UNIVERSITY OF SCIENCE

ADVANCED PROGRAM IN COMPUTER SCIENCE

NGUYỄN TOÀN ANH – TRÌNH XUÂN SƠN

VIDEO OBJECT SEGMENTATION USING VISUAL SALIENCY AND OPTICAL FLOW

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

HO CHI MINH CITY, 2017

UNIVERSITY OF SCIENCE

ADVANCED PROGRAM IN COMPUTER SCIENCE

NGUYỄN TOÀN ANH	1351001
TRÌNH XUÂN SƠN	1351058

VIDEO OBJECT SEGMENTATION USING VISUAL SALIENCY AND OPTICAL FLOW

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

THESIS ADVISOR

ASSOC. PROF. TRẦN MINH TRIẾT

HO CHI MINH CITY, 2017

COMMENTS OF ADVISOR

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

COMMENTS OF REVIEWER

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

ACKNOWLEDGEMENT

First and foremost, we would like to express our most sincere thanks to our instructor, Assoc. Prof. Trần Minh Triết. We are utterly grateful and indebted to him for his invaluable instructions, extremely helpful guidance, and continuous morale encouragement. Thanks to his extraordinary experience in computer science and teaching, we acquired new, interesting, and valuable knowledge in the fields of computer vision and machine learning. Furthermore, he is the source of inspiration that leads us through the obstacles during the time working on our thesis. Without his help, we would not have been able to complete this thesis.

We would also thank to Mr. Lê Trung Nghĩa for his guidance in visual saliency and other team members for their collaboration in participating in DAVIS Challenge 2017.

We are also grateful to all of our lecturers of the Advanced Program in Computer Science. Thanks to their great dedication to us, we have a strong foundation in computer science and acquire knowledge of the current state of the technology market. Owing to them, we are capable of learning and applying new technologies more effectively.

We also would like to thank some of our classmates for always being there when we need their help.

Last but not least, we would like to thank our families for offering their encouragement and financial aids when we need them. We are forever grateful for their unconditional love.

UNIVERSITY OF SCIENCE
ADVANCED PROGRAM IN COMPUTER SCIENCE

Thesis Proposal

(submitted by students)

Thesis title:

VIDEO OBJECT SEGMENTATION

USING VISUAL SALIENCY AND OPTICAL FLOW

Thesis advisor: Assoc. Prof. TRAN MINH TRIET

Students: Nguyen Toan Anh (1351001) – **Trinh Xuan Son** (1351058)

Type of thesis: *Research with demonstration application*

Duration: From *Jan 15th, 2017* to *July 30th 2017*

Contents of thesis:

I. Overview:

Video object segmentation has a number of applications in real-life scenarios, including tracking object given limited info, automatic video matting systems and robotic technologies. However, effective implementation of these applications is a challenge in computer vision. Unlike human vision, which can extract a huge amount of information from an image within a considerably short time, the exceptional ability to take out a fine cut of an object from its background is not easily modeled in computer vision. As a result, there has been various types of approaches to object segmentation to better harness the potential of the problem, and deep learning plays a huge part in those methods. The authors choose object saliency method as a fresh approach to this problem. The reason for the authors' choice is because throughout the authors' study of the matter of object segmentation, object saliency stands out as a new apprach to object segmentation

task. With a good object saliency algorithm, it can be used to extract fairly accurately the object from its background.

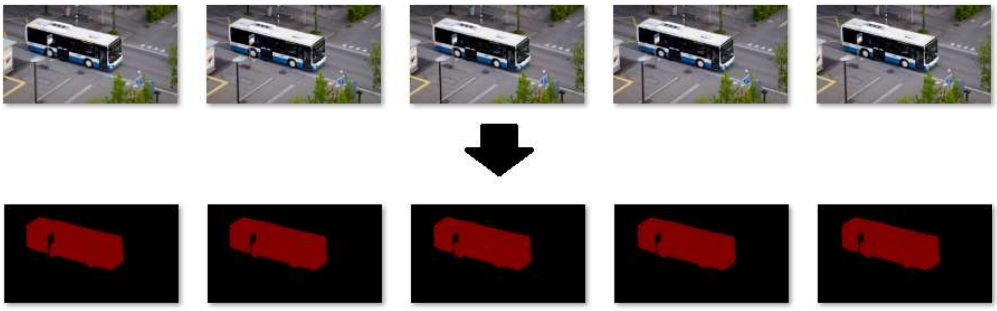


Figure 1. Motivation figure, DAVIS dataset 2016

Figure 1 is reused from the motivation/concept figure from DAVIS dataset 2016. The authors' motivation is that from an image, the authors can cut the object out of its scene as accurately as possible. Figure 1 presents the most idealistic achievement for the authors' study.

II. Background and Related works:

- Yi-Hsuan Tsai, Ming-Hsuan Yang and Michael J. Black present a new method for video object segmentation called Object Flow. They create an algorithm that processes video segmentation information and optical flow estimation simultaneously.
- Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui and Cordelia Schmid propose an approach to handle large displacement in optical flow called Deepflow. Within this method, they blend a descriptor matching algorithm to help improve the performance on fast motions.
- Joseph Redmon et al. present a new approach to object detection called You Only Look Once (YOLO). They treat the object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network plays a part as bounding boxes and class probabilities predictor straight from the full image within one assessment.

– Nian Liu and Junwei Han propose a novel convolutional neural network – based end-to-end deep hierarchical saliency network (DHSNet) for detecting salient objects. Initially, DHSNet makes a rough guess about global features. Then, they use a hierarchical recurrent convolutional neural network to refine the detail of saliency maps thanks to local context information.

III. Objective

The main objective of this thesis is to propose a simple approach to take advantage of state-of-the-art methods to apply to video object segmentation problem using visual saliency with mask propagation and reappearance detection. The output of the approach should be a ready-to-use matte that can be applied to compose two scenes serving automatic matting purpose. Ultimately, the authors create a demo application for demonstrating the ability to apply object segmentation to video matting.

IV. Tasks

- Study the basic knowledge about Machine Learning: regressions (linear regression, logistic regression), neural networks (fully connected neural network, convolutional neural network), support vector machines, k-mean clustering and optimization techniques (gradient descent, normal equation).
- Conduct study on various approaches on object segmentation problems. That is when the authors come across object salience papers.
- Propose a novel method for Mask propagation and Reappearance Detection for Video Salient Object Segmentation. The method is based on three key ideas:
    - Take advantage of no-mask salience method and use mask to guide the final segmentation result.
    - Use flow-based tracking algorithm to perform object tracking.
    - Use object detection algorithm to detect disappearance and reappearance.
- Propose using the result of the experiments to solve automatic video matting problem via a demo application.

Research timelines:

Start	End	Tasks
January 15th, 2017	February 28th, 2017	Studying Machine Learning, particularly Neural Networks and related techniques.
March 1st, 2017	March 7th, 2017	Conduct study on various approaches on object segmentation problems.
March 8th, 2017	May 31st, 2017	Propose a new method to object segmentation problem and conduct experiments to evaluate.
June 1st, 2017	July 7th, 2017	Propose using the result of the experiments to solve automatic video matting problem via a demo application.
July 8th, 2017	July 30th, 2017	Complete the application and writing the thesis.

Approved by the advisor

Signature of advisor

Assoc. Prof. Trần Minh Triết

Ho Chi Minh city, January 24th, 2017

Signature(s) of student(s)

Nguyễn Toàn Anh

Trình Xuân Sơn

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Object segmentation regards finding objects and making a boundary around every object as tight as possible. With this ability, its applications include tracking objects of interest given limited info. Throughout the authors' study of the matter of object segmentation, object saliency stands out as a fresh approach to object segmentation task. With a good object saliency algorithm, an object can be it can be fairly accurately extracted from its background.

Video matting is combining the foreground from one video to the background of another. For example, extracting the bear from a zoo's video and add to the beach from a tourism video. Video matting requires certain preparations before the composition process can be carried out. More detailed and complex composition requires more complicated preparation. One of the most standard ways utilized in video matting is chroma key compositing. Chroma keying is the technique that removes all areas of an explicit color out of the video. the most commonly used colors for this method are blue and green. The disadvantage of this process is that the foreground could get green/blue tint if the lighting is not set up correctly.

Prelimary experimental results show that in the ideal case, visual saliency used for segmentation can rival state-of-the-art methods, with Jaccard and Boundary mean scores reaching 0.760 and 0.722 respectively for the *val* set of DAVIS 2016 dataset, higher than the third-ranked method as of July 2017 (VPN with Jaccard mean 0.702 and Boundary mean 0.655). Therefore, in this thesis, the author propose a simple approach to take advantage of state-of-the-art salient object detection methods to solve object segmentation problem. In this approach, the authors also use an optical flow computation and object detection to exploit the temporal data to enhance the segmentation result. The proposed method achieves results comparable those of the top ranking methods for video object segmentation on the val set of DAVIS 2016 dataset as of July 2017, with Jaccard mean score being 0.719 and Boundary mean score being 0.678.

Ultimately, the authors produce a demo application for demonstrating the proposed method in video matting.

Chapter 1

Introduction

✍ *Chapter 1 presents the definition and importance of object segmentation and visual saliency along with their applications in real life. Chroma key compositing method for video matting is also discussed. The chapter then describes the motivation of this thesis: finding a fresh solution to object segmentation problem. Objectives and detailed work are presented next, listing out the goals of the authors' work and how to achieve them. Finally, Chapter 1 wraps up with the outline of the content of the whole thesis.*

1.1. Overview

1.1.1. Semantic Segmentation

Semantic segmentation is the process of partitioning the input image into different meaningful segments that belong to any of the predetermined classes, turning the image into a representation that is easier to analyze [1]. The core difference between semantic segmentation and traditional image segmentation, i.e. the partitioning of the image into sets of pixels (super-pixels) of similar characteristics, is that the segments must now have meanings. For traditional image segmentation, the output is a set of super-pixels in which individual pixel contains similar intensity, color, or other characteristics. The purpose of this is to tell the location of objects and boundaries, so the segments themselves are merely patches of pixels sharing certain characteristics. With semantic segmentation, on the other hand, what we need to do is dividing the image into regions with specific meanings. Thus, the pixels in each segments are now no longer share just their own characteristics. If our set of classes contains "person", but not "shirt" or "pants", then a person wearing a white shirt and blue jeans appearing in an image must be returned as a whole segment labeled "person", not unclassified patches of white (from the shirt), gray (shadow on the shirt), blue (from the pair of jeans), dark blue (the shadow on the jeans), and such. Therefore,

the task is much more difficult as it can be hard to tell, even for humans, if certain pixels belongs to one class or another. For instance, for an image of a snowdrop amongst the snow, it is not easy to determine if a white pixel belongs to the snowdrop's petals or the snowfield.

Scene parsing is a type of semantic segmentation problem. The goal of scene parsing is to segment the whole image into distinct regions belonging to different semantic categories [2]. Figure 1.1, taken from the paper of Zhao et al., presents the results of two state-of-the-art methods for scene parsing: FCN [3] and PSPNet [4]

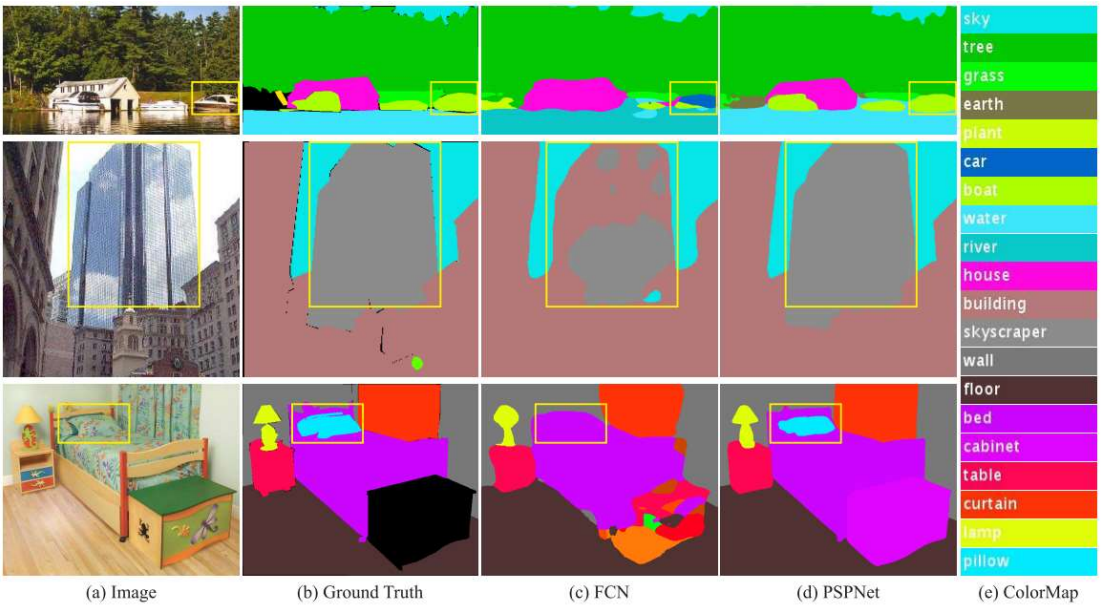


Figure 1.1. Comparison of scene parsing methods FCN [3] and PSPNet [4]

In contrast, object segmentation, another semantic segmentation problem, cares about the partitioning of a single object, or objects, in the scene while background information is generally discarded. Figure 1.2 shows the results of several methods participating in the 2016 DAVIS Challenge on video object segmentation, in which it can be seen that only the swan is segmented because it is the object of interest. The problem is somewhat related to object recognition, with the main difference is that the output requires every pixel belonging to an object to be correctly labeled.

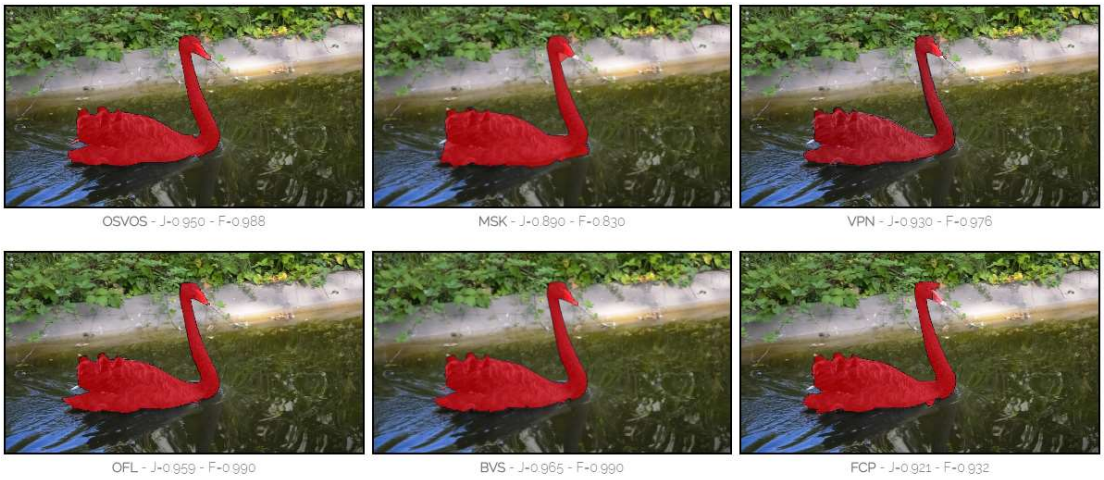


Figure 1.2. Comparison of object segmentation methods participating the 2016 DAVIS Challenge [5]

Semantic segmentation bears great significance to many fields, especially robotics. Semantic segmentation enhances robotic visions by allowing better and faster ways to gain information about a scene, whether there is an object in front of a robot or if its path is passable. Semantic segmentation is also useful in medical imaging, helping to delineate different anatomical structures and to detect unhealthy tissues [6]. Some recognition tasks also make use of semantic segmentation [7].

1.1.2. Visual Saliency

Saliency of an object is defined as the quality of that object to stand out from its neighbors and surroundings [8]. In other words, it can be understood as the quality to draw attention from observers. For biological entities, visual saliency detection helps bring about efficient selection of visual locations for the brains to process, narrowing down the visual field for easier execution of tasks such as visual recognition [9] [10] [11]. Similarly, for computer systems, visual saliency detection let us move from the brute-force approach to processing images, i.e. scanning the every part of an image for the region or object of interests, to rapid candidate selection method [12] [13]. Visual saliency detection is thus a good problem to solve in the field of Computer Vision.

Saliency object detection, the problem of detecting which objects, instead of just regions, in the scene draw focus, also contribute to many different applications. Salient Object Detection can be applied to many other problems, including image/video compression [14], attention retargeting [15], image/video segmentation [16], and image summarizing [17].

Understanding visual stimuli that promote saliency can help us create objects that easily draw attention, as we have done with orange traffic cones or traffic signs. Conversely, objects that are salient emit different visual cues that render themselves noticeable, so if we are to detect those objects, using those cues might let us achieve our goal. Traditional methods of salient object detection also employ various visual cues that indicate the whereabouts of the object [18], with the most widely studied and used cue was contrast, the distinction between regions or pixels of an image to global or local contexts [18]. These methods, however, still rely on image features that are designed by humans, without full understanding of visual attention [18]. Modern methods of detecting saliency objects employ various types of prior knowledge, e.g. background or compactness, while other methods seek to combine different saliency cues.

1.2. Video matting and the use of chroma key compositing

Video matting is an interesting application that is used by both Hollywood and aspiring movie creators. Matting refers to the process of pulling out the foreground of a scene and producing the "matte", a kind of mask that can be used both to separate the object out of the current scene and to combine the object to another scene. Traditionally a very tedious or hard process in film making, matting methods has slowly improved through the years, creating more and more natural feeling to video compositions. As of 2016, one of the most widely used method for creating video composition is chroma key compositing, commonly known as green screen compositing.

The method is extremely simple and straightforward in theory. To extract the subject of interest out and place it in a more desirable scene, the subject must first be placed in front of a monochromatic background, usually green because it is highly different from the human skin color and digital cameras are more sensitive to green light. With the captured scene, everything having the determined background color (green) will be removed and replaced with a transparent background, allowing the scene to be placed on top of any desired background.

Figure 1.3. Singer Elizabeth Bougerol of The Hot Sardines tinted green in their music video "Running Wild" (2016)

The drawback of this method is that it is not easy to set up correctly. Lighting of the subject and the background must be carefully orchestrated to ensure a clean cut. Placement of the subject also matters as green light can be reflected onto the subject, giving it a green tint. Figure 1.3 shows an example where the main subject is affected by the light, giving an unnatural feel. Furthermore, relying on a color to remove the background makes it hard to extract subjects that share the color with the background. If green is used as the background color, weather reporters cannot wear anything green when they are being filmed.

Other methods for segmenting objects out of a scene that do not use green screen often require tedious work. Automatic matting for video is still somewhat under-developed and thus there are hardly anyway to replace chroma key composition.

The example above shows that there is a need for a better method of segmenting objects in a scene automatically. Green screen compositing can only be used with a proper set-up, while methods not relying on chroma keying require manually editing for them to work. A novel segmentation method that allows automatic segmentation would be valuable to video matting.

1.3. Motivation

This section discusses the factors that motivate our thesis. First, the authors mention the need for video object segmentation. Second, the authors state the lack for video object segmentation solutions. Third, the authors want to have a rather different approach to the object segmentation problem. Finally, the authors want to find a way to improve the current video matting process by omitting the need of extra equipment. These are discussed in the two subsections, where Subsection 1.3.1 discusses the first motivation, the Subsection 1.3.2 discusses the second one, the Subsection 1.3.3 discusses the third one and the Subsection 1.3.4 discusses the last one.

1.3.1. The need for video object segmentation

Along with the growth of information technology, multimedia, especially videos, becomes a gigantic source of data. Therefore, retrieving desired information from such amount of data is a complicated problem. It is necessary for efficient algorithms and methods for video segmentation to be researched and developed. A good video segmentation algorithm/method is evaluated based on: granularity, shape precision, temporal coherence, time executed, etc.

There are many applications for video object segmentation that are already put into experiment or practice.

- Automatic Video Matting.

- Object-based Surveillance analysis (offline/online).
- Object-based Scalable Video Encoding.
- Augmented Reality.
- Virtual Surgery.

1.3.2. Lack of video object segmentation methods

With the increasing need for video object segmentation, there is still a shortage for algorithms and methods to resolve it. Most of current segmentation methods, however, target a single image, a single frame instead of sequences of frames (video). Therefore, there are some activities created to encourage the development of more efficient algorithms, methods to do video object segmentation task. One of the typical annual activity is DAVIS challenge, a challenge focusing on Video Object Segmentation, sponsored by Google, Disney Research, Nvidia, etc [19].

1.3.3. A different approach to object segmentation problem

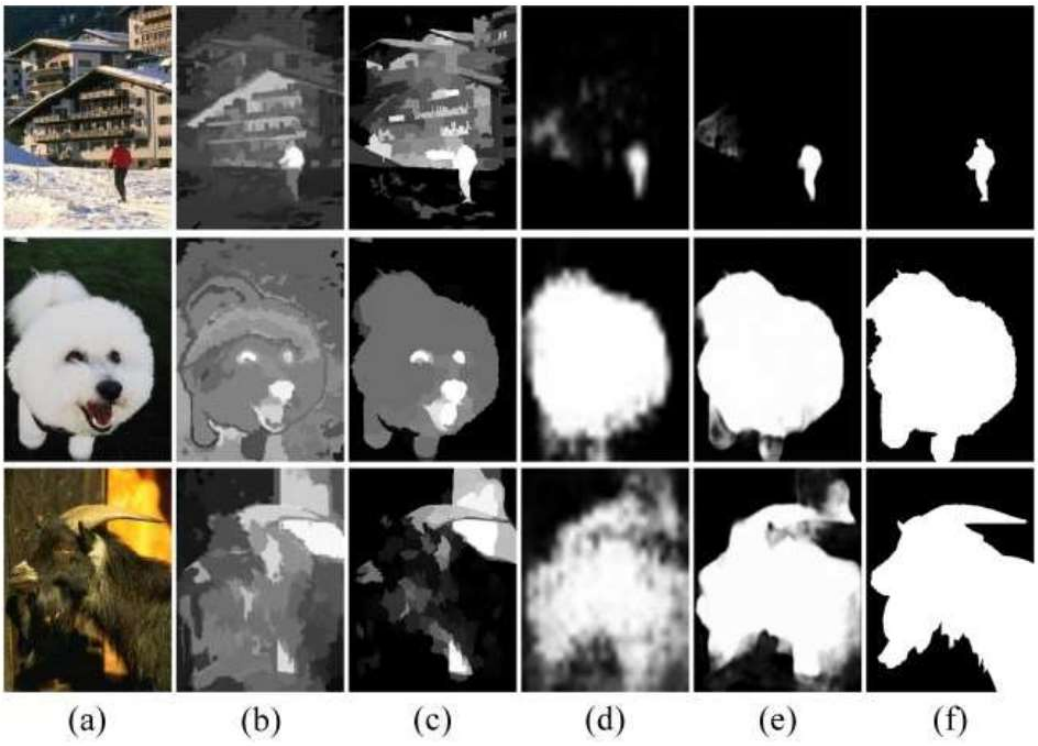


Figure 1.4. Comparison of different object saliency methods [18]

Object segmentation is about finding objects and creating a boundary around each object as tightened as possible. With this ability, its applications include tracking objects of interest given limited information. During the authors' study of the problem of object segmentation, the authors come across object saliency papers [18] [20] and find that these papers can lead to a fresh approach, the one that is not applied widely, to object segmentation task.

The saliency of an object is the state by which that object is easily noticeable in comparison with its neighbors. A good object salience method can not only give the information about how much the object stands out but also extracts fairly accurately the object from its background, as seen from Figure 1.4. Furthermore, by studying the DAVIS dataset on video object segmentation [21], the authors realize that **objects to segment from videos are usually the most prominent object in the video themselves**. Therefore, applying good object saliency method to solve object segmentation problem does not seem like a farfetched approach.

1.3.4. Video matting

Video matting, as presented in section 1.2, still contains some problems. Video matting requires a certain preparation before the combination process can be carried out. The more detailed and complex combination, the more complicated preparation.

Even with the commonly used method of chroma key compositing, there are still cases where the produced results are undesirable. The disadvantage of this technique is that the foreground may get green/blue haze when the color is removed from the video.

Inspired by the idea to make the process of video matting automatic and use little external equipment, the authors propose an application using object segmentation (with minimal user input) for automatically combining the foreground from one video to the background from another.

1.4. Objectives

As discussed in previous sections, visual saliency and semantic segmentation are two widely studied problems in Computer Vision.

The **main objective** of this thesis is to propose a simple approach to take advantage of state-of-the-art methods to apply to **video object segmentation problem using visual saliency with mask propagation and reappearance detection**. The **output** of the approach should be a **ready-to-use matte** that can be applied to compose two scenes serving **automatic matting** purpose. Ultimately, the authors create a **demo application** for demonstrating the ability to **apply object segmentation to video matting**.

The detailed work that we have done in this thesis is:

- Study the basic knowledge about Machine Learning: regressions (linear regression, logistic regression), neural networks (fully connected neural network, convolutional neural network), support vector machines, k-mean clustering and optimization techniques (gradient descent, normal equation).
- Conduct study on various approaches on object segmentation problems. That is when the authors come across object salience papers.
- Propose a novel method for Mask propagation and Reappearance Detection for Video Salient Object Segmentation. The method is based on three key ideas:
    - Take advantage of no-mask salience method and use mask to guide the final segmentation result.
    - Use flow-based tracking algorithm to perform object tracking.
    - Use object detection algorithm to detect disappearance and reappearance.
- Propose using the result of the experiments to solve automatic video matting problem via a demo application.

1.5. Thesis Content

This thesis is structured into 6 chapters:

Chapter 1: Introduction

Chapter 1 presents the definition and importance of object segmentation and visual saliency along with their applications in real life. Chroma key compositing method for video matting is also discussed. The chapter then describes the motivation of this thesis: finding a fresh solution to object segmentation problem. Objectives and detailed work are presented next, listing out the goals of the authors' work and how to achieve them. Finally, Chapter 1 wraps up with the outline of the content of the whole thesis.

Chapter 2: Machine Leaning, Deep Learning, and Neural Networks

Chapter 2 provides information on background knowledge of Machine Learning and Deep Learning in general as well as Neural Networks and Convolutional Neural Networks and how they are applied to problems in Computer Vision. The next subsections present publications related to the thesis. Finally, the authors provide a brief introduction of common frameworks and libraries on deep learning.

Chapter 3: Salient object segmentation for video

Chapter 3 first makes the problem statement about salient object segmentation. The authors propose a novel method for video object segmentation that uses visual saliency as the segmentation method, with mask propagation and reappearance detection to improve performance.

The authors also propose an application that uses our method for object segmentation to do video matting with minimal user input. The application takes as input two video sequences and optional user-defined regions having objects of interest and returns a set of mattes that can be used to compose the separated foreground from a video to the other video.

Chapter 4: Experimental Results

Chapter 4 describes the DAVIS dataset and reports the results of the conducted experiments. The first set of experiments consists of tests on the efficiency of DHSNet in solving salient object detection for images and video sequences and how well the method can be used in a segmentation problem. The results of the first set of experiments are analyzed to devise a suitable method for salient object segmentation. The second set of experiments looks for a suitable tracking method to build up and enhance proposed model.

Chapter 5: Conclusion

Chapter 5 presents the results of this thesis, including what we have learned and achieved through the experiments. The chapter closes with our proposal for future work.

Chapter 2

Machine Leaning, Deep Learning, and Neural Networks

✍ *Chapter 2 provides information on background knowledge of Machine Learning and Deep Learning in general as well as Neural Networks and Convolutional Neural Networks and how they are applied to problems in Computer Vision. The next subsections present publications related to the thesis. Finally, the authors provide a brief introduction of common frameworks and libraries on deep learning.*

2.1. Machine Learning and Deep Learning

Machine Learning stemmed from the wish of having computers possessing the ability to learn from the data by themselves, without requiring any explicit coding from humans [22]. The need for computers to do so is in increasing demand not just in Computer Vision but also in other fields as well, because there are tasks that cannot be (efficiently) done by giving computers static programming instructions. Such tasks include optical character recognition (OCR), weather predictions, or natural language parsing.

Machine learning, specifically the later-emerged Deep Learning, provides a data-driven approach to solving problems in Computer Vision. Traditional methods to solving Computer Vision problems depends mostly on human-crafted or human-designed features extracted from the images without attempting to copy how the best visual system (our eyes and brains) works. While this worked for some cases such as edge detection or object recognition, even resulting in outstanding methods, they do not generalize well. Furthermore, without neural networks, problems such as image classification will take a hit to their performance. Basic image classification making use of k-Nearest Neighbor classifiers hinders practical use by a number of disadvantages. The classifier must store all available training data to use for comparisons with the test data, but datasets can be up to gigabytes in size so they are

costly in terms of space. Furthermore, for a classification of a single image, the image must be compared to all of the available training images, which leads to the undesirable property that testing is computationally costly.

Deep learning opens the path to solutions modeled after the human perceptive system. The rise of Neural Network helps us advance further in the field of Computer Vision. Recent methods making use of neural network have been providing amazing results for various kinds of problems, such as object detection and recognition, visual saliency detection, and semantic segmentation to name a few.

2.2. Neural Network

2.2.1. Overview

Neural networks refers to systems of interconnected components (neurons) that took inspiration from neural networks that make up the biological brains. This kind of system is increasingly popular amongst researcher, becoming the go-to approach to problems of various areas, including medical diagnosis [23] [24], sequence recognition [25], and pattern recognition [26].

2.2.2. Artificial Neuron and Perceptron

The idea behind neural networks, i.e. having computational units that produce "intelligent" results only through interactions with each other, was inspired by the brain. For instance, the Neocognitron system, proposed by Kunihiko Fukushima in 1980, took inspiration from the mammalian visual system and laid the foundation for modern convolutional networks [22]. Thus, the artificial neurons in these networks mimics the structure of biological ones.

As can be seen in Figure 2.1, not unlike the biological neuron, an artificial neuron contains a "cell body", which is the central part of the neuron that connects with "dendrites", which provide inputs, and "axon", which returns output. The cell body is in charge of sending signals out through the axon based on the signals that it received through dendrites. Signals sent through axon will be picked up by another

neuron through a dendrite. Axon of a neuron connects to dendrite of another neuron through a structure known as synapse, and the synaptic strength dictates the significance of the signal.

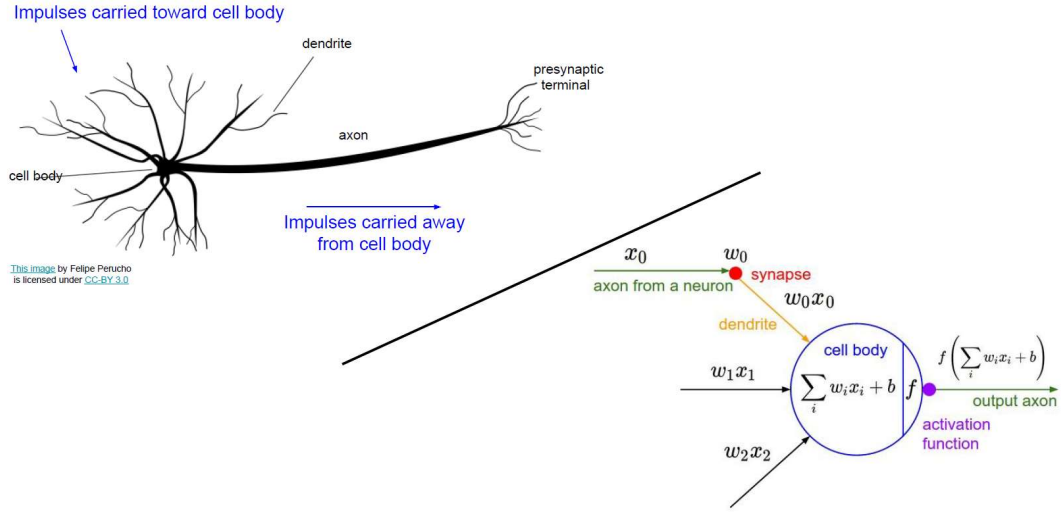


Figure 2.1. Analogy between a biological neuron and an artificial one of a neural network.

The very first artificial model of the biological neuron is in fact perceptron, introduced by Frank Rosenblatt in 1958. Perceptron is an algorithm for supervised learning in machine learning. Perceptron, in essence, is a simple function that turn inputs (usually a real-valued vector) into one binary output.

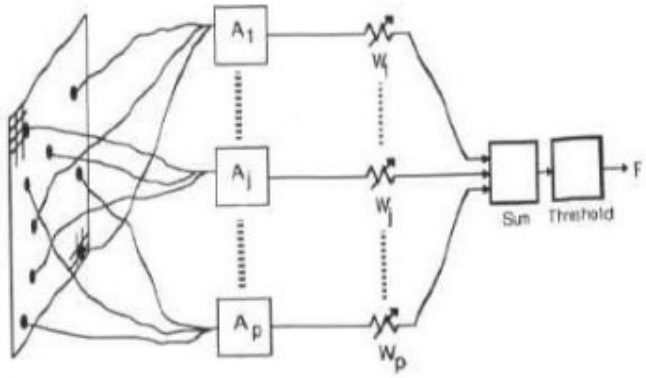


Figure 2.2. Basic structure of a perceptron

Following Figure 2.2, we can see the perceptron receives p inputs, $A_1, A_2, \dots A_p$ (or $x_1, x_2, \dots, x_p$, depending on the source). The inputs are then respectively weighted by $w_1, w_2, \dots, w_p$, which are real numbers indicating the importance of each of the input values. The output F will then be calculated using the sum of those weighted inputs. However, since the output is a binary value, a threshold is used to achieve the desired result. To be more specific, a perceptron is written as follows:

$$F = \begin{cases} 1 & if \sum_{i=1}^{p} A_i w_i \geq Threshold \\ 0 & if \sum_{i=1}^{p} A_i w_i < Threshold \end{cases}$$

The output of a perceptron is thus controlled by two things: the weights $w_1, w_2, \dots, w_p$ and the $Threshold$. In modern neuron networks, however, the equation changed a bit by bringing the $Threshold$ to the other side of the inequalities. The additive inverse of $Threshold$ is then known as $Bias$ and the perceptron will be rewritten as:

$$F = \begin{cases} 1 & if \sum_{i=1}^{p} A_i w_i + Bias \geq 0 \\ 0 & if \sum_{i=1}^{p} A_i w_i + Bias < 0 \end{cases}$$

2.2.3. Multi-layer perceptron

A concept that later arose from perceptron and became the definitive example of deep learning is multi-layer perceptron. Multi-layer perceptron (MLP) is nothing more than layers of perceptrons connected together in a unidirectional manner. In modern neural networks, artificial neurons are arranged to layers and "axons" of neurons in a layer will be connected to "dendrites" of neurons in the next layer. In other words, the output of a layer will be fed as the input for the next layer in a network. This helps enabling outputs to be more complex and abstract. For a layer in a neural network, if each of its neuron is connected to all other layers in the previous and next layer, that

layer is defined as a *fully connected* layer. The last layer of a network is typically a fully connected one. The structure of MLP makes it possible to solve complicated problems such as object recognition and finding optical flows.

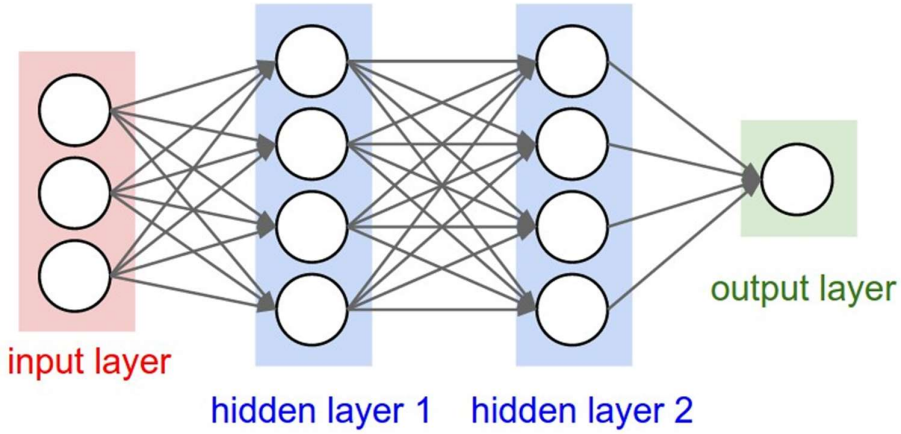


Figure 2.3 Structure of a Multi-layer Perceptron

In general, the *input* and *output layers* (the leftmost and rightmost layers in Figure 2.3) of a network are always required. Their job is straightforward, being the in and out gates for the network. The layers between them, however, are what makes neural networks complicated. They are referred to as *hidden layers*. Unlike input and output layers, hidden layers are hard to directly designed. Thus, there are typically some level of heuristics involved when it comes to creating them. Hidden layers are designed differently for each network, depending on the network's purpose. Therefore, the number of hidden layers also varies from network to network.

2.2.4. Activation function

Perceptron is the precursor to modern artificial neurons. Instead of returning only a binary output, an artificial neuron now produces values that are anywhere in the range [0,1]. The reason is to help in the learning process of a network. Neural networks have the ability to learn, i.e. finding the appropriate weights and biases given sufficient input data. The learning process, however, needs to be gradual, which means weights and biases get increasingly close to the "good" values. Having a binary output for each of the neuron makes it hard for this process to be done. For training a

neural network, we use an error function to see how far or close the network is to the optimal results. Because of the binary output of a perceptron, a small change in the network's parameters can lead to a stark difference for the output, making it hard to tune the parameters to achieve a good result. Therefore, modifications must be made to the original perceptron model. Instead of using 0 as the threshold at which signals are allowed to be fired, we can use an activation function to appropriately map the output to the range we need.

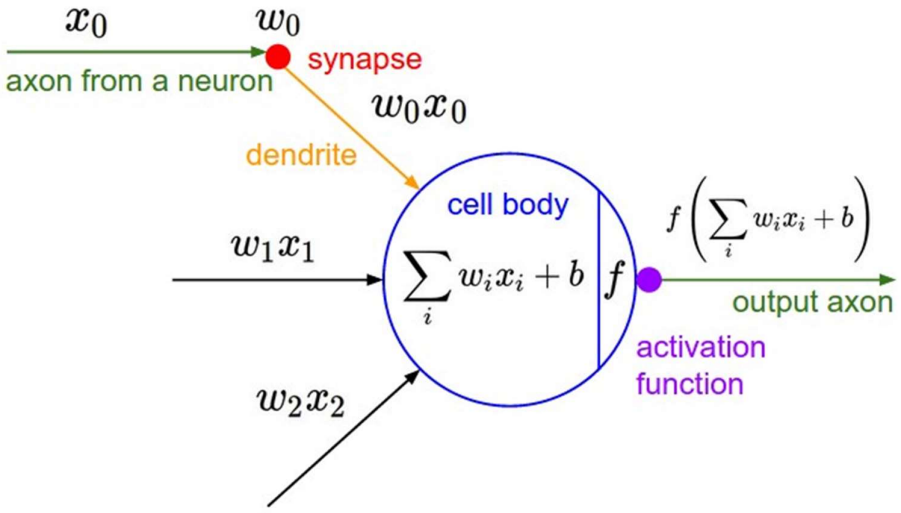


Figure 2.4. Structure of an artificial neuron

The activation function takes as input the weighted sum of the input values fed to the perceptron and returns a value in the range [0,1]. Historically, the Sigmoid function was used for this purpose as it is able to squash the sum to the desired range [27]. Modern networks, however, employ different functions as well, such as hyperpolic tan function (tanh) or rectified linear unit (ReLU), to achieve better performance [27].

For an artificial neuron whose output before going through the activation function is $\sum_i w_i x_i + b$ (Figure 2.4), the Sigmoid function is defined as follows:

$$\sigma\left(\sum_i w_i x_i + b\right) = \frac{1}{1 + e^{-\sum_i w_i x_i + b}}$$

Figure 2.5 is the graph of the Sigmoid function, from which we can see the function maps inputs to a smooth curve in the range [0,1]. As explained above, this makes it possible for networks to learn their parameters.

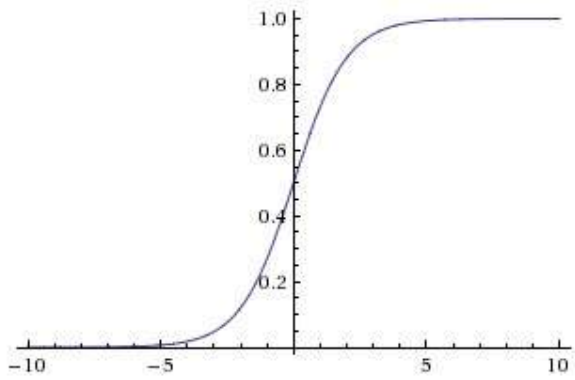


Figure 2.5. Function graph of the Sigmoid function

In practice, however, Sigmoid function has a few drawbacks and thus are not preferred over tanh and ReLU functions [27].

2.2.5. Feedforward and Backpropagation

Two operations are particularly important in neural networks: Feedforward and Backpropagation.

Feedforward is used in both the training and testing stage of a network. The task we need to do in feedforward is straightforward: Passing output of a layer as input of the next layer. Since all we are doing is unidirectionally putting values through the network from the input layer to the output layer, the operation is called feedforward.

Backpropagation is generally used in the training stage to help neural networks learn their parameters and it is considered an optimization. Different from feedforward, the job of this operation is to propagate error of the output values back to the network to update the network parameters. Backpropagation works by first do the normal feedforward operation on the network with the given input. After the output is obtained, we compare the output to the desired output, using a loss function to generate an error term for each of the neuron in the output layer. The error values are

then propagated backwards from the output layer until every neuron receive their respective error term. The error terms will be used to calculate the gradient, with which we can update the weights of the network to minimize the loss function as the process repeats. For the purpose of finding the most fitting parameters, *gradient descend* is usually applied.

2.3. Convolutional Neural Network

2.3.1. Overview

Convolutional Neural Network (CNN) is similar to the normal neural network in that the core structure that facilitates the network is perceptron controlled by weights and bias. What set CNN different is that it is specifically designed to work with grid-like data such as images.

Although regular neural networks are great for solving problems in general, they do not scale well for images. Typically, the input for a neural network is a real-valued vector. For a colored image of size 200×200, the first hidden fully connected layer would have each of its neuron depend on $200 \times 200 \times 3 = 120000$ weights. The number of parameters does not stop there though, as it is likely that we also have other neurons in the same layer and there might be other layers. This renders regular networks ineffective for problems in Computer Vision. Furthermore, too many parameters can lead to overfitting of the input data and thus makes it hard to generalize.

The CNN solves this problem by introducing a different constraint on the architecture. Specifically, the layers of a CNN have their neurons arranged in 3 dimensions: width, height, and depth. The neurons of a layer in CNN also only have to connect to a subset of the neurons of the layer right before them instead of each and every neuron as in the case of a regular neural network. Another thing that make CNN different from regular network is that its output is a three dimensional vector $1 \times 1 \times k$ containing the class scores, with k is the number of classes.

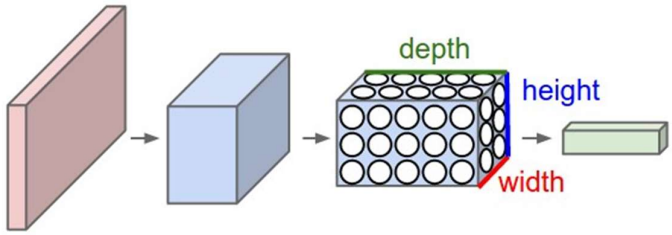


Figure 2.6. Illustration of a Convolutional Neural Network

2.3.2. Architecture

Convolutional neural networks are still sequences of layers of neurons, but the layers in a CNN are different from those of a normal neural network. There are several types of layer that make up a typical CNN.

2.3.2.1. Convolutional Layer

Practically the staple of any CNN, this layer computes the dot product between each of its neurons' weights and the region of the input the neurons connect to. Since having neurons in a layer connect to all neurons in the previous layer is impractical with images and other high-dimensional inputs, neurons in CNN only need to connect to a region of the image. As neurons in CNN are arranged to dimensions width, height, and depth, the extent of connectivity along the width and height of a filter is a hyperparameter known as the receptive field of the neuron. The parameters of a convolutional layer include a set of learnable filters, each of which is small in terms of width and height, but takes in the full depth of the input volume. When we do the forward pass, we move each of the filter across the width and height of the input volume and compute the dot product between the filter's entries and the input to produce a 2-dimensional activation map that returns the responses of that filter for every spatial location convolved. Thus, the network learns filters that will activate when they encounter some types of visual features such as edges or interesting patterns. The activation map of each filter in the convolutional layer will then be stacked along the depth dimension to produce the output volume for the layer.

Convolutional layers allows CNN to take spatial information into account by letting neurons connect only to a region of the input image instead the whole one. This mimics our visual system in that we do not process information from the entire scene at once but only focus on certain areas for information extraction before we move on to the next ones. This also enables CNN to produce better response to local patterns as we do tasks that require finer details.

2.3.2.2. Non-linear Gating

What fully connected layers and convolutional layers do is linear transformation. When we use them to form a network, the network itself becomes a linear transformation, which would not work well on many types of problems. Therefore, a non-linear activation function must be applied in the network to counter this. The sigmoid function will work just as it did with normal neural networks, but there has been a favor of a better function known as ReLU because of its computation simplicity. ReLU is element-wise activation function defined as $max(0, x)$. There exists several variations to the function as well, allowing adaptation to different problems.

In a typical convolutional neural network, a ReLU layer is placed directly after a convolutional layer.

2.3.2.3. Pooling Layer

Pooling in one interesting concept of CNN. The job of a pooling layer is to down-sample the input in a non-linear manner and thus provides some important characteristics to the CNN. Pooling layers reduce the spatial size, i.e. width and height but not the depth, of their input using a non-linear function.

The reasons pooling concept is important to CNN is twofold. First, by reducing the input size, we are able to reduce the number of computations needed. Second, by discarding some parameters, we can avoid overfitting. Although we are working with the spatial information of an image, we care less about the exact location of a certain feature than we care about where that feature is relative to other features.

The most common way to implement pooling is referred to as max pooling, as it uses the max function. To be more specific, max pooling is done by first divide the input into non-overlapping regions. Then at each region, we returns the maximum value found in that region. The process is done separately on each of the depth slice of the input.

2.3.2.4. Dropout Layer

Dropout refers to a technique to reduce overfitting in neural networks. For any regularization method controlling overfitting, the main idea that drives them is to try to involve the parameters directly in the loss function. In doing so, when the loss function is being minimized, the parameters can both increase (to better fit the data) and decrease (because it is a term of the loss function). The opposition results in an optimization of the network's parameters that let the model fit the data and without losing generality.

Dropout layers in convolutional neural networks are used for the same purpose. Basically, at each training stage, every neuron has a probability p of being dropped from the network. This helps the network in two ways. First, the method restrict the amount of information the network receives to train and thus allows for learning of better generalizing and more robust features. Second, by cutting off some neurons and all of their connections, we can improve the training speed for the network. Generally, the probability of dropping a neuron is lower when the neuron belongs to the input layer, since we are directly discarding information from the input.

Dropout helps us create models easier for there is no need of manually cater to the network, the network automatically adapts to the data without being overfitting.

2.3.2.5. Loss Layer

The final layer in the training stage of a network is generally the loss layer, which is used to compute the error of the predicted output. Several loss functions can be applied to the loss layer depending on the network, with Euclidean and Softmax log functions are two of the most popular ones.

Euclidean loss function calculates the loss value based on the Euclidean distance between two vectors. For any two real-valued vectors of n dimensions $\vec{u} = (u_1, u_2, \dots, u_n)$ and $\vec{v} = (v_1, v_2, \dots, v_n)$, the Euclidean distance between $\vec{u}$ and $\vec{v}$ is:

$$d = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \cdots + (u_n - v_n)^2}$$

Since maximizing the square of Euclidean distance is also maximizing Euclidean distance itself, in practice, the squared Euclidean distance is used instead to avoid the calculation of the square root. Euclidean loss function is used for regression to real values in $(-\infty, \infty)$.

For K-label classifications, Softmax loss functions is used instead. It is the generalization the Sigmoid function that is used to squash real-valued outputs to values in the range $[0,1]$ that whose sum adds up to 1.

2.4. Caffe – A Deep Learning Framework

2.4.1. Overview

Caffe is an open-sourced deep learning framework developed and maintained by Berkeley AI Research (BAIR). The framework values expression, speed, and modularity [28] and is popular amongst people in the deep learning community, garnering over 18000 stars on their Github repository. Implemented in C++ with interface for both Python and Matlab, Caffe allows for easy implementation while maintaining a good running speed. Caffe is known to possess the following qualities:

- **Expressive architecture** that allows for flexibility, encourages application and innovation. All models and optimization are defined in separate configuration files, enabling users to avoid hard-coding. Switching between CPU and GPU is also made easy by allowing a flag to be set.
- **Extensible code** enables active development. With over 1000 users forking and contributing to the original repository, Caffe is able to keep up with state-of-the-art models and code.

- **Speed** that is suitable for both research and development, able to reach 1ms/image for inference and 4ms/image for learning with a single NVIDIA K40 GPU [28]. It is believed that Caffe ranks among the fastest implementations available of convolutional neural network.
- **Community**: the Caffe community consists of a wide range of users, from academic researchers to industrial software engineers, covering many types of projects, including academic research projects, startup prototypes, and large-scale industrial applications in vision, speech, and multimedia.

Caffe powers many state-of-the-art results across many research interests, including salient object detection (DHSNet), semantic segmentation (PSPNet).

2.5. DHSNet

DHSNet, short for Deep Hierarchical Saliency Network, is a convolutional neural network proposed by Nian Liu et al. in CVPR 2016. The network achieved state-of-the-art results as of 2016 with incredible detection speed and good results.

2.5.1. Motivation

In the paper, Liu et al. list factors that held back other salient object detection methods presents and explain how the factors affect the results and performance of those methods.

Traditional salient object detection methods does not satisfactorily do the job. Global and local contrast methods fails on images having large salient objects with complex textures, and object details are often missed. Furthermore, some of them make use of human-designed mechanism that may make it hard to generalize. Other methods using prior knowledge, such as hypothesizing that regions near the edges of an image are usually from the background or that salient objects are often homogeneous, have the same drawback that is they are modelled by formulations that may arrive from empirical evidence. Moreover, some methods use over-segmentations of images (partitioning images to many small regions, by extracting super-pixels for example), increasing the time taken to detect salient objects.

There are also problems with modern salient object detection methods that uses CNN. The first problem is that prior salient object detection methods failed to address global context sufficiently. Most of the work on salient object detection that make use of CNN focused more on the local context, with global view only used to improve saliency score simultaneously or at a later stage. The second problem with those methods is that they process local regions separately and do not account for the correlation of regions in different locations. Furthermore, they depends on image over-segmentation and thus takes up lots of time. All of those weaknesses were overcome with DHSNet.

2.5.2. Architecture

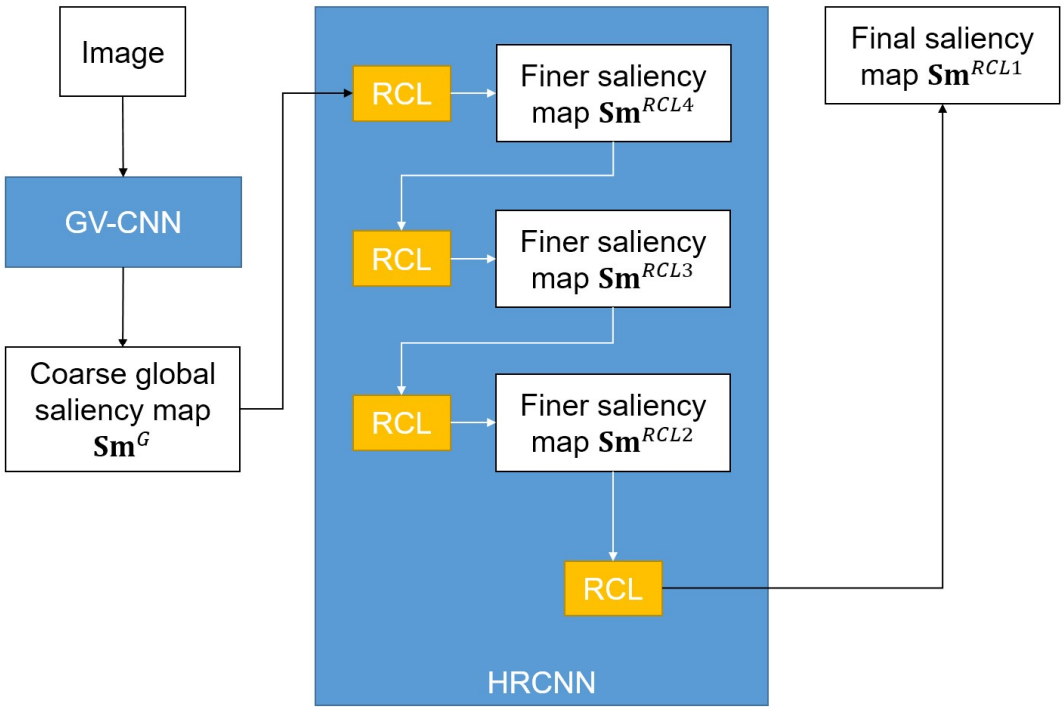


Figure 2.7. High-level view of the architecture of DHSNet

One of the main idea of DHSNet is that the whole image is used as a computational unit and global context information is then passed hierarchically and progressively to

local contexts. Hence, global information is well used and distraction from local contexts are minimized from the beginning.

Figure 2.7 illustrates the high-level view of DHSNet. DHSNet is made up of two parts: the global-view convolutional neural network (GV-CNN) for a coarse saliency map and hierarchical recurrent convolutional neural network (HRCNN) for refinement of the saliency map.

GV-CNN:

The GV-CNN itself consists of a total of 15 layers: the 13 convolutional layers from the VGG 16-layer network, 1 fully connected layer, and a reshape layer. The convolutional layers from VGG network are used to extract deep features, which will then go through the fully-connected layer that learns to detect salient objects and finally a reshape layer that produces a coarse global saliency map $\mathbf{Sm}^G$ of size 28×28. The result only roughly indicates the salient objects but not their boundaries and details, so it needs further refinements in the next stage.

HRCNN:

After the coarse global saliency map is acquired, the HRCNN is employed to get more details from the image. The coarse saliency map will first be concatenated with a convolutional layer of VGG network. The convolutional layer is squashed to the range $[0,1]$ and decreased in the number of features before it is combined with the saliency map to help mitigate computational cost. After the combination is done, the coarse saliency map and the local features in the VGG layer are used to generate a finer saliency map with the help of a recurrent convolutional layer. The result is a refined saliency map that is then up-sampled to a larger size. That saliency map will be put through the process again: combine with a VGG convolutional layer, used to generate a finer saliency map, and up-sampled. After going through the process for a total of 4 times, we obtain a final saliency map from the starting coarse global saliency map.

2.5.3. Results

DHSNet was compared to 11 other state-of-the-art salient object detection models and it was shown to out-perform all other methods visually and quantitatively, especially on complex datasets.

DHSNet can detect and localize salient objects accurately and efficiently while still preserves subtle object details. Furthermore, it is able to handle various situations, such as complex foregrounds, cluttered backgrounds, and objects cropped by image boundaries.

2.6. Optical Flow

2.6.1. Motivation

Human vision has a wonderful ability to detect motion from consecutive frames in a video naturally and effortlessly. When a movement from the scene is detected by the brain, it will signal the eyes to move to the opposite side of the detected movements to equilibrate the scene.

This kind of ability is helpful for detecting a moving object or recognizing danger. Therefore, it is advantageous if this kind of ability can be applied to computer vision somehow.

2.6.2. Definition

Optical flow (or optic flow) is defined as "is defined as the change of structured light in the image, e.g. on the retina or the camera's sensor, due to a relative motion between the eyeball or camera and the scene" [29]

The outcome of an optical flow algorithm is a flow map. This flow map can tell the displacement along x-axis, y-axis of each pixel in a 2d frame and also along the z-axis in a 3d picture.

The optical flow is demonstrated in Figure 2.8.

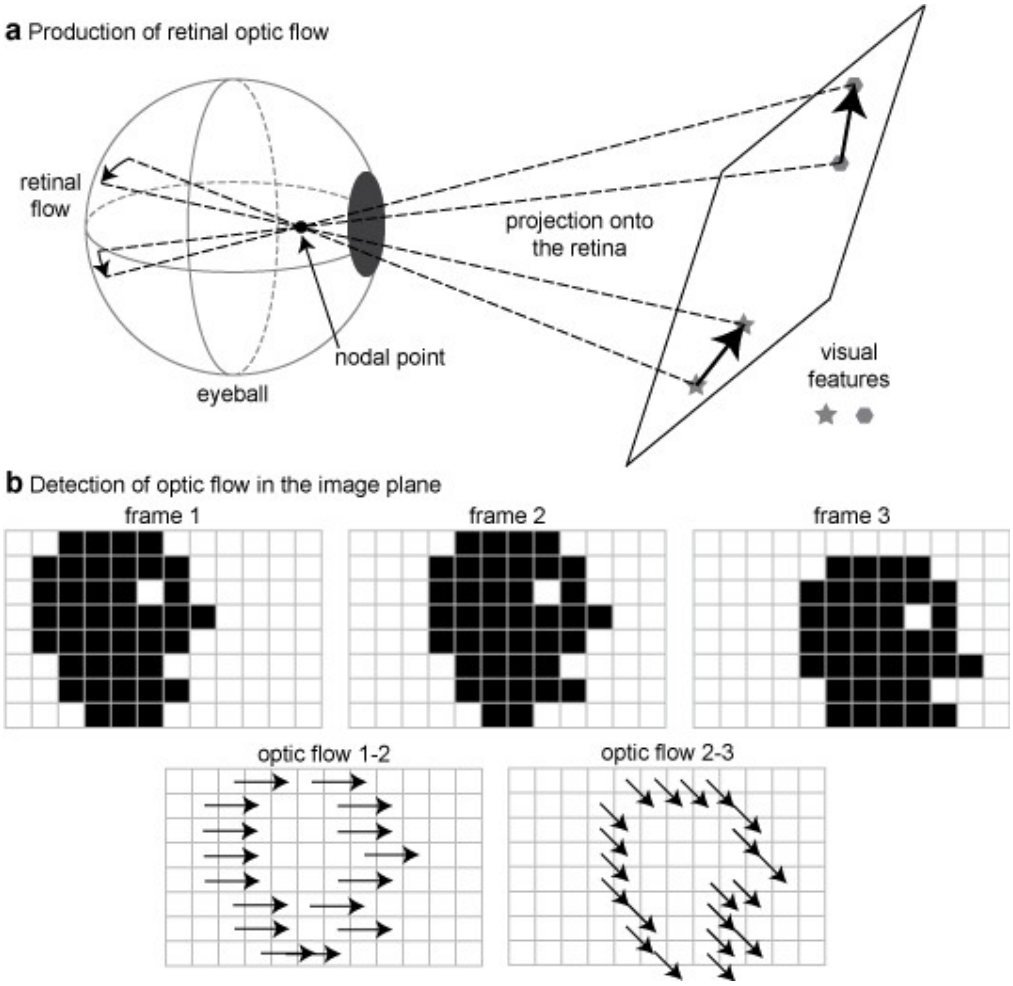


Figure 2.8. The production of retinal optical flow and the detection of optical flow in the image plane [29]

2.6.3. Application

Optical flow is used to let the machine learn about the perception of objects, perception of the world around, perception of movement.

In computer vision, optical flow is used for the task of movement detection, activity recognition, image and object segmentation.

2.7. Deepflow (Large displacement optical flow with deep matching)

2.7.1. Motivation

Optical flow is a useful tool for many tasks in computer vision. Even with decades of advances and developments, however, handling large displacements is a real struggle and gets little attention before the work of Weinzaepfel et al.

2.7.2. Architecture

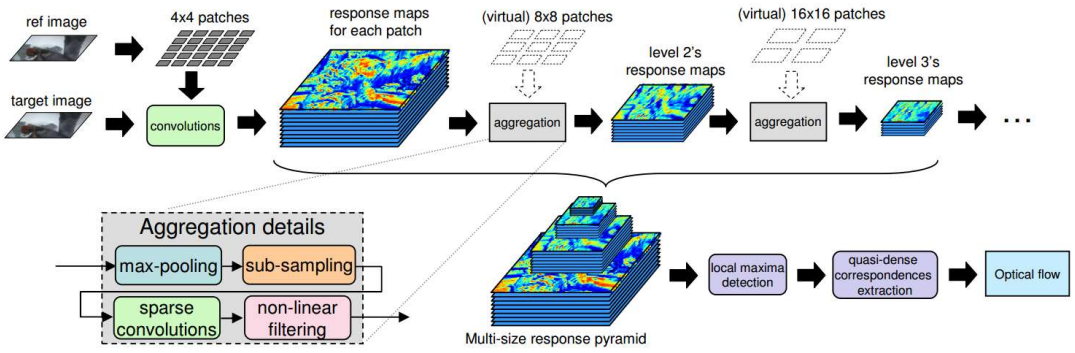


Figure 2.9. Outline of Deepflow [30]

The approach of Deepflow's authors is similar to one from Brox and Malik [31]. There are, however, some modifications in the architecture. First, it is the matching algorithm. Weinzaepfel et al. use Deep Matching algorithm, the one that is also developed by them, into the energy minimization framework. Second, Deepflow's authors also add a normalization in the data term to reduce the influence of the locations with high spatial derivatives. Moreover, a different weight is used at each level to help reduce the impact of matching term at finer scales. An overview of the system can be observed through Figure 2.9.

2.7.3. Datasets

The network is trained and experimented through three different datasets:

- The Middlebury dataset: this is a complex dataset with small displacements.

- The MPI-Sintel dataset: this dataset contains long sequences with large motions, specular reflections, motion blur, defocus blur and atmospheric effects.
- The KITTI dataset: real-world sequences taken from driving platform.

2.7.4. Results

2.7.4.1. On MPI-Sintel dataset

Table 2.1 compares the result of state-of-the-art algorithm against Deepflow on a version of MPI-Sintel test set.

Method	EPE all	s0-10	s10-40	s40+	Time
DeepFlow	**7.212**	1.284	**4.107**	**44.118**	**19**
S2D-Matching [16]	7.872	1.172	4.695	48.782	~2000
MDP-Flow2 [33]	8.445	1.420	5.449	50.507	709
LDOF [6]	9.116	1.485	4.839	57.296	30
Classic+NL [25]	9.153	**1.113**	4.496	60.291	301

Table 2.1. Results on a version of MPI-Sintel test set [30]

2.7.4.2. On Middlebury dataset

Table 2.2 compares the results on the test set between Deepflow and some state-of-the-art algorithms.

Method	AEE	AAE
DeepFlow	0.42	4.22
MDP-Flow2 [33]	**0.25**	**2.45**
LDOF [6]	0.56	4.55
Classic+NL [25]	0.32	2.90

Table 2.2. Results on Middlebury. Average endpoint error (AEE) and average angular error (AAE) of some algorithms [30]

2.7.4.3. On KITTI dataset

Table 2.3 summarizes the results on KITTI benchmark

Method	AEE	AEE-Noc	Out3	Out-Noc3
DeepFlow	5.8	**1.5**	17.93%	7.49%
Data-Flow [29]	**5.5**	1.9	**14.85%**	**7.47%**
LDOF [6]	12.4	5.5	31.31%	21.86%
Classic+NL [25]	7.2	2.8	20.66%	10.60%

Table 2.3. Results on KITTI dataset [30]

2.8. Epicflow (Edge-Preserving Interpolation of Correspondences for Optical Flow)

2.8.1. Motivation

State-of-the-art algorithms in finding optical flow still deals with many challenges: occlusions, motion discontinuities, and large displacements.

2.8.2. Architecture

The Epicflow method is illustrated by Figure 2.10. The input for the model is the two images needed to find the optical flow. The first image is also used to calculate edges using SED algorithm. The model then combines the two cues (the contour map and the matches) to densely interpolate into one dense correspondence field. This intermediate output is used as initialization for the energy minimization framework.

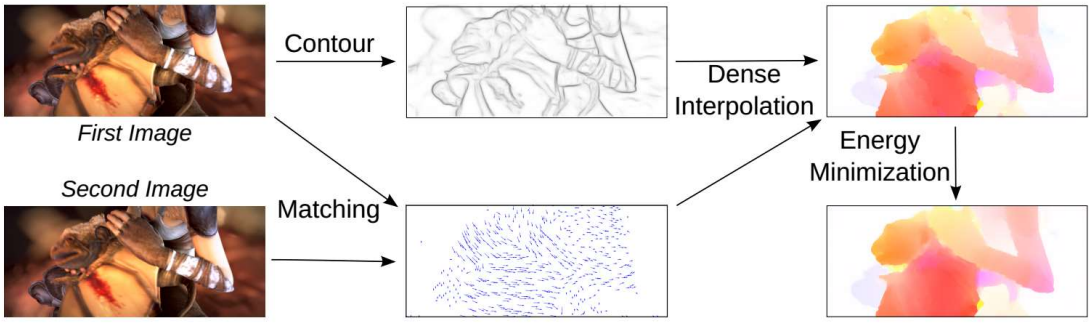


Figure 2.10. Overview of Epicflow [32]

2.8.3. Datasets

The datasets used for training and testing for EpicFlow are the same as those used for DeepFlow: MPI-Sintel dataset, KITTI dataset, and Middlebury dataset.

2.8.4. Results

2.8.4.1. On MPI-Sintel dataset

Method	AEE	AEE-occ	s0-10	s10-40	s40+	Time
EpicFlow	**6.285**	**32.564**	**1.135**	**3.727**	**38.021**	**16.4s**
TF+OFM [20]	6.727	33.929	1.512	3.765	39.761	~400s
DeepFlow [34]	7.212	38.781	1.284	4.107	44.118	19s
S2D-Matching [22]	7.872	40.093	1.172	4.695	48.782	~2000s
Classic+NLP [28]	8.291	40.925	1.208	5.090	51.162	~800s
MDP-Flow2 [38]	8.445	43.430	1.420	5.449	50.507	709s
NLTGV-SC [25]	8.746	42.242	1.587	4.780	53.860	
LDOF [9]	9.116	42.344	1.485	4.839	57.296	30s

Table 2.4. Results on MPI-Sintel test set.

As shown in Table 2.4, EpicFlow outperforms the state-of-the-art methods on the MIP-Sintel dataset. The processing time for EpicFlow is also better than the second best method, TF+OFM, by a large margin.

2.8.4.2. On KITTI dataset

Method	AEE-noc	AEE	Out-Noc 3	Out-All 3	Time
EpicFlow	**1.5**	3.8	7.88%	17.08%	16s
NLTGV-SC [25]	1.6	3.8	**5.93%**	11.96%	16s (GPU)
BTF-ILLUM [14]	**1.5**	**2.8**	6.52%	**11.03%**	80s
TGV2ADCSIFT [7]	**1.5**	4.5	6.20%	15.15%	12s (GPU)
Data-Flow [30]	1.9	5.5	7.11%	14.57%	180s
DeepFlow [34]	**1.5**	5.8	7.22%	17.79%	17s
TF+OFM [20]	2.0	5.0	10.22%	18.46%	350s

Table 2.5. Results on Kitti test set.

The results on the KITTI dataset is shown in Table 2.5, EpicFlow gives good performance in terms of AEE on non-occluded areas. Overall, the method produces results comparable to other state-of-the-art methods.

2.9. YOLO

2.9.1. Motivation

After hundreds of thousands of years of evolution, human's visual perception proves to be accurate and fast. From a quick look at a picture, human can precisely tell what objects are in the picture and their information such as color, shape, usage. In real-time scenarios, short processing time and effectiveness of human vision allow a person to handle the situation fine without putting a lot of thought in what he/she is doing like driving, swimming, dodging a vehicle suddenly coming. The same principle can be applied to computers. If a computer is equipped with fast and high-precision algorithms for object detection, it can convey more accurate information to users or make a better and real-time decision itself without many dedicated sensors. The applications for this include self-driving cars, face detection, counting problems.

Prior to YOLO, detection systems use classifiers as detectors. The process of detecting an object starts with the use of that object's classifier. The classifier is then used on a lot of locations and scales of the picture. Some of those systems use sliding window classifiers to detect objects. With each window, the classifier performs testing on each and every subwindow and the window will slide through the image with defined equal pace.

More recent approaches like R-CNN start the detection process with proposing a lot of bounding boxes, or regional proposals, before applying classifiers on those boxes. After finding the objects in the boxes, they run the boxes through some models to tighten the boxes' coordinates (or refine the results). The whole process is not only slow but also hard to train or improve. The poor speed is caused by the act of passing every potential box through the neural net. Training or improving also meet difficulties due to the complex pipeline: the CNN to extract features, the classifier and the regression model, all of them need to be trained separately.

2.9.2. Architecture

Idea: to convert object detection into a single regression problem, bringing together all the separate segments in the process of detecting object into one neural network [33].

At first, the input image is divided into $S \times S$ grid. Each cell in the grid is responsible for predicting B bounding boxes and their corresponding confidence scores. The confidence score of one cell is calculated by the formula:

$$Confidence\ score = \begin{cases} 0\ if\ no\ object\ exists\ in\ the\ cell \\ IOU_{pred}^{trut}\ , otherwise \end{cases}$$

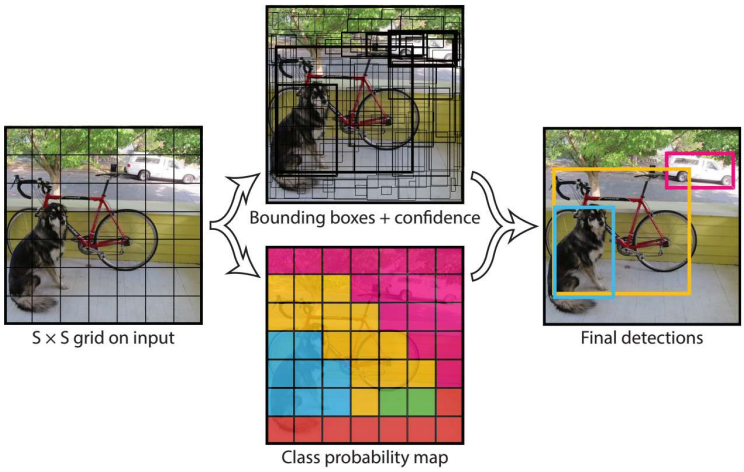


Figure 2.11. YOLO's model [33]

Each cell in the grid also predicts C conditional probabilities $\Pr(Class_i|Object)$, which are calculated on the grid cell containing the object. This set of conditional probabilities are only measured once for each gird cell, no matter how many the bounding boxes.

During the test time, the YOLO authors use the formula:

$$\Pr(Class_i|Object) * \Pr(Object) * IOU_{pred}^{truth} = \Pr(Class_i) * IOU_{pred}^{truth}$$

to find “class-specific confidence scores” for each box. These scores give the information about the probability that the class exists in the box and how well the predicted bounding box matches the object.

The full model is visually illustrated in Figure 2.11.

Network design:

The core of the YOLO’s model is a Convolutional Neural Network.

The first convolutional layers are responsible for features extraction while the last two fully connected layers do the job of predicting the probabilities and outputting the coordinates of predicted bounding boxes.

Figure 2.12 gives an overview about the full architecture of YOLO: 24 convolutional layers, which bear resemblance to GoogLeNet apart from the part that inception modules are replaced by 1×1 layers followed by 3×3 layers, succeeded by 2 fully connected layers.

There is also another version of YOLO that is used for fast object detection. Its architecture is cut off a number of convolutional layers, it only has 9 while the original YOLO has 24. Other specifications are kept the same compared with YOLO.

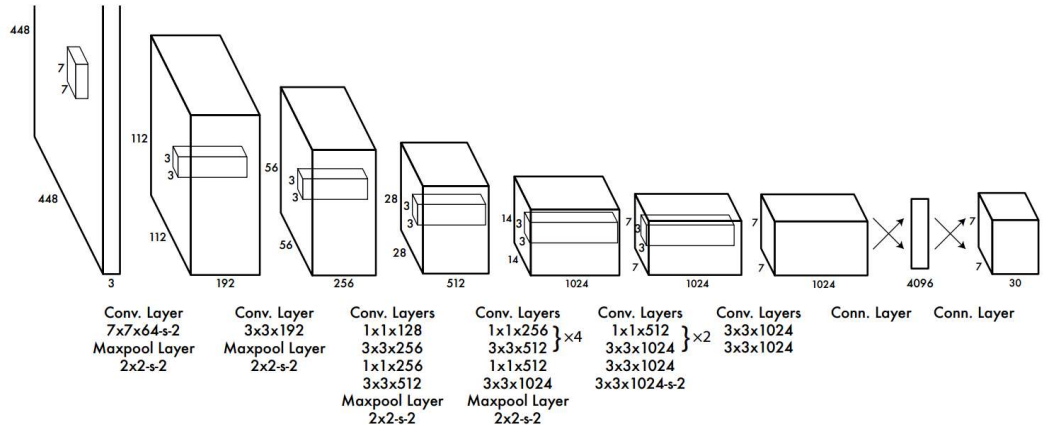


Figure 2.12. The architecture of YOLO [33]

2.9.3. Training

Pretraining: The convolutional layers are pretrained on the ImageNet 1000-class competition dataset [33]. The pretrain process is conducted on the first 20 convolutional layers shown on Figure 2.12 and followed by an average pooling layer and a fully connected layer.

After pretraining, the YOLO's authors attach 4 additional convolutional layers to existing 20 layers and add 2 fully connected layers. Those newly added layers are initially set random weights.

The network is trained for about 135 epochs on the training and validation datasets from PASCAL VOC 2007 and 2012. On testing on PASCAL VOC 2012, the YOLO's creators also use the test set in PASCAL VOC 2007 for training process.

Training process is conducted with batch size 64, a momentum of 0.9 and a decay of 0.0005.

Learning rate is slowly raised from 10^{-3} to 10^{-2} for the first epochs. The next 75 epochs are configured with learning rate 10^{-2}, then it falls to 10^{-3} for the next 30 epochs and 10^{-4} for the last 30 epochs.

During training process, to avoid overfitting, the YOLO's authors use dropout and extensive data augmentation. About the data augmentation method, the YOLO's authors scale and translate randomly up to 20% of the original image size. The exposure and saturation of the image are also adjusted by up to a factor of 1.5 in HSV color space.

2.9.4. Strength and Limitations

2.9.4.1. Strength

- YOLO, as the YOLO's creators state, is "extremely fast". The reason behind its speed is that the creators frame detection as a regression problem so there are no longer complex pipelines with hard-to-train/optimize components. On Nvidia Titan X, the network can process 45 per second without batch

processing. The fast version of YOLO even reaches more than 150 frames per second [33]. Noticeably, YOLO's mean average precision exceeds twice the number of other real-time systems.

- YOLO makes sense of an image globally before giving its "opinions". Therefore, unlike sliding window classifiers or regional proposed-based methods, YOLO looks at the whole image during training and test time so it implicitly encodes the information of context about classes and also their appearance. As a result, YOLO makes two times less mistakes than Fast R-CNN when it comes to background.
- YOLO is capable of generalizing the representations of the objects. When trained on natural images and tested on artwork, YOLO performs a lot better than top detection methods like DPM and R-CNN. With this ability, YOLO is not likely to fail when it comes to new domains or unexpected inputs.

2.9.4.2. Limitations

- Compared with state-of-the-art detection systems, YOLO falls behind in accuracy. It has hard times accurately localizing some object, especially things that are small.
- With strong spatial constraints on bounding box predictions, the number of nearby objects that a grid cell can predict is limited.
- Learning to predict bounding boxes from data gets YOLO into a problem. It has hard times generalizing the objects that are in new or unusual aspect ratios or configurations.
- While the model is trained on a loss function that estimates detection performance, this loss function treats errors in small bounding boxes no more different than those in large bounding boxes. Therefore, this leads to the situation that a small error in a large bounding box can be overlooked but a small error appearing in a small bounding box may affect a lot on the IoU.

2.10. Conclusion

In this chapter, the authors have presented knowledge of Machine Learning, Deep Learning, and Neural Networks, particularly Convolutional Neural Networks, from their history to their importance. Information on state-of-the-art methods for visual saliency and object detection essential to this thesis and the frameworks that power some of these methods are also presented.

Chapter 3

Salient object segmentation for videos

✍ *Chapter 3 first makes the problem statement about salient object segmentation. The authors propose a novel method for video object segmentation that uses visual saliency as the segmentation method, with mask propagation and reappearance detection to improve performance. The authors also propose an application that uses our method for object segmentation to do video matting with minimal user input. The application takes as input two video sequences and optional user-defined regions having objects of interest and returns a set of mattes that can be used to compose the separated foreground from a video to the other video.*

3.1. Problem Statement

3.1.1. Image object segmentation

3.1.1.1. Overview

Image object segmentation is a process of image analysis. Its sole purpose is to extract the region that contains the object of interest in scene. In the ideal case, the region found is the object itself (Figure 3.1).

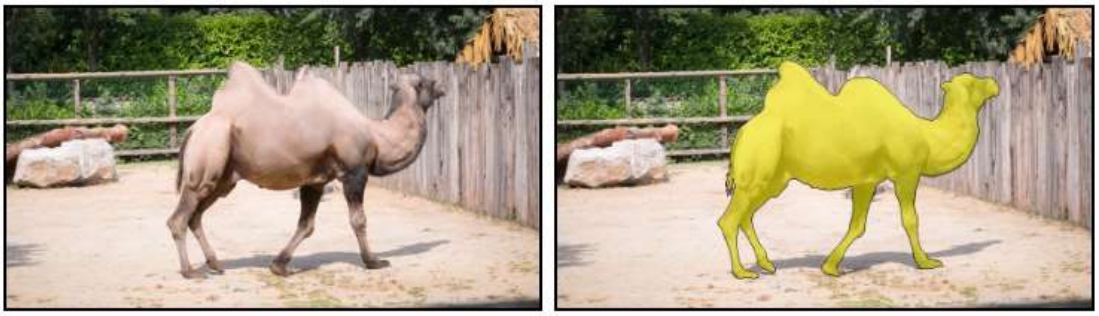


Figure 3.1. An ideal case for image object segmentation result, DAVIS dataset 2016

Image object segmentation is the key to extract information about the structure of the objects, and to split and distinguish a range of objects inside the data.

3.1.1.2. Common methods

There are some common techniques used to solve this problem such as thresholding, edge detection, and pattern matching.

- **Thresholding**: This is among the simplest and most common techniques for image object segmentation problem. Based on the similarity in intensity, thresholding method divides pixels into background and object classes. The simplest form of threshold is the act of replacing any pixel with intensity $I_{i,j}$ with a black pixel if $I(i,j) < T$ and white otherwise. For color images, there is an approach called multi-spectral threshold. It sets threshold values in each spectral band and then combines the results into one single image.
- **Edge detection**: To segment an object out of its background, closed boundaries/edges are needed. The most popular of these techniques is boundary detection by detecting object boundaries as the point of maximal gradient.
- **Pattern matching**: These algorithms work based on an assumption, which is that a high degree of correlativity between the pattern/template and the object. These algorithms may work well under controlled environments such as factories, where there are automatic robots.

3.1.2. Video object segmentation

3.1.2.1. Overview

In a video, there is a type of information that cannot be found in a single image and that is temporal data. Realizing movement, taking advantage of this source of information is an important task when it comes to video object segmentation.

Video object segmentation is the preprocessing task for a lot of high-level computer vision applications, such as video information retrieval, activity summary or understanding.

3.1.2.2. Some methods

- **OSVOS**: S. Caelles, K.-K. Maninis et al. propose a method called One-shot Video Object Segmentation (OSVOS). OSVOS's architecture is based on fully

convolutional neural network that is able to successively transfer generic semantic data. The network is trained on ImageNet for the task of foreground segmentation and then it learns the appearance of the object of interest [34].

- **OFL**: Yi-Hsuan Tsai, Ming-Hsuan Yang and Michael J. Black propose an algorithm that considers video segmentation and optical flow estimation simultaneously. With the optical flow estimation, OFL's authors compute the flow independently in the segmented regions and recompose the results [35].

3.1.3. About our method

Applying object salience methods to perform object segmentation task is a rather fresh approach to this problem. The object saliency methods, led by state-of-the-art DHSNet, give generally good speed, up to 23 FPS on modern GPUs [18] and also potentially fair accuracy for the job of segmentation (Figure 3.2). In this method, the authors use DHSNet as object saliency module because DHSNet not only obtains state-of-the-art results, but also gains real-time speed [18].

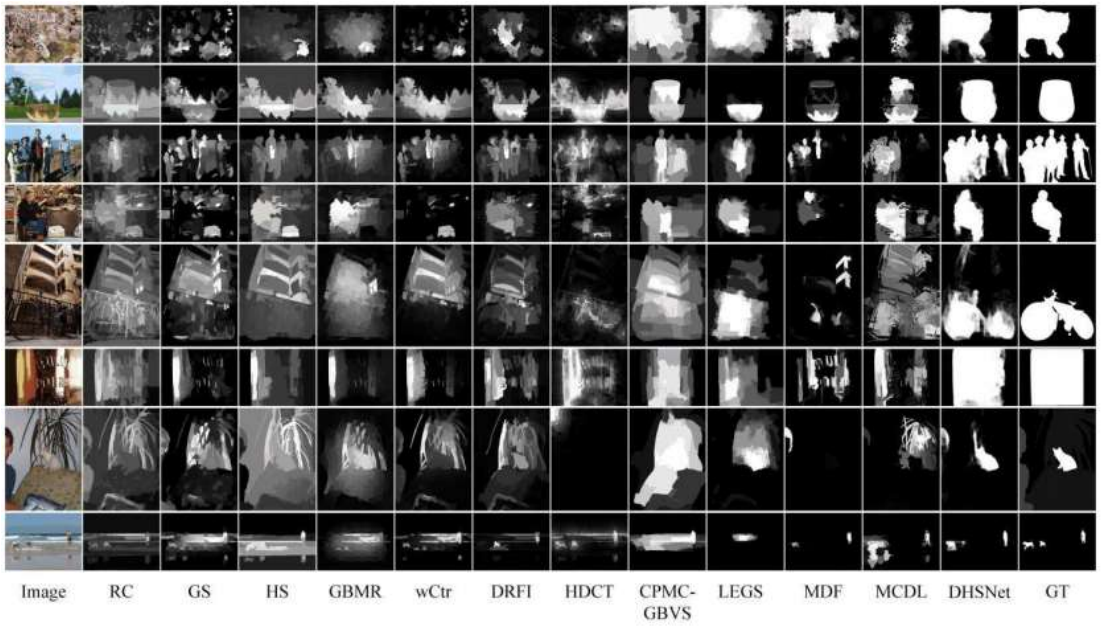


Figure 3.2. Qualitative results of some saliency models [18]

However, there remain two problems to be solved. Firstly, the results of DHSNet is flawed in many ways: there are holes in the middle of the detected object, the object is disjointed, or the boundary is unstable between frames (Figure 3.3). Secondly, those results do not take advantage of temporal data. Therefore, processing a video is the same as processing each frame independently.

Figure 3.3. Some errors of DHSNet results

To take advantage of temporal data when solving object segmentation on video, the authors use **DeepMatching** and **EpicFlow**. With these powerful tools, the authors can predict the movement of each pixel from one frame to another; therefore hopefully raise a better result in comparison with independent frame processing.

3.2. Mask propagation and reappearance detection for video salient object segmentation

We propose a novel method for video object segmentation that uses salient object detection as the main segmentation module. Since our method targets videos, we implement two changes to the normal salient object detection methods:

- Salient object detection are often done on images only, which means if we apply it to videos, the method must run on individual frames separately. Thus, we are not utilizing the temporal information of the sequence, which can be helpful to improve segmentation result. In our proposed method, we employ optical flow to propagate mask information through the sequence, effectively make use of sequence information to help achieving better performance.

- Salient object detection also does not make use of information from the mask. For video object segmentation, mask information can help pinpointing the object in the video and refine the segmentation result if it is properly propagated through the frames. In our method, we try to incorporate mask information to the frame to help guide the salient object detection method.

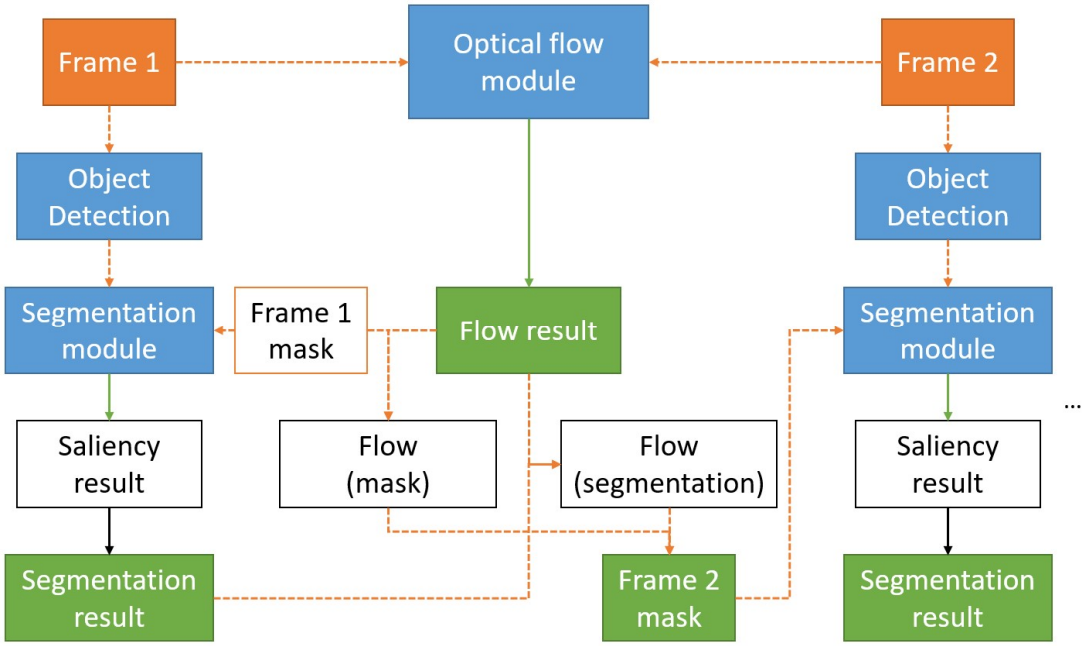


Figure 3.4. Model for videos salient object segmentation with mask propagation and reappearance detection

Three main components make up this model:

- **Segmentation module:** in charge of segmenting object out of the scene.
- **Optical flow module**: in charge of propagating information from the mask through the sequence, helps create new masks and bounding boxes.
- **Object detection module**: detect objects in the scene, in charge of tracking object disappearance and reappearance.

For each of the components, any method that can carry out the described tasks can be chosen for that component. Therefore, the model can be easily updated to make use of better methods in order to increase the performance. In this thesis, the following three methods are chosen for the components:

- **YOLO**: Object detection module. YOLO is one of the state-of-the-art methods for object detection with respectable processing speed.
- **EpicFlow**: Optical flow module. EpicFlow is a strong method for generating optical flow. The advantage of using EpicFlow for the optical flow module is that it can be used to compute optical flow for cases with large displacements or significant occlusion.
- **DHSNet**: Segmentation module. Since we also focus on studying the effectiveness of visual saliency in video object segmentation, we need a good salient object detection method. DHSNet is a great candidate for the component because of its incredible performance. Furthermore, visually, the result of DHSNet closely highlights the prominent object in the scene and can resemble a segmentation result.

For video object segmentation using our method, the object detection module is first applied to the first frame. The result is a labeling of all objects in the frame. The object specified by the mask is treated as the object of interest and the class labels of the object is noted. Then, the bounding box for the object of interest is extracted. The box is used to crop out a region surrounding the object of interest, which is fed to the segmentation module to get the segmentation result for the frame. The next frame is then used along with the first frame as inputs to the optical flow module. After the optical flow is calculated, a new mask will be created based on the mask of the current frame and the segmentation result. The new mask will then participate in the segmentation process for the next frame. If tracking on the object using optical flow is lost, the frame is skipped and the object detection method will be used to detect for reappearance.

3.3. Salient Object Segmentation for Video Matting

3.3.1. Motivation

Video matting is a commonly used technique in filmmaking. As crude and early as George Méliès' 1898 short film "Un homme de têtes", matting has been used to create outstanding and awe-inspiring effects that leave viewers yearning for more. Video matting increasingly became influencing on Hollywood movies in the 20th century, requiring better methods to create mattes to be invented. However, matting still remained a very complicated process. Newer technologies allow for a new and easier kind of matting: chroma key compositing.

Chroma key compositing, in short, is filtering out a range of colors such that they are no longer visible. Blue and green are the most widely used colors for this process, with green being more dominant in modern chroma keying. To perform matting with chroma key, we set up a monochromatic screen (green, blue, etc.) as the background on which the foreground object is placed. When filmed, anything having the color chosen for the background is removed from the screen. This makes it far easier to compose videos together.

The drawback of this method is that careful setups are still required. Lighting must not be taken for granted as insufficient lighting makes it difficult to produce a sharp matte. Subjects and props positions must be carefully placed as well, otherwise the light of the background can be reflected onto their surfaces.

Those disadvantages of chroma key compositing motivate us to propose an application that can help doing video matting with minimal setups.

3.3.2. Description

The application needs the following inputs:

- A video sequence for foreground extraction
- A video sequence on which the extract foreground of the first video will be placed.

The user will be asked to specify the object of interest in the foreground video. After processing is done, the output of the application is a composition of the two given videos. The object that is specified by the use in the first video will be placed into the second video, blending in as much as possible.

The application will make use of our proposed segmentation method to extract the foreground objects out of the first video. The extracted foreground can then be cloned into the second video.

3.4. Conclusion

In this chapter, the authors describe clearly the problem that are targeted to solve, which is salient object segmentation on video. Moreover, the authors mention the approach to tackle the problem, the model for solving video object segmentation, and the differences when applying visual saliency to video.

The chapter also presents chroma key compositing for video matting, the process of creating “mattes” or masks that can be used to combine two different videos. The drawbacks of chroma key compositing drives us to propose an application for video matting that takes minimal amount of set up. The application makes use of our proposed method for segmentation.

Chapter 4

Experimental Results

✍ *Chapter 4 describes the DAVIS dataset and reports the results of the conducted experiments. The first set of experiments consists of tests on the efficiency of DHSNet in solving salient object detection for images and video sequences and how well the method can be used in a segmentation problem. The results of the first set of experiments are analyzed to devise a suitable method for salient object segmentation. The second set of experiments looks for a suitable tracking method to build up and enhance proposed model.*

4.1. DAVIS dataset for video object segmentation

DAVIS (**D**ensely **A**nnotated **VI**deo **S**egmentation) is a newly proposed benchmark dataset specifically designed for the video object segmentation problem [21]. Each sequence comes with pixel-accurate ground-truth binary masks for each of the frames. The dataset is said to cover major challenges for video object segmentation, ranging from cluttered background and scale variation to occlusion and motion blur [21]. Dataset content also varies, spanning several actions and four evenly distributed classes: humans, animals, vehicles, and objects [21]. Figure 4.1 shows a few examples from the dataset, with ground-truth masks overlaying sequence samples. The DAVIS dataset is comprised of two datasets. The DAVIS 2016 dataset aims to aid study on single object segmentation while the DAVIS 2017 dataset focuses on multiple objects segmentation.

The 2016 dataset is fitting for our purpose of testing the proposed methods. First, the content is diverse enough to ensure we cover enough cases when doing the first set of experiments. Second, the accompanying metrics proposed with the DAVIS dataset provides a meaningful and well-defined validation metrics to assess our method. Third, because the DAVIS 2017 dataset is not fully available, it is hard to

quantitatively evaluate our method. Therefore, the DAVIS 2016 is chosen for our study.

The 2016 dataset contains 50 sequences of high quality that last 2-4 seconds, totaling 3455 frames, captured at $24fps$ with Full HD resolution. A 480p version of the dataset is also provided for easier training and testing. As a note, the DAVIS 2016 dataset actually comprises of three sets, *train* – having 30 sequences, *val* – having 20 sequences, and *trainval* – the combination of the other two sets. For our study, the *trainval* set is used.

Figure 4.1. Examples from the DAVIS dataset

4.2. DHSNet for Video Object Segmentation

4.2.1. DHSNet for video salient object detection

The purpose of DHSNet is to solve image salient object detection, i.e. finding the most prominent object in the image. It is introduced as the state-of-the-art methods for salient object detection in 2016, producing marginally better results comparing to previous methods **[18]**. Thus, we choose the method as part of out system in hope that it performance would help making salient object segmentation practically possible.

In order to truly understand the power of DHSNet and what we can do with it, we conduct a set of experiments using DHSNet on the DAVIS 2016 dataset to see how well it performs on a new dataset. In short, the objective of this test is to:

- See how well DHSNet, an image salient object detection method, works on a new but diverse dataset.
- Assess the ability of DHSNet to apply to a segmentation problem.
- See if DHSNet running on individual video frames can be used as video object segmentation, i.e. judging the result on a whole video sequence instead of on a single image.

DHSNet is first run on each frame of each DAVIS sequence separately to produce saliency maps of 3455 individual frames. The testing is done on a computer with 16GB RAM and a NVIDIA GTX 1060 6GB. Note that we threshold the saliency maps, whose values range from 0 to 1, at 0.5 to turn them into segmentation results.

(a) blackswan (b) dance-twirl

Figure 4.2. Examples of testing DHSNet on the DAVIS 2016 dataset

Results:

Detailed results are as follows:

- Running time is astounding, taking only 0.061018s for each of the frames, totaling to only three and a half minutes for the whole DAVIS dataset.
- Saliency maps for sequences with simple background and simple (e.g. monochromatic) objects are great, with results almost resembles those from segmentation methods (Figure 4.2.a).

- For cluttered background sequences, the saliency map results are hit-or-miss visually, with several saliency maps contain noises from the background (Figure 4.2.b). DHSNet also fails to highlight the entire salient object if the object is complex, such as a person wearing clothes of colors that deviate strongly from the human skin color.

Apart from cases presented above, there are some situations when DHSNet cannot detect any salient object at all. This problem exists for images in which there are barely any object in the foreground or the main object in the scene is occluded or tinted such that it is not easily distinguishable from the background.

Overall, there are sequences that DHSNet performs extremely well as a salient object detection method, having almost no trouble separating objects from cluttered background. Some of the results are even good enough to be considered as a segmentation result. On the other hand, with some of the sequences, it produces noisy results or no results at all. The results when evaluated as sequences instead of individual frames show another problem. A good video object segmentation method typically has the object of interest segmented consistently throughout the frames, e.g. the edges of the segmented regions are stable. If we view the results produced by DHSNet as a video object segmentation result, the saliency map for each frame can vary a lot and thus makes it an unreliable method for segmentation. Furthermore, our purpose of using a salient object detection method is to apply to a segmentation problem. The results of the first test on DHSNet show that while this is an incredibly fast method that produces good results for salient object detection, the vanilla method does not work well in object segmentation in general.

From the results of the first test, we can see that the global information extracted by the GV-CNN to produce the coarse saliency map greatly helps pinpointing the prominent object in the scene. The HRCNN in DHSNet also helps refining the **saliency map to closely fit the salient object**, making the method very promising to be used for segmenting objects. However, there are two reasons why DHSNet cannot work out-of-the-box as a segmentation method. First of all, the method does not aim

to solve this problem and thus does not employ a first-frame mask that is typically required in video object segmentation methods. Without a mask guidance, we are missing some crucial information about the scene. Global information helps a great deal in DHSNet, but **for objects that do not strictly deviate from the background, DHSNet treats them as part of the background**. Second, the **temporal information is not utilized** when we run DHSNet on individual frames. If each frame is considered separately, the segmentation output cannot be guaranteed stable throughout the sequence.

Each of the two problems needs to be studied and solved before we can integrate DHSNet to the proposed system. For the first problem, we propose a hypothesis: Pre-locating objects of interest helps DHSNet detect objects better and produce better segmentation results. The next experiment proceeds to test the hypothesis.

4.2.2. Guided DHSNet for video object segmentation

We know that one of the strengths of DHSNet is being able to incorporate global view information effectively to detect and localize salient objects. However, there are cases when global information cannot localize any salient object in the first few frames of the sequence. To counter this, we use the information from the mask first frame of the sequence to help guide the method to detect objects better. The reason is that DHSNet does not solve the object segmentation problem and thus does not take in any information regarding the whereabouts of the object of interest. Bringing information from the mask might help adding required.

The experiments thus aims to achieve to goals:

- Test the hypothesis: Pre-locating objects of interest helps DHSNet detect objects better and produce better segmentation results, i.e. restricting the region of interest will let DHSNet produce better results.
- See if the first-frame mask of a sequence can improve segmentation results of that sequence.

Figure 4.3 illustrates the structure of how mask information is used with DHSNet. The ideas is as follows: With the first frame of the sequence, we use the mask to extract the bounding box for the object of interest. The bounding boxes are indications of object locations and the region within each bounding box can be treated as separate image. These regions are thus cut out and fed to DHSNet. Saliency map for each of the extracted regions will then be merged into a final saliency map acting as the result of the whole frame. The final step includes thresholding and doing some refinements to the saliency map to obtain the segmentation result.

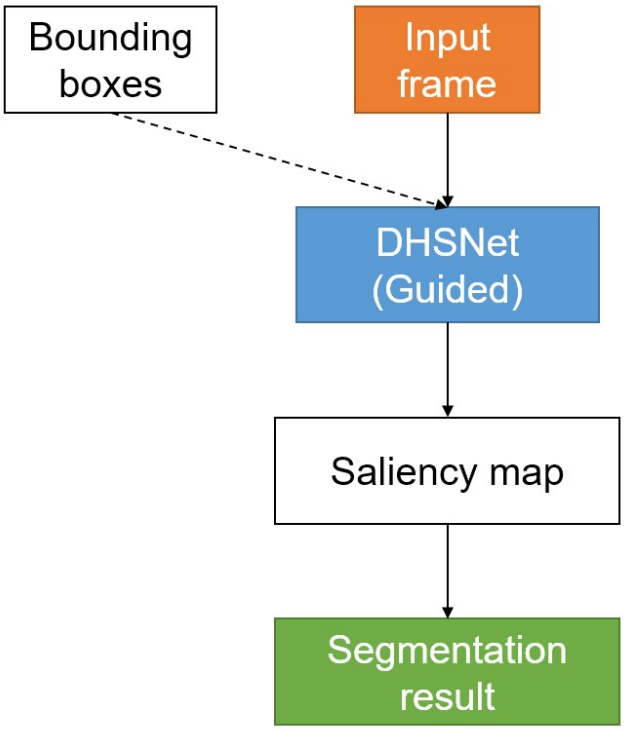


Figure 4.3. Structure of a guided DHSNet

For each of the frame following the first one, the object indicated by the first frame will be tracked to create bounding boxes associating with the frame. The process continues as normal, using the bounding boxes and the frame data to produce the segmentation result.

For cleaner implementation and modularization of the method, we decide to separate the tracking step from this test and use the ground-truth masks for all of the frames instead. The reasons are twofold. First, tracking plays an important role in this implementation. Thus, a good tracking method that can provide accurate and consistent result throughout the sequence is needed. Should any faults happen when we do the tracking, the test is thrown off. Separating the tracking out of this test allows us evaluate the results of Guided DHSNet more accurately. Second, using the ground-truth masks to extract bounding boxes as replacement to tracking methods gives us a kind of baseline result to evaluate our tracking module when the whole system is put together. Since the ground-truth masks ultimately give the best tracking possible, when it is applied to Guided DHSNet, we should obtain the best possible results. The results when applied any tracking modules later can be compared to the ones produced by this test.

(a) (b)

Figure 4.4. Comparisons between DHSNet and Guided DHSNet. (a) DHSNet results. (b) Guided DHSNet results.

The bounding boxes extracted from the masks are also expanded by 30% for each dimension because limiting the regions of interest too much may cause a heavy loss in background information, dampening the effectiveness of the method.

The results of this test are as follows:

- The running time takes a mild hit but is still very fast. It takes 0.090633s to run the Guided DHSNet for a frame.
- The method is able to detect salient objects better now thanks to the pre-localization and region restriction. Visually, it greatly improves the results for some cases, e.g. saliency maps for sequences with drifting cars can be used for segmentation now as they highlight exactly the cars instead of some random regions or none at all (Figure 4.4). There are also less noise comparing to the full-frame version, since the region is smaller (Figure 4.4).

Quantitative comparison between DHSNet and Guided DHSNet is given in Table 4.1, in which the Jaccard (region similarity) and Boundary (contour accuracy) scores are calculated. It is indeed that Guided DHSNet gives a better performance.

	DHSNet	Guided DHSNet (Ideal)
$\mathcal{J}$ **mean** ↑	0.618	0.756
$\mathcal{J}$ **recall** ↑	0.719	0.902
$\mathcal{J}$ **decay** ↓	0.033	0.043
$\mathcal{F}$ **mean** ↑	0.572	0.722
$\mathcal{F}$ **recall** ↑	0.615	0.870
$\mathcal{F}$ **decay** ↓	0.037	0.052
$\mathcal{T}$ **(GT 0.095)** ↓	0.395	0.348

Table 4.1. Quantitative comparison between DHSNet and Guided DHSNet (Ideal) for the *trainval* set of DAVIS 2016

The temporal stability score $\mathcal{T}$ for Guided DHSNet is also lower than that of DHSNet (the lower the better), which indicates that the segmented regions for a sequence are stable throughout the frames. This might be attributed to the decrease in false positive regions returned when running DHSNet with bounding boxes as guides.

A quick comparison with the state-of-the-art methods for video object segmentation on single object on DAVIS 2016 evaluation set shows that **Guided DHSNet with perfect bounding box outperforms VPN, the third-ranked method as of July 2017**. Furthermore, we test Guided DHSNet on the *test-challenge* set of the DAVIS 2017 dataset and qualitative results can be seen in Figure 4.5.

Note that quantitative results are not possible on the DAVIS 2017 *test-challenge* set because the ground-truth is not made publicly available. Since there is no ground-truth, we inherit the bounding boxes tracking the objects from the publication we contributed to (See APPENDIX - PUBLICATION) to create semi-ideal results. Guided DHSNet has some problems identifying different objects of interest, but overall, the method can still be used to segment objects out of the scenes.

Figure 4.5. Guided DHSNet on the *test-challenge* set of DAVIS 2017 dataset

The improvement to DHSNet results shows that the use of saliency as an object segmentation method is possible. However, before we move on to the next set of experiment, we still want to test if we can improve the method further.

4.2.3. Guided DHSNet with multiple divided regions

The next small experiment deals with studying and finding a way to improve the current Guided DHSNet results. In some of the current results, the segmentations still leave out some regions within the object. Furthermore, background that are quite prominence diverts attention from the object of interest, producing results with high rates of false positive. We propose an idea to tackle this problem. Based on the assumption that large region of interest still cannot sufficiently pinpoint the object of interest, we divide the region of interest taken from the bounding box into smaller parts. Each of the parts will be fed into DHSNet and then the results are merged into a single final result. The smaller parts can be either overlapping or not. The quantitative results of a few test runs are as follows:

	non-overlapping 224×224 regions	overlapping 448×448 regions	overlapping 360×360 regions
$\mathcal{J}$ **mean** ↑	0.623	0.727	0.727
$\mathcal{J}$ **recall** ↑	0.700	0.879	0.883
$\mathcal{J}$ **decay** ↓	0.062	0.036	0.039
$\mathcal{F}$ **mean** ↑	0.636	0.689	0.692
$\mathcal{F}$ **recall** ↑	0.723	0.841	0.856
$\mathcal{F}$ **decay** ↓	0.069	0.051	0.052
$\mathcal{T}$ **(GT 0.095)** ↓	0.798	0.461	0.484

Table 4.2. Comparisons between proposed improvement methods for the *trainval* set of DAVIS 2016

From Table 4.2, we see by dividing the region of interest into smaller parts, the performance of the system decreases, giving worse $\mathcal{J}, \mathcal{F},$ and $\mathcal{T}$ than the Guided DHSNet. As the size of the divided parts increases, we achieve better results. However, those results do not have any significant meaning as the input images for this test are of size 854 by 480 pixels. Increasing the size of each part to 360×360 or 448×448 means that for most frames, the bounding box contains exactly one part

that has the size of the box itself. Therefore, the performance approaches the performance of the Guided DHSNet as the part size gets bigger. Therefore, dividing the bounding box of a frame to smaller parts gives worse performance than the Guided DHSNet.

Furthermore, visual results show that there are regions in the objects that cannot be segmented because the associating divided regions of interest are monochromatic, leading to no salient object detected. In fact, the ratio between the area of the object and the area of the region of interest seems to affect saliency results returned by DHSNet.

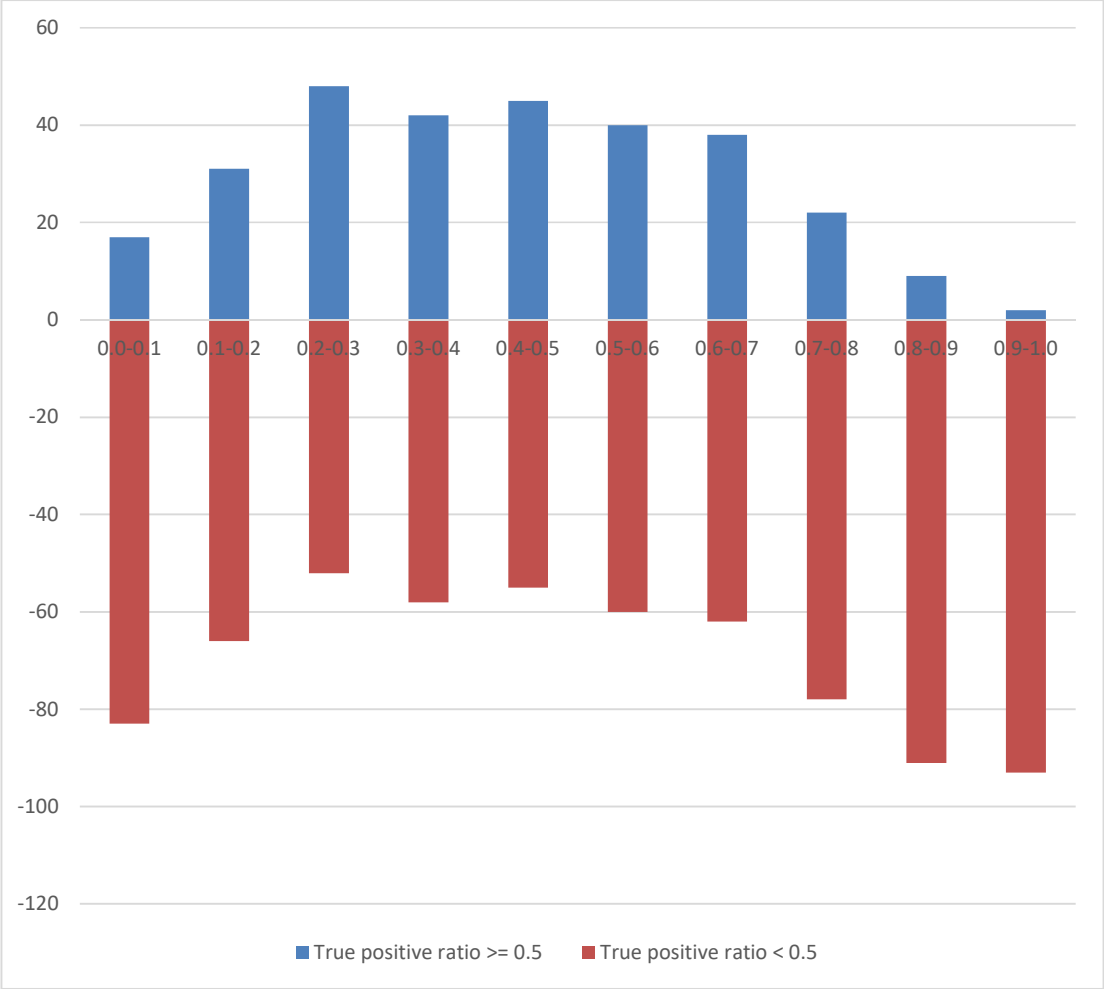


Figure 4.6. True positive rate for different object over region ratios

Figure 4.6 illustrates the relationship between true positive ratio of saliency results and object over region ratio. The data for the plot is created by taking randomly 1000 patches (small regions) from the DAVIS dataset that contains an object of interest and using the patches as input for salient object detection. The results are coupled with the ground-truth to calculate the true positive ratio. We can see that **good results mostly have object over region ratio fall in the range** $[\mathbf{0.2}, \mathbf{0.7}]$. The regions of interest used in our test of the Guided DHSNet already match this and dividing those regions further only decreases the performance. Thus, we see no point in continuing the little experiment and move on to the next set of experiments.

4.3. Tracking objects

In order to pass object location information from the first frame to the rest, we decide to use a tracking method. Two experiments are conducted in this section. The first experiment aims to test YOLO as a tracking method and the second experiment uses optical flow for tracking purpose.

4.3.1. YOLO as a tracking method

YOLO is an object detection method that can produce great detection results amazing speed.

There are two reasons why we choose YOLO as a tracking method. First, bounding boxes is of great importance to our system because they directly affect the result of saliency detection. If the bounding boxes returned by the tracking method fail to sufficiently cover the objects, the Guided DHSNet cannot produce saliency maps fitting for video object segmentation. A detection method, such as YOLO, can overcome this problem. With a detection method, the output pinpoints where the objects are and what their classes are. The boxes returned closely fit the objects and a little expansion can ensure sufficient regions of interest, enabling the use of saliency. Second, YOLO covers a wide variety of classes and YOLO9000 can even detect over 9000 object categories. This coverage allows YOLO to detect and thus generate bounding boxes for a large range of objects, making it more unlikely to fail to create regions for salient object detection.

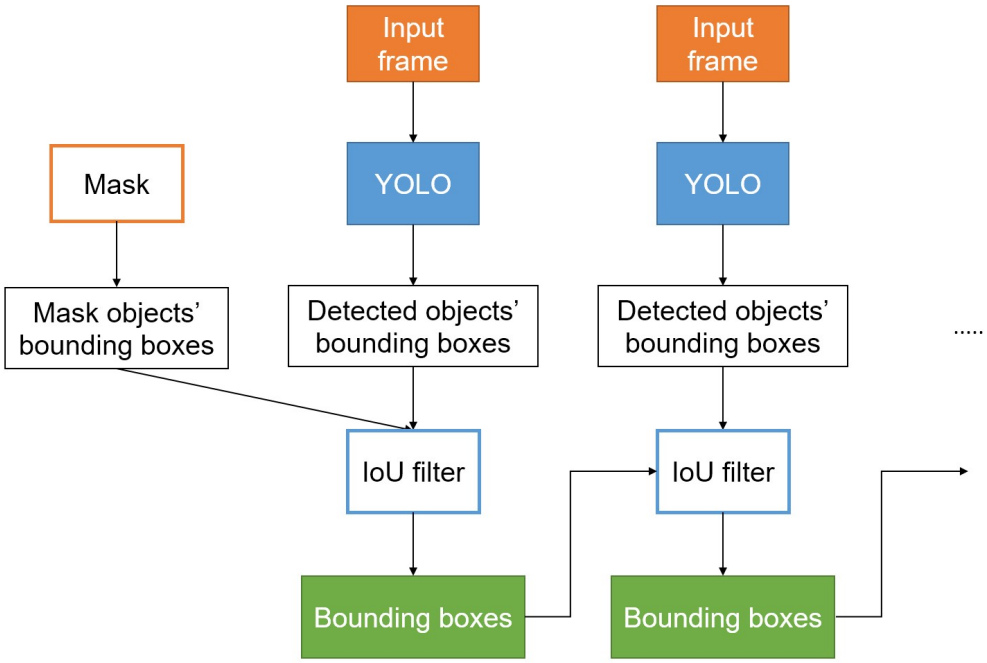


Figure 4.7. Tracking objects and generate bounding boxes using YOLO

Figure 4.7 illustrate the process of tracking and generating bounding boxes using YOLO. With each frame in a sequence, YOLO outputs a list of bounding boxes for all detected objects. For the first frame in the sequence, we extract bounding boxes from the ground-truth mask. Those mask objects' bounding boxes are used as filters to decide which YOLO-detected objects' bounding boxes should be kept. The result is a list of bounding boxes for the frame that will be treated as regions of interest over which Guided DHSNet is used. For the second frame onward, YOLO also generates a list of bounding boxes for each frame. However, the filters used this time are no longer extracted from the masks. Instead, the resulting bounding boxes of the previous frame are used. The process continues until the end of the sequence.

To filter the detected objects' bounding boxes, we make use of **Intersection over Union** (IoU), also known as the **Jaccard index**. Simply put, to find IoU of two bounding boxes, we calculate the ratio between the area of intersection of the two boxes and the area of boxes' union. Any of the detected objects' boxes having an IoU with any of the filter boxes greater than a determined ratio is considered a bounding

box for the current frame. The idea behind this is the locations of objects in the next frame are in the vicinity of objects in the current frame.

The results of this tracking method are as expected, giving bounding boxes usable by the Guided DHSNet, and thus works for our purpose. However, **the method fails for sequences which contain objects that move fast**. Because IoU is affected by the positions of the detected boxes, objects moving fast generally gives a lower IoU between frames. Setting the threshold for IoU lower to try to cover these instances is not recommended since that may lead to falsely capturing objects not of interest. Therefore, the method works nicely, but rather limited and not feasible in the long run.

There is also one more problem with this method. With YOLO, we **cannot effectively handle temporal information**. In other words, there are no real connections between frame data. Tracking with YOLO this way may help us segment objects out of the scene, but the segmentation is still unstable because each frame is still considered individually. The next experiment is created to investigate this problem.

4.3.2. Optical flow as a tracking method

4.3.2.1. DeepMatching, DeepFlow, and EpicFlow

DeepFlow and EpicFlow are two of the most prominent method to find optical flow. Coupled with DeepMatching, the method can achieve good performance for fast motion cases. DeepFlow and EpicFlow are used for this experiment as both a means for tracking object positions and for inclusion of temporal data.

The problem with the previous experiment with YOLO is that although it is able to give a bounding box for each object in each frame needed by Guided DHSNet, there is no way to utilize the temporal data and the segmentation results are still not stable. The two problems are closely knitted, intuitively because if we are not linking the frame data, there is hardly any guidance on to create a consistent segmentation through the frames.

Optical flow helps solving the problems by letting us propagate some information of a frame to the next one. From two input frames of a sequence, DeepFlow and EpicFlow produce pixel displacement (flow) for each of the pixels in the first frame. This helps create a kind of mapping of the pixels in the first frames to the pixels in the second frame, giving us information on how the pixels of the first frames move through time. Figure 4.8 gives examples of the result of running DeepFlow on a sequence in the DAVIS 2016 dataset. For each of the examples, two frames are used to compute the displacement. Given an object mask, e.g. the mask for the bear in the example in Figure 4.8, the displacement can be applied to the pixels in the mask, causing them to translate. The result is exactly what is shown in Figure 4.8.

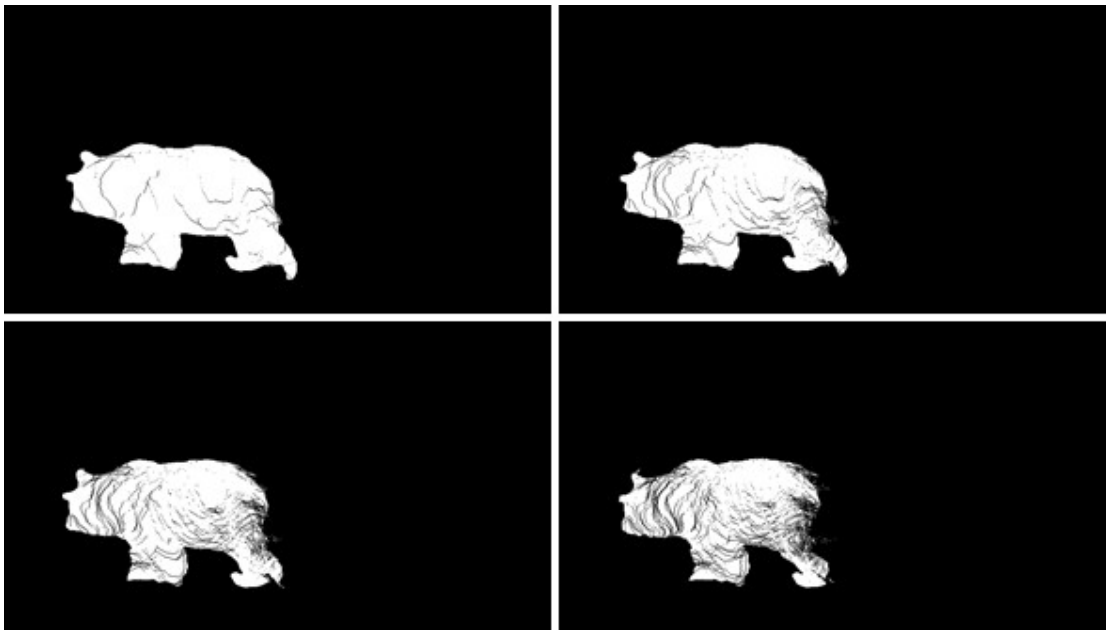


Figure 4.8. Results generated by DeepFlow

The translated pixels of the result indicate where the pixels in the first frame are now in the second frame. By letting pixels move this way, we allow information of the shapes of objects obtained by segmentation in the first frame to pass through to the rest of the frames. Thanks to this information, segmentation in subsequent frames can be more stable. Furthermore, DeepFlow also updates the shapes of the objects according to the frames, bringing extra information to refine the segmentation.

This experiment sets out to determine the efficiency of DeepFlow and EpicFlow as a tracking method for the system, i.e. how well can it be used with Guided DHSNet. We first use DeepFlow and the exact process described above for the example on each of the sequences in DAVIS 2016 dataset.

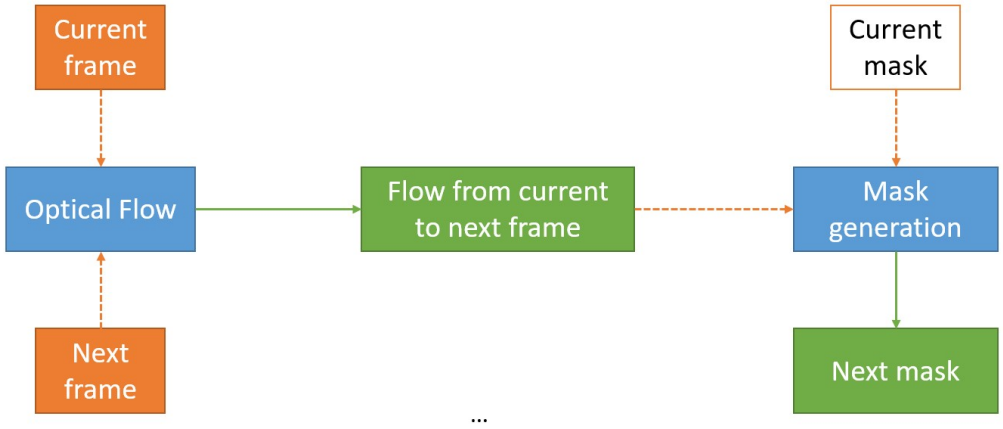


Figure 4.9. Mask generation using optical flow

For this test, the mask of the first frame of each sequence is used. Figure 4.9 is the flow chart showing how masks are updated using optical flow. For each frame in the sequence, we use the current frame and the next frame as inputs to DeepMatching and DeepFlow, producing a flow result. The mask is then updated using the flow, producing the mask for the next frame. The mask is continuously updated using this method, propagating mask information through the sequence. Using this method, a new bounding box for the object in the next frame can be generated, giving a bounding box for the next frame to run Guided DHSNet.

The test fails to meet our expectation as problems with DeepFlow heavily affect the accuracy of the updated mask. There are three main problems with the method.

Figure 4.10 illustrates *the first problem* with the approach. Updating the mask continuously using DeepFlow may leave behind noises when pixels in the object cannot be mapped to the next frame. The current implementation for this test creates bounding box containing all pixels whose values are positive in the mask. **Updating the mask through optical flow using only the current mask increases the amount of noise**, gradually expanding the bounding box of the object to cover regions not of interest.

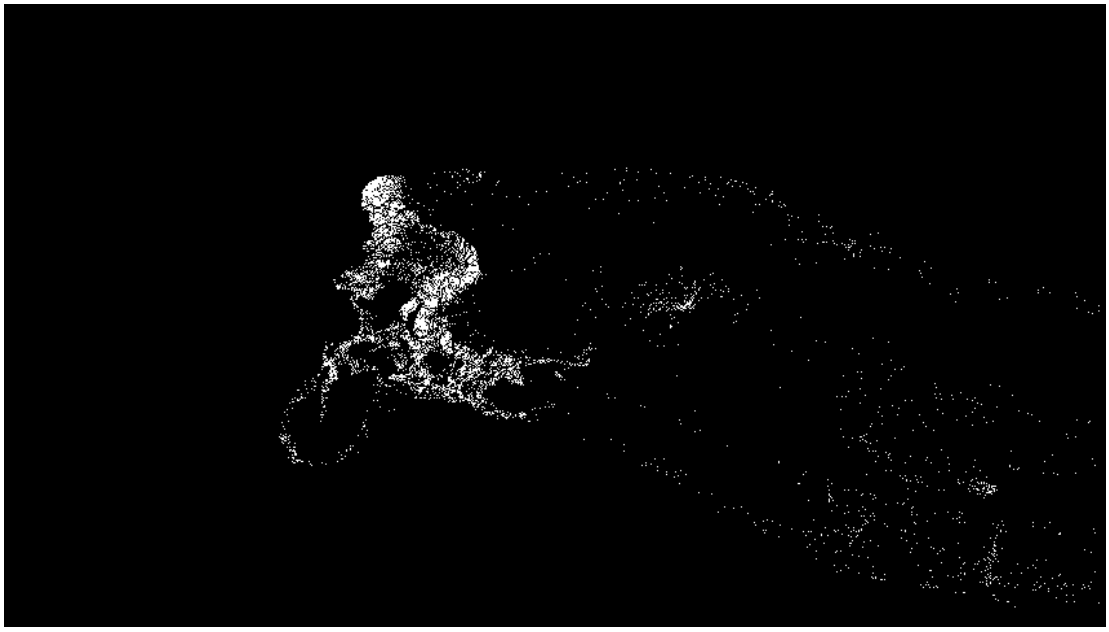


Figure 4.10. Updated mask result for a frame in the sequence *bmx-bumps*

Gradual expansion of object bounding boxes results in a worse performance for the system, as Guided DHSNet slowly turns into the normal DHSNet running on a full frame. The test results show that there needs to be a way to remove the noise from the mask.

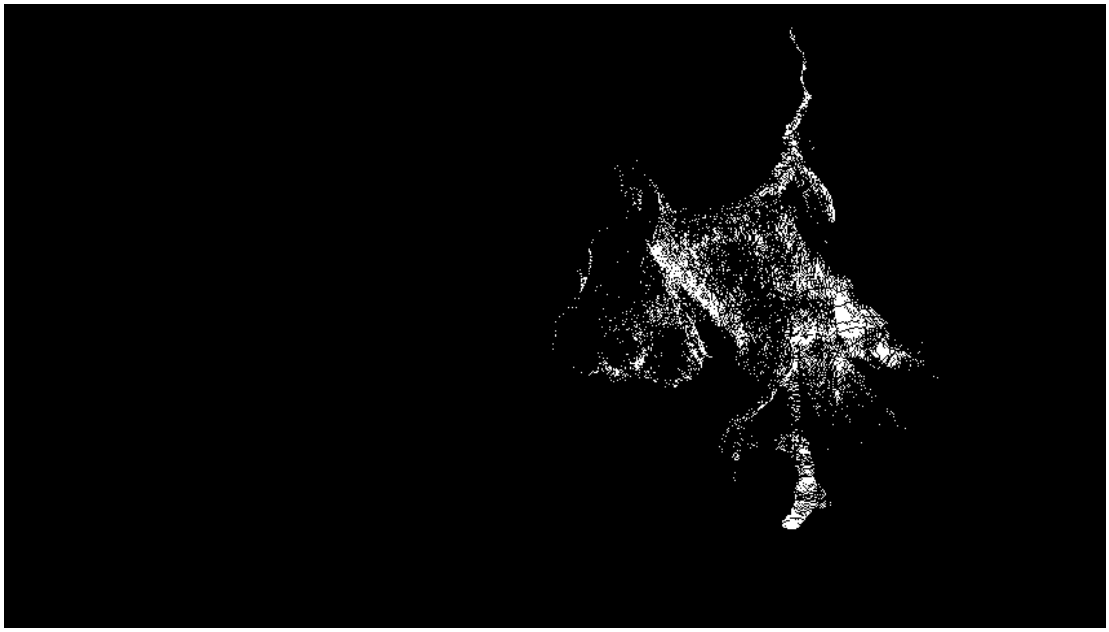


Figure 4.11. Updated mask result for a frame in the sequence *dance-jump*

The second problem with using DeepFlow to generate new masks from old ones is **the method does not work well with objects that transform with motion**. Figure 4.11 is the result of updating the mask of *dance-jump* using optical flow result of DeepFlow. As the dancer in the sequence twirls and jumps while she moves, the update leave behind many pixels that cannot be mapped, producing a mask that can neither be used for tracking nor refining.

The third problem with this method is the **tracking is interrupted** whenever the object is occluded **and stopped** whenever the object disappears from the scene. Since we are matching consecutive frames and updating mask pixel positions, lost pixels on the object due to occlusion cannot be mapped and thus remains as noise. Figure 4.12 illustrate this problem. The sequence *bmx-trees* consists of a man riding a bike, occluded by a tree as he passes by it. The updated mask shows mask pixels stuck at the occluding tree. When the object disappears from the scene, tracking is lost and cannot resume because matching consecutive frames does not give enough information on the whereabouts of the object when it reappears in the scene.

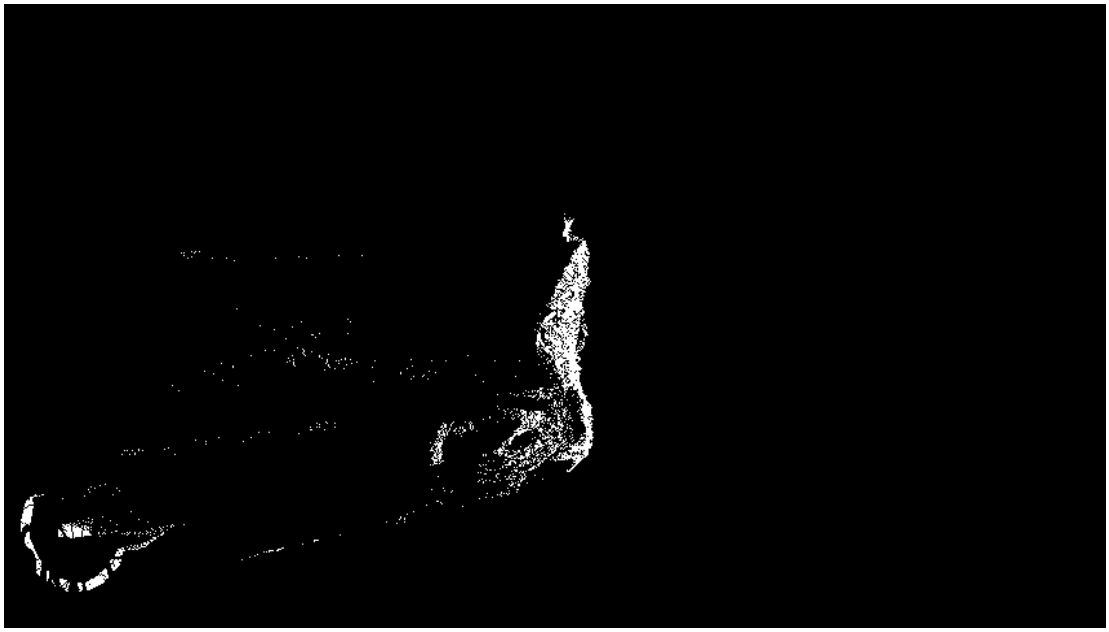
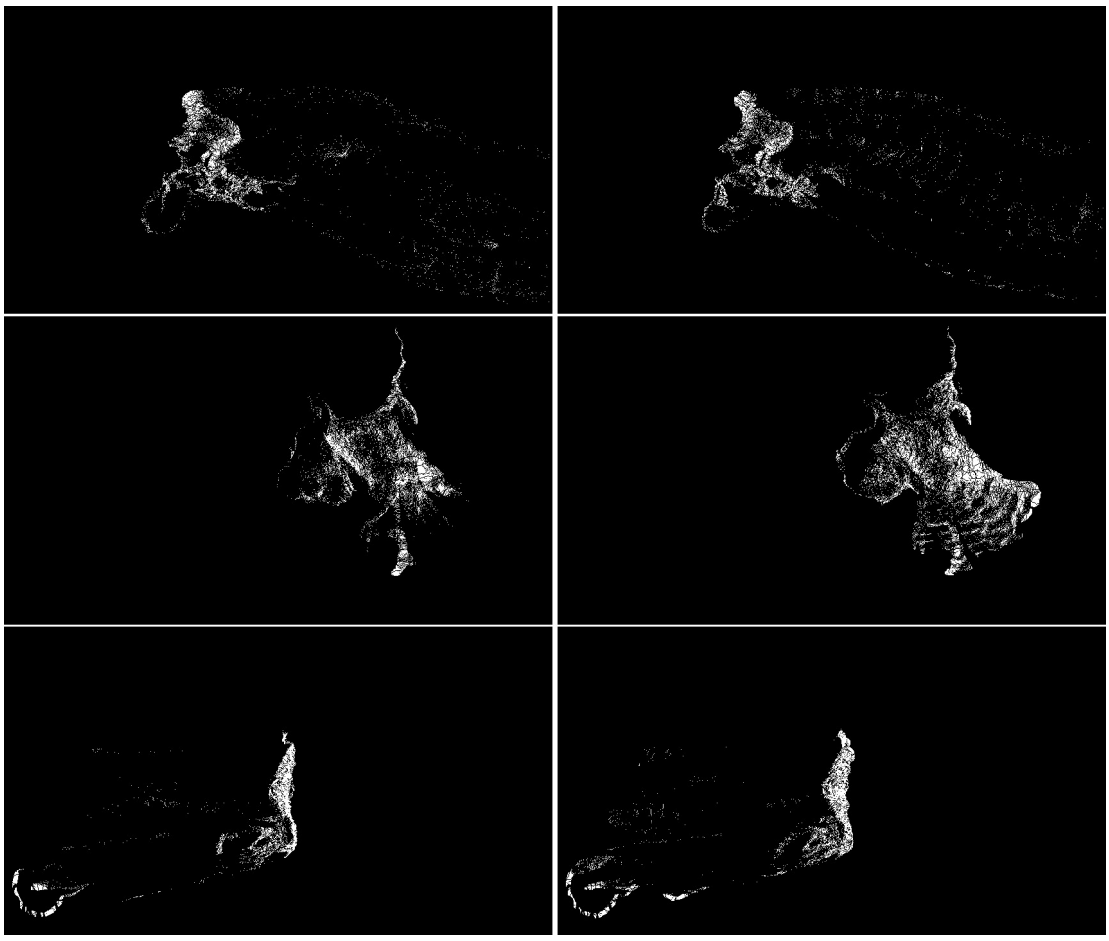


Figure 4.12. Updated mask result for a frame in the sequence *bmx-trees*

EpicFlow is a better method for creating optical flow with large displacements proposed by the same authors of DeepFlow. The method outperforms DeepFlow when compared on different datasets. To determine whether the failure of our test is due to DeepFlow or optical flow in general, we decide to redo the test with EpicFlow. The results show that the three problems exist even when we use EpicFlow to generate optical flow (illustrated in Figure 4.13).



DeepFlow EpicFlow

Figure 4.13. Problems with DeepFlow persists with EpicFlow

Therefore, the problem lies with using optical flow to update the mask in general. The next sections present our proposal to tackle these problems. It should be noted that from this point onwards, optical flows are obtained by using EpicFlow instead of DeepFlow because of its performance.

4.3.2.2. Noise removal using mean and standard deviation

This test aims to solve the first problem presented in section 4.3.2.1. The size of the bounding box is affected by noise and thus gradually expands as the mask is updated throughout the sequence. This is undesirable since we miss both the position and the size of the object.

From our observation, although some pixels in the mask object are left behind during the update process and become noise, most of the mask pixels still follows the true object motion and form a cluster of indicating the position of the object in the next frame (Figure 4.10). By determining where the cluster is, it may become possible to get the position of the object in the next frame.

Arithmetic mean can be used to approximate the position of the center of the cluster. With each mask, we have the distribution of pixels on the x and y axes. Getting the mean values for the distributions on x and y axes should approximately pinpoint where the center of the cluster is. With the position of the cluster's center obtained, the next problem is how to obtain a suitable box size, i.e. how to get the size of the cluster.

On the same train of thought, the distribution of the positions of the pixels on x and y axes can be used to tell us how the pixels are distributed with respect to the center of the cluster. Assuming the positions of the pixels on the x and y axes of the real-valued mask updated by the flow form normal distributions, by calculating standard deviations of the two distribution and apply the 68-95-99.7 rule, we can approximate the width and height of the bounding box for an object. Given the mean of the positions, taking the area within a number of standard deviation should yield a box containing a good part of the cluster.

Observations show that most objects in the sequence move in a horizontal direction. Therefore, we make the assumption that for videos, the motion shown in a sequence is horizontal to make full use of the capturing width. That means updated masks are more susceptible to loss and noise in the horizontal direction. Using the assumptions

above, we propose a way to crop the updated mask such that it is less affected by noise.

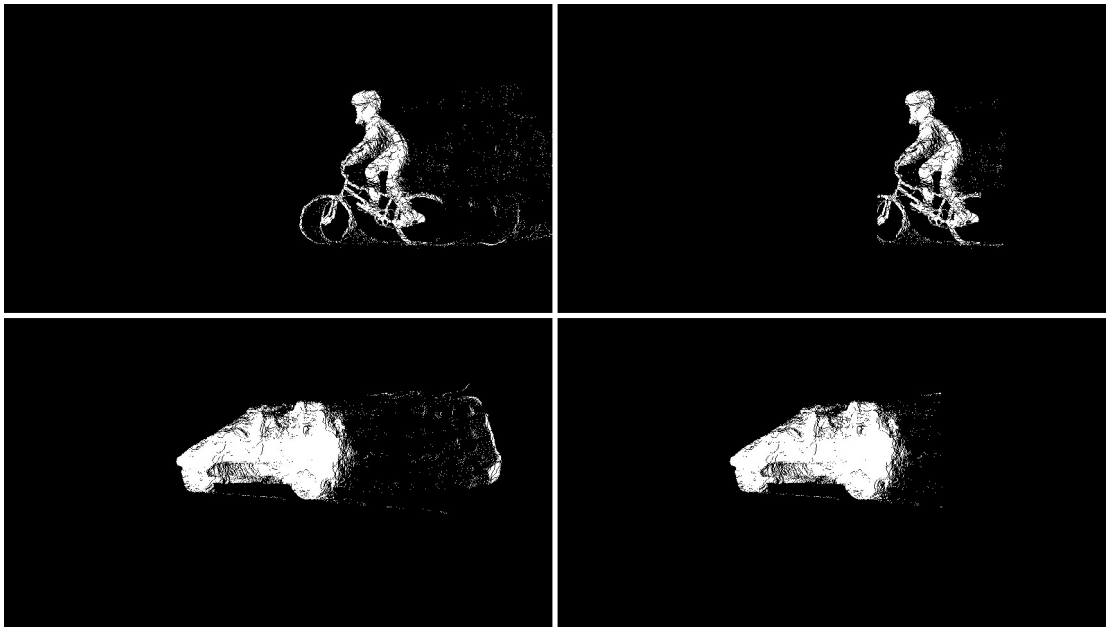
Given X and Y as the set of all real-valued x-positions and y-position of object pixels respectively, μ_X and μ_Y the mean of X and Y, and σ_X and σ_Y the standard deviation of X and Y:

$$left = \mu_X - k \times \sigma_X$$
$$top = \mu_Y - 3 \times \sigma_Y$$
$$width = 2 \times k \times \sigma_X$$
$$height = 2 \times 3 \times \sigma_Y$$

Since nearly all values lies within 3 standard deviation of the mean for a normal distribution, setting the box height to $6 \times \sigma_Y$ allows the height of the box from the updated mask to remains nearly the same. For the width of the box, since we are trying to remove the noise in the horizontal direction by discarding outliers in X, we must choose a suitable value for k such that a good amount of noise is removed while the object is not overly cropped.

Figure 4.14 shows the result of applying mean and standard deviation as a bounding box refinement method. The bounding boxes are indeed closer to the object, making the mask contains less noise.

However, the noise removal method relies on the assumption that positions of points in the flow for the mask gives normal distribution, when it is actually not the case. Distributions are extremely dependent on the shape of the object. Figure 4.15 demonstrate an example where this noise removal method crops points belonging to the object because the mean is shifted to the body of the swan.



(a) (b)

Figure 4.14. Updated masks (a) without using mean a standard deviation to refine the bounding box (b) using mean and standard deviation to refine the bounding box

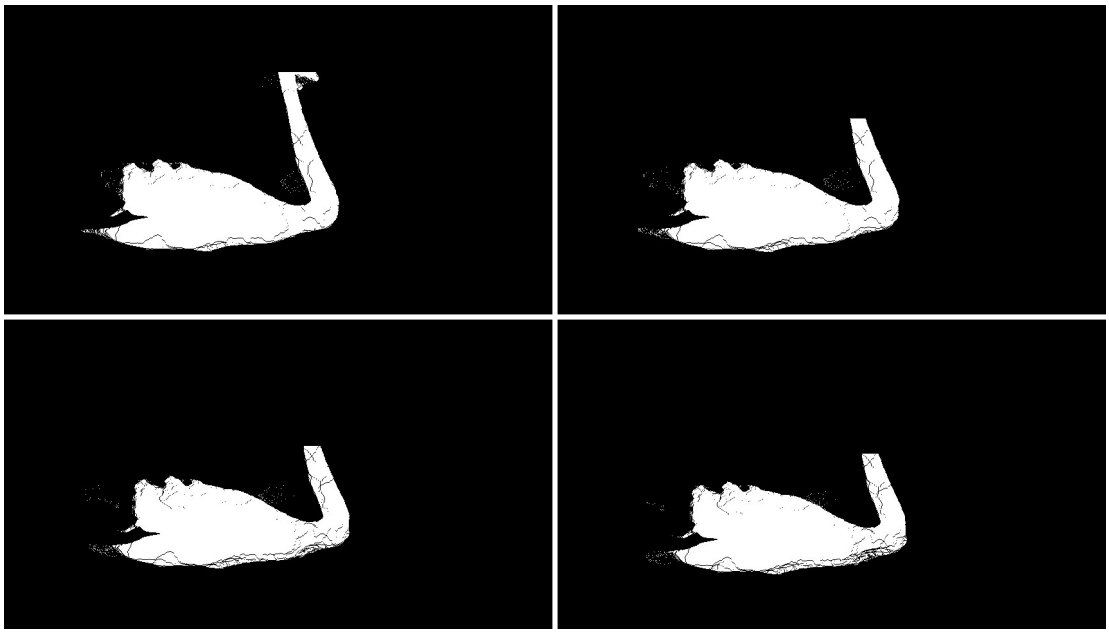


Figure 4.15. Bounding box refinement crops out object information in subsequent frames (left to right, top to bottom)

Therefore, this noise removal method cannot be used at all. Furthermore, even if the method is applicable, it there are still two more problems that need to be solved. The limitation to the current proposed method is that the tracking module and the segmentation module are still separated. The initial reason for this separation is to make it easier to improve the modules. However, as we conduct our experiments, it seems that we are **losing out information during the update of the frames, with later frames have increasingly more noise and more loss**. To be more specific, only the mask of the first frame is used as guidance for tracking in a sequence in the previous experiment. During propagation, a lot of the information of the mask is lost as pixels fails to match. The loss and the noise pertains through the mask, and each update to the mask create more loss and noise. Making use of more recent information, such as the segmentation result, to help create the mask for the next frame to counter loss seems reasonable. Therefore, we decide to update the proposed model illustrated by Figure 4.3 to include the tracking module.

4.3.2.3. Optical flow and segmentation result for mask generation

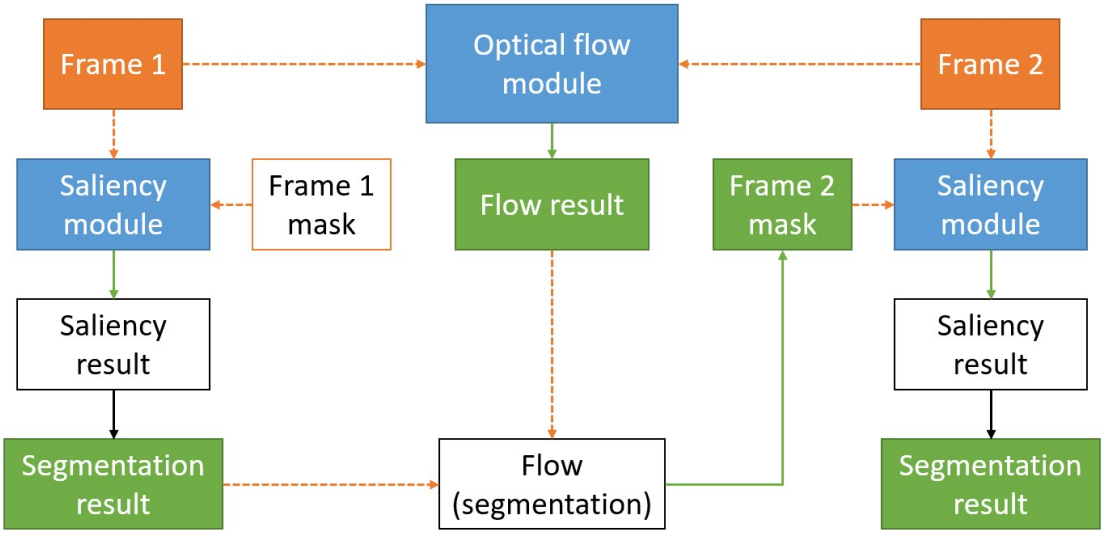
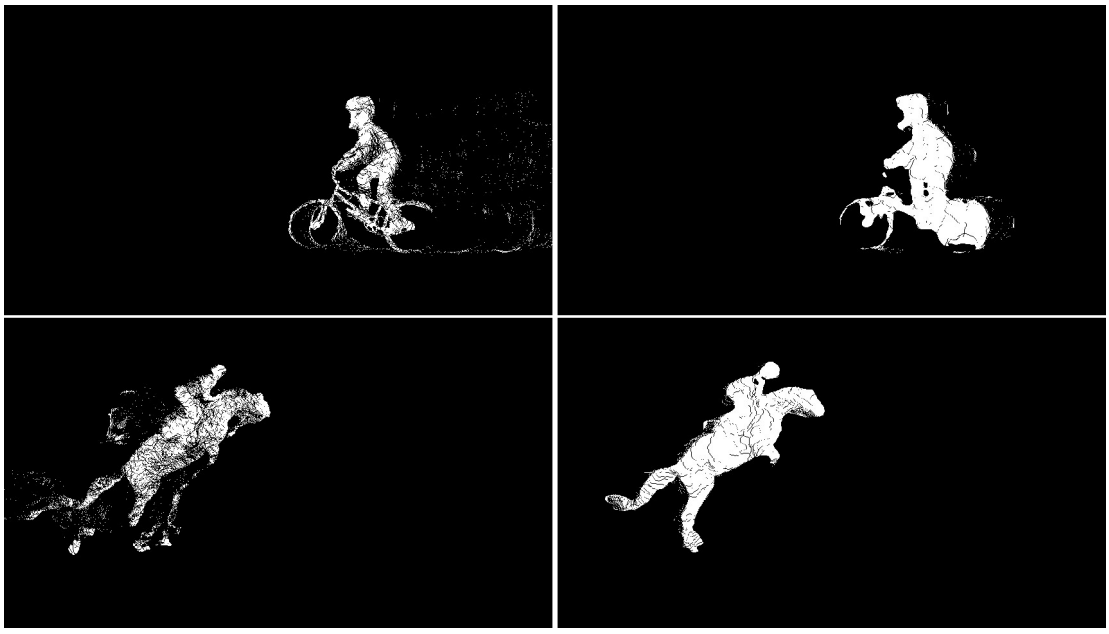


Figure 4.16. Updated model for video object segmentation that includes both segmentation module and tracking module

Figure 4.16 is the updated model that includes the segmentation module and the tracking module. With the previous model, only the information of the mask of the

first frame is used. Although optical flow can give some ideas about the location of the current object, information on the object obtained by the mask is subject to loss. Furthermore, lost information cannot be recovered and no way to update new information to the mask limit the current model.

The updated model can help ensure new information is updated to the mask at each frame. Instead of using the first frame's mask, we use the segmentation result with the generated optical flow to create the mask for the next frame. The reason we do this is because based on our observation of the previous tracking module, there are far less noise in the first few mask. Because we are updating the mask with new information from the segmentation, previous loss and noise are discarded instead of saved to the mask. Assuming good segmentation results, updating the mask this way might help us solve the first and second problem with the previous tracking module using optical flow.



(a) (b)

Figure 4.17. Updated masks using optical flow and (a) mask of the first frame (b) the segmentation result

(a) (b)

Figure 4.17Figure 4.17 presents the comparison between using the mask of the first frame with optical flow to update the mask and using the segmentation result for the same job. As can be seen, with new information used at each of the frame, the amount of noise in the mask drops. Because there are less noise in the mask, the bounding box for the object no longer covers areas not of interest. The updated mask for each frame becomes better guidance to the shape and position of the object.

However, careful inspection of the result of this method shows a problem. With the current segmentation module, the segmentation result is not always good. Bad segmentation result immediately distorts object mask for subsequent frames. Figure 4.18 is the result of the updated mask for a number of frames in the sequence *boat*. It can be seen that as bad segmentation result is used for the mask, the bounding box gets increasingly smaller and becomes unfitting to use for extraction of the region of interest. Furthermore, segmentation is carried out separately for each frame, causing inconsistency between masks. Those are the shortcomings when only the segmentation result is used for updating the next mask.

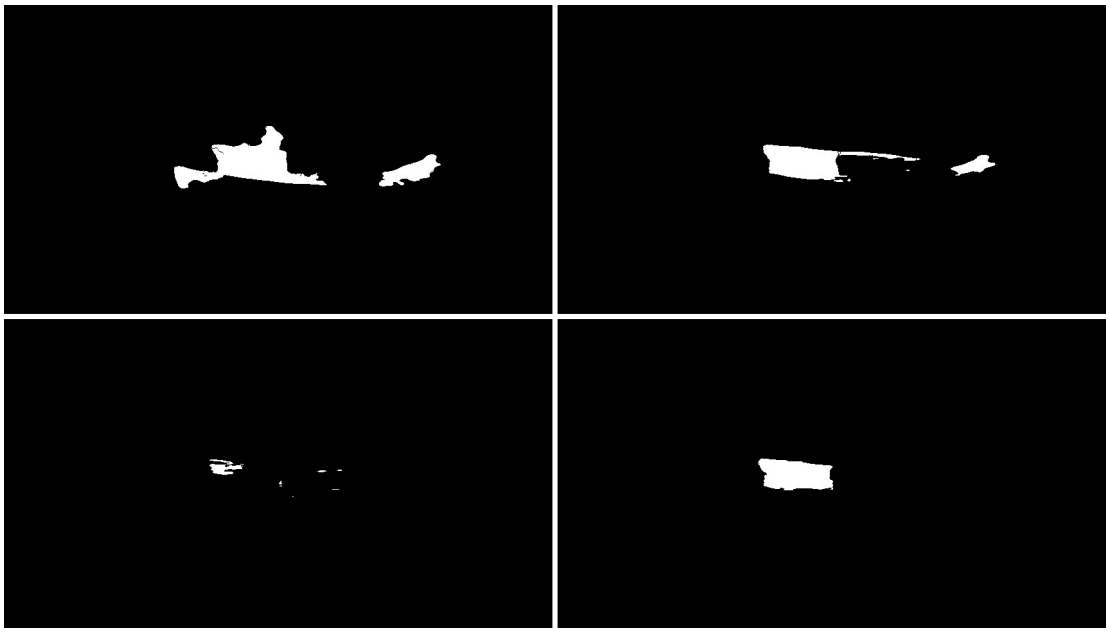


Figure 4.18. Updated masks for *boat* (left to right, top to bottom)

This solve this problem, we apply a lower limit for the size of the bounding boxes. The reason only a lower limit is applied to the bounding box size is because observations show that the bounding box size only increases when the object size grows. Using the segmentation result to update the mask help making sure the boxes cannot be expanded by noise. Thus, there is no point imposing an upper limit.

We propose a method to determine the lower limit of the bounding box size based on the following heuristics:

- The union of the mask and the segmentation result for a frame when used with optical flow provides better tracking results because of the increase in true positive.
- When the object of interest in a scene changes its size, the smallest size of the object is not lower than 20% the size when the object first appears.

The first heuristic is from observations made on experimental results. Our experiments show that when applying optical flow to the first frame mask, the result for the first few frames keeps the shape of the object close to the original albeit containing a small amount of noise. Furthermore, the object masks for those first few frames are not prone to shrinkage due to information loss, since information of the object for those frame is relatively new. When applying optical flow to the segmentation result at each frame, the size of the bounding box is no longer subject to expansion by noise and thus the overall box location is better. By combining the mask for the current frame and the segmentation result to update the mask, we might be able to counter both shrinkage and expansion to the bounding box size that is caused by size change to the object itself. Moreover, a union between the mask of the current frame and the segmentation results let new information of the object to be updated to the mask while keeping consistency in mask information.

The basis for the second heuristic stems from the result presented in section 0, in which we find out that when the object over region ratio fall in the range [0.2,0.7], DHSNet produces a good rate of true positive. By assuming the object size never gets

smaller than 20% the size when it first appears in the scene, we are able to easily define the lower limit for the size of the bounding box at each frame. Specifically, we can set the lower limit as the size of the bounding box for the object in the first frame of the sequence. In any frame of the sequence, the size of the object only gets as small as 20% its size in the first frame, leading to the object area over region ratio for that frame being approximately 0.2, at which DHSNet results are likely to be good.

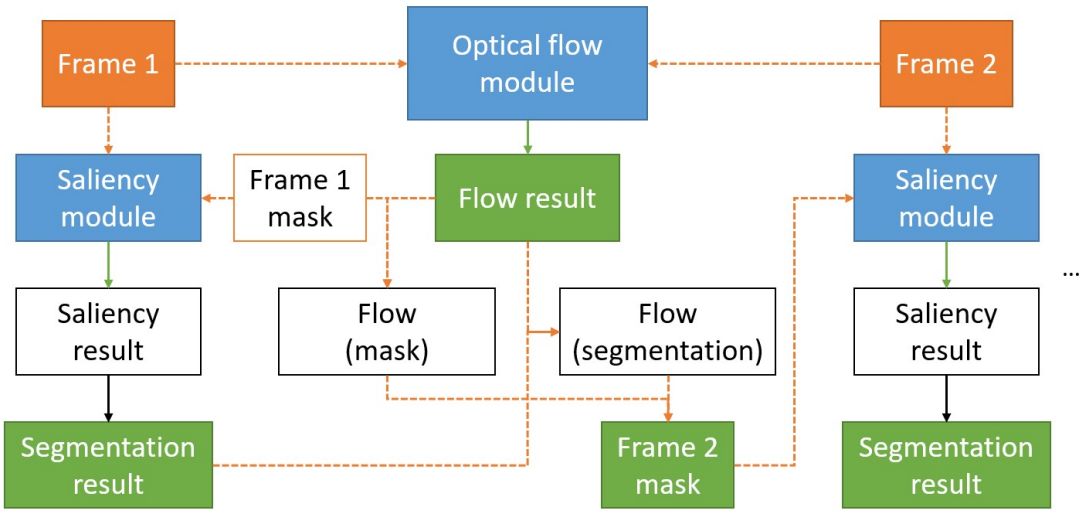
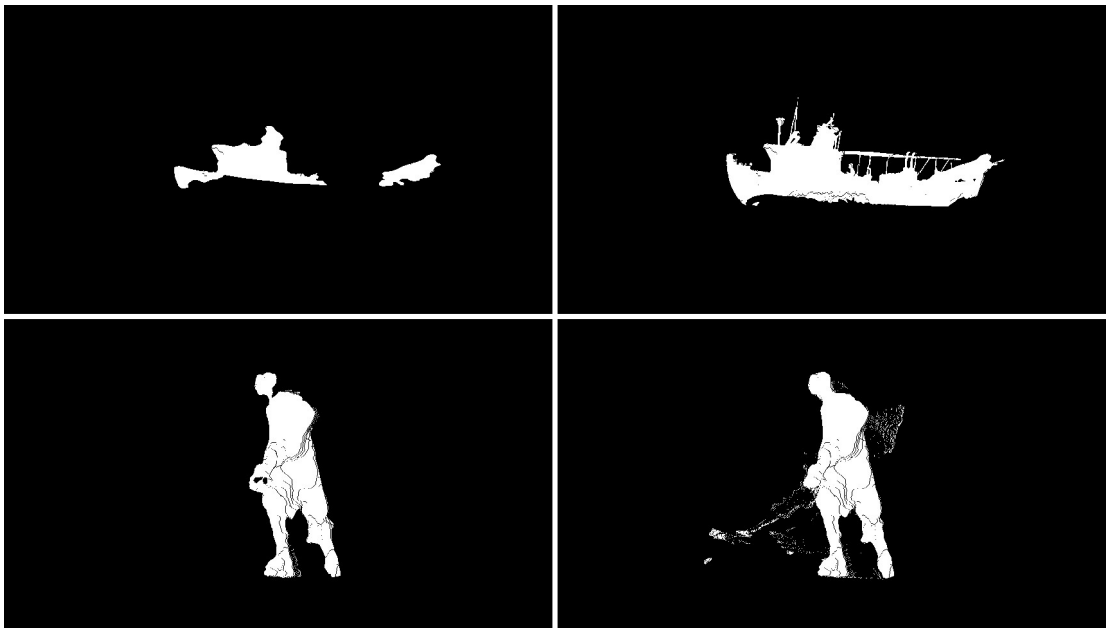


Figure 4.19. Updated model for video object segmentation

With those two heuristics, we propose an update to the model. Figure 4.19 illustrates the new model. The process of updating a mask for the next frame in a sequence is as follows. First, optical flow is computed using the current frame and the next frame. Then, the optical flow is applied to the mask of the current frame to obtain the flow result for the mask and applied to the segmentation result of the current frame to obtain the flow result for the segmentation. Next, a union of the flow result for the mask and segmentation is carried out, producing an initial updated mask. Since we want to avoid excessive noise causing the bounding box to expand, the initial updated mask is cropped to leave out as much noise as possible. The rule to crop the mask is as follows:

- If the **flow result for segmentation can produce a bounding box**, we check for the size of that bounding box. If the size of the box is smaller than the size of the bounding box for the object in the first frame, then the size for the first frame's bounding box is used instead. Otherwise, the bounding box obtained from the flow result for segmentation keeps its size. The size of the box is kept as S. Next, the central position P_c of the bounding box from the segmentation flow is extracted. That position is considered the center of the object of interest. In the initial updated mask, we seek to position P_c and create a box of size S. The box is centered on P_c and covers a region R. Anything in the initial updated mask not in the region R will be considered as noise and discarded.
- If the **flow result for segmentation cannot produce a bounding box** due to bad segmentation results, then we must depend on the mask of the current frame to generate the mask of the next frame. Although the mask is constantly updated when we run the system so that loss and noise are controlled, using the flow result of the mask straight away as an updated mask may cause noise to accumulate. We need a way to remove noise without information from the flow result for segmentation. Using the idea that points in the object form a cluster in the mask, we find the mean of the distribution of positions on the x and y axes of points in the flow result of mask in order to get what would be the central point P_c of the cluster. A box of the same size S as the bounding box obtained from the mask of the first frame is created and centered on the position P_c of the initial updated mask. The box forms a region R on the initial updated mask and any point lying outside R is treated as noise and discarded. The reason the region is chosen to be of size S is because of our following analysis. There are only two cases for the object size in the current frame. If the object size is less than or equal to the object size in the first frame, then restricting the box size to S works by the second heuristic. If the object size of the current frame is larger than the object size in the first frame, then the segmentation result will be able to produce a bounding box in the first place. This conclusion is drawn from observation on the vanilla DHSNet and Guided DHSNet results. As the object grows larger in the sequence, it becomes more prominent and the segmentation result will produce a bounding box.

Applying this model produces more consistent mask and bounding box updates, as can be seen in Figure 4.20.



(a) (b)

Figure 4.20. Comparison between updated masks generated by (a) the segmentation result (b) the union of the segmentation result and the mask

4.3.2.4. Guided DHSNet with object re-tracking using YOLO

The current model for segmentation cannot take care of the case when the object is lost and then reappears back to the scene. We propose using YOLO to solve the task with the following heuristics:

- Object for the current frame is considered lost when a bounding box cannot be generated from the current mask.
- Lost object in a scene does not change drastically comparing to the point right before its disappearance, e.g. a person riding a bicycle as an object of interest when reappearing is still a person riding a bicycle, not just a person alone.

Whenever the system lose track of the object, the process described in Figure 4.16 temporarily stops and the system switches to reappearance identification. Our proposal for this task is a simple one.

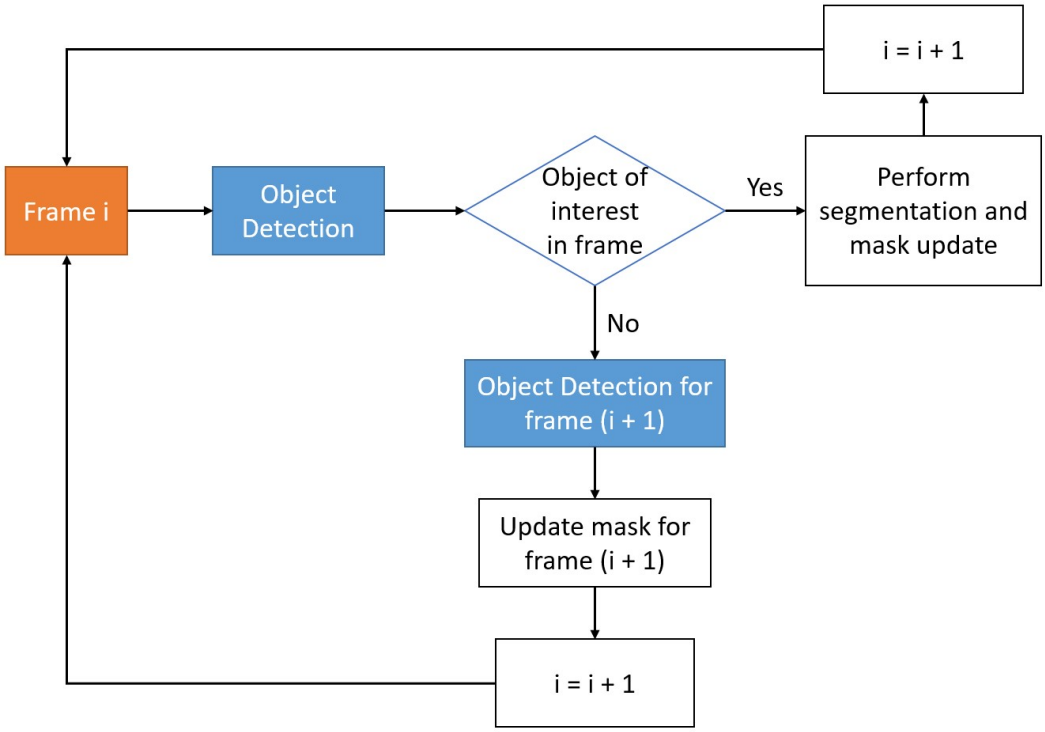


Figure 4.21. Disappearance detection and reappearance tracking

Illustrated in Figure 4.21, for each processing frame in the sequence, YOLO is used to detect all the objects within the bounding box extracted from the mask. The detection returns a set of classes, which will be compared to the set of classes in the bounding box of the first frame. If everything matches, the segmentation is performed as normal. Otherwise, the object is considered lost, an empty result is returned as the segmentation, and a redetection is done. Redetecting is done on every frame after the current frame onwards until the object is found. When it is considered that the tracking is lost, the next frame will go through object detection by YOLO. The result will be a labelling for all objects in the frame. We then traverse through the labels to see if everything in the set of classes of the first frame's bounding box can be found in the frame. If they can be, then the regions in the frame having those classes are marked as potential candidate for the object of interest. Those regions are treated as the mask of the next frame, with which the bounding box will be extracted for the next frame to do segmentation. If the object detection for the next frame does not return the sufficient classes, then we determine that the object of interest has not

reappeared. The mask will be updated as empty, and in the next frame's processing, no available bounding box from the mask means the redetection process will be done instead of segmentation.

To ensure better generalization and to avoid cases when labeling is ambiguous, we group concepts of similar appearance into sets in which everything is treated as equivalent. For example, "bicycle" and "motorcycle" are in the same group. The result when applying this re-tracking method helps the model perform better. Quantitative results are shown in Table 4.3, in which we can see the method performs better and closer to the ideal result.

	Ideal Guided DHSNet	Without object re-identification	With object re-identification
$\mathcal{J}$ **mean** ↑	0.756	0.688	0.719
$\mathcal{J}$ **recall** ↑	0.902	0.802	0.842
$\mathcal{J}$ **decay** ↓	0.043	0.112	0.085
$\mathcal{F}$ **mean** ↑	0.722	0.649	0.680
$\mathcal{F}$ **recall** ↑	0.870	0.768	0.800
$\mathcal{F}$ **decay** ↓	0.052	0.119	0.091
$\mathcal{T}$ **(GT 0.095)** ↓	0.348	0.379	0.382

Table 4.3. Quantitative results before and after applying re-identification comparing to the ideal case for the *val* set of DAVIS 2016

4.4. Conclusion

In this chapter, two sets of experiments are conducted. The first set of experiments deals with studying the potential of using salient object detection for video object segmentation. The second set of experiments revolves around finding a good way to track the object of interest in a scene, how to detect disappearance and reappearance of objects, and gradual enhancements to the proposed model.

Experiments also reveal the performance of the proposed method, with quantitative results indicating that the method is comparable to the top five state-of-the-art methods for video object segmentation on the DAVIS 2016 dataset as of July 2017.

Chapter 5
Conclusion

✍ *Chapter 5 presents the results of this thesis, including what we have learned and achieved through the experiments. The chapter closes with our proposal for future work.*

5.1. Results

Over the course of doing this thesis, we have spent a good amount of time studying Machine Learning and Deep Learning, including Neural Network and Convolutional Neural Network, to acquire essential knowledge. From the history of Neural Networks to techniques applied to training and testing them, we have learned and come to understand a wide range concepts.

Technical detail and meaning of each layer in Convolutional Neural Network, namely Convolutional Layer, Non-linear Gating, Pooling Layer, Dropout Layer, and Loss Layer, are also learned through the process. Understanding them is important both for understanding state-of-the-art network structures and for building and optimizing Convolutional Neural Networks.

We have proposed a method for doing video object segmentation using visual saliency, with optical flow to help in mask propagation and object detection to detect object reappearance. We conduct many experiments to study the feasibility of our method and to find ways to improve the current results. The ideal result obtained by incorporating information from the groundtruth encourages us to create a good model. The experimental results are shown in Table 5.1 and Table 5.2, where it can be seen that our final model achieve results comparable to OFL, the fourth ranked method on the DAVIS 2016 *trainval* set as of July 2017. On the *val* set of DAVIS 2016, our method, both with and withour object re-identification, is comparable to the third-ranked method VPN while the Ideal Guided DHSNet outperforms the third place.

To demonstrate the potential of our proposed system and to provide an attempt into provide a new way to create video mattes, we have proposed an application that employs our proposed method for segmentation.

In conclusion, our thesis lays the foundation for further research into applying semantic segmentation for smart object highlighting in interactive systems.

	$\mathcal{J}$ mean	$\mathcal{J}$ recall	$\mathcal{J}$ decay	$\mathcal{F}$ mean	$\mathcal{F}$ recall	$\mathcal{F}$ decay	$\mathcal{T}$ (GT 0.095)	Official Rank
OSVOS	-	-	-	-	-	-	-	-
MSK	**0.803**	**0.935**	0.089	**0.758**	**0.882**	0.095	**0.189**	1
Ideal	0.756	0.902	**0.043**	0.722	0.870	**0.052**	0.348	-
VPN	0.750	0.901	0.093	0.724	0.842	0.136	0.300	2
Model2	0.719	0.842	0.085	0.680	0.800	0.091	0.382	-
OFL	0.711	0.800	0.227	0.679	0.780	0.240	0.224	3
Model1	0.688	0.802	0.112	0.649	0.768	0.119	0.379	-
BVS	0.665	0.764	0.260	0.656	0.774	0.236	0.317	4

Table 5.1. Comparison to other methods on the DAVIS 2016 *trainval* set. Official ranks are as of July 2017

	$\mathcal{J}$ mean	$\mathcal{J}$ recall	$\mathcal{J}$ decay	$\mathcal{F}$ mean	$\mathcal{F}$ recall	$\mathcal{F}$ decay	$\mathcal{T}$ (GT 0.088)	Official Rank
OSVOS	**0.798**	**0.936**	0.149	**0.806**	**0.926**	0.150	0.378	1
MSK	0.797	0.931	0.089	0.754	0.871	0.090	**0.218**	2
Ideal	0.760	0.900	**0.050**	0.722	0.860	**0.050**	0.343	-
Model2	0.719	0.854	0.081	0.678	0.788	0.099	0.376	-
VPN	0.702	0.823	0.124	0.655	0.690	0.144	0.324	3
Model1	0.697	0.825	0.079	0.660	0.765	0.084	0.379	-
OFL	0.680	0.756	0.264	0.634	0.704	0.272	0.222	4
BVS	0.600	0.669	0.289	0.588	0.679	0.213	0.347	5

Table 5.2. Comparison to other methods on the DAVIS 2016 *val* set. Official ranks are as of July 2017

Note that in Table 5.1 and Table 5.2, *Ideal* refers to the Guided DHSNet run with ideal tracking, *Model1* refers to our method without object re-identification, and *Model2* refers to our method with object re-identification with YOLO.

5.2. Future Work

The results for the authors' experiments demonstrates some limitations to the method. There are two main problems with the proposed method:

- For sequences that subjects to occlusion, the method cannot return consistent results.
- Using YOLO for disappearance and reappearance tracking is limited in that the object of interest must be consistently labeled as the same class throughout the sequence.

More experiments need to be conducted and more dataset must be used to help us address these problems, as well as improve the current results. Furthermore, experiments and testing are done on the same dataset, testing on different datasets may provide new insights.

REFERENCES

[1] L. G. Shapiro and G. C. Stockman, Computer Vision, Pearson, 2001.

[2] "MIT Scene Parsing Benchmark," MIT CSAIL Computer Vision Group 2016, [Online]. Available: http://sceneparsing.csail.mit.edu/. [Accessed 16 7 2017].

[3] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Conference on Computer Vision and Pattern Recognition 2015, CVPR 2015*, Boston, 2015.

[4] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid Scene Parsing Network," in *Conference on Computer Vision and Pattern Recognition 2017, CVPR 2017*, Honolulu, 2016.

[5] "DAVIS: Densely Annotated VIdeo Segmentation," [Online]. Available: http://davischallenge.org/index.html. [Accessed 16 7 2017].

[6] B. Kayalibay, G. Jensen and P. v. d. Smagt, "CNN-based Segmentation of Medical Imaging Data," 2017.

[7] S. Bana and D. D. Kaur, "Fingerprint Recognition using Image Segmentation," *International Journal of Advanced Engineering Sciences and Technologies,* vol. 5, no. 1, pp. 012-023, 2011.

[8] R. Achanta, S. Hemami, F. Estrada and S. Susstrunk, "Frequency-tuned Salient Region Detection," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009*, Miami Beach, 2009.

[9] A. M. Treisman and G. Gelade, "A Feature-Integration Theory of Attention," *Cognitive Psychology,* vol. 12, pp. 97-136, 1980.

[10] F. Crick, "Function of the thalamic reticular complex: The searchlight," *Neurobiology,* vol. 81, pp. 4586-4590, 1984.

[11] E. Weichselgartner and G. Sperling, "Dynamics of Automatic and Controlled Visual Attention," *Science,* vol. 238, pp. 778-780, 1987.

[12] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention," *Vision Research,* vol. 40, p. 1489–1506, 2000.

[13] V. Navalpakkam and L. Itti, "Modeling the influence of task on attention," *Vision Research,* vol. 45, p. 205–231, 2005.

[14] A. Maeder, "The evolution of massive stars," *The evolution of massive stars.,* vol. 189, pp. 313 - 322, 1997.

[15] L. Itti, "Automatic Foveation for Video Compression Using a Neurobiological Model of Visual Attention," *IEEE Transactions on Image Processing,* vol. 13, pp. 1304-1318, 2004.

[16] W. Wang, J. Shen and F. Porikli, "Saliency-Aware Geodesic Video Object Segmentation," in *Conference on Computer Vision and Pattern Recognition 2015, CVPR15*, Honolulu, 2015.

[17] L. Shi, J. Wang, L. Xu, H. Lu and C. Xu, "Context saliency based image summarization," in *2009 IEEE International Conference on Multimedia & Expo (ICME 2009)*, New York, 2009.

[18] N. Liu and J. Han, "DHSNet: Deep hierarchical saliency network for salient object detection," in *Conference on Computer Vision and Pattern Recognition 2016, CVPR 2016*, Las Vegas, 2016.

[19] "DAVIS Challenge on Video Object Segmentation 2017," [Online]. Available: http://davischallenge.org/challenge2017/index.html. [Accessed 20 7 2017].

[20] H. Pan, B. Wang and H. Jiang, "Deep Learning for Object Saliency Detection and Image Segmentation," in *Computing Research Repository, CoRR* , 2015.

[21] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross and A. Sorkine-Hornung, "A Benchmark Dataset and Evaluation Methodology for Video

Object Segmentation," in *Conference on Computer Vision and Pattern Recognition 2016, CVPR 2016*, Honolulu, 2016.

[22] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[23] F. Amato, A. López, E. M. Peña-Méndez, P. Vaňhara, A. Hampl and J. Havel, "Artificial neural networks in medical diagnosis," *Journal of Applied Biomedicine,* vol. 11, p. 47–58, 2013.

[24] A. Maithili, V. R. Kumari and S. Rajamanickam, "Neural networks towards medical," *International Journal of Modern Engineering Research (IJMER),* vol. 1, no. 1, pp. 57-64, 2011.

[25] C. I. Patel, R. Patel and P. Patel, "Handwritten character recognition using neural network," *International Journal of Scientific & Engineering Research,* vol. 2, no. 5, pp. 1-6, 2011.

[26] Y. Pao, Adaptive pattern recognition and neural networks, Reading, MA (US); Addison-Wesley Publishing Co., Inc., 1989.

[27] A. Karpathy, "CS231n Convolutional Neural Networks for Visual Recognition," [Online]. Available: http://cs231n.github.io/. [Accessed 15 7 2017].

[28] "Caffe," [Online]. Available: http://caffe.berkeleyvision.org/. [Accessed 15 7 2017].

[29] F. Raudies, "Optic flow," [Online]. Available: http://www.scholarpedia.org/article/Optic_flow. [Accessed 11 8 2017].

[30] P. Weinzaepfel, J. Revaud, Z. Harchaoui and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," *Proceedings of the IEEE International Conference on Computer Vision,* pp. 1385-1392, 2013.

[31] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE transactions on pattern analysis and machine intelligence,* vol. 33, no. 3, pp. 500-513, 2011.

[32] J. Revaud, P. Weinzaepfel, Z. Harchaoui and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1164-1172.

[33] J. Redmon, S. K. Divvala, R. B. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Computing Research Repository - CoRR,* 2015.

[34] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers and L. Van Gool, "One-shot video object segmentation," 2016.

[35] Y.-H. Tsai, M.-H. Yang and M. J. Black, "Video segmentation via object flow," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 3899-3908, 2016.

[36] A. Khoreva, F. Perazzi, R. Benenson, B. Schiele and A. Sorkine-Hornung, "Learning video object segmentation from static images," 2016.

[37] D. J. Butler, J. Wulff, G. B. Stanley and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, Springer-Verlag, 2012, pp. 611-625.

[38] "Benchmark Video Object Segmentation on DAVIS - Quantitative state-of-the-art comparison (Single Object)," [Online]. Available: http://davischallenge.org/soa_compare.html. [Accessed 20 7 2017].

APPENDIX - PUBLICATION

Trung-Nghia Le, Khac-Tuan Nguyen, Manh-Hung Nguyen-Phan, That-Vinh Ton, **Toan-Anh Nguyen, Xuan-Son Trinh**, Quang-Hieu Dinh, Vinh-Tiep Nguyen, Anh-Duc Duong, Akihiro Sugimoto, Tam V. Nguyen, and Minh-Triet Tran, "Instance Re-Identification Flow for Video Object Segmentation", DAVIS Challenge on Video Object Segmentation 2017, CPVR Workshop 2017, Honolulu, Hawaii, 21-27/06/2017.