe, B., 2018a. PReMVOS: Proposal-generation, refinement and merging for video object segmentation. In: Asian Conference on Computer Vision. pp. 565–580.

Luiten, J., Voigtlaender, P., Leibe, B., 2018b. Premvos: Proposal-generation, refinement and merging for video object segmentation. In: Asian Conference on Computer Vision. Springer, pp. 565–580.

Luiten, J., Zulfikar, I.E., Leibe, B., 2020. Unovost: Unsupervised offline video object segmentation and tracking. In: The IEEE Winter Conference on Applications of Computer Vision. pp. 2000–2009.

Maninis, K.-K., Pont-Tuset, J., Arbeláez, P., Van Gool, L., 2017. Convolutional oriented boundaries: From image segmentation to high-level tasks. IEEE Trans. Pattern Anal. Mach. Intell. 40 (4), 819–833.

Miao, J., Wei, Y., Yang, Y., 2020. Memory aggregation networks for efficient interactive video object segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10366–10375.

Najman, L., Schmitt, M., 1996. Geodesic saliency of watershed contours and hierarchical segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 18 (12), 1163–1173.

Oh, S.W., Lee, J.-Y., Xu, N., Kim, S.J., 2019. Video object segmentation using space-time memory networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9226–9235.

Perazzi, F., Wang, O., Gross, M., Sorkine-Hornung, A., 2015. Fully connected object proposals for video segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3227–3234.

Perona, P., Shiota, T., Malik, J., 1994. Anisotropic diffusion. In: Geometry-Driven Diffusion in Computer Vision. Springer, pp. 73–92.

Pont-Tuset, J., Arbelaez, P., Barron, J.T., Marques, F., Malik, J., 2016. Multiscale combinatorial grouping for image segmentation and object proposal generation. IEEE Trans. Pattern Anal. Mach. Intell. 39 (1), 128–140.

Pont-Tuset, J., Perazzi, F., Caelles, S., Arbeláez, P., Sorkine-Hornung, A., Van Gool, L., 2017. The 2017 davis challenge on video object segmentation. arXiv preprint arXiv:1704.00675.

Ranjan, A., Jampani, V., Balles, L., Kim, K., Sun, D., Wulff, J., Black, M.J., 2019. 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. pp. 12240–12249.

Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G., 2008. The graph neural network model. IEEE Trans. Neural Netw. 20 (1), 61–80.

Shao, D., Xiong, Y., Zhao, Y., Huang, Q., Qiao, Y., Lin, D., 2018. Find and focus: Retrieve and localize video events with natural language queries. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 200–216.

Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 22 (8), 888–905.

Song, H., Wang, W., Zhao, S., Shen, J., Lam, K.-M., 2018. Pyramid dilated deeper convlstm for video salient object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 715–731.

Sun, D., Roth, S., Black, M.J., 2014. A quantitative analysis of current practices in optical flow estimation and the principles behind them. Int. J. Comput. Vis. 106 (2), 115–137.

Sun, J., Zheng, W., Zhang, Q., Xu, Z., 2021. Graph neural network encoding for community detection in attribute networks. IEEE Trans. Cybern..

Tokmakov, P., Alahari, K., Schmid, C., 2017a. Learning motion patterns in videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3386–3394.

Tokmakov, P., Alahari, K., Schmid, C., 2017b. Learning video object segmentation with visual memory. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4481–4490.

Tokmakov, P., Schmid, C., Alahari, K., 2019. Learning to segment moving objects. Int. J. Comput. Vis. 127 (3), 282–301.

Tsai, Y.-H., Yang, M.-H., Black, M.J., 2016. Video segmentation via object flow. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3899–3908.

Ventura, C., Bellver, M., Girbau, A., Salvador, A., Marques, F., Giro-i Nieto, X., 2019. Rvos: End-to-end recurrent network for video object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5277–5286.

Wan, J., Liu, Y., Wei, D., Bai, X., Xu, Y., 2020. Super-BPD: Super boundary-to-pixel direction for fast image segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9253–9262.

Wang, W., Lu, X., Shen, J., Crandall, D.J., Shao, L., 2019a. Zero-shot video object segmentation via attentive graph neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9236–9245.

Wang, W., Shen, J., Lu, X., Hoi, S.C., Ling, H., 2020. Paying attention to video object pattern understanding. IEEE Trans. Pattern Anal. Mach. Intell..

Wang, W., Song, H., Zhao, S., Shen, J., Zhao, S., Hoi, S.C., Ling, H., 2019b. Learning unsupervised video object segmentation through visual attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3064–3074.

Xie, S., Tu, Z., 2015. Holistically-nested edge detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1395–1403.

Yang, C., Lamdouar, H., Lu, E., Zisserman, A., Xie, W., 2021. Self-supervised video object segmentation by motion grouping. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7177–7188.

Yang, Z., Wang, Q., Bai, S., Hu, W., Torr, P.H., 2019. Video segmentation by detection for the 2019 unsupervised davis challenge. IEEE.

Yang, Z., Wei, Y., Yang, Y., 2020. Collaborative video object segmentation by foreground-background integration. In: European Conference on Computer Vision. Springer, pp. 332–348.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J., 2018. Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems. pp. 4800–4810.

Zhang, D., Han, J., Yang, L., Xu, D., 2018. SPFTN: A joint learning framework for localizing and segmenting objects in weakly labeled videos. IEEE Trans. Pattern Anal. Mach. Intell. 42 (2), 475–489.

Zhao, W., Zhang, J., Li, L., Barnes, N., Liu, N., Han, J., 2021. Weakly supervised video salient object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16826–16835.

Zhou, T., Li, J., Li, X., Shao, L., 2021a. Target-aware object discovery and association for unsupervised video multi-object segmentation. arXiv preprint arXiv:2104.04782.

Zhou, T., Wang, S., Zhou, Y., Yao, Y., Li, J., Shao, L., 2020. Motion-attentive transition for zero-shot video object segmentation. In: AAAI. Vol. 2, p. 3.

Zhou, Y., Xu, X., Shen, F., Zhu, X., Shen, H.T., 2021b. Flow-edge guided unsupervised video object segmentation. IEEE Trans. Circuits Syst. Video Technol..

Zhu, S.C., Yuille, A., 1996. Region competition: Unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 18 (9), 884–900.

Zulfikar, I.E., Luiten, J., Leibe, B., 2019. Unovost: Unsupervised offline video object segmentation and tracking for the 2019 unsupervised davis challenge. arXiv preprint arXiv:2001.05425.