University of Ljubljana
Faculty of Computer and Information Science

Jer Pelhan

A semi-automatic video object segmentation method

Bachelor’s Thesis

Undergraduate University Study Programme
Computer and Information Science

Supervisor: Assoc. Prof. dr. Matej Kristan

Ljubljana, 2021
Jer Pelhan

Delno-avtomatska metoda za segmentacijo objekta v videoposnetku

DIPLOMSKO DELO

UNIVERZITETNI ŠTIJESKI PROGRAM
PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: izr. prof. dr. Matej Kristan

Ljubljana, 2021
COPYRIGHT. The results of bachelor’s thesis are intellectual property of the author and the Faculty of Computer and Information Science, University of Ljubljana. For the publication or usage of the bachelor’s thesis results, a written consent of the author, the Faculty of Computer and Information Science, and the supervisor is necessary.

The text is formatted with \textit{B\LaTeX}text editor.
Faculty of Computer and Information Science issues the following thesis:

A semi-automatic video object segmentation method

Subject of the thesis:

Convolutional neural networks have been the driving force behind the advancement of computer vision over the last decade. But these methods require an extremely large amount of accurately annotated data, which is often expensive to obtain by manual annotation. In fact, the lack of large learning and test video data with segmented objects presents a significant hurdle in the development of modern deep video segmentation methods. In your thesis, address the problem of semi-automatic video object segmentation, suggest a new method and implement it in a practical application. Compare the method with related work on the visual object tracking by segmentation problem and on the standard video object segmentation problem.
Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Delno-avtomatska metoda za segmentacijo objekta v videoposnetku

Tematika naloge:

Gonilo napredka računalniškega vida v zadnjem desetletju so bile konvolucijeske nevronske mreže. Te metode zahtevajo izjemno veliko količino natančno anotiranih podatkov, katerih ročna anotacija pa je lahko zelo draga. Ravno pomanjkanje učnih in testnih podatkovnih množic videoposnetkov s segmentiranimi objekti, je pomembna ovira pri razvoju nove generacije globokih modelov za segmentacijo videoposnetkov. V diplomskem delu obdelajte problem delno-avtomatske segmentacije objektov v videoposnetku in predlagajte metodo ter jo implementirajte v aplikaciji za praktično uporabo. Metodo primerjajte s sorodnimi deli na temu segmentacije videoposnetkov za vizualno sledenje objektov in standardnem problemu segmentacije objektov v videoposnetku.
Rad bi se iskreno zahvalil svojemu mentorju izr. prof. dr. Mateju Kristanu za ves čas in potrpežljivost, vse ideje in vso navdušenje, katerega je name prenesel tekom izdelave diplomske naloge. Seveda pa se zahvaljujem tudi svojim prijateljem in predvsem družini, ki mi vedno stoji ob strani.
Abstract

Title: A semi-automatic video object segmentation method

Author: Jer Pelhan

Visual object tracking has recently shifted towards target segmentation, which has increased the demand for video datasets with objects segmented in each frame. However, manually obtaining large segmented video datasets is time-consuming and costly. We address this problem by introducing a Semi-supervised Annotation by Tracking algorithm (SAT), which is specialized for target segmentation specifically for visual object tracking domain with minimal user input. The annotation pipeline is split into two modules. The anchor frame segmentation module predicts a segmentation mask by few (approximately four) user clicks on the object of interest. The module is used to segment the target in a subset of frames, anchors, throughout the sequence. Then a mask propagation module propagates the segmentation masks from the anchors to the in-between frames. On the VOT dataset, SAT achieves an IoU of 73% already at 5% of user annotated frames and outperforms the winner of the DAVIS2020 challenge IVOS [14] and the winner of DAVIS2018 challenge IVS [27] by 40% and 67%, respectively and shortens the annotation time by 98%. On the DAVIS interactive challenges, SAT performs comparably to the state-of-the-art in video object segmentation.

Keywords: convolutional neural network, video object segmentation, video object tracking.
Povzetek

Naslov: Delno-avtomatska metoda za segmentacijo objekta v videoposnetku

Avtor: Jer Pelhan

Na področju vizualnega sledenja se je pred kratkim zaradi hitrega razvoja uveljavilo poročanje lokacije tarče s segmentacijskimi maskami, kar je povečalo zahtevo po popolnoma segmentiranih zbirkah videoposnetkov. Postopek ročne anotacije zbirk videoposnetkov je dolgotrajen in drag, zato v diplomskem delu naslovimo prav ta problem. Predstavimo metodo za pol-avtomatsko segmentacijo objektov na videoposnetku SAT, specializirano za učinkovito anotiranje videoposnetkov vizualnega sledenja. Segmentiranje videoposnetka smo razdelili na dva modula. Prvi modul učinkovito segmentira objekte na pozameznih slikah, saj za oceno segmentacijske maske potrebuje zgolj nekaj klikov na rob objekta. Drugi modul, ki temelji na pred kratkim predstavljenim sledilnikom D3S [22], pa skrbi za prenos mask na preostale slike videoposnetka. Na podatkovni zbirki VOT2020 metoda SAT doseže IoU 73%, z zgolj 5% anotiranih slik, kar je 40% izboljšava v primerjavi z zmagovalno metodo interaktivnega izziva DAVIS2020, IVOS [14], in kar 67% izboljšava v primerjavi z zmagovalno metodo interaktivnega izziva DAVIS2018, IVS [27]. SAT skrajša čas ročnega anotiranja videoposnetka za kar 98%. Na DAVIS interaktivnem izzivu SAT doseže rezultate, ki so primerljivi z naprednimi metodami na področju segmentacije videoposnetkov.

Ključne besede: konvolucijske nevronске mreže, segmentacija videoposnetka, sledenje objektom.
Razširjeni povzetek


Drugi pomemben prispevek diplomske naloge predstavlja aplikacija, ki implementira metodo SAT. Aplikacija je bila zasnovana s ciljem, da bo omogočala intuitivno uporabo in hitro anotiranje. Pomemben faktor predstavlja tudi dejstvo, da jo lahko uporablja kdorkoli, ne samo strokovnjak z računalniškega področja.

Kot glavno evalvacijsko množico videoposnetkov smo si izbrali VOT20 [18]. Zato smo v sodelovanju z 6 anotatorji najprej določili zgornjo mejo natančnosti segmentacij, ki odraža mejo nad katero so vse segmentacijske maske enakovredno kvalitetne. Zgornjo mejo natančnosti smo ocenili na 0.90 s performančno mero IoU, in na 0.94 s performančno mero J&F. Ocenili smo tudi prisotnost šuma na podatkovni zbirki VOT20. Vse segmentacijske maske te zbirke so namreč izdelane ročno, zato so podvržene človeškim napakam.
Natančnost segmentacijskih mask na podatkovni zbirki VOT20 ocenimo na 0.76 IoU in 0.82 J&F.

Metodo SAT smo testirali na množici videosnetkov za vizualno sledenje VOT20 [18] in dosegli 0.73 IoU že pri 5% anotiranih slik z metodo DEXTR, kar je 40% izboljšanje v primerjavi z zmagovalno metodo interaktivnega izziva DAVIS2020 IVOS [14] in kar 67% izboljšanje v primerjavi z zmagovalno metodo interaktivnega izziva DAVIS2018 IVS [27]. Oceno natančnosti anotacij podatkovne množice VOT20 metoda SAT doseže z manj kot 10% anotiranih slik z metodo DEXTR.

Ker naša aplikacija omogoča tudi ročno popravljanje mask s čopičem in radirko, smo simulirali tudi primer uporabe, pri katerem anotator ocene segmentacijskih mask, pridobljene z DEXTR metodo, popravi do popolnega ujemanja. SAT je edina iz med testiranih metod, ki na podatkovni zbirki VOT20 [18] doseže oceno zgornje meje natančnosti anotacij, 0.9 IoU in 0.94 J&F, pri približno 40% ročno anotiranih slik.

Z namenom, da pokažemo nivo generalizacije metode SAT, smo jo testirali tudi na podatkovni množici DAVIS 2017 [33]. SAT se izkaže za primerljiv zmagovalcu interaktivnega izziva DAVIS2018 IVS [27] s performančno mero IoU. SAT preseže rezultat IVS s performančno mero J&F. Poleg tega pa preseže tudi rezultat sorodnih metod [26, 34].

Glavni cilj, skrajšati čas anotiranja, dosežemo. Pokažemo namreč, da se z uporabo SAT aplikacije čas anotacije povprečne sekvence videosnetka z VOT [18] podatkovne zbirke skrajša s približno 400 minut na samo 5 minut časa anotiranja za uporabnika, torej za 98%.
Chapter 1

Introduction

Artificial intelligence is one of the fastest growing trends in recent times, being used in every field imaginable. Mimicking human brains with artificial neural networks is leading to an explosion in speed and performance on various computer science problems. The era of digital cameras marked the beginning of the rapid progress of computer vision and with the advancing technologies, more applications of computer vision appear. Convolutional neural networks [44] achieve better performance than any other computer vision techniques because they automate the process of feature extraction from the image as well as build classifiers tailored for the task. This process used to be done manually, and as we know, images as computer sees them are not that intuitive for a human. As a result, convolutional neural networks are dominant methodology of computer vision.

One of the fast developing fields in computer vision is image segmentation. Essentially, image segmentation is a process of partitioning an image into multiple regions of pixels belonging to the same object. The segmentation mask, separating the object from the background is a result of image segmentation. Video object segmentation (VOS) [29] similar to image segmentation outputs segmentation mask, but on all frames of the video. While a related area of visual object tracking traditionally considered target location as bounding boxes, as object trackers evolved, Visual Object Tracking
Challenge [18] recently introduced reporting the target location as a segmentation mask. All of the above areas have strong impact on development of autonomous robots [17], that are used in many types of industries. The field of video processing and editing is also constrained by the progress of segmentation. The trend towards autonomous vehicles [15] strongly motivates many computer vision tasks, especially the video segmentation and tracking.

A significant issue of convolutional neural networks is that they are data hungry. This means that they require a large amount of diverse training datasets. Specifically, for training video object segmentation methods, per pixel-precise segmentation masks of objects in all frames of many videos are needed. Regardless of how well VOS or VOT methods work in theory, the quality of the predicted segmentation masks in the end depends on accuracy of the training segmentation masks.

The task of manual labelling, creating the segmentation mask on each frame of the dataset, is extremely time consuming. A person annotating the same object in hundreds of frames becomes tired and annotation errors may increase. Repeating this for every single frame of the video, that can be up to 1000 frames long, is a tedious process. Note that performance evaluation of trackers not only requires training datasets, but also accurately segmented test datasets. Besides, the annotated segmentation masks need to be checked and corrected multiple times. Accurate semi-automatic segmentation methods are thus required to reduce the manual work load, while increasing the segmentation accuracy of the training and testing datasets.

Although the gap between video object segmentation (VOS) and visual object tracking (VOT) is closing, the difference between the two is still significant. In video object segmentation, larger objects are segmented out from the background, and their appearance changes only slightly. The sequences also tend to be shorter. In contrast, in visual object tracking, objects are smaller and move faster, occlusions are stronger, and objects are often blurred. Therefore, VOS methods perform poorly on VOT datasets [18, 22]. Since the tracking datasets are longer their annotation involves even more
human work. The proposed work aims to make the annotation procedure of the latter simpler, faster and more efficient.

1.1 Related Work

1.1.1 Semi-Supervised Video Annotation

Semi-supervised video annotation methods provide segmentation masks of the entire video based on first frame segmentation mask created by the user. Many methods [40, 4] use first frame learning to obtain object appearance features that are used for segmenting out the target on entire video sequence. Other less time-consuming methods [30] provide masks with only offline trained networks by propagating previously estimated masks to the next frame statically without using the temporal dimension. Tomakov et al. [38] rely on optical flow with motion network for motion segmentation. To overcome the problem of objects being fixed on successive frames they combined it with an appearance network that keeps appearance evolution of query object. Fully connected object proposals for VOS [35] presents a three step approach where multiple sequence-specific region proposals are generated for each video frame based on the first mask. Segment proposal tracking algorithm is used to label regions of query objects and lastly spatial refinement is applied. For the training and testing procedure of modern trackers very accurate segmentation masks are needed that cannot be estimated with any of stated methods as they do not perform good enough.

1.1.2 Interactive Video Annotation

Interactive video annotation [5] is a step towards solving the need to have more and larger datasets for training and evaluating VOS methods. With interactive VOS methods, users have the ability to repeatedly correct segmentation predictions, until satisfied. This allows user to create higher quality segmentation masks throughout the sequence in less time.
Maninis et al. [24] state that it takes 79 seconds on average to create a segmentation mask. For this reason, interactive VOS methods propose interactive image segmentation techniques that extract query objects from background on pixel level. Some use specific object points [24, 21], bounding-boxes [45, 42] or scribbles (strokes) [20] as user input. Most of interactive VOS methods [14, 27, 7] use scribbles as DAVIS challenge on VOS introduced interactive evaluation procedure [5]. Scribbles lower the image annotation time, but are still more time consuming to create than the clicks on specific object points. Different types of user inputs are shown in Figure 1.1.

Many current state of the art interactive VOS methods [27, 14, 34, 13, 26] are based on two operations that both run on CNNs: (i) interaction or annotation and (ii) propagation or transfer. The interaction operation obtains a mask estimation from scribbles with aggregating features in between. The propagation network in [27, 14] temporally propagates mask bidirectionally using visual memory from feature aggregation module. Chen, Ling et al. [7] start interactive VOS procedure with bounding box tracking and parametric curve fitting. Mask labeling procedure is ran inside cropped regions using region scribbles. Annotators are interactively to correct both bounding box tracks and also estimations of segmentation masks with scribbles. To achieve higher accuracy, they propagate not only masks obtained from user interaction, but also user input - scribbles.

Figure 1.1: Overview of inputs for different interactive image segmentation methods. From left to right: extreme points for DEXTR method [24], bounding box for [42], and scribbles for [7, 20, 14, 27].
1.2 Contributions

In this thesis, we address the expensive and time consuming procedure of manual video annotation. We present SAT (Semi-Supervised Annotation by Tracking), a novel method for annotating object instances with segmentation masks in videos, which is our main contribution. In particular, we split the process into two steps: interactive annotation and segmentation propagation of these masks to all other frames. The segmentation propagation module is based on modified D3S tracker [22] which allows more robust and accurate segmentation of the visual tracking datasets than related state of the art methods. We also developed an video segmentation application that implements SAT with a graphic user interface to speed up the annotation process, which is our secondary contribution.

SAT is extensively evaluated on the most popular segmentation-based visual tracking dataset VOT2020 [18] and SAT outperforms all current state of the art methods. We also showed that SAT works on par with the state of the art on a related video object segmentation problem [5].

1.3 Thesis Structure

The remainder of the thesis is composed of four chapters. In Chapter 2 we briefly describe convolutional neural networks and the main image and video segmentation techniques used in this thesis. In Chapter 3, we present our Semi-Supervised Annotation by Tracking method (SAT). Experimental results are reported in Chapter 4. In Chapter 5, we conclude thesis and discuss future work.
Chapter 2

Methods

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a class of artificial neural networks that are mostly applied to complex data types - images, sound. Thus, CNNs are very popular in computer vision. CNNs consist of neurons that are weighted and have biases. Multiple described neurons stacked in parallel form a layer of the neural network. Convolutional Neural Networks consist of several units of connected operations - layers.

Given set of inputs $X$ (e.g. images) and corresponding outputs $Y$ (e.g. segmentation masks), neural network needs to learn a function $f : X \rightarrow Y$. Neural networks implement a function $f_0 : X \rightarrow Y$ that is a composition of simple functions, i.e.,

$$f_0 = f_1 \circ f_2 \circ ... \circ f_N,$$  \hspace{1cm} (2.1)

that present the layers of the CNN.

Regular neural networks receive an input vector that is transformed through series of hidden layers. The hidden layers are fully connected to all neurons in the previous layer. Just a single neuron in the hidden layer of a regular neural network given inputs of tiny images of size $32 \times 32 \times 3$ – (width $\times$ height $\times$ number of color channels), would have 3072 weights [16]. The stated number is manageable, but it increases with the square of the image width (for
images having same width as height). Simply flattening the image from the matrix would result in complex and computationally demanding operations. Considering this, CNNs exploit constrained image data type, which makes the image processing less computationally intensive. The neurons in CNNs are arranged in three dimensions - same as images (width, height, number of color channels). Every layer consequently transforms the input of a 3D volume into a 3D output. Convolutional Neural Networks consist of several layers of different kinds. In this section we will describe the main types of layers.

2.1.1 Convolutional Layer

The convolutional layer is the first layer and fundamental component of Convolutional Neural Networks. Convolution is an operation that preserves relationships between the pixels on the image which is extremely important in computer vision. The layer can be presented with:

$$Y = f_{\text{act}}(XW + b),$$

(2.2)

with \( Y \) being output tensor, \( X \) being input tensor and \( W \) being weight tensor – trainable convolutional kernel or filter, and \( f(.)_{\text{act}} \) activation function. Bias \( b \) is a tensor of trainable parameters. When trained, it translates values of feature map obtained from convolution.

Convolution in Convolutional Neural Networks

Convolution is a mathematical operation on two functions \( f \) and \( g \) that as a result returns a new function \( f \ast g \):

$$(f \ast g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau.$$  

(2.3)

Returned function expresses how the shape of one function is modified by the other one. Convolution is also a foundation for the convolutional layer in CNNs.
Convolution (2.3) of functions $f(\cdot)$ and $g(\cdot)$ is an integral that expresses the amount of overlap of one function $g(\cdot)$ as it is moved over another function $f(\cdot)$. Equation above is applicable for continuous variables. Due to discreetness of computer world, we use summation instead of integrals:

$$\text{(f \ast g)}[x] = \sum_{n=-\infty}^{\infty} f[n]g[x - n] \quad (2.4)$$

Equation (2.4) expresses discrete convolution in one dimension that is applicable if functions $f$ and $g$ are signals. As images are at least two-dimensional equation above evolves to:

$$\text{(f \ast g)}[x,y] = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f[n,m]g[x - n, y - m]. \quad (2.5)$$

Now, the convolution in equation 2.5 is applicable for two infinite grayscale images or one layer of RGB color images. In image processing function $g$ is called a convolutional filter. Images are finite, thus equation (2.5) also needs to be finite. We define the height and width of filter as $s = 2k + 1$, $k$ being arbitrary natural number:

$$\text{(f \ast g)}[x,y] = \sum_{n=-k}^{k} \sum_{m=-k}^{k} f[n,m]g[x - n, y - m]. \quad (2.6)$$

**Filters in Convolutional Layer**

Convolutional layer contains a set of filters $\mathbf{W}$ that are usually smaller than input image and are trained in training phase. Each of the filters is convolved with the input tensor – image, resulting in activation map. In this process the filter is slid across the image. Dot product between image and filter is calculated at every step. Filters are of the same depth as the input image:

$$Y[x,y] = \sum_{c=0}^{C} \sum_{n=-k}^{k} \sum_{m=-k}^{k} f[n,m,c]g[x - n, y - m, c]. \quad (2.7)$$
Figure 2.1: Convolution operation visualization from equation 2.6. Blue tensor is presenting image or discrete function $f$, and green tensor presenting filter also called kernel or function $g$. The $f[n, m]$ is value at $[n, m]$ position in array. In continuing steps filter slides over all positions in image.

Equation (2.7) presents convolution for color images, with $C$ being number of color channels, $f$ image, and $g$ filter. The $f[n, m, c]$ is value of image pixel at width $n$, height $m$ on color channel $c$, same applies for filter $g$. Activation or feature map of one filter as seen in (2.7) is does not have third dimension. The depth of the output at convolutional layer is obtained by concatenating activation maps obtained from convolution with multiple filters.

Receptive field, or portion of image that inputs to a single neuron is defined by filter size. Size of filter in other words describes the extend of the scope of input data one neuron can be exposed to.

**Controlling Output Size in Convolutional Layer**

As seen in Figure 2.1 convolution procedure causes *loss of information at image borders* as convolution on borders is not defined. Consequently, with filter of width and height of $F = 2k + 1$ it shortens image size to:
Figure 2.2: Padded convolution operation visualization. Blue tensor presents image or discrete function \( f \), and green tensor filter also called kernel or function \( g \). Image is padded with padding of size 1 before the convolution. As shown in 2.9 if we use appropriate amount of padding, image size does not reduce in the process of convolution.

\[
W_{\text{out}} = W_{\text{in}} - (F - 1). \tag{2.8}
\]

In this case, the convolution process places an upper limit on the number of repetitions it can perform. After each repetition, the border of lost information becomes bigger. **Padding** is a term that refers to the amount of pixels added to the edge of the image (as seen in Figure 2.2) in order to prevent information leakage at the image border. Usually pixels with a value of zero are added, but in some cases replicating edges, or mirroring image edges can be used. If we adjust the amount of added pixels \( P \) with respect to filter size \( F \), the output can be of the same dimensions as the input:

\[
W_{\text{out}} = W_{\text{in}} + 2P - (F - 1). \tag{2.9}
\]

Another way to control the output size of convolution is *stride*. Stride \( S \) is a hyperparameter that expresses the step at which the filters are moved across the image. It consequently controls the connectivity of neurons. If the stride is set to \( S = 1 \), the filter is moved for one pixel at each step, as is
the case with regular convolution. Output can easily be downsampled with a stride of \( S = 2 \) or more [16]. Thus, we control the size of the output with:

\[
W_{\text{out}} = \frac{W_{\text{in}} + 2P - F}{S} + 1. \tag{2.10}
\]

All three hyperparameters, filter size \( F = 2k + 1 \), padding \( P \), and stride \( S \) have mutual constraints depending on input image size \( W_{\text{in}} \):

\[
W_{\text{in}} - F + 2P \quad S, \tag{2.11}
\]

has to be an integer allowing neurons to fit symmetrically.

### 2.1.2 Nonlinear Layer and Activation Function

Adding nonlinearities – activation functions in convolutional neural networks is important as it increases the approximation capability of the CNN. Without the nonlinear layer, the CNNs would be just a complex linear function. Nonlinear layers are usually added right after convolutional layer. Several activation functions exist. Each of them takes a single value from feature map and performs a simple mathematical operation on it. We describe the main three activation functions in the following.

**Sigmoid**

Sigmoid is a function that takes an arbitrary real-valued number and assigns a it a number from interval \((0, 1)\) (visualized in Figure 2.3). Specifically the limit in \(-\infty\) is 0 and in \(+\infty\) is 1. It is described with next formula:

\[
\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.12}
\]

The effect of sigmoid on feature maps is to saturate and "kill" gradients, since the gradient is greatest at 0, and is nearly zero at small and large numbers. If the local gradient is small or large it will tone down the gradient and consequently almost no data is transmitted through neuron to its weights. This results in slow to no learning at the tails of the function.
ReLU

Rectified Unit (ReLU) is a nonlinear activation function that solves the sigmoid problem mentioned above. It is a very simple function described by:

\[ f(x) = \max(0, x), \quad (2.13) \]

that can simply be implemented by thresholding all values at 0 (visualized in Figure 2.4). It allows better flows of gradients in backpropagation procedure which leads to faster training. The main problem is that it might "kill" neurons. If once a large gradient flows through a neuron, it can happen that weights are updated in a way that neuron "dies" and is never activated again.

Leaky ReLU

Leaky rectified unit (Leaky ReLU) solves the problem of "dying" neurons. Instead of thresholding input at 0 it is defined by next equation:

\[ f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}, \quad (2.14) \]

with \( \alpha \) being a small constant. It was reported that with leaky ReLU results are not as consistent [16].
Figure 2.4: On left ReLU function and on right leaky ReLU with $\alpha$ set to 0.02.

**ELU**

Exponential Linear Unit is an activation function that it is defined by a function:

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha \ast (e^x - 1) & \text{if } x \leq 0 \end{cases},$$

(2.15)

where $\alpha > 0$ (vizzualized on Figure 2.5). ELU tends to produce more accurate results. ELU is very similar to ReLU, except negative inputs not outputing constant 0s. It is also similar to leaky ReLU, but in contrast to it, it smoothly transitis with negative input, until it reaches $-\alpha$.

**PeLU**

Parametrized Exponential Linear Units (PeLU) is an activation function similar to ELU, but has more parameters controlling different aspects of the function:

$$f(x) = \begin{cases} cx & \text{if } x > 0 \\ a(e^x - 1) & \text{if } x \leq 0 \end{cases},$$

(2.16)

with $a, b, c > 0$. The parameter $a$ controls the saturation with the negative input, $b$ controls the scale of the exponential decay, and $c$ controls the slope in the positive quadrant. If we set hyperparameters $b = c = 1$, we get ELU.
2.1.3 Pooling Layer

Typically, pooling layers are inserted between some instances of convolutional layers after the nonlinear layer. The main task of pooling layer is dimensionality reduction – decreasing width and height of feature maps, also called subsampling. It reduces the size of feature maps by briefing parts of the image as seen in Figure 2.6. This procedure allows to enlarge receptive field for the next convolutional layer. Similar can be achieved with increasing the size of the convolutional filters, but it also increases the number of parameters that must be trained, which is not desired.

After a combination of multiple convolutional and pooling layers, larger structures can be detected with small filters as the receptive field enlarges. Consequently, the same filter size could be used in each convolutional layer of CNN to detect different features of various sizes.

Pooling works with every channel of the input separately – on single depth slices, and similar to convolution carries out specified operation in a specific region of the tensor called window. The window size or spatial extent is one of two hyperparameters of pooling layer. The other hyperparameter is stride.
Figure 2.6: Visualization of most common form of pooling operation – max pooling with window size 2x2 applied with stride of 2.

- as mentioned in the section on the convolution layer it describes the step at which the window is moved across the input. In contrast, this operation is not trainable in training phase, but is explicitly given in advance. This step allows to reduce computational power needed in the network and also prevents overfitting to a certain extend. Different types of operations can be used in pooling layer. Max, and average pooling are most commonly used. The bad effect of using pooling layers is the loss of the information. With pooling multiple pixels are joined resulting in information leakage.

Output size regulation from pooling layer with stride \( S \) and window size \( F \times F \) is:

\[
W_{out} = \frac{W_{in} - F}{S} + 1. \tag{2.17}
\]

**Max pooling**: In area that window covers max pooling chooses the largest value. Consequently it transfers to output stronger activations.

**Average pooling**: Average pooling passes on element that is average of all elements that belong to region of one window size. In contrast with max pooling it extracts features much more smoother.
2.1.4 Upsampling Layers / Decoder

Up to this point, we have dealt with convolutions, downsampling – encoding part of the network. But in segmentation procedure the output map – predicted segmentation mask needs to be of the same dimensionality as the input. Thus, we need to reverse the process of convolution or pooling to achieve desired result knowing that lost information in convolutional and pooling layer cannot be fully restored. We will describe the decoding part of the CNNs.

Upsampling with Interpolation

It is a fixed, non trainable method for increasing size of input channel. Input data is evenly dispersed to the desired output size with spacing in-between the original data.

**Nearest neighbour interpolation**: The missing elements between the input data are replaced with the nearest element of the original data. This method does not introduce new values – uses just the original data values and expands them to size of output.

**Bilinear interpolation**: The unknown values are calculated as a linear combination of the left and right elements and then as a linear combination of the top and bottom data. This method of upsampling introduces new values and consequently smoothness the output.

**Bi-cubic interpolation**: The datapoints between the original data are calculated with cubic function of four neighbouring datapoints in one dimension (e.g. left to right). The procedure is repeated on the other dimension (e.g. top to bottom) on the image. Bicubic interpolation outputs the smoothest interpolated image of all three as is visible in Figure 2.7.

Upsampling with Transposed Convolution

Unlike the upsampling with interpolation procedure, the transposed convolution is not fixed, and the parameters are learned in the training phase.
Figure 2.7: Visualization of the three mentioned interpolations. From left to right: Original data, nearest neighbour interpolation, bilinear interpolation and lastly bicubic interpolation

The transposed convolution learns the upsampling for each problem domain separately depending on the training data. It is the mathematical inverse of convolutional layer operation. Thus, the transposed convolution is based on swapping backward and forward passes of the ordinary convolution in neural networks. Transposed convolution results in checkered patterns over the image and is not oftenly used in practice.

**Atrous or Dilated Convolution**

Atrous convolution provides a simple solution to using transposed convolution. Atrous convolution is a convolution with upsampled filters that is usually followed by a simple bilinear upsampling. The convolutional filters are upsampled by inserting zeros in between the original filter’s data. Thus, atrous convolution does not have more parameters in the learning phase. In [36] they showed that atrous convolution is very successful in dense prediction tasks.

**2.1.5 Fully-Conntected Layer**

Fully-connected layers are the part of CNNs where the classification takes place. The input tensor is flattened to a single column vector, that is passed through the layers. Fully-connected network or FCN is artificial neural network that consists of fully-connected layers.
The main specific of FCNs is that two consecutive layers are fully connected. All the neurons of the previous layer are connected to all the neurons in the next one. Since convolution similarly to FCN has weights but shares them with multiple input elements we can easily enlarge convolutional filter or kernel to size of the input to the FCN. Due to the fact, that filter could not move across the input we can intuitively say that the output would be of size $1 \times 1 \times N$ where $N$ is the number of filters used in this step of convolution.

### 2.1.6 Network training

All artificial neural networks are trained with backward passes to be able to make good predictions, which are inferred by a forward pass through the network.

**Loss Function**

Since network training is an optimization problem that seeks the best prediction we need to define a loss function that will be minimized in the process of the network training. The loss function allows to update the weights of the network according to the value it outputs. The success of the final trained network highly dependent on the selection of an appropriate loss function.

**Cross-entropy loss** between the predicted and ground truth segmentation mask is used as a standard in the field of segmentation:

$$ L(y, \hat{y}) = - \sum_{j \in J} y_j \log(\hat{y}_j) \quad j \in 1, \ldots, |J|, \tag{2.18} $$

with $y_j$ being the correct value for class $j$, $\hat{y}_j$ prediction probability for class $j$, and $J$ number of classes.

**Backpropagation**

Backpropagation is a technique of back-propagating the difference (error) between the prediction and ground truth for updating the weights of the network. At each step of back-propagating the error we calculate how the
change of one value affects the previously stated loss function. We do so by computing the gradient–vector of partial derivatives of the loss function $L$. Gradients are due to chain rule most easily calculated analytically.

**Batch Size**

The batch size defines number of samples propagated through the neural network in one step of training. Faster network training is usually achieved using small batch sizes, since the weights are corrected with backpropagation after the propagation. Another advantage of using small batches is smaller memory usage. However, smaller batches can less accurately estimate gradients. In one epoch (one forward and backward pass of the training samples), multiple batches of data in batch size are sampled.

**Learning Rate**

The learning rate is an important hyperparameter that regulates the impact of the weight corrections in backpropagation. The choice of learning rate is crucial as it influences the speed with which our model converges. If the learning rate is too large, it may never converge as it will miss a local minimum, hence diverge. On the other hand, if we choose too small value, the learning will be very slow.

Ideally, we would have a learning rate that is initially large – allowing fast learning, and gradually gets smaller with each step - allowing convergence.
2.2 DEXTR

In this section we present Deep Extreme Cut (DEXTR) [24] method for interactive segmentation. It allows the creation of object segmentation masks with minimal user input. The person annotating the object has to mark the extreme points of the objects, resulting in fast annotation times.

2.2.1 Extreme Points as User Input

The user input for DEXTR are the extreme points – top, bottom, leftmost, and rightmost pixels of the object to be segmented (see Figure 2.8). According to [28], placing extreme points is much more efficient than drawing a bounding box for several reasons. Firstly, the bounding box obtained from extreme points tends to be more accurate. Secondly, extreme clicks are reported to take 7.2 seconds, which is more than four times less than drawing a bounding box around the object. Moreover, extreme points provide more information since they can report not only bounding boxes but also the four points that lie on the bounding box.

![Figure 2.8: Segmentation masks obtained with DEXTR [24].](image)

2.2.2 DEXTR Overview

From the input (extreme points) the algorithm obtains a bounding box. The queried object is cropped from the image using relaxed bounding box. In this way, the method can obtain information not only about the regions
that certainly belong to the object, but also about regions that certainly do not belong to the object – the background. The cropped part of the object is rescaled and a heat map is created in the size of the new image. 2D Gaussians are placed as activations on the heatmap in the locations of the extreme points. The RGB image input is then concatenated with the heatmap to form a 4-channel input for the convolutional neural network.

Figure 2.9: Overview of DEXTR architecture. The extreme points that are feed into the method are labelled on the input image, and resulting segmentation mask is shown on the output. Image is taken from [24].

The DEXTR network (shown in Figure 2.9) uses ResNet-101 [12] as its backbone architecture. Fully connected parts are removed from the network. The last two downsampling operators (max pooling layers) are replaced by upsampling using atrous convolution in order to maintain sufficient resolution of the feature map. After the ResNet-101, a pyramid pooling module [46] is used to analyse different sized regions of the feature maps, that are upsampled and concatenated into joined feature representation. The pyramid pooling module allows to obtain both local and global context. This means that features from multiple scales are used when classifying a single pixel. The mentioned modifications are part of the Deeplab-v2 [36] network. The network used outputs a map expressing the probabilities whether each pixel belongs to the queried object or not. Stated modifications allow dense predic-
tions, since ResNet-101 was originally designed for image classification and not for segmentation. The Deeplab-v2 [36] model used in DEXTR method is pre-trained on ImageNet and fine-tuned on PASCAL [10].

2.3 The D3S Tracker

In this section we present the Discriminative Single Shot Segmentation Tracker (D3S) [22], which is a foundation of our mask propagation module in SAT (Section 3.1.2). D3S is a template based robust tracker that successfully breached the barrier of reporting bounding box by reporting segmentation mask of the target. Robust target localization is important part of visual object tracking and is one of the reason we choose it. The other reason was already listed, it reports target position in a matter of segmentation mask.

2.3.1 The Architecture

The D3S architecture uses ResNet50 as the backbone. The Resnet50 [12] network is originally designed for image classification and is pre-trained on ImageNet. Thus, D3S uses only first four layers of ResNet50 to extract useful features from the image, since for localization and segmentation purposes it is critical to keep sufficient dimensionality of feature map.

The D3S architecture (see Figure 2.10) consists of two pathways that follow the ResNet50 backbone: (i) the geometrically invariant model (GIM) and (ii) the geometrically constrained Euclidean model. The two models allow robust tracking with robust background discrimination and segmentation of the target. Vice versa GIM and GEM complement each other. Both models receive feature maps extracted with described backbone.
2.3.2 Geometrically Invariant Model

The geometrically invariant model (GIM) is a pathway of the D3s tracker that allows target detection under significant deformation (see Figure 2.11). It is invariant to non-rigid transformations and a broad range of other transformations. It sacrifices spatial relations of the object, but allows target localization and accurate target-background separation in many circumstances. GIM consists of two deep feature vectors: one corresponds to query object and the other to background:

\[ X_{GIM} = \{X^F, X^B\}, \quad (2.19) \]

GIM feature vectors are created in the initialization phase on the first frame. The foreground feature vector \(X^F\) is generated from pixel level features corresponding to query object. Similar applies to background feature vectors \(X^B\) but the query pixels in this case correspond to the near neighbourhood of object.

During tracking, features are extracted using the ResNet50 from search region (four times the size of the target). Feature maps are convolved with 1x1 filter, resulting in reduced dimensionality. Followed by ReLU, 3x3 convolution and another ReLU nonlinearity. Both convolutional filters are learned in the training stage.
Each feature pixel, labeled $y_i$ is compared with all the features from the foreground feature vector $x^F \in X^F$ and the background vector $x^B \in X^B$ by a L2 normalized dot product. The background or foreground similarity at each pixel is obtained as average of top K similarities to features extracted in the initialization phase. Based on the correlation of the per pixel results between the models and extracted features, coarse similarity foreground $F$ and background $B$ channels are generated. The two are joined by softmax into the target posterior channel $P$. The biggest problem of GIM pathway is that if multiple similar instances of query objects exist in the search area, it cannot separate the tracked object from others.

2.3.3 Geometrically Constrained Model

The geometrically constrained model (GEM) is the other model used in the D3S tracker, that reports only target position (see Figure 2.12). Nevertheless, it allows robust and accurate localization which is important. The target is represented by a rectangular filter [9] which is the geometrical constraint of the model. Therefore it is not invariant to non-rigid transformations. After severe target appearance changes (e.g. rotation and deformation) target cannot be accurately or at all localized. Discriminative correlation filters solve
the mentioned problem by adapting target discriminative features. That allows robust localization even in the presence of appearance changes.

![Figure 2.12: Visualization of GEM model of the D3S tracker. Image is taken from [22].](image)

On arrival of new frame in tracking procedure extracted features from search region are convolved with 1x1 filter for dimensionality reduction. Discriminative correlation filters that learned with a pre-defined responses on training images containing object and a part of the background are correlated with features from convolutional layer. After the correlation operation PeLu nonlinearity is applied. Higher maxima of correlation response implies a more reliable match. Thus, where the maximum response is, there our target most likely is. In order to create target location channel, it calculates Euclidean distance from maximum to all pixels in search area. This way target location channel expresses possibility of the target being on each pixel.

### 2.3.4 The Refinement Pathway

Up to this stage, D3S has obtained low resolution encoding of the target location from GIM and GEM pathways. The main task of the refinement pathway is to combine the mentioned channels and upscale the mask estimation to the resolution of the input.

The refinement pathway (see Figure 2.13) takes target location channel from GEM and the foreground and posterior similarity channels from GIM. Concatenated channels are convolved with 3x3 filter followed by a ReLU
nonlinearity, resulting in 64 channels that are ready for upscaling process. Upscaling takes input tensor, scales it with factor of 2, applies 3x3 convolution two times (after each convolutional layer ReLU nonlinearity is applied). Obtained channels are then summed with adjusted features from the backbone corresponding layer of same dimensionality. This upscaling procedure is repeated three times. One additional upscaling is added in which consists only of upscaling and a 3x3 convolution with ReLU. At the end the channels are softmaxed to produce the final segmentation mask.

Figure 2.13: Refinement pathway upcales coarse target locations report maps obtained from GIM and GEM to a segmentation mask of input resolution. Image is taken from [22].
Chapter 3

SAT

In this chapter we present SAT – a Semi-supervised Annotation by Tracking method. We also describe the application that implements SAT into a fast video segmentation tool, that allows easy usage of the method and presents an intuitive graphical user interface for annotation procedure.

3.1 The SAT Method

The main goal of our approach is to efficiently annotate (with minimal user labour) objects in videos. Specifically, to create object segmentation masks for each frame of the desired video. We divided the problem into two stages (see Figure 3.1). In the first stage (Subsection 3.1.1), a temporally sparse subset of images (i.e., anchors) is selected throughout the sequence and individually segmented with low-cost user interactions. Then, in the second stage (Subsection 3.1.2), the segmentation masks are propagated from anchors to all the other frames of the video.
Figure 3.1: Overview of proposed Semi-Supervised Annotation by Tracking method pipeline. Our approach splits annotation procedure into two stages: creation of segmentation mask with anchor segmentation module and transfer of the masks in segmentation propagation module.
3.1.1 Anchor Segmentation Module

To yield segmentation masks of individual frames with minimal user labor, anchor segmentation module of SAT uses DEXTR [24] interactive image segmentation method. The input to DEXTR are 4 extreme points (right-most, left-most, top and bottom) of the object. More points can be added at the edge of the object to get a more accurate mask estimation as seen in Figure 3.2.

Figure 3.2: In case the desired segmentation accuracy is not achieved, additional points in the erroneous area on the object border can be added as input to anchor segmentation module.

3.1.2 Segmentation Propagation Module

The segmentation propagation module is responsible for transferring the segmentation masks from pairs of anchor frames to the frames in between. Concretely, D3S tracker is initialized at each anchor and the selected object is
tracked to the nearest anchor in both directions (see Figure 3.1). D3S is originally initialized on only one frame, but since the neighbouring anchor carries additional appearance information relevant for tracking the object on frames in between the anchors, it is modified to initialize on two successive anchors. Concretely, the GIM module is initialized on starting anchor (i.e. 2∆th frame) and on the next neighboring anchor in tracking direction (for forward tracking that is 3∆th and for backward tracking ∆th frame). GIM feature vector acquires target and background visual appearance (feature vectors) from both first and last tracking frame. This step allows better, more robust segmentation, since these two frames are in general the most contrasting in the whole tracking run (the target appearance in principle changes with time). Anchors cover a variety of target as well as its immediate background appearance. Thus, we use this information to fine-tune D3S [22] to the selected target by 3 epochs.

3.1.3 Automatic Mask Selection Protocol

After the propagation of anchor ground truth segmentation masks $M_{nδ}$ to intermediate frames (Section 3.1.2), each frame has two corresponding segmentation masks: (i) mask from forward tracking run $P_{t}^{\text{forward}}$, and (ii) mask from backward tracking run $P_{t}^{\text{backward}}$. Thus, an algorithm that performs the selection between the two segmentation masks is required.

The trivial mask selection is choosing the mask that is propagated from the closest anchor. One potential issue observed is that the segmentation mask propagated from the anchor closer may be worse than the other corresponding segmentation mask. We conjecture that this happens especially if the object is (partially) occluded or suddenly and considerably changes appearance. Thus, the trivial solution fails.

The mask prediction quality score used by SAT is defined to maximize the probability of selecting the more accurate mask between the two alternatives. A score is computed for each prediction considering two aspects. The first part is the quality of segmentation mask propagated from one an-
anchor to the neighbouring anchor. The second aspect takes into account the target localization certainty from the mask propagation module. The final segmentation mask prediction score is a product of both.

In a preliminary study we noticed that overlap between the anchor ground truth segmentation mask $M_\delta$ and predicted segmentation mask $P_\delta$ propagated from neighbouring anchor reflects the quality of all the segmentation masks produced during single propagation run. The partial score $\Omega_{\delta:2\delta}^{\text{forward}}$ of all intermediate masks between the anchor with user made segmentation mask $M_\delta$ and the neighbouring anchor with user made segmentation mask $M_{2\delta}$ from forward propagation is:

$$\Omega_{\delta:2\delta}^{\text{forward}} = \frac{\exp(\Lambda(M_{2\delta}, P_{2\delta}^{\text{forward}}))}{\exp(\Lambda(M_{2\delta}, P_{2\delta}^{\text{forward}})) + \exp(\Lambda(M_\delta, P_\delta^{\text{backward}}))},$$

(3.1)

where $\Lambda$ is J&F mask overlap score [33] and $P_{2\delta}^{\text{forward}}$ and $P_\delta^{\text{backward}}$ are the forward prediction on $(2\delta)$th frame and backward prediction on $\delta$th frame, respectively.

The second aspect of mask prediction quality score considers localization certainty that is estimated as maximum correlation response of GEM. The final score, $\gamma(P_{\delta+\mu}^{\text{forward}})$ of predicted mask at $(\delta + \mu)$th frame ($\mu < \delta$) is:

$$\gamma(P_{\delta+\mu}^{\text{forward}}) = \Omega_{\delta:2\delta}^{\text{forward}} \Pi_{\delta+\mu}^{\text{forward}},$$

(3.2)

where $\Pi_{\delta+\mu}^{\text{forward}}$ is maximum correlation response of GEM from the D3S tracker at $(\delta + \mu)$th frame in forward tracking direction. The backward tracking score is computed following the same principle.
Figure 3.3: Figure shows J&F overlap of backward propagated and forward propagated masks from anchors at each frame. Below the plot both forward and backward propagated masks for selected frames are shown. Red dots on plot and red squares around the frames indicate which mask from two possibilities is chosen.

Figure 3.3 shows mask selection procedure on a challenging sequence from VOT dataset. Ground truth was set as input to SAT at every 10th frame of the sequence. The proposed method nearly always selects the better prediction between the two segmentation masks. Even if the algorithm in some cases selects worse estimated mask (in terms of J&F), both masks are visually equally good/bad. Examples of this situations are visualized at second and third arrow in Figure 3.3, where the J&F score of selected mask is worse, but segmentation masks are still equally good. In contrast, at last arrow, where higher scoring segmentation mask is selected, it is clearly better.
3.2 The SAT Application

In this section we describe supervised video segmentation application based on SAT, which we plan to make publicly available. The graphic user interface is designed with goal to make its usage intuitive for non-computer-scientists, thus increasing usability of the method.

Figure 3.4: The main view of graphical user interface of the SAT application. It is composed of the file menu bar on the top, and the toolbar that is located on left and bottom border. The toolbar consists of brush adjustment bar marked with red color, the main functionality bar marked with blue color, bar for playing the video sequence in green, the navigation bar in yellow and the bar for adjusting mask transparency in black.
3.2.1 The Graphical User Interface

The Graphical user interface is composed of main view (see Figure 3.4), that allows the user to run main functionalities, choose tools and interact with the application and propagation view (see Figure 3.6) which shows mask propagation in real time.

The File Menu Bar

The File menu bar is located on the upper part of the GUI. It serves for file handling of the application. It allows the user to open a folder with sequence inside or manually save the changes made on the segmentation masks of the sequence.

The Toolbar

The toolbar is located on the left and bottom borders of the GUI (see Figure 3.4) and exposes main functionalities of the application to the user.

1. **The brush adjustment bar** allows the user to choose between brush and eraser using buttons and change its size using a slider.

2. **The navigation bar** allows the user to navigate through the sequence. User can navigate to next successive frame or previous. Our application also allows user to skip 4 frames, navigating 5 frames forward or backward. This feature comes handy when annotating every 5th or 10th frame. User can also navigate to a desired frame using the spin-box located in the middle of navigation bar.

3. **The main functionality bar** lets the user to run DEXTR [24], propagate masks, fine-tune the network, hide the image or hide the mask and autosave option.

4. **Bar for playing the video sequence** allows the user to automatically skip frames using a frames per second rate that can be changed using
the spin-box. The user can quickly scan sequence and find frames with segmentation masks that need to be corrected. Playing can be paused to correct current mask and stopped as desired.

5. **The transparency bar** allows the user to adjust the transparency of segmentation mask.

### The Shortcuts

Several shortcuts are implemented to improve the user experience:

1. **Adjusting size of brush or eraser** is possible by holding `SHIFT` and scrolling the wheel on mouse.

2. **Zoom in/out** on the image can be done using pinch to zoom on your trackpad or by holding `CTRL` and scrolling the wheel on mouse.

3. **Switching between brush/eraser** can be done also with right clicking anywhere on image.

4. Holding down the key ”M” to shows only segmentation mask – hides the image.

5. Holding down the key ”N” allows to hide mask and show only image.

6. Instead of using scroll bars, the user can hold key ”A” and drag the image around.

### 3.2.2 The Video Segmentation Pipeline

Firstly, the annotator opens the directory containing whole sequence of images to be annotated. The first frame of the sequence appears in the GUI. Based on our analysis (reported in Subsection 4.4.3), we recommend to annotate at least every 10th frame manually, or with DEXTR segmentation or best using combination of both – DEXTR prediction and later minor manual corrections before propagating the masks.
Anchor Segmentation

Using DEXTR, the annotator can simply click on extreme pixels (left-most, right-most, top, and bottom pixels) of the object to obtain a segmentation mask (see Figure 3.5). If it is not good enough, the annotator can simply add a point in the erroneous area on edge of object. Or else the annotator can correct the segmentation mask with brush/eraser tool.

Figure 3.5: Figure demonstrates the usage of DEXTR image segmentation method incorporated in our application.

Making manual annotations or just corrections of predictions obtained from DEXTR is important part of application as the final quality of the video sequence masks highly depends on the quality of masks used for the propagation procedure.
Segmentation Propagation

Once a representative amount of images are annotated throughout the sequence (e.g. every 10th image) the user clicks on "Propagate" to propagate masks to all the frames in the video. A window (as shown in Figure 3.6), appears that shows mask propagation in real time. Once the window closes, the procedure is completed.

Figure 3.6: Figure demonstrates the usage of DEXTR image segmentation method incorporated in our application.

Once the video has been segmented, the user can preview the results by using "Play sequence" tool. The user can also set the frames per second rate as desired to find frames with not good enough segmentation masks. The user is to correct the masks and start the procedure of propagating the masks again. If the user adds a representative amount of corrections on different frames it is encouraged to fine-tune the network by clicking on the "Finetune" button. As this procedure takes some time it is automatically ran only the first time of the propagation. The user repeatedly corrects segmentation predictions and propagates them until satisfied.
Chapter 4

Experimental Evaluation

In this chapter we describe implementation details of the SAT method and application. We also present performance measures used to quantitatively evaluate segmentation methods. The experimental evaluation part of the thesis is divided into three parts. Firstly, we measure the ground truth accuracy of the VOT dataset [18]. Secondly, we test SAT and baselines on the VOT dataset. Thirdly, we compare performance of SAT with state of the art on the DAVIS dataset [33].

4.1 Implementation Details

All experiments were conducted on the Visual cognitive systems laboratory GPU server at the Faculty of computer and information science, University of Ljubljana. The server has Intel(R) Xeon(R) Silver 4114 CPU and Nvidia Geforce RTX 2080 Ti 12GB graphics card.

The SAT application is implemented in Python using Pytorch framework. The graphical user interface of the application was implemented with PyQt5. We also used NumPy and OpenCV libraries for matrix operations and image processing.
4.1.1 The Networks Training

\textit{D3S} (GIM pathway and refinement pathway) is pretrained on 3471 training sequences from Youtube-VOS [43]. At initialization, the networks are fine-tuned on anchor images of considered sequence to maximally adapt to the selected target before running the mask propagation procedure. The network is fine-tuned for 3 epochs with 50 iterations and learning rate starting at $1e^{-4}$ with 0.2 decay every epoch. The original crossentropy loss from [22] is applied between the predicted and the user-specified anchor masks.

We consider \textit{DEXTR} [24] pretrained for 100 epochs on PASCAL2012 [10] segmentation dataset. We then trained for 50 epochs on DAVIS2017 [33] train dataset with learning rate of $1e^{-8}$, momentum of 0.9 and weight decay to $5e^{-4}$ with batch size of 5 objects. Training on the DAVIS dataset took approximately 75 hours on a Nvidia GeForce RTX 2080 Ti.

4.2 Datasets and Performance Measures

4.2.1 Evaluation Datasets

Several video object segmentation datasets have been proposed recently. Since our main goal is to simplify segmentation annotating procedure of tracking sequences, we consider the standard visual tracking dataset, the \textit{Visual Object Tracking} (VOT2020) dataset [18], as our main in-domain dataset for analyzing our method. To test generalization capabilities we also test SAT on out-of-domain dataset – the \textit{Densely Annotated Video Segmentation} (DAVIS2017) dataset [33]. DAVIS2017 was also chosen for evaluation because it holds interactive video segmentation challenge and most state of the art video object segmentation methods are tested on it.

The average length of the sequence of VOT2020 dataset is approximately 300 frames. The VOT2020 dataset provides per-pixel segmentation masks for the entire dataset. Tracking objects in Visual Object Tracking dataset are more dynamic and often blurred. Average size of objects in VOT2020 is
approximately 3000 pixels, in contrast, 20000 is average of object size in pixels on DAVIS17 dataset. DAVIS17 dataset provides frames with corresponding per-pixel annotations from videos captured at 24fps with average length of sequences is approximately 70 frames. The publicly available dataset contains 60 sequences that compose train, validation, test-dev, and test-challenge subsequences.

4.2.2 Performance measures

Several performance measures have been proposed in literature to measure how well the predicted mask $P$ matches the ground truth mask $G$.

Jaccard Index

Jaccard index also called intersection over union (IoU) is defined as the ratio between the union and intersection between the ground truth and predicted mask, i.e.,

$$
\text{IoU} = \frac{|G \cup P|}{|G \cap P|} = \frac{|TP|}{|TP| + |FN| + |FP|},
$$

(4.1)

where TP is the number of true positive pixels (marked as object in both $G$ and $P$), FN is the number of false negative pixels (pixels that are marked as object in $G$ but not in $P$) and FP is the number of false positive pixels (pixels that are marked as object in $P$ but not in $G$).

Contour Accuracy $F$

IoU measure is pixel-based measure. This means it only measures the accuracy of pixels overlapping. The very important part of high quality segmentation map is the accuracy at the boundary between the object and background (contour). Hence, in addition to IoU, we use contour accuracy $F_{bnd}$ measure that explicitly emphasises accuracy of the segmentation mask at the border of the object proposed in [29]. In particular, the contour match is calculated using bipartite graph matching, but for faster evaluation we will use morpho-
logical operators as in [29]. This measure applies a precision-recall principle to measure the boundary fit, \( i.e., \)

\[
F_{\text{bnd}} = \frac{2 \times \text{Pr} \times \text{Re}}{\text{Pr} + \text{Re}} = \frac{|TP|}{|TP| + 0.5 \times (|FP| + |FN|)},
\]

where the boundary precision and recall are calculated according to the Algorithm 1.

**Algorithm 1:** Morphologically measure accuracy at boundary

```python
1 function F_measure_Precision_Recall(M, G)
2     gt_bnd = get_boundary(M)
3     pred_bnd = get_boundary(P)
4     // Dilate boundary contours
5     gt_dil = dilate(gt_bnd)
6     pred_dil = dilate(pred_bnd)
7     // Calculate intersection
8     gt_overlap = gt_boundary * pred_dil
9     pred_overlap = pred_boundary * gt_dil
10    // Calculate area of the intersection and compute precision and recall
11    pr = sum(pred_overlap) / sum(pred_boundary)
12    re = sum(gt_overlap) / sum(gt_boundary)
13    return pr, re
```

**J&F**

To take in consideration both region similarity \( i.e., \text{IoU} \) and contour accuracy \( i.e., \text{F}_{\text{bnd}} \) between the predicted segmentation mask and ground truth mask J&F is proposed as a standard measure in [29]. J&F measure is defined as the average between the IoU and the \( \text{F}_{\text{bnd}} \) measure, \( i.e., \)

\[
J&F = \frac{\text{IoU} + \text{F}_{\text{bnd}}}{2}.
\]

(4.3)
4.3 The VOT Dataset Annotation Accuracy

Since we evaluate our segmentation approach on the VOT2020 dataset, we first analyzed the accuracy of the VOT2020 ground truth masks, which have been created, according to VOT [18] by fully manual annotation. To this end, we selected 8 frames with diverse objects (see Figure 4.1) from different sequences to capture the variability of objects in the dataset. We then asked 6 persons to carefully annotate all objects twice in a row. We thus obtained 12 segmentation masks for each frame and overall 96 segmentation masks.

Figure 4.1: Examples of object segmentation masks carefully annotated by annotators in the user study.
We visually verified that all 96 segmentation masks were of highest quality. We also noticed that there were slight differences between the masks corresponding to the same objects. However, due to careful annotation, all the masks should be considered as ground truth and their variation specifies the level under which the differences should be considered negligible for practical evaluation.

We thus calculated similarities in terms of IoU and J&F among all pairs of masks for the same objects. A histogram of these similarities pooled from all objects is shown in Figure 4.2. The average similarities were 0.9 and 0.94 for the IoU and J&F, respectively. Both values are high, which reflects the consistency of the human made annotations in our experiment. In the following we refer to these values as the upper accuracy bound or \( \mu_{\text{IoU}} = 0.9 \) and \( \mu_{\text{J&F}} = 0.94 \).

![Figure 4.2](image)

Figure 4.2: Distribution of similarities between alternative ground truth segmentations in terms of IoU and J&F measures.

We also evaluated the relation between the object sizes and the average annotation similarities of single object in terms of IoU and J&F. These values are shown for all 8 considered objects in Figure 4.3.
We observe that small objects yield lower equivalence bounds than large objects. In general bigger object have higher ratio between whole surface and object contour in pixels. Therefore larger deviations on contour are tolerated for bigger objects. J&F takes into account also accuracy at border of object, and is thus less influenced by size of the object. The difference between maximum and minimum average accuracy is approximately 0.20 and 0.13 for IoU and J&F, respectively (see Figure 4.3). We demonstrate this in Figure 4.4, where we show per-object similarity histograms for two objects of distinctly different sizes. Size of zebrafish is just under 1000 pixels, meanwhile the size of the cat is approximately 11.000 pixels. Bigger deviation of ground truth segmentation masks is seen with the cat, which still scores much higher average J&F due to its size in pixels. Mean J&F on cat is 0.98 meanwhile on zebrafish just 0.89.
Figure 4.4: Histograms show accuracy of user annotations with respect to J&F performance measure, left one for frame of zebrafish sequence, and right one for frame from cat sequence of VOT. On images with segmentation masks all user ground truth segmentaitons are overlaid.
We compared all the masks obtained in the user study with the VOT2020 ground truth masks. Each ground truth mask was compared with all the human made masks of the same frame. We thus estimated the level of segmentation accuracy on VOT, i.e., the performance measure bounds beyond which all alternative segmentation masks should be considered as equivalent, due to limited accuracy of the VOT ground truth masks. The results are shown in Figure 4.5. The average of IoU and J&F measures are 0.84 and 0.89, respectively, and are considered in our subsequent analysis as upper accuracy bounds that can be achieved on VOT without ground truth initialization. These two values are in the following referred as the \( \mu_{VOT} \). Moreover, all predicted masks exceeding \( \rho_{VOT} = \mu_{VOT} - \sigma_{VOT} \) on VOT evaluation should be considered annotated at accuracy beyond the VOT annotation noise. The \( \rho_{IOU}^{VOT} = 0.76 \) and \( \rho_{J&F}^{VOT} = 0.82 \) are the following referred as the VOT annotation accuracy.

Figure 4.5: Figure shows comparison between the ground truth segmentation masks and all user annotations.
4.4 Evaluation on Tracking Dataset

4.4.1 Accuracy of DEXTR Segmentation Method

SAT applies DEXTR as the anchor masks initialization method. We thus performed an experiment to evaluate the DEXTR accuracy on the task of single frame object segmentation. We asked annotators to segment the selected 8 objects from the experiment in Section 4.3 using DEXTR. Initially, the annotators placed 4 points on the object and continued adding points until satisfactory mask was obtained or the mask could no longer be improved by adding new points. On average, approximately 7 clicks (average 6.1) were required. The obtained masks were then compared to the alternative ground truth masks obtained from the user study (Section 4.3). A histogram of distances is shown in Figure 4.6. The average values were 0.86 (IoU) and 0.92 (J&F), which means that the DEXTR masks accuracy is comparable to the $\mu_{\text{VOT}}$.

![Histograms of distances for DEXTR segmentation method](image)

Figure 4.6: With just a few clicks (circa 7) DEXTR achieves an accuracy on par with VOT ground truth.

For further insight, we show the average similarity values in Figure 4.7 for the individual objects in the user study. Note that DEXTR accuracy is above the VOT2020 accuracy in several cases and comparable in the rest. We thus consider DEXTR as a sufficiently accurate tool for obtaining segmentation.
Figure 4.7: Average and standard deviation between 6 annotators and VOT2020 ground truth (red) and the 6 annotators and DEXTR segmentation (blue).

masks on par with those in tracking benchmarks.

4.4.2 Experimental Setup

The following interactive evaluation protocol was used to reflect the practical tracking segmentation annotation pipeline on a given sequence. First, an anchor mask is generated on every $\Delta$th frame and the masks are propagated to the frames in between the frames. Then an anchor mask is created on the frame with worst predicted mask in the sequence and the mask propagation is repeated on the two corresponding intervals (to the left and right from the anchor).

DEXTR needs the control points for estimating image segmentation mask,
thus the control point placement is simulated as follows. At first four extreme points, taken from ground truth segmentation mask, are input to the DEXTR. Then an additional point is added where the margin between the contour of predicted mask and ground truth mask is the most significant. We put a limit of 8 to number of the points. This process generates several segmentation masks per frame. The mask with largest J&F score compared to ground truth is chosen as the final user-selected anchor mask in our user interaction simulation.

SAT is compared with the recent state of the art methods from a related interactive video object segmentation domain, the winner of DAVIS2020 interactive challenge IVOS [14] and the winner of DAVIS2018 interactive challenge IVS [27]. These methods require approximately selecting the target by providing scribbles – a set of curves roughly covering the object area. For generating scribbles we use a standard approach from DAVIS interactive challenge [5] that state of the art methods were trained for.

4.4.3 Quantitative Analysis

Annotation performance on VOT2020 with respect to the percentage of annotated frames in the sequence is shown in Figure 4.8 and Table 4.1. On average SAT required 7 clicks per anchor mask. At already 5% of annotated frames, SAT achieves on average 0.732 IoU, thus outperforming IVOS [14] and IVS [27] by 40% and 67%, respectively. SAT achieves the VOT IoU annotation accuracy $\rho_{\text{VOT}}$ boundary already at 9% annotated frames (0.76 IoU) and reaches the J&F VOT annotation accuracy boundary already at 7% annotated frames (0.82 J&F). Neither IVS nor IVOS comes close to the boundary within 20% of annotated frames.

To remove the influence of DEXTR-based initialization, we repeated the simulated annotation experiment, but this time used the VOT2020 ground truths directly as user-specified masks on anchors. We denote the resulting variant as SAT$^{GT}$ and the results are shown in Figure 4.9. If we assumed that the VOT ground truth masks are completely accurate (which is not the
Figure 4.8: Segmentation accuracy on VOT2020 with respect to the percentage of annotated frames. SAT achieves the VOT2020 accuracy bound $\rho_{VOT}$ at already under 10% of all frames annotated with both measures, significantly outperforming the state of the art.

case according to our analysis), the upper accuracy bound (i.e., the maximal human-level accuracy) would have been reached at annotating approximately 40% of all frames in both performance measures.

Our analysis shows that in video annotating procedure with SAT with manual segmentation on anchors approximately every 3rd frame has to be annotated to produce the segmentation masks that reach the upper anno-

Figure 4.9: Performance of SAT$_{GT}$ with ground truth masks as input.
Table 4.1: Segmentation accuracy on VOT2020 with respect to percentage of annotated frames – Ω. Best results are boldfaced.

<table>
<thead>
<tr>
<th>Ω</th>
<th>Method</th>
<th>IoU</th>
<th>J&amp;F</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>IVS [27]</td>
<td>0.438</td>
<td>0.500</td>
</tr>
<tr>
<td></td>
<td>IVOS [14]</td>
<td>0.524</td>
<td>0.592</td>
</tr>
<tr>
<td></td>
<td>SAT</td>
<td><strong>0.732</strong></td>
<td><strong>0.810</strong></td>
</tr>
<tr>
<td>10%</td>
<td>IVS [27]</td>
<td>0.529</td>
<td>0.598</td>
</tr>
<tr>
<td></td>
<td>IVOS [14]</td>
<td>0.576</td>
<td>0.655</td>
</tr>
<tr>
<td></td>
<td>SAT</td>
<td><strong>0.764</strong></td>
<td><strong>0.842</strong></td>
</tr>
<tr>
<td>20%</td>
<td>IVS [27]</td>
<td>0.606</td>
<td>0.684</td>
</tr>
<tr>
<td></td>
<td>IVOS [14]</td>
<td>0.620</td>
<td>0.702</td>
</tr>
<tr>
<td></td>
<td>SAT</td>
<td><strong>0.793</strong></td>
<td><strong>0.869</strong></td>
</tr>
</tbody>
</table>

Segmentation accuracy bound. If we assume that the minimal time required to segment an object in a frame is 79 seconds as stated in [24]. Then for manually annotating a VOT video sequence with average length of 300 frames, we need approximately 400 minutes. We reduced this time by 60% down to 160 minutes of the user time for creating segmentation masks of the highest quality. Furthermore, using DEXTR-based initialization with 7 clicks, every 10th frame needs to be annotated to produce similar quality segmentation masks that VOT challenge has. And if we assume that annotator uses on average 10 seconds for clicking on object, the time reduces down to 5 minutes of user labour. All together, SAT decreases user labour for creating segmentation masks similar quality to VOT by 98%.
4.4.4 Qualitative Analysis

For additional insights, a qualitative study was conducted on three VOT sequences: (i) the first sequence depicts tracking a rectangular book with out-of-plane rotations and folding, (ii) the second sequence depicts a car with substantial blurring, and (iii) the third sequence depicts an articulated object. All sequences are approximately 150 frames long and every 50-th frame is annotated by the user, placing either 4 extreme points for SAT or scribbles for IVS [27] and IVOS [14].

Results are shown in Figure 4.10. In general IVS [27] has worst object tracking ability, i.e., it completely loses track of the book in Figure 4.10. IVOS [14] also struggles with the book sequence as it begins to drift to the hands of the girl holding the book. SAT on the other hands does not lose track of the object, nor it segments hands as part of the object. Similar occurs with the car sequence, where IVS starts to lose the object frame by frame. With the ice skater sequence IVOS struggles with ice between the legs of the ice skater and labels it as part of the object. IVS copes better with this problem, but still includes some background as part of the person. SAT segments all three objects most accurately.
Figure 4.10: Qualitative segmentation results on VOT2020. First column presents initialization frames.
4.5 Evaluation on Video Segmentation Dataset

We further evaluate SAT on a related problem of video object segmentation, that typically involves shorter sequences with large objects which do not visually change as significantly as in the tracking sequences. While, objects in tracking sequences are more often occluded and blurred and move faster than in the visual object segmentation datasets. In particular, DAVIS 2017 validation dataset [33], standard dataset used for video object segmentation evaluation, was chosen for this experiment.

4.5.1 Experimental Setup

We implemented an experimental setup akin to the DAVIS2017 interactive challenge protocol [5], that allows a limited set of user interaction steps for segmenting the entire video. The first interaction step involves annotating the first frame of the sequence. The method is then run to propagate the annotated mask to all the following frames. In each subsequent interaction step, the frame with the worst mask is selected for re-annotating and the masks are propagated again. The annotation experiment is stopped after eight interactions.

The original DAVIS 2017 interactive challenge simulates interaction on frames in form of scribbles. However, SAT requires control points placement for mask initialization. For that reason, the control points placement is simulated for SAT by taking them from ground truth segmentation masks provided in DAVIS2017 validation dataset. Additional points are added where the largest deviations between the predicted mask and ground truth mask contour is detected. The number of points for creation of one segmentation masks is limited to 8. This process outputs more segmentation masks, and the best scoring mask (in terms of J&F) is chosen as final user defined mask. For fair comparison, we consider the results of the state of the art methods from the respective papers that report results from DAVIS2017 interactive challenges.
4.5.2 Results

Table 4.2 summarizes the video object segmentation experiment results. SAT slightly outperforms the winner of DAVIS2018 interactive challenge IVS [27] and outperforms all state of the art methods except the winner of DAVIS2020 interactive challenge IVOS [14].

<table>
<thead>
<tr>
<th>Method</th>
<th>IoU</th>
<th>J&amp;F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Najafi et al.  [26]</td>
<td>0.548</td>
<td>-</td>
</tr>
<tr>
<td>Heo et al.  [13]</td>
<td>0.725</td>
<td>0.752</td>
</tr>
<tr>
<td>IVS [27]</td>
<td>0.734</td>
<td>-</td>
</tr>
<tr>
<td>IVOS [14]</td>
<td>0.790</td>
<td>0.827</td>
</tr>
<tr>
<td>SAT</td>
<td>0.745</td>
<td>0.775</td>
</tr>
</tbody>
</table>

Table 4.2: Video object segmentation results on DAVIS 2017 validation dataset after 8 initializations. SAT outperforms the winner of DAVIS 2018 interactive challenge in IoU.

We further run the tracking annotation evaluation experiment defined in Subsection 4.4.3. Results are shown in Figure 4.11. SAT performs on par with the IVS [27] in terms of IoU, and it outperforms the IVS in terms of J&F. SAT$\text{GT}$ reaches 0.83 IoU at just 5 fully annotated objects on sequence, meanwhile the winner of DAVIS 2020 interactive challenge IVOS [14] reaches the same performance at 40 scribble interactions per object on sequence. We argue that annotating with SAT$\text{GT}$ is more efficient. Even though SAT$\text{GT}$ needs 6 minutes of user labour (iff segmentation masks are created fully manually, not in combination with DEXTR), and IVOS needs approximately 5 minutes of user labour, searching the worst annotated frame in sequence, interaction and propagation procedure with IVOS has to be repeated 40 times. With SAT$\text{GT}$ the procedure is repeated only 5 times, resulting in easier and more efficient sequence annotation.
Figure 4.11: IoU and J&F performances of SAT method on the validation set of the DAVIS2017 according to number of annotated frames.
Chapter 5

Conclusion

In this thesis, we presented a new method for semi-automatic segmentation of tracking videos that is capable of generating segmentation masks of high quality with minimal user labor which is our first contribution. SAT is the first method designed specifically to work on tracking sequences. These are long sequences containing smaller faster moving objects that significantly change appearance and are often occluded. Our approach of annotation procedure is divided into two stages: (i) image segmentation in which the user annotates anchor frames using interactive segmentation technique or manually, and (ii) segmentation propagation module that transfers annotations to all the remaining frames of the video. Our secondary contribution is developing video segmentation application that implements SAT with graphic user interface that allows usage to a user without any computer science knowledge. We also plan to make SAT application publicly available.

Our main focus was to create a framework that shortens the long annotation procedure for tracking sequences. The goal was achieved, as we showed on the VOT dataset that just by annotating about 10% of the frames initialized with the DEXTR image segmentation method, we achieve sufficiently good segmentation masks throughout the sequence. If the DEXTR predictions are corrected with brush/eraser (simulated with feeding ground truth masks as input) we have shown that we achieve maximal human-level ac-
accuracy within 40% of annotated frames with both measures. At just 5% of annotated frames, SAT achieves 0.73 IoU and outperforming state of the art by a significant margin. It increases performance with respect to IVOS [14] and IVS [27] by 40% and 67%, respectively. We showed that both state of the art methods have worse tracking ability than SAT and tend to lose track of the object in tracking sequences. We also showed that using SAT reduces time used for annotating an average tracking sequence by 98%. In other words, estimated 400 minutes used for manually segmenting out a tracking video sequence reduce down to approximately 5 minutes of user labour. We further evaluated SAT to test generalization capabilities on a related problem of video segmentation that typically involves larger objects on shorter sequences with minimal appearance changes. We showed that in terms of J&F SAT outperforms IVS, the winner of DAVIS 2018 interactive challenge and in terms of IoU works on par.

The application has been tested extensively by the authors and has been already used by third-party researchers for annotation of a large tracking dataset. Based on our own observations and feedbacks, we have identified several possible future improvements. One practical drawback of the annotation application is that after the mask propagation the user has to search throughout the whole sequence to find and correct frames with bad annotations. The user experience could be improved by developing an algorithm for automatic suggestion which frames are very likely to have been poorly segmented. This would alleviate the frame scrolling process. Another beneficial improvement would be modifying the architecture in such way that it would allow manual correction with ”negative” clicks. This way it would be possible to exclude wrongly segmented part of background. These are some of the topics we would like to pursue in our future work.
Bibliography


