N° d"ordre :.....
N° de série :.....

People's Democratic Republic of Algeria
Ministry of Higher Education and Research Scientist

UNIVERSITY ECHAHID HAMMA LAKHDAR EL OUED
FACULTY OF EXACT SCIENCES
Computer Science department

END-OF-STUDY DISSERTATION
Presented for the Diploma of

ACADEMIC MASTER

Field: Mathematics and Computer Science
Domain: Computer science
Specialty: Distributed Systems and Artistic Intelligence

Theme

Salient Object Detection Technique based on Color Divergence and deep learning semantic segmentation

Presented by:
- GUEDA Zineb

Under the supervision of:
- Dr LAOUID Abdelkader
- GUIA Sana Sahar

Sustained on 22-09-2021

Board of Examiners:

Chairman/ President: ABBAS Messaoud	University of El Oued
Supervisor: LAOUID Abdelkader	University of El Oued
Examiner: NDEOUI Abdelhamid	University of El Oued

Academic Year: 2020/2021

THANKS

First and foremost, I thank God Almighty Who guided me to his book and by which I was able to complete this humble work.

Then I would like to express my thanks and respect to

Dr Laouid Abdelkader ,

administrators, who has attached this work to me,

I also extend my sincere thanks and gratitude from the bottom of my heart to

Guia Sana Sahar ,

who helped me, directed me, and treated me like a dear sister and colleague.m

I wish them happiness and success in various fields,

I sincerely thank all the members of the jury who accepted the verdict on my humble work,

and all who helped me from near or far:

our teachers, our families, our friends, our colleagues all named.

Dedication

To my parents who gave birth to me, may God have mercy on them

To my parents who raised me, may God have mercy on them

To my virtuous sister, her husband and children: Nawal, Hana, Manar, Ali, Abd al-Rahim

To all my family

To all my friends

to everyone who loves me and I love them in God.

Abstract

Salient object detection (SOD) tries to simulate the visual attention mechanism to highlight objects that attract the attention from each given image. The output of the SOD is a saliency map, in which each pixel is labeled by a real value within the range of [0,1] to indicate its probability of belonging to a salient object. Higher value represents higher saliency.

Proposed method includes image segmentation using deep learning semantic segmentation, next computes the first saliency map using the Frequency-tuned Salient Region method which detect the most visually salient region in the segmented image. The last step generates the final saliency map using the Color Divergence Technique.

Keywords

Object Detection, visual attention, color divergence, semantic segmentation.

ملخص

إن محاولة اكتشاف الكائن البارز (SOD) محاكاة آلية للانتباه البصري حيث نـقم بتمييز العناصر التي تجذب الانتباه من كل صورة معينة. فإخراج SOD هي خريطة البروز ، حيث يتم تمييز كل بكسل بقيمة حقيقية ضمن النطاق [0،1] للإشارة إلى احتمال الانتماء إلى كائن بارز. فأعلى قيمة هي أعلى بروز.

تتضمن الطريقة المقترحة تجزئة الصورة باستخدام دلالات التعلم العميق للتجزئة ، ثم يحسب أول خريطة بروز باستخدام تحفيز ضبط التردد بطريقة المنطقة التي تكتشف المنطقة الأكثر بروزًا من الناحية المرئية في الصورة المجزأة. تنشئ الخطوة الأخيرة خريطة النتوء النهائية باستخدام تقنية تباين الألوان.

الكلمات المفتاحية:

إكتشاف الكائن، الإنتباه البصري ، تباين الألوان ، التجزئة الدلالية.

CONTENTS

LIST OF FIGURES

GENERAL INTRODUCTION

Salient Object Detection (SOD) has become a hot topic in computer vision research., mainly because it helps to find the objects or regions that efficiently represent a scene, a useful step in complex vision problems such as scene understanding. The human vision system has an effective ability to easily recognize regions of interest from complex scenes, even if the focused regions have similar colors or shapes as the background. Salient object detection (SOD) aims to detect the salient object that attracts the most the visual attention. Given space limitations, this thesis cannot fully explore all of the aforementioned research directions. Instead, we only focus on salient object detection, a research area that has greatly developed in the past years, and in particular since 2007 [20]. 1.1 This thesis consists of four chapters:

In **the first chapter** , we introduce some basic concepts of salient object detection, then we the main traditional method of SOD closest to our approach , and finally we review state-of-the-art deep learning in this field.

In **the second chapter** , We will talk about how to discover the salient regions with the latest semantic image segmentation techniques that enable us to approximate the understanding of image segmentation simulated by human sight.

In **the third chapter** ,We will talk about using color divergence technique to create the final saliency map after going through the steps of semantic segmentation of the image and visually revealing it.

In **final chapter**,We describe and detail in this last chapter the application part of what we mentioned in the aforementioned axes, and the various necessary tools used to work on it. We also ended up with a set of predictions and experimental results that showed a good detection of salient objects in the images.

CHAPTER 1

BASIC KNOWLEDGE ON SALIENT OBJECT DETECTION

1.1 Introduction

Humans are able to detect visually distinctive, so called salient, scene regions effortlessly and rapidly in a pre-attentive stage. These filtered regions are then perceived and processed in finer detail for the extraction of richer high-level information, in an attentive stage.

In this chapter, we briefly present a subset of some basic concepts of salient objects detection and a review of traditional techniques that was most frequently used in salient object detection methods, and finally state of the art of methods using deep learning for salient object detection.

1.2 Basic concepts

SALIENT object detection (SOD) aims at highlighting salient object regions in images. In the following some basic concepts about the SOD.

1.2.1 Research directions in object detection

Object detection, including objectness detection (OD), salient object detection (SOD), and category-specific object detection (COD), is one of the most fundamental yet challenging problems in the computer vision community. The research works in object detection can be roughly categorized into three directions: OD, SOD, and COD [1].

objectness detection (OD)

OD aims at detecting all possible objects appearing in each given image, regardless of the specific object category. Usually, OD algorithms output thousands of object proposals or hypotheses as shown in Figure 1.1, which can benefit a wide range of computer vision tasks like weakly supervised learning and object tracking.

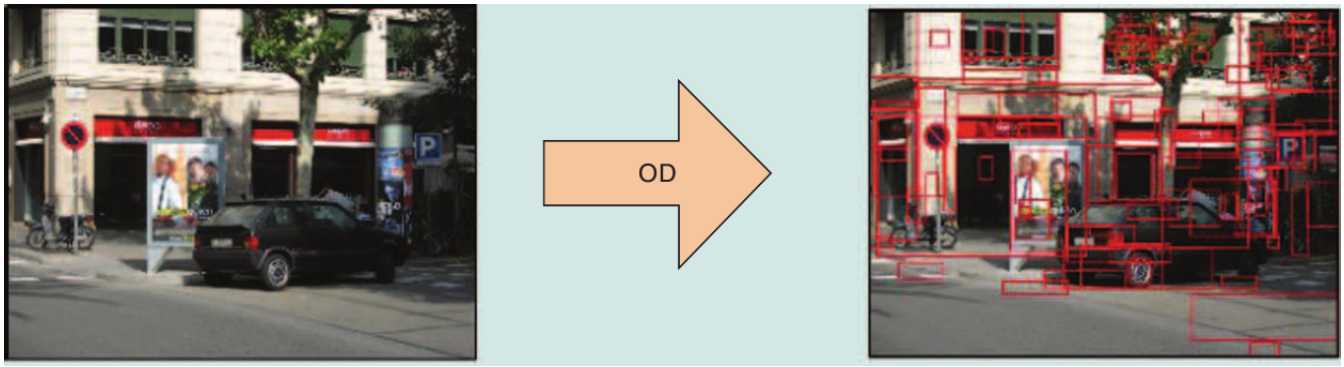

Figure 1.1: Objectness detection (OD) one of research direction of object detection.

Salient object detection (SOD)

SOD, is another direction in object detection, which aims at mimicking the visual attention mechanism to highlight objects that draw our attention from each given image. This is inspired by the human visual attention system, which can guide humans to pay special attention to a few informative image regions that are naturally distinct. Usually, SOD algorithms output a limited number of object regions based on the obtained saliency maps, as shown in Figure 1.2

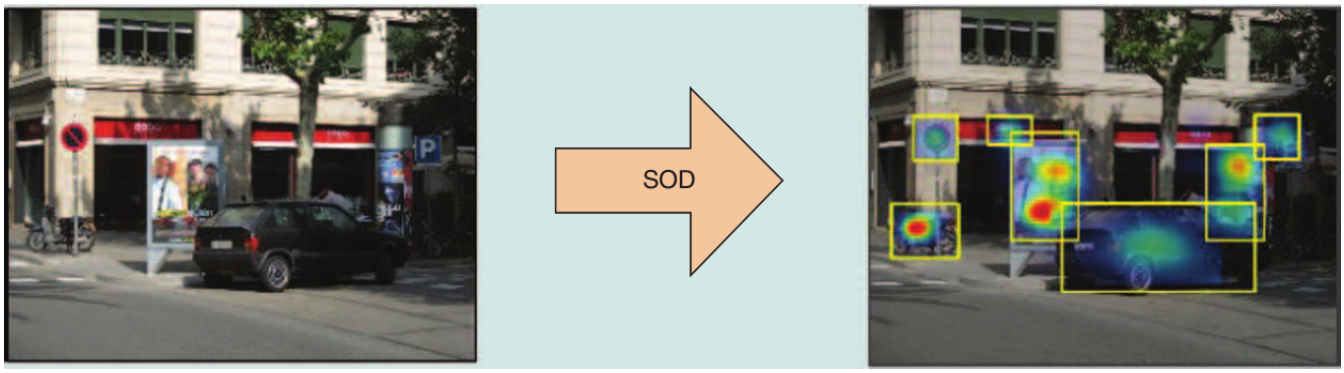

Figure 1.2: Salient object detection (SOD) as object detection research direction

Category-specific object detection (COD)

The third direction of object detection is COD. Different from OD, COD aims at detecting multiple predefined object categories from each given image. It needs to not only identify the image regions that may contain the objects of interest but also recognize the specific object category of each detected image region. Usually, COD is converted to a multiclass classification problem, where discriminative classification functions are trained to separate the extracted image regions in the corresponding feature domain. As shown in Figure 1.3,COD approaches usually output multiple image regions assigned with the identified object category. COD can be applied to computer vision tasks like scene parsing and human action recognition.

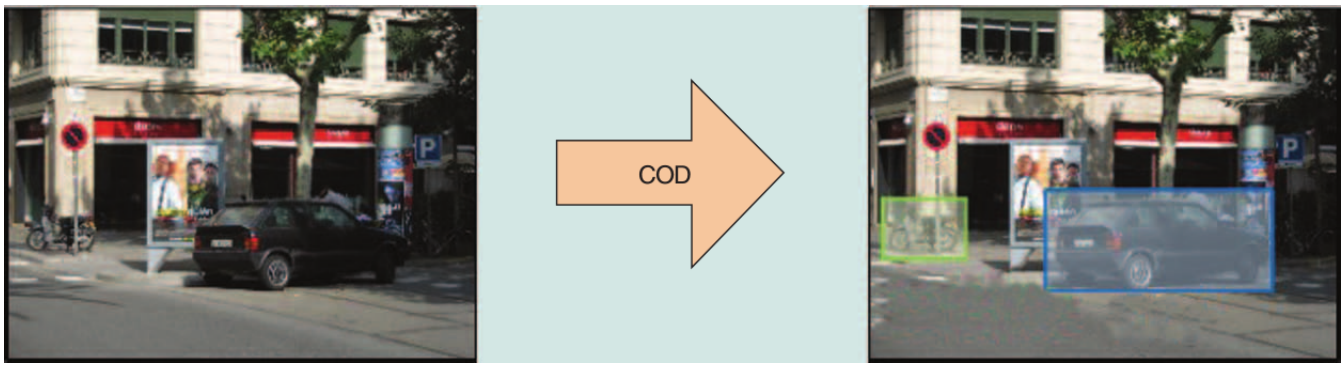

Figure 1.3: Category-specific object detection (COD)

1.2.2 Overview

Salient object detection or salient object segmentation is commonly interpreted in computer vision as a process that includes two stages:

✓ detecting the most salient object

✓ segmenting the accurate region of that object.

An image can contain zero or many different salient objects, and the purpose of SOD is to separate salient objects from the background. "Perfect" detection, known as Ground Truth (GT), will show salient object(s) in white and everything else in black. Examples are shown in Figure 1.4. The output of the SOD is a saliency map, in which each pixel is labeled by a real value within the range of [0,1] to indicate its probability of belonging to a salient object. Higher value represents higher saliency. In general, it is agreed that for good saliency detection a model should meet at least the following three criteria:

✓ good detection: the probability of missing real salient regions and falsely marking the background as a salient region should be low.

✓ high resolution: saliency maps should have high or full resolution to accurately locate salient objects and retain original image information.

✓ computational efficiency: as front-ends to other complex processes, these models should detect salient regions quickly.

Figure 1.4: Example images and their corresponding Ground truth.

1.2.3 Problem Formulation

Common formulation of the image-based SOD

A common formulation of the image-based SOD problem can be presented as follow [2]: Given an input image $I \in R\, W \times H \times 3$ of size $W \times H$, an SOD algorithm f maps the I to a binary salient object mask $S = f(I) \in \{0,1\}W \times H$. For learning-based SOD, the model f is learned through a set of training samples. Given a set of N static images $I = \{I_n \in R\ W \times H \times 3\}\ N\ n = 1$ and the corresponding binary ground-truth annotations $G = \{G_n \in \{0,1\}\ W \times H\}\ N\ n = 1$, the goal of learning is to find $finF$ that minimizes the prediction error, $n = 1\ m(S_n, G_n)$, where $m \in M$ is some distance measure (as will be reviewed in this section), $Sn = f(I_n)$, and F is the set of potential mapping functions. Deep SOD algorithms typically model f through modern deep learning techniques. The ground-truths G can be collected by different methodologies, i.e., direct human-annotation or eye-fixation-guided labeling, and may have different formats, i.e., pixel-wise or bounding-box level.

EVALUATION METRICS

Specifically, given a saliency map S and the corresponding ground-truth saliency mask G , we first normalize S to [0], [1]. There are several ways to measure the agreement between model predictions and human annotations. In the following four universally-agreed and popularly adopted measures for SOD model evaluation [2].

• Precision-Recall (PR) is calculated based on the binarized saliency mask and the ground-truth:

$$Precision = \frac{TP}{TP + FP} \tag{1.1}$$

$$Recall = \frac{TP}{TP + FN}$$

where TP, TN, FP, FN denote true-positive, true-negative, false-positive, and false-negative, respectively. To get the binary mask, a set of thresholds ranging from 0 to 255 is applied, each of which produces a pair of Precision/Recall value to form a PR curve for describing model performance.

• F-measure comprehensively considers both Precision and Recall by computing the weighted harmonic mean:

$$F_\beta = \frac{(1 + \beta^2)\ Precision \times Recall}{\beta^2\ Precision + Recall} \tag{1.2}$$

β^2 is empirically set to 0.3 to emphasize more on precision. Instead of reporting the whole F-measure plot, some methods directly use the maximal F β values from the plot, and some others use an adaptive threshold , i.e., twice the mean value of the predicted saliency map, to generate the binary saliency map and report the corresponding mean F-measure value.

• Mean Absolute Error (MAE) . Despite their popularity, the above two metrics fail to take into consideration the true negative pixels. MAE is used to remedy this problem by measuring the

average pixel-wise absolute error between normalized map $S \in [0,1]W \times H$ and ground-truth mask $G \in \{0,1\}W \times H$:

$$MAE = \frac{1}{W \times H} \sum_{w}^{i=1} |G(i,j) - S(i,j)| \tag{1.3}$$

• Weighted F *beta* measure (Fbw) intuitively generalizes F-measure by alternating the way to calculate the Precision and Recall. It extends the four basic quantities TP, TN, FP and FN to real values, and assigns different weights (ω) to different errors at different locations considering the neighborhood information, defined as:

$$F_{\beta}^{\omega} = \frac{(1+\beta^2)\ Precision^{\omega} \times Recall^{\omega}}{\beta^2\ Precision^{\omega} + Recall^{\omega}} \tag{1.4}$$

1.2.4 Challenges

There are some challenges in object detection. Sometimes background of image is complex. There are similarities between several types of object like human and doll. It recognize sometimes doll as human. So, it not only founded on single feature like shape, color, size etc. Same species object will be changed because of size, color, viewpoint, shape etc.

Similar to OD, SOD has great challenges :

• As different objects, either within the same object category or from different object categories, can have dramatic appearance variation, due to

✓ their internal intrinsic characteristics (e.g., living creatures such as cats generally have more deformable appearances than man-made objects such as vehicles)

✓ or external capturing conditions such as viewing distances or angles (e.g., deformable objects may appear somewhat rigid at a distance, and even rigid objects may exhibit variations under different viewing angles)

• SOD faces the challenge of how object detection frameworks to incorporate the extracted features to effectively associate the desired visual stimulus (usually at a with the carefully designed classifiers (e.g., random forest and semantic level) and the corresponding regions in visual scenes.

1.2.5 Applications of salient object detection

Some topics that are closely or remotely related to visual saliency include: salient object detection, fixation prediction, object importance, memorability, scene clutter, video interestingness, image quality assessment, scene typicality, aesthetics, and scene attributes [3]. The value of salient object detection models lies on their applications in many areas of computer vision, graphics, and robotics. Salient object detection models have been utilized for several applications such as object detection and recognition, image and video compression, video summarization, photo collage/media re-targeting/cropping/thumb-nailing, image quality assessment, image segmentation, content-based

image retrieval and image collection browsing, image editing and manipulating, visual tracking, object discovery, and human-robot interaction.

1.3 Traditional methods for salient object detection

Compared with other computer vision tasks, the traditional methods of SOD is relatively short and can be traced back to the pioneer works in [4] and [5]. Most of non-deep learning SOD models are based on low-level features and rely on certain heuristics (e.g., color contrast, background prior). However, they all generally have a few stages in common : image segmentation, prior calculation, prior combination, as described in the following.

1.3.1 Image segmentation

Image segmentation, the first and most important step, separates the image into manageable pieces. If large numbers of salient and non-salient pixels are considered together, it is impossible to accurately detect saliency. Thus the choices of region type and methodology are critical. Much work performed in the last few decades regarding saliency was based at the pixel level. However, a pixel is not a very accurate portrayal of an image or even a local object. A pixel offers no context or indication to what object it belongs. As such, modern SOD methods work on groups of pixels. State-of-the-art segmentation methods generally fall into two categories: superpixel based and cluster-based segmentation. Superpixels have the advantage of being continuous; clusters, on the other hand, are not necessarily grouped spatially. However, an object that appears in multiple areas of an image can be grouped into one cluster. The choice of segmentation determines what priors can be used to calculate saliency.

1.3.2 Prior Calculation

Following segmentation, calculations are performed on characteristics (called priors) of the regions. The most common prior is color contrast. Some color contrast calculations compare regions globally to all other regions. Other methods use color contrast related to neighboring regions. Another common prior has to do with location, sometimes called a spatial or center prior. It is based on the idea that the HVS focuses on objects closer to the center of an image. Most methods use some form of center bias, even when it is not directly a prior calculation. A more recent prior is color distribution. It is defined as the spatial variation of the color in a region. A salient region will typically have compact color distribution.

1.3.3 Prior Combination

Once the priors have been calculated, they must be combined to form a final saliency map. Priors can be combined using a number of methods including: average, weighted average, and multiplication. The average method calculates the average of the scores obtained from the prior calculation. The weighted average method assigns different weights to each prior, and then computes the average of the weighted scores. The multiplication method multiplies the scores obtained from the prior calculation. Multiplication is the most common approach; however, it can be linear or nonlinear.

1.4 DEEP LEARNING BASED SALIENT OBJECT DETECTION (SOD) MODELS

1.4.1 Preliminary knowledge

In recent years, there has been rapid development in the research area of deep learning, including its popularization in computer vision. In this section, we briefly introduce one of the advanced deep-learning techniques that has been widely used in the object detection task, i.e., the CNN [1]. The CNN is one of the most well-known and widely used deep-learning architectures inspired by the natural visual perception mechanism of living creatures, which was first proposed in 1980 by Fukushima and then improved by LeCun. CNNs are designed to process data that come in the form of multiple arrays, for example, a color image composed of three two-dimensional arrays that contain pixel intensities in the three color channels. There are four key ideas behind CNNs that take advantage of the properties of natural signals: local connections, shared weights, pooling, and the use of many layers. As shown in Figure 4.16, the architecture of a typical CNN

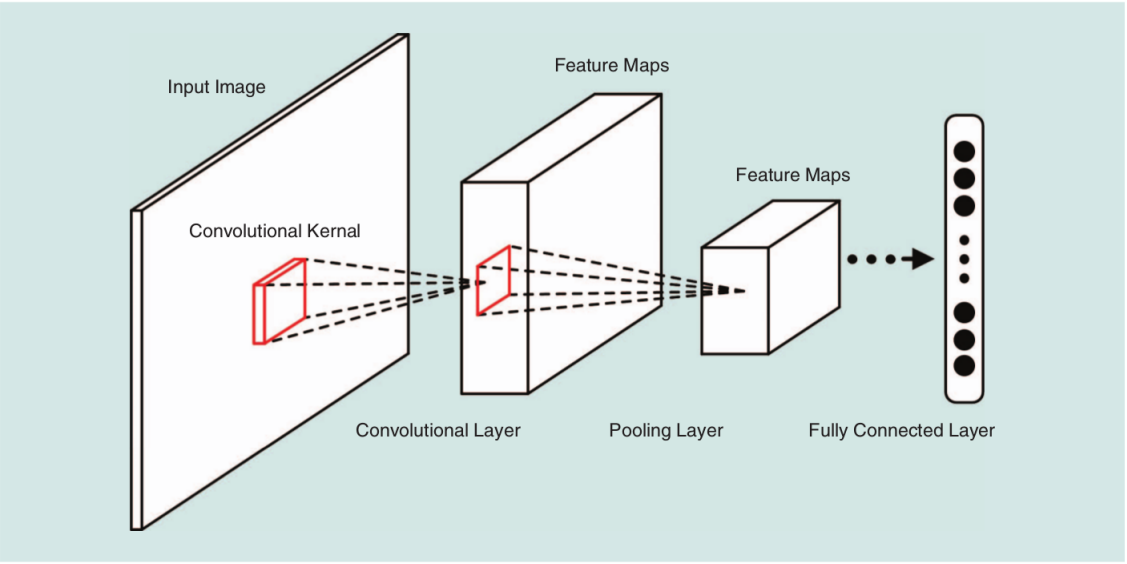

Figure 1.5: Example of architecture of CNN model.

model is structured as a series of layers as follows:

Convolutional layers

Convolutional layers are the most important for feature extraction. The first several layers usually capture low-level features (like edges, lines, and corners) while the deeper layers are able to learn high-level features (like structures, objects, and shapes) by combining low-level ones. Each unit in a convolutional layer is connected to a local patch in the feature maps of the previous layer through a set of kernels called a filter bank. The result of this local weighted sum is then passed through a nonlinearity operation such as a rectified linear unit (ReLU). All units in a feature map share the same filter bank. Different feature maps in a convolutional layer use different filter banks.

Pooling layers

Pooling layers aim to reduce the dimension of the representation and create invariance to small shifts and distortions. A pooling layer is usually placed between two convolutional layers. Each feature map of a pooling layer is connected to its corresponding feature map of the previous convolutional layer. A typical pooling unit computes the maximum of a local patch of units in one feature map.

Fully connected layers

Fully connected layers are typically used as the last few layers of the network for better summarizing the information conveyed by lower-level layers in view of the final decision. As a fully connected layer occupies most of the parameters, overfitting can easily happen. To prevent this, the dropout method is usually employed.

1.4.2 State of the art deep learning based SOD

Nowadays, object detection including salient object detection is one of the most fundamental yet challenging problems in the computer vision community [1]. The set of saliency detection methods are extremely rich. With the great success of deep learning technologies in computer vision, several deep learning based SOD methods have emerged since 2015 [2] and a series of algorithms have been published to learn the saliency representations from large amounts of data [3]. The authors of this paper focus on some representative work having great practical importance to improve the performance of SOD.

Specifically, Li et al. [6] introduce an end-to-end deep contrast network with two complementary components: a fully convolutional stream and a segment-wise spatial pooling. In Kuen et al. [7], a recurrent attention and convolutional-deconvolutional networks are used to perform saliency progressive refinement from exploiting spatial transformer and recurrent network units. Kruthiventi et al. in [8]

consider both fixation prediction and segmentation of salient objects in a unified network. However, these models still fail in several cases including scenes with transparent objects, low contrast between foreground and background, and complex backgrounds.

In order to adress these problems, Zhang et al. [9] detect salient objects by using saliency cues at different levels through fully convolutional neural networks and multi-level fusion mechanisms. Hou et al.[10] suggest adding a series of short connections to the Holistically-Nested Edge Detector (HED) architecture [11] to combine different side outputs from deeper layers to shallower ones to better detect the salient objects/regions. Pingping Zhang et al. [12] aggregate a multi-level features framework at multiple resolutions. A boundary refinement is used at each aggregated feature before using the fused prediction as a final saliency map.

However, salient results in these methods attend to contain many non-salient objects and suffer from the loss of certain details of the salient object when directly merging several level features. Inspired by these observations, Xiaowei Hu et al. [13] introduced the Fully Convolutional neural Network (FCN) with recurrently aggregated deep features for salient object detection. They use the built-in side features to refine the features of each layer of the FCN and such processes are repeated to gradually produce finer salience predictions. Another improvement proposed by Xin Li et al. in [14], comes from creating a two-branch Contour-to-Saliency Network (C2S-Net) for salient object detection. With the capability of combining both saliency and contour knowledge, some enhanced results are obtained by inserting an SOD branch to a pre-trained contour detection model.

All these models use context regions holistically to construct contextual features. Even with their robustness, they still need to get the most accurate results. Recently, in [15] Nian et al. formulate a Pixel-wise Contextual Attention Network (PiCANet) within a U-Net architecture [16] to attend global and local context regions for each pixel. The work of [17] expands the potentials of pooling in convolutional neural networks based on the feature pyramid networks (FPNs)[18]. Yuzhu Ji et al. in [19] solve the problem of salient object detection by fusing multiple saliency cues and objectness based on a graph model bottom-up method. The work of [20] presents a simplified end-to-end method based on a saliency regression network which runs as a compact pipeline to directly output a full resolution saliency map (image-to-image prediction).

1.5 Conclusion

Despite the good performance obtained by deep learning approaches, there is still a great room for improvement. In the next chapter, we present semantic segmentation : the deep learning step of the proposed salient object detection methodology.

CHAPTER 2

DEEP LEARNING SEMANTIC SEGMENTATION

2.1 Introduction

Nowadays, semantic segmentation is one of the major problems in the field of computer vision. Looking at the big picture, semantic segmentation is one of the high-level tasks that pave the way for understanding the whole landscape.

2.2 Image Segmentation

The purpose of image analysis is to extract the distinct information contained in the image. The result of this analysis is often called a structural description. Basically, image analysis involves segmentation where we will try to relate a label to each pixel in the image based on

- the information provided (grayscale),
- the spatial distribution at the center of the image,
- the patterns simple (often geometric models) [21].

2.2.1 Definition

Image segmentation partitions an image into regions called segments. Image segmentation creates segments of connected pixels by analyzing some similarity criteria: intensity, color, texture, histogram, features,.... [22].

2.2.2 The Approach

Whenever one tries to take a bird's eye view of the Image Segmentation tasks, one gets to observe a crucial process that happens here – object identification. Any simple to complex application areas, everything is based out of object detection. And detection is made possible because the image segmentation algorithms try to collect similar pixels together and separate out dissimilar pixels. This is done by following two approaches based on the image properties.

Similarity Detection (Region Approach)

This fundamental approach relies on detecting similar pixels in an image – based on a threshold, region growing, region spreading, and region merging. Machine learning algorithms like clustering relies on this approach of similarity detection on an unknown set of features, so does classification, which detects similarity based on a pre-defined (known) set of features.

Discontinuity Detection (Boundary Approach)

This is a stark opposite of similarity detection approach where the algorithm rather searches for discontinuity. Image Segmentation Algorithms like Edge Detection, Point Detection, Line Detection

follows this approach – where edges get detected based on various metrics of discontinuity like intensity etc.

2.3 Semantic Segmentation

2.3.1 Machine learning

Machine Learning is a natural outgrowth of the intersection of Computer Science and Statistics, to create statistical models, used to perform major tasks like predictions and inference. These models are groups of mathematical relationships through the inputs and outputs of a given system. The learning procedure is the process of guessing the models parameters such that the model can achieve the stated task. machine learning is a common in Medical, E-commerce, Finance, Transportation and various sectors.

2.3.2 Deep Learning

Definition

is a machine learning technique that constructs artificial neural networks to mimic the structure and function of the human brain. In practice, deep learning, also known as deep structured learning or hierarchical learning, uses a large number hidden layers -typically more than 6 but often much higher - of nonlinear processing to extract features from data and transform the data into different levels of abstraction (representations) [23].

Deep Learning principle

As an example, assume the input data is a matrix of pixels. The first layer typically abstracts the pixels and recognizes the edges of features in the image. The next layer might build simple features from the edges such as leaves and branches. The next layer could then recognize a tree and so on. The data passing from one layer to the next is considered a transformation, turning the output of one layer into the input for the next. Each layer corresponds with a different level of abstraction and the machine can learn which features of the data to place in which layer/level on its own. Deep learning is differentiated from traditional “shallow learning” because it learns much deeper levels of hierarchical abstraction and representations.

Practical Uses of Deep Learning

✓ **Automatic Speech Recognition:** All major commercial speech recognition systems (think your smart phone assistant) use a deep learning technique with recurrent neural networks currently being the most popular.

✓ **Computer Vision :** Images are used to train the machine to recognize features and now the machines are demonstrating "superhuman" accuracy for image recognition.

✓ **Natural Language Processing**Modern deep learning techniques have led to improvements in translation and language modeling. Google Translate uses deep learning techniques to translate based on the semantics of an entire sentence instead of just memorizing phrase-to-phrase translations.

some standard deep networks

that have made significant contributions to the field of computer vision, as they are often used as the basis of semantic segmentation systems:

✓ **AlexNet:** Toronto's pioneering deep CNN that won the 2012 ImageNet competition with a test accuracy of (84.6%). It consists of 5 convolutional layers, max-pooling ones, ReLUs as non-linearities, 3 fully-convolutional layers, and dropout.

✓ **VGG-16:** This Oxford's model won the 2013 ImageNet competition with (92.7%) accuracy. It uses a stack of convolution layers with small receptive fields in the first layers instead of few layers with big receptive fields.

✓ **GoogLeNet:** This Google's network won the 2014 ImageNet competition with accuracy of 93.3%. It is composed by 22 layers and a newly introduced building block called inception module. The module consists of a Network-in-Network layer, a pooling operation, a large-sized convolution layer, and small-sized convolution layer.

✓ **ResNet:** This Microsoft's model won the 2016 ImageNet competition with 96.4% accuracy. It is well-known due to its depth (152 layers) and the introduction of residual blocks. The residual blocks address the problem of training a really deep architecture by introducing identity skip connections so that layers can copy their inputs to the next layer.

2.3.3 Semantic Segmentation Definition

is a natural step in the progression from coarse to fine inference: The origin could be located at classification, which consists of making a prediction for a whole input.The next step is localization / detection, which provide not only the classes but also additional information regarding the spatial location of those classes. Finally, semantic segmentation achieves fine-grained inference by making dense predictions inferring labels for every pixel, so that each pixel is labeled with the class of its enclosing object ore region[24].

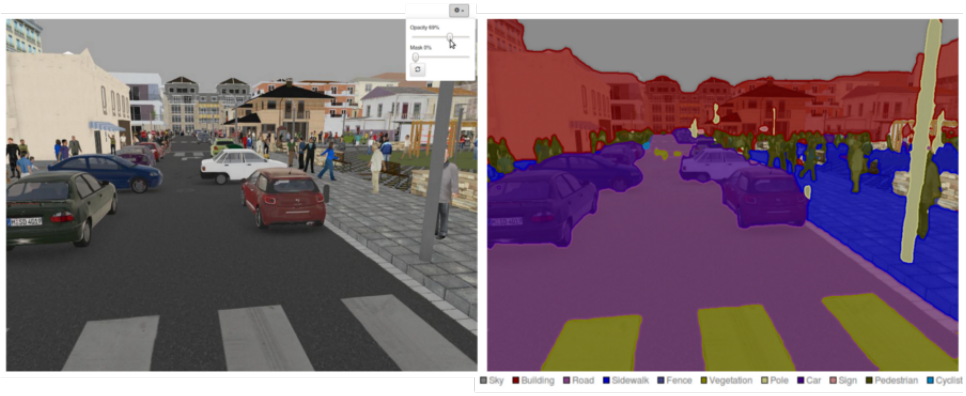

Figure 2.1: An example of semantic segmentation

2.3.4 Semantic Segmentation approaches

A general semantic segmentation architecture can be broadly thought of as an encoder network followed by a decoder network: The encoder is usually is a pre-trained classification network like VGG/ResNet followed by a decoder network. The task of the decoder is to semantically project the discriminative features (lower resolution) learnt by the encoder onto the pixel space (higher resolution) to get a dense classification. Unlike classification where the end result of the very deep network is the only important thing, semantic segmentation not only requires discrimination at pixel level but also a mechanism to project the discriminative features learnt at different stages of the encoder onto the pixel space. Different approaches employ different mechanisms as a part of the decoding mechanism. Let's explore the 4 main approaches:

Region-Based Semantic Segmentation

The region-based methods generally follow the "segmentation x using recognition" pipeline, which first extracts free-form regions from an image and describes them, followed by region-based classification. At test time, the region-based predictions are transformed to pixel predictions, usually by labeling a pixel according to the highest scoring region that contains it.

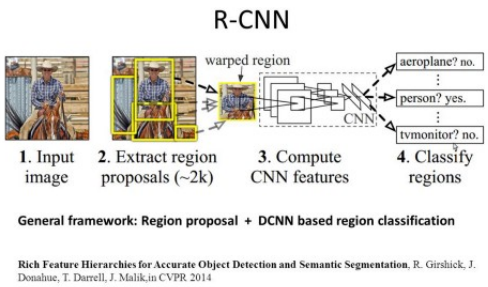

Figure 2.2: R-CNN Architecture

R-CNN (Regions with CNN feature) is one representative work for the region-based methods. It performs the semantic segmentation based on the object detection results. To be specific, R-CNN first utilizes selective search to extract a large quantity of object proposals and then computes CNN features for each of them. Finally, it classifies each region using the class-specific linear SVMs. Compared with traditional CNN structures which are mainly intended for image classification, R-CNN can address more complicated tasks, such as object detection and image segmentation, and it even becomes one important basis for both fields. Moreover, R-CNN can be built on top of any CNN benchmark structures, such as AlexNet, VGG, GoogLeNet, and ResNet. For the image segmentation task, R-CNN extracted 2 types of features for each region: full region feature and foreground feature, and found that it could lead to better performance when concatenating them together as the region feature. R-CNN achieved significant performance improvements due to using the highly discriminative CNN features. However, it also suffers from a couple of drawbacks for the segmentation task:

- The feature is not compatible with the segmentation task.
- The feature does not contain enough spatial information for precise boundary generation.
- Generating segment-based proposals takes time and would greatly affect the final performance.

Due to these bottlenecks, recent research has been proposed to address the problems, including SDS, Hypercolumns, Mask R-CNN.

Fully Convolutional Network-Based Semantic Segmentation

The original Fully Convolutional Network (FCN) learns a mapping from pixels to pixels, without extracting the region proposals. The FCN network pipeline is an extension of the classical CNN. The main idea is to make the classical CNN take as input arbitrary-sized images. The restriction of CNNs to accept and produce labels only for specific sized inputs comes from the fully-connected layers which are fixed. Contrary to them, FCNs only have convolutional and pooling layers which give them the ability to make predictions on arbitrary-sized inputs.

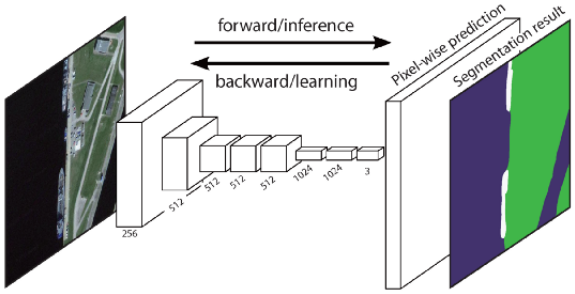

Figure 2.3: FCN Architecture

One issue in this specific FCN is that by propagating through several alternated convolutional

and pooling layers, the resolution of the output feature maps is down sampled. Therefore, the direct predictions of FCN are typically in low resolution, resulting in relatively fuzzy object boundaries. A variety of more advanced FCN-based approaches have been proposed to address this issue, including SegNet, DeepLab-CRF, and Dilated Convolutions.

Weakly Supervised Semantic Segmentation

Most of the relevant methods in semantic segmentation rely on a large number of images with pixel-wise segmentation masks. However, manually annotating these masks is quite time-consuming, frustrating and commercially expensive. Therefore, some weakly supervised methods have recently been proposed, which are dedicated to fulfilling the semantic segmentation by utilizing annotated bounding boxes.

Figure 2.4: Boxsup Training

For example, Boxsup employed the bounding box annotations as a supervision to train the network and iteratively improve the estimated masks for semantic segmentation. Simple Does It treated the weak supervision limitation as an issue of input label noise and explored recursive training as a de-noising strategy. Pixel-level Labeling interpreted the segmentation task within the multiple-instance learning framework and added an extra layer to constrain the model to assign more weight to important pixels for image-level classification.

U-Net

This is a network whose training strategy relies on heavy use of data augmentation in order to use the annotated images efficiently. The architecture is made up of:

- a contracting path that captures context
- and a symmetric expanding path whose role is to enable precise localization

In this architecture, a fully connected convolutional layer is modified so as to work on a few training images yet yielding more accurate precision. This is important since this architecture was initially proposed for medical image processing where lots of training data is scarce. The model applies elastic deformations on the available data in order to achieve augmentation [16].

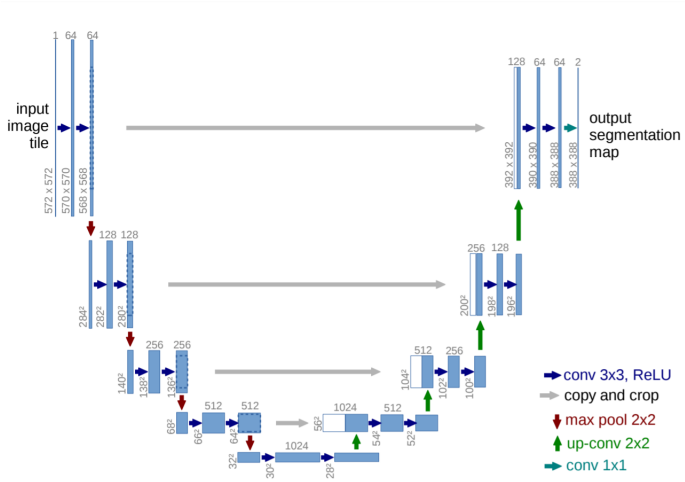

Figure 2.5: Unet Architecture(example for 32*32 pixels in the lowest resolution).Each blue box corresponds to a multi channe lis denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represnt copied feature maps. The arrows denote the different operations.

The contracting path consists of two 3 by 3 convolutions which are followed by a rectified linear unit and a 2 by 2 max pooling operation. Downsampling is done by the pooling operation. At each downsampling stage the number of feature channels are doubled. The feature samples are upsampled by the expansive path. Upsampling is followed by a 2 by 2 up-convolution that halves the number of feature channels. The final layer maps the component feature vectors to the needed number of classes. This layer is a 1 by 1 convolution. The model is trained using the input images, their segmentation maps and a stochastic gradient descent based on Caffe. The full implementation and the trained networks can be found here. On experimentation, the model achieved a mean intersection-over-union (IOU) of 92IOU is a common metric used to evaluate the performance of object detection and segmentation models. This metric is computed from the ground truth mask and the predicted mask.

Figure 2.6: From Left original image, original annotation (ground truth) masks and predicted masks

2.4 Conclusion

Deep learning semantic segmentation attracts a large number of researchers; there exist already a high number of contributions. In the next chapter, we present the SOD algorithm that localize salient region in images using the Pyramid Scene Parsing Network (PSPNet) for semantic segmentation step.

CHAPTER 3

SALIENT OBJECT DETECTION METHODOLOGY

3.1 Introduction

In this chapter we present the methodology that aim to directly segment and localize salient objects, begining from semantic segmentation step, then the Detection of visually salient image regions until the color divergence technique to generate the final saliency map.

3.2 Methodology of salient object detection

A large number of approaches have been proposed for detecting salient objects in the past two decades. most of these approaches aim to identify the salient subsets from images first (i.e., compute a saliency map) and then integrate them to segment the entire salient object.

Our approach builds on previous work with several extensions of the latest algorithms.

3.2.1 Salient object detection algorithm

the detection of the salient object in digital image consist of three basic steps :

- Deep learning semantic segmentation using Pyramid Scene Parsing Network (PSPNet)[25]
- Detection of visually salient image regions using Frequency-tuned Salient Region algorithm
- Generating the final saliency map based on the Color Divergence Technique. An overview of the algorithm is shown in Figure **??** and more detailed descriptions of the algorithm are provided in the next section.

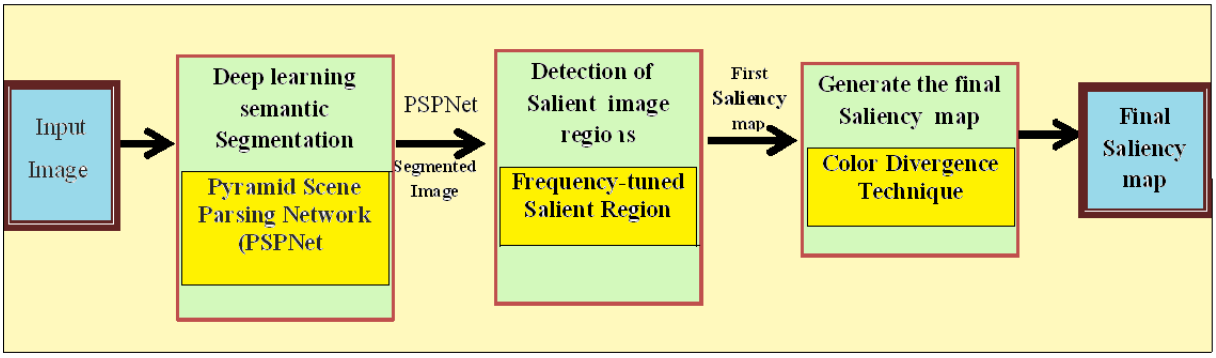

Figure 3.1: Steps of the SOD algorithm.

3.3 Description of the salient object detection algorithm

We will explain the three stages of the SOD algorithm. These steps are sequential; each operation is performed on the output image of the previous operation.

3.3.1 Deep learning semantic segmentation

There are many other semantic segmentation algorithms like PSPNet, Deeplab, etc. In this thesis, we will mainly focus on Pyramid scene parsing network (PSPNet) which is one of the most well-recognized image segmentation algorithms

Transfer learning

Using the principle of transfer learning, we can leverage knowledge (features, weights etc) from previously trained models for training newer models and even tackle problems like having less data for the newer task. Transfer learning enable us to utilize knowledge from previously learned tasks and apply them to newer, related ones. There are several models available for semantic segmentation. The model architecture shall be chosen properly depending on the use case. There are several things which should be taken into account:

1. The number of training images
2. Size of the images
3. The domain of the images

For images containing indoor and outdoor scenes, PSPNet is preferred, as the objects are often present in different sizes.

Pyramid Scene Parsing Network (PSPNet)

The Pyramid Scene Parsing Network is optimized to learn better global context representation of a scene. First, the image is passed to the base network to get a feature map. The feature map is downsampled to different scales. Convolution is applied to the pooled feature maps. After that, all the feature maps are upsampled to a common scale and concatenated together. Finally a another convolution layer is used to produce the final segmentation outputs. Which is illustrated in figure 3.2 Here, the smaller objects are captured well by the features pooled to a high resolution, whereas the large objects are captured by the features pooled to a smaller size.

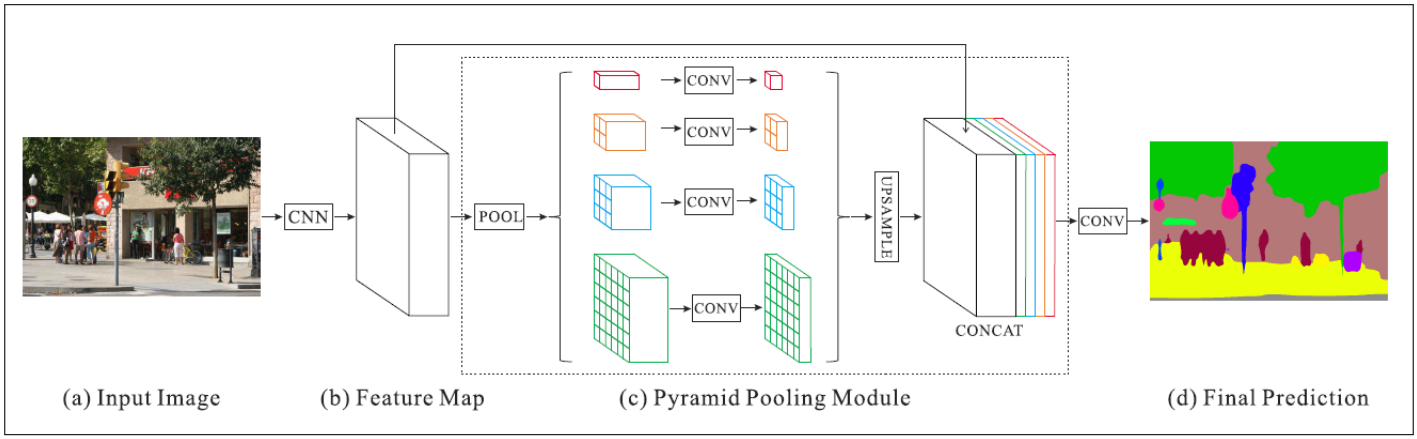

Figure 3.2: Overview of the Pyramid Scene Parsing Network (PSPNet)

1. Input Image:

   image of any shape usually dimensions greater than (256, 256) feed to the network.

2. Feature Map:

   input image and constructs feature maps for image. Feature maps are extracted by feeding image using transfer learning or scratch network with dilated convolutions.

   As large size kernels extracts more useful information than small size kernel but computation cost is higher, dilated convolutions gathers large size area information with small size kernel for higher dilation rates to keep dimensions same as input Image. Generally residual blocks with dilations are used to construct feature maps.

3. Pyramid Pooling Module:

   Image contains objects of sizes ranging from small area to large area in different regions.Fully Convolution Network(FCN),U-Net and other networks constructs feature maps by upsampling and doing segmentation at different levels for segmentation of all objects in all regions.But in PSPNet to correctly segment all size objects, feature maps are pooled average pooled at different pool size.

   Sub-region average pooling is done at different scales like Global Average Pooling,$(2 \times 2), (3 \times 3), (4 \times 4), (6 \times 6), (8 \times 8) \ldots$ The pyramid pooling module fuses features under four different pyramid scales. The output of different levels in the pyramid pooling module contains the feature map with varied sizes. Using the 4-level pyramid, the pooling kernels cover the whole, half of, and small portions of the image. They are fused as the global prior. Then we concatenate the prior with the original feature map in the final part of (c) of figure 3.2

4. Final Prediction:

Feature maps of the representation are feed to convolution layer and final prediction of classes are generated on how output layer is constructed say different channels for different objects or single channel.

the figure 3.3 shows the result of the PSPNet semantic segmentation

Figure 3.3: Input image and the result of semantic segmentation

3.3.2 Detection of visually salient image regions

Visual saliency is the perceptual quality that makes an object, person, or pixel stand out relative to its neighbors and thus capture our attention. in this subsection we present the Frequency-tuned Salient Region method.

Frequency-tuned Salient Region

In this method find the Euclidean distance between the Lab pixel vector in a Gaussian filtered image with the average Lab vector for the input image. This is illustrated in the figure below [5].

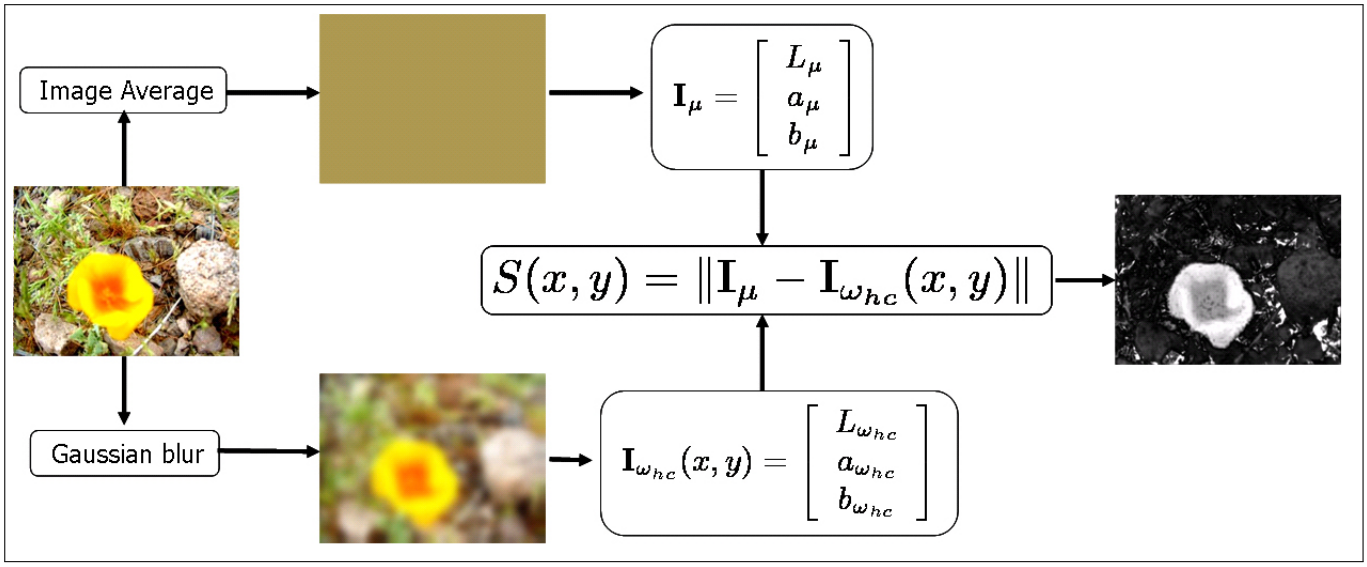

Figure 3.4: The principle of the Frequency-tuned Salient Region

So, finding the saliency map S for an image I of width W and height H pixels can thus be formulated as:

$$S(x, y) = ||I_\mu - I_{whc}(x, y)|| \tag{3.1}$$

where I μ is the mean image feature vector, I ω hc (x, y) is the corresponding image pixel vector value in the Gaussian blurred version (using a 5 $\times$ 5 separable binomial kernel) of the input image, and || is the L 2 norm. Using the Lab color space, each pixel location is an [L, a, b] T vector, and the L 2 norm is the Euclidean distance. This is illustrated in the figure 3.5 below

Figure 3.5: The result of the Frequency-tuned Salient Region method

3.3.3 Generating the final saliency map

We observe that the difficulty to detect the salient objects or to draw the boundary of an object depends on the quality and complexity of the image. To generate the final saliency map, we use the

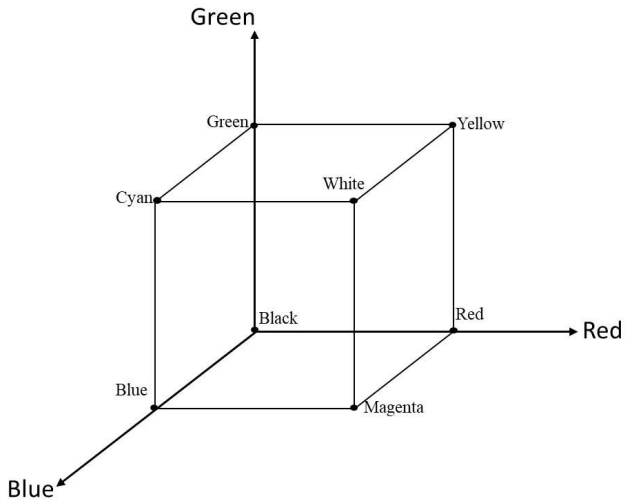

Figure 3.6: A 3D cube geometrically representing the RGB color space

principale of the color divergence technique.

Color Divergence Technique

The RGB color space is characterized by a cube with R, G, and B axis, and colors are interpreted as three-dimensional $(3D)$ vectors, with each vector element having an 8-bit dynamic range. Each of the three edges of the cube has length 256, since each color component may assume any value in the range [0, 255] as shown in Figure 3.6.

In this step, we will redistribute the grayscale value of each pixel of the input image in a 3-D cube. For each pixel, eight values (distances between this point and eight vertices of the cube) will be computed. The distance vector of a given pixel p is defined as $D_p = [d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7]$, where d_i corresponds to the distance between the pixel p and the vertex $i(i = 0 \dots 7)$ of the cube

The ninth value is the standard deviation of the eight values computed. We define the standard deviation vector sdv.

$$sdv_p = \frac{\sqrt{\sum(d_i - averageD_p)^2}}{8} \tag{3.2}$$

where d_i corresponds to the distance of the pixel p and the vertices $i(i = 0 \dots 7)$ of the cube. $averageD_p$ is the average of the distance vector D_p.

Specifically, we first use the Euclidean distance matrix between each grayscale value pixel of the input image and the eight vertices of the 3-D cube. then we define the coefficient f . The cube that presents the image will be presented in a new cube with irregular edges. Hence, we assign a different value of f for each cube vertex.

$$d_i = f_i * d_i \tag{3.3}$$

To detect the salient objects, we use a table which contains the value of the standard deviation sdv and the pixel values of the input image. The table will be sorted in an ascending order following

the standard deviation *sdv* . Then we divide it into classes where the pixels of the same class will represent a class of objects. The last class represents the most salient objects that exist in the input image. Example of the final saliency map is shown in the figure 3.7

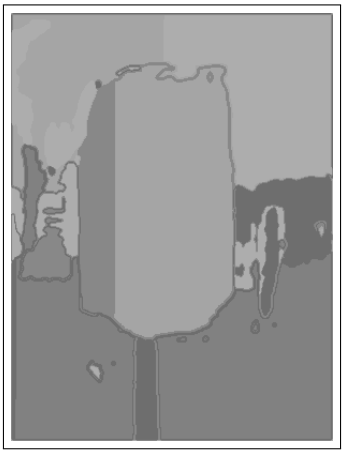

Figure 3.7: The final saliency map

3.4 Conclusion

In this chapter, we presented a scheme for detecting salient object in images. The implementation of the algorithm explained in this chapter and the presentation of the experimental results are the subject of the next chapter.

CHAPTER 4

IMPLEMENTATION

4.1 Introduction

Finally, in this chapter we will present the work environment of the application. Then we will describe the implementation of different steps of the SOD algorithm.

4.2 workspace environment

4.2.1 Python:

is a popular programming language, it was created by Guido van Rossum, and released in 1991, it is used for (web development, software development, mathematics, system scripting). A key benefit of using Python is its active developer community and the large amount of available software packages available from "pypi.org" . Python works on different platforms (Windows, Mac, Linux, etc.),it has a simple syntax similar to the English language, which allows developers to write programs with fewer lines than most programming languages. Python runs on an interpreter system, meaning that code can be executed as soon as it is written, This means that prototyping can be very quick, Python can be treated in (a procedural way, an object-orientated way or a functional way).

4.2.2 Keras

1. keras definition:

   Keras is an open-source library of neural network components written in Python [26]. Keras is capable of running atop TensorFlow, Theano, PlaidML and others. The library was developed to be modular and user-friendly, however it initially began as part of a research project for the Open-ended Neuro-Electronic Intelligent Operating System or ONEIROS. The principal author of Keras is Francois Chollet, a Google engineer who also wrote XCeption, a deep neural network model. While Keras officially launched, it was not integrated into Google's TensorFlow core library until 2017. Additional support has also been added for Keras integration with Microsoft Cognitive Toolkit.

2. Keras Principles:

   Keras was created to be user friendly, modular, easy to extend, and to work with Python. The API was "designed for human beings, not machines," and "follows best practices for reducing cognitive load." Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files..

3. Keras Advantages :

The biggest reasons to use Keras stem from its guiding principles, primarily the one about being user friendly. Beyond ease of learning and ease of model building, Keras offers the advantages of broad adoption, support for a wide range of production deployment options, integration with at least five back-end engines (TensorFlow, CNTK, Theano, MXNet, and PlaidML), and strong support for multiple GPUs and distributed training. Plus, Keras is backed by Google, Microsoft, Amazon, Apple, Nvidia, Uber, and others.

4.2.3 TensorFlow

1. TensorFlow definition :

   is an open source framework developed by Google researchers to run machine learning [27], deep learning and other statistical and predictive analytics workloads. Like similar platforms, it's designed to streamline the process of developing and executing advanced analytics applications for users such as data scientists, statisticians and predictive modelers.

2. TensorFlow Advantages:

   TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

   - Easy model building: Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.
   - Robust ML production anywhere: Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use.
   - Powerful experimentation for research: A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

3. TensorFlow Architecture:

   Tensorflow architecture works in three parts:

   - Preprocessing the data
   - Build the model
   - Train and estimate the model.

   It is called Tensorflow because it takes input as a multi-dimensional array, also known as tensors. You can construct a sort of flowchart of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output. This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

4.2.4 Google Colaboratory:

1. Colaboratory definition:

   Colab is a free Jupyter notebook environment that runs entirely in the cloud [28]. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

2. Colab Offers:

   As a programmer, you can perform the following using Google Colab.

   - Write and execute code in Python segmentation.
   - Document your code that supports mathematical equations
   - Create/Upload/Share notebooks
   - Import/Save notebooks from/to Google Drive
   - Import/Publish notebooks from GitHub
   - Import external datasets e.g. from Kaggle
   - Integrate PyTorch, TensorFlow, Keras, OpenCV
   - Free Cloud service with free GPU

4.3 Implementation of the SOD algorithm

In this part, we will see the code of applying different stages of the SOD algorithm

4.3.1 semantic segmentation

For the deep learning semantic segmentation we use the Pyramid Scene Parsing Network (PSPNet) the pretrained model trained on Cityscapes dataset, which is illustrated in figure 4.1.

```
from keras_segmentation.pretrained import pspnet_101_cityscapes

model = pspnet_101_cityscapes() # load the pretrained model trained on Cityscapes dataset

# load the pretrained model

out = model.predict_segmentation(
    inp="../image/280.jpg",
    out_fname="outi.jpg"
)
```

Figure 4.1: load the pretrained model of the PSPNet

The first step in training our segmentation model is to download the dataset [29] already prepared. Each input RGB images have the corresponding ground truth images. The code for downloading the dataset is shown in figure 4.2.

```
! wget https://github.com/divamgupta/datasets/releases/download/seg/dataset1.zip && unzip dataset1.zip
inflating: __MACOSX/dataset1/annotations_prepped_test/._0016E5_08077.png
inflating: dataset1/annotations_prepped_test/0016E5_08063.png
inflating: __MACOSX/dataset1/annotations_prepped_test/._0016E5_08063.png
inflating: dataset1/annotations_prepped_test/0016E5_08103.png
inflating: __MACOSX/dataset1/annotations_prepped_test/._0016E5_08103.png
inflating: dataset1/annotations_prepped_test/0016E5_08117.png
inflating: __MACOSX/dataset1/annotations_prepped_test/._0016E5_08117.png
inflating: dataset1/annotations_prepped_test/0016E5_07973.png
inflating: __MACOSX/dataset1/annotations_prepped_test/._0016E5_07973.png
inflating: dataset1/annotations_prepped_test/0016E5_07967.png
inflating: __MACOSX/dataset1/annotations_prepped_test/._0016E5_07967.png
inflating: dataset1/annotations_prepped_test/0016E5_07971.png
inflating: __MACOSX/dataset1/annotations_prepped_test/._0016E5_07971.png
inflating: dataset1/annotations_prepped_test/0016E5_07965.png
```

Figure 4.2: Download the dataset for training the model

Then we initialize the transfer learning model as described in figure 4.3.

```
from keras_segmentation.models.pspnet import pspnet_101


model = pspnet_101(n_classes=50 ,  input_height=473, input_width=473, channels=3  )
```

Figure 4.3: Initialize the model for training

Using the transfer learning principle we train the model

```
model.train(
    train_images =  "dataset1/images_prepped_train/",
    train_annotations = "dataset1/annotations_prepped_train/",
    checkpoints_path = "/tmp/pspnet_101_1" , epochs=5
)
```

Figure 4.4: Training the model

when executing the model.summary(), we see the three part of the PSPNet architecture :

```
model.summary()
```

Figure 4.5: To see the summary of the architecture of the trained model

- The first part : architecture of the CNN to get the feature map :

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 473, 473, 3)	0	
conv1_1_3x3_s2 (Conv2D)	(None, 237, 237, 64)	1728	input_1[0][0]
conv1_1_3x3_s2_bn (BatchNormali	(None, 237, 237, 64)	256	conv1_1_3x3_s2[0][0]
activation (Activation)	(None, 237, 237, 64)	0	conv1_1_3x3_s2_bn[0][0]
conv1_2_3x3 (Conv2D)	(None, 237, 237, 64)	36864	activation[0][0]
conv1_2_3x3_bn (BatchNormalizat	(None, 237, 237, 64)	256	conv1_2_3x3[0][0]
activation_1 (Activation)	(None, 237, 237, 64)	0	conv1_2_3x3_bn[0][0]
conv1_3_3x3 (Conv2D)	(None, 237, 237, 128	73728	activation_1[0][0]
conv1_3_3x3_bn (BatchNormalizat	(None, 237, 237, 128	512	conv1_3_3x3[0][0]
activation_2 (Activation)	(None, 237, 237, 128	0	conv1_3_3x3_bn[0][0]
max_pooling2d (MaxPooling2D)	(None, 119, 119, 128	0	activation_2[0][0]
activation_3 (Activation)	(None, 119, 119, 128	0	max_pooling2d[0][0]

activation_100 (Activation)	(None, 60, 60, 512)	0	conv5_3_1x1_reduce_bn[0][0]
zero_padding2d_32 (ZeroPadding2	(None, 68, 68, 512)	0	activation_100[0][0]
conv5_3_3x3 (Conv2D)	(None, 60, 60, 512)	2359296	zero_padding2d_32[0][0]
conv5_3_3x3_bn (BatchNormalizat	(None, 60, 60, 512)	2048	conv5_3_3x3[0][0]
activation_101 (Activation)	(None, 60, 60, 512)	0	conv5_3_3x3_bn[0][0]
conv5_3_1x1_increase (Conv2D)	(None, 60, 60, 2048)	1048576	activation_101[0][0]
conv5_3_1x1_increase_bn (BatchN	(None, 60, 60, 2048)	8192	conv5_3_1x1_increase[0][0]
add_32 (Add)	(None, 60, 60, 2048)	0	conv5_3_1x1_increase_bn[0][0] activation_99[0][0]
activation_102 (Activation)	(None, 60, 60, 2048)	0	add_32[0][0]

Figure 4.6: Overview of the CNN layers

- The second part : the Pyramid Pooling Module

activation_102 (Activation)	(None, 60, 60, 2048)	0	add_32[0][0]
average_pooling2d_3 (AveragePoo	(None, 6, 6, 2048)	0	activation_102[0][0]
average_pooling2d_2 (AveragePoo	(None, 3, 3, 2048)	0	activation_102[0][0]
average_pooling2d_1 (AveragePoo	(None, 2, 2, 2048)	0	activation_102[0][0]
average_pooling2d (AveragePooli	(None, 1, 1, 2048)	0	activation_102[0][0]
conv5_3_pool6_conv (Conv2D)	(None, 6, 6, 512)	1048576	average_pooling2d_3[0][0]
conv5_3_pool3_conv (Conv2D)	(None, 3, 3, 512)	1048576	average_pooling2d_2[0][0]
conv5_3_pool2_conv (Conv2D)	(None, 2, 2, 512)	1048576	average_pooling2d_1[0][0]
conv5_3_pool1_conv (Conv2D)	(None, 1, 1, 512)	1048576	average_pooling2d[0][0]
conv5_3_pool6_conv_bn (BatchNor	(None, 6, 6, 512)	2048	conv5_3_pool6_conv[0][0]
conv5_3_pool3_conv_bn (BatchNor	(None, 3, 3, 512)	2048	conv5_3_pool3_conv[0][0]
conv5_3_pool2_conv_bn (BatchNor	(None, 2, 2, 512)	2048	conv5_3_pool2_conv[0][0]
conv5_3_pool1_conv_bn (BatchNor	(None, 1, 1, 512)	2048	conv5_3_pool1_conv[0][0]
activation_106 (Activation)	(None, 6, 6, 512)	0	conv5_3_pool6_conv_bn[0][0]
activation_105 (Activation)	(None, 3, 3, 512)	0	conv5_3_pool3_conv_bn[0][0]
activation_104 (Activation)	(None, 2, 2, 512)	0	conv5_3_pool2_conv_bn[0][0]
activation_103 (Activation)	(None, 1, 1, 512)	0	conv5_3_pool1_conv_bn[0][0]
interp_3 (Interp)	(None, 60, 60, 512)	0	activation_106[0][0]
interp_2 (Interp)	(None, 60, 60, 512)	0	activation_105[0][0]
interp_1 (Interp)	(None, 60, 60, 512)	0	activation_104[0][0]
interp (Interp)	(None, 60, 60, 512)	0	activation_103[0][0]
concatenate (Concatenate)	(None, 60, 60, 4096)	0	activation_102[0][0] interp_3[0][0] interp_2[0][0] interp_1[0][0] interp[0][0]

Figure 4.7: Pyramid Pooling Module layers

the third part : Final Prediction

conv5_4 (Conv2D)	(None, 60, 60, 512)	18874368	concatenate[0][0]
conv5_4_bn (BatchNormalization)	(None, 60, 60, 512)	2048	conv5_4[0][0]
activation_107 (Activation)	(None, 60, 60, 512)	0	conv5_4_bn[0][0]
dropout (Dropout)	(None, 60, 60, 512)	0	activation_107[0][0]
conv6 (Conv2D)	(None, 60, 60, 50)	25650	dropout[0][0]
interp_4 (Interp)	(None, 473, 473, 50)	0	conv6[0][0]
reshape (Reshape)	(None, 223729, 50)	0	interp_4[0][0]
activation_108 (Activation)	(None, 223729, 50)	0	reshape[0][0]

Total params: 65,834,226
Trainable params: 65,723,378
Non-trainable params: 110,848

Figure 4.8: Final Prediction layers

4.3.2 Detection of visually salient image regions

Using the Frequency-tuned Salient Region method we get the first saliency map :

```
import pyimgsaliency as psal
import cv2

# path to the image
filename = '../image/out2.png'

# get the saliency map using the implemented Frequency-tuned Salient Region methods

ft = psal.get_saliency_ft(filename).astype('uint8')

img = cv2.imread(filename)

cv2.imshow('img',img)
#cv2.imshow('rbd',rbd)
cv2.imshow('ft',ft)
#cv2.imshow('mbd',mbd)

cv2.waitKey(0)
```

Figure 4.9: The call to the Frequency-tuned Salient Region method

4.3.3 Generate the final saliency map

The code of the Color Divergence method to get the final saliency map

```
#matrix that compute the dimension between pixel and 3D-cube
imge = np.zeros((w , h, 9), dtype='f')

for i in range (w) :
        for j in range (h) :
                x = img.getpixel((i,j))
                imge[i][j][0] = 100* sqrt (x**2 + x**2 + x**2 ) # distance between grayscal pixel and cube vertex with (0,0,0) coordinate
                if (d1max < imge[i][j][0]):
                        d1max = imge[i][j][0]
                imge[i][j][1] = 150 * sqrt ((255 - x)**2 + x**2 + x**2 ) # distance between grayscal pixel and cube vertex with (255,0,0) coordinate
                imge[i][j][2] = 50 * sqrt (x**2 + (255 - x)**2 + x**2 ) # distance between grayscal pixel and cube vertex with (0,255,0) coordinate
                imge[i][j][3] = 50 * sqrt (x**2 + x**2 + (255 - x**2 )) # distance between grayscal pixel and cube vertex with (0,0,255) coordinate
                imge[i][j][4] = 100 * sqrt ((255 - x)**2 + (255 - x)**2 + x**2 ) # distance between grayscal pixel and cube vertex with (255,255,0) c
                imge[i][j][5] = 100 * sqrt ((255 - x)**2 + x**2 + (255 - x)**2 ) # distance between grayscal pixel and cube vertex with (255,0,255) c
                imge[i][j][6] = 1 * sqrt (x**2 + (255 - x)**2 + (255 - x)**2 ) # distance between grayscal pixel and cube vertex with (0,255,255) coo
                imge[i][j][7] = 50 * sqrt ((255 - x)**2 + (255 - x)**2 + (255 - x)**2 ) # distance between grayscal pixel and cube vertex with (255,2
                imge[i][j][8] = ecartype([imge[i][j][k] for k in range (8)]) #standard deviation of the computed distances
```

Figure 4.10: The distance between each grayscal pixel and the 3D-cube vertex

Generating the final saliency map following the standard deviation of the computed distance

```
RGBsort = np.zeros((w*h, 4), dtype='f')
n=0
for i in range (w) :
        for j in range (h) :
                x,y = [i,j]
                RGBsort[n] = (imge[i][j][0],imge[i][j][8],x,y )
                n = n+1
sortlist = sorted(RGBsort, key=itemgetter(1))

length = int(h*w*0.1)
imgr = np.array(img)

px = img.load()
l=0
inc = 0
x,y = int(sortlist[0][2]),int(sortlist[0][3])
rgb = px[x,y]
for i in range(w):
    for j in range(h):
        if l <= length :
            x,y = int(sortlist[inc][2]),int(sortlist[inc][3])
            imgr[y,x] = rgb
            inc = inc + 1
            l = l+1
        else:
            x,y = int(sortlist[inc][2]),int(sortlist[inc][3])
            imgr[y,x] = rgb
            inc = inc + 1
            rgb = px[x,y]
            l = 0
```

Figure 4.11: To get the final saliency map

4.4 Results and Experiments

In the following the result of the application of the SOD algorithm before transfer learning of the PSPNet semantic segmentation

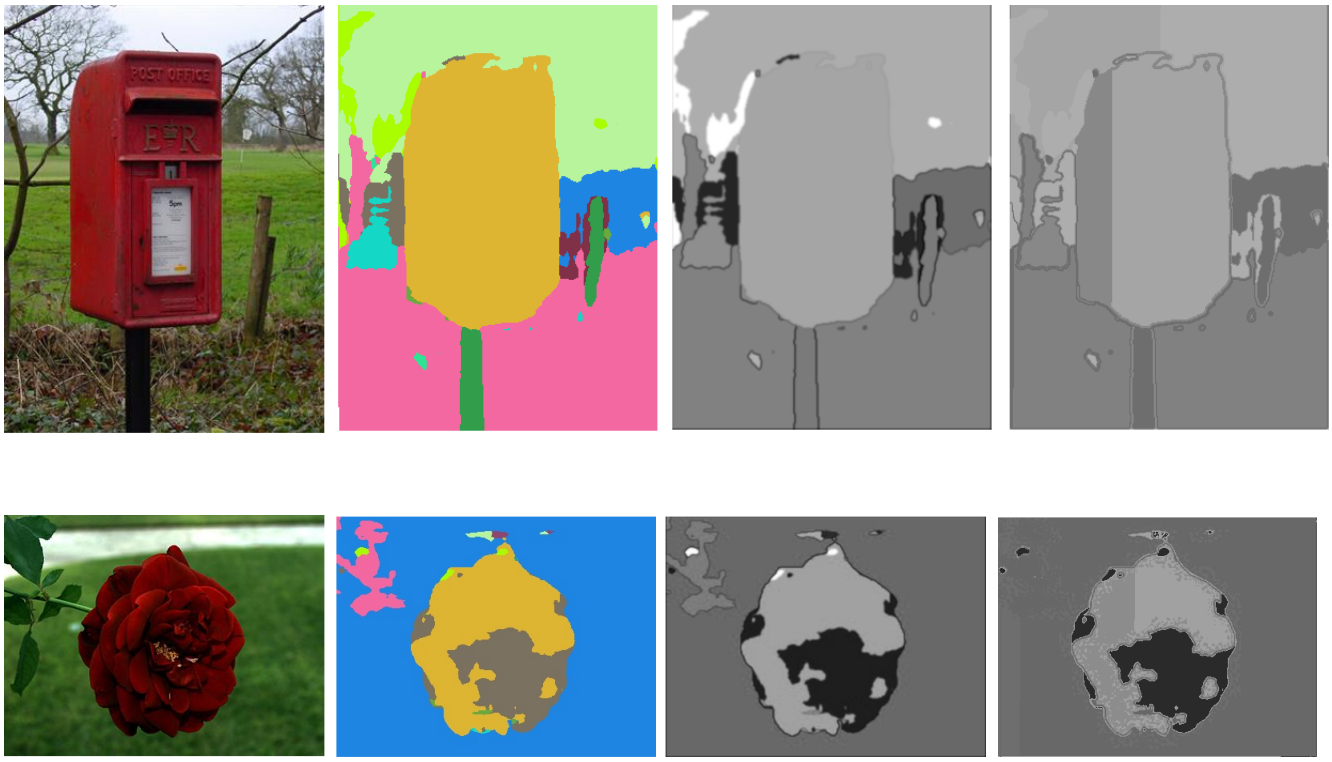

Figure 4.12: from left to right original image, segmented image, first saliency map, final saliency map

Often, it is desirable to have a binary saliency map, here we use the Color Divergence technique also.

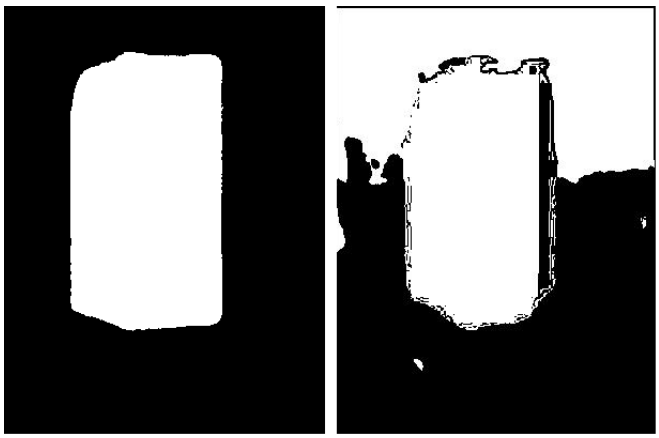

Figure 4.13: Left : Ground Truth map. Right : Binary saliency map of the first row image

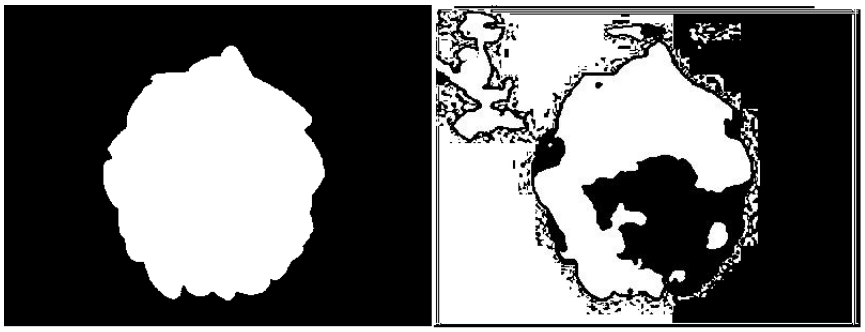

Figure 4.14: Left : Ground Truth map. Right : Binary saliency map of the second row image

In the following the result of the application of the SOD algorithm after the transfer learning of the PSPNet semantic segmentation using the SOD dataset [29]

Figure 4.15: from left to right original image, segmented image, first saliency map, final saliency map

We use the Color Divergence technique to get the binary saliency map.

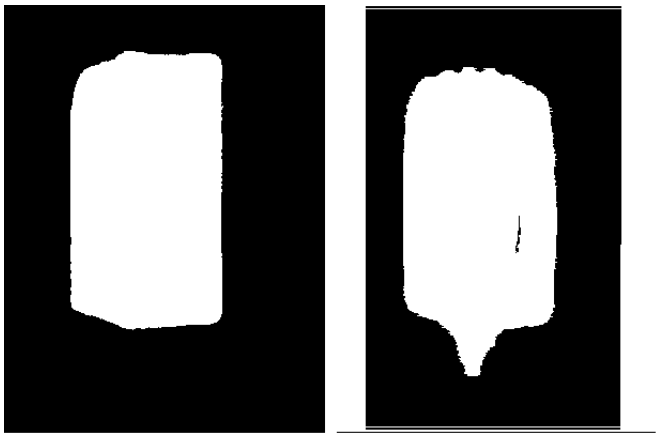

Figure 4.16: Left : Ground Truth map. Right : Binary saliency map of the first row image

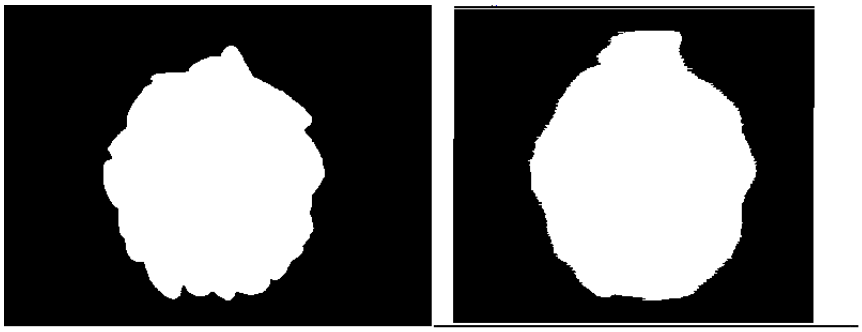

Figure 4.17: Left : Ground Truth map. Right : Binary saliency map of the second row image

4.5 Conclusion

After we explained the concepts, in the previous chapters, in this chapter we implemented the SOD algorithm with all the stages that contain various steps that we saw previously. In the end, we detect the most salient region in the original image, the result was acceptable especially after the transfer learning of the deep learning step with the SOD dataset.

GENERAL CONCLUSION

The human vision system intends to extract the most informative objects and regions in a scene naturally, and then combines these local information to efficiently understand the whole scene. Salient object detection aims at finding the most important objects in a scene in order to simulate the functionality of biological vision systems. In this thesis, we presented the proposed approach that aim to detect salient region in image, this approach includes three main steps : Begining from deep learning semantic segmentation based on the Pyramid Scene Parsing Network (PSPNet) which is one of the most well-recognized image segmentation algorithms for images containing indoor and outdoor scenes, and it is preferred as the objects are often present in different sizes. Next step is the detection of salient region in image using the Frequency-tuned salient region algorithm, this method is simple to implement and computationally efficient. The last step, will generate the final saliency map based on the color divergence technique and RGB-cube classification focusing on the distance vector and standard deviation values for the clustering of grayscale pixels. The results obtained in a collection of test images are encouraging and confirm the success of the approach.

As perspective, we will construct our own model and train it to efficiently generate the salient region in image.

BIBLIOGRAPHY

[1] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, "Advanced deep-learning techniques for salient and category-specific object detection: a survey," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 84–100, 2018.

[2] W. Wang, Q. Lai, H. Fu, J. Shen, and H. Ling, "Salient object detection in the deep learning era: An in-depth survey," *arXiv preprint arXiv:1904.09146*, 2019.

[3] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li, "Salient object detection: A survey," *Computational Visual Media*, pp. 1–34, 2014.

[4] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum, "Learning to detect a salient object," *IEEE Transactions on Pattern analysis and machine intelligence*, vol. 33, no. 2, pp. 353–367, 2010.

[5] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned salient region detection," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 1597–1604.

[6] G. Li and Y. Yu, "Deep contrast learning for salient object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 478–487.

[7] J. Kuen, Z. Wang, and G. Wang, "Recurrent attentional networks for saliency detection," in *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3668–3677.

[8] S. S. Kruthiventi, V. Gudisa, J. H. Dholakiya, and R. Venkatesh Babu, "Saliency unified: A deep architecture for simultaneous eye fixation prediction and salient object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5781–5790.

[9] J. Zhang, Y. Dai, and F. Porikli, "Deep salient object detection by integrating multi-level cues," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 1–10.

[10] Q. Hou, M.-M. Cheng, X. Hu, A. Borji, Z. Tu, and P. H. Torr, "Deeply supervised salient object detection with short connections," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3203–3212.

[11] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1395–1403.

[12] P. Zhang, D. Wang, H. Lu, H. Wang, and X. Ruan, "Amulet: Aggregating multi-level convolutional features for salient object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 202–211.

[13] X. Hu, L. Zhu, J. Qin, C.-W. Fu, and P.-A. Heng, "Recurrently aggregating deep features for salient object detection," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] X. Li, F. Yang, H. Cheng, W. Liu, and D. Shen, "Contour knowledge transfer for salient object detection," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[15] N. Liu, J. Han, and M.-H. Yang, "Picanet: Learning pixel-wise contextual attention for saliency detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3089–3098.

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[17] J.-J. Liu, Q. Hou, M.-M. Cheng, J. Feng, and J. Jiang, "A simple pooling-based design for real-time salient object detection," *arXiv preprint arXiv:1904.09569*, 2019.

[18] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[19] Y. Ji, H. Zhang, K.-K. Tseng, T. W. Chow, and Q. J. Wu, "Graph model-based salient object detection using objectness and multiple saliency cues," *Neurocomputing*, vol. 323, pp. 188–202, 2019.

[20] X. Xi, Y. Luo, P. Wang, and H. Qiao, "Salient object detection based on an efficient end-to-end saliency regression network," *Neurocomputing*, vol. 323, pp. 265–276, 2019.

[21] N. Voisine, "Approche adaptative de coopération hiérarchique de méthodes de segmentation: application aux images multicomposantes," Ph.D. dissertation, Rennes 1, 2002.

[22] Y. J. Zhang, "A survey on evaluation methods for image segmentation," *Pattern recognition*, vol. 29, no. 8, pp. 1335–1346, 1996.

[23] "Deep learning," https://deepai.org/machine-learning-glossary-and-terms/deep-learning, accessed: 2021-08-20.

[24] J. Le, "How to do semantic segmentation using deep learning," https://nanonets.com/blog/how-to-do-semantic-segmentation-using-deep-learning/, accessed: 2021-08-20.

[25] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[26] "Keras definition," https://deepai.org/machine-learning-glossary-and-terms/keras, accessed: 2021-09-10.

[27] "What is tensorflow? - definition from whatis.com," view-source:https://searchdatamanagement.techtarget.com/definition/TensorFlow, accessed: 2021-09-10.

[28] "What is google colab?" https://ledumjg.medium.com/what-is-google-colab-281c5b59638f, accessed: 2021-09-10.

[29] A. Borji, M.-M. Cheng, H. Jiang, and J. Li, "Salient object detection: A benchmark," *IEEE TIP*, vol. 24, no. 12, pp. 5706–5722, 2015.