Iterative multi-path tracking for video and volume segmentation with sparse point supervision

Laurent Lejeune, Jan Grossrieder, Raphael Sznitman

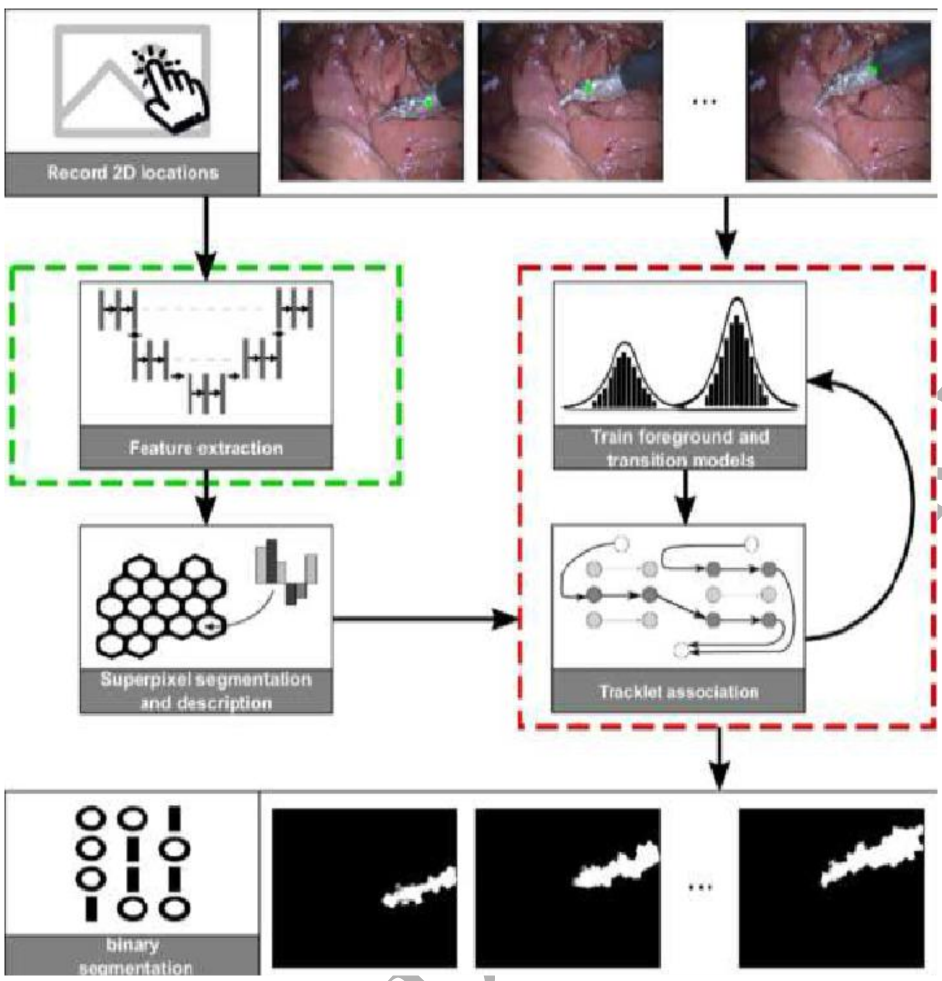
PII:	S1361-8415(18)30663-7
DOI:	https://doi.org/10.1016/j.media.2018.08.007
Reference:	MEDIMA 1404

To appear in: *Medical Image Analysis*

Received date:	21 February 2018
Revised date:	21 August 2018
Accepted date:	27 August 2018

Highlights

- Ground-truth annotation are necessary to train machine learning models.
- We annotate video and volumetric sequences using a single 2D point per frame.
- No constraints on appearance, shape, and motion/displacement of object of interest.
- Promising results on surgical tool and slitlamp videos, brain MRI, CT scans of inner ear.


Record 2D locations
Feature extraction
Superpixel segmentation and description
Train foreground and transition models
Tracklet association
binary segmentation
...

Iterative multi-path tracking for video and volume segmentation with sparse point supervision

Laurent Lejeune[*], Jan Grossrieder, Raphael Sznitman

Ophthalmic Technology Laboratory, ARTORG Center, University of Bern, Murtenstrasse 50, 3008 Bern, Switzerland

Abstract

Recent machine learning strategies for segmentation tasks have shown great ability when trained on large pixel-wise annotated image datasets. It remains a major challenge however to aggregate such datasets, as the time and monetary cost associated with collecting extensive annotations is extremely high. This is particularly the case for generating precise pixel-wise annotations in video and volumetric image data. To this end, this work presents a novel framework to produce pixel-wise segmentations using minimal supervision. Our method relies on 2D point supervision, whereby a single 2D location within an object of interest is provided on each image of the data. Our method then estimates the object appearance in a semi-supervised fashion by learning object-image-specific features and by using these in a semi-supervised learning framework. Our object model is then used in a graph-based optimization problem that takes into account all provided locations and the image data in order to infer the complete pixel-wise segmentation. In practice, we solve this optimally as a tracking problem using a K-shortest path approach. Both the object model and segmentation are then refined iteratively to further improve the final segmentation. We show that by collecting 2D locations using a gaze tracker, our approach can provide state-of-the-art segmentations on a range of objects and image modalities (video and 3D volumes), and that these can then be used to train supervised machine learning classifiers.

Keywords: Semi-supervised learning, Semantic segmentation, Multi-path tracking, Point-wise supervision

1. Introduction

At its core, semantic segmentation is tasked with associating pixels, or voxels, of an image with a label that corresponds to a meaningful category. As a fundamental problem in medical image computing, an impressive amount of research on the topic has been conducted in recent years, spanning methods that segment tumors in MRI volumes (Zikic et al., 2014; Menze, 2014), airways from chest CT scans (Miyawaki et al., 2017), vessels in retinal scans (Pilch et al., 2012) or mitochondria in electron microscopes (Seyedhosseini et al.,

[*]Corresponding author
Email address: laurent.lejeune@artorg.unibe.ch (Laurent Lejeune)

2013) to name a few.

To this day, the vast majority of state-of-the-art segmentation approaches rely heavily on supervised machine learning frameworks (Sweeney et al., 2014; Menze, 2014), and in particular deep learning (Garcia-Garcia et al., 2017), to produce excellent segmentation results. In general, these methods depend on large amounts of training examples to learn complex prediction models. Critically, most of these models are trained using pixel-wise annotations associated with training examples. While highly effective, the cost of acquiring such pixel-wise annotations for training machine learning methods is often overlooked and yet a central limiting factor for aggregating extremely large training datasets. This is particularly the case for video and 3D image data where annotations are extremely costly (*i.e.* days per video sequence). This in turn negatively impacts the capacity to train high-performing segmentation models, as the number of training samples remains relatively small.

To reduce the burden of producing pixel-wise annotations, a number of semi-supervised concepts have been proposed such as active learning (Konyushkova et al., 2015), domain adaption (Tzeng et al., 2017) and crowd-sourcing (Mavandadi et al., 2012). Alternatively, a number of recent methods propose to infer pixel-wise segmentation directly from image labels (*i.e.* the image contains a tumor) by leveraging strong object or shape priors (Menze et al., 2010). In effect these methods attempt to refine segmentations from pre-trained neural networks for generic object classes (Su et al., 2015) or trained attention models (Kingma et al., 2014). While extremely promising, such methods still only produce coarse segmentations and remain ill suited for training complex prediction models.

At the same time, the method by which annotations are provided has important practical implications in terms of convenience for the annotator and can also greatly speed-up the annotation process (Ferreira et al., 2012). For example, providing tumor pixel segmentations in volumetric data would be infeasible if each pixel were to be specified individually. Instead, there is an important body of work that has considered alternative user-interaction mechanisms. For instance, (Konyushkova et al., 2015) used 3D image planes to specify the boundary between image backgrounds and objects in 3D modalities. Scribbles of positive and negative image regions were also shown to be effective in speeding up annotation generation (Roberts et al., 2011). Even more efficient, was the use of 2D points to sparsely annotate image data (Bromiley et al., 2014; Bearman et al., 2016). Such 2D point supervision is interesting as it can be produced by manually clicking on images from 3D volumes or video sequences, or by using a gaze tracker to passively record 2D coordinates of the object as they are viewed (Yun et al., 2013; Khosravan et al., 2017; Lejeune et al., 2017). This latter strategy is promising as it holds the potential to annotate at high framerate, but is challenging due to limited, if any, information regarding the background. To overcome this, previous methods that generate segmentations from 2D supervision have relied on strong assumptions on the object size, the background scene, or clouds of 2D locations, whereby limiting usability.

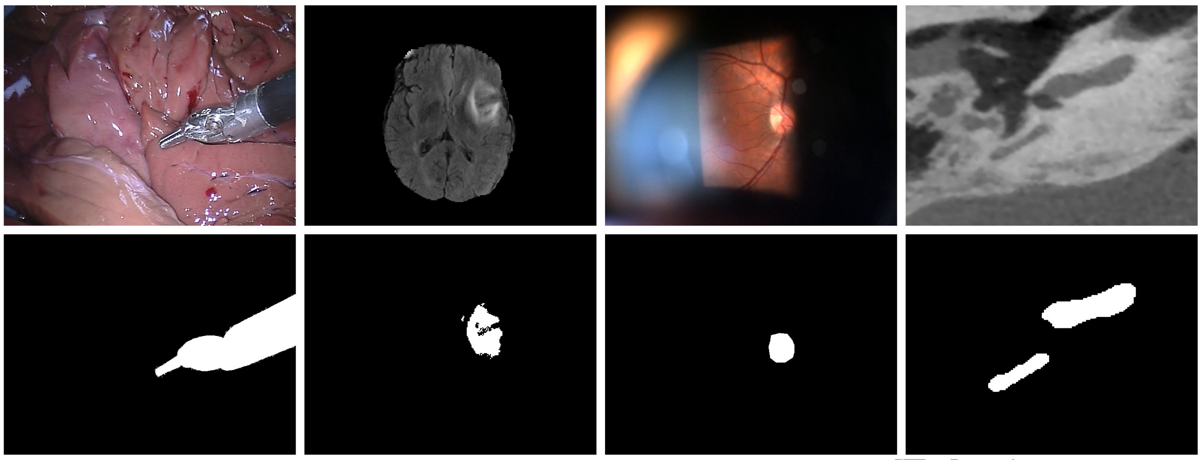


Figure 1: Example of objects of interest in different video and volumetric modalities. The bottom row shows the pixel wise segmentation for each case: From left to right: A surgical instrument during an endoscopic procedure, a 3D MRI scan containing a brain tumor, an optic disc seen from a slit lamp microscope and a cochlea cross-section in a 3D CT scan.

To generalize the use of sparse 2D point supervision to infer segmentations in a given image volume or sequence, we propose a novel framework that avoids the need to assume much about the object and only requires 2D object locations to be specified. In particular, we assume that the object of interest is compact and always present in the image, but can have arbitrary size regardless of the image modality considered. With the goal of segmenting this object of interest, our method first builds an object appearance model from the provided 2D locations. We then construct a 3D graph over the entire image data, from which the complete object segmentation is estimated in each frame. To do this, the graph is optimized in a multi-source, single sink max-flow setting that uses the object model and we show that this problem can be solved optimally using a K-shortest path approach. We then iterate this optimization procedure using the previously obtained result to update our object model and thus refine the produced segmentations. In order to achieve consistent segmentations from such sparse annotations regardless of the image type or object, we also propose to use image and object specific features that are learned from the image data. This is achieved by using a Convolutional Neural Network (CNN) and a novel loss function that takes into account the 2D image locations. Combined, we show that our framework provides a significant improvement over existing methods on a number of varied datasets (see Fig. 1). We show that our results are not only more similar to traditional hand segmentations to those produced by state-of-the-art methods, but that they only induce mild reductions in performance when used to train prediction models. Beyond this, we show that by using a low-cost gaze tracker to generate supervised 2D locations, a user can generate annotations at high framerate.

In the following sections, we begin by describing existing methods closest to our work. We then introduce our framework in Sec. 3 and describe its detail thereafter. In Sec. 7, we show how our method performs on a number of tasks and datasets, as well as compared to existing methods.

2. Related Works

While semi-supervised methods for segmentation tasks encompass a wide range of applications and settings (Chapelle et al., 2006), we briefly discuss a number of methods that are related to the present paper. In particular, we focus on graph-based and learning-based methods, as well as methods that leverage different user-input mechanisms.

Graph-based methods: Graph-based methods are well studied in both the computer vision and medical imaging communities. The seminal work of Boykov and Funka-Lea (2006) first introduced an efficient and optimal method for binary segmentation using both object and background appearence models. GrabCut and other variants (Rother et al., 2004; Yu and Qi, 2014) further improved the approach using iterative optimizations. At their core, these methods rely on object and background models, computed from provided supervision, to segment objects. More recently, the approach of Karthikeyan et al. (2015) extracts visual *tracklets* by combining gaze inputs from multiple individuals and optimizes a patchwork of locations using a Hungarian algorithm to globally extract bounding boxes that are then refined using GrabCut. In particular, by leveraging crowds of users to provide pointwise indications of object of interest, the method effectively produces segmentations from clouds of points. In contrast, in the approach we propose, only a single point per frame within the object of interest is given. This is similar to the work of Khosravan et al. (2017), who make use of saliency maps (Itti et al., 1998) derived from gaze locations in CT scans to segment lung lesions. The saliency maps serve as object and background models (assuming bounds on the lesion sizes) in a graph cut optimizer.

Semi-supervised learning methods: A wide range of semi-supervised learning methods are related to our present work. Given that in our setting, only positive examples consisting of parts of the object are provided, our problem is closely related to transductive learning (Rohrbach et al., 2013; Yang and Mohri, 2017) and more specifically P(ositive)-U(nlabeled) learning (Li and Liu, 2005; Kiryo et al., 2017). In such cases, only part of the positive set is labeled in addition to a large amount of unlabeled data. To tackle this setting, most methods focus on providing more adapted loss functions during training or leveraging priors to constrain the ensuing classifier.

Early on and also using a gaze tracker, Vilariño et al. (2007) suggested a P-U learning setting to detect polyps from endoscopic video frames. This approach bares some semblance to ours, except that we explicitly take into account temporal information by means of a graph to further constrain our segmentation. At the same time, unlike their approach, we do not assume that the object is of a given size. Along this line, Lejeune et al. (2017) considered a P-U setting by explicitly learning a classifier using a loss function that takes into account the uncertainty associated with unlabeled samples. These uncertainties are derived from gaze locations while Probability Propagation (Zhou et al., 2004) is used to estimate unknown samples. Within a deep learning framework, Bearman et al. (2016) suggested learning a CNN using gaze information as well as a strong object prior in order to improve convergence of their network. The method performs well on natural images of complex scenes, as the objectness prior is learned from a large corpus of natural images. Similarly, FusionSeg (Jain et al., 2017) used a deep learning approach with an initial object outline to segment object boundaries in video sequences. This approach, which is highly related to tracking, combines both motion and appearance to track the object with limited user interaction.

User-input models: Given the wide use of machine learning, the extensive research on user-input methods and interactive algorithms, is by no means surprising. Beyond traditional polygon outlining, scribbling has been proposed to annotate faster. 2D point locations, either on individual images or in video streams has also been shown to be effective when providing coarse information in extremely fast amounts of time (Papadopoulos et al., 2017).

Related to the work here, gaze trackers have received an increasing amount of attention given that the technology has greatly improved over the last decade and seen a strong reduction in cost (Soliman et al., 2016; Mettes et al., 2016; Bearman et al., 2016). In these works, gaze information provides a form of sparse annotations to train machine learning classifiers extremely quickly. In particular, large amounts of annotations can be accumulated by crowds of individuals observing natural video data for example. In the context of medical imaging, gaze locations have also been investigated to see how image annotation could be performed (Sadeghi et al., 2009), or how pathologies could be identified by a limited number of viewings of video or volumetric image data (Vilariño et al., 2007; Khosravan et al., 2017). Unfortunately, most approaches so far have only been shown to work in extremely limited scenarios (*e.g.* one type of object in a single modality). In our work, we show how object segmentations can be computed by using a gaze tracker to collect 2D locations of the object at framerate, in a single pass, without collecting or assuming information on the background scene, the object size or its motion speed. This allows our approach to be highly generic and effective on a variety of image modalities and object types.

3. Overview and problem formulation

We now present our method that takes as input an image sequence (or an image volume) containing a single object of interest and produces a pixel-wise segmentation of this object for all frames. In general, we assume that at most one object of interest is in the sequence and that part of the object is visible in each frame. For each frame in the sequence, we assume that a 2D location (*i.e.* a pixel) within the object is provided by a user. Hence, while we are interested in determining the entire object segmentation, the provided information only specifies local and compact regions of the object. These provided locations may be spatially disjoint and can refer to different or the same areas of the object. For this reason, our approach treats the task as a tracking problem where the image regions specified by the 2D locations must be jointly and coherently tracked so to recover the complete object segmentation. To do this, our approach hinges on two components.

The first is a strategy to characterize the object of interest by using the provided image sequence and associated 2D locations. This is achieved by learning a classifier in a transductive fashion. In particular, we resort to bagging a set of decision trees. Instead of combining the aforementioned classifier with hand-crafted or learned features from large datasets, we learn features explicitly from the considered image sequence while using the 2D locations as a soft prior. This is achieved by training a U-Net architecture (Ronneberger et al., 2015) as an autoencoder in combination with a loss function that takes into account known object locations. By using the image features from this network, the classifier can then be used to assess the likelihood of image regions belonging to the object of interest.

The second component considers each specified 2D location as a potential target to track. To segment the object from these locations, we construct a graph over all compact image regions (*i.e.* superpixels) in the image sequence. The object segmentation is then inferred with a network flow optimization strategy whereby each of the provided 2D locations correspond to flow sources and we use the object likelihood to establish a series of costs between adjacent edges in our graph. We show that this graph can be optimized exactly and efficiently using a K-shortest path approach. To further improve the segmentation, we update our classifier using the previously found segmentation and repeat the K-shortest path optimization to produce an improved segmentation. This process is iterated until convergence of the final produced segmentation.

By combining these two components, we show in our experiments that effective segmentations can be generated from extremely few 2D locations, without further prior assumptions on the image modality or the object of interest.

We now briefly describe some notation that will be used throughout this paper and which are summarized in Table. 1. Let the image sequence considered be denoted $\mathcal{I} = \{I_0, \dots, I_T\}$ and let $\boldsymbol{g} = \{g_t\}_{t=0}^{T}$ with $g_t \in \mathbb{R}^2$ be a 2D pixel location in I_t. While we are ideally interested in a pixel-wise segmentation, we decompose each image as a set of superpixels so to reduce computational complexity. Given the volumetric nature of the problem considered, we opt to use 3D superpixels (Chang et al., 2013) to group similar pixels over multiple frames. We thus let I_t be described by the set of N_t non-overlapping superpixels $S_t = \{s_t^n\}_{n=0}^{N_t}$ and define the set of all superpixels across all images as $\mathcal{S} = \{S_t\}_{t=0}^{T}$. In addition, we assign to each superpixel s_t^n an appearance feature vector a_t^n and define $\boldsymbol{a} = \{a_t^n | t = 0, \dots, T \quad n = 0, \dots, N_t\}$. We denote the set $\mathcal{S}^p = \{s_t^n | g_t \in s_t^n, t = 0, \dots, T, n = 0, \dots N_t\}$ as all superpixels observed and the rest as $\mathcal{S}^u = \mathcal{S} \setminus \mathcal{S}^p$.

Symbol	Description
T	Number of frames
I_t	Image at time t
g_t	Coordinates of 2D location at time t
N_t	Number of superpixels at time t
s_t^n	Superpixel n at time t
a_t^n	Feature vector of s_t^n
u_t^n	Histogram of oriented optical flow of s_t^n
Y_t^n	Binary random variable that models objectness of s_t^n
ρ_t^n	Probability of s_t^n being object given the object model
$\mathcal{T}_t^n$	Tracklet starting at time t and superpixel n
r_t^n	Centroid of s_t^n
e_t^n, f_t^n, C_t^n	Edge, flow and cost for passing through $\mathcal{T}_t^n$
$e_t^{n,m}$, $f_t^{n,m}$, $C_t^{n,m}$	Edge, flow and cost for linking $\mathcal{T}_t^n$ and $\mathcal{T}_{t+1}^m$
$e_t^{\mathcal{E},n}$, $f_t^{\mathcal{E},n}$, $C_t^{\mathcal{E},n}$	Edge, flow, and cost for entering the network from $\mathcal{T}_t^n$
τ_ρ	Threshold applied on edges e_t^n according to ρ_t^n
τ_u	Threshold applied on edges $e_t^{n,m}$ according to u_t^n
τ_{trans}	Threshold on edges $e_t^{i,j}$ and $e_t^{g,n}$ according to u_t^n
R	Radius around 2D location (entrance), and tracklet transitions
Z_t	Objectness prior at time t
σ_g	Standard-deviation of objectness prior for feature extraction

Table 1: Notation summary

4. Transductive foreground model

To build a model of the object appearance, we take a transductive learning approach. Here we follow a P-U learning scheme in which only a few positive samples are given along with a larger set of unknown samples. In practice for an image, we expect one superpixel to be annotated compared to hundreds of unobserved ones. While using Neural Networks would be effective for supervised binary segmentation problems, it is unclear what loss function one should minimize in a P-U regime. Instead, we propose to train a simple bagging classifier with novel features that are both image and object specific, and allow coarse superpixel regions to be characterized.

4.1. Probabilistic estimation by bagging

To build a prediction model, we train M binary decision trees by using different data subsets. Each tree takes as input the feature vector $a_t^n \in \mathbb{R}^D$ characterizing the superpixel s_t^n and estimates $Y_t^n \in \{0,1\}$, where $Y_t^n = 1$ if it belongs to the object and 0 otherwise. For each tree, the entire positive set $\mathcal{S}^p$ is used for training, in addition to $|\mathcal{S}^p|$ randomly selected samples with replacement from the unlabeled set $\mathcal{S}^u$. The latter are treated as negative samples. The trees are then trained using the Gini impurity loss function (Menze et al., 2009), with $\sqrt{D}$ randomly selected features considered at each node of a tree. Then for a given superpixel s_t^n, the probability that it is part of the object, $\rho_t^n = P(Y_t^n = 1 | a_t^n)$ can be computed by averaging the predictions over all M trees.

4.2. Image-object specific features

In general, there are many different features that could be used in the above classifier. In this context however, we wish to use features that are effective for segmenting a specific object in a given image modality. That is, we are interested in learning features that are both image-object specific (IOS), but do not need to generalize to other unseen data.

To this end, we learn features by making use of an autoencoder neural network. As illustrated in Fig. 2, we let the network take as input an image from a sequence and is tasked to predict the same image as output. In our network, we use three stacked convolutional layers with 3×3 filter with strides of 1 per level in an encoding and decoding path. We also perform batch normalization with ReLU activations after each convolutional layer. The last layer is a convolutional layer with filter size 1×1 and a sigmoid activation. In practice, as our network has 4 levels, where we first downscale the images to the nearest width and height divisible by 16.

While such an autoencoder can be trained by minimizing the L^2-norm (Vincent et al., 2008), we are interested in forcing the network to have strong performances on regions of the object of interest rather than on potentially irrelevant parts of the image. As such, we propose to impose an "objectness prior", which we add to our loss function in the form of a soft constraint. Specifically, we define $Z_t \sim \mathcal{N}(g_t, \sigma_g^2 \mathbf{1})$ to be a 2D weighted map of the same size as I_t. The mean of this image is centered on the location g_t and has symmetric variance σ_g^2. We then integrate this map in our loss as

$$\mathcal{L} = \sum_{I_t \in \mathcal{I}} \sum_{k,l} Z_t(k,l) \| I_t(k,l) - \hat{I}_t(k,l) \|^2, \tag{1}$$

where $\hat{I}_t$ is the output of the network, k and l are pixel indices in a $W \cdot H$ sized image I_t. In effect, this loss penalizes incorrect reconstructions more heavily on the regions that are known to be part of the object.

After training, a forward pass is performed on an image and features of dimension 512 are extracted at the output of the deepest layer. These features correspond to a downscaled version of the input image which

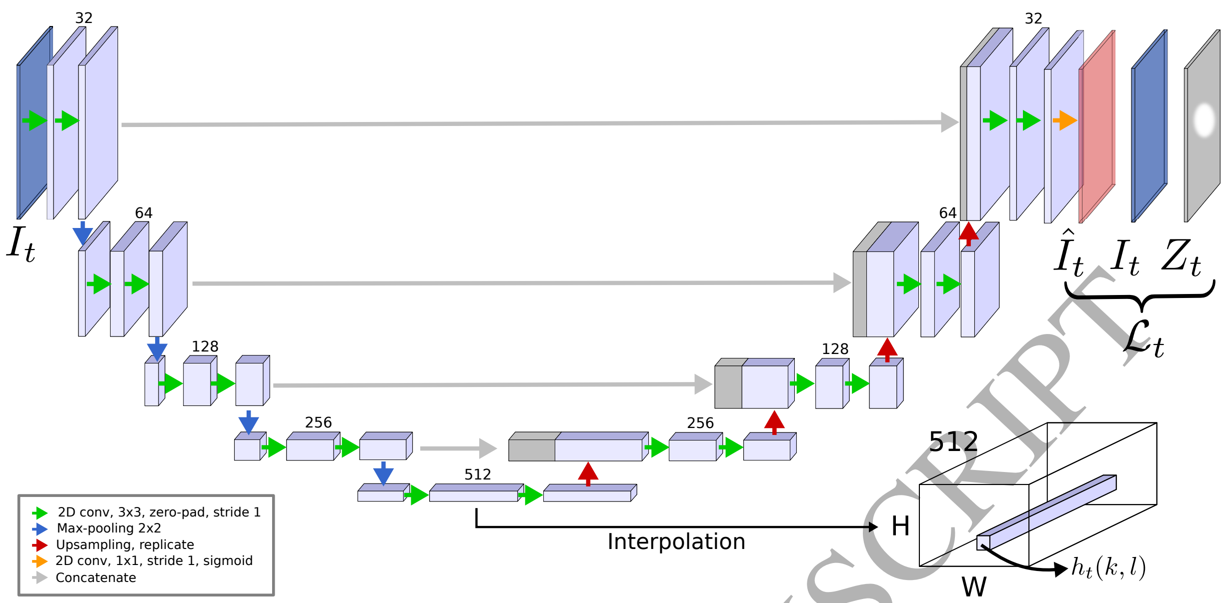


Figure 2: Image-object specific features. The network is tasked to reconstruct the input image I_t (dark blue). By means of a loss function $\mathcal{L}_t$, the reconstructed image, $\hat{I}_t$ (red), is strongly penalized at 2D locations provided by means of the soft prior, Z_t. At test time, the features $h_t(k,l)$ are extracted by interpolating the bottom layer to the original input size.

are then upscaled to the original image size using bicubic interpolation. The feature vector associated to a superpixel s_t^n is then taken to be the mean over all pixels contained within it,

$$a_t^n = \frac{1}{|s_t^n|} \sum_{(k,l)\in s_t^n} h_t(k,l), \tag{2}$$

where $h_t(k,l) \in \mathbb{R}^{512}$ is the feature vector extracted at pixel (k,l). As we will show in our experiments, this strategy generally improves the overall performance of our method.

5. Segmentation by tracking

Given the above local object model, we wish to provide a global strategy to infer an accurate segmentation of the object across all frames. As we make no assumption on the object of interest (*e.g.* shape, color, motion, etc.), we hypothesise that by tracking these local regions over the entire data volume, that a complete segmentation of the object can be coherently inferred. That is, we consider each region specified by a provided 2D location to be an individual target, that could potentially depict different parts of the same object. In what follows, we show how these different regions can be tracked optimally so as to provide the complete object segmentation.

5.1. *MAP Formulation*

To track the 2D locations as function of the object, we define $\boldsymbol{Y} = \{Y_t^n | \forall(t,n)\}$ as the set of all Y labels. As defined in Sec. 3, $\boldsymbol{g}, \boldsymbol{a}$ are the grouping variables of the provided 2D locations and the extracted superpixel features, respectively. We then define our segmentation problem as a Maximum a posteriori (MAP) optimization,

$$y^* = \arg\max_{y \in \mathcal{Y}} P(\boldsymbol{Y} = \boldsymbol{y} | \boldsymbol{a}, \boldsymbol{g}), \tag{3}$$

where y^* is the sought out binary labels for all frames. Assuming that Y_t^n is conditionally independent given the observed variables $\boldsymbol{a}$, we rewrite Eq. (3) as,

$$y^* = \arg\max_{y \in \mathcal{Y}} \prod_{m,n,t} P(Y_t^n | \boldsymbol{a}, \boldsymbol{g}) P(Y_t^n | a_{t-1}^m) P(Y_t^n | a_t, g_t). \tag{4}$$

In particular, the three terms of the decomposition of Eq. (4) correspond to different aspects of the object appearance models. Concretely,

- $P(Y_t^n | \boldsymbol{a}, \boldsymbol{g})$ is modeled using our classifier (Sec. 4.1) and behaves as an object appearance model.
- $P(Y_t^n | a_{t-1}^m)$ models the similarity between two superpixels in successive frames, so to describe how frame-to-frame probabilities propagate.
- $P(Y_t^n | a_t, g_t)$ models the likelihood that a given superpixel s_t^n is visually similar to the one selected by the 2D location g_t. In practice, in the case where the object of interest is large and visually homogeneous, this term allows to initiate several tracks for a single given 2D annotation.

While optimizing Eq. (4) appears complex, we show in the following section that it can be performed efficiently by means of an integer program formulation and a K-Shortest Path optimization.

5.2. *Flow network formulation*

Thanks to its inherent structure, our MAP problem can be mapped into a cost-flow problem and can be solved efficiently. Specifically, we wish to determine where flow emitted from a source node must traverse a graph in order to minimize the traversal cost to a sink node (Zhang et al., 2008). For that matter, we associate to each superpixel s_t^n a tracklet $\mathcal{T}_t^n$ (*i.e.* an edge that represents the entrance and exit of a superpixel) and define $r_t^n \in \mathbb{R}^2$ as the central pixel of superpixel s_t^n. Fig. 3 shows a graphical representation of our flow network formulation.

As a first step, the MAP problem of Eq. (4) is transformed into an Integer Program (IP) (Schrijver, 1998). To simplify notations, let $\alpha_t^{m,n} := P(Y_t^n = 1 | a_{t-1}^m)$, $\beta_t^n := P(Y_t^n = 1 | a_t, g_t)$, and $\rho_t^n := P(Y_t^n = 1 | \boldsymbol{a}, \boldsymbol{g})$. We also introduce a sink node $\mathcal{X}$, and set of source nodes $\mathcal{E}_t$ such that each pushes flow onto the corresponding frame I_t. Additionally, we introduce the variables f_t^n, $f_t^{m,n}$, $f_t^{\mathcal{E},n}$, and $f^{n,\mathcal{X}}$ to denote tracklet, transition, entrance and exit flows, respectively. The corresponding IP is thus given by,

Maximize

$$\sum_{t,n} \log \frac{\rho_t^n}{1-\rho_t^n} f_t^n + \sum_{t,m} \log \frac{\alpha_t^{m,n}}{1-\alpha_t^{m,n}} \sum_{t,n} f_t^{m,n} + \sum_{t,n} \log \frac{\beta_t^n}{1-\beta_t^n} f_t^{\mathcal{E}_t,n}, \tag{5a}$$

subject to,

$$\sum_n f_t^{m,n} \le 1, \qquad \forall t,m,n \tag{5b}$$

$$\sum_m f_t^{m,n} - \sum_p f_{t-1}^{p,m} \le 0, \qquad \forall t,m,n,p \tag{5c}$$

$$\sum_{m,t} f_t^{\mathcal{E}_t,m} - \sum_p f^{p,\mathcal{X}} \le 0, \qquad \forall t,m \tag{5d}$$

where the above objective function, Eq. (5a), corresponds to the log-likelihood of Eq. (4) and where each flow variable associated to a cost term corresponds to a Bernoulli variable. The constraint defined by Eq. (5b) imposes a maximum flow capacitance of value one, thereby expressing the assumption that a superpixel can only contain a single target. Eq. (5c) imposes flow conservation, *i.e.* at each node the input flow must be equal to the output flow (except for the source and sink nodes). Last, Eq. (5d) imposes that the sum of flow emitted by the source node $\mathcal{E}$ must reach the sink node $\mathcal{X}$. By design, the solution of this IP gives $Y_t^n = 1$ if f_t^n is equal to the edge capacity and 0 otherwise.

To further specify constraints for our given application, we now outline three additional edge-pruning measures. While these could be added directly in Eq. (5a), we opt to describe them here instead.

- **Entrance edge pruning:** The provided 2D locations allow for a strong prior on the location of the object. Intuitively, we wish to force flow where the 2D locations are known (*i.e.* g_t). We therefore connect $\mathcal{E}_t$ to all $\mathcal{T}_t^n$ such that the centroid of the corresponding superpixel, r_t^n, is included in a neighborhood centered at g_t with radius R (see Fig. 4(left)). The parameter R therefore control the quantity of flow that can be pushed from a given source node into its corresponding image. Edges that do not fulfil this condition are pruned.

- **Transition edge pruning:** For edges that link tracklets, we use location and motion constraints to remove edges. For locations, we prune edges where r_t^n is outside of a neighborhood centered on r_{t-1}^m of radius R. Similarly, we estimate superpixel motion by means of a histogram of oriented optical flow (Chaudhry et al., 2009). Defining this motion by $u_t^n \in \mathbb{R}^l$, we prune edges such that $S_m(u_{t-1}^m, u_t^n) < \tau_u$, where $S_m(\cdot,\cdot)$ is the histogram intersection similarity.

- **Tracklet edge pruning:** We let the probabilistic estimation described in Sec. 4.1 be related to the cost of pushing flow through tracklets. Depending on the sequence, this estimate can lead to false

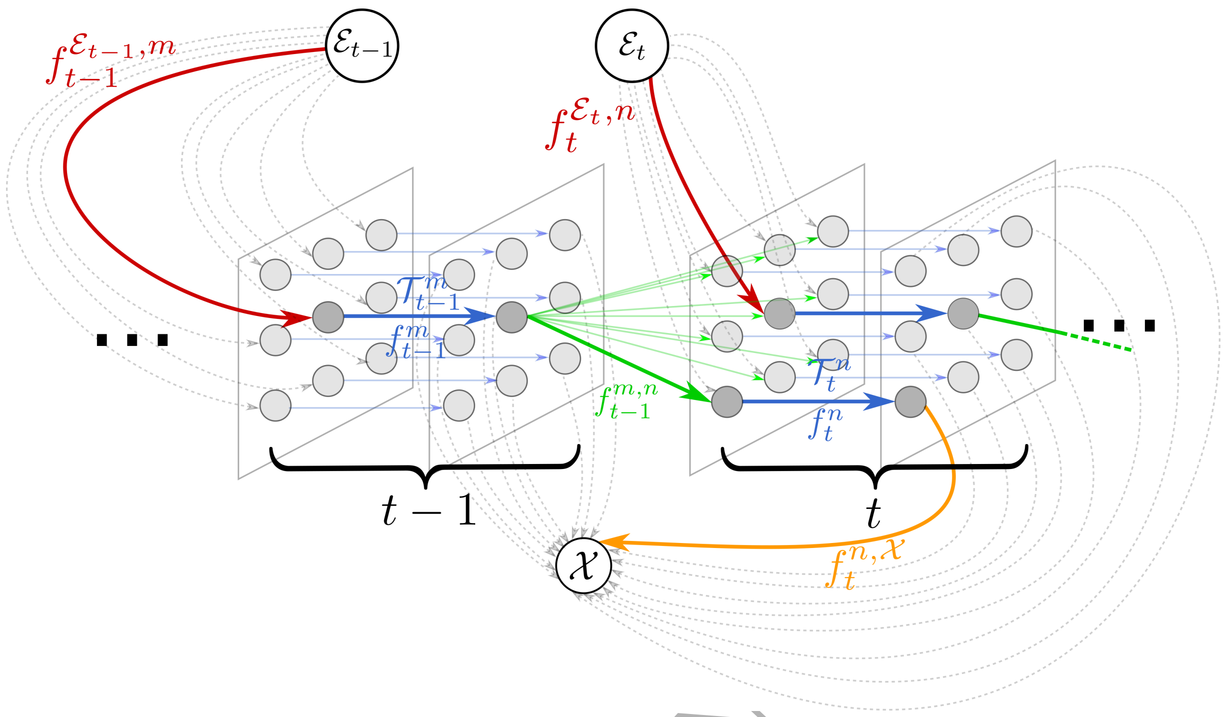


Figure 3: Max-Flow graph (forward case). At each time frame t, a "pseudo" source node $\mathcal{E}_t$ is connected via an edge with flow $f_t^{\mathcal{E},n}$ (red) to tracklet $\mathcal{T}_t^n$. Each tracklet incurs a flow f_t^n (blue) to pass through a superpixel s_t^n. Tracklets in frame t are connected to tracklets in the next frame and allow for flows $f_t^{m,n}$ (green). The flow $f_t^{n,\mathcal{X}}$ can leave any tracklet in the network (orange).

positives (*i.e.* give a high probability value on the background). To circumvent this phenomenon, we prune tracklet edges whose probability are below a threshold, $\rho_t^n < \tau_\rho$.

Note that given this IP formulation, the number of pseudo source nodes $\mathcal{E}_t$ does not change the optimization problem of Eq. (5a). This implies that whether multiple 2D locations or none were be specified on each frame t, the IP would remain unchanged. Naturally, omitted source nodes on frames would reduce the quality of the solution as less information would be available to the foreground model (sec. 4.1). However, as we will show in our experiments, our global optimization recovers paths that span several frames and limits the impact of such cases. Similarly, the pruning of edges does not affect the solution of Eq. (5a) given that these would have infinite cost were they to be explicitly kept, and thus never allow flow to pass through them.

5.3. K-shortest path optimization

As with all IP optimization problems, Eq. (5) is NP-hard (Papadimitriou, 1981). However, as noted in Berclaz et al. (2011), our problem can be relaxed to a Linear Program thanks to the total unimodularity of the constraints matrix. The latter condition guarantees that the solution will converge to an integer solution, making off-the-shelf optimizers suitable (*e.g.* Simplex (Klee and Minty, 1970), Interior point (Kojima et al., 1989)). However, we use a more efficient alternative – the K-shortest paths algorithm (KSP) applied to the case where all edges have unit capacitance. In contrast with generic LP solvers, KSP explicitly leverages the connectivity of nodes in the graph. While Berclaz et al. (2011) used a node-disjoint optimization to restrict nodes from receiving flow from different sources, our tracklet costs, C_t^n, allow a simpler edge-disjoint K-shortest paths algorithm by minimizing the negative of Eq. (5a). We provide further details on our implementation in Appendix A.

Last, to take into account information from previous and future frames, we compute both a forward and backward graph so to track superpixels forward and backward in time. This gives rise to two independent MAP problems to solve: one in each time direction. While we only present the forward case here, the backward case can easily be derived. The final labeling of a sequence is then given by the union of the two solution sets.

5.4. Model costs

In what follows, we describe in detail how edge costs associated to Eq. (5a) are computed.

- **Tracklet costs:** As indicated in Sec. 5.2, ρ_t^n is the probability that the superpixel s_t^n is part of the object according to the classifier. The cost of the corresponding flow f_t^n is thus given by

$$C_t^n = -\log \frac{\rho_t^n}{1-\rho_t^n} \tag{6}$$

  and is illustrated with blue edges in Fig. 3.

- **Transition costs:** We model $\alpha_t^{m,n}$, the likelihood that superpixels s_t^n and s_{t+1}^m correspond to the same region in the sequence. In our flow-network, this corresponds to the cost of transiting from tracklet $\mathcal{T}_t^n$ to $\mathcal{T}_{t+1}^m$ (green edges in Fig. 3).

  While defining costs based on image features for such transitions is complex when the object size and background is unknown, we propose to learn and use an appropriate representation instead. In particular, we use Local Fisher Discriminant Analysis (LFDA), a supervised metric learning method (Sugiyama, 2006), to measure the appearance similarity between two superpixels. In addition to Fisher Discriminant Analysis (FDA) (Welling, 2005), which maximizes between-class scatter while minimizing within-class scatter, LFDA considers multi-modal classes, thereby preserving the local structure of data. In practice, we set the two LFDA parameters empirically: The k nearest-neighbour data points used to compute an affinity matrix, and Q, the dimension of the output space. We select as positive samples the a_t^n with associated probability ρ_t^n larger than a threshold τ_{trans}. As negatives, we randomly select an equal amount of samples below that threshold. Letting V be the LFDA projection matrix, we set

$$\alpha_t^{m,n} = \exp\left(-||V(a_t^n - a_{t+1}^m)||_2^2\right), \tag{7}$$

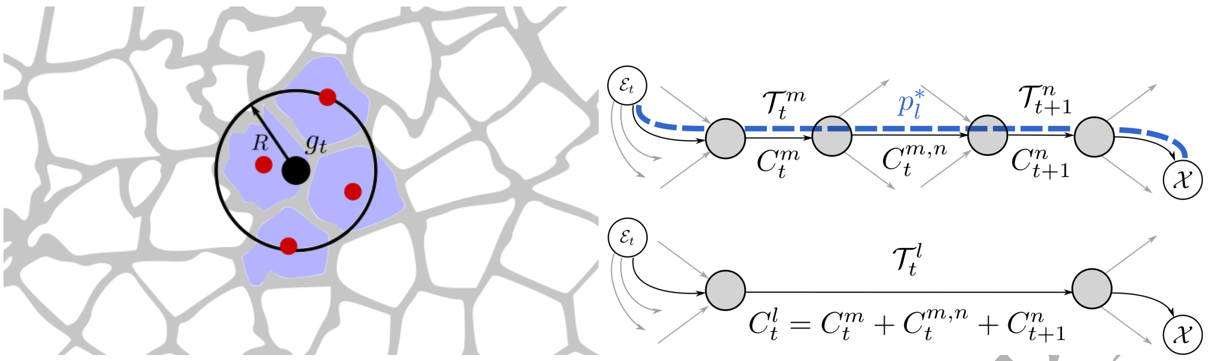


Figure 4: (left) Flow entrance. Superpixel boundaries are drawn in grey. All superpixels whose centroids (red circles) are contained in a circle of radius R will accept entrance flow. (Right) Example of a temporal merge. Top: Path p_l^* (in blue) crosses tracklets $\mathcal{T}_t^n$ and $\mathcal{T}_{t+1}^m$. Bottom: At the next iteration, the corresponding tracklets merge into a single tracklet $\mathcal{T}_t^l$ with cost $C_t^n + C_t^{m,n} + C_{t+1}^m$.

and the flow cost $f_t^{m,n}$ is then given by

$$C_t^{m,n} = -\log \frac{\alpha_t^{m,n}}{1-\alpha_t^{m,n}}. \tag{8}$$

- **Entrance-Exit costs:** The source nodes $\mathcal{E}_t$ allow flow to be pushed through the network. Intuitively, at a given frame, we want to push flow starting from superpixels that are similar to the region given byg_t. Letting $\beta_t^n = P(Y_t^n|a_t, g_t)$ denote the probability of entering the network from $\mathcal{T}_t^n$, we compute,

$$\beta_t^n = \exp\left(-||V(a_t^n - a_t)||_2^2\right) \tag{9}$$

where V is the previously computed LFDA projection matrix, a_t corresponds to the feature vector of the superpixel selected by g_t, whereas a_t^n corresponds to the superpixel feature vector in question. The flow cost associated $f_t^{\mathcal{E}_t,n}$ (red edges in Fig. 3) is then taken as $C_t^{\mathcal{E}_t,n} = -\log(\beta_t^n/(1-\beta_t^n))$. In addition, since we do not model the likelihood of terminating a path, we set the cost of exiting to the sink node $C_t^{n,\mathcal{X}}$ to be 0 for all tracklets.

5.5. Iterative tracking

Given that our object model described in Sec. 4.1, is trained on very few positive samples, solving Eq. (5a) can lead to a number of missed positive superpixels. To circumvent this limitation, we propose to augment the positive set $\mathcal{S}_p$ in an iterative way, by adding new positive samples recovered from our produced KSP estimation. That is, after initially inferring the object segmentation, new object superpixels are considered as positive samples in order to re-train the classifer and recompute the graph costs.

To do this, the costs C_t^n, $C_t^{m,n}$ and $C_t^{\mathcal{E},n}$ are updated after each KSP optimization. More specifically, we define $\mathcal{P}^* = \{p_0^*, ..., p_{K-1}^*\}$ to be the set of solution paths given by the KSP optimization, with p_l^* being the

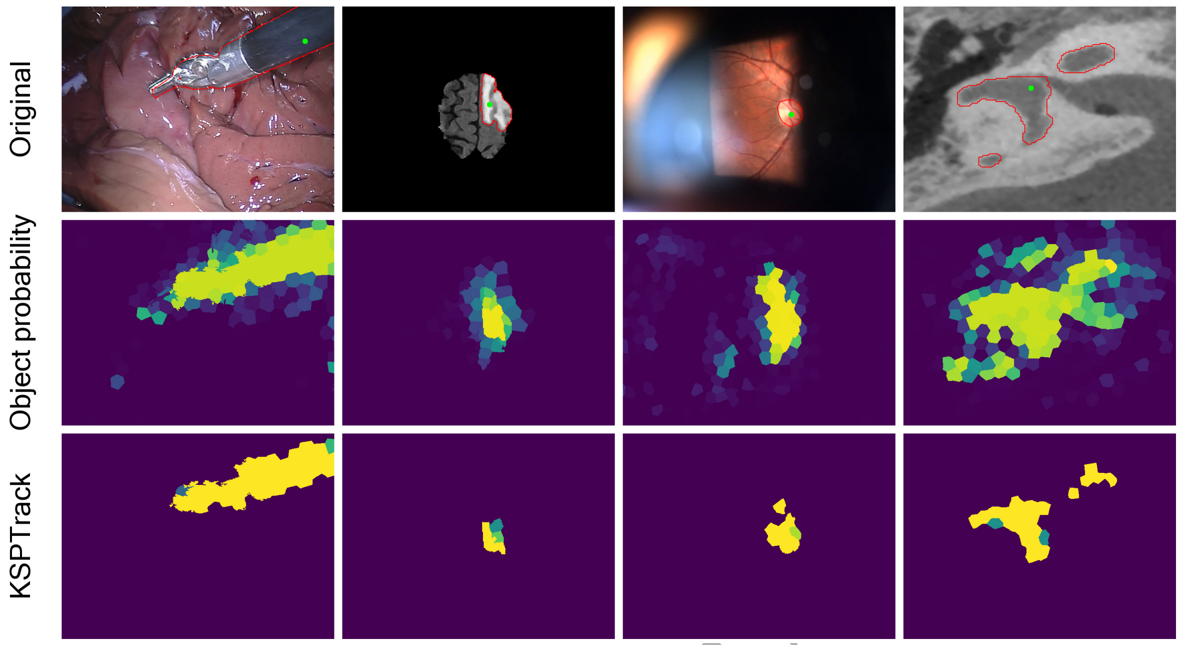


Figure 5: (Top row): Original images from different datasets. Ground truth contour of the structure of interest is depicted in red and the supervised 2D locations are shown in green. (Middle row): ρ_t^n, probability estimates of the object given by our classifier after the final iteration of our approach. (Bottom row): Pixel-wise sum of binary segmentations after each iteration of the KSP optimization. Total number of iterations from left to right are: 3, 4, 3, 2.

set of tracklets in path l. We make the assumption that at the next iteration, the solver would most likely extend found paths given by the previous result. We can then merge tracklets belonging to the same path (*i.e.* concatenate them temporally to form a new tracklet). This brings the practical advantage of reducing the complexity of our problem as the number of edges decreases at each iteration.

In this case, we set the edge cost following the merge to be

$$C_l := \sum_{n,t} C_t^n + \sum_{m,n,t} C_t^{m,n} \tag{10}$$

with tuples (n, t) and (m, n, t) corresponding to edges occupied by path p_l^*. The algorithm then terminates when no new tracklets are added to the set $\mathcal{P}$. Fig. 4(right) illustrates this temporal merging step while the pseudo code of our iterative solver, which we denote **KSPTrack**, is shown in Alg. 1. Fig. 5 shows example frames of how different samples are sequentially added to the positive set, allowing for better classification and KSP solutions.

6. Experiments

The following section details the implementation of our approach, as well as the parameter values used. We then outline the image datasets that we evaluate and the baselines methods used to compare perfor-

Algorithm 1: KSPTrack Algorithm (single direction).

```
Input  : $\mathcal{S}_p$: Initial set of positive superpixels, $\mathcal{S}_u$: Initial set of unlabeled superpixels, $\mathcal{S}$: set of
         superpixels (with associated variables $\boldsymbol{a}$, $\boldsymbol{u}$, and $\boldsymbol{r}$), $\mathcal{G}$: 2D locations
Output: $\mathcal{P}$: Set of K-shortest paths
1  $g \leftarrow$ make_graph($\mathcal{G}, \mathcal{S}_p, \mathcal{S}_u, \mathcal{S}$)                 // As in sections 4.1 and 5.4
2  $\mathcal{P} \leftarrow \emptyset$                                                   // Initialize output set to null
3  $find\_paths \leftarrow$ True                                     // Flag variable (disabled at convergence)
4  while $find\_paths$ do
5      $\mathcal{P}^* \leftarrow$ run_k_shortest_paths($g$)                              // As in  Appendix  A
6      if $\phi(\mathcal{P}) = \phi(\mathcal{P}^*)$ then                       // $\phi(.)$ gives the quantity of superpixels
7          $find\_paths \leftarrow false$
8      else
9          $g \leftarrow$ update_tracklet_costs($g, \mathcal{P}^*$)              // Update $\mathcal{S}_p$ and do as in Sec. 4.1
10         $g \leftarrow$ update_entrance_transition_costs($g, \mathcal{P}^*$)             // As in sections 5.4
11         $g \leftarrow$ temporal_merge($g, \mathcal{P}^*$)                                    // As in Sec. 5.5
12     end
13     $\mathcal{P} \leftarrow \mathcal{P}^*$
14 end
```

mances.

6.1. Implementation and Computational cost

Our **KSPTrack** method is implemented in Python/C++[1]. Using a Linux machine equipped with a Quad-core 3.2 GHz Intel CPU, the construction of both the forward and backward graphs take 15 minutes each. Superpixel segmentation, including the extraction of dense optical flow, requires 10 minutes. Each iteration in Alg. 1 takes 5 minutes, including the training of the classifier and computing the entrance-exit models. The number of KSP iterations varies between 1 and 5 depending on the sequence and the provided 2D locations. A GeForce GTX 1080 Ti GPU and a Keras based implementation of our network was used for our IOS features, taking 3 hours for 20 epochs (at 500 iterations per epoch). In total, our method therefore takes roughly 4 hours for a sequence of 120 frames.

[1]We make our implementation, along with the tested datasets and corresponding manual ground truth segmentations available at www.gazelabel.com

6.2. Selection of Parameters

Table 2 specifies the values of the parameters used in the experiments that follow. Note that these are fixed once and for all, over all experiments and for all datasets. These values were selected empirically so to perform well over all tested sequences.

Symbol	Description	Value
N_t	Approximate number of superpixels per frame	520
M	Number of trees of bagging classifier	500
τ_ρ	Threshold on probabilities of foreground model	0.5
τ_u	Threshold on histogram intersection cost	0.75
τ_{trans}	Threshold on appearance-transition probabilities	0.9
k	Number of clusters for LFDA	5
D	Number of dimensions for LFDA	7
R	Normalized radius of entrance/transition neighborhood	0.05
σ_g	Normalized standard-deviation for prior in feature extraction	0.3

Table 2: Summary of the parameters used in **KSPTrack**.

6.3. Datasets

We evaluate our method on a mixture of datasets consisting of video sequences and volumetric images. Note that the datasets include a variety of different image modalities with a wide range of applications. The singularities of each sequence are given so as to emphasize the flexibility of our method. Note that for each sequence and for all datasets, a single object of interest is present on any give image:

- **Brain:** 4 randomly selected volumetric sequences from the publicly available BRATS challenge dataset (Menze, 2014). Each volume contains a 3D T2-weighted MRI scans of a brains containing a tumor, which we choose as the object of interest. The tested volumes contain $73, 69, 75, 74$ slices each of size (240×240). Tumors have in general a ball-like shape, *i.e.* their radius increases and decreases as slices unfold.

- **Tweezer:** We extract 4 sequences from the training set of the publicly dataset MICCAI Endoscopic Vision challenge: Robotic Instruments segmentation (MICCAI, 2015). Each extracted sequence contains 121 frames and are acquired at 25 fps. The object to segment in each sequence is a surgical instrument, and where each frame is of size (640×480). The tool is piecewise-rigid and is subject to translations and rotations in an otherwise stable environment.

- **Slitlamp:** 4 slit-lamp video recordings of human retinas, where the optic disk must be segmented. The sequences contain $129, 121, 75, 130$ frames of size (680×512), all acquired at 25 fps. The object has a relatively constant shape and texture, but undergoes abrupt translations occasionally. Due to this imaging technique, the background also changes lighting abruptly, with non-global bright beams of yellow and blue light appearing.

- **Cochlea:** 4 volumes of 3D CT scans of the inner ear, where the cochlea must be annotated. The challenge with this dataset lies in the fact that the object of interest branches out in several parts and merges back. Volumes contain $99, 96, 116, 104$ slices of size (300×290).

For the **Slitlamp** and **Cochlea** datasets, we manually segmented the object ground truth on each frame in each image sequence. Manual pixel-wise annotations are publicly available for the **Brain** and **Tweezer** datasets. The complete set of used image sequences and manual ground truths are publicly available [1].

6.4. Generating 2D coordinate locations

Unless otherwise specified, 2D coordinate locations, g_t for each sequence were collected using an off-the-shelf gaze tracker (Eye Tribe Tracker, Copenhagen, Denmark) as in Lejeune et al. (2017). To do this, the tracker was placed beneath a 12.3" tablet roughly $50cm$ away from a user's face. For each recording session, an initial calibration procedure was performed using the inbuilt software of the tracker and validated before all gaze information recordings took place, allowing less than $1°$ tracking accuracy at 30fps.

Gaze recordings were then collected by a domain expert who had been instructed to observe the object of interest throughout the sequence. During the displaying of the video, gaze locations were recorded using a dedicated software[1]. Videos were displayed at 10 fps and 2D locations were then taken to be the average (x, y) coordinates given by the tracker over the corresponding time interval. As such, excluding the initial calibration phase, annotating a 100 frame sequence with a single 2D location per frame took roughly 10 seconds.

6.5. Baselines

To compare our approach to existing methods in the literature, we evaluate the following closest methods:

- **P-SVM**: Patch-based SVM is a transductive learning approach (Vilariño et al., 2007) explicitly developed to use gaze information to produce segmentations when viewing endoscopy video sequences.

- **Gaze2Segment**: This approach used gaze trackers to annotate CT volumes using a saliency map-construction and a Random-Walker to segment the object of interest (Khosravan et al., 2017).

- **EEL**: An expected exponential loss was proposed to learn robust classifiers in a PU learning setting (Lejeune et al., 2017). As in this paper, the method is presented over a variety of image datasets and used a gaze tracker to specify 2D coordinates.

- **DL-prior**: Point location supervision was used to train a CNN while using a strong object prior to provide additional information to the network (Bearman et al., 2016). The method was demonstrated to perform well on natural images.

To compare these methods, we implemented both **P-SVM** and **Gaze2Segment** following their description, while we used provided code for **EEL** and **DL-prior**. **P-SVM**, **Gaze2Segment**, **EEL** and **DL-prior** require approximately $8, 1, 3, 4.5$ hours respectively to process sequences. This computation time is roughly equivalent to our proposed method.

7. Results

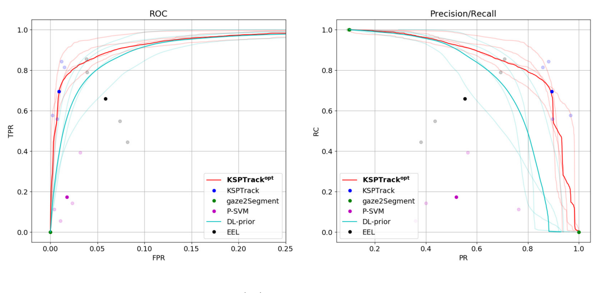
To validate our approach across a wide range of settings, we report the following 5 experiments: (i) The performance of the proposed method is compared to the baseline methods for all sequences in terms of segmentation accuracy; (ii) we compare the performance of a supervised prediction method when trained with ground truth generated by hand or with our approach; (iii) we assess the robustness of our method with respect to the selection of the 2D locations; (iv) we evaluate our IOS feature extraction strategy and compare it to different alternatives; (v) we assess the robustness of our method with respect to outliers and missing 2D locations.

7.1. Experiment 1: Accuracy of produced segmentation

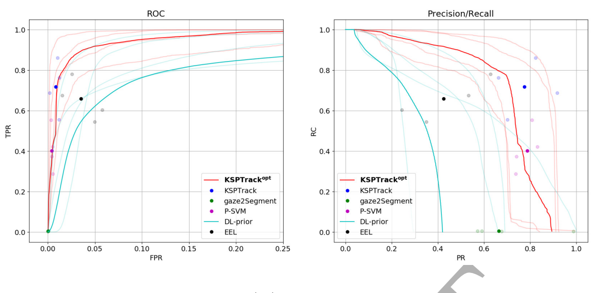
We first compare the accuracy of pixel-wise segmentations produced by our method and the baselines. We illustrate in Fig. 6 the ROC and Precision-Recall curves for each method and for each dataset. In each case, we show the performance on each sequence (in light color) and averaged over each dataset (in bold). To measure segmentation accuracy, we also compute the F1-score for each method on each sequence and report these values in Table. 3.

In addition, we distinguish our method in two: **KSPTrack** and **KSPTrack**^{opt} . In the former, we take the output of the method from the optimization directly. In the latter, we use the probabilities provided by our foreground model after training on the solution of **KSPTrack**. We then select the best threshold to maximum performance. As such, **KSPTrack**^{opt} can be viewed as the optimal segmentation one could hope for if we had a validation set, while **KSPTrack** uses no additional information to infer any threshold.

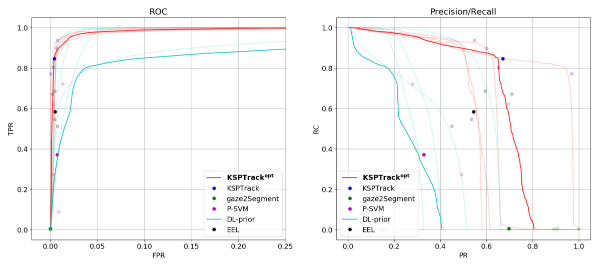
We report substantial improvement over all sequences in both categories. On the **Tweezer** sequence, we improve the best baseline by 196% (**KSPTrack** vs. **P-SVM**) and 12% (**KSPTrack**^{opt} vs. **DL-prior**). On the **Brain** sequences, we improve by 40% (**KSPTrack** vs. **P-SVM**) and 36% (**KSPTrack**^{opt} vs. **DL-prior**). On the **Slitlamp** sequence, we report an improvement of 108% (**KSPTrack** vs. **P-SVM**), and 32% (**KSPTrack**^{opt} vs. **DL-prior**). Similarly, for **Cochlea**, we improve over the best baseline by 370% (**KSPTrack** vs. **P-SVM**), and 113% (**KSPTrack**^{opt} vs. **DL-prior**).



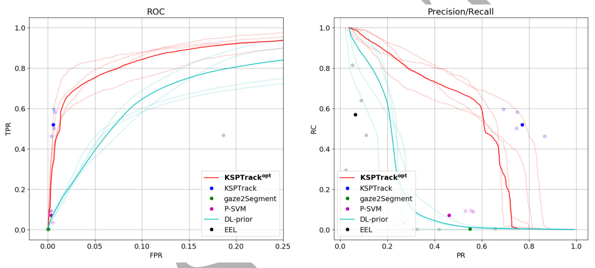
(a) Tweezer



(b) Brain



(c) Slitlamp



(d) Cochlea

Figure 6: ROC and Precision-Recall curves for all types of sequence. In each case, we show the performance on each sequence (in light color) and averaged over each dataset (in bold)

As illustrated in Fig. 8, the results of our methods show improved segmentations compared to tested baselines from a qualitative point of view. To further depict the tracking that our approach produces, Fig. 9 shows how the tracklet association of different superpixels across frames in a **Brain** sequence for both the forward and backward passes of the optimization. Here the tumor is initially small (*i.e.* frame 1), then grows (*i.e.* frames 16, 31 and 46) to ultimately shrink again (*i.e.* frame 61). We can see that certain superpixels are tracked over multiple frames even though the number of regions to segment varies across the frames.

Note that the performance of our approach is bounded by the quality of the superpixels used. In particular, some superpixels may contain both foreground and background pixels which reduces the best case performance of our approach. To quantify the impact of the superpixels on the produced segmentations, we were interested in looking at the F1 score if our approach produced a "perfect" labeling. To do this, we computed the F1 score between the manual ground truth and the set of positive superpixels when positive superpixels are defined by having more than a given proportion of positive pixels (*i.e.* proportions of 0.25, 0.5, 0.75 and 1). That is, a proportion of 1 is when all pixels in a superpixel are in fact positive pixels. Fig. 7 illustrates the relation between these proportions and the average best case F1 score for each datasets. From this, we note that even if our method correctly labeled all superpixels, the F1 score would not be 1. Also, if a strict threshold were to be used to denote positive superpixels (*i.e.* 1 or no negative pixels in a superpixel), our approach would provide near perfect performances.

Type	Method	F1				F1	PR	RC
		1	2	3	4	mean $\pm$ std	mean $\pm$ std	mean $\pm$ std
	KSPTrack^{opt}	0.81	**0.89**	**0.8**	0.74	**0.81** $\pm$ 0.05	0.85 $\pm$ 0.03	0.78 $\pm$ 0.07
	KSPTrack	**0.82**	0.87	0.7	0.67	0.77 $\pm$ 0.08	0.92 $\pm$ 0.03	0.67 $\pm$ 0.13
Tweezer	P-SVM	0.21	0.46	0.2	0.18	0.26 $\pm$ 0.12	0.46 $\pm$ 0.24	0.41 $\pm$ 0.36
	Gaze2Segment	0.18	0.17	0.18	0.18	0.18 $\pm$ 0.0	0.1 $\pm$ 0.0	1.0 $\pm$ 0.0
	EEL	0.78	0.49	0.41	**0.74**	0.6 $\pm$ 0.16	0.55 $\pm$ 0.15	0.66 $\pm$ 0.17
	DL-prior	0.71	0.76	0.79	0.63	0.72 $\pm$ 0.06	0.47 $\pm$ 0.28	0.6 $\pm$ 0.34
	KSPTrack^{opt}	**0.9**	0.7	0.81	**0.63**	**0.76** $\pm$ 0.1	0.72 $\pm$ 0.14	0.81 $\pm$ 0.07
	KSPTrack	0.8	**0.71**	**0.84**	0.63	0.74 $\pm$ 0.08	0.77 $\pm$ 0.11	0.74 $\pm$ 0.1
Brain	P-SVM	0.66	0.5	0.56	0.41	0.53 $\pm$ 0.09	0.78 $\pm$ 0.04	0.41 $\pm$ 0.1
	Gaze2Segment	0.05	0.06	0.1	0.09	0.07 $\pm$ 0.02	0.04 $\pm$ 0.01	1.0 $\pm$ 0.0
	EEL	0.6	0.35	0.7	0.43	0.52 $\pm$ 0.14	0.44 $\pm$ 0.15	0.65 $\pm$ 0.09
	DL-prior	0.43	0.66	0.56	0.6	0.56 $\pm$ 0.08	0.16 $\pm$ 0.27	0.14 $\pm$ 0.25
	KSPTrack^{opt}	**0.72**	**0.73**	**0.74**	**0.93**	**0.78** $\pm$ 0.09	0.7 $\pm$ 0.15	0.91 $\pm$ 0.04
	KSPTrack	0.71	0.72	0.74	0.92	0.77 $\pm$ 0.08	0.7 $\pm$ 0.17	0.9 $\pm$ 0.06
Slitlamp	P-SVM	0.09	0.4	0.65	0.35	0.37 $\pm$ 0.2	0.39 $\pm$ 0.23	0.43 $\pm$ 0.26
	Gaze2Segment	0.02	0.01	0.02	0.02	0.02 $\pm$ 0.0	0.01 $\pm$ 0.0	1.0 $\pm$ 0.0
	EEL	0.54	0.69	0.64	0.48	0.59 $\pm$ 0.08	0.57 $\pm$ 0.09	0.6 $\pm$ 0.08
	DL-prior	0.37	0.48	0.67	0.51	0.51 $\pm$ 0.11	0.0 $\pm$ 0.0	0.0 $\pm$ 0.0
	KSPTrack^{opt}	0.59	**0.66**	**0.69**	0.59	0.64 $\pm$ 0.04	0.64 $\pm$ 0.05	0.63 $\pm$ 0.04
	KSPTrack	**0.69**	0.66	0.65	**0.63**	**0.66** $\pm$ 0.02	0.75 $\pm$ 0.06	0.59 $\pm$ 0.06
Cochlea	P-SVM	0.16	0.15	0.09	0.16	0.14 $\pm$ 0.03	0.42 $\pm$ 0.22	0.32 $\pm$ 0.39
	Gaze2Segment	0.07	0.04	0.09	0.07	0.07 $\pm$ 0.02	0.04 $\pm$ 0.01	1.0 $\pm$ 0.0
	EEL	0.1	0.05	0.18	0.15	0.12 $\pm$ 0.05	0.07 $\pm$ 0.03	0.55 $\pm$ 0.19
	DL-prior	0.33	0.24	0.32	0.33	0.3 $\pm$ 0.04	0.18 $\pm$ 0.1	0.41 $\pm$ 0.24

Table 3: Comparison of quantitative results on all datasets. We report the F1 score for each method on each tested sequence using 4 different 2D gaze sets. In addition, for each sequence type, we give the mean and standard deviation F1, precision (PR) and recall (RC) scores.

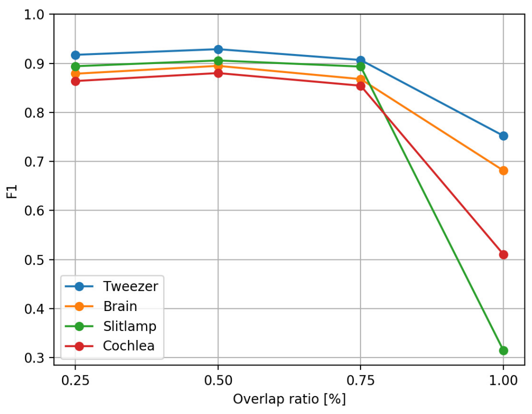


Figure 7: Accuracy of superpixel segmentation (F1 score) for each dataset type when using different proportions of positive pixels within a superpixel to define a positive superpixel. F1 scores are averaged over 4 sequences.

7.2. Experiment 2: Segmentation performance when using generated segmentations

We now investigate the setting where one wishes to train a segmentation classifier to predict unseen sequences using either manually generated ground truths or **KSPTrack** produced segmentations. That is, we wish to assess the bias of our method may induce by comparing the performance of a classifier that is trained with one type of segmentation.

In our experiments, for each dataset, we use 3 out of the 4 sequences to train a standard U-Net using hand-annotated or **KSPTrack** produced pixel wise segmentation. We use the binary cross-entropy as a loss function and perform a leave-one-out prediction (*i.e.* train on 3 sequences and predict on the last sequence). We keep as validation set 5% of the frames belonging to the train sequence. Each model is trained for 40 epochs at 500 iterations per epoch. The model with the lowest validation loss is used in the prediction phase. We compute for each type and each fold the maximum F1-score obtained by both types of segmentations.

Table 4 shows the mean scores over the 4 folds when using the hand and **KSPTrack** segmentations, while Fig. 10 illustrates example predictions. In particular, we report a gap of -10% in prediction on the **Tweezer** dataset. The **Brain** sequence shows a gap of -5%. For the **Slitlamp** sequence, a gap of -16% is obtained. The **Cochlea** sequence shows a gap of -10%.

In general, the fact that our approach provides segmentations that are qualitatively inferior to their hand-based counterpart affects the performance of the classifier in the prediction setup. However, this performance decrease is not overwhelming and depends significantly on the variability of the sequences. In particular, the **Slitlamp** datasets contains a wide range of sequences that appear different from one another. As such, it is not surprising that this dataset suffers the least when compared to the others.

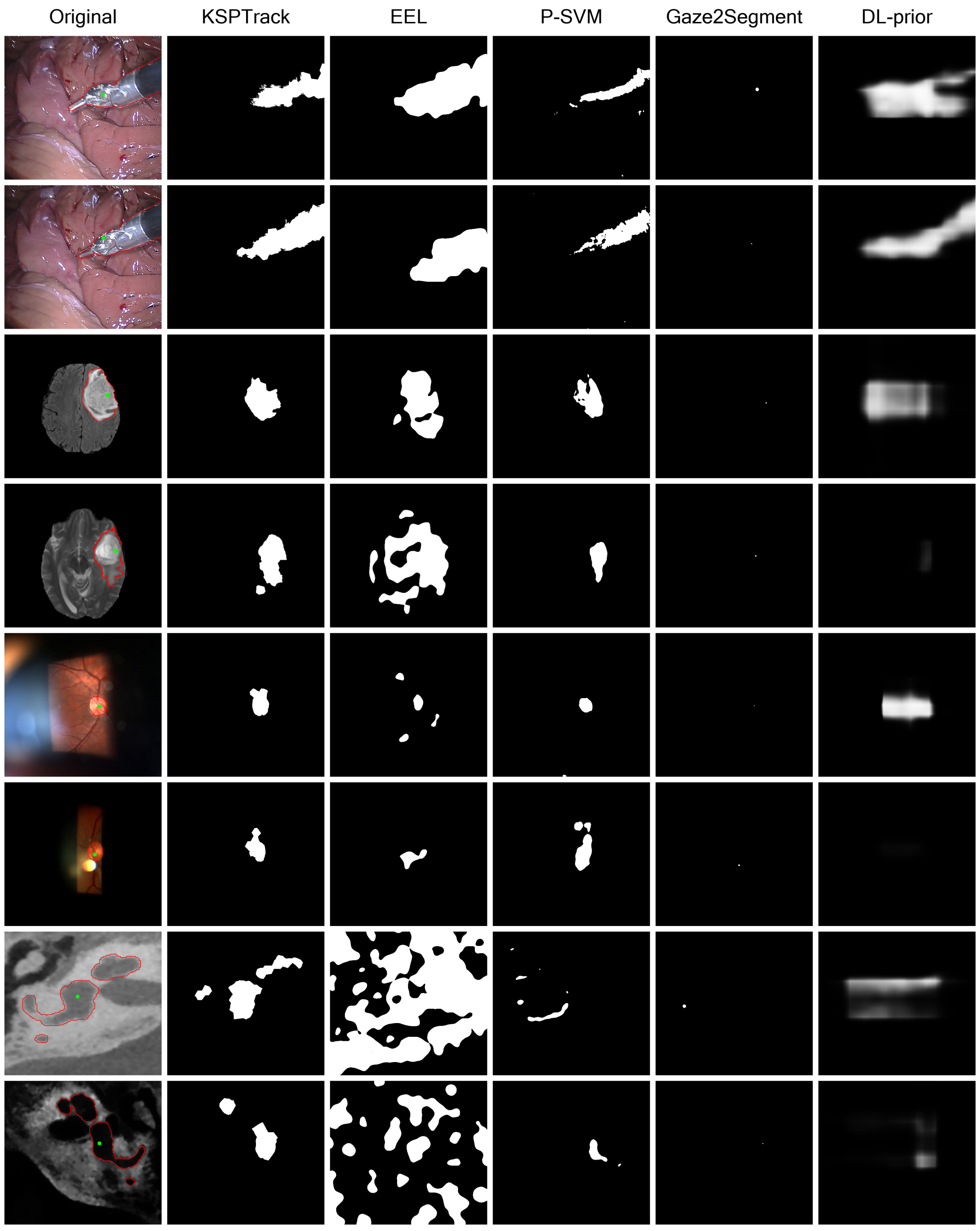


Figure 8: Qualitative results of compared methods on the tested datasets. (First column) Original image. Ground truth contour of structure of interest is depicted in red and the 2D location is shown in green. (Second row onward) Binary segmentation of methods: KSPTrack (Proposed), EEL, P-SVM, Gaze2Segment, and DL-prior.

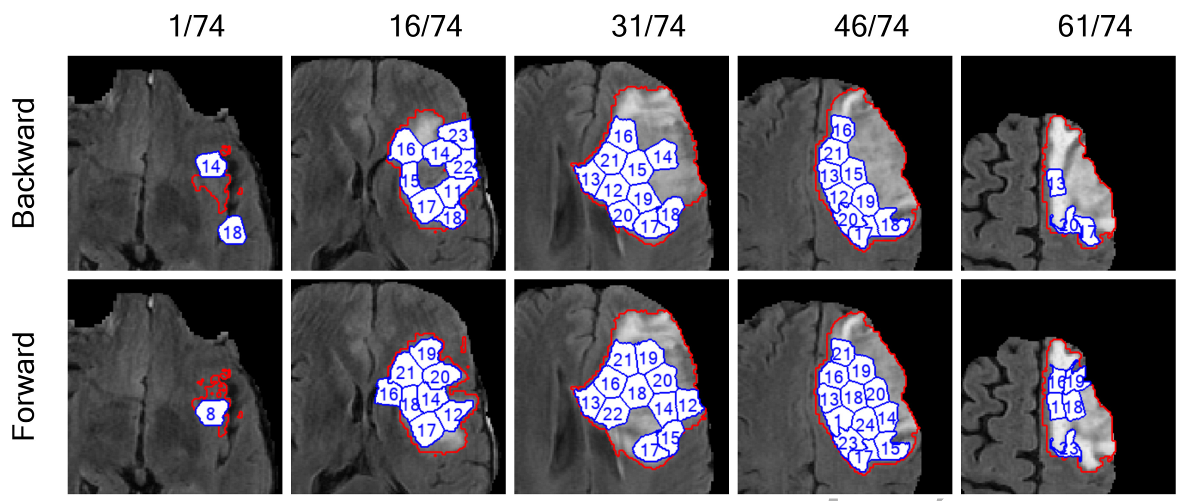


Figure 9: Example paths in the forward and backward tracking directions on a Brain sequence. The ground truth is highlighted in red. Segmented superpixels are highlighted in blue. Numbers within superpixel regions denote indices of paths.

Types	KSPTrack	Manual
Tweezer	0.556	0.611
Brain	0.398	0.421
Slitlamp	0.077	0.092
Cochlea	0.112	0.124

Table 4: Prediction using manual or produced training annotations. For each type, the mean of the maximum F1 scores for the proposed method (KSP) and the mouse-labeled case are shown.

7.3. Experiment 3: Impact of coverage ratio and supervision

When dealing with relatively large objects, it is quite possible that only the most salient parts of the object would be provided as locations, leaving large or homogeneous parts of the object unobserved. This is predominant in the case of the **Tweezer** image sequence where a larger part of the shaft is often distant from any given g_t. In this experiment, we are interested in knowing to what degree the supervised 2D locations play a role in the quality of the segmented objects.

To estimate the impact of this effect, we evaluate the segmentation performance of our method as function of the coverage ratio (*i.e.* ratio of the covered area over the total area of the object). In this setting, we first selected a reference frame where the object appears in its entirety. We then manually select a set of positive superpixels from other frames so as to cover a pre-defined coverage ratio of the object surface. Note that only a single 2D location is still provided on each frame but that it is the entire set that specifies the coverage proportion of the object. Each one of the 4 sequences of the **Tweezer** dataset were assigned 5 sets

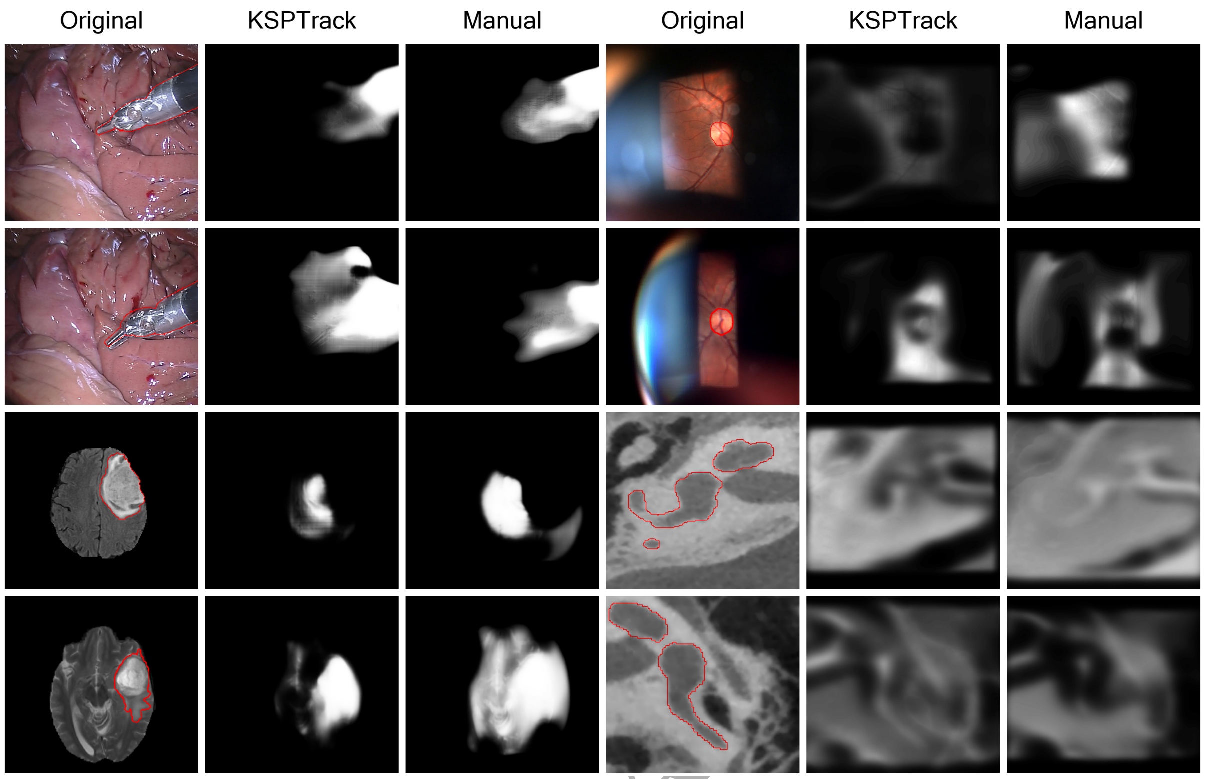
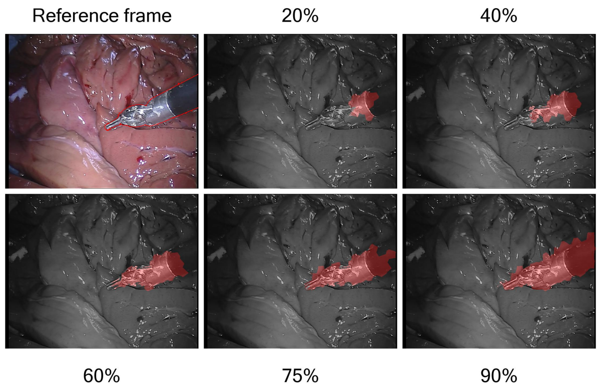


Figure 10: Maximum F1 scores when using KSPTrack or manual segmentations to train a supervised CNN for pixel wise predictions.

of 2D locations, each corresponding to approximately $\{20, 40, 60, 75, 90\}\%$ of the total area of the object (see Fig. 11, Left).

440 We report the F1 score with respect to the coverage ratio in Fig. 11(Right). As our method does not resort to frame-wise filling, we observe that the coverage ratio affects the F1-score as only "seen" regions end up being segmented. As such, attempting to recover the entire object over all frames from a few points remains extremely difficult. We note that even at 90% coverage ratio, the F1 score is of roughly 0.82 and not higher. As can be observed in the bottom right frame of Fig. 11(Left), this is largely due to oversegmentations 445 introduced by the superpixels used.

In addition, Fig. 12 shows the mean segmentation over 5 different sets of provided 2D locations. We observe the largest inter-set variability on the **Cochlea** dataset, where a standard deviation ranging from 5% to 16% is observed. The lowest variability is attained on the **Tweezer** dataset within 2% to 4% range. The latter results can be explained by the fact that the **Cochlea** dataset gives different possibilities as to 450 which branch of the cochlea will be observed. In contrast, the **Tweezer** dataset shows a stable appearance and shape throughout the sequences, thereby limiting the inter-user variability as each set covers roughly



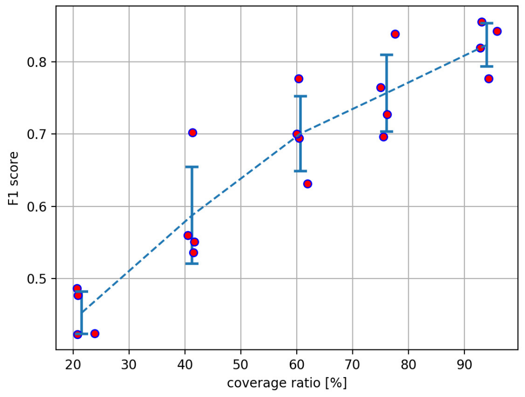


Figure 11: (Left) Graphical examples of coverage ratio for a Tweezer sequence using each of the following ratios: 20, 40, 60, 75, and 90%. (Right) Boxplot of F1 scores with respect to coverage ratio on a Tweezer sequence.

the same regions of the object.

7.4. Experiment 4: Image-Object Features

We also wish to assess the gain in performance of the proposed IOS method with respect to simpler approaches. In particular, we compare produced segmentation performances when using the following alternatives:

- **U-Net:** Using the same architecture as presented in Sec. 4.2 in combination of a simple L^2 reconstruction loss,

$$\mathcal{L}' = \sum_{I_t \in \mathcal{I}} \sum_{k,l} \|I_t(k,l) - \hat{I}_t(k,l)\|^2. \tag{11}$$

  This is similar to our object-image features, but without the object prior given by the 2D locations.

- **OverFeat:** We use the pre-trained CNN of Sermanet et al. (2013). The model is trained in a supervised classification setup on the ImageNet 2012 training set (Deng et al., 2009), which contains about 1.2 million natural images from 1000 classes. In our setup, we use the provided *fast* model, and extract square patches centered on the centroids r_t^n. We set the size of patches so as to include all pixels in superpixel s_t^n. The patch is then resized to (231×231) and fed into the network to give a feature vector of size 4096 at the output of the penultimate layer.

Table 5 shows the maximum F1-score and standard deviation when using different features in **KSPTrack**. Here we show the performance of each feature type over each sequence for each dataset. On average, IOS

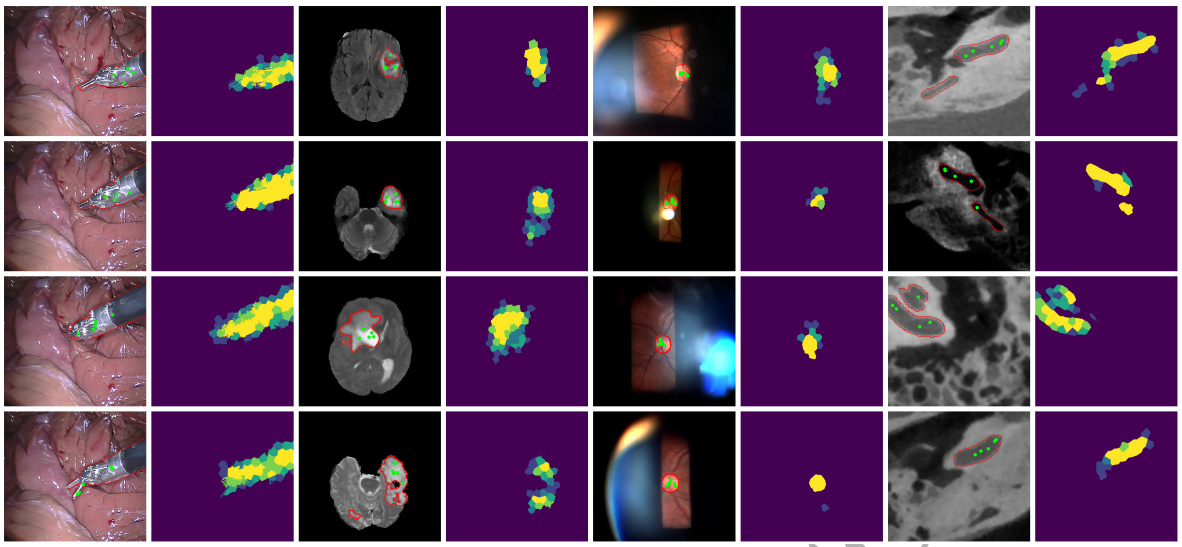


Figure 12: Qualitative results of our approach when using different sets of supervised 2D locations. Columns 1, 3, 5, and 7 show the original image with highlighted ground truth contour in red. The 2D locations are in green. Columns 2, 4, 6, and 8 show the mean of all binary segmentations over 5 sets of 2D locations.

features provides superior performances over alternatives. In some cases (*e.g.* Brain dataset), the standard U-Net or Overfeat features appear to perform better on two sequences in particular. Naturally, the performance variance with IOS features is higher given that they depend on the provided 2D locations.

7.5. Experiment 5: Impact of outliers and missing 2D locations

Next, we look at the effect of the quality and quantity of provided 2D locations, that may vary in practice depending on various factors (*e.g.* speed of object, gaze-tracker calibration accuracy etc.). To do this, we produce noisy versions of the 2D locations used in Sec. 7.1. In particular, we generate for a given outlier proportion $\delta \in \{5, 10, 20, 40, 50\}\%$, three sets of 2D locations. The first samples outliers uniformly on the background, the second samples uniformly on a neighborhood of the object at a distance of 5% (normalized w.r.t. largest dimension of the image), and the last samples at a distance of 10%. Examples for the last two cases are illustrated in Fig. 13a and 13b, respectively. We also show the case where a proportion of 2D locations are missing entirely. Fig. 13c depicts the F1 score with respect to the proportion of outlier or missing locations on the **Tweezer** sequence #1. We report for missing 2D locations a maximum decrease in F1 score of 12% when 40% of locations are missing. This showcases the robustness of our method brought by the global data association optimization. Outliers however impact our method more severely as for incorrect 2D locations of up to 5% decrease performances by 14% with 20% of incorrect locations. Similarly, with errors at a maximum of the 10% distance, the performance decreases by 18% when 40% of the locations are corrupted. For severe corruptions of 50% over the entire background, then the performance drops by 45%.

Type	Features	F1				F1
		1	2	3	4	mean $\pm$ std
Tweezer	Image-object	**0.81** $\pm$ 0.05	0.85 $\pm$ 0.02	**0.82** $\pm$ 0.05	0.75 $\pm$ 0.04	**0.81** $\pm$ 0.04
	U-Net	0.8 $\pm$ 0.06	**0.86** $\pm$ 0.02	0.81 $\pm$ 0.05	**0.78** $\pm$ 0.03	0.81 $\pm$ 0.04
	OverFeat	0.67 $\pm$ 0.03	0.75 $\pm$ 0.02	0.69 $\pm$ 0.06	0.6 $\pm$ 0.05	0.68 $\pm$ 0.04
Brain	Image-object	**0.8** $\pm$ 0.01	0.67 $\pm$ 0.03	0.78 $\pm$ 0.04	**0.64** $\pm$ 0.02	0.72 $\pm$ 0.02
	U-Net	0.8 $\pm$ 0.01	0.67 $\pm$ 0.02	**0.84** $\pm$ 0.0	0.64 $\pm$ 0.02	**0.74** $\pm$ 0.01
	OverFeat	0.77 $\pm$ 0.01	**0.74** $\pm$ 0.01	0.82 $\pm$ 0.01	0.61 $\pm$ 0.02	0.74 $\pm$ 0.01
Slitlamp	Image-object	0.61 $\pm$ 0.07	0.73 $\pm$ 0.02	**0.71** $\pm$ 0.07	**0.83** $\pm$ 0.04	**0.72** $\pm$ 0.05
	U-Net	**0.72** $\pm$ 0.07	0.74 $\pm$ 0.05	0.61 $\pm$ 0.02	0.6 $\pm$ 0.03	0.67 $\pm$ 0.04
	OverFeat	0.51 $\pm$ 0.02	**0.76** $\pm$ 0.02	0.71 $\pm$ 0.03	0.75 $\pm$ 0.03	0.68 $\pm$ 0.02
Cochlea	Image-object	**0.6** $\pm$ 0.05	**0.59** $\pm$ 0.03	**0.64** $\pm$ 0.06	0.63 $\pm$ 0.04	**0.62** $\pm$ 0.05
	U-Net	0.59 $\pm$ 0.03	0.55 $\pm$ 0.03	0.63 $\pm$ 0.05	**0.64** $\pm$ 0.02	0.6 $\pm$ 0.03
	OverFeat	0.52 $\pm$ 0.03	0.5 $\pm$ 0.08	0.54 $\pm$ 0.05	0.58 $\pm$ 0.03	0.54 $\pm$ 0.05

Table 5: Quantitative results of KSPTrack with different feature used on all datasets with five sets of 2D locations per sequence. Mean and standard deviationm F1 scores are given for 2D locations sets.

In general however, we note that the performance remains acceptable up to a 40% outlier proportion. This is explained by the fact that our foreground model effectively penalizes such outliers through its bagging component.

8. Conclusion

In this paper, we presented a framework that allows for pixel-wise segmentation of objects of interest to be generated from sparse sets of 2D locations in video and volumetric image data. In this context, we have provided a strategy that produces an object segmentation by formulating the task as a global multiple-object tracking problem and solving it using an efficient K-shortest paths algorithm. Using an object model estimated from the sparse set of locations, we iteratively refine our solution by progressively improving our object model. To do this effectively, we introduce the use of image-object specific (IOS) features for our purpose and which are generated from an autoencoder that leverages the 2D object locations as a soft prior.

We show in our experiments that our approach is capable of reliably segmenting complex objects of interest over a wide range of image sequences and 3D volumes. Unlike previous methods, our **KSPTrack** method does not assume that the object is of a given size or any information about the background is known. Yet by combining our multi-path tracker and our object model, we achieve superior segmentation results compared to a number of state-of-the-art methods. Beyond this, we show that our results are stable under a number of conditions including the specific nature of the provided 2D locations, even when these are collected at

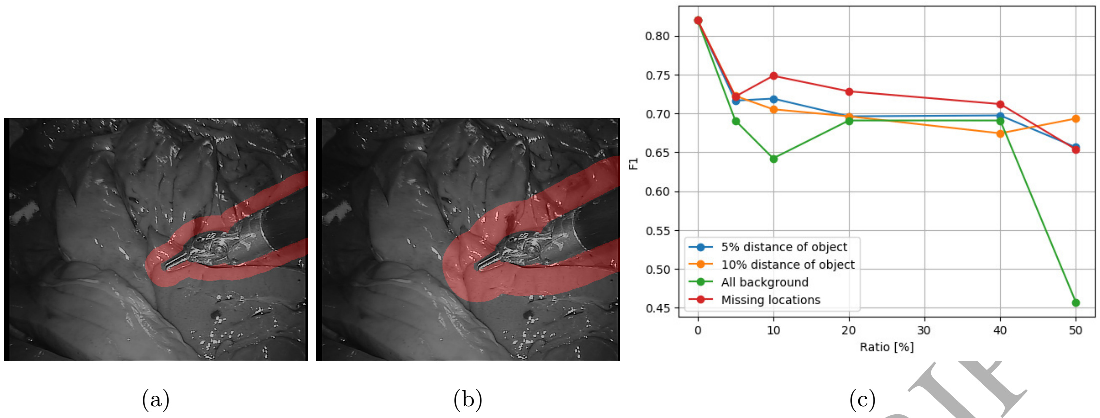


Figure 13: (a and b) Example frames with outlier regions highlighted in red corresponding to distance of 5% and 10%, respectively. (c) F1 scores with respect to the corrupted proportion of provided 2D locations.

framerate using a low cost gaze tracker.

While we demonstrate in our experiments that generated segmentations could be used to train supervised machine learning segmentation methods without suffering too greatly, we show that the performance of our method does depend on the spatial coverage of the provided 2D locations. In the future, we will look to further exploit inter-frame consistency to refine segmentations, in particular to recover fine object details. In addition, so far we have assumed that only a single object is within the data volume. As such, we will look to overcome this limitation and investigate how transfer learning strategies could benefit both feature extraction and segmentation towards this end. Last, in our current set-up, the size of the superpixels used can negatively impact the segmentation produced. To limit the impact of this shortcoming, the use of strategies that refine or merge superpixels to produce more accurate final results will be investigated.

Acknowledgements

This work was supported in part by the Swiss National Science Foundation Grant 200021 162347 and the University of Bern.

Appendix A. Edge-disjoint K-shortest paths

For the sake of completeness, we provide a summary of the edge-disjoint K-shortest paths algorithm implemented in this work while the full version is given in Suurballe (1974). Alg. 2 describes the pseudo code of what follows. Given a directed acyclic graph G, the edge-disjoint K-shortest paths algorithm iteratively augments the set of l shortest paths P_l, to obtain the optimal set P_{l+1}. Starting with $l = 0$, we use a generic shortest-path algorithm to compute $P_0 = \{p_0\}$. In practice, we use the Bellman-Ford's algorithm (Bellman, 1958), which is adequate to cases where edges have negative costs. We then perform two kinds of transformations on G.

- **Reverse operation:** The direction and algebraic signs of edge costs occupied by path(s) of P_l are reversed.

- **Edge costs transform:** The generic shortest-paths algorithm gives for every node the cost of the shortest path from the source. We perform a cost transformation step to make all edges of our graph non-negative. This allows us to then use the more efficient Dijkstra's single source shortest path algorithm (Dijkstra, 1959), which requires non-negatives edge costs as well. In particular, we let v_t^n and w_t^n be the input and output nodes of tracklets $\mathcal{T}_t^n$, and let $L(v_t^n)$ be the cost of the shortest path from the source to node v_t^n. We apply $\forall m, n, t$:

$$C_t^n \coloneqq C_t^n + L(v_t^n) - L(w_t^n) \tag{A.1a}$$

$$C_{t-1}^{m,n} \coloneqq C_{t-1}^{m,n} + L(w_{t-1}^m) - L(v_t^n) \tag{A.1b}$$

$$C_t^{\mathcal{E}_t,n} \coloneqq 0 \tag{A.1c}$$

$$C_t^{n,\mathcal{X}} \coloneqq C_t^{n,\mathcal{X}} + L(w_t^n) - L(\mathcal{X}). \tag{A.1d}$$

On this modified graph, we compute the interlacing path $\tilde{p}_0$. The set P_1 is obtained by *augmenting* P_0 with $\tilde{p}_0$. Concretely, we assign a negative label to the edges of $\tilde{p}_0$ that are directed towards the source, and a positive label otherwise. We then construct the optimal pair of paths P_1 by adding positive edges of $\tilde{p}$ to P_0 and removing negative edges from P_0, as shown on Fig. A.14. The next iterations follow the same procedure. Note that in general, $\tilde{p}_l$ can interlace several paths of P_l.

Hence, our algorithm runs Dijkstra's single source shortest path K times. We therefore have a complexity time linear with K, *i.e.* in a worst case scenario: $\mathcal{O}\left(K(E + V \cdot \log V)\right)$, with K, V, E the number of path sets, nodes, and edges, respectively.

References

Bearman A, Russakovsky O, Ferrari V, Fei-Fei L. What's the Point: Semantic Segmentation with Point Supervision. European Conference on Computer Vision 2016;.

Bellman R. On a routing problem. Quart Appl Math 1958;16:87–90. URL: `https://doi.org/10.1090/qam/102435`. doi:`10.1090/qam/102435`.

Berclaz J, Fleuret F, Turetken E, Fua P. Multiple object tracking using k-shortest paths optimization. IEEE Trans Pattern Anal Mach Intell 2011;.

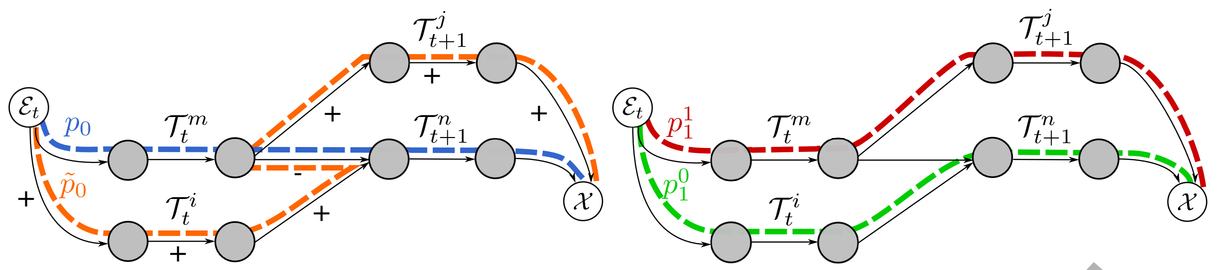


Figure A.14: Illustration of the interlacing and augmentation procedure for K=2. (Left) p_0 is the (single) shortest-path of set P_0. $\tilde{p}_0$ is the shortest interlacing path obtained after inverting the direction and algebraic sign of edge costs of P_0. Positive and negative labels are assigned to the edges of P_0. (Right) The optimal set $P_1 = \{p_1^0, p_1^1\}$ is obtained by removing the edges with negative labels from p_0, and adding positive labels.

Boykov Y, Funka-Lea G. Graph cuts and efficient n-d image segmentation. International Journal of Computer Vision 2006;70(2):109–31.

Bromiley PA, Schunke AC, Ragheb H, Thacker NA, Tautz D. Semi-automatic landmark point annotation for geometric morphometrics. Frontiers in Zoology 2014;11(1):61.

Chang J, Wei D, Fisher JW. A video representation using temporal superpixels. In: IEEE Conference on Computer Vision and Pattern Recognition. 2013. p. 2051–8.

Chapelle O, Schölkopf B, Zien A. Semi-supervised learning. MIT Press, 2006.

Chaudhry R, Ravichandran A, Hager G, Vidal R. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In: IEEE Conference on Computer Vision and Pattern Recognition. 2009. p. 1932–9.

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. 2009. p. 248–55. doi:10.1109/CVPR.2009.5206848.

Dijkstra EW. A note on two problems in connexion with graphs. Numer Math 1959;1(1):269–71. URL: http://dx.doi.org/10.1007/BF01386390. doi:10.1007/BF01386390.

Ferreira PM, Mendonça T, Rozeira J, Rocha P. An annotation tool for dermoscopic image segmentation. In: Proceedings of the 1st International Workshop on Visual Interfaces for Ground Truth Collection in Computer Vision Applications. ACM; 2012. p. 5.

Garcia-Garcia A, Orts-Escolano S, Oprea S, Villena-Martinez V, Rodríguez JG. A review on deep learning techniques applied to semantic segmentation. ArXiv 2017;1704.06857.

Itti L, Koch C, Niebur E. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 1998;20(11):1254–9.

Algorithm 2: K-shortest paths algorithm.

```
Input  : $G$: Directed Acyclic Graph constructed as in Sec. 5.2
Output: $P$: Set of K-shortest paths
1  $p_0 \leftarrow$ bellman_ford_shortest_paths($G$)
2  $P_0 \leftarrow \{p_0\}$
3  for $l \leftarrow 0$ to $l_{max}$ do
4      if $l \neq 0$ then
5          if cost($P_l$) $\geq$ cost($P_{l-1}$) then
6              return $P_{l-1}$
7          end
8      end
9      $G_r \leftarrow$ reverse($G, P_l$)                     // Reverse edges directions and algebraic signs
10     $G_r^+ \leftarrow$ edge_costs_transform($G_r$)          // As in Eq. (A.1)
11     $\tilde{p}_l \leftarrow$ dijkstra_shortest_paths($G_r^+$)   // Returns interlacing path
12     $P_{l+1} \leftarrow$ augment($P_l, \tilde{p}_l$)        // As on Fig. A.14
13 end
```

Jain SD, Xiong B, Grauman K. Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. ArXiv 2017;1701.05384.

Karthikeyan S, Ngo T, Eckstein M, Manjunath BS. Eye tracking assisted extraction of attentionally important objects from videos. In: IEEE Conference on Computer Vision and Pattern Recognition. 2015. p. 3241–50.

Khosravan N, Celik H, Turkbey B, Cheng R, McCreedy E, McAuliffe M, Bednarova S, Jones E, Chen X, Choyke P, Wood B, Bagci U. Gaze2segment: A pilot study for integrating eye-tracking technology into medical image segmentation. In: Workshop on Medical Computer Vision, Internation Conference on Medical Image Computing and Computer Aided Intervention. 2017. p. 94–104.

Kingma DP, Mohamed S, Jimenez Rezende D, Welling M. Semi-supervised learning with deep generative models. In: Advances in Neural Information Processing Systems. NOPUBLISHER; 2014. p. 3581–9.

Kiryo R, Niu G, Du Plessis MC, Sugiyama M. Positive-Unlabeled Learning with Non-Negative Risk Estimator. In: Neural Information Processing Systems. 2017. p. NOPAGES.

Klee V, Minty GJ. How good is the simplex algorithm. Technical Report; WASHINGTON UNIV SEATTLE DEPT OF MATHEMATICS; 1970.

Kojima M, Mizuno S, Yoshise A. A primal-dual interior point algorithm for linear programming. In: Progress in mathematical programming. Springer; 1989. p. 29–47.

Konyushkova K, Sznitman R, Fua P. Introducing geometry in active learning for image segmentation. In: Internationl Conference on Computer Vision. 2015. p. 2974–82.

Lejeune L, Christoudias M, Sznitman R. Expected exponential loss for gaze-based video and volume ground truth annotation. In: Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis, International Conference on Medical Image Computing and Computer Assisted Intervention. 2017. p. 106–15.

Li XL, Liu B. Learning from Positive and Unlabeled Examples with Different Data Distributions. In: European Conference on Machine Learning. 2005. p. 218–29.

Mavandadi S, Dimitrov S, Feng S, Yu F, Sikora U, Yaglidere O, Padmanabhan S, Nielsen K, Ozcan A. Distributed medical image analysis and diagnosis through crowd-sourced games: a malaria case study. PloS one 2012;7(5):e37245.

Menze Be. The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS). IEEE Transactions on Medical Imaging 2014;34(10):1993–2024.

Menze BH, Kelm BM, Masuch R, Himmelreich U, Bachert P, Petrich W, Hamprecht FA. A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. BMC bioinformatics 2009;10(1):213.

Menze BH, Van Leemput K, Lashkari D, Weber MA, Ayache N, Golland P. A generative model for brain tumor segmentation in multi-modal images. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer; 2010. p. 151–9.

Mettes P, van Gemert JC, Snoek CGM. Spot on: Action localization from pointly-supervised proposals. In: Leibe B, Matas J, Sebe N, Welling M, editors. European Conference on Computer Vision. 2016. p. 437–53.

MICCAI . Endoscopic vision challenge, 2015. 2015. URL: `https://endovis.grand-challenge.org/`.

Miyawaki S, Tawhai MH, Hoffman EA, Wenzel SE, Lin CL. Automatic construction of subject-specific human airway geometry including trifurcations based on a ct-segmented airway skeleton and surface. Biomechanics and Modeling in Mechanobiology 2017;16(2):583–96.

Papadimitriou CH. On the complexity of integer programming. Journal of the ACM (JACM) 1981;28(4):765–8.

Papadopoulos DP, Uijlings JRR, Keller F, Ferrari V. Training object class detectors with click supervision. In: IEEE Conference on Computer Vision and Pattern Recognition. 2017. p. 180–9.

Pilch M, Wenner Y, Strohmayr E, Preising M, Friedburg C, zu Bexten EM, Lorenz B, Stieger K. Automated segmentation of retinal blood vessels in spectral domain optical coherence tomography scans. Biomedical Optics Express 2012;3(7):1478–91.

Roberts M, Jeong WK, Vázquez-Reina A, Unger M, Bischof H, Lichtman J, Pfister H. Neural process reconstruction from sparse user scribbles. In: Fichtinger G, Martel A, Peters T, editors. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2011. Berlin, Heidelberg: Springer Berlin Heidelberg; 2011. p. 621–8.

Rohrbach M, Ebert S, Schiele B. Transfer learning in a transductive setting. In: Advances in Neural Information Processing Systems 26. NOPUBLISHER; 2013. p. 46–54.

Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation; NOPUBLISHER. p. 234–41.

Rother C, Kolmogorov V, Blake A. "grabcut": Interactive foreground extraction using iterated graph cuts. ACM Transactions on Graphics 2004;23(3):309–14.

Sadeghi M, Tien G, Hamarneh G, Atkins MS. Hands-free interactive image segmentation using eyegaze. In: Proceedings SPIE, Medical Imaging Computer-Aided Diagnosis. 2009. p. 7260.

Schrijver A. Theory of linear and integer programming. John Wiley & Sons, 1998.

Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. CoRR 2013;abs/1312.6229. URL: `http://arxiv.org/abs/1312.6229`. `arXiv:1312.6229`.

Seyedhosseini M, Ellisman MH, Tasdizen T. Segmentation of mitochondria in electron microscopy images using algebraic curves. In: International Symposium on Biomedical Imaging. 2013. p. 860–3.

Soliman M, Tavakoli HR, Laaksonen J. Towards gaze-based video annotation. In: International Conference on Image Processing Theory, Tools and Applications. 2016. p. 1–5.

Su H, Maji S, Kalogerakis E, Learned-Miller E. Multi-view convolutional neural networks for 3d shape recognition. In: International Conference on Computer Vision. 2015. p. 945–53.

Sugiyama M. Local fisher discriminant analysis for supervised dimensionality reduction. In: International Conference on Machine Learning. 2006. p. 905–12.

Suurballe JW. Disjoint paths in a network. Networks 1974;4(2):125–45.

Sweeney EM, Vogelstein JT, Cuzzocreo JL, Calabresi PA, Reich DS, Crainiceanu CM, Shinohara RT. A comparison of supervised machine learning algorithms and feature vectors for ms lesion segmentation using multimodal structural mri. PLOS ONE 2014;9(4):1–14.

Tzeng E, Hoffman J, Saenko K, Darrell T. Adversarial discriminative domain adaptation. ArXiv 2017;1702.05464.

Vilariño F, Lacey G, Zhou J, Mulcahy H, Patchett S. Automatic labeling of colonoscopy video for cancer detection. In: Iberian Conference on Pattern Recognition and Image Analysis. 2007. p. 290–7.

Vincent P, Larochelle H, Bengio Y, Manzagol PA. Extracting and composing robust features with denoising autoencoders. In: International Conference on Machine Learning. 2008. p. 1096–103.

Welling M. Fisher linear discriminant analysis. Department of Computer Science, University of Toronto 2005;3(1).

Yang S, Mohri M. Online learning with transductive regret. In: Advances in Neural Information Processing Systems. NOPUBLISHER; 2017. p. 5220–30.

Yu H, Qi X. Unsupervised cosegmentation based on superpixel matching and fastgrabcut. In: IEEE International Conference on Multimedia. 2014. p. 1–6.

Yun K, Peng Y, Zelinsky G. amd Samaras D, Berg T. Studying relationships between human gaze, description, and computer vision. In: IEEE Computer Vision and Pattern Recognition. 2013. p. 739–46.

Zhang L, Li Y, Nevatia R. Global data association for multi-object tracking using network flows. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. 2008. p. 1–8. doi:10.1109/CVPR.2008.4587584.

Zhou D, Bousquet O, Lal TN, Weston J, Schölkopf B. Learning with local and global consistency. In: Advances in Neural Information Processing Systems. NOPUBLISHER; 2004. p. 321–8.

Zikic D, Glocker B, Criminisi A. Encoding atlases by randomized classification forests for efficient multi-atlas label propagation. Medical Image Analysis 2014;18(8):1262–73.