UNIVERSITÀ degli STUDI di CATANIA | DIPARTIMENTO DI INGEGNERIA ELETTRICA, ELETTRONICA E INFORMATICA

Dottorato di Ricerca in Ingegneria dei Sistemi, Energetica, Informatica e delle Telecomunicazioni
XXIX Ciclo

Ph.D. Thesis

HYBRID HUMAN–MACHINE VISION SYSTEMS FOR AUTOMATED OBJECT SEGMENTATION AND CATEGORIZATION

Ing. Simone Palazzo

Coordinatore	Tutor
Chiar.mo Prof.	Chiar.ma Prof.ssa
P. Arena	D. Giordano

ABSTRACT

Emulating human perception is a foundational component in the research towards artificial intelligence (AI). Computer vision, in particular, is now one of the most active and fastest growing research topics in AI, and its field of practical applications range from video-survaillance to robotics to ecological monitoring.

However, in spite of all the recent progress, humans still greatly outperform machines in most visual tasks, and even competitive artificial models require thousands of examples to learn concepts that children learn easily.

Hence, given the objective difficulty in emulating the human visual system, the question that we intended to investigate in this thesis is in which ways humans can support the advancement of computer vision techniques. More precisely, we investigated how the synergy between human vision expertise and automated methods can be shifted from a “top-down” paradigm — where direct user action or human perception principles explicitly guide the software component — to a “bottom-up” paradigm, where instead of trying to copy the way our mind works, we exploit the “by-product” (i.e. some kind of measured feedback) of its workings to extract information on how visual tasks are performed.

Starting from a purely top-down approach, where a fully-automated video object segmentation algorithm is extended to encode and include principles of human perceptual organization, we moved to interactive methods, where the same task is performed involving humans in the loop by means of gamification and eye-gaze–analysis strategies, in a progressively increasing bottom-up fashion. Lastly, we pushed this trend to the limit by investigating brain-driven image classification approaches, where brain signals were used to extract compact representation of image contents.

Performance evaluation of the tested approaches shows that involving people in automated vision methods can enhance their accuracy. Our experiments, carried out at different degrees of awareness and control of the generated human feedback, show that top-down approaches may achieve a better accuracy than bottom-up ones, at the cost of higher user interaction time and effort. As for our most ambitious objective, the purely bottom-up image classification system from brain pattern analysis, we were able to outperform the current state of the art with a method trained to extract brain-inspired visual content descriptors, thus removing the need of undergoing EEG recording for unseen images.

CONTENTS

CHAPTER

ONE

MOTIVATIONS AND OBJECTIVES

Creating artificial intelligence (AI) has been one of science's favourite and hardest challenges of the last century. Popular culture and science fiction have thrived on stories featuring robots and thinking machines ever since the beginning of 20th century (not to mention mythological references such as Hephaestus' golden assistants from the Iliad), and as technology quickly evolved, soon these stories attracted the interest of the scientific community, who naturally wondered how wide the gap between current fiction and future reality was.

Curiously enough, and understandably, AI research initially seemed to focus on how to design *artificial thought*, ignoring the importance and difficulty of some related tasks, e.g., how to emulate *perception*. In 1966, AI pioneer and MIT professor Marvin Minsky assigned a summer project consisting in linking a camera to a computer and have it recognize the surrounding environment and objects[1]. Unfortunately, the task turned out to be a bit more difficult than anticipated, as dozens of research groups, both in academia and industry, are still struggling to make machines understand what their electronic eyes see.

Computer vision is now one of the most active and fastest growing research fields in AI, and it has found its way into a wide range of applications, including robotics, video-surveillance, ecological monitoring, automated driving, visual-impairment support. At the same time, its data-oriented nature and the intrinsic patterned structure of the visual world made it an ideal playground for machine learning techniques, which were historically adopted as the main way to tackle this kind of problems.

Currently, the most widely spread and successful models for computer vision applications are Convolutional Neural Networks (CNNs) [1, 2, 3] (and variants thereof), a recently-rediscovered incarnation of a class of biologically-inspired mathematical models suitable for learning patterns from data. In the last half-decade, a combination of factors — mainly related to technological advances in data storage, content sharing and computing power — have led CNNs to advance practically any computer vision–related problem and to support the achievement of unprecedented results in more classic AI tasks, as perfectly exemplified by the recent victory of Google's AlphaGo software over a professional player at Go, an ancient game exponentially more complex than chess in terms of possible game states. Of course, the price for such improvements is paid in model complexity, hardware

[1]Original project description at `http://dspace.mit.edu/bitstream/handle/1721.1/6125/AIM-100.pdf`.

requirements and training time.

However, in spite of all the progress and optimism, it is undeniable that humans are still way better than machines at making sense of the world that they see, thanks to our multi-millenary experience in — well — seeing things and to our main advantage over current artificial intelligences, that is the capability to learn without anybody having to tell us what is what and to provide examples of all aspects of how the world works. In a more technical jargon, machines lack the ability to perform *unsupervised learning*, which is the current main reason why we are not likely to be seeing *Star Wars* droids walking among us anytime soon, and which is also the reason why even the most sophisticated computer vision systems need thousands of images to learn to recognize concepts (e.g., "tree") that a three-year-old child easily grasps with a few examples.

Given these premises and acknowledged the objective difficulty of current models to explicitly emulate the human visual system, especially the scene understanding processes, the question that we intend to investigate in this thesis is in which ways humans can support the advancement of computer vision techniques. More precisely, we intend to test and evaluate different modalities, with different degrees of involvement and awareness, for including humans in automated vision approaches.

Per se, this is nothing new: several "interactive" approaches following the "human-in-the-loop" paradigm have been proposed to help solving computer vision tasks — most notably, *object segmentation*, i.e. extracting object contours from an image. At a more basic level, crowdsourcing has been largely employed in computer vision to annotate image datasets for algorithm benchmarking, by asking users to specify the category (*class*) or to draw contours of objects in an image.

However, these kinds of interactions are still too *explicit*, by which we mean that users' actions are consciously targeted at the computer vision task they are working at, and as a consequence they need an incentive to motivate them to perform the task, which often amounts to monetary reward. A slightly less explicit user involvement modality is *gamification*, that is the process of putting a job in the form of a game, thus aiming to attract potential participants in a more natural way than crowdsourcing. The difficulty lies in how to make a game out of a job which is inherently non-entertaing, for there would be no need to perform gamification otherwise.

In these examples, human feedback is generated through an explicit and voluntary action by an operator working at a task. Here, we aim at exploring more implicit modalities for human knowledge and capabilities to back up computer vision, by integrating a kind of human feedback which comes in the form of physiological measurements taken while performing a task designed to solicit such reactions and to make them useful to automated methods. The idea behind this work is that implicit feedback can be potentially more informative than explicit, due to being generated by — and, in a sense, "closer to" — the very underlying physiological processes employed by humans to solve vision tasks, thus switching the paradigm from "human-in-the-loop" to "human-driven". In a different but orthogonal sense, we will investigate how the synergy between human potential and automated methods can be shifted from a "top-down" paradigm — where direct user action or human perception principles explicitly guide the software component — to a "bottom-up" paradigm, where instead of trying to copy the way our mind works, we exploit the "by-product" (i.e. some kind of measured feedback) of its workings to extract information on how visual tasks are performed.

In particular, we will investigate the employment of eye-tracking devices and electroencephalography (EEG) machines to collect, respectively, eye-gaze data and brain activity patterns while users simply watch videos or look at images on a screen, and how to extract information useful to the analysis of the observed scenes from the collected data. One might object that having a person sit down and watch a computer screen or, even worse, having him or her go through the hassle of mounting an EEG cap *and* watch a monitor screen are hardly "implicit" ways to get information on visual processes, in the sense that they still require the user to dedicate time exclusively to the experiment. While it is true that current eye-tracking and EEG recording protocols are relatively invasive (not in a medical sense, but in a time- and effort-demanding sense), this is merely a technological limitation: in fact, eye-tracking through smartphone cameras is currently being investigated [4], as well as headbands integrating EEG recording electronics[2]. While technology non-invasiveness is desirable, as it helps to attract potential participants, this is not what we mean by "implicitness", which instead refers to the degree of consciousness and control over the collected data — for instance, user clicks collected while playing a game represent a more "explicit" kind of feedback than EEG tracks recorded during daily routines.

In the following chapters, after an introduction to the state of the art of the current literature on the mentioned subjects, we will review different strategies of integrating the human factor in computer vision

[2]`http://www.daqri.com`

methods and provide examples of practical applications.

The very first method we propose, in Chapter 3, is actually purely automated, with no human intervention, and exemplifies the class of approaches where human knowledge is "injected" in a top-down fashion, by explicitly defining a mathematical model encoding principles of human perceptual organization. The problem which we tackle is *video object segmentation*, i.e. identifying and extracting the contours of moving objects in a video. Although this is a well-known problem, optimal and general solutions are hard to find, as current methods strongly depend on video and target object characteristics. By enforcing perceptual organization, we aim at providing useful constraints to an automated system for identifying objects which are more likely to represent structures from the real world.

In Chapter 4, we start by attempting to solve the same task through gamification. Since people can easily recognize moving objects from moving background (which can be present due to camera motion), the easiest way to support automated methods is to have the users identify those objects for them. The game we designed consists in asking the user to click on such objects while the video is played at above-than-normal speed, thus making the task challenging. The gathered data may be affected noisy and requires some pre-processing, but provides useful insights on object locations that can be exploited to simplify the approach presented in Chapter 3. Then, we explore a more implicit and bottom-up kind of human involvement, by replacing user clicks with eye-gaze data captured while users watch the videos. Although the nature of the recorded data is homogeneous to those collected through the game (both being punctual screen coordinates), here we remove any kind of top-down guidance or instructions, leaving only the subject's own attention processes to decide where to look. While this necessarily affects the reliability of the feedback, this approach has the advantage of requiring less interaction effort from the participant subjects, and we will see that it allows to achieve satisfying results, at the expenses of segmentation accuracy.

Our most ambitious attempt at transferring human capabilities to computer vision approaches is presented in Chapter 5: "reading the mind" of a person to predict the content of what he or she is currently looking at. In more practical and less magic-show terms, we investigate an automated approach to the problem of image classification — i.e. assigning a category to an image, based on its content — through the analysis of a subject's EEG tracks. Moreover, we also propose a methodology for learning a brain-inspired mathematical representation of visual content and building a model which extracts such representation from unseen images, thus allowing to perform the classification even without associated EEG signals.

CHAPTER

TWO

LITERATURE REVIEW

Computer vision is a vast and rapidly-evolving research field, to review which would require several books of their own. Our focus, however, does not lie in computer vision itself, but rather on exploring the different roles that humans can play to support computer vision approaches. Hence, for the sake of brevity, we will take the liberty of simplifying the architecture of an automated vision system by splitting it into two main tasks: *detecting the presence of objects* in visual media (i.e. images or videos) and *identifying what these objects are.*

Of course, since our main interest is studying new modalities of human involvement in automated vision algorithms, a background on interaction strategies targeted at computer vision is in order, and it should encompass both traditional top-down approaches employing "direct" interfaces such as computer mice, and more subtle modalities based on eye-tracking and brain activity analysis.

Therefore, in this chapter, we will introduce the reader with the state of the art on the above-mentioned notions, in order to give the necessary basis to understand all specifics of this thesis work.

2.1 Automated object segmentation and classification

Identifying moving objects is one of the first and basic steps in a video processing pipeline. The most intuitive solution — which has been widely adopted and is particularly suitable for scenarios with a static, non-moving camera — is *background modeling* [5]: under the assumption that in the absence of motion each pixel in a video is not subject to strong colour/luminosity variations, moving object detection can be performed by identifying pixels which exhibit a variation with respect to a model of its recent history of colour/intensity values. Unfortunately, in realistic settings, this assumption is only partially valid. Even when a pixel localizes a background area, external factors (e.g., weather, artifical illumination changes, shadows, environmental and camera noise) may strongly affect its appearance on the captured image. In addition, certain moving areas should actually be considered as background, for example trees swaying in the wind or undewater plants moved by currents.

Classic background modeling methods are *density-based*, where background pixel appearance is modeled by a probability density function over pixel colour; commonly employed distributions are Gaussian [6] and Gaussian Mixture Models [7]. More recently, methods based on kernel density estimation [8] or non-parametric ones such as [9] have demonstrated superior performance also in complex dynamic scenes

[10]. In order to reduce the risks of false positives associated to using pixel colour only, recent research [11, 12, 13] has moved towards using a proper combination of visual features (colour, texture, motion). Also, explicitly modeling the characteristics of foreground pixels, as well as the background's, seems to enhance performance as well [8].

However, modeling individual pixels ignores the spatial regularity of the visual world, since it is statistically unlikely that two adjacent pixels have markedly different appearance models. In the last years, a trend towards modeling spatio-temporal uniform (with respect to apperance or motion) regions instead of single pixels has been observed [14, 15]. Spatial uniformity has been usually enforced by segmenting images into *superpixels* (small groups of similar pixels), which can then be used as atomic image units to reduce the amount of computation and simplify the models. Establishing temporal relations between pixels or superpixels in consecutive frames is more complicated and computationally costly. It is typically performed by means of *optical flow* (e.g., [16]), a class of algorithms which attempt at establishing pixel correspondences across two frames. In this thesis, we will take into consideration superpixel-based methods only, due to their simplicity, efficiency and suitability to being integrated with human feedback.

In some scenarios, for example video-surveillance and home alarm systems, identifying moving objects can be the final purpose of a vision system. In most cases, however, this may not be enough: for example, an ecological monitoring system needs to be able identify the species of the living organisms under observation [17].

Understanding what is illustrated in an image is a classic problem in computer vision. Depending on the image type and on the desired detail of the analysis, several variants of this task can be defined, from *image captioning* (generating a description of the content of an image as a natural-language sentence) [18] to *object detection* (localizing istances of specific objects) [19] to *image classification* (assigning a single textual label to the whole image), the last being the task tackled in Chapter 5 and thus the focus of our brief introduction to this class of problems.

Why is image classification a complex problem? First of all, the space of all possible images (even with a fixed dimension) is huge: a 256×256 colour image can be trivially represented as a (roughly) 200,000-dimension vector, but it would be hardly useful for any kind of non-trivial analysis. Therefore, the first problem is to find a more compact representation which retains all distinguishing features from the original image. Although there exist approaches which employ *global descriptors* to encode a whole image's content, such as histograms of the distribution of colour values [20], *local descriptors* (describing small but significant features) have been commonly employed in the last two decades, particularly in combination with a technique called "bag of words", which models an image as the distribution of a set of characteristic local visual features, just as a text document can be described by the distribution of its words [21, 22, 23, 24, 25]. Using a relatively small number of local descriptors to describe a whole image allows to greatly reduce the dimensionality of the problem (typical descriptor sizes are usually below 5,000 elements) and to make the representation more stable to global-level changes such as illumination, translation, rotation, scaling, and so on. The next main problem then becomes *how* to extract local descriptors from an image, i.e., how to choose which are the distinguishing features. Probably the most successful approach used for extracting so-called *keypoints* is SIFT [26], which also defines a representation of such keypoints based on histograms of intensity gradients.

However, half a decade ago the image classification field was revolutionized by the (re-)discovery of Convolutional Neural Networks (CNNs) [1, 2, 27, 28], a technique from a class of algorithms collectively referred to as “deep learning”, dating back to the eighties and recently come back in fashion thanks to the advent of GPUs for general purpose algorithm parallelization and to the availability of large-scale datasets such as ImageNet [29], with over a million images. The main advantage of CNNs over traditional methods is their capability of automatically learning how to detect discriminative and generalizable visual features at different complexity levels, from simple edges to complex structures, thus relieving the system designer of the choice between a large set of manually-crafted features. Currently, CNNs are the state of the art for image classification, as demonstrated by their results in the recent editions of the ImageNet Large-Scale Visual Recognition Challenge [27, 28, 30].

2.2 Interactive methods

Once it became clear that computer vision was not going to be a problem with an easy solution, and that emulating the human vision system still lacked the necessary knowledge of the relevant physiological and cognitive processes, researchers turned to harness human potential in a more straightforward way: put people directly inside the algorithm pipeline. Thus was born the *human-in-the-loop* paradigm, and a new family of *interactive* or semi-supervised methods.

In the beginning, interactive approaches attracted a lot of interest from researchers on image segmentation, and became very popular with the GrabCut [31] algorithm, which paved the way to a whole class of methods [32, 33, 34, 35, 36] based on asking a user to annotate an image in the form of clicks or strokes separating the background from the foreground, and using these annotations as an input to automated segmentation algorithms.

Though less popular or successful than their still-image counterparts, similar interactive approaches were devised for video processing [37, 38, 39], in the attempt to overcome the limitations of automated unsupervised methods, which suffer from oversegmentation in presence of camera motion and occlusions between targets. Most of these methods rely on users to provide input as clicks or strokes, and employ optical flow [40] or models based on Markov chains [41, 42] to make temporal connections between frames; unfortunately, they work well only in simple cases, failing in defining object boundaries precisely. Other approaches pose segmentation as a graph-based or Markov Random Field optimization problem [43, 44], with user input specifying constraints in the cost function. In some works [45], temporal linking between superpixels is done manually: although these methods are able to alleviate human effort for video annotation, at large scale they are ineffective and still too time-consuming.

Another option to support low-level computer vision tasks is to understand how humans perform them and to seek how human inference/reasoning can be integrated into computer programs. Examples include systems which ask people to provide explicitly annotation rationales [46, 47] or to elicit the kinds of visual features employed to discriminate between image/object classes [48, 49, 34].

Besides their effectiveness in solving these tasks, the current main limitation of interactive approaches is that, unlike computers, humans need incentives to carry out a task. Under this scenario, on-line games represent an effective mechanism to involve people in solving challenging problems [50, 51, 52, 53, 33]. Two popular approaches exploiting web games for collecting human feedback for computer vision are the ESP Game [54] and Peekaboom [55]. Both approaches make use of human collective intelligence by having two players collaborate to guess (and thus provide to the system) appropriate labels for images. Since their release, thousands of people have played them, generating millions of annotations. However, these games are devised only for image analysis and, although they have inspired applications to video tagging [56, 57], to the best of our knowledge no similar approaches have been proposed to support video object segmentation.

If gamification has risen as a way to attract users while keeping the same interaction modalities, an alternative to traditional approaches, aimed at alleviating the effort of carrying out the tasks, consists in making the interaction less demanding from the users, in terms of both time and concentration. In this context, eye-tracking techniques fit perfectly as a way to capture human feedback in a simple, non-invasive and non-demanding way. In addition, they also caught our interest as eye gaze is a very reliable indicator of what attracts people attention (hence, what are significant visual features), and at the same time it allows to design bottom-up interaction protocols, without constraining the user with an imposed — and more tiring — task. Eye-tracking also satisfies our "implicitness" requirements: ideally, we can imagine a future with eye-tracking devices incorporated into regular glasses, where scientists would be able to gather huge amounts of data associating human activity and gaze direction without forcing the user to do anything than his or her own regular everyday routines. At present, however, eye-tracking devices are not as portable: commonly-employed video-based trackers are often incorporated into computer screens, and exploit infrared reflection on the cornea to assess gaze direction, after a calibration procedure of the subject.

In science, even before the computer era, eye-tracking had been extensively used to study the relations between eye movements and attention [58]. More recently, they have been used to conduct studies on human–computer interaction [59] or on website attention patterns [60], with applications, for instance, to optimal advertisement placement.

In the computer vision literature, eye-gaze analysis has been typically used for image tagging [61], image indexing/retrieval [62, 63] and image segmentation [64, 65]. Still-image analysis has the advantage of allowing users to look at and “explore” an image for a relatively long time; in a video, things may be more complicated: although common eye-trackers can reach a capture frequency of 60 Hz, this may still be not enough to accumulate a significant number of gaze points on each frame of a normal video played at 25 frames per second, which is one of the challenges we have to deal with in the second part of Chapter 4.

2.3 Human-driven computation systems for visual tasks

Moving along the line of bottom-up user involvement approches applied to computer vision, at its extreme we currently find the class of methods where interaction is as implicit and involuntary as possible, i.e. those based on the analysis of brain activity patterns corresponding to perceptual and cognitive processes.

The idea of reading the mind of people performing specific tasks has been long investigated, especially for building brain–computer interfaces. Most of these studies have targeted EEG-data analysis for detecting the presence of absence of a specific pattern, e.g., P300 detection [66] or seizure detection [67].

The recent deep learning explosion also affected research in brain signal analysis, in particular thanks to a class of models known as Recurrent Neural Networks (RNNs) [68], which are particularly suitable for learning and predicting sequential/temporal patterns as those recorded through EEG. For example, RNNs and CNNs have been used to learn EEG representations for cognitive load classification [69], while a similar approach has been studied to classify EEG tracks evoked when listening to music [70].

As far as vision is concerned, several cognitive neuroscience studies [71, 72, 73] have investigated which parts of human visual cortex and brain are responsible for vision-related cognitive processes, but no clear explanation has emerged yet. At least, it is has been acknowledged that brain activity recordings contain information about visual object categories [74, 75, 76, 77, 78, 79, 80]. However, such evidence has not been fully exploited yet, as few methods have been developed to address the problem of decoding visual information from EEG data [81, 82, 83, 79, 84], and most of these methods focused on binary classification (i.e, presence or absence of a given object class). One of the most recent and comprehensive methods was proposed by Kaneshiro *et al.* [79], who trained a classifier able to distinguish EEG brain signals evoked by twelve different visual object classes, with an accuracy of about 29% that represents the state-of-art performance so far, and the starting point of the research presented in Chapter 5.

CHAPTER

THREE

AUTOMATIC VIDEO OBJECT SEGMENTATION

Our journey towards human-driven computer vision starts with a work which does not envisage direct human involvement, but rather aims at including principles of human perception into the problem of video object segmentation.

3.1 Introduction

The algorithm we propose is inspired by similar approaches based on superpixel segmentation and energy minimization [14], but it aims at efficiency, by avoiding to compute optical flow to identify candidate motion regions, instead analyzing those where significant variations on superpixel segmentation in consecutive frames have been observed. The initial coarse foreground segmentation proposes a set of location priors, which are used as the basis for splitting the task into smaller segmentation problems. Each of these problems is solved by minimizing an energy function which takes into account how combinations of superpixels resemble both foreground/background models and "real-world" objects. Our algorithm aims at emulating the capability of humans to capture the whole from the parts [85] by including constraints on *perceptual organization*, defined by the Gestalt principles of *attachment*, *similarity*, *continuity* and *symmetry*.

The performance evaluation carried out on three standard datasets shows that the proposed approach: 1) is able to deal with complex scenes, with several non-rigid objects undergoing sudden appearance changes, and with fast varying and multimodal backgrounds; 2) is able to generalize over different object classes since no offline training or *a priori* knowledge is required; 3) outperforms existing and more powerful video object segmentation approaches, e.g., [9, 14]; 4) achieves encouraging results on challenging datasets such as SegTrack [86], Underwater Dataset [10], I2R [87].

3.2 Overview

The basic principles which led the design of the algorithm are the following:

- *Superpixels as segmentation units*: Working with pixels is susceptible to noise and fuzzy region boundaries, besides being in general more computationally expensive as the number of elements to analyze becomes very large. Superpixels extraction, instead, allows to greatly simplify the problem, both in terms of formulation and from a computational perspective.

- *Objects as superpixel aggregations*: Since superpixels typically tend to largely oversegment an image, we can assume that object boundaries always correspond to superpixel boundaries — i.e. no superpixel spans two objects. Therefore, as superpixel segmentation already guarantees a fairly robust boundary detection, we can formulate the segmentation task as the identification of connected foreground and background superpixels.

- *Motion superpixels as location priors*: Assuming that static video regions produce no *motion superpixels*, defined as superpixels on which we detect motion activity in two consecutive frames, we can limit our analysis to areas where motion superpixels aggregate, and process them independently as several subtasks, which is more efficient than performing a global segmentation on all superpixels and yields better results.

- *Appearance similarity*: By managing foreground and background models, we are able to know what objects look like in terms of color. Therefore, the segmentation algorithm should try and keep similar superpixels together.

- *Perceptual organization*: Objects in the real world have a generally regular and compact geometrical structure, according to the Gestalt principles of *attachment*, *similarity*, *continuity* and *symmetry*. Enforcing such principles in the way superpixels are combined together can help to compute segments which are more likely to match the actual objects in the scene.

Based on these criteria, our algorithm consists of the following steps:

Initial foreground estimation. In this phase, *motion regions*, defined as the bounding boxes (expanded by $D_{\text{pad}} = 3$ pixels) around connected groups of *motion superpixels*, are identified. Unlike previous methods [14], this preliminary segmentation is carried out without computing optical flow, but simply analyzing superpixel segmentation changes in consecutive frames (see Figure 3.1). Ideally, in two consecutive frames, superpixel segmentation changes only in areas with moving objects. This gives a straightforward condition to rapidly identify foreground but, practically, background object movements and light changes may generate false positives that need to be removed. Each motion region is then treated as a single optimization problem for the subsequent accurate object segmentation step.

Background/foreground models estimation. Usually background subtraction approaches maintain a model for each background pixel, which is initialized in an off-line phase where only background frames are taken into account and then updated using the classification map. In our approach, we do not build a background model for each pixel; instead, we have an on-line model for each background region and each foreground object.

Accurate object segmentation. The goal of this step is to accurately identify object boundaries by refining the segmentation initialized by the motion regions in order to enforce spatial smoothness. To obtain an accurate object segmentation, we group superpixels by optimizing an energy function which includes appearance similarity to the background/foreground models and perceptual organization principles. This energy minimization process is done for each detected motion region, as opposed to global minimization approaches [14] (see Figure 3.3). We do not impose any constraints on motion smoothness (unlike [14, 15]) since it makes the entire process too dependent on the frame rate of the analyzed videos.

3.3 Method

3.3.1 Initial foreground segmentation

Our approach starts with superpixel segmentation carried out by means of SLIC [88], which is an efficient adaptation of k-means in the *labxy* image space for robust superpixel generation. This step operates on pairs of consecutive frames $(t, t+1)$ and identifies motion regions based on the consideration that superpixel segmentation, in two subsequent frames, remains more or less stable in background regions, while it changes substantially in the case of moving objects.

Let S_t and S_{t+1} be the sets of the superpixels computed, respectively, at frame t and $t+1$. For each superpixel $s^i_{t+1} \in S_{t+1}$ we compute the Jaccard distances (d_J) between its backprojection at time t $(s^i_{t+1 \to t})$ and all the superpixels in S_t. If the minimum of such distances is above a threshold, we mark the superpixel as “motion superpixel” (see Figure 3.2). Therefore, the initial foreground mask M^{t+1} at time

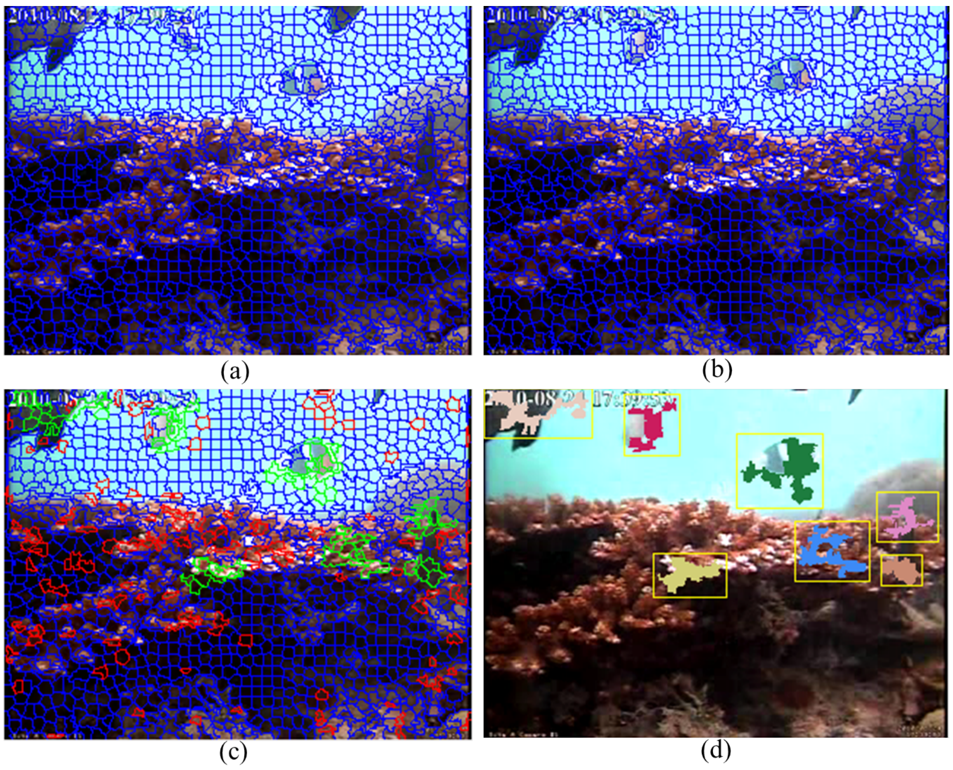

Figure 3.1: *(a) and (b) two input frames. (c) Set of motion superpixels: false positives — filtered out at this stage — are shown in red, while correct ones are shown in green. (d) Motion regions built according to the filtered motion superpixels. In each motion region, we then perform accurate segmentation by energy minimization.*

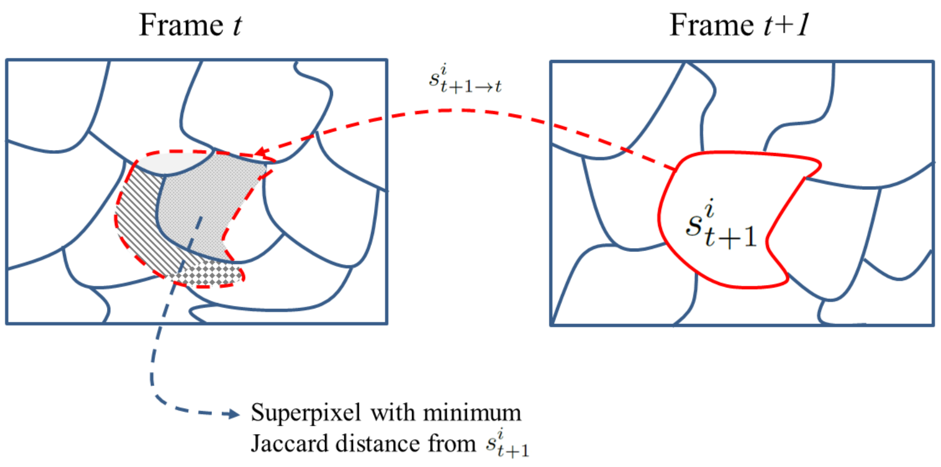

Figure 3.2: *Example of motion superpixel identification: superpixel s^i_{t+1} at frame $t+1$ is backprojected on frame t, overlapping four superpixels. s^i_{t+1} is marked as "motion superpixel" if the Jaccard distance between it and the superpixel with the highest overlap (i.e. minimum Jaccard distance) is above threshold T.*

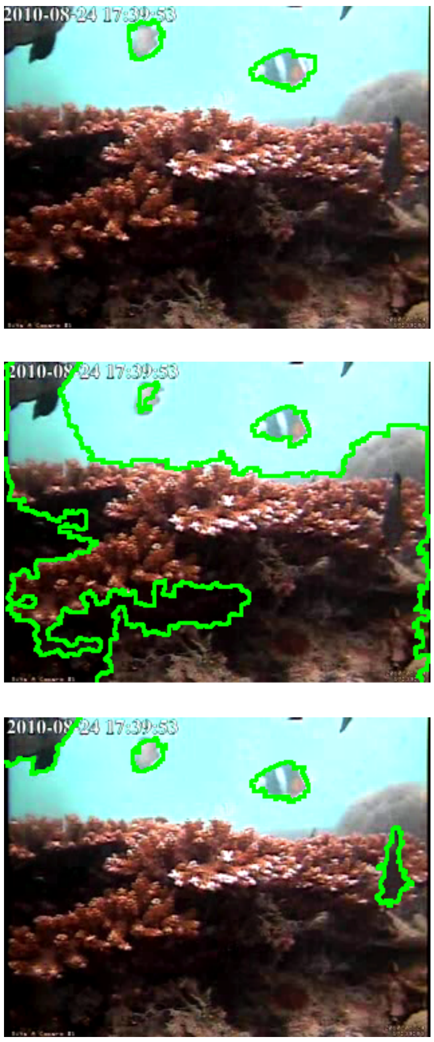

Figure 3.3: *(a) Output mask obtained by [14] performing energy minimization of the whole image; (b) Output mask of our method when excluding location priors, thus performing energy minimization taking into account all image superpixels; (c) Output mask of our approach with location priors.*

$t+1$ is given by:

$$M_{s_{t+1}^i}^{t+1} = \begin{cases} 1 & \text{if } \min_{s \in S_t} d_J(s_{t+1\to t}^i, s) > T \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

The threshold T is adaptively computed as the average of the minimum Jaccard distance between all superpixels in frame $t+1$: this allows to handle, even in this early stage, slow object motion (T will be low, enabling the detection of fine superpixel variations) and camera motion (T will be high, and many superpixels with apparent motion will be filtered). To further remove false positives (red-coloured superpixels in Figure 3.1), isolated motion superpixels or small groups of connected superpixels are discarded. Moreover, as soon as the background/foreground models become reliable (after three frames; see Section 3.3.2) they are used to remove background superpixels misclassified as motion ones, by fitting a Mixture of Gaussians (MoG) to each superpixel and computing the Kullback-Leibler divergence from the background/foreground models.

3.3.2 Background/foreground model estimation

In order to include constraints on the visual appearance of the objects in the scene, we maintain a set of background and foreground color models. Using several models for background and foreground, instead of only one each (as in [14]), allows to handle appearance multimodality: this is critical when several moving objects are present in the scene, so that each can be modeled and matched independently. It is important to understand that, although in this section we will describe the construction of the models from a pixel-based point of view (and necessarily so, due to the nature of colour features), all other parts of the proposed method (from the identification of motion superpixels to segmentation as a minimization problem, in the next section) deal with superpixels as a basic and atomic unit.

Model initialization is performed at the first processable frame (i.e. the second video frame, when the first motion superpixel segmentation is available) by fitting a set of mixtures of Gaussians[1] to each *background region* and *foreground region*: background regions are obtained using adaptive k-means clustering on the whole image excluding motion superpixels and very small clusters, whereas foreground regions simply consist in connected sets of motion superpixels, ideally associated to each moving object in the scene. Having separate foreground models rather than a single global one allows us to have simpler models (i.e., with a smaller number of components in the mixture) and to avoid "contamination" between models associated to different objects.

After model initialization is performed, we have a set of background models $\{\psi_{b,1}, \psi_{b,2}, \dots, \psi_{b,N_b}\}$ and one of foreground models $\{\psi_{f,1}, \psi_{f,2}, \dots, \psi_{f,N_f}\}$, where N_b is the number of clusters obtained from the adaptive k-means on the background pixels and N_f is the number of foreground regions from the initial motion superpixel segmentation.

Background model re-initialization is performed at the second processable frame (i.e., the third video frame, when the first object segmentation is available) and after every T_{init} frames, since as time passes scene conditions may change and the models may become outdated: this happens, for example, if foreground regions become stable and

[1]The number of components is adaptively set by minimizing the Akaike information criterion.

are absorbed into the background, or if new moving objects appear. Moreover, the need for re-initializing the background model at the second processable frame comes from the fact that the initial models are based on the inaccurate segmentation provided when using motion superpixels only, whereas at this point we can use the accurate object segmentation map for the previous frame to separate background and foreground regions.

Model update is performed at every frame (except when the model is re-initialized) after segmentation is completed. The update process for the background models at frame t consists of the following steps:

1. Initialize sets $P_{b,1} = \emptyset$, $\dots$, $P_{b,N_b} = \emptyset$, representing the sets of pixel values (as RGB triplets) which will be used to update the corresponding background model.
2. Put into each $P_{b,i}$ all pixel values from frame $t-1$ which had been associated to background model $\psi_{b,i}$.
3. Compute background model priors $\pi_1, \dots, \pi_{N_b}$ as the ratios between the number of pixels belonging to each model and the total number of background pixels at frame $t-1$.
4. For each pixel p belonging to superpixels labeled as background, add it to set $P_{b,i}$ according to a maximum-a-posteriori criterion, i.e. such that: $i = \arg\max_j P(\psi_{b,j}|p)$.
5. Fit a MoG ψ_{b_i} from each set $P_{b,i}$, using the current models as initial conditions for the expectation-maximization algorithm.
6. Remove models $\psi_{b,i}$ from the model set if $P_{b,i} = \emptyset$.

Using also pixels from the previous frame to fit the models (item 2) helps to prevent problems with the fitting algorithm when the initial conditions are too different from the target data set.

Foreground models are updated on a per-object basis, as follows:

1. Initialize sets $P_{f,1}$, ..., P_{f,N_f}, similarly as above.
2. For each foreground object O_i segmented at frame t which contains at least one motion superpixel (see Section 3.3.3), fit a MoG Γ_i on the object's pixels.
3. Identify the foreground model $\psi_{f,j}$ which best matches O_i using the Kullback-Leibler (KL) divergence: $j = \arg\min_k d_{\text{KL}}(\psi_{f,k}, \Gamma_i)$
4. If the KL divergence between $\psi_{f,j}$ and Γ_i is smaller than a threshold T_{fg}, add O_i's pixels to $P_{f,j}$. Otherwise, create a new set P_{f,N_f+1} containing O_i's pixels, and increase N_f by 1.
5. Fit a MoG ψ_{f_i} from each set $P_{f,i}$, similarly as above.
6. Remove models $\psi_{f,i}$ from the model set if it has matched no objects for the past T_{f} frames.

3.3.3 Accurate object segmentation

The initial segmentation based on motion superpixels is not accurate enough, as it does not take into account any information on visual appearance or on how well a set of superpixels geometrically fit together, as shown in Figure 3.1. Nevertheless, motion regions provide initial location priors for accurate segmentation based on appearance similarity and perceptual organization. These location priors are combined with the latest foreground map to allow segmenting objects which become temporarily stationary. However, in order to avoid a self-feeding effect on background regions incorrectly identified as foreground, and to let the algorithm "forget" foreground regions which are absorbed into the background, foreground models for appearance are updated only from superpixels belonging to regions which originally contained motion superpixels.

Then, for each motion region, a local segmentation subtask is defined by taking into account also non-motion superpixels intersecting the region's bounding box. Depending on the size of the object, the number of superpixels involved in each subtask is relatively small (in the order of the tens), which allows to solve the problem efficiently. If several motion regions intersect, we join them into a unique region. After that, considering each subtask independently, we pose the segmentation task as an energy minimization problem, where higher segmentation costs are due when the algorithm assigns different labels to similar contiguous superpixels or to contiguous superpixels which perceptually fit together. Formally, given the set of superpixels $S = \{s_1, \dots, s_N\}$ and a set of corresponding labels $\mathcal{L} = \{l_1, \dots, l_N\}$, where each $l_i \in \{0 : \text{background}, 1 : \text{foreground}\}$, the overall energy function is as follows:

$$E(\mathcal{L}) = A(\mathcal{L}) + P(\mathcal{L}) \tag{3.2}$$

$$A(\mathcal{L}) = \sum_{l_i \in \mathcal{L}} a_1(l_i) + \sum_{(l_i,l_j) \in \mathcal{N}(\mathcal{L},S)} a_2(l_i, l_j) \tag{3.3}$$

$$P(\mathcal{L}) = \sum_{(l_i,l_j) \in \mathcal{N}(\mathcal{L},S)} p(l_i, l_j) \tag{3.4}$$

where $A(\mathcal{L})$ and $P(\mathcal{L})$ respectively represent the overall appearance and perceptual organization energies, $\mathcal{N}(\mathcal{L}, S)$ is the set of all pairs of neighbor superpixels (i.e., with part of boundary in common), and the potentials $a_1(\cdot)$, $a_2(\cdot, \cdot)$ and $p(\cdot, \cdot)$ enforce our design principles on visual similarity and perceptual organization. As shown below, these potentials are defined so that $E(\mathcal{L})$ is a binary pairwise function with sub-modular pairwise potentials, thus efficiently minimizable using graph cuts in order to obtain the final segmentation:

$$\overline{\mathcal{L}} = \arg\min_{\mathcal{L}} E(\mathcal{L}) \tag{3.5}$$

In the following, each potential function is described in detail.

Background/foreground similarity

The unary potential $a_1(\cdot)$ indicates whether a superpixel is best associated to the foreground or the background. Given superpixel $s_i = \{p_1, \ldots, p_n\}$, let us assume we want to compute the cost of assigning label 0 (i.e., background) to s_i. For each pixel $p_j \in s_i$ and for each background model $\psi_{b,k}$, we compute the posterior probability $P(\psi_{b,k}|p_j)$. We then average these probabilities for each background model and choose the maximum among the averages as the overall background probability P_b for s_i; the negative log-posterior is then used as value for $a_1(0)$ (since we are considering the background case).

If l_i is 1 (foreground), the prior for model $\psi_{f,k}$ is $c \cdot \frac{1}{t_k + N_f}$, where t_k denotes how many frames ago the model was last updated and c is a normalization factor.

Mathematically, the overall formula can be written as:

$$a_1(l_i) = -\log \max \left\{ \frac{1}{|s_i|} \sum_{p_j \in s_i} P(\psi_{x,k}|p_j) \right\}_{k=1 \ldots N_x} \tag{3.6}$$

where $|s_i|$ is the number of pixels in s_i and the pair $(\psi_{x,k}, N_x)$ depends on l_i:

$$(\psi_{x,k}, N_x) = \begin{cases} (\psi_{b,k}, N_b) & \text{if } l_i = 0 \\ (\psi_{f,k}, N_f) & \text{if } l_i = 1 \end{cases} \tag{3.7}$$

Local similarity

The binary potential $a_2(\cdot,\cdot)$ defines the cost of assigning different labels to two neighbor superpixels, based on their color similarity. Our approach on estimating this quantity is based on the following consideration: the similarity of two superpixels can be seen as the probability that their union is generated by the same color distribution, be it a background or a foreground one; if they are not similar, their union will be unlikely to be generated by any background/foreground model. Thus, given superpixels s_i and s_j, we fit a MoG Γ_{ij} from the pixels belonging to $s_i \cup s_j$, then compute the minimum KL divergence between Γ_{ij} and all background and foreground models, and use it as a dissimilarity measure between s_i and s_j; in order to guarantee submodularity [89], the final value of potential $a_2(l_i, l_j)$ is non-zero only if $l_i \neq l_j$.

Formally, the potential function is:

$$a_2(l_i, l_j) = [l_i \neq l_j] \left[1 - \min \left\{ d_{\mathrm{KL}}(\Gamma_{ij}, \psi) \right\}_{\psi \in \Psi} \right] \tag{3.8}$$

where $[l_i \neq l_j]$ is 1 if the labels are different and 0 otherwise, $d_{\mathrm{KL}}(\cdot)$ is the KL divergence function, and $\Psi = \left\{ \psi_{b,1}, \dots, \psi_{b,N_b}, \psi_{f,1}, \dots, \psi_{f,N_f} \right\}$ is the set of all background and foreground models. Comparing the superpixels' union to the background/foreground models helps to prevent problems when fitting the pixel distribution to a MoG, since superpixels, by construction, are small and internally homogenous. Sometimes, when pixels from both superpixels are almost identical and some color channels are practically constant, it is impossible to compute Γ_{ij}: in such cases, we set $a_2(l_i, l_j) = [l_i \neq l_j]$, which reflects the high similarity between the two superpixels. The reduced number of neighbor pairs (due to the small number of superpixels in each segmentation subtask) and the small number of pixels in each superpixel makes the evaluation of $a_2(\cdot, \cdot)$ very fast, in spite of the number of models to build.

Perceptual organization

The binary potential $p(\cdot, \cdot)$ defines the cost of assigning different labels to two neighbor superpixels, based on how well they fit together from a perceptual and geometrical point of view. To estimate this quantity, we employ a variant of the approach proposed by [85]. The potential function is computed as:

$$p(l_i, l_j) = [l_i \neq l_j] e^{-\theta \cdot [B(s_i, s_j), C(s_i, s_j)]} \tag{3.9}$$

where $\theta = [18, 3.5]$ is a weighing vector (suggested in [85]), $B(s_i, s_j)$ is the *boundary complexity* of region $s_i \cup s_j$, and $C(s_i, s_j)$ is the *cohesiveness* between superpixels s_i and s_j.

Boundary complexity measures the regularity of the contour obtained by joining two superpixels: intuitively, if they belong to the same object, the contour of their union should be ideally as smooth as if it were a single object in the first place; similarly, if the contour of the union is not regular, it is less likely that they belong to the same image segment. In order to numerically encode this principle, an analysis of convexity and of the number of notches (non-convex angles) on the contour is performed: as we compute boundary complexity in the same way as in [85], we refer the reader to that paper for details.

Cohesiveness also measures how well two superpixels fit to each other, but is defined according to principles of *symmetry*, *continuity*, and *attachment strength*. If the superpixels' sizes are similar (i.e. their sizes' ratio is smaller than 3), it is computed as:

$$C(s_i, s_j) = \lambda_{ij}(\phi_{ij} + \varphi_{ij}) \tag{3.10}$$

The symmetry score, ϕ_{ij}, evaluates whether the centers of mass of s_i and s_j are aligned vertically (same x coordinates) or horizontally (same y coordinates). If we define (x_i, y_i) and (x_j, y_j) to be the centers of mass of superpixels s_i and s_j respectively, the symmetry score is computed as:

$$\phi_{ij} = \min\{|x_i - x_j|, 1\} \cdot \min\{|y_i - y_j|, 1\} \tag{3.11}$$

which will return a small value if the differences between either pair of corresponding coordinates is close to zero.

The continuity property indicates whether the line along which s_i and s_j's common boundary is oriented does not intersect either object at any other points. When this condition is verified, the union of the two superpixels yields an object with a perceptual impression of "continuity", in the sense that it is not evident that it is made up of two distinct regions. The corresponding score, φ_{ij}, is defined as:

$$\varphi_{ij} = \begin{cases} 0 & \text{if } e(\partial ij) \cap \partial i = \emptyset \wedge e(\partial ij) \cap \partial j = \emptyset \\ 1 & \text{otherwise} \end{cases} \tag{3.12}$$

where ∂i is s_i's contour, ∂j is s_j's contour, ∂ij is the common boundary, and $e(\partial ij)$ is the portion of the line passing by the extrema of the common boundary, excluding the segment between the extrema.

Attachment strength depends on how large the common boundary is with respect to the superpixels' whole boundaries:

$$\lambda_{ij} = \beta e^{-\alpha \frac{L(\partial ij)}{L(\partial i) + L(\partial j)}} \tag{3.13}$$

where $\beta = 3$ and $\alpha = 20$ are two parameters (again, suggested in [85]), and $L(\cdot)$ returns the length in pixels of a boundary. In other words, if two objects are "well-attached" (imagine two halves of a disk), the length of the common boundary should be large; similarly, if the attachment is weak (imagine two tangent circles), the length of common boundary will be very small in comparison to the superpixels' contour lengths.

In the particular case when one superpixel (say, s_i) is markedly larger than the other (s_j), symmetry and continuity may not be meaningful. Therefore, our cohesiveness score for these situations becomes:

$$C'(s_i, s_j) = \lambda'_{ij} = \beta e^{-\alpha \frac{L(\partial ij)}{L(\partial j)}} \tag{3.14}$$

that is, we only evaluate attachment strength on the smallest superpixel only.

Once all potentials in the energy function are defined, we can perform graph cut-based minimization to find the optimal segmentation. However, since each segmentation subtask is performed locally on a small set of superpixels, it may happen that the region we are analyzing is included into a large object, only a part of which was initially detected by the motion superpixels[2]. Therefore, the above approach is iteratively applied until no changes are detected in consecutive iterations, both to capture large objects and to refine the obtained masks. At each iteration, we perform motion region–based segmentation (as

[2]Of course, the opposite case is not a problem: if a set of connected motion superpixels span a much larger area than the actual object, the excess part will be segmented out by the energy minimization phase.

above) with the difference that the object blobs detected at previous iterations are now considered as single large superpixels (hard-constrained to be labeled as foreground), thus allowing to iteratively refine object segmentation at a low processing cost, since all superpixels merged into blobs in previous iterations do not need to be processed again.

3.4 Experimental results

In this section we present qualitative and quantitative results of our approach on three datasets — the Underwater dataset [10], SegTrack v2 [86], and I2R [87] — to show how our method performs in cases of slow motion, camera motion, small objects and cluttered scenes. The parameters T_{init}, T_{fg} and T_{f} are set, respectively, to 15, 0.8 and 10 for all the employed datasets. Superpixel size was set to 7×7, as a compromise between the risk of segmentation errors, sensitivity of threshold T to noise, and processing speed. As [14] is also based on superpixel segmentation, but employs optical flow, it was used as the main baseline in all evaluations, using the public source code with default parameters.

3.4.1 Underwater Dataset

The underwater dataset is a collection of 14 “real-life” underwater videos (10-minute videos with spatial resolution from 320×240 to 640×480, at 5 *fps*) taken with static cameras to monitor Taiwan coral reef, and is featured by small objects and cluttered scenes. The videos are classified into seven different classes: *Blurred* (low-contrast scenes with well-separated background and foreground), *Complex Background* (background featuring complex textures, thus suitable to test superpixel-based methods), *Crowded* (highly cluttered scenes with several occlusions), *Dynamic Background* (background movements, e.g., due to plants), *Luminosity Change* (abrupt light changes), *Hybrid* (plant movements together with luminosity changes), *Camouflage Foreground Object* (e.g., objects very similar to the background). The dataset provides also ground-truth, consisting of about 20 frames per video segmented at pixel level. We compared our method to some background modeling state-of-the-art approaches [8, 90, 9, 10] and also included the well-known Gaussian Mixture Model [7] as baseline. For these methods we report their performance as stated in [10] where the original implementations (provided by the respective authors) were used, thus avoiding implementation bias in the performance analysis. The evaluations in terms of F-measure scores (computed at pixel-level) are shown in Table 3.1: on average, our method outperformed the other approaches in all videos, achieving good results in handling light changes, deformable objects and cluttered scenes. Figure 3.3 shows a qualitative comparison between our method and [14]; it is possible to notice how our method was able to identify objects hidden in background areas (see the fish on the right side in Figure 3.3) while [14] missed them.

3.4.2 SegTrack Dataset

SegTrack [86], originally built for testing tracking algorithms, has been widely employed as a video object segmentation benchmark [91]. It contains six videos (*monkeydog*, *girl*, *birdfall*, *parachute*, *cheetah*, *pen-*

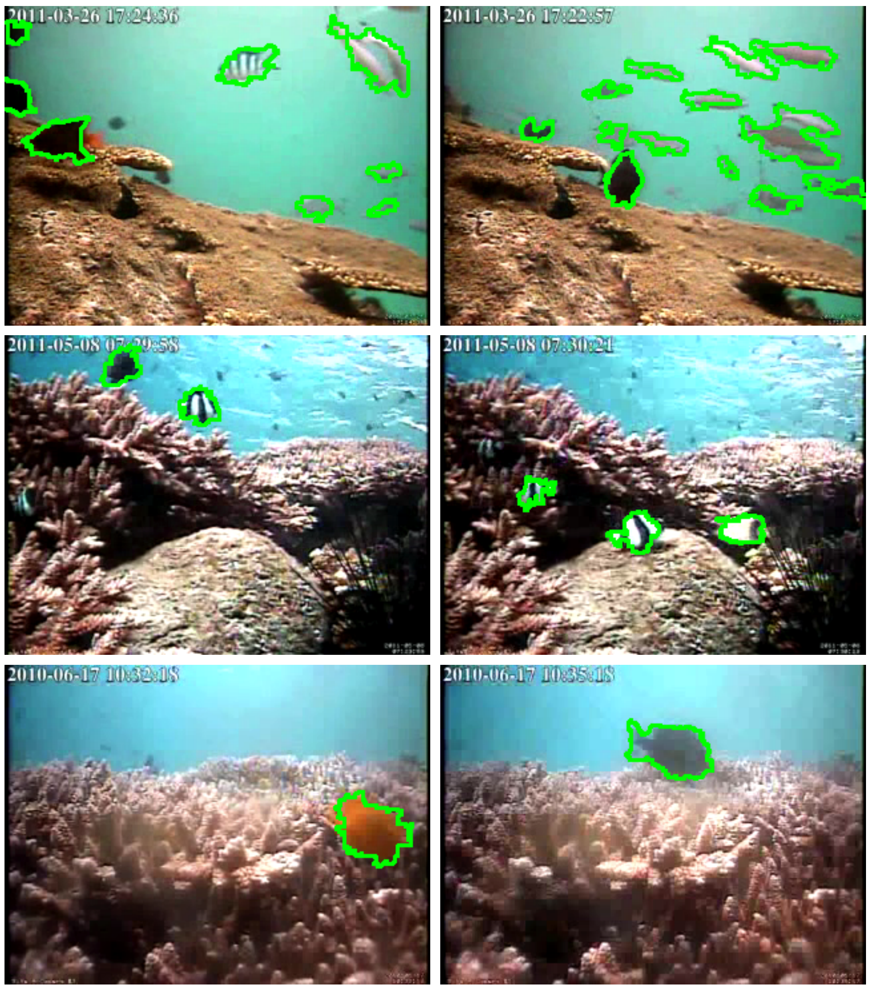

Figure 3.4: **Example results for Underwater Dataset**. *Each row shows two images from a video, with our final mask superimposed in green. From top to bottom: Crowded, Dynamic, Luminosity.*

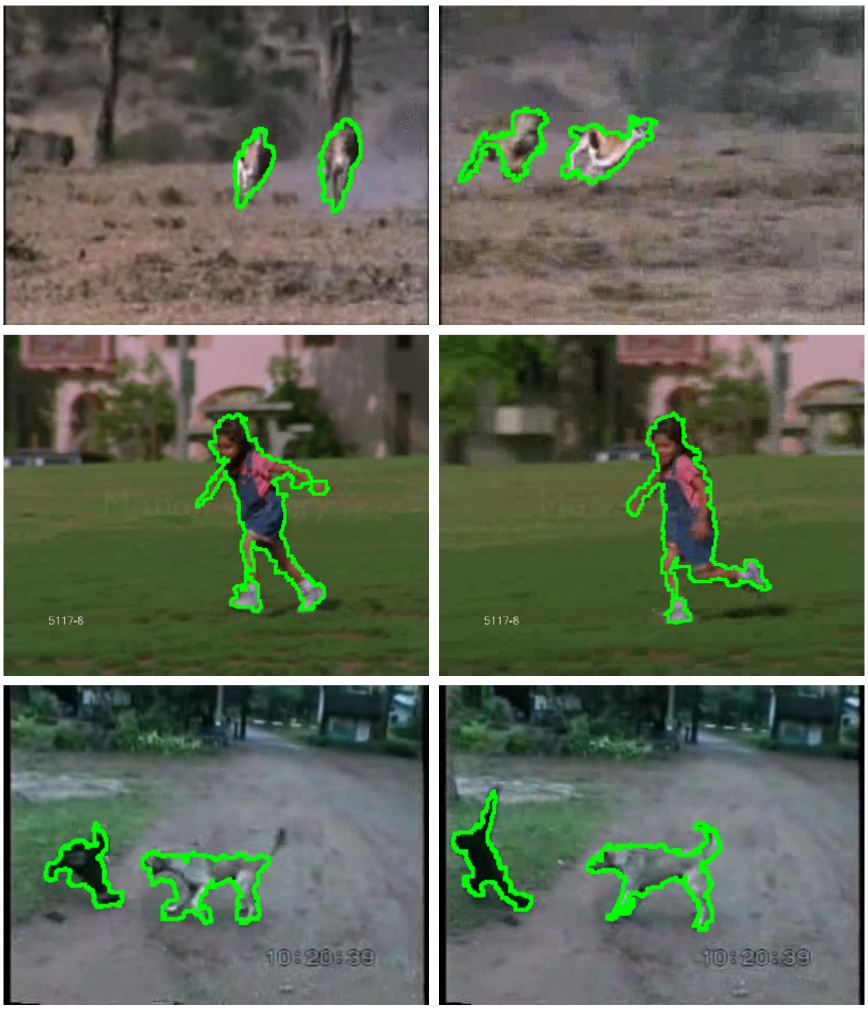

Figure 3.5: **Example results for SegTrack**. *Each row shows two images from a video, with our final mask superimposed in green. From top to bottom: cheetah, girl, monkeydog.*

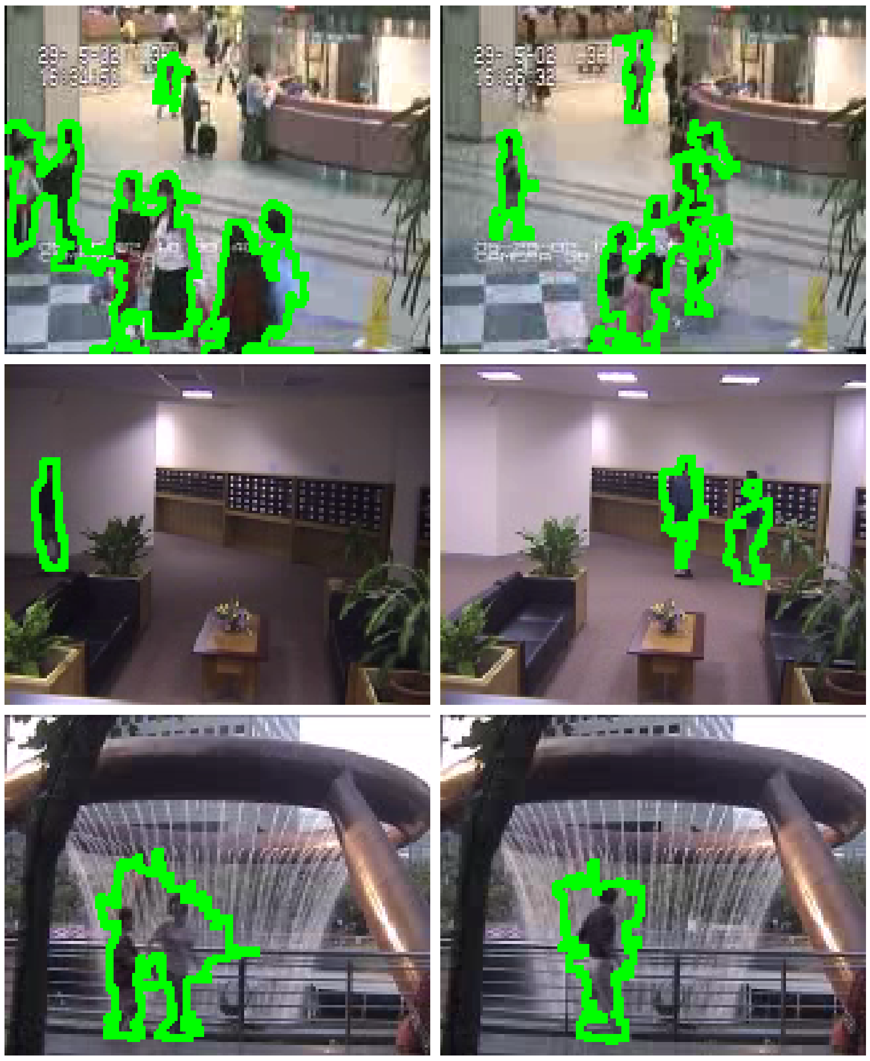

Figure 3.6: **Example results for I2R**. *Each row shows two images from a video, with our final mask superimposed in green. From top to bottom: AirportHall, Lobby, Fountain.*

Class	[14]	[7]	[90]	[9]	[10]	*Our method*
Blurred	35.1	83.3	70.3	85.1	93.3	89.8
Complex	36.1	67.0	83.7	74.2	81.8	86.3
Crowded	73.7	85.2	79.8	84.6	84.2	84.2
Dynamic	18.6	62.0	77.5	67.0	75.6	83.7
Hybrid	5.5	62.7	72.2	79.8	82.6	88.9
Luminosity	53.1	63.1	82.7	70.4	73.0	89.6
Camouflage	18.4	66.3	73.5	76.3	82.2	85.7
Avg	34.3	69.9	77.1	76.7	81.8	86.9
Std	23.2	9.2	4.9	6.4	6.0	2.4

Table 3.1: **Results on the Underwater dataset**. *F-measure scores (in percentage) for different methods on the Underwater dataset. Our method is very robust to light changes and background movements (see rows 4 and 6).*

guin) and the ground truth provides pixel-level foreground object annotations for each video frame. The dataset is known for being very challenging due to camera motion, slow object motion, object-background similarity, non-rigid deformations and articulated objects. We compared our method to [91, 92, 38, 9] and reported their performance as stated in [14]. Table 3.2 shows the achieved performance as the average number of misclassified pixels per frame. Our method performed remarkably well when compared to the other methods, especially on the *girl* video where our method shows its ability to segment articulated objects. In fact, we were able to segment very well

	[14]	[91]	[92]	[38]	[9]	*Our method*
Birdfall	217	288	155	468	606	278
Cheetah	890	905	633	1968	11210	824
Girl	3859	1785	1488	7595	26409	1029
Monkey	284	521	365	1434	12662	192
Parachute	855	201	220	1113	40251	251

Table 3.2: **Results on SegTrack**. *The penguin video was discarded since the annotations provided in the ground truth were not reliable as only one penguin in a group of penguins was segmented.*

also legs and arms, which were missed by [14].

3.4.3 I2R Dataset

The last evaluation was carried out on the I2R Dataset [87], which contains nine videos (at 120×160 resolution) taken with static cameras showing people in different indoor and outdoor scenes. This dataset is commonly employed for testing video object segmentation approaches and presents several challenges including slow motion, cluttered scenes, non-rigid deformations, articulated objects, camouflage. The ground truth consists of 20 labeled frames (at pixel-level) per video. Table 3.3 compares the F-measure scores of our method to the ones achieved by the recent background modeling approaches [12], [93], [10] that, similarly to our approach, model (at pixel-level) background and foreground and use combination of visual cues including texture. Our method outperformed all the other methods on the I2R dataset, especially on crowded scenes (e.g. *AirportHall*) and with articulated objects (e.g. *Escalator*). The high performance obtained on the *Escalator* class is remarkable given the presence of many occlusions.

Figures 3.4, 3.5, 3.6 show some example results where it is possible to appreciate the capability of our approach to adapt to different complex scenes (e.g., with very sudden light changes, as in Figure 3.4) and targets (from highly deformable ones, e.g., fish, to articulated ones, e.g., girl) without performance loss.

We believe that including perceptual organization constraints into the method has effectively boosted its performance: as further confirmation, the overall segmentation accuracy decreased by more than 30% when excluding the $P(\mathcal{L})$ term in Eq. 3.2.

3.4.4 Processing times

Our method takes on average 0.2 seconds per frame on the Underwater and SegTrack datasets (image resolution about 320×240) and 0.05 sec/frame on the I2R dataset (image resolution of 160×120), which is fast enough to be used for "on-the-fly" video processing. This is remarkable given that [14] takes 0.5 sec/frame on the SegTrack dataset without considering optical flow and superpixel processing times, that take processing time to about 3 seconds per frame. The reason of the increased speed of our approach is mainly due to modeling/classifying superpixels instead on single pixels and to local energy minimization.

Class	[14]	[12]	[93]	[10]	*Our Method*
AirportHall	29.6	68.0	71.3	69.2	77.4
Bootstrap	17.9	72.9	76.9	76.5	81.0
Curtain	23.2	92.4	94.1	94.9	96.3
Escalator	26.1	68.7	49.4	72.0	84.8
Fountain	15.1	85.0	86.0	83.2	84.1
ShoppingMall	13.1	79.7	83.0	78.5	86.7
Lobby	5.0	79.2	60.8	66.3	82.5
Trees	21.6	67.8	87.9	81.9	89.0
WaterSurface	83.7	83.2	92.6	92.5	93.9
Avg	26.1	77.4	78.0	79.5	86.2
Std	24.3	8.2	14.2	9.3	5.7

Table 3.3: **Results on I2R**. *F-measure scores (in percentage) for different methods on the I2R dataset. Our method outperforms all the reported methods, especially on the Escalator class.*

Of course, faster methods exist, e.g. [10] achieving 0.05 sec/frame on the Underwater dataset (although it relied on a C++ implementation, while our method is currently written in Matlab 2013a), but we believe that our method shows a good speed/accuracy trade-off.

3.4.5 Discussion

Automatic video object segmentation is still a complex and unsolved problem, due to the variety of contexts and scenarios. In this chapter, we present our initial attempt at solving the problem without constraints and the specific application settings, by extending state-of-the-art approaches and combining them with principles of human perceptual organization. The result is a method which achieves a satisfactory trade-off between computation time and accuracy, thanks to our initial foreground estimation based on superpixel differences, rather than the less efficient optical flow computation. The energy-based setting of the problem allowed to easily integrate perceptual organization constraints into the solution without altering the general framework, which had proved effective in the literature.

Of course, human involvement in this work is practically absent, and all analysis is limited to bare visual processing. In the next chapter, we will see how this approach can be modified to include humans in the process and how the provided feedback can be used to guide and enhance automatic analysis.

3.5 Publications

The approach described in this paper was presented at the Computer Vision and Pattern Recognition conference in Boston, USA, 2015 [94].

CHAPTER

FOUR

GAMIFICATION VERSUS EYE-TRACKING FOR INTERACTIVE VIDEO OBJECT SEGMENTATION

In the previous chapter we introduced the problem of automatic video object segmentation and illustrated an algorithm combining a state-of-the-art approach based on energy minimization with principles of perceptual organization, as performed by humans according to Gestalt laws [85].

It is time to see how humans can concretely be involved in a computer vision task, and how the information they provide can enhance the performance of automatic algorithms. In this chapter we will compare the performance of two different strategies of human involvement: gamification and eye-tracking–based participation. These two approaches share the nature of the feedback that they produce — punctual screen coordinates — but are very dissimilar in many important ways.

As mentioned in Chapter 2, gamification — the process of "disguising" a job as an entertaining activity, in order to provide an incentive to potential participants — is one of the most successful approaches for human involvement in computer vision tasks. However, it is still a rather explicit interaction modality: users have to dedicate their time to playing the game, and as entertaining as this may sound, a less "invasive" involvement strategy may be desirable. Moreover, gamification is extremely task-oriented: the design of the game and the kind of generated human feedback are directly mapped to a set of rules or instructions imposed to the user.

Instead, eye-tracking technologies can alleviate the effort demanded from participants, since "watching" is in general a less intense activity than "playing". Also, leaving the user's attention focus on objects or regions according to bottom-up saliency allows to draw from a well of information inaccessible to explicit approaches, where user attention is guided and constrained by what the system designer wants to obtain.

4.1 Gamification-based video object segmentation

A straightforward way to involve users into the process of identifying moving objects in videos is to have them click, using a computer mouse, on the relevant regions. We can then turn this task into a game by adding a *scoring system* associated to click locations and make the clicking process non-trivial, e.g., by increasing the videos' playing speed. This is the basic idea for the gamification approach described in this chapter: although clicking on moving regions can be a boring task per se, the simplicity of the game and the addition of the competitiveness factor make it attractive to occasional users, who are actually more motivated by playing against their friends than by contributing to the advancement of academic research.

Unfortunately, making a game popular and involving a lot of users is only an aspect of the whole problem. First of all, the quality of the generated feedback data is questionable: sometimes users only play half-mindedly, which may result in either too little clicking (not enough data provided) or too much (higher risk of noise); in general, they do not care about the accuracy of their clicks, unless the scoring system forces them to — which gives an idea of the importance of choosing a scoring system embodying all required constraints.

Finally, it may be tricky to translate human feedback into algorithmic input. For example, how much should the algorithm trust the correctness of the input and how can we modify existing approaches to deal with the additional information? In the following, we will describe a variant of the energy minimization–based approach from Chapter 3, and add terms in the energy potentials which take into consideration which superpixels have been the target of user clicks, and whether being a target resulted from intentional and accurate clicking or from noise and imprecision.

The interactive video object segmentation approach associated to our gamification strategy can be seen as a two-step spatio-temporal optimization problem: the first one with a cost function exploiting spatial information at the frame level and encoding user feedback and appearance cues in order to extract homogeneous object regions; and the second one enforcing spatio-temporal consistency between the segmented object regions in consecutive frames, thus refining the preliminary segmentation.

There are three main modules in the whole approach:

- *The game*: The starting point of the whole process, our game is thought to gather user clicks in correspondence of objects of interest in videos. The game is designed to be challenging and competitive, so that users are encouraged to play: while this helps keeping the competition between users, game difficulty often affects the noisiness of the generated data.

- *Superclick extraction*: The initial stage of our algorithm converts the noisy set of clicks into a set of more accurate "clicked superpixels", or *superclicks*. Posing the problem in terms of superpixels rather than pixels reduces the numerical complexity of the task and enforces spatial coherency between clicked object regions. On top of this viewpoint, we identify and group together superclicks as an energy minimization problem.

- *Temporal smoothing*: Single-frame superclick extraction produces a fairly accurate segmentation of the objects in the scene, however it ignores temporal consistency between frames, which can be exploited to further improve the segmentation. Based on the superclicks extracted from a span of consecutive frames, a spatio-temporal energy function is designed in order to transfer information on the labels assigned to corresponding superpixels at different frames.

4.1.1 The game

Human visual attention is the process of gating visual information to be processed by the brain, according to the intrinsic characteristics of visual scenes (*bottom-up process*), and to the task to be performed (*top-down process*) [95]. In particular, bottom-up visual attention is an uncontrolled process, which operates as a zoom lens highlighting areas of high contrast (both spatial and temporal) of the processed visual scenes. Top-down visual attention is, instead, a volition-controlled process that depends mainly on the task to be performed. We exploited these mechanisms, together with principles of game simplicity and competition for effective players' engagement, to design our game where players have simply to take photos of most salient objects in videos. In detail, its main features are:

- The game activates both bottom-up and top-down attention mechanisms. *Bottom-up process:* showing fast video sequences forces players' attention towards the most salient objects (usually the biggest and high-contrast moving ones). *Top-down process:* at the same time, the goal of hitting moving objects, makes players follow objects across consecutive frames.

- Videos usually contain multiple objects, and we want users to clicks on all (or most) of them. Although it is already known that gamers are able to monitor effectively multiple objects while playing [96], we added an *inhibition-of-return* mechanism (which blurs the most salient objects, i.e. those that have received most clicks, thus shifting saliency to the next most salient ones) in order to drive visual attention to all objects in the displayed videos.

- The game strategy, i.e. taking photos of moving objects, was intentionally devised simple in order to enable the general public (without any knowledge of computer vision and machine learning) to play the game. No interaction with the video (e.g., pausing on a frame, moving forward and backward) other than clicking was allowed.
- The competition mechanism consisted of awarding points to correct clicks and making players complete game levels upon the achievement of a goal score for each level. All-time ranking of the best scores is shown in the game main page and at the end of each game, in order to favor competition between players.

4.1.2 Game interface

Figure 4.1 shows an example of a typical in-game screenshot. The video for the current level is, of course, the most important element of the interface and takes up most of the space; the current score obtained by the user is shown at the top; the remaining time before the level's end is shown on the top-left corner (indicated by the OXYGEN icon—a legacy from the initial underwater-oriented application of the game), and the number of points needed to pass the current level is at the bottom of the screen. The mouse cursor is shaped like a camera reticle, and at each click the taken "photo" is shown at the bottom-left corner of the screen. When the user clicks correctly on a target, points are awarded, shown as upward-floating bubbles ("+81" in the example image). Finally, further option buttons are shown at the top-right

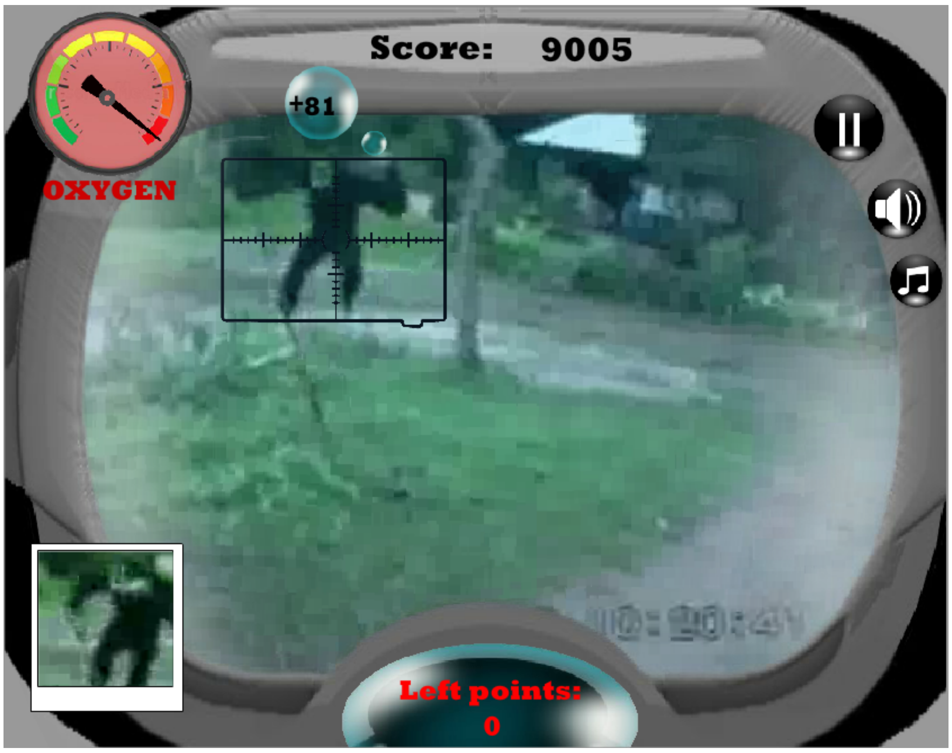

Figure 4.1: *In-game screenshot of the user interface.*

corner of the screen.

Levels

Each level in the game is associated to an input video, which is supposed to be at least 30 seconds long at 10 frames per second (if longer, only the first 30 seconds will be shown). In order to pass a level, a certain amount of points must be scored, starting from 4000 at the first level, and increasing by 2000 at each successive level. As the game is actually relatively simple, this increase represents the main challenge, since it makes it more and more difficult to achieve the required points.

Level-video association is done randomly at each game: i.e., in different game sessions, the video ordering changes to prevent players from knowing in advance where objects might be located. This is necessary since players often tend to maximize their scores also using tricks.

A related issue was the *saliency bias* of some objects with respect to others, which caused users to click always on the same objects (the most salient ones) in a scene even if several others were present (see Figure 4.2 — left image). In order to induce players to click on all available objects, we applied an *inhibition-of-return* mechanism by blurring videos in areas where clicks (by all users) accumulate: this reduced the saliency of underlying objects, and led users to click to other objects in the scene (see Figure 4.2 — right image). Note that users were not instructed not to click on blurred regions: they did this naturally, as those regions became less salient to the human visual attention mechanism. Video blurring was performed by estimating a click density function by means of kernel density estimation, using normal kernels centered at click locations, and blending the frame image with a smoothed version of itself using the click density estimation as blending coefficients for each pixel. This simple, but effective, mechanism allowed us to gather clicks on multiple objects as shown in Figure 4.2.

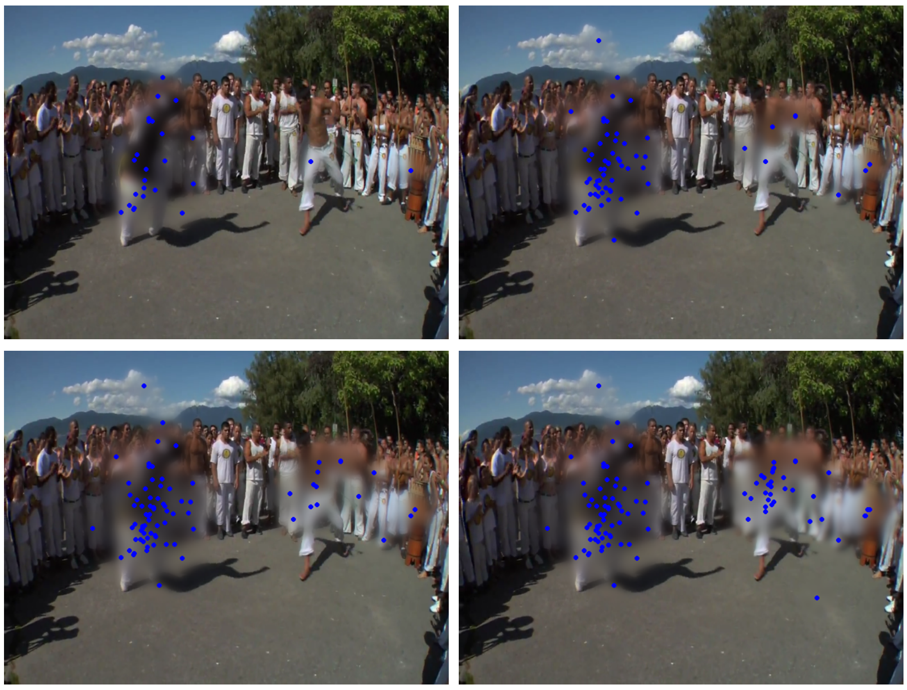

Figure 4.2: *(Left to right and top to bottom) Initially, users tend mainly to click only over the left person. As clicks accumulate on him, the corresponding region is blurred, thus making players shift their attention towards the right person.*

Points

As in any gamification process, it is necessary to pose the task as a competitive one, providing the users with a feedback on how good they are with respect to their previous results or their friends. We employ a point-based system to reflect users' performance on the game, and keep an all-time ranking of the best scores. Points are awarded by clicking correctly on an object of interest, depending on the size of the object and on previous clicks: bigger objects are awarded more points, but successive clicks in the same area earn the user less and less points, according to the formula:

$$P^{+} = \max\left\{A \cdot F_1 \cdot \left(1 - \frac{t}{d}\right), P_{\min}\right\} \tag{4.1}$$

where P^{+} is number of points earned for a correct click, A is the area in pixels of the clicked object, F_1 represents the weight that object with area A has for score computation, t is the number of consecutive clicks within a $M \times M$ pixels region and d is a weighting factor of the number of clicks. In practice, $A \cdot F_1$ (in our case $F_1 = 0.05$, i.e. 5% of the object area A) is the maximum score awarded for a click on a certain object, but this score is progressively reduced (by the quantity $\frac{t}{d}$) down to a minimum of $P_{\min}$ points if the user keeps clicking on the same spot: when the most salient objects get too many clicks, this reduction forces users to click on different objects to earn more points, while at the same time helps to provide data on as many objects as possible. It may seem that our scoring method tends to favor bigger objects, but the maximum score for a click is only a fraction (in our case only 5%) of the clicked object's area and it is reduced if objects are clicked many times. Therefore, t is the most influencing factor

and it allows us to obtain data on multiple objects by shifting saliency progressively to all objects in the scenes.
In order to reduce noisy clicks, we foresee a penalty mechanism, which reduces points awarded by users in case they click too far from targets; the penalty is computed as:

$$P^{-} = F_2 \cdot t \tag{4.2}$$

where P^{-} is the amount of points subtracted to the current score due to a wrong click, t is the number of consecutive clicks falling further than Z pixels from the closest correct object and F_2 is a weighting factor. Penalties prevent users from clicking randomly across the frame and force them to be as more accurate as possible. The values of F_2, M, Z depend on frame resolution and in our case, since all videos are rescaled to 640×480 to fit the game interface, they were set to 20, 30 and 200, respectively; d, instead, is a parameter balancing the trade-off between object areas and clicks in the score assignment and was set to 10 in our experiments; $P_{\min}$ was set to 5. The overall score per level is achieved by summing up P^{+} and P^{-}.
However, to award points to players we need object segmentations (not necessarily highly accurate) on the input videos to tell whether clicks hit or miss objects. In order to have a reference signal — *score video segmentation* — according to which we assign points to players, we use the *output of the system itself*. When the system is first set up and no data is available yet, the initial video object segmentation is obtained by running a classic background modeling method ([9] in our case); although in the beginning this may not be enough to cover all and only objects in the scene, it still provides an adequate base for setting the game up. After users have started to play, the object

segmentation is simply updated based on users' clicks by running the algorithm presented in this paper. It is not strictly necessary for the *score video segmentation* to be extremely accurate: scores are only provided for the benefit of users, in order to keep them interested by means of competition.

Click quality

We also estimate the "quality" (in the sense of "accuracy of clicks with respect to objects") of the data provided by users while playing the game. Quality score is computed on a per-level and per-user basis, as the fraction of user clicks hitting the objects in the level. The quality score $Q_{u,l}$ for user u and level l is defined as:

$$Q_{u,l} = \frac{1}{C_{u,l}} \sum_{i=1}^{C_{u,l}} \mathbb{I}(c_i \in \mathcal{S}) \tag{4.3}$$

where $C_{u,l}$ is the overall number of clicks by user u in the game level l, $\mathbb{I}$ is the indicator function, while $\mathcal{S}$ is the set of objects' superpixels for level l.

We assume that all clicked pixels in a game level by a user gets the same quality score computed as above. We could have computed a global quality score for a single game (i.e., the sequence of levels a user plays before completing the game) or for the user, however different levels may return very different quality scores even within the same game session, due to each video's scene and object characteristics, so a global score would become too generic to describe individual click quality in a level's context.

4.1.3 Superclick extraction

The clicks collected through the web game are used to extract information on the location of objects in the scene for each video frame and to carry out a preliminary object segmentation. We pose the problem as a binary segmentation task (background and foreground) by means of the minimization of an energy function defining the cost of a segmentation. Like some of the most recent methods for video object segmentation [94, 14, 15], we use *superpixels* (computed by SLIC [88]) as basic image parts instead of pixels as they provide two main advantages: 1) reducing the number of variables greatly speeds up the minimization algorithm (the number of variables is scaled down by a factor of 30-50, depending on superpixel settings); 2) the initial segmentation provided by superpixels is usually effective in detecting edges, which allows to simply focus on finding the optimal aggregation, taking boundary detection for granted.

The first processing step, given our target frame F, consists of *superclick extraction*, where a *superclick* is the intuitive extension of the concept of clicks to superpixels. This step is necessary to be able to pose the problem in terms of superpixels only, by “converting” point data (e.g., clicks) to superpixel-oriented ones. Of course, the principle behind this operation is that superpixels containing clicked pixels should be more likely to be marked as superclicks, which are then converted into optimization constraints. However, clicks are generally noisy, thus other factors, such as click density, click quality (as defined previously), closeness to other clicked superpixels need to be taken into account for superclick identification.

Before explaining how superclicks are computed, a more basic question is: *what clicks should we use to analyze a certain frame?* Depending on video frame rate and target speed, users' reaction times may introduce a delay which results in a shift between the frame at which the user clicks on an object and the frame at which the user *intended* to click. Figure 4.3 shows a few examples: it is possible to notice that the delay effect is more visible on some videos than others, mainly according to objects' speed. Since this issue involves complex biological phenomena [97], which are out of the scope of the paper, we adopt a simple but effective empirical approach: we assume that all clicks are delayed by a constant number of frames for all videos. In detail, we empirically found that shifting all clicks back by 2 frames (although the optimal delay may vary from 1 to 4 frames which depends on several factors, one above all the video frame rate) represents a good trade-off between accuracy and complexity.

Let $C = \{c_1, c_2, \ldots, c_{n_C}\} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{n_C}, y_{n_C})\}$ be the players' clicks for frame F, with corresponding quality scores $Q = \left\{q_{c_1}, q_{c_2}, \ldots, q_{c_{N_C}}\right\}$ (each click gets the quality score assigned to the user who did it on a per-level basis). We define a graph-representable energy function [89] over the set of F's superpixels $S = \{s_1, s_2, \ldots, s_{n_S}\}$, with a cost function able to model the "clickedness" of each superpixel independently, and at the same time, to enforce constraints on visual smoothness and click continuity. Our main assumptions are:

1. Superpixels containing a large number of clicks should be marked as superclicks (and vice versa), i.e., they can be seen as hard constraints for segmentation.

2. Clicked pixels should be weighed by the relative quality when

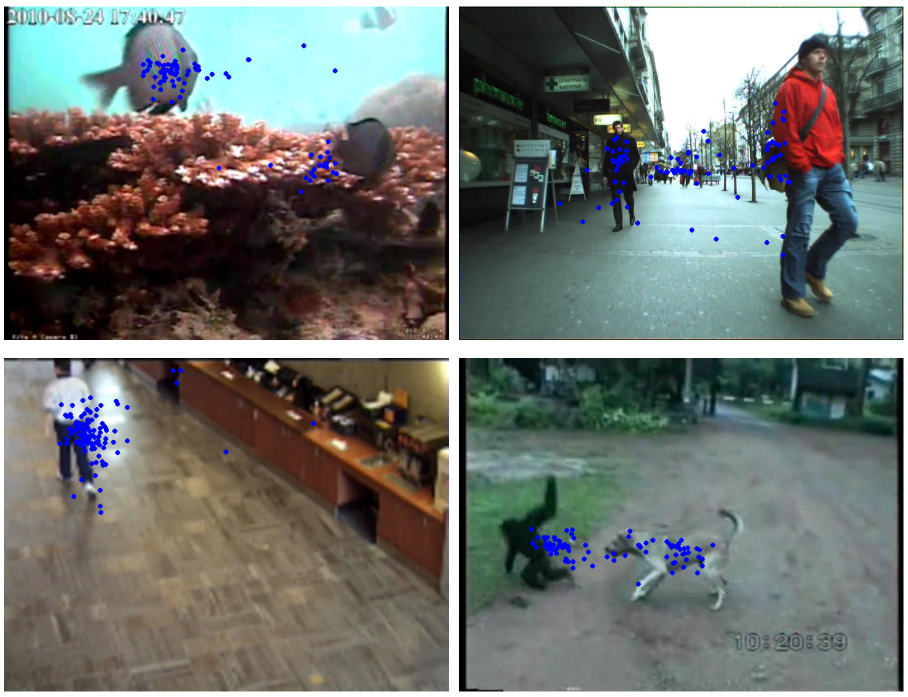

Figure 4.3: *Due to users' reaction times, clicks may be delayed with respect to the "intended" frame. It is possible to notice that this phenomenon may be more or less evident even within the same image, depending not only on the user but also on the objects in the scene.*

evaluating their contribution to a superclick.

3. Unclicked superpixels which are close to clicked and visually-similar superclicks should be marked as superclicks as well, since they are likely to belong to the same object.

4. Isolated clicked superpixels (even if in small groups) should be ignored as being likely noise.

Translating these assumptions into energy potentials, we obtain the following cost function for energy optimization:

$$E_1(\mathcal{L}) = \alpha_1 \sum_{s \in S} V_1(s, l_s, C) + \sum_{(s_1, s_2) \in \mathcal{N}(S)} V_2(s_1, s_2, l_{s_1}, l_{s_2}) \tag{4.4}$$

where $\mathcal{L} = \left\{ l_{s_1}, l_{s_2}, \ldots, l_{s_{n_S}} \right\}$ is the superclick label assignment (l_{s_i} is the binary superclick label for superpixel s_i), $\mathcal{N}(S)$ is the set of pairs of neighbor superpixels (that is, having part of boundary in common; we will also use the notation $\mathcal{N}(s)$ to denote the set of neighbors of the single superpixel s), and α_1 is a weighing factor.

Unary potential V_1 models whether superpixel s is likely to be a superclick or not. This "likeliness" depends on the number and quality of clicks inside the superpixel's region and on the vicinity to clicked superpixels[1]. Therefore, V_1 is defined by the following contributions:

[1]The reader might think that "vicinity to clicked superpixels" should be modeled as a pairwise potential, rather than unary. In fact, it should be modeled as unary because it is not an indication of whether two elements should be assigned the same label (which is what pairwise potentials represent); instead, it uses local information to indicate whether that item, *individually*, is more likely to be assigned to a specific label (1 for "superclick" or 0 for "not superclick")

- ***Clickedness*** K_s: the more (high-quality) clicks a superpixel has received, the more it is likely to be a good candidate superclick. The clickedness score K_s for superpixel s is:

$$K_s = \underbrace{\frac{|C \cap s|}{\max\limits_{t \in S} |C \cap t|}}_{(4.5a)} \underbrace{\frac{1}{|C \cap s|} \sum_{c \in C \cap s} q_c}_{(4.5b)} = \frac{\sum\limits_{c \in C \cap s} q_c}{\max\limits_{t \in S} |C \cap t|} \tag{4.5}$$

  where $C \cap s$ is the set of clicks hitting superpixel s and $|\cdot|$ is set cardinality. The first (unreduced) version explains more clearly what this formula is meant for: Eq. term (4.5a) indicates how many clicks superpixel s contains with respect to the superpixel containing most clicks in the processed frame; Eq. term (4.5b) is, instead, the average quality of clicks inside s, and encodes quality information in the score. The way this score is computed thus addresses items 1 and 2 of the above design principles.

- **Proximity to clicked superpixels** V_s: if s has not received many clicks but is close to superpixels which did, we might want to take it into consideration as a potential superclick. Of course, being close to clicked superpixels by itself is not enough: any superpixel just outside an object's boundary satisfies this requirement; this issue will be addressed by the pairwise potential V_2.

  Our proximity score V_s is computed as the fraction of neighbor superpixels with clickedness score $K_{s_n} > 0.5$, with $s_n \in \mathcal{N}(s)$:

$$V_s = \frac{|\{s_n \in \mathcal{N}(s) : K_{s_n} > 0.5\}|}{|\mathcal{N}(s)|} \tag{4.6}$$

If s gets enough clicks but is isolated, V_s will be low and K_s won't suffice to label it as a superclick. Thus, V_s balances items 3 and 4 of our design principles.

- **Unclicked regularizer**: the point of introducing the V_s score is to allow a superpixel with few or no clicks to be labeled as superclick if its neighborhood hints that it should; however, if an unclicked superpixel is not adjacent to any clicked superpixels, its V_1 potential is zero, which is something we want to avoid. Consider, for example, the case of an object consisting of a large uniform region with a non-uniform users' click distribution (which is actually often the case, as users tend to click at the center of objects): by setting unclicked superpixels to a low (but not null) potential, we allow labels to "spread" from superclicks (as per item 3 of our design principles above)—as long as uniformity requirements, defined by potential V_2, apply. For this reason, we add a constant U_s term to the V_1 potential, which should be small enough not to "push" too much toward the "superclick" label (since clickedness and vicinity clues suggest it should not be), but not so small that it cannot ever be labeled as such.

The definitions of K_s, V_s and U_s have been chosen so that the sum of those terms (clipped to 1 if necessary) can be interpreted as the probability that superpixel s belongs to class "superclick", $P_{s,1} = P(l_s = 1|C, S) = \min(K_s + V_s + U_s, 1)$. Similarly, the complementary probability $P_{s,0} = P(l_s = 0|C, S) = 1 - P_{s,1}$ is the probability that s is "not a superclick". In the energy function, V_1 is meant to represent the cost of assigning a certain label to each superpixel: such costs can be computed as the negative log-likelihood of the two probabilities above:

$$V_1(s, l_s, C) = \begin{cases} -\log P_{s,1} & \text{if } l_s = 1 \\ -\log P_{s,0} & \text{if } l_s = 0 \end{cases} \tag{4.7}$$

Pairwise potential V_2 is the cost of assigning different labels to two adjacent superpixels s_1 and s_2: ideally, it should be large for “similar” superpixels (so that they are assigned the same label) and small for superpixels which are too visually dissimilar to be likely to belong to the same class. Although in general this function could depend on the specific labels being assigned (so that, for example, the cost of assigning labels $(l_{s_1} = 1, l_{s_2} = 0)$ might be different than the cost of assigning labels $(l_{s_1} = 0, l_{s_2} = 1)$), in our case we focus only on estimating the optimal separation point between the “superclick”/“non-superclick” regions, based on visual similarity.
Therefore, potential V_2 is simply expressed as follows:

$$V_2(s_1, s_2, l_{s_1}, l_{s_2}) = \exp\left[-\beta_1 \chi^2(H_{s_1}, H_{s_2})\right] \mathbb{I}(l_{s_1} \neq l_{s_2}) \tag{4.8}$$

where $\chi^2(\cdot, \cdot)$ is the Chi-square distance, H_{s_i} is the RGB color histogram of superpixel s_i, β_1 is a constant, and $\mathbb{I}(l_{s_1} \neq l_{s_2})$ ensures that V_2 is a submodular function, thus making the whole energy function graph-representable [89]. Using a simple similarity measure such as the color histogram has a twofold justification: 1) by construction, superpixels have very little internal structure, so using more complex descriptors is unnecessary; 2) since the function has to be evaluated for all pairs of adjacent superpixels, it is important to perform as efficient operations as possible, in order to keep computation times reasonable.

Once $E_1(\mathcal{L})$ has been minimized by means of graph cut, the extracted superclicks already provide a good approximated segmentation of the objects of interest in the scene, as shown by the examples in Figure 4.4. Nevertheless, output images at this stage can show segmentation errors, e.g., holes, oversegmentations, etc., and further processing by taking into account motion information is carried out to refine the obtained segmentation masks.

4.1.4 Temporal smoothing

The superclick extraction step turns a set of noisy clicks into a set of spatial coherent superclicks per frame, but ignores any temporal information which, instead, is a key factor for video analysis. Therefore, the next step for segmentation refinement consists in exploiting the *temporal consistency* between consecutive frames to “transfer” labels across segmentations. The idea is that if a set of consecutive (in time) segmentations all mark a certain object as foreground, then it is likely that they are correct; similarly, if no (or only few) segmentations include that object, it is probably safer to ignore it in the final output, especially if it is relatively isolated from other candidate foreground objects. Two issues arise when trying to implement the above criterion: first, superclick segmentations are defined in terms of superpixels, and superpixel segmentation is not consistent in presence of motion; second, objects in a video typically move, thus the notion of “a certain object” across several frames implies the employment of an object/point tracking method.

Our approach addresses both issues: 1) we define a *temporal linking* between superclicks by extending the energy function employed

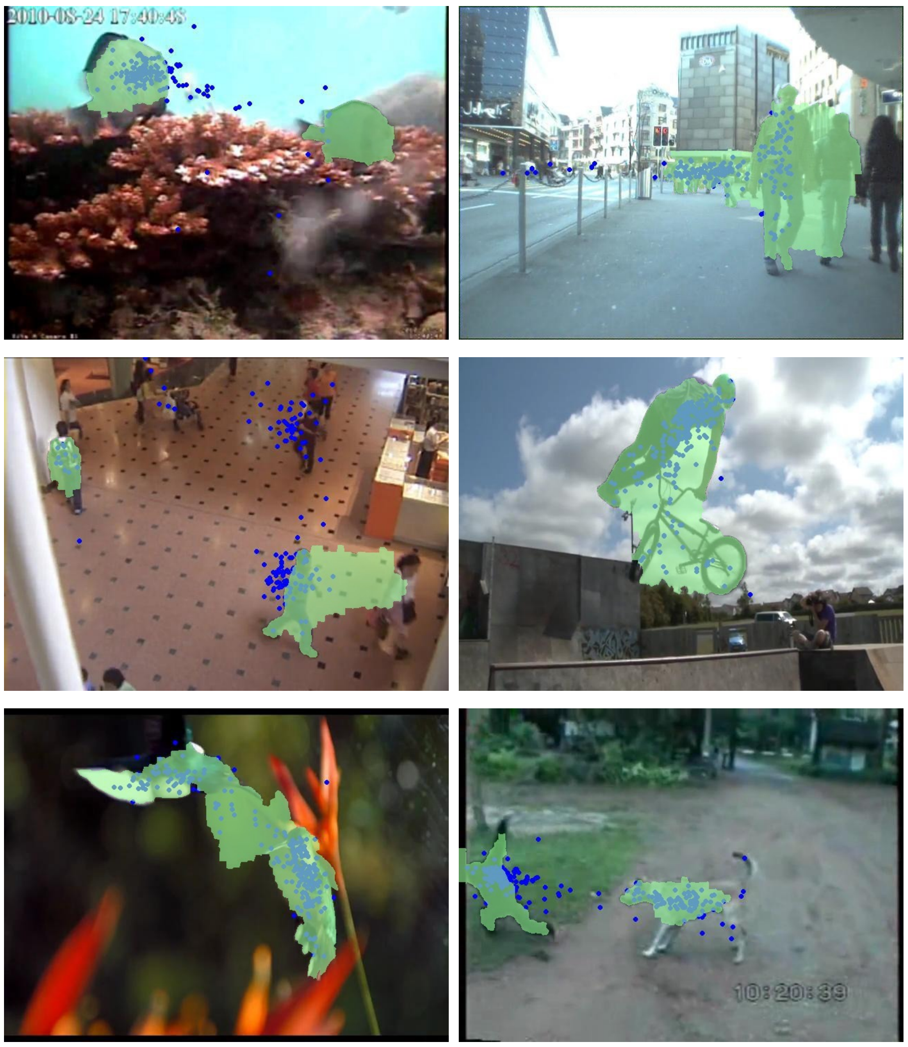

Figure 4.4: *Output examples for superclick identification: blue dots are users' clicks while green regions show the yielded segmentation masks. Segmentation refinement is carried out by including temporal constraints.*

for superclick extraction to take into account visual similarity between spatio-temporal regions; 2) we employ optical flow [16] to estimate where superpixels in frame t may have moved in frame $t+1$: in practice, we introduce pairwise potentials on all pairs of superpixels $\{s_t, s_{t+1}\}$ such that s_t contains at least one pixel p_t whose projection $p_t + v_{p_t}$ into frame $t+1$ under the motion vector v_{p_t} (i.e. v_{p_t} is the motion vector computed between frame t and frame $t+1$ for location p_t) is part of superpixel s_{t+1} in frame $t+1$. Of course, it is unlikely that each superpixel will appear in only one such link, which allows to better "explore" the space around the estimated motion area, thus reducing the amount of error due to the optical flow and performing a more comprehensive analysis on the surrounding superpixels.
In the definition of the cost function employed for the temporal smoothing across frames $t-T$ and $t+T$, where t is the current processed frame and T is a constant which affects the number of frames involved in the temporal smoothing (i.e., $2T+1$), we assume to have identified superclicks for all the involved frames. In particular, we will refer to the same quantities as defined in Section 4.1.3 and add an apex relative to the frame they refer to: for example, l_s^t is the superclick label for superpixel s in frame t, S^{t+1} is the set of superpixels in frame $t+1$, and so on. The output label set will be identified by $\mathcal{L}$, and each label by l_s, without the temporal apex, and they refer to the segmentation of the current processed frame. We can now introduce

the energy function used for the final segmentation:

$$\begin{aligned} E_2(\mathcal{L}) = & \sum_{\tau=t-T}^{t+T} \left[\alpha_2 \sum_{s \in S^\tau} W_1(s, l_s, l_s^\tau) \right] + \\ & + \sum_{\tau=t-T}^{t+T} \left[\sum_{(s_1,s_2) \in \mathcal{N}(S^\tau)} V_2(s_1, s_2, l_{s_1}, l_{s_2}) \right] + \\ & + \sum_{(s_1,s_2) \in \mathcal{N}_T(\cup_{\tau=t-T}^{t+T} S^\tau)} V_2(s_1, s_2, l_{s_1}, l_{s_2}) \end{aligned} \tag{4.9}$$

The first two lines of the cost function includes single-frame potentials, which consist of, respectively, unary potentials for each identified superpixel (first line) and pairwise potentials (second line) for each pair of superpixels belonging to the same frame. The last term (third line) enforces temporal smoothing, and consists of pairwise potentials computed over the set $\mathcal{N}_T(\cup_{\tau=t-T}^{t+T} S^\tau)$, which represents all pairs of superpixels (from all the frames in the considered time interval) satisfying the "temporal linking" criterion described above, i.e. such that the two superpixels in each pair belong to temporally consecutive frames, and that at least one pixel belonging to one of them is projected onto the other by means of optical flow.

Similarly to V_1 (defined in Section 4.1.3), unary potential W_1 models whether superpixel s is more likely to be assigned to background or foreground *per se*. In this stage, we simply assign a constant value to the potential depending on whether it had been identified, at the previous stage (see Section 4.1.3), as a superclick or not (i.e. depending on l_s^τ). In detail, given superpixel s, we set its corresponding *foreground cost* and *background cost*; the value of each cost depends on l_s^τ: if s was labeled as a superclick, we expect it to be more likely that it is foreground, so the background cost should be higher, and vice versa. W_1 is therefore computed as follows:

$$W_1(s, l_s, l_s^\tau) = \begin{cases} \gamma_1 & \text{if } (l_s = 1 \wedge l_s^\tau = 1) \vee (l_s = 0 \wedge l_s^\tau = 0) \\ \gamma_2 & \text{otherwise} \end{cases} \tag{4.10}$$

with $\gamma_1 < \gamma_2$.

Pairwise potential V_2 is defined as in Section 4.1.3, but in the last term (third line of Formula 4.9) of E_2 we employ it to evaluate the similarity not only between adjacent superpixels in the same frame, but also “temporally-adjacent” (according to $\mathcal{N}_T$) superpixels in consecutive frames. In order to deal with errors in optical flow computation, we do not simply assign a constant based on the presence or absence of a temporal link between two superpixels in consecutive frames, but also verify that they are visually similar and in fact refer to the same object/region in both frames. Thus, we manage to enforce the criteria according to which superpixels overlaying the same region across different frames should all be assigned the same label.

Both E_1 and E_2 are binary pairwise energy functions with submodular pairwise potentials, and as such we minimize them exactly by graph-cuts in order to get the final segmentation for frame t. Some qualitative examples are shown in Figures 4.5 and 4.6 (compared to those obtained by using only superclick extraction shown in Figure 4.4): it is easy to notice the difference in terms of segmentation quality achieved by analyzing a single frame only and by employing temporal smoothing, which is able to extract much better objects' shapes. It should also be noted that most of the processing (e.g. superpixel extraction and optical flow) can be shared when processing frames one after another, thus reducing the main processing time to

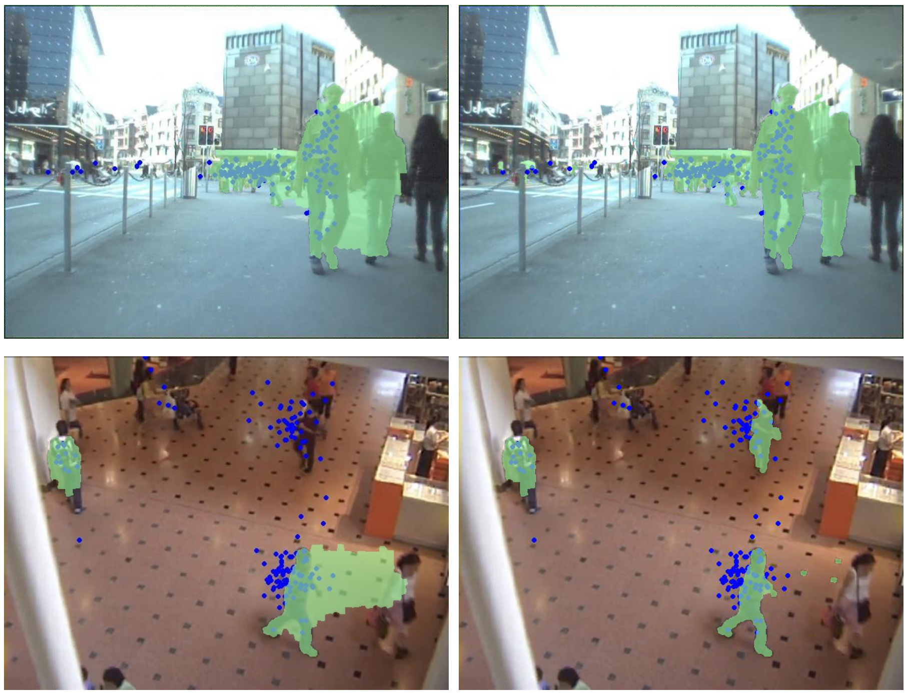

Figure 4.5: Qualitative comparison between segmentations obtained when excluding (first column, see Fig. 4.4) and including temporal smoothing (second column): examples from the ETH BIWI and I2R datasets.

superpixel segmentation and computation of optical flow for a single frame only.

4.1.5 Experimental results

In this section we present the experimental results obtained by testing our gamification approach and link them to the state of the art on

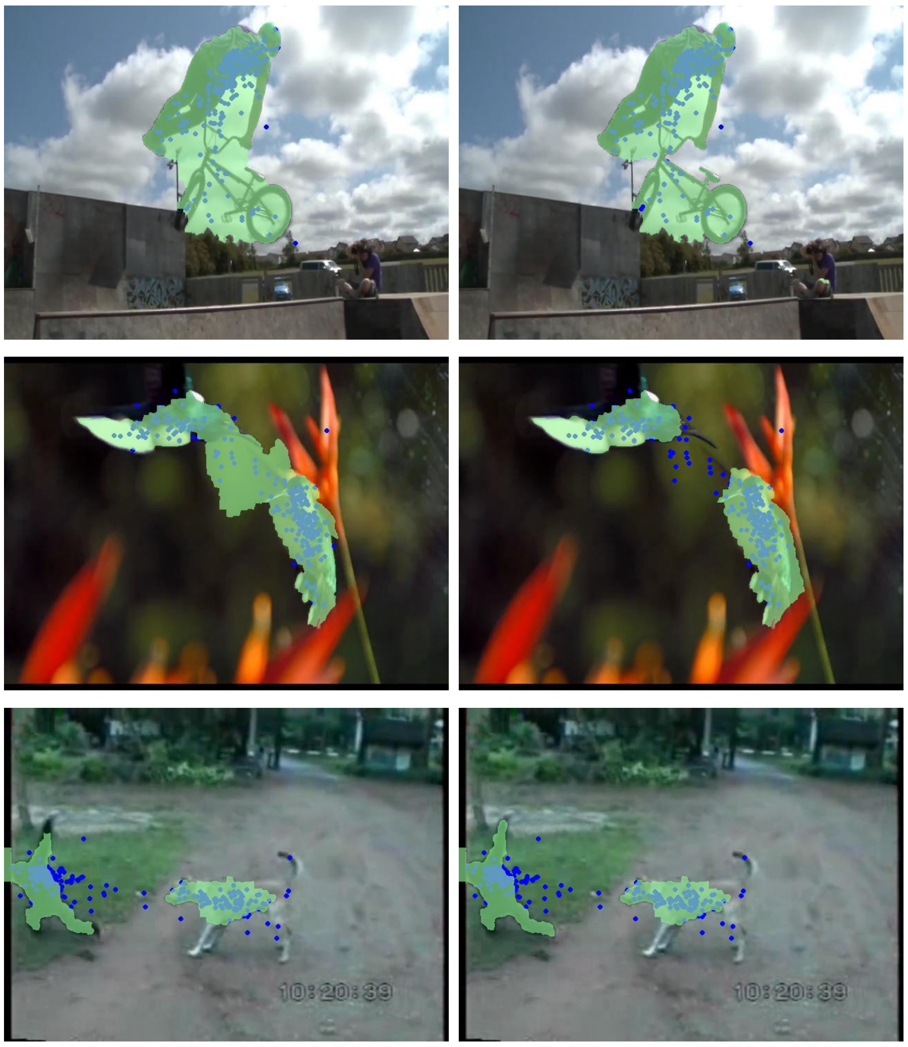

Figure 4.6: *Qualitative comparison between segmentations obtained when excluding (first column, see Fig. 4.4) and including temporal smoothing (second column): examples from the SegTrack dataset.*

interactive video annotation and automated video object segmentation methods.

Datasets

For testing the accuracy of our method we created 24 game levels (each one 300 frames long) using 73 videos from standard video benchmarking datasets: YouTube Objects [98], FBMS-59 [99], VSB-100 [100], Underwater Dataset [10], I2R [87], ETH BIWI [101], SegTrack v2 [86]. In order to avoid any bias in the results, the 73 videos (we did not use all videos since it would have needed too many players and too long time to obtain meaningful results) were chosen randomly from the pool (1318 videos, of which 1116 from YouTube Objects only) of all available videos, in such a way as to have the same number of videos from each dataset. All selected videos were downsampled or upsampled in time, in order to have each video be played at a challenging speed. The first frame of each video was replicated 5 times, to allow collecting clicks from the very beginning of each video sequence. Videos were separated through black images in order to prevent users from inertially clicking across video boundaries.

Collected data

Our experiments involved 63 players, who were simply asked to compete with each other by achieving the highest possible score. The following information describes the amount of collected data and playing statistics:

- **Level time:** 30 seconds.

- **Game time (24 levels):** 12 minutes.
- **Played games:** 136.
- **Total play time:** 27.1 hours. On average, each participant played for about 26 minutes.
- **Total number of clicks:** 597,802.

Algorithm parameters

In Sections 4.1.3 and 4.1.4, we introduced some parameters which control the trade-off between clicks and visual regularity in the segmentation process. We empirically set the values for those parameters, as follows: $\alpha_1 = 1/4$, $\alpha_2 = 1/5$, $U_s = 0.4$, $\beta_1 = 5$, $T = 2$, $\gamma_1 = 0.1$, $\gamma_2 = 0.9$.

These parameter values were used to compute the results shown in the next section. It is important to note that the same parameters were used for all videos, although they have distinct differences in scenery, type of targets, motion patterns, motion speed, frame rate, etc. It is foreseeable that applying the same method to videos belonging to a more homogeneous set of videos would yield higher accuracy.

Segmentation results

The metrics employed for performance analysis were pixel-level precision (Pr), recall (Rec), F-measure (F_1) and average Pascal Overlap Measure (POM: intersection over union between ground truth and output segmentations masks). Such metrics are computed for each video, and results are reported by dataset as the average values of those metrics for the videos in the dataset.

Precision and recall are defined in terms of true positives (*TP*), false positives (*FP*) and false negatives (*FN*). For a given video, *TP* is defined as the cumulative (i.e., summing across all frames) number of pixels correctly labeled as foreground; *FP* and *FN* are computed analogously. Then, precision, recall and F-measure are computed as:

$$\mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{4.11}$$

$$\mathrm{Rec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4.12}$$

$$\mathrm{F}_1 = 2 \cdot \frac{\mathrm{Pr} \cdot \mathrm{Rec}}{\mathrm{Pr} + \mathrm{Rec}} \tag{4.13}$$

Instead, by its nature, *POM* is computed frame by frame, and the overall score for a video is obtained by averaging across frames. More precisely, if we let T be the number of frames in a video, G_i the ground truth mask for frame i, S_i the output segmentation mask for frame i, and $|\cdot|$ the operator returning the number of "true" pixels in a boolean mask, the *POM* score for a video is:

$$\mathrm{POM} = \frac{1}{T} \sum_{i=1}^{T} \frac{|G_i \cap S_i|}{|G_i \cup S_i|} \tag{4.14}$$

Role of spatio-temporal segmentation refinement

Figures 4.5 and 4.6 show a qualitative comparison in terms of segmentation outputs when employing only superclick extraction phase (see Section 4.1.3) and when exploiting the temporal consistency between consecutive frames of superclicks. Table 4.1 reports quantitatively how including spatio-temporal based refinement enhanced the segmentation accuracy. It can be noted that in some cases the accuracy gain was lower (ETH BIWI, I2R, YouTube Objects) than in others (VSB-100, Fish, SegTrack v2). This depends on the dynamics of the video sequences in each dataset: for example, ETH BIWI, I2R and YouTube Objects all feature slow objects or static cameras or camera-compensated motion, which make it easier for players to click on the objects, thus reducing the impact that spatio-temporal refinement provides. Conversely, videos in SegTrack v2 or VSB-100 are characterized by strong camera and object motion, resulting in noisier input data: in these cases, spatio-temporal refinement demonstrated effective to recover users' failures in identifying objects.

For all the following evaluations, we used the variant including the spatio-temporal segmentation refinement.

Accuracy w.r.t. users' play time

Figure 4.7 shows how segmentation results vary in relation to the amount of users' play time in terms of pixel-level precision, recall and F-measure. Users' play time is computed cumulatively from the number of games users have played at a certain point. For example, the results at 5 hours of users' play time were obtained by considering the clicks from the first 25 games (12 minutes per game $\times$ 25 games = 5 hours) played on the system by all users. Of course, the longer the play time, the larger the number of clicks used by the segmentation algorithm.

At the moment we interrupted the experiment, for some datasets the accuracy had already stopped improving, and in some cases it had even started to decrease as the participants played more games

	Superclick extraction			**Spatio-temporal refinement**		
Dataset	**Pr**	**Rec**	**F_1**	**Pr**	**Rec**	**F_1**
Youtube Objects	0.925	0.466	<u>0.596</u>	0.929	0.498	<u>0.621</u>
FBMS-29	0.618	0.843	<u>0.698</u>	0.655	0.861	<u>0.728</u>
VSB-100	0.571	0.783	<u>0.627</u>	0.622	0.789	<u>0.660</u>
SegTrack v2	0.606	0.860	<u>0.703</u>	0.685	0.898	<u>0.773</u>
ETH BIWI	0.640	0.765	<u>0.697</u>	0.656	0.819	<u>0.728</u>
I2R	0.621	0.694	<u>0.655</u>	0.633	0.722	<u>0.675</u>
Fish Videos	0.579	0.927	<u>0.705</u>	0.636	0.972	<u>0.760</u>
Average	0.651	0.762	<u>0.668</u>	0.688	0.794	<u>0.706</u>

Table 4.1: *Average segmentation accuracy for the video categories employed in our game in terms of precision, recall and F-measure, when we employ only superclick extraction (first column) and when we refine the output segmentation by means of spatio-temporal linking between superclicks in consecutive frames (second column).*

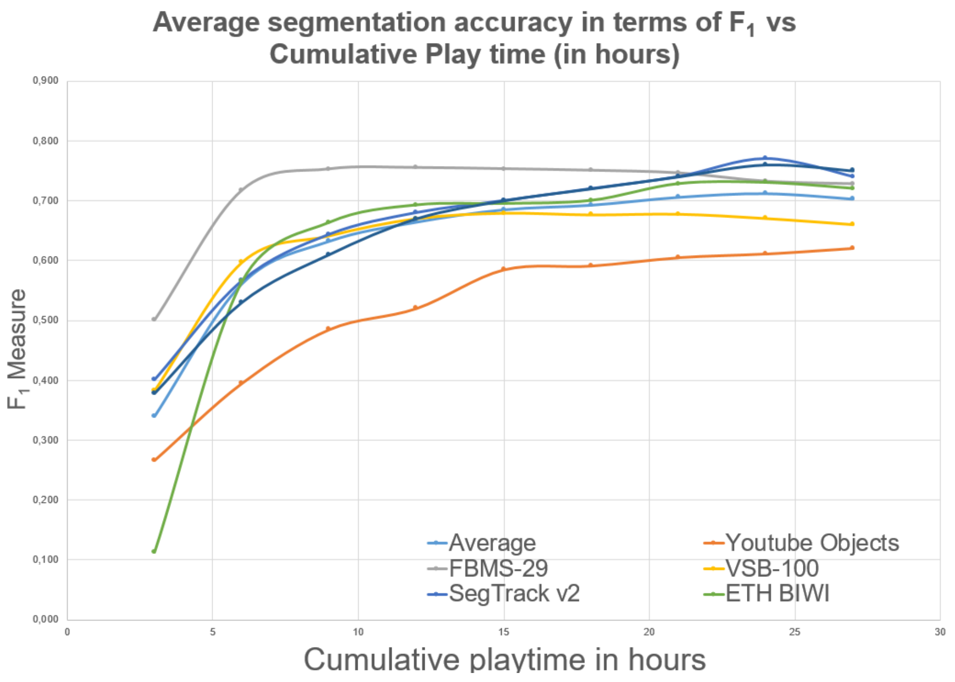

Figure 4.7: *Average F_1 measure over different datasets as cumulative users' play time increases. The results for a certain value T of cumulative play time are obtained by considering the clicks from the first $T/12$ games played on the system, where 12 is the duration in minutes of each game.*

(i.e. as more clicks were being collected). This is likely a limitation of the proposed segmentation approach: as soon as enough clicks have been collected which allow to sufficiently highlight superclicks from background, additional correct clicks do not provide any new useful information to the algorithm, while additional incorrect clicks increase the amount of noise. Nevertheless, this limitation did not prevent our method to achieve satisfactory results, especially in comparison to the state of the art (see Section 4.1.5). Furthermore, discovering and highlighting this kind of problems provides useful indications on how to improve effectively the approach, e.g., driving players to click on those object parts (especially at the borders) and/or to correct wrong annotations, which may indeed enhance segmentation performance.

Comparison with interactive video annotation methods

We then compared the accuracy of the gamification approach to existing interactive video segmentation ones [102, 45, 44]. While [44] is a pure interactive video object segmentation approach, [45] and [102] are, respectively, a semi-supervised method for video annotation and an approach for optimizing segmentation energies, thus they were adapted to our case. More specifically, for the method in [45] we asked a user to provide accurate annotations to initial frames; these annotations were then propagated to consecutive frames for the final segmentation. As for the method in [102] we used the L_2 metric between user-provided shape priors (as suggested by the authors) and target geometric shape moments for foreground segmentation, thus making the method an interactive segmentation one.

For a single user the interaction time of our method depends only on the video length, i.e. there is a linear dependency between the number of annotated frames and annotation times, while existing interactive methods [102, 45, 44] show a non-linear (exponential) dependency. Nevertheless, using data generated by a single user in a single played game would result in poor accuracy performance (as discussed in the previous section), thus we consider, for comparison with the state of the art, the cumulative time spent by a single user in multiple game sessions (generally one frame is shown for 0.2 seconds in our game, thus if we select the clicks of one user in five game sessions, the whole interaction time would be of 1 second). We selected randomly 10 frames from SegTrack v2 and assessed how the segmentation accuracy changed with respect to interaction times. For our method, we considered the clicks of the user who played most games. Quantitatively, Table 4.2 reports the achieved segmentation accuracy (expressed by POM in percentage) over the 10 considered frames by all the comparing methods in two cases: a) within 50 seconds of annotation and b) the maximum achieved accuracy. Let us recall that the interaction time 50 seconds (corresponding to about 25 game sessions) for our method is given by the playtime of a single user, and that the same value can be achieved by involving 25 users, in parallel, playing only one game session, i.e. with a very little human annotation effort as opposed to the other solutions [102, 45, 44].

Comparison with automated video segmentation methods

Despite the proposed method is more inline with the research on interactive video annotation, it can be seen as a video object segmentation approach (with very little human intervention) and as such

	POM within 50 secs	Maximum POM
[102]	39.8	56.6 ($\sim$1,200 secs)
[45]	42.2	65.2 ($\sim$500 secs)
[44]	61.5	84.3 ($\sim$1,400 secs)
Our method	**72.4**	**72.4** ($\sim$50 sec)

Table 4.2: *Comparison in terms of segmentation accuracy — measured as average POM in percentage, respectively, achieved within the first 50 secs of annotation (first row) and maximum value (with the related interaction times) — between our approach and other interactive video annotation methods on a subset of 10 frames extracted from SegTrack v2 dataset.*

it is useful to link its performance with the state of the art on the automated methods. The comparison was performed on the Youtube-Objects dataset (largely employed as a benchmarking dataset for video object segmentation), and we selected as comparing methods those exploiting superpixels for video object segmentation, namely, [99, 103, 104, 105, 14, 45]. The results, in terms of average POM in percentage, are reported in Table 4.3.

Tables 4.2 and 4.3 indicate that our method performs better than automated video object segmentation methods and slightly worse than interactive video annotation approaches. This is not surprising, since interactive video annotation tools require users to spend more time in providing accurate annotations; this may be an advantage for small problems, but makes such methods hardly applicable to large video datasets (such as Youtube-Objects). In this sense, our gamification

	[99]	[103]	[104]	[105]	[14]	[45]	Ours
Category							
aeroplane	13.7	17.8	73.6	75.8	70.9	86.3	56.6
bird	12.2	19.8	56.1	60.8	70.6	81.0	67.6
boat	10.8	22.5	57.8	43.7	42.5	68.6	68.3
car	23.7	38.3	33.9	71.1	65.2	69.4	78.9
cat	18.6	23.6	30.5	46.5	52.1	58.9	50.0
cow	16.3	26.8	41.8	54.6	44.5	68.6	74.8
dog	18.0	23.7	36.8	55.5	65.3	61.8	74.2
horse	11.5	14.0	44.3	54.9	53.5	54.0	89.0
motorbike	10.6	12.5	48.9	42.4	44.2	60.9	63.1
train	19.6	40.4	39.2	31.4	29.6	66.3	66.2
average	15.5	23.9	46.3	53.7	53.8	67.6	68.9

Table 4.3: *POM in percentage for the Youtube-Objects dataset*

approach provides an interesting trade-off between accuracy and annotation times.

As an additional test, we computed the performance of our segmentation algorithm assuming absolute correctness of input clicks. To do so, we artificially computed “clicks” by selecting points in a grid pattern from all ground truth masks (the density of the grid varying in order to obtain the same average number of clicks as from our previous experiments), as shown in Figure 4.8. In this test, we obtained an average POM score of 0.78, computed over all videos from the employed datasets, which is a markedly higher value than the tested start-of-the-art methods (the average POM score we achieved on Youtube-objects dataset was 0.73) and sensibly outperformed the methods reported in Table 4.3).

Processing times

Using $T = 2$, processing a single frame requires minimizing five energy functions for superclick extraction and one (with the added temporal dimension) for accurate segmentation. Our Matlab implementation, running on a PC with a quad-core i7 CPU and 8 GB RAM, takes 3 seconds for superclick extraction in a single frame and 30 seconds for the temporal optimization (actually, for the four optical flow computations which link superpixels in time; graph-based minimization time is negligible), which would amount to 45 seconds in total. However, after the initial bootstrap phase, segmenting a new frame can benefit from having already extracted superpixels and computer the optical flows for previous frames, so processing is reduced to a single superpixel extraction and a single optical flow computation, resulting in a

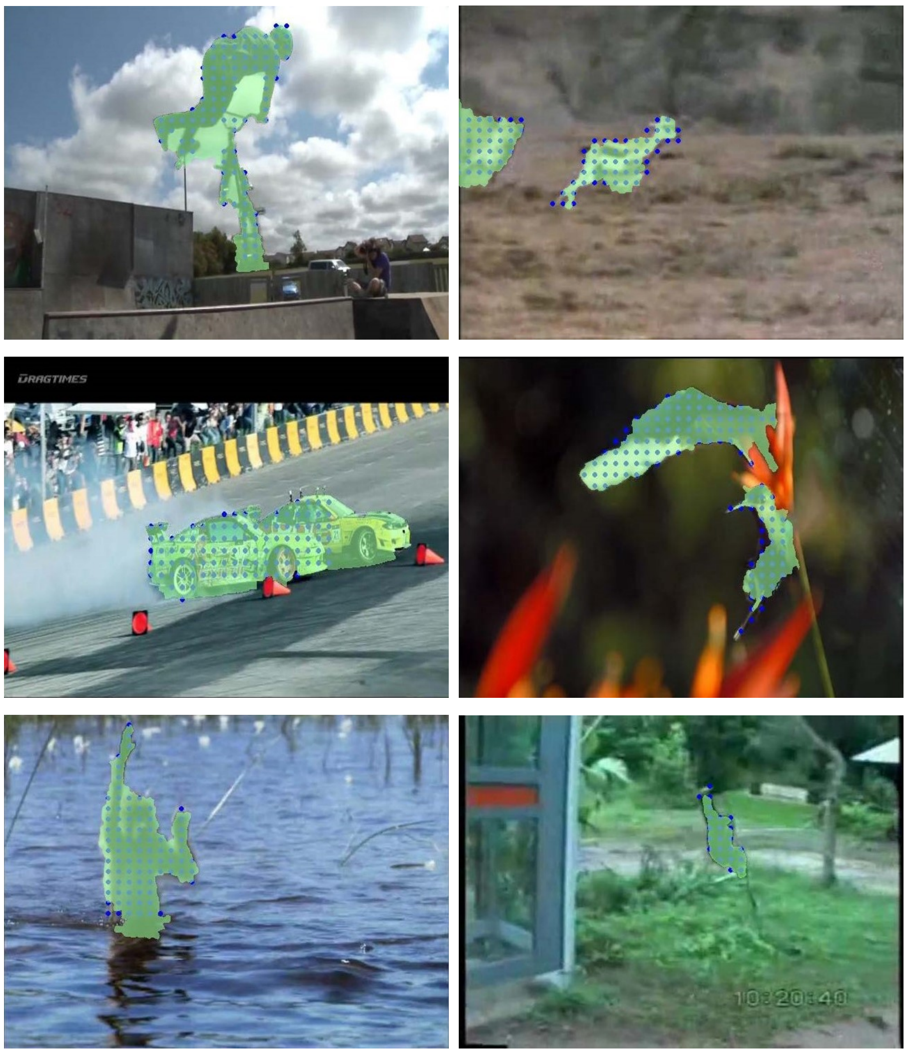

Figure 4.8: *Output segmentations when using only points within objects of interest (i.e. taken from ground truth segmentation masks): blue dots are ground truth points while green regions show the output segmentation masks.*

frame processing time of 10.5 seconds.

4.2 Eye-tracking–based video object segmentation

So far, we saw that video object segmentation accuracy can be enhanced by having users click with a computer mouse on object locations — even if approximately. Now, it is time to see what results can be obtained when we take the mouse away and let the user choose autonomously where to look in a video, while following his or her gaze with an eye-tracker.

Modern eye-tracking devices are able to follow a user's gaze quite accurately — e.g., the error committed by the Tobii T60 eye-tracker employed in these experiments amounts to about 5 mm at 65 cm[2] — and can be reliably used to detect where a user is looking on a screen. Moreover, the typical operating frequency of 60 Hz is usually high enough to accurately follow eye movements and identify *fixations* (steadily maintaining the visual gaze on a single location) and *saccades* (quick movements between fixations). Given these premises, making the switch from mouse clicks to gaze points may seem a seamless transition. However, there are some aspects worth investigation. First of all, the degree of control over one's own eyes is not as precise the hands; often, it is guided by involuntary attention shifts (*bottom-up saliency*), which in our application may arise as a source of noise. On the other hand, not even the most enthusiastic player can click

[2]`http://www.tobiipro.com/siteassets/tobii-pro/product-descriptions/tobii-pro-tx-product-description.pdf`

sixty times a second, which suggests that the cost in precision may be compensated by being able to collect a larger mass of data in a shorter interaction time.

The experiments we present in this chapter were carried out in a laboratory settings using a video-based eye-tracker, which still requires users to interact with a computer and to perform specific tasks — hardly a less explicit modality than playing a game. However, as we noted in Chapter 1, while playing a game requires someone's dedication regardless of the playing modalities, the commitment needed to undergo an eye-tracking experiment is only related to the specific technological implementation; future developments, e.g., eye-trackers on phone front cameras or on glasses, will allow to remove this limitation and make eye-tracking a background process during everyday activities.

Given the substantially homogeneous nature of mouse clicks and eye gazes, both being sources of punctual data, we employ a similar energy-minimization framework as the one shown previously in this chapter, in order to reduce the bias in the comparison due to algorithmic differences rather than data reliability.

Therefore, our interactive video segmentation method is posed as a region labelling problem with two labels (foreground and background) treated as a spatio-temporal energy minimization problem. In the first step, we propose a spatial frame segmentation by defining a cost function which takes into account both visual cues and gaze data. In the second step, we refine the initial segmentation by enforcing spatio-temporal consistency between the segmented objects in consecutive frames.

4.2.1 Initial frame segmentation

As above, the first step of the algorithm performs image segmentation at the frame level, which is treated as a binary pixel labeling task (background and foreground). We start from *superpixel* segmentation (again, using SLIC [88]) and then group superpixels through minimization of an energy cost which enforces spatial and visual coherency between superpixels as well as including gaze data, as constraints, in the labeling cost.

The energy function is defined over the set of superpixels $S = \{s_1, \dots, s_{n_S}\}$ in order to identify superpixels "looked at" by users and, at the same time, to enforce spatial constraints on visual smoothness at the frame level. The objectives of the definitions are: 1) superpixels containing gaze points should be candidates for being part of the foreground, but the constraint must not be hard, due to the noisiness of the data; 2) fixation points should be considered more reliable than saccades; 3) superpixels without gaze points but which are spatially close and visually-similar to those containing gaze points should be included as soft constraints; 4) isolated superpixels with gaze points should be ignored as being likely noise; 5) spatial regularity over the assigned labels should be encouraged.

The cost function for energy optimization reflecting these assumptions can be defined as:

$$E_1(\mathcal{L}) = \sum_{s \in S} V_1(s, l_s) + \sum_{(s_1,s_2) \in \mathcal{N}(S)} V_2(s_1, s_2, l_{s_1}, l_{s_2}) \tag{4.15}$$

where $\mathcal{L} = \left\{l_{s_1}, l_{s_2}, \dots, l_{s_{n_S}}\right\}$ is the superclick label assignment (l_{s_i} is the binary superclick label for superpixel s_i), $\mathcal{N}(S)$ is the set of pairs of neighbor superpixels (that is, having part of boundary in common;

we will also use the notation $\mathcal{N}(s)$ to denote the set of neighbors of the single superpixel s).

Unary potential V_1 models whether a selected superpixel s should be considered as part of the foreground or not. In order to encode the required constraints on gaze point location, we compute for each superpixel s a G_s score, taking into account the distance between a superpixel's centroid and the closest gaze point (either fixations or saccades) and the duration of the fixation (mesured in frames) associated to the relevant gaze points:

$$G_s = e^{-\frac{d^2(s,\hat{s})}{\sigma_d^2} - \frac{1}{\hat{t}\sigma_t^2}} \tag{4.16}$$

where $d^2(s, \hat{s})$ is the squared distance between superpixel s's centroid and the centroid of its closest superpixel with gaze points, $\hat{s}$, and $\hat{t}$ is the longest eye fixation duration associated to the gaze points in $\hat{s}$, thus giving more importance to longer and more "reliable" fixations; σ_d^2 and σ_t^2 are two constants. In practice, G_s is higher (up to 1) if a superpixel is likely to belong to the foreground, *based on gaze point constraints only.*

The value of unary potential V_1 is computed based on G_s and on the assigned label: if G_s is high, then the costs of assigning it to the foreground ($l_s = 1$) or to the background ($l_s = 0$) should be respectively low and high; the contrary applies if G_s is low. Using a standard negative log-likelihood representation, the resulting V_1 can be formulated as:

$$V_1(s, l_s) = \begin{cases} -\log G_s & \text{if } l_s = 1 \\ -\log(1 - G_s) & \text{if } l_s = 0 \end{cases} \tag{4.17}$$

Pairwise potential V_2 is the cost of assigning different labels to two adjacent and visually similar superpixels s_1 and s_2 and in our case is expressed as:

$$V_2(s_1, s_2, l_{s_1}, l_{s_2}) = e^{-\beta_1 \chi^2(H_{s_1}, H_{s_2})} \mathbb{I}(l_{s_1} \neq l_{s_2}) \tag{4.18}$$

where $\chi^2(\cdot, \cdot)$ is the Chi-square distance, H_{s_i} is the RGB color histogram of superpixel s_i and β_1 is a constant; $\mathbb{I}$ is the indicator function, as required in order to guarantee submodularity and efficiently solve the minimization problem [89].

The initial per-frame segmentation is then yielded by minimizing $E_1(\mathcal{L})$ through graph cut.

4.2.2 Temporal-based segmentation refinement

The previous step converts noisy gaze data into a set of spatial coherent superpixels per frame, but does not take into account any temporal consistency between superpixels over consecutive frames, which, instead, is necessary to cope with per-frame segmentation errors. The idea is that if a set of time-consecutive segmentations all mark a certain object as "interesting", then it is likely that they are correct; similarly, if no (or only few) segmentations include that object, it is probably safer to ignore it in the final output. However, two aspects need to be taken into account: 1) superpixel segmentation is not consistent in presence of motion; 2) the employment of a tracking method might be necessary to link per-frame segmentations in consecutive frames. Our temporal-based segmentation refinement method addresses both issues: 1) we link *temporally* superpixels by defining an energy function able to take into account visual similarity between spatio-temporal regions; 2) we employ optical flow [16] to estimate where superpixels in frame t may have moved in frame $t+1$. In practice, we "temporally link" superpixels in consecutive frames by introducing pairwise potentials on all pairs of superpixels $\{s_t, s_{t+1}\}$ such that s_t contains at least one pixel p_t whose projection $p_{t\to t+1}^{v_{p_t}} = p_t + v_{p_t}$ into frame $t+1$ under the motion vector v_{p_t} (i.e., v_{p_t} is the motion vector computed between frame t and frame $t+1$ for location p_t) is part of superpixel s_{t+1} in frame $t+1$. Of course, it is unlikely that each superpixel will appear in only one such link, which allows to better "explore" the space around the estimated motion area, thus reducing the amount of error due to the optical flow and performing a more comprehensive analysis on the surrounding superpixels.

In the definition of the cost function employed for the temporal-based segmentation refinement in $2T+1$ consecutive frames from $t-T$ to $t+T$, with t being the current processed frame, we assume to have the segmentations maps (the ones coming from per-frame segmentation defined in Sect. 4.2.1) for all the $2T+1$ involved frames. The energy function used for the spatio-temporal segmentation is defined as follows[3]:

$$
\begin{aligned}
E_2(\mathcal{L}) = & \sum_{\tau=t-T}^{t+T} \left[\sum_{s \in S^\tau} W_1(s, l_s, l_s^\tau) \right] + \\
& + \sum_{\tau=t-T}^{t+T} \left[\sum_{(s_1,s_2) \in \mathcal{N}(S^\tau)} V_2(s_1, s_2, l_{s_1}, l_{s_2}) \right] + \\
& + \sum_{(s_1,s_2) \in \mathcal{N}_T(\cup_{\tau=t-T}^{t+T} S^\tau)} V_2(s_1, s_2, l_{s_1}, l_{s_2})
\end{aligned}
\tag{4.19}
$$

The first two lines of Eq. 4.19 are, respectively, a unary potential for each identified superpixel (first line) and a pairwise potential for each pair of superpixels belonging to the same frame (second line). The last term (third line), instead, aims at enforcing temporal smoothing through a pairwise potential defined over the set $\mathcal{N}_T(\cup_{\tau=t-T}^{t+T} S^\tau)$, i.e. the set of all pairs of superpixels in the $2T+1$ "temporally linked", as described above. Unary potential W_1 models whether superpixel s is more likely to be background or foreground: if s was labeled as foreground in the frame-segmentation we expect it to be more likely that it is foreground (with a cost lower than being background), and vice versa. W_1 is therefore computed as:

$$
W_1(s, l_s, l_s^\tau) = \begin{cases} -\log \gamma_1 & \text{if } (l_s = 1 \wedge l_s^\tau = 1) \vee (l_s = 0 \wedge l_s^\tau = 0) \\ -\log \gamma_2 & \text{otherwise} \end{cases}
\tag{4.20}
$$

with $\gamma_1 > \gamma_2$.

Pairwise potential V_2 is defined as in Section 4.2.1, but in the last term (third line of Formula 4.19) of E_2 we employ it to evaluate the

[3]We employ the same symbols of Section 4.2.1 adding an apex relative to the frame they refer to: e.g., l_s^t is the superclick label for superpixel s in frame t, etc.

similarity not only between adjacent superpixels in the same frame, but also “temporally-adjacent” (according to $\mathcal{N}_T$) superpixels in consecutive frames. E_2, similarly to E_1, is minimized by graph-cut in order to get the output segmentation for frame t.

4.2.3 Experimental results

Experiment settings

The eye-gaze experiments involved 16 subjects, who were asked to watch freely a set of short videos with moving objects; no further instructions on what they were supposed to look at were given. During the experiments, users’ gaze was recorded by a Tobii T60 eye tracker (capturing 60 gaze points per second), which provides information on saccades and fixation duration.

The method’s parameters were identified by exhaustive search on the parameter space and set as follows: $\sigma_d^2 = 10000$, $\sigma_t^2 = \frac{1}{3}$, $\beta_1 = 1$, $\gamma_1 = 0.8$, $\gamma_2 = 0.4$. Superpixel segmentation was carried out through SLIC [88] and superpixel size was set to 7×7, as a compromise between segmentation errors, sensitivity to noise, and processing speed.

Datasets and baselines

Performance evaluation was carried out on 9 video sequences, with pixel-accurate segmentation of moving objects, taken from three visual benchmarks for video object segmentation: SegTrack v2 [86], FBMS-59 [99], and VSB-100 [100]. The selected videos included features such as: camera motion, slow object motion, object-background similarity, non-rigid deformations and articulated objects. The superpixel-based automated video object segmentation method in [14], using public source code and default parameters, was used as baseline.

Moreover, we set up the gamification approach previously described in this chapter on the 9 videos employed for the eye-tracking experiments, in order to perform a comparison between on the two strategies of human involvement.

Collected data

Each eye-gaze experiment took 2 minutes per subject, and provided on average 35.7 points per frame in 32-minute user engagement time.

In order to make the comparison fair, we had a certain number of participants (who had not been involved in the eye-tracking experiment and did not know the videos in advance) play the game until the resulting average number of clicks per frame was comparable to the number of gaze points. In the end, 12 subjects played the game experiment for approximately 9.5 minutes each, which resulted in the collection of 40.4 clicks per frame on average, corresponding to a total engagement time of 115 minutes.

This difference in the amount of time required to collect similar amounts of data was expected, due to the higher acquisition rate of the eye-tracker compared to human clicking speed.

4.3 Comparison

The accuracy of the baseline method and of the devised approach, using eye-gaze data and game click data, on the target dataset is shown in Table 4.4, in terms of F-measure at pixel level, averaged over frames. Some examples of segmentation outputs are illustrated in Figures 4.9, 4.10 and 4.11.

While both the proposed gaze-based algorithm and the game-based segmentation outperform the baseline, it can be seen that, although the number of clicks per frame were approximately the same, using players' clicks from the game yields a markedly better performance (shown in the "Clicks$_M$" columns in Table 4.4) than when using eye-tracking points as input source (column "Gaze"). The primary explanation for the lower performance of the eye-gaze–based approach lies in the noisy nature of eye movements: eye gaze, indeed, involves both fixations and saccades with the latter spanning the whole visual scene from fixation to fixation. Thus, the approach tended to perform fairly well in presence of isolated objects (fixations and saccades were close), while in case of multiple objects it failed, as also exemplified in Figures 4.9, 4.10 and 4.11.

However, there is an unfairness element which the simple count of average points per frame is not able to capture. In our experiment, game players were allowed to play the game for little less than 10 minutes, on average, which implied that they were able to play the game more than once. This is actually a big advantage, since playing multiple games provides the user with information on where objects are located and what to expect, and this is exactly the kind of tips that game players tend to exploit as much as they can in order to maximize their scores. The eye-tracking experiment participants, however, were allowed to watch the video only once. In order to estimate how much this factor affected the comparison, we also computed the results for the game taking into account only the clicks generated by the 16 subjects the first time they played the game. The achieved performance

Video	[14]	Gaze	Clicks$_S$	Clicks$_M$
animal_chase (VSB)	0.39	0.62	0.26	0.78
sled_dog_race (VSB)	0.42	0.52	0.53	0.76
tennis (VSB)	0.42	0.26	0.57	0.62
cheetah (SegT)	0.41	0.67	0.52	0.68
frog (SegT)	0.37	0.57	0.57	0.74
monkeydog (SegT)	0.36	0.43	0.40	0.64
camel01 (FBMS)	0.49	0.65	0.52	0.59
rabbits02 (FBMS)	0.37	0.70	0.72	0.89
rabbits04 (FBMS)	0.23	0.50	0.76	0.77
Average	0.38	0.55	0.54	0.72

Table 4.4: *F-measure scores obtained on the tested videos for [14] and the proposed method, using either eye-gaze data or user-click data. As for user-click we performed two evaluations: a) exploiting clicks of first-time-play by users in order to eliminate the bias due to prior knowledge on object location (column Clicks$_S$), and b) using all collected clicks (column Clicks$_M$).*

is shown in the column "Clicks$_S$" of Table 4.4: in this case, it is possible to notice how the comparison with the eye-gaze results is more balanced, thus indicating that object location prior is a key factor for accurate results.

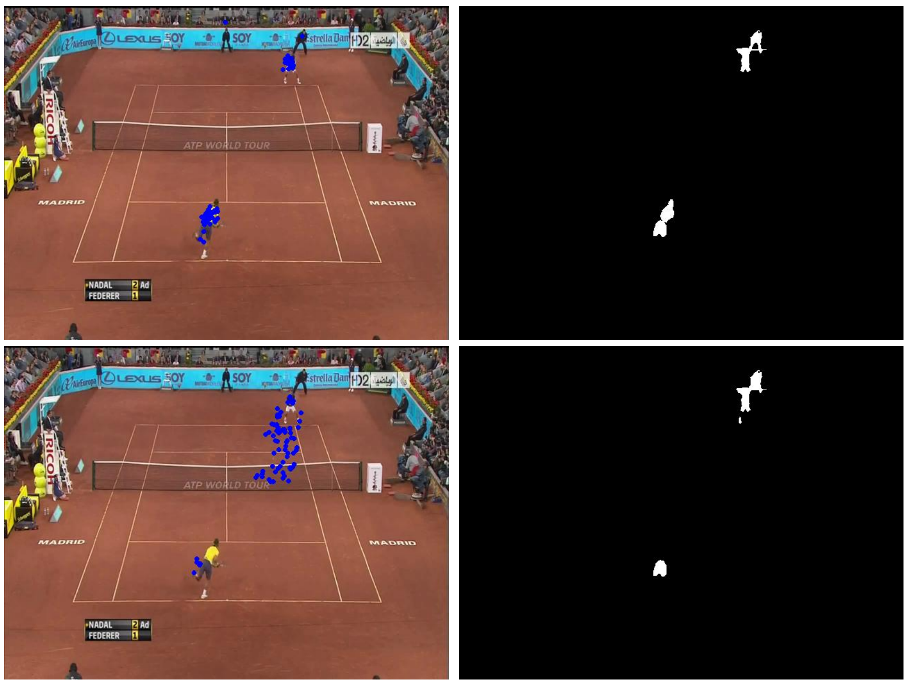

Figure 4.9: *Segmentation output examples. First row: user clicks from the game and segmentation output. Second row: eye-gaze points and segmentation output.*

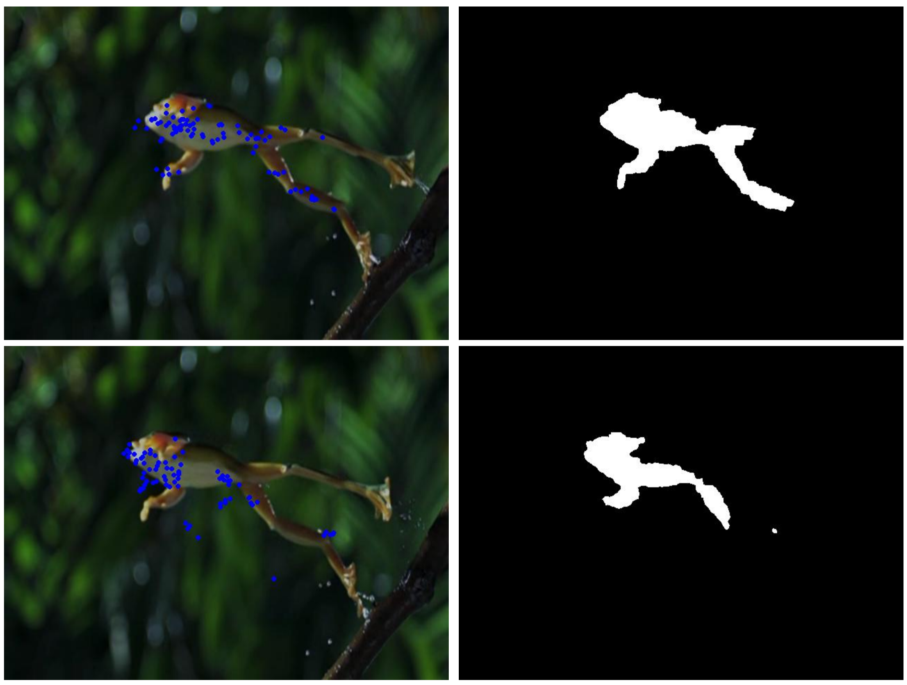

Figure 4.10: *Segmentation output examples. First row: user clicks from the game and segmentation output. Second row: eye-gaze points and segmentation output.*

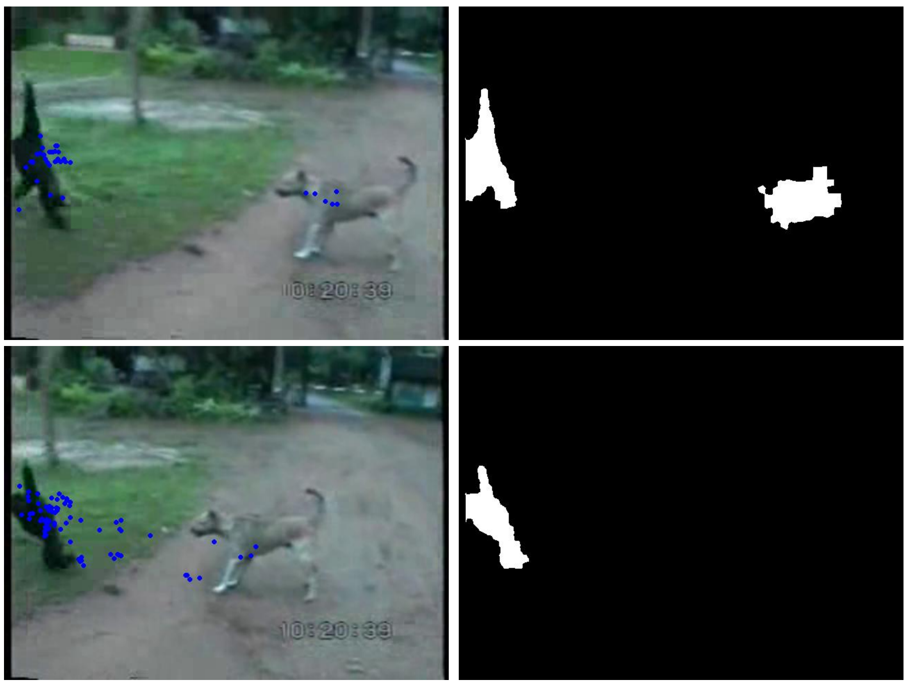

Figure 4.11: *Segmentation output examples. First row: user clicks from the game and segmentation output. Second row: eye-gaze points and segmentation output.*

4.4 Discussion

The interactive video object segmentation method described in this chapter allows to combine effectively the "games with a purpose" strategy with collaborative human efforts. In spite of the noisiness of the human input data, we are able to obtain a satisfactory accuracy by setting soft constraints on clicked superpixels and by employing a two-phase minimization, which performs a "smoothing" of superpixel labels in multiple frames, thus further reducing the impact of clicks on background superpixels. Unlike the fully-automatic method presented in Chapter 3, we chose to employ optical flow in this approach, at the cost of higher computation time. However, this was necessary in order to have a more precise linking between superpixels in time, which compensated the initial per-frame noisy segmentation. The performance analysis showed a gain over the automated approach, which can easily be attributed to the higher quality of players' annotations with respect to the analysis of motion superpixels in Chapter 3.

We then proceeded in comparing the performance of the gamification approach with those obtained when using a more implicit and bottom-up user involvement modality, by having subjects watch at videos and employing the recorded eye-gaze data as input to a similar automated algorithm as the one employed for user clicks from the game.

The average number of user clicks or gaze points per frame being equal, we found that the gamification approach achieves a higher accuracy than the eye-tracking–based one. An explanation for this difference has been suggested in Section 4.3, and lies in the higher noisiness of gaze data, which in turn is related to the lower degree of voluntary control of one's own eyes, compared to when clicking using a computer mouse.

Another aspect which biased the evaluation in favour of the gamification approach was the possibility of game players to watch the videos multiple times, as they played more and more, which allowed them to exploit the acquired prior knowledge to improve their own score and therefore the accuracy of the collected data.

However, two more factors are worth being taken into account: saliency direction and engagement time.

A critical difference in the two feedback collection modalities consists in how users' attention was directed in the experiments. In the eye-gaze experiment, bottom-up visual saliency was enforced since subjects were left free to analyze the scenes shown in the videos; however, in the game, user behaviour was driven by the reward mechanism imposed through the scoring system, which induced a top-down (or volition-controlled) visual saliency and limited the amount of "exploration" that the players would perform, by directing them explicitly towards moving objects.

Finally, it should be noted that the results computed for the game were based on a total interaction time of 115 minutes, versus a total of 32 minutes required of the eye-tracking participants. If we also take into account the difference in explicitness of the data gathering modalities (i.e., ideally, unlike playing the game, eye-tracking could have been performed in the background of whatever activity a user had been carrying out), the accuracy obtained using gaze data, albeit lower, is actually a very satisfactory and promising achievement in the way towards automated visual analysis through the employment of implicit eye-tracking data collection strategies.

4.5 Publications

A preliminary version of the eye-tracking–based segmentation approach was presented at ACM International Conference on Multimedia, Brisbane, Australia, 2015.

In September 2016, the gamification approach presented in this chapter has been accepted for publication on IEEE Transactions of Pattern Analysis and Machine Intelligence.

CHAPTER

FIVE

BRAIN-DRIVEN COMPUTATION FOR IMAGE CLASSIFICATION

In the previous chapter, we investigated the potential impact of human integration approaches into automated video segmentation methods. Our preliminary results found that implicit human feedback may provide an effective comprimise between the higher accuracy that can be achieved when using more explicit feedback sources and the higher engagement effort that such more explicit modalities demand of users when collecting the required data.

Encouraged by the results, we further pushed the limits of human feedback implicitness: switching to the problem of image classification, we explored whether it is possible to identify the type of the image that a person is looking at by analyzing the patterns of brain activity recorder through an EEG.

5.1 Introduction

No person can help but remain fascinated by the possibility of reading the mind. Unfortunately, no scientific evidence has been found that it is actually possible, but if genuine telepathy might be destined to be relegated to the realm of fiction, "artificial" telepathy may actually give scientists more satisfaction, as technology has allowed to record brain activity signals that can be connected to mental processes associated to specific tasks.

In this chapter, we focus on the role of electroencephalography (EEG) in uncovering the processes associated to visual analysis, thus reaching probably the highest level of implicitness that can be hoped for in the context of capturing human feedback for employment in automated computer vision methods.

The intuition at the foundation of this work is simple. Current computer vision methods analyze images by extracting numeric descriptors (*features*) of the represented content. Such features may be either hand-crafted (e.g., SIFT [26], HOG [106]) or learned by mathematical models (e.g., CNNs). However, the fact remains that human capabilities in understanding the visual world largely surpass machines'. Given that, instead of devising or having a model learn visual features, why not try to *copy* the features that humans use, by studying the relevant activity happening in the brain?

In the remainder of this chapter, we will see how this approach can be applied to the problem of image classification, i.e. the task of assigning a short label to an image describing its overall content.

5.2 Method

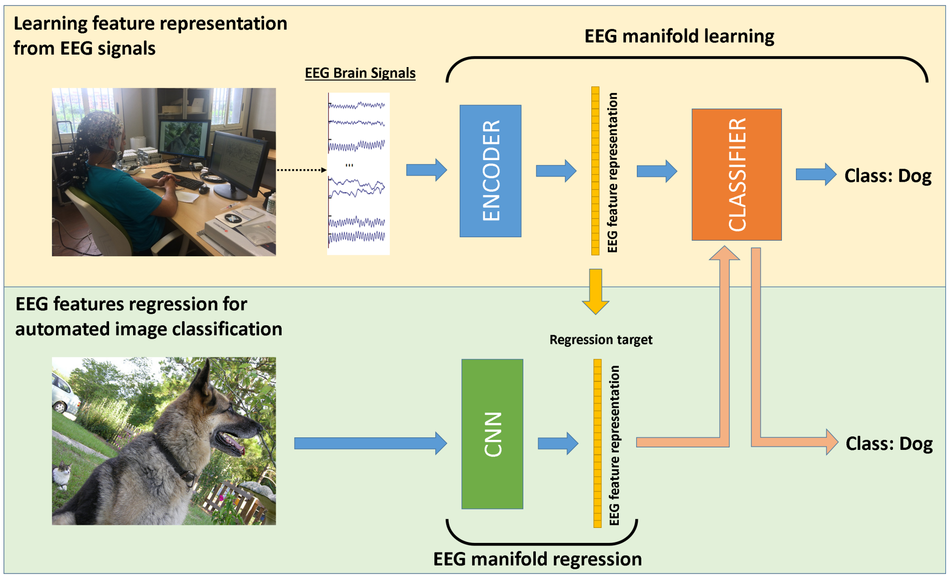
Learning feature representation from EEG signals
EEG manifold learning
EEG Brain Signals
ENCODER
EEG feature representation
CLASSIFIER
Class: Dog
EEG features regression for automated image classification
Regression target
CNN
EEG feature representation
Class: Dog
EEG manifold regression

Figure 5.1: *Overview of the proposed approach. Top: a low-dimensional representation for temporal EEG signals recorded while users looked at images is learned by the encoder module; the computed EEG features are employed to train an image classifier. Bottom: a CNN is trained to estimate EEG features directly from images; then, the classifier trained in the previous stage can be used for automated classification without the need of EEG data for new images.*

The work described in this paper relies on three key intuitions:

- EEG signals recorded while a subject looks at an image convey feature-level and cognitive-level information about the image content.
- A low-dimensional manifold within the multi-dimensional and

temporally-varying EEG signals exists and can be extracted to obtain a 1D representation which we refer to as *EEG features.*

- *EEG features* are assumed to mainly encode visual data, thus it is possible to extract the corresponding image features for automated classification.

These three ideas provide the design basis for the overall two-stage image classification architecture proposed in this work and shown in Figure 5.1.

The first stage of our approach aims to find a low-dimensional manifold within the two-dimensional (channels and time) EEG space, such that the representation within that manifold is discriminant over object classes. In order to learn this representation, we employed EEG data recorded while users looked at images on a screen. Then, we trained an *encoder* network (implemented through recurrent neural networks – RNNs – for temporal analysis) to extract *EEG features* from raw EEG signals; the training process is supervised by the class of the images for which each input EEG sequences were recorded, and a classifier for EEG features is jointly trained in the process.

Of course, it is unreasonable to assume the availability of EEG data for each image to be classified. Therefore, the second stage of the method aims at extracting *EEG features* directly from images, by learning a mapping from CNN image features to EEG features (learned through RNN encoder). After that, new images can be classified by simply estimating their *EEG features* through the trained CNN-based regressor and employ the stage-one classifier to predict the corresponding image class.

Number of classes	40
Number of images per class	50
Total number of images	2000
Visualization order	Sequential
Time for each image	0.5 s
Pause time between classes	10 s
Number of sessions	4
Session running time	350 s
Total running time	1400 s

Table 5.1: *The parameters of the experimental protocol.*

5.2.1 EEG data acquisition

Seven subjects (six male and one female) were shown visual stimuli of objects while EEG data was recorded. All subjects were homogeneous in terms of age, education level and cultural background and were evaluated by a professional physicist in order to exclude possible conditions (e.g., diseases) interfering with the acquisition process.

The dataset used for visual stimuli was a subset of ImageNet [29], containing 40 classes of easily recognizable objects[1]. During the experiment, 2,000 images (50 from each class) were shown in bursts for 0.5 seconds each. A burst lasts for 25 seconds, followed by a 10-second

[1]ImageNet classes used: dog, cat, butterfly, sorrel, capuchin, elephant, panda, fish, airliner, broom, canoe, phone, mug, convertible, computer, watch, guitar, locomotive, espresso, chair, golf, piano, iron, jack, mailbag, missile, mitten, bike, tent, pajama, parachute, pool, radio, camera, gun, shoe, banana, pizza, daisy and bolete (fungus)

pause where a black image was shown for a total running time of 1,400 seconds (23 minutes and 20 seconds). A summary of the adopted experimental paradigm is shown in Table 5.1.

The experiments were conducted using a 32-channel cap with passive, low-impedance electrodes distributed according to the 10-20 placement system. Three out of the 32 electrodes did not convey any useful information (ground, reference and an auxiliary electrode for removing heart-related artifacts) reducing the number of effective signals to 29. Brainvision[2] DAQs and amplifiers were used for the acquisition of the EEG signals. Sampling frequency and data resolution were set, respectively, to 250 Hz and 16 bits.

A notch filter (49-51 Hz) and a second-order band-pass Butterworth filter (low cut-off frequency 14 Hz, high cut-off frequency 71 Hz) were set up so that the recorded signal included the Beta (15-31 Hz) and Gamma (32-70 Hz) bands, as they convey information about the cognitive processes involved in the visual perception [107]. From each recorded EEG sequence, the first 10 samples (40 ms) for each image were discarded in order to exclude any possible interference from the previously shown image (i.e., to permit the stimulus to propagate from the retina through the optical tract to the primary visual cortex [108]). The following 110 (440 ms) samples were used for the experiments.

By using the protocol described above we acquired 14,000 (2,000 images for 7 subjects) multi-channel (29 channels) EEG sequences. In the following descriptions, we will refer to a generic input EEG sequence as $s(c, t)$, where c (from 1 to 29) indexes a channel and t

[2]http://www.brainvision.com/

(from 1 to 110) indexes a sample in time. We will also use the symbol $(\cdot)$ to indicate "all values", so $s(\cdot, t)$ represents the vector of all channel values at time t, and $s(c, \cdot)$ represents the whole set of time samples for channel c.

5.2.2 Learning EEG manifold

The first type of analysis aims at translating an input multi-channel temporal EEG sequence into a low dimensional feature vector summarizing the relevant content of the input sequence. Previous approaches [79, 83] simply concatenate time sequences from multiple channels into a single feature vector, ignoring temporal dynamics, which, instead, contains fundamental information for EEG activity analysis [79]. In order to include such dynamics in our representation, we employ LSTM recurrent neural networks [68], which are commonly used for the analysis of several kinds of sequence data, thanks to their capability to track long-term dependencies in the input data.

In an LSTM cell model, the cell input is modulated by an *input gate* i_t; the cell state is updated by combining the modulated cell input, the previous cell state (fed back through a fixed-weight connection called *constant error carousel*) and a *forget gate*, which allows to reset the cell state. Similarly, the cell output is computed by modulating the cell state through an *output gate*. Each gate implements a learnable biased linear combination between current cell input, previous cell state and previous cell output; in the case of a layer of cells, all cell states and outputs from the previous time step are fed to each cell in the current time step. A commonly-employed variant of the model, also used in this work, discards the cell state in the gate equations, thus producing the following LSTM update equations:

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + b_i\right) \tag{5.1}$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right) \tag{5.2}$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \tag{5.3}$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + b_o\right) \tag{5.4}$$

$$h_t = o_t \tanh\left(c_t\right) \tag{5.5}$$

where σ is the logistic sigmoid function, i_t, f_t, o_t are respectively the *input*, *forget* and *output* gates, c_t and h_t are the cell state and output (hidden) vectors, and the W matrices and b vectors are learnable parameters.

The top half of Figure 5.1 shows the general architecture of our EEG manifold representation model. The EEG multi-channel temporal signals, preprocessed as described in Section 5.2.1, are provided as input to an *encoder* module, which processes the whole time sequence and outputs an *EEG feature vector* as a compact representation of the input. Ideally, if an input sequence consists of the EEG signals recorded while looking at an image, our objective is to have the resulting output vector encode relevant brain activity information for discriminating different image classes. The encoder network is trained by adding, at its output, a classification module (in all our experiments, it will be a *softmax* layer), and using gradient descent to learn

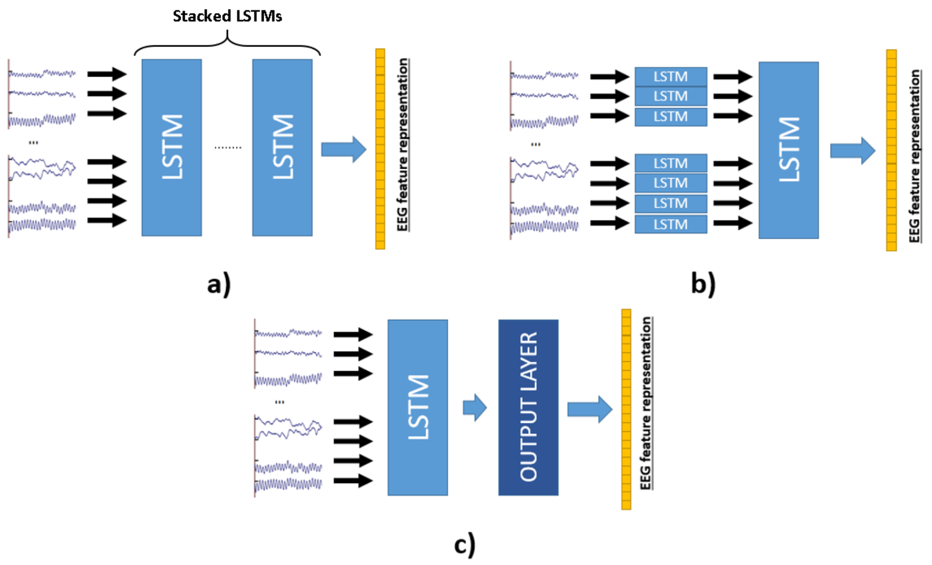
Stacked LSTMs
LSTM
LSTM
EEG feature representation
a)
LSTM
LSTM
EEG feature representation
b)
LSTM
OUTPUT LAYER
EEG feature representation
c)

Figure 5.2: *Tested encoder architectures: a) common LSTM; b) channel LSTM + common LSTM; c) common LSTM + output layer.*

the whole model's parameters end-to-end. In our experiments, we tested several configurations of the encoder network:

- *Common LSTM* (Figure 5.2a): the encoder network is made up of a stack of LSTM layers. At each time step t, the first layer takes the input $s(\cdot, t)$ (in this sense, "common" means that all EEG channels are initially fed into the same LSTM layer); if other LSTM layers are present, the output of the first layer (which may have a different size than the original input) is provided as input to the second layer and so on. The output of the deepest LSTM layer at the last time step is used as the EEG feature representation for the whole input sequence.

- *Channel LSTM + Common LSTM* (Figure 5.2*b*): the first encoding layer consists of several LSTMs, each connected to only one input channel: for example, the first LSTM processes input data $s(1, \cdot)$, the second LSTM processes $s(2, \cdot)$, and so on. In this way, the output of each "channel LSTM" is a summary of a single channel's data. The second encoding layer then performs inter-channel analysis, by receiving as input the concatenated output vectors of all channel LSTMs. As above, the output of the deepest LSTM at the last time step is used as the encoder's output vector.

- *Common LSTM + output layer* (Fig. 5.2*c*): similar to the *common LSTM* architecture, but an additional output layer (linear combinations of input, followed by ReLU nonlinearity) is added after the LSTM, in order to increase model capacity at little computational expenses (if compared to the two-layer common LSTM architecture). In this case, the encoded feature vector is the output of the final layer.

Encoder and classifier training is performed through gradient descent by providing the class label associated to the image shown while each EEG sequence was recorded. After training, the encoder can be used to generate EEG features from an input EEG sequences, while the classification network will be used to predict the image class for an input EEG feature representation, which can be computed from either EEG signals or images, as described in the next section.

5.2.3 CNN-based EEG manifold regression

In order to employ the RNN learned feature representation for general images, it is necessary to bypass the EEG recording stage and extract features directly from the image, which should be possible by our assumption that the learned EEG features reflect the image content which evoked the original EEG signals.We employed two CNN-based approaches to extract EEG features (or, at least, a close approximation) from an input image:

- *Fine-tuning.* The first approach is to train a CNN to map images to corresponding EEG feature vectors. Typically, the first layers of CNN attempt to learn the general (global) features of the images, which are common between many tasks, thus we initialize the weights of these layers using pre-trained models, and then learn the weights of last layers from scratch. In particular, we used the pre-trained AlexNet CNN [2], and modified it by replacing the softmax classification layer with a regression layer (containing as many neurons as the dimensionality of the EEG feature vectors), using Euclidean loss as objective function.
- *Deep feature extraction.* The second approach consists of extracting image features using pre-trained CNN models and then employ regression methods to map image features to EEG feature vectors. We used our fine-tuned AlexNet [2], GoogleNet [109] and VGG [110] as feature extractors by reading the output of the last fully-connected layer, and then applied several regression methods (namely, k-NN regression, ridge regression, random forest regression) to obtain the predicted feature vectors.

We opted to fine-tune only AlexNet, instead of GoogleNet and VGG, because these two CNNs contain more convolutional layers and, as such, they were more prone to overfitting given the relatively small dataset size. The resulting CNN-based regressor allows to extract the brain-learned features from any input image; the extracted features can then be fed to the classifier trained during EEG feature learning to perform automated visual classification.

5.3 Performance analysis

Performance analysis is split into three parts since our method consists of: 1) learning visual stimuli–evoked EEG data using RNN (implemented in Torch); 2) CNN-based regression to map images to RNN-learned EEG-based features (implemented in Caffe); 3) the combination of the above two steps to implement automated visual classifiers.

5.3.1 Learning EEG representations

We first tested the three architectures reported in Sect. 5.2.2 using our EEG dataset. Our dataset was split into training, validation and test sets, with respective fractions 80% (1600 images), 10% (200), 10% (200). We ensured that the signals generated by all participants for a single image are all included in a single split. All model architecture choices were taken only based on the results on the validation split, making the test split a reliable and "uncontaminated" quality indicator for final evaluations. The overall number of EEG sequences used for training the RNN encoder was 13,944 out of the available 14,000, since some of them were strongly affected by environmental noise.

Existing works, such as [111, 69], employing Support Vector Machines (SVM), Random Forests and Sparse Logistic Regression for learning EEG representation, cannot be employed as baseline since they do not operate on whole brain signals (but on feature vectors) and are applied to other tasks (e.g., music classification, seizure detection, etc.) than visual object–evoked EEG data.
Table 5.2 reports the achieved performance by the three encoder configurations with various architecture details. The classifier used to compute the accuracy is the one jointly trained in the encoder; we will use the same classifier (without any further training) also for automated visual classification on CNN-regressed EEG features. The proposed RNN-based approach was able to reach about 40% classification accuracy, which greatly outperforms the state-of-the-art performance of 29% achieved by [79], with fewer image classes (12 against 40 of our work).

To further contribute to the research on how visual scenes are processed by the human brain, we investigated how image visualization times may affect classification performance. Thus far, it has been known that feature extraction for object recognition in humans happens during the first 50-120 ms [108] (stimuli propagation time from the eye to the visual cortex), whereas less is known after 120 ms. Since in our experiments, we displayed each image for 500 ms; we evaluated classification performance in different visualization time ranges, i.e., [40–480 ms], [40–160 ms], [40–320 ms] and [320–480 ms]. Table 5.3 shows the achieved accuracies when using the RNN model which obtained the highest validation accuracy (see Table 5.2), i.e., the common 128-neuron LSTM followed by the 128-neuron output layer. Contrary to what was expected, the best performance was obtained in the time

Model	**Details**	**Max VA**	**TA at max VA**
Common	64 common	34.8%	34.4%
	128 common	37.6%	36.5%
	64,64 common	37.8%	38.2%
	128,64 common	39.0%	36.7%
	128,128 common	39.2%	37.8%
Channel + Common	5 channel, 32 common	34.2%	31.9%
	5 channel, 64 common	34.9%	36.6%
Common + output	128 common, 64 output	38.3%	34.4%
	128 common, 128 output	**40.1%**	**35.8%**

Table 5.2: *Maximum validation accuracy ("Max VA") and corresponding test accuracy ("TA at max VA") for different configurations of the three RNN architectures shown in Sect. 5.2.2. The model yielding the best validation results is in bold.*

range [320–480 ms], instead of during the first 120 ms. This suggests that a key role in visual classification may be played by neural processes outside the visual cortex that are activated after initial visual recognition and might be responsible for the conception part mentioned in the introduction. Of course, this needs further and deeper investigations that are outside the scope of this paper.

5.3.2 CNN-based regression

CNN-based regression aims at projecting visual images onto the learned EEG manifold. According to the results shown in the previous section, the best encoding performance is obtained given by the common 128-neuron LSTM followed by the 128-neuron output layer.

Visualization time	**Max VA**	**TA at max VA**
40-480 ms	40.1%	35.8%
40-160 ms	38.1%	32.2%
40-320 ms	39.3%	35.4%
320-480 ms	**41.8%**	**35.9%**

Table 5.3: *Classification accuracy achieved by the RNN encoder using different portions of EEG signal data. Best results in bold.*

This implies that our regressor takes as input single images and provides as output a 128-feature vector, which should ideally resemble the one learned by encoder.

To test the regressor performance, we used the ImageNet subset presented in Section 5.2.1 and the same image splits employed for the RNN encoder. However, unlike the encoder training stage, where different subjects generated different EEG signal tracks even when looking at the same image, for CNN-based regression we require that each image be associated to only one EEG feature vector, in order to avoid "confusing" the network by providing different target outputs for the same input. We tested two different approaches for selecting the single feature vector associated to each image:

- *average*: the EEG feature vector associated to an image is computed as the average over all subjects when viewing that image.

- *best features*: for each image, the associated EEG feature vector is the one having the smallest classification loss over all subjects during RNN encoder training.

Table 5.4 shows the mean square error (MSE) obtained with each of the tested regression approaches. The lowest-error configuration, i.e., feature extraction with GoogleNet combined to k-NN regressor, was finally employed as EEG feature extractor from arbitrary images. Note that the accuracy values for *average* are markedly better than the *best features*' one. This is in line with the literature on cognitive neuroscience, for which changes in EEG signals elicited by visual object stimuli are typically observed when averaging data from multiple trials and subjects [83].

5.3.3 Automated visual classification

Our automated visual classifier consists of the combination of the CNN-based feature regressor achieving the lowest MSE (GoogleNet features with k-NN regressor, trained on *average* features) with the softmax classifier trained during EEG manifold learning. We evaluated image classification performance on the images from our dataset's test split, which were never used in either EEG manifold learning or CNN-based feature regression, obtaining a mean classification accuracy of 85.1%, which, albeit lower than state-of-the-art CNN performance[3], demonstrates the effectiveness of our approach.
In order to test the generalization capabilities of our brain-learned features, we also performed an evaluation of the proposed method as a feature extraction technique, and compared it to using VGG and GoogleNet (we did not test AlexNet given its lower performance as shown in Table 5.4) as feature extractors. We tested the three approaches on a 30-class subset of Caltech-101 [112] (chosen so as to

[3]http://image-net.org/challenges/LSVRC/2015/results

Regressor		**Feature set** Average	**Feature set** Best
AlexNet FT		2.0	2.4
AlexNet FE	k-NN	1.8	2.1
	Ridge	1.7	1.8
	RF	1.7	1.9
GoogleNet	k-NN	**0.8**	3.8
	Ridge	2.0	7.8
	RF	1.1	4.2
VGG	k-NN	0.9	3.8
	Ridge	1.7	7.2
	RF	1.1	4.1

Table 5.4: *Mean square error (MSE) values obtained by different regression methods for extracting EEG features from images. "FT": fine-tuned; "FE": feature extractor. Best performance underlined and in bold.*

avoid overlap with the classes used for developing our model) by training separate SVM classifiers and comparing by classification accuracy. The results are reported in Table 5.5.

Although our approach achieves lower accuracy than the compared models, it is actually an impressive result, considering that VGG and GoogleNet were trained on ImageNet, which is basically a superset of Caltech-101, while our EEG encoder and regressor were trained not only on a different set of object classes, but mainly on a feature space not even directly related to visual features.

GoogleNet	**VGG**	**Our method**
92.6%	80.0%	69.3%

Table 5.5: *Classification accuracy achieved when using GoogleNet, VGG and the proposed method as image feature extractors for training an SVM classifier on a subset of Caltech-101.*

5.4 Discussion

In this chapter, we investigated the possibility of extracting information related to the content of an image shown from EEG tracks recorded while showing that image to a subject. Our approach consisted first in learning a representation which mapped the high-dimensional, temporal, raw EEG signals into a compact feature vector encoding information relevant to the classification task; then, the learnt manifold was taught to a CNN model so that the EEG-inspired features could be extracted from any image, without having to record the corresponding EEG signals from a user.

The results are very promising in both aspects of the work. The EEG features extracted from raw signals (i.e. with no additional visual input) allow to achieve a 35.8% correct classification rate over 40 classes, thus improving the previous state-of-the-art result obtained by [79] (on a different dataset with 12 classes).

The learnt CNN regression model allows to extract EEG features from images, without the need of undergoing the EEG recording protocol, which makes the method fully suitable for automatic image analysis.

Finally, the feature themselves demonstrate generalization capabilities and are able to obtain satisfactory results in the classification of images from different categories than the ones used to train the models.

The promising results achieved in this initial work make us hope that human brain processes involved in visual recognition can be effectively decoded for further inclusion into automated methods.

Moreover, we demonstrated how this kind of human feedback, while absolutely implicit and out of direct control by a subject, conveys extremely meaningful and usable information, and represents a potential step forward towards intedisciplinary research across computer vision, machine learning and cognitive neuroscience for understanding how this information can be exploited to gain further insight on vision and on its applications to computer vision.

CHAPTER

SIX

CONCLUSIONS

In this thesis, we delved into the study of several modalities for harnessing human vision capabilities to support automated computer vision tasks. We tried to go deeper and deeper in the investigation of *implicit* or *bottom-up* interaction strategies, where the human component is not treated as a "black box" which magically solves task that our algorithms cannot handle, but rather as a resource, whose internal workings are largely unknown to us and currently impossibile to replicate with sufficient approximation, but whose collateral feedback "signals" (from voluntary clicks to partially-voluntary gaze movements to completely uncontrolled brain activity measurements) can be studied and decoded to both support automated methods directly, and learn human-inspired representations of visual phenomena.

Our journey started with a fully-automated method for video object segmentation based on energy minimization of superpixel background/foreground labeling. The innovative aspect of this work lay in the inclusion of energy potentials inspired by the Gestalt perceptual organization principles of *attachment*, *similarity*, *continuity* and *symmetry*, which are believed to encode human criteria for recognizing structure in the real-world contexts, and which proved effective in enhancing the performance of the devised method.

We then went on to explore strategies which involved people concretely. Gamification is naturally one of the most interesting participation strategies, thanks to its attractivity to occasional players and to the possibility of clearly defining the interaction modality in order to support a certain application. However, this possibility is also its main limitation, from our point of view, since it guides and constrains the user to perform a certain activity in a top-down fashion, without leaving room for individual bottom-up initiatives — in addition to being a rather demanding modality on user time, concentration and effort. Therefore, the natural "antagonist" of such approach had to be based on something which provided a similar kind of punctual feedback to make comparison fair, but where any top-down guidance had been removed, and all generated feedback data was a sole reaction to the visual stimulus: in our case, that "something" was eye gaze data. Our comparison between the two involvement strategies, both combined with similar segmentation methods to reduce any bias due to the automated components, showed that gamification allows to gather more high-quality — but still noisy — feedback (where the concept of "quality" is relative to the capability of the automated method to use it), which positively affected the segmentation accuracy. Eye-gaze–based methods, however, provided promising results: segmentation accuracy, while not as good as that achieved through gamification, still outperformed more complex fully-automated segmentation approaches. Besides, eye-tracking requires a kind of involvement much less intense and demanding from the user than gamification, in terms of time, concentration and effort; therefore, as technology advances and eye-tracking devices become more portable and integrated into personal computers and smartphones, it is easy to envisage that this kind of approaches will attract more and more interest from the research community.

Finally, we adventured into the realm of human mind — or, at least, the part of human mind that is accessible to us by means of brain activity responses recorded by EEG. We developed an approach for mapping raw EEG signals into a compact vector representation encoding visual information useful to identify the category of the image being watched by the subject. Our method was able to outperform the current state of the art of image classification from EEG data only, by increasing both the achieved accuracy and the number of object categories taken into account. Moreover, in order to provide a fully-automated variant of the method, we trained a CNN model to project images onto the same brain-inspired representation space, thus allowing to perform classification also on images for which no EEG signals are available.

All in all, the techniques presented in this thesis demonstrated how useful human involvement can be to support computer vision methods, not only as a "work force" but, much more importantly, as a source of knowledge. Even if we may not know how to replicate this knowledge yet, we now know that can tap it by studying and analyzing the variegated kinds of feedback that people provide when performing an activity requiring visual analysis, whether with supervised guidance or freely during everyday routines — of course, with proper technological support.

The potential for future developments of this kind of human-based computation systems is huge (and not even limited to computer vision). In particular, we intend to focus on brain-activity analysis for supporting computer vision tasks: for example, the combination of gaze data and brain signals has an immediate application to attention prediction and saliency detection. However, much more fascinating challenges lie in the “reading the mind” direction. Since understanding the content of an image, if generically, seems to be possible, it would be interesting to explore the level of visual detail encoded in EEG tracks or in functional MRI images, and see how feasible it is to reconstruct an entire image from brain activity patterns only. Maybe this is a naïve hope, and we will find out that the amount of information that we can access using this kind of machinery is too coarse for achieving the goal. Or maybe, dream recording is not that far away in the future...

BIBLIOGRAPHY

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances In Neural Information Processing Systems*, pp. 1–9, 2012.

[3] N. Rusk, "Deep learning," *Nature Methods*, vol. 13, no. 1, pp. 35–35, 2015.

[4] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for Everyone," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, USA), jun 2016.

[5] S. Brutzer, B. Höferlin, and G. Heidemann, "Evaluation of background subtraction techniques for video surveillance," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1937–1944, jun 2011.

[6] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *International Conference on Pattern Recognition (ICPR)*, (Cambridge, UK), 2004.

[7] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 246–252, 1999.

[8] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1778–1792, 2005.

[9] O. Barnich and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *IEEE Transactions on Image Processing*, vol. 20, pp. 1709–1724, jun 2011.

[10] C. Spampinato, S. Palazzo, and I. Kavasidis, "A Texton-based Kernel Density Estimation Approach for Background Modeling under Extreme Conditions," *Computer Vision and Image Understanding*, vol. 122, pp. 74–83, 2014.

[11] B. Han and L. S. Davis, "Density-based multifeature background subtraction with support vector machine," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 1017–1023, 2012.

[12] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikainen, and S. Z. Li, "Modeling pixel process with scale invariant local patterns

for background subtraction in complex scenes," *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1301–1306, 2010.

[13] B. Zhang, Y. Gao, S. Zhao, and B. Zhong, "Kernel similarity modeling of texture pattern flow for motion detection in complex background," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 1, pp. 29–38, 2011.

[14] A. Papazoglou and V. Ferrari, "Fast object segmentation in unconstrained video," in *IEEE International Conference on Computer Vision*, (Sidney, Australia), pp. 1777–1784, 2013.

[15] J. Lim and B. Han, "Generalized Background Subtraction Using Superpixels with Label Integrated," in *IEEE European Conference on Computer Vision (ECCV)*, pp. 173–187, 2014.

[16] C. Liu, *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis.* PhD thesis, MIT.

[17] R. B. Fisher, Y.-H. Chen-Burger, D. Giordano, L. Hardman, and F.-P. Lin, *Fish4Knowledge: Collecting and Analyzing Massive Coral Reef Fish Video Data*, vol. 104. Springer, 2016.

[18] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, p. 1, 2016.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Net-

works," in *Conference on Neural Information Processing Systems (NIPS)*, (Montreal, Canada), 2015.

[20] O. Chapelle, P. Haffner, and V. Vapnik, "SVMs for Histogram-Based Image Classification," tech. rep., 1999.

[21] S. Lazebnik and C. Schmid, "Beyond Bags of Features : Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (New York, USA), pp. 2169–2178, 2006.

[22] C. S. Venegas-Barrera and J. Manjarrez, "Patrones espaciales de la riqueza especifica de las culebras Thamnophis en Mexico," *Revista Mexicana de Biodiversidad*, vol. 82, no. 1, pp. 179–191, 2011.

[23] L. Fei-Fei and P. Perona, "A Bayesian Hierarchical Model for Learning Natural Scene Categories," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, (San Diego, USA), pp. 524–531, 2005.

[24] T. Serre and T. Poggio, "Object Recognition with Features Inspired by Visual Cortex," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, (San Diego, USA), pp. 994–1000, 2005.

[25] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local Features and Kernels for Classication of Texture and Object Categories: A Comprehensive Study," *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[26] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, USA), 2016.

[28] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, USA), 2016.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Miami, USA), 2009.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and Others, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[31] C. Rother and V. Kolmogorov, “Grabcut: Interactive Foreground Extraction Using Iterated Graph Cuts,” *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[32] J. Wu, Y. Zhao, J.-Y. Zhu, S. Luo, and Z. Tu, “MILCut: A Sweeping Line Multiple Instance Learning Paradigm for Interactive Image Segmentation,” in *IEEE Conference on Computer*

Vision and Pattern Recognition (CVPR), (Columbus, USA), 2014.

[33] S. Vijayanarasimhan and K. Grauman, "Large-scale live active learning: Training object detectors with crawled data and crowds," *International Journal of Computer Vision*, vol. 108, pp. 97–114, apr 2014.

[34] B. Peng, L. Zhang, D. Zhang, and J. Yang, "Image segmentation by iterated region merging with localized graph cuts," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2527–2538, 2011.

[35] G. Druck, B. Settles, and A. McCallum, "Active learning by labeling features," in *Conference on Empirical Methods in Natural Language Processing*, (Singapore), 2009.

[36] I. Kavasidis, C. Spampinato, and D. Giordano, "Generation of ground truth for object detection while playing an online game: Productive gaming or recreational working?," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, (Portland, USA), pp. 694–699, jun 2013.

[37] V. Badrinarayanan, F. Galasso, and R. Cipolla, "Label propagation in video sequences," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (San Francisco, USA), pp. 3265–3272, 2010.

[38] P. Ochs and T. Brox, "Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions," in *IEEE International Conference on Computer Vision (ICCV)*, (Barcelona, Spain), pp. 1583–1590, 2011.

[39] V. Badrinarayanan, I. Budvytis, and R. Cipolla, "Mixture of trees probabilistic graphical model for video segmentation," *International Journal of Computer Vision*, vol. 110, no. 1, pp. 14–29, 2014.

[40] A. Fathi, M. F. Balcan, X. Ren, and J. M. Rehg, "Combining Self Training and Active Learning for Video Segmentation," in *British Machine Vision Conference*, (Dundee, UK), 2011.

[41] B. L. Price, B. S. Morse, and S. Cohen, "LIVEcut: Learning-based interactive video segmentation by evaluation of multiple propagated cues," in *IEEE International Conference on Computer Vision (ICCV)*, (Kyoto, Japan), pp. 779–786, 2009.

[42] I. Budvytis, V. Badrinarayanan, and R. Cipolla, "Semi-supervised video segmentation using tree structured graphical models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2257–2264, 2011.

[43] Y. Li, J. Sun, and H.-Y. Shum, "Video object cut and paste," *ACM Transactions on Graphics*, vol. 24, no. 3, p. 595, 2005.

[44] N. Shankar Nagaraja, F. R. Schmidt, and T. Brox, "Video Segmentation With Just a Few Strokes," in *IEEE International Conference on Computer Vision (ICCV)*, (Santiago, Chile), pp. 3235–3243, dec 2015.

[45] S. D. Jain and K. Grauman, "Supervoxel-Consistent Foreground Propagation in Video," in *IEEE European Conference on Computer Vision (ECCV)* (D. Fleet, T. Pajdla, B. Schiele,

and T. Tuytelaars, eds.), Lecture Notes in Computer Science, (Zurich, Switzerland), Springer International Publishing, 2014.

[46] J. Donahue and K. Grauman, "Annotator rationales for visual recognition," in *IEEE International Conference on Computer Vision (ICCV)*, (Barcelona, Spain), Ieee, nov 2011.

[47] S. Maji, "Discovering a lexicon of parts and attributes," in *IEEE European Conference on Computer Vision (ECCV)*, (Florence, Italy), 2012.

[48] A. Vedaldi, S. Mahendran, R. Girshick, J. Kannala, M. B. Blaschko, D. Weiss, N. Saphra, S. Tsogkas, S. Maji, B. Taskar, K. Simonyan, and S. Mohamed, "Understanding Objects in Detail with Fine-grained Attributes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Columbus, USA), 2014.

[49] J. Deng, J. Krause, and L. Fei-Fei, "Fine-grained crowdsourcing for fine-grained recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Portland, USA), 2013.

[50] L. von Ahn and L. Dabbish, "Designing games with a purpose," *Communications of the ACM*, vol. 51, p. 57, aug 2008.

[51] D. Morrison, S. Marchand-Maillet, and É. Bruno, "Tag-Captcha," in *ACM SIGKDD Workshop on Human Computation*, (New York, USA), 2009.

[52] S. Hacker and L. von Ahn, "Matchin," in *ACM International Conference on Human Factors in Computing Systems*, (New York, USA), 2009.

[53] L. Von Ahn, S. Ginosar, M. Kedia, R. Liu, and M. Blum, "Improving accessibility of the web with a computer game," in *ACM International Conference on Human Factors in Computing Systems*, CHI '06, (New York, USA), ACM, 2006.

[54] L. Von Ahn and L. Dabbish, "Labeling Images with a Computer Game," in *ACM International Conference on Human Factors in Computing Systems*, (Boston, USA), 2004.

[55] L. von Ahn, R. Liu, and M. Blum, "Peekaboom," in *ACM International Conference on Human Factors in Computing Systems*, (New York, USA), 2006.

[56] M. Addis, W. Bailer, L. Boch, F. Gallo, and R. Wright, "100 Million Hours of Audiovisual Content : Digital Preservation and Access in the PrestoPRIME Project Categories and Subject Descriptors," in *ACM International Digital Preservation Interoperability Framework Symposium*, (Dresden, Germany), ACM, mar 2010.

[57] K. Siorpaes and M. Hepp, "OntoGame: Weaving the semantic web by online games," in *European Semantic Web Conference*, (Tenerife, Spain), 2008.

[58] A. L. Yarbus, "Eye movements and vision," *Neuropsychologia*, vol. 6, no. 4, p. 222, 1967.

[59] R. J. K. Jacob, “Eye Tracking in Advanced Interface Design,” in *Virtual Environments and Advanced Interface Design*, pp. 258–290, Oxford University Press on Demand, 1995.

[60] M. Schiessl, S. Duda, A. Thölke, and R. Fischer, “Eye tracking and its application in usability and media research,” *MMI-interaktiv Journal*, no. July, pp. 1–10, 2003.

[61] T. Walber, A. Scherp, and S. Staab, “Can you see it? Two novel eye-tracking-based measures for assigning tags to image regions,” in *Advances in Multimedia Modeling*, vol. 7732 LNCS, pp. 36–46, 2013.

[62] A. Faro, D. Giordano, C. Pino, and C. Spampinato, “Visual Attention for Implicit Relevance Feedback in a Content Based Image Retrieval,” in *ACM Symposium on Eye-Tracking Research and Applications*, (Safety Harbor, USA), pp. 73–76, 2010.

[63] G. Buscher, A. Dengel, and L. van Elst, “Eye movements as implicit relevance feedback,” in *ACM International Conference on Human Factors in Computing Systems*, (New York USA), 2008.

[64] A. Mishra, Y. Aloimonos, and C. L. Fah, “Active segmentation with fixation,” in *IEEE International Conference on Computer Vision (ICCV)*, (Kyoto, Japan), 2009.

[65] K. Shanmuga Vadivel, T. Ngo, M. Eckstein, and B. S. Manjunath, “Eye Tracking Assisted Extraction of Attentionally Important Objects From Videos,” in *IEEE International Conference*

on Computer Vision and Pattern Recognition (CVPR), (Boston, USA), 2015.

[66] H. Cecotti and A. Graser, “Convolutional Neural Networks for P300 Detection with Application to Brain-Computer Interfaces,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 433–445, 2011.

[67] P. Mirowski, D. Madhavan, Y. LeCun, and R. Kuzniecky, “Classification of patterns of EEG synchronization for seizure prediction,” *Clinical Neurophysiology*, vol. 120, pp. 1927–1940, nov 2009.

[68] S. Hochreiter, S. Hochreiter, J. Schmidhuber, and J. Schmidhuber, “Long short-term memory.,” *Neural computation*, vol. 9, no. 8, pp. 1735–80, 1997.

[69] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, “Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks,” *arXiv preprint*, 2015.

[70] S. Stober, A. Sternin, A. M. Owen, and J. A. Grahn, “Deep Feature Learning for EEG Recordings,” in *arXiv preprint*, 2015.

[71] Z. Kourtzi and N. Kanwisher, “Cortical Regions Involved in Perceiving Object Shape,” *Journal of Neuroscience*, vol. 20, pp. 3310–3318, may 2000.

[72] H. P. Op de Beeck, K. Torfs, and J. Wagemans, “Perceived shape similarity among unfamiliar objects and the organization

of the human object vision pathway.," *Journal of Neuroscience*, vol. 28, no. 40, pp. 10111–10123, 2008.

[73] M. V. Peelen and P. E. Downing, "The neural basis of visual body perception.," *Nature reviews. Neuroscience*, vol. 8, pp. 636–48, aug 2007.

[74] K. Das, B. Giesbrecht, and M. P. Eckstein, "Predicting variations of perceptual performance across individuals from neural activity using pattern classi fiers," *NeuroImage*, vol. 51, no. 4, pp. 1425–1437, 2010.

[75] C. Wang, S. Xiong, X. Hu, L. Yao, and J. Zhang, "Combining features from ERP components in single-trial EEG for discriminating four-category visual objects," *Journal of Neural Engineering*, vol. 9, oct 2012.

[76] P. Shenoy and D. S. Tan, "Human-aided computing: Utilizing implicit human processing to classify images," in *ACM International Conference on Human Factors in Computing Systems*, (Florence, Italy), 2008.

[77] T. A. Carlson, H. Hogendoorn, R. Kanai, J. Mesik, and J. Turret, "High temporal resolution decoding of object position and category.," *Journal of Vision*, vol. 11, no. 9, pp. 1–17, 2011.

[78] T. Carlson, D. Tovar, A. Alink, and N. Kriegeskorte, "Representational dynamics of object vision: The first 1000 ms," *Journal of Vision*, vol. 13, no. 10, pp. 1–19, 2013.

[79] B. Kaneshiro, M. Perreau Guimaraes, H.-S. Kim, A. M. Norcia, and P. Suppes, “A Representational Similarity Analysis of the Dynamics of Object Processing Using Single-Trial EEG Classification,” *PloS one*, vol. 10, no. 8, 2015.

[80] I. Simanova, M. van Gerven, R. Oostenveld, and P. Hagoort, “Identifying object categories from event-related EEG: Toward decoding of conceptual representations,” *PLoS one*, vol. 5, no. 12, 2010.

[81] N. Bigdely-Shamlo, A. Vankov, R. R. Ramirez, and S. Makeig, “Brain activity-based image classification from rapid serial visual presentation,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, no. 5, pp. 432–441, 2008.

[82] A. Kapoor, P. Shenoy, and D. Tan, “Combining brain computer interfaces with vision for object categorization,” in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, (Anchorage, USA), 2008.

[83] A. X. Stewart, A. Nuthmann, and G. Sanguinetti, “Single-trial classification of EEG in a visual object task using ICA and machine learning,” *Journal of Neuroscience Methods*, vol. 228, pp. 1–14, 2014.

[84] E. Mohedano, G. Healy, K. McGuinness, X. Giro-i Nieto, N. E. O’Connor, and A. F. Smeaton, “Object Segmentation in Images using EEG Signals,” in *ACM International Conference on Multimedia*, (Orlando, USA), 2014.

[85] C. Cheng, A. Koschan, C. H. Chen, D. L. Page, and M. A. Abidi, "Outdoor scene image segmentation based on background recognition and perceptual organization," *IEEE Transactions on Image Processing*, vol. 21, pp. 1007–1019, mar 2012.

[86] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg, "Video segmentation by tracking many figure-ground segments," in *IEEE International Conference on Computer Vision (ICCV)*, (Sydney, Australia), 2013.

[87] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian, "Foreground object detection from videos containing complex background," in *ACM international conference on Multimedia*, (Berkeley, USA), ACM, ACM Press, 2003.

[88] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2281, 2012.

[89] V. Kolmogorov and R. Zabih, "What Energy Functions can be Minimized via Graph Cuts?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, 2004.

[90] J. Yao and J.-M. Odobez, "Multi-Layer Background Subtraction Based on Color and Texture," in *IEEE International Conference on Computer Vision and Pattern Recognition, Workshop on Visual Surveillance (CVPR-VS)*, (Minneapolis, USA), IEEE, 2007.

[91] J. L. Yong, J. Kim, and K. Grauman, "Key-segments for video object segmentation," in *IEEE International Conference on Computer Vision (ICCV)*, (Barcelona, Spain), 2011.

[92] D. Zhang, O. Javed, and M. Shah, "Video Object Segmentation through Spatially Accurate and Temporally Dense," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, (Portland, USA), IEEE Computer Society, 2013.

[93] M. Narayana, A. Hanson, and E. Learned-Miller, "Background modeling using adaptive pixelwise kernel variances in a hybrid feature space," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, (Providence, USA), 2012.

[94] D. Giordano, F. Murabito, S. Palazzo, and C. Spampinato, "Superpixel-Based Video Object Segmentation Using Perceptual Organization and Location Prior," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, (Boston, USA), 2015.

[95] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention," *Vision Research*, vol. 40, no. 10-12, pp. 1489–1506, 2000.

[96] A. C. Oei and M. D. Patterson, "Enhancing Cognition with Video Games: A Multiple Game Training Study," *PLoS one*, vol. 8, no. 3, pp. 1–16, 2013.

[97] A. Jain, R. Bansal, A. Kumar, and K. D. Singh, "A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students," *International Journal of Applied and Basic Medical Research*, vol. 5, no. 2, pp. 124–127, 2015.

[98] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari, "Learning Object Class Detectors from Weakly Annotated Video," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, (Providence, USA), pp. 3282–3289, 2012.

[99] P. Ochs, J. Malik, and T. Brox, "Segmentation of moving objects by long term video analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 1187–1200, jun 2014.

[100] F. Galasso, N. S. Nagaraja, T. J. Cardenas, T. Brox, and B.Schiele, "A Unified Video Segmentation Benchmark: Annotation, Metrics and Analysis," in *IEEE International Conference on Computer Vision (ICCV)*, (Sydney, Australia), dec 2013.

[101] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "A mobile vision system for robust multi-person tracking," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, (Anchorage, USA), 2008.

[102] L. Gorelick, F. R. Schmidt, and Y. Boykov, "Fast Trust Region for Segmentation," in *IEEE International Conference on*

Computer Vision and Pattern Recognition (CVPR), (Portland, USA), 2013.

[103] K. Tang, R. Sukthankar, J. Yagnik, and L. Fei-Fei, "Discriminative Segment Annotation in Weakly Labeled Video," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, (Portland, USA), 2013.

[104] M. Godec, P. M. Roth, and H. Bischof, "Hough-based tracking of non-rigid objects," in *IEEE International Conference on Computer Vision (ICCV)*, (Barcelona, Spain), 2011.

[105] Y. Zhang, X. Chen, J. Li, C. Wang, and C. Xia, "Semantic Object Segmentation via Detection in Weakly Labeled Video," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, (Boston, USA), 2015.

[106] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, (San Diego, USA), Ieee, 2005.

[107] E. Niedermeyer and F. H. L. da Silva, *Electroencephalography: basic principles, clinical applications, and related fields.* Lippincott Williams & Wilkins, 2005.

[108] J. R. Heckenlively and G. B. Arden, *Principles and practice of clinical electrophysiology of vision.* MIT press, 2006.

[109] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper

with convolutions," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, (Boston, USA), 2015.

[110] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint*, 2014.

[111] A. Subasi and M. Ismail Gursoy, "EEG Signal Classification Using PCA, ICA, LDA and Support Vector Machines," *Expert Systems with Applications*, vol. 37, no. 12, pp. 8659–8666, 2010.

[112] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 594–611, apr 2006.