THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (INP Toulouse)

Discipline ou spécialité :

Image, Information et Hypermédia

Présentée et soutenue par :

M. AXEL CARLIER

le mardi 30 septembre 2014

Titre :

COMPREHENSION DE CONTENUS VISUELS PAR ANALYSE CONJOINTE DU CONTENU ET DES USAGES

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (I.R.I.T.)

Directeur(s) de Thèse :

M. VINCENT CHARVILLAT

MME GÉRALDINE MORIN

Rapporteurs :

Mme JENNY BENOIS-PINEAU, UNIVERSITE BORDEAUX 1

M. WOLFGANG EFFELSBERG, UNIVERSITAT MANNHEIM

Membre(s) du jury :

M. OGE MARQUES, FLORIDA ATLANTIC UNIVERSITY, Président

M. FERRAN MARQUES, UNIV POLITECNICA DE CATALUNYA BARCELONA, Membre

Mme GÉRALDINE MORIN, INP TOULOUSE, Membre

M. VINCENT CHARVILLAT, INP TOULOUSE, Membre

M. WEI TSANG OOI, NATIONAL UNIVERSITY OF SINGAPORE, Membre

M. XAVIER GIRO, UNIV POLITECNICA DE CATALUNYA BARCELONA, Membre

Titre :

Compréhension des contenus visuels par analyse conjointe du contenu et des usages

Résumé :

Dans cette thèse, nous traitons de la compréhension de contenus visuels, qu'il s'agisse d'images, de vidéos ou encore de contenus 3D. On entend par compréhension la capacité à inférer des informations sémantiques sur le contenu visuel. L'objectif de ce travail est d'étudier des méthodes combinant deux approches : 1) l'analyse automatique des contenus et 2) l'analyse des interactions liées à l'utilisation de ces contenus (analyse des usages, en plus bref).

Dans un premier temps, nous étudions l'état de l'art issu des communautés de la vision par ordinateur et du multimédia. Il y a 20 ans, l'approche dominante visait une compréhension complètement automatique des images. Cette approche laisse aujourd'hui plus de place à différentes formes d'interventions humaines. Ces dernières peuvent se traduire par la constitution d'une base d'apprentissage annotée, par la résolution interactive de problèmes (par exemple de détection ou de segmentation) ou encore par la collecte d'informations implicites issues des usages du contenu. Il existe des liens riches et complexes entre supervision humaine d'algorithmes automatiques et adaptation des contributions humaines via la mise en oeuvre d'algorithmes automatiques. Ces liens sont à l'origine de questions de recherche modernes : comment motiver des intervenants humains ? Comment concevoir des scénarii interactifs pour lesquels les interactions contribuent à comprendre le contenu manipulé ? Comment vérifier la qualité des traces collectées ? Comment agréger les données d'usage ? Comment fusionner les données d'usage avec celles, plus classiques, issues d'une analyse automatique ? Notre revue de la littérature aborde ces questions et permet de positionner les contributions de cette thèse. Celles-ci s'articulent en deux grandes parties.

La première partie de nos travaux revisite la détection de régions importantes ou saillantes au travers de retours implicites d'utilisateurs qui visualisent ou acquièrent des contenus visuels. En 2D d'abord, plusieurs interfaces de vidéos interactives (en particulier la vidéo zoomable) sont conçues pour coordonner des analyses basées sur le contenu avec celles basées sur l'usage. On généralise ces résultats en 3D avec l'introduction d'un nouveau détecteur de régions saillantes déduit de la capture simultanée de vidéos de la même performance artistique publique (spectacles de danse, de chant etc.) par de nombreux utilisateurs.

La seconde contribution de notre travail vise une compréhension sémantique d'images fixes. Nous exploitons les données récoltées à travers un jeu, Ask'nSeek, que nous avons créé. Les interactions élémentaires (comme les clics) et les données textuelles saisies par les joueurs sont, comme précédemment, rapprochées d'analyses automatiques des images. Nous montrons en particulier l'intérêt d'interactions révélatrices des relations spatiales entre différents objets détectables dans une même scène. Après la détection des objets d'intérêt dans une scène, nous abordons aussi le problème, plus ambitieux, de la segmentation.

Title:

Combining Content Analysis with Usage Analysis to better understand visual contents

Abstract:

This thesis focuses on the problem of understanding visual contents, which can be images, videos or 3D contents. Understanding means that we aim at inferring semantic information about the visual content. The goal of our work is to study methods that combine two types of approaches: 1) automatic content analysis and 2) an analysis of how humans interact with the content (in other words, usage analysis).

We start by reviewing the state of the art from both Computer Vision and Multimedia communities. Twenty years ago, the main approach was aiming at a fully automatic understanding of images. This approach today gives way to different forms of human intervention, whether it is through the constitution of annotated datasets, or by solving problems interactively (e.g. detection or segmentation), or by the implicit collection of information gathered from content usages. These different types of human intervention are at the heart of modern research questions: how to motivate human contributors? How to design interactive scenarii that will generate interactions that contribute to content understanding? How to check or ensure the quality of human contributions? How to aggregate human contributions? How to fuse inputs obtained from usage analysis with traditional outputs from content analysis? Our literature review adresses these questions and allows us to position the contributions of this thesis.

In our first set of contributions we revisit the detection of important (or salient) regions through implicit feedback from users that either consume or produce visual contents. In 2D, we develop several interfaces of interactive video (e.g. zoomable video) in order to coordinate content analysis and usage analysis. We also generalize these results to 3D by introducing a new detector of salient regions that builds upon simultaneous video recordings of the same public artistic performance (dance show, chant, etc.) by multiple users.

The second contribution of our work aims at a semantic understanding of fixed images. With this goal in mind, we use data gathered through a game, Ask'nSeek, that we created. Elementary interactions (such as clicks) together with textual input data from players are, as before, mixed with automatic analysis of images. In particular, we show the usefulness of interactions that help revealing spatial relations between different objects in a scene. After studying the problem of detecting objects on a scene, we also adress the more ambitious problem of segmentation.

Remerciements

First, I would like to thank my advisors Vincent Charvillat and Géraldine Morin for their helpful guidance through my three years of PhD. All the meetings and informal discussions we had about research, teaching, and even personal life have undoubtedly allowed me to reach smoothly the end of my PhD.

I also want to thank the reviewers of this manuscript, Jenny Benois-Pineau and Wolfgang Effelsberg, for their numerous comments and constructive feedback.

I have started working in research before my PhD under the guidance of Wei Tsang Ooi in Singapore. I thank him for receiving me there, and making me part of his wonderful team, with Ravindra Guntur and Ngo Quang Minh Khiem (and later with Minhui and Shanghong). I really want to thank Wei Tsang for all he has taught me, in addition to introducing me to my PhD topic and being a constant help and very valuable collaborator during the last four years.

During the second part of my PhD, I have come to work closely with Oge Marques, Xavier Giro-i-Nieto and Amaia Salvador. Many thanks to them for the richness of our interactions. I sincerely hope we can keep this fruitful collaboration in the upcoming years.

I also want to thank all the people who made my time at the laboratory enjoyable. Many thanks to Jean-Denis, Pierre, Sylvie and Simone for their always welcome advices. I also want to thank all the Phd students and postdoctoral fellows: Benoit, Pauline, Viorica, Jérôme, Florent, Lilian, Rabih, Viet, Phuong, Yvain, Nicolas, Marianne, Vincent and Bastien for the good times (and coffees) we have shared over the years. I also want to specially thank Sylvie A., Sylvie E. and Audrey for all the help they have provided me regarding administrative matters.

I would also like to thank the many friends who have accepted to participate in some (for a few people, even in all) of the user studies that are presented in this manuscript. It is a true display of friendship to go through the sometimes boring tasks that I asked them to fulfill.

And finally, I deeply thank my parents who supported me through my entire studies: this PhD is also theirs. And of course, many thanks to Isabelle who is at the same time a confident, an advisor, a reviewer and a very supportive and understanding partner.

Contents

Chapter 1

Introduction

1.1 Understanding visual content

Understanding visual content is one of the key challenges that computer scientists have been facing for many years. In this Thesis, the term *visual content* is used to encompass the different forms of digital representations of the world that can be interpreted by the human eye. These representations are often distorted (e.g. by a perspective transformation), noisy, imprecise or even incomplete. In the field of computer vision, understanding visual content could be defined as "the attempt to infer properties of the world from its digital representations". In the multimedia paradigm, visual content is often augmented with correlated data of a different nature, such as audio (e.g. a video's soundtrack) or text (e.g. metadata). In its essence, understanding means inferring semantic knowledge from the visual content (Figure fig:intro-mca).

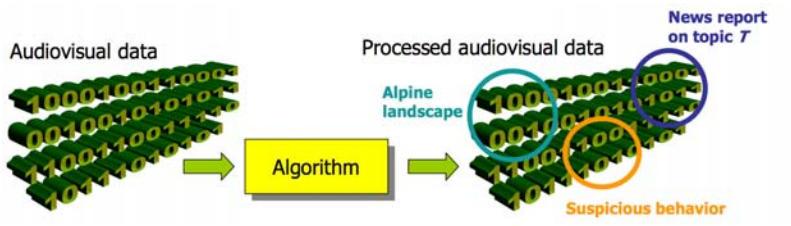


Figure 1.1: Multimedia Content Analysis

1.2 Historical Perspective

The original goal for visual content understanding was to design a system that could "make computers see". Applications were mainly focused on designing autonomous systems that could use an artificial vision to perform automatic tasks. These applications where therefore mostly targetting robotics. At the time, because of the limit in ressources (CPU, memory) and digital material available, researchers were concentrating their efforts on images. Consider for example the problem of object segmentation. Approaches focused on finding objects boundaries using edge detectors [Canny 1986], or on determining regions [Vincent 1991] that are candidate objects. Figure 1.2 shows the cover of a book on com-

puter vision from Ayache [Ayache 1991]. This figure describes an inside scene understanding algorithm that has clearly no semantic meaning, but rather a metrological purpose.

Figure 1.2: Cover photo of Nicholas Ayache's book (Artificial vision for mobile robots, [Ayache 1991])

This was in a nutshell the state of the art 20 years ago. The end of the nineties saw digital cameras become a common device. The first digital camera was invented in 1975 (by Steven Sasson in Kodak), but the market for digital cameras started to flourish in the mid-90s.This progress, parallel to the emergence of the Internet, allowed large scale photo sharing platforms to be created. Webshots (in 1999) and Yahoo Photos (2000) were the first Internet platforms where users could upload, share and exchange digital pictures. In 1997, the website `shareyourworld.com` became the first video sharing platform. It eventually shut down in 2001 due to bandwidth issues.

Ten years ago, in 2004, understanding visual content had a completely different meaning than in 1994. Because of the rising amount of available visual content, the challenges that researchers had to face shifted a little bit. Of course, the computer vision community along with newly formed multimedia community were still interested in the problems of designing effective systems for artifical vision and multimedia content analysis. But the growing quantity of images and videos introduced new problems : how to efficiently index all this data ? And then how to efficiently access the data ? Technical challenges such as streaming or retrieval became central and involve, at different scales, at least a partial understanding of the content. As for object segmentation, in ten years the researchers had considerably changed the way they approached the problem. The massive redundancy of visual content (e.g. images on Flickr) allowed for new formulations of difficult problems. For example, the co-segmentation problem is a variant of segmentation in which the images to segment contain a common object. Even reformulated, these problems remain hard and some efforts focused on techniques of interactive segmentation : since automatic algorithms often failed to differentiate one region from another, people looked for ways to have humans guide them

with simple interactions (GrabCut, 2004). Another branch of the community followed a general trend that had appeared during the nineties: supervised machine learning. The idea was to manually segment a set of images, then train algorithms to expand this information to other images.

The Berkeley BSDS300 (2001) was one of the early efforts made to gather annotations on a set of images. It was later followed by the PASCAL Visual Objects Challenge (VOC, [Everingham 2010]) which involved thousands of images, and has now made room for the ImageNet challenge, which contains hundreds of thousands of images. This evolution is representative of a general trend: since 2004 everything went bigger. Flickr, and then Facebook, became leaders for photo sharing. In 2013, there were 350 million pictures uploaded every day on Facebook[1]. In the mean time, Youtube led the development of video sharing. The statistics page on `www.youtube.com` claims that there are 100 hours of video uploaded on Youtube every minute.

These observations lead us to the present. Today in 2014, Big Data has become a trendy buzzword. The applications involving visual content are numerous, but we are nowhere near what scientists were rooting for 20 years ago: a complete and automatic understanding of the visual contents is still far from becoming reality. Artificial vision is still very far from achieving performances that are even comparable to the human vision system. Richard Szeliski stresses at the beginning of his recent book [Szeliski 2011] that "the dream of having a computer interpret an image at the same level as a two-year old remains elusive". Computers can achieve honorable performances in problems that are constrained and simple enough, for which a priori models are accurate enough. But mostly humans are much more efficient in problems that involve even a little bit of semantic knowledge.

It is also worthy to note that most applications of modern visual content analysis are human centric. In security (e.g. video surveillance, traffic monitoring, etc.) current algorithms are designed to help a human operator but can not replace him. In the medical domain (doctor), in automotive safety (driver) or in special effects (graphist designer), current visual content analysis offers only an assistance to humans.

In a meeting organized at ACM Multimedia 2003 also known as the ACM SIGMM retreat, it was clearly stated [Rowe 2005] that "research that incorporates the user is more difficult because human behavior is so variable. Nevertheless, the goal of nearly all multimedia application is to solve a problem for a user". Almost ten years after this meeting, the same opinion leader, L. A. Rowe [Rowe 2013], reinforces this idea and reviews observed progress based on the 20th anniversary panel at ACM Multimedia 2012 "Coulda, Woulda, Shoulda: 20 years of Multimedia Opportunities" : "although it was not discussed, retreat participants understood that humans would become assistants to computer algorithms (e.g. Amazon Mechanical Turk) and computers would grow more important as assistants to humans (e.g. Apple Siri)".

In this context, it is only natural to see that approaches where users participate and

[1] `http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9`

help computers are now flourishing. The question now is not whether humans can help the machine, but **how** they can do it. In this spirit, conferences such as HComp (since 2009) and CrowdMM (since 2012) bring together researchers that study the problems that are inherent to human involvement: how to motivate humans to contribute? How to bring together inputs from different humans? How to cope with errors from humans?

An even more recent trend study how the complementarity between humans and computers should push scientists to devise methods where they work together. The firstly held ECCV'14 workshop HMCV (Human-Machine Communication for Visual Recognition and Search) is one of the first initiatives in this direction. A particular work in this domain on the problem of fine-grained object categorization [Branson 2014] is particularly enlightening. In this paper called "The ignorant led by the blind", the authors devise interactions between a human who does not know the main features of all different species of birds (the ignorant) and a computer whose artificial visual system is too imprecise to recognize the features (the blind). By designing proper interactions and underlying model, the authors build a convincing categorization system that use both human and computer capabilities.

1.3 Thesis contributions and outline

In this dissertation we present and contribute to a part of these works that tries to make the human and the machine work together. We believe that humans are necessary since they bring the semantics that computers are not capable of having yet, and that automatic approaches can facilitate, reduce and correct human inputs. Together, content analysis and usage analysis can achieve high performance, and maybe reach the original goal of our field of study: understanding visual content.

This thesis introduces three main contributions.

First, we present a detailed review of the state of the art on the combination of content analysis and usage analysis. We choose to categorize the different types of combination in two types. We believe there are content analysis algorithms that humans can help performing better. In this context we define different paradigms for human intervention (Crowdsourcing, Human Computation) and present the challenges raised by these techniques. On the other hand, there are content analysis algorithms that allow human interventions to be more efficient. We focus on this section on interactive techniques for segmentation and recognition, as well as on interfaces that build on content analysis to create improved interactions for humans.

Our second contribution is a set of new saliency detection techniques that are based on an analysis of humans' interactions with various interfaces as well as content analysis techniques. Saliency detection aims at detecting a visual content's regions that attract the human's eyes, and we argue that the use of some interfaces can implicitly reveal these regions. We build saliency maps for videos based on an analysis of browsing traces of a Zoomable Video player. We then build 3D saliency maps (or interest maps) by looking

at how humans film a scene. We propose applications for these objects, including video retargeting, video mashup and 3D query of a video.

Finally we study the problem of interactive segmentation, first by proposing a new interface for crowdsourced interactive segmentation called *Click'n'Cut*. We show that our interface can be very efficient for crowdsourcing segmentation. We then go a step further and study how to perform an implicit crowdsourcing for segmentation. We introduce a Game With A Purpose (GWAP) called Ask'nSeek, that allows to gather textual and spatial information about images. We present an algorithm for both object detection and figure-ground segmentation based on these traces, and show promising results.

Chapter 2

Related Work

2.1 When humans help Content Analysis

In the context of this dissertation, we use the term *Content analysis* in the way it is used and understood in the multimedia community. To illustrate, the call for papers of the Multimedia Analysis track at the ACM Multimedia 2013 conference refers to *content analysis* as "information extraction and processing from multimedia data". In this document we will be limiting ourselves to visual content only, such as image, video and 3D content. Restricting ourselves to visual content is not really a limitation with respect to the challenges brought by multimedia data: we'll manipulate video streams along with clickstreams (produced by the users) or many simultaneously recorded video streams which are, in essence, multimedia data.

In this context, content analysis consists in asking the following questions: Is it possible to automatically categorize the objects present in the content? Can we determine the precise position of each of these objects? Can we understand what the content is about ?

Figure 2.1: Challenges in visual content analysis: original image and saliency as computed by [Itti 1998] (top), object detection and object segmentation (bottom).

These questions have led researchers to define a number of problems that have been at the heart of communities such as computer vision, image processing or multimedia. Here is a non exhaustive list of these problems for images (illustrated in figure 2.1), as they were introduced in the PASCAL VOC [Everingham 2010].

- *Classification*: given a visual content (typically image or video), the problem of classification consists in determining the classes of objects that are present on the content.

- *Object detection*: given a visual content, object detection consists in locating instances of semantic objects on the content, usually under the form of a bounding box.

- *Image segmentation*: this task consists in associating to every pixel of an image (or of each frame of a video) a semantic label. This is one of the ultimate goals of computer vision.

- *Action Classification*: this task consists in finding the action(s) performed by a character in an image (e.g. jumping, riding a bike, phoning etc.).

Associated problems for video understanding can be naturally derived from the previous list. They lead to even more challenging issues such as video (object) segmentation, event detection and so forth. In the same spirit, automatic segmentation of 3D content is also a generalization of the image segmentation problem. In all cases (image, video, 3D), segmentation has been a hard challenge for years: besides the intrinsic difficulty of an inverse formulation (converting visual digital data into information about a physical object), the ultimate objective is to recover *semantic* information (e.g. object or scene decomposition in semantically meaningful parts).

Historically, these challenges have been adressed progressively. The first digital image was created in 1957 (it was a scanned picture of a baby) and the first digital camera was invented in 1975 (by Steven Sasson, at Eastman Kodak). The analysis of digital images by computers soon became a prominent research subject, first in the signal processing and pattern recognition communities, then in the computer vision community and finally in multimedia, in which it is part of more general multi-modal analysis.

As already written in the introduction, researchers were looking for fully automatic approaches to perform content analysis or, in other words, for solutions to replace humans in visual analysis tasks. The idea was really to "make computers see", and the applications that were targeted were mainly focused on robotics. The hope was to give robots a detailed enough perception of their environment in order to be able to move and perform tasks automatically.

With the progressive increasing of the amount of data (digital images were still rare twenty years ago, before the development of mass produced digital cameras), researchers understood that the problem of content analysis was very difficult. Many approaches had been developed and worked well on some images, but were ineffective on others. Researchers started to get confronted with the fundamental problem of semantics: how to infer complex scene's descriptions from a multidimensional (generally ranging from 2D to 4D) signal?

Since automatic algorithms were failing to solve these complicated research problems because of the lack of semantics, researchers have started to investigate a simple way to get semantics: bringing humans into the loop. All the content analysis tasks cited earlier,

which are rather difficult to perform for computers, are mostly straightforward to humans. This simple fact was the basis of a new trend in which computers take human knowledge into account to automatically perform content understanding tasks.

Besides interactive vision or many sophisticated human-in-the-loop approaches that we review in the second part of this chapter, we identified an early exploitation of human expertise through the supervised learning paradigm. In the following section, we briefly explain how human experts can annotate learning datasets and how machine learning techniques have attempted, by inference, to bridge the *semantic gap* between automatically computed low-level information and high-level expertise from the annotators.

2.1.1 Supervised Machine Learning

The mid-nineties saw an explosion in the number of methods based on Supervised Machine Learning algorithms. Supervised Machine Learning is a form of artificial intelligence, where algorithms are designed to learn the solution of a problem from annotated data in order to later be able to solve the problem on a different set of data. Machine Learning had a dedicated session in CVPR 1997 (in computer vision), and later started to become a prominent keyword in ACM Multimedia (in the multimedia community) in 2001. These methods already existed before, but were often using synthetic data as a training set. The originality of the methods that became popular at the end of the 1990s was that the training set was annotated by humans. This is a first answer to the lack of semantics encountered in previous efforts in content analysis. In this context, semantic information is brought by human annotations, learned by algorithms and finally expanded to new data.

Prediction Models. Machine Learning is a broad topic which encompasses very different approaches which generally fit well with the general principle exposed in the following schema:

$$\mathbf{f} \in \mathcal{X} \rightarrow \boxed{\text{prediction model h}} \rightarrow \hat{y} = h(\mathbf{f}) \in \mathcal{Y}$$

In supervised machine learning, a prediction model (h) is the unknown entity. This predictor may sometimes take a mathematical form (e.g. h may be a function with unknown parameters) but may also be seen as procedural (e.g. a decision tree with unknown structure). In any case, the prediction model aims at automatically predicting a high-level (e.g. semantic) information $\hat{y} = h(\mathbf{f})$ from a low-level feature vector $\mathbf{f}$ (e.g. visual descriptors). Before being used on a test set of new visual data, the quality of this inference process is first guaranteed on a learning set. A learning set is a set of supervised labeled instances $\{(\mathbf{f}_i, y_i)\}_{i=1...n}$ for which a human expert has manually bridged the gap between each low-level stimulus $\mathbf{f}_i$ (e.g. an image or a compact representation of it) and the associated high-level information (e.g. a semantic label). Such learning sets (or ground truth annotated datasets) are extremely useful to the research community to learn predictors and evaluate them. As a consequence, releasing annotated datasets or annotating tools is now

considered a valuable contribution to the community.

Ground truth datasets. To illustrate this trend, consider for example the problem of object recognition. There have been several efforts to create annotated datasets. In 2004, Li et al. introduced the Caltech 101 dataset [Fei-Fei 2004], which includes images of 101 different classes. This dataset was later expanded to reach 256 classes.

Starting in 2005, the PASCAL Visual Object Challenge [Everingham 2010] provided not only annotated ground truth, but also an online platform for benchmarking the algorithms. The challenge ran from 2005 to 2012, starting with 4 classes and 1,578 images and ending up with 20 classes and 11,530 images.

More recently, since the end of the PASCAL Challenge, a similar challenge called ILSVRC (Imagenet Large Scale Visual Recognition Challenge) has appeared, bringing the annotated datasets to the next level. In 2014, the challenge includes 456,567 images and 200 classes, which is still far from the estimated number of visual categories observed in real life (20K-30K according to [Biederman 1987]).

Performances. All these datasets have favored advances in the object recognition field, by providing common test sets to ease inter-methods comparison, as well as furnishing huge amounts of annotated data. But these benchmarks also give an insight about the main limitation of methods based on the supervised machine learning paradigm: algorithms can only be as good as the training set allows them to be, and the nature of the testing set can lead to deceiving results. Figure 2.2 shows the evolution of the best performances by class in the PASCAL object detection over the four last years of the challenge. The overall performance went from 0.3 to 0.4 in four years, which is a significant improvement but still very far from what could be considered good results.

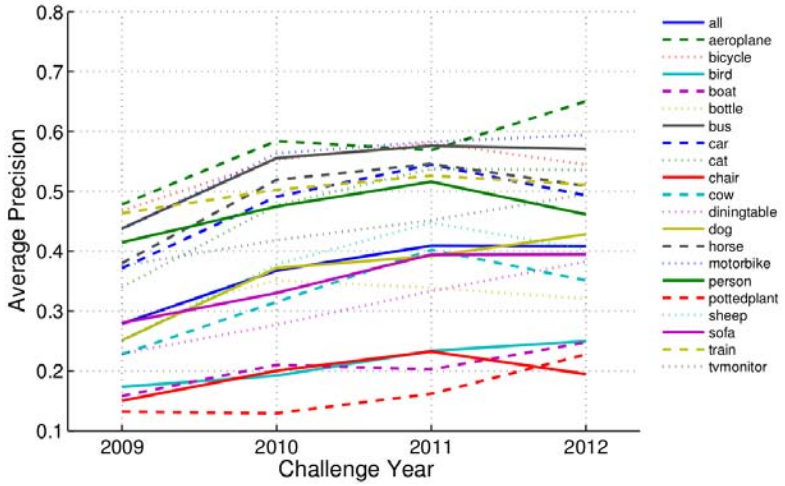


Figure 2.2: Evolution of the performances in the PASCAL VOC object detection challenge. The values plotted correspond to the best performer for each class at each year.

Consider for example the Regionlets algorithm by Wang et al [Wang 2013b]. In this

method, the authors propose a new object representation that is based on a hierarchy of rectangular regions and sub-regions (referred to as regionlets). The model is then trained using a boosting algorithm and is used as a detector on the test data. This algorithm exhibits results that place it among the top performers in the state of the art. However the mean of Average Precision that is achieved on the PASCAL dataset only reaches 41.7% and drops to 14.7 % on the much larger Imagenet dataset. This illustrates several facts: 1) a supervised learning algorithm's performance is very dependent on the dataset; 2) a larger training set will not necessarily improve the algorithm's performance and 3) results on 200 classes are still very poor, and we are far from achieving a similar result on all existing classes of objects.

Supervised learning techniques are still close to the historical goal of computer vision researchers, which was to design fully automatic algorithms that would perform content analysis. Indeed, humans intervene only at the beginning of the process to train the algorithm. Once trained, the algorithm is capable of functioning on its own, without humans.

Annotation tools.

As a consequence, the interactions between humans (the annotators) and the computers or algorithms are limited to the annotation phase in the supervised machine learning paradigm. Many visual annotation tools have been created to support the creation of ground truth datasets (for which a workshop was created [Spampinato 2012]).. The survey by [Dasiopoulou 2011] discusses this matter in depth and compares some important tools for image annotation (Caliph [Lux 2003, Lux 2009], PhotoStuff [Halaschek-Wiener 2005], Aktivemedia [Chakravarthy 2006], Kat [Saathoff 2008] and LabelMe [Russell 2008]) and some representative tools for video annotation (ViPER-GT [Doermann 2000], Anvil [Kipp 2010] and LabelMe video [Yuen 2009].

Without paraphrasing the survey from [Dasiopoulou 2011], our own analysis will put the emphasis on the coherence between these tools and the supervised learning prediction schema:

$$\mathbf{f} \in \mathcal{X} \rightarrow \boxed{\text{prediction model h}} \rightarrow \hat{y} = h(\mathbf{f}) \in \mathcal{Y}$$

Many reviewed image annotation tools (like PhotoStuff, Aktivemedia or Kat, among others) use semantic web technologies and ontologies to properly manage the semantic space $\mathcal{Y}$. LabelMe [Russell 2008] simply uses free text and keywords without any strong constraint on $\mathcal{Y}$. Caliph [Lux 2003] also uses free text while ensuring a MPEG-7 compatibility. For the video annotation tools, free text is also allowed in LabelMe video [Yuen 2009] and XML formats are used both in VIPER-GT and Anvil.

When considering the input space $\mathcal{X}$ of the predictor, some tools (like Kat [Saathoff 2008]) host plugin modules to compute low-level visual descriptors. These computations must be done over precise image regions when the annotation is made at object level: figure 2.3 shows the LabelMe interface where polygons are manually drawn to annotate the image at object level. Most of the tools integrate a polygon coding technique to handle these annotations and also support propagation of polygons in the case of video annotation. However,

these propagation techniques are not content-aware in the sense adopted by more recent tools to be reviewed later in this chapter.

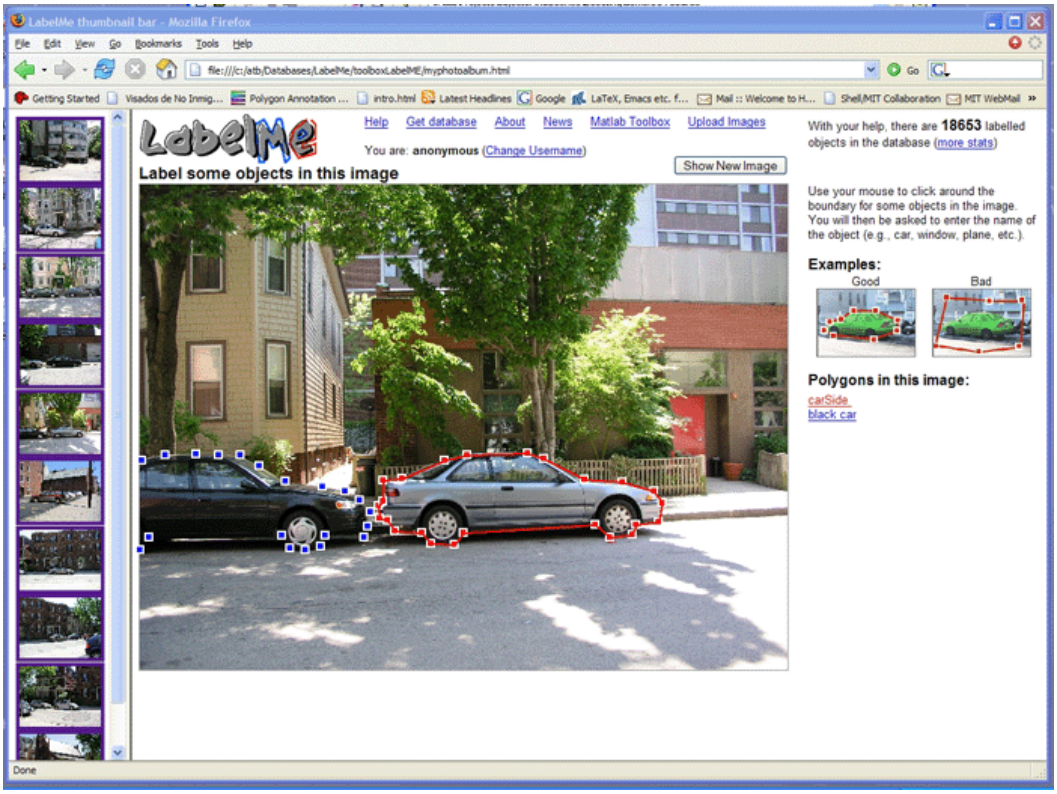


Figure 2.3: LabelMe interface.

A supervised learning approach may succeed in automating a vision task if the visual representation space $\mathcal{X}$ is well chosen. It should be representative, discriminative and compact. The visual descriptor should also be adapted to support a priori knowledge such as a prior distribution over $\mathcal{X}$ (in the deep-learning sense) or a specialized description for specific objects/scenes/tasks. For instance, specific descriptors can be devised when the interesting visual objects are humans (with applications in analysis of people or crowd, human action recognition, human interactions).

When a supervised learning approach fails, mainly two elements are questionable. First of all, the visual representation space $\mathcal{X}$ can be insufficient to discriminate several high-level concepts. For example, in their inspiring work [Vondrick 2013], the authors explain what goes wrong when a feature like SIFT fails to see what it should. Second, the dataset itself could explain the failure. The size of the training set could be a limitation if the ground truth doesn't sample enough strategic regions like: (i) low density areas in $\mathcal{X}$, (ii) areas in $\mathcal{X}$ where the decision boundaries between categories from $\mathcal{Y}$ lies. Last but not least, the annotations could be noisy and the bad quality of labeled data could be the source of failure. Key questions are then raised: how many annotators should be enrolled? How

many experts? How to motivate them? Should several annotations be collected for the same image? The next section presents an analysis of the answers to such questions by an emerging research community.

2.1.2 Human Computation / Crowdsourcing

In parallel to the development of machine learning, there was another trend that started in the 2000s, which was called human computation. The idea is to break a complicated problem that computers can not solve into incremental tasks that are very easy for humans to do, and then find a way to actually make some humans do it. In this paradigm, humans participate in every step of the algorithm which is completely different in nature from supervised learning techniques. This idea was promoted in Luis Von Ahn's work about online security, called ReCAPTCHA [Von Ahn 2008b].

Figure 2.4: The ReCAPTCHA user interface. The left word is computer generated and the right word is extracted from a book to be digitalized.

The concept of CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) was already well known in the security community. Since the point of CAPTCHAs is to come up with challenges that humans can solve but that can not be solved by computers (to prevent, for example, massive email account creations for spamming), Von Ahn took a problem that remained challenging (OCR) and had the idea to use the words that computers failed to recognize automatically as a CAPTCHA (see Figure 2.4). By doing so, Von Ahn achieved two goals: 1) securing web platforms and 2) getting humans instead of computers to perform OCR.

In his PhD dissertation called *Human Computation* [Von Ahn 2005], Von Ahn defined human computation as *a paradigm for utilizing processing power to solve problems that computers cannot yet solve.* This definition is very close to another concept which emerged around the same period, and that is called Crowdsourcing. Introduced in the Wired magazine by Jeff Howe [Howe 2006], *Crowdsourcing is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call.*

These two definitions only differ on the matter of the nature of the task that is considered. In human computation the task is originally devoted to a computer, whereas in crowdsourcing the task is originally performed by a single human. In both cases, it involves

outsourcing the task to a group of people, gathering the answers and deducing a final solution out of it. These two concepts have become so important that they now correspond to actual communities: the HComp conference (about human computation) has been running since 2009, and the CrowdMM workshop (about Crowdsourcing) is organized since 2012.

The survey from Quinn and Bederson [Quinn 2011] tries to categorize algorithms that fall into these definitions, and use six components of these methods to categorize them. Since in this dissertation we are more interested in the challenges posed by these methods, we will detail only two fundamental aspects:

- Incentive: how to motivate users to perform the tasks?
- Quality Control: how to check the rightfulness of users input?

Table 2.1: Users incentives in Crowdsourcing and Human Computation

	Incentive	Example
Intrinsic Motivation	Enjoyment	GWAP
	Social development	GWAP
	Personal Development	Duolingo
Extrinsic Motivation	Payment	Mechanical Turk
	Action Significance	*Tomnod*
Implicit	via Interface	Zoomable Video
	Security	ReCAPTCHA

Incentives. The first challenge that researchers have to face is to motivate users. Kaufmann and Schulze [Kaufmann 2011] have conducted a study on Amazon Mechanical Turk to understand the reasons that make workers participate in tasks. Not surprisingly, authors found out that payment was the main incentive for workers on Amazon Mechanical Turk. They also established that intrinsic motivation is more important to workers than extrinsic motivation. In addition, they proposed a classification of incentives that is similar to the Open Source community incentives categorization. In this dissertation, since we are interested in tasks that target visual content analysis, we only discuss relevant categories in this context. Table 2.1 detail the three main types of incentives: intrinsic motivation, extrinsic motivation and implicit motivation.

Intrinsic motivation regroups the reasons that invest the workers personnally on the tasks. Enjoyment is an obvious intrinsic incentive: when users have fun while performing a task, they are less likely to cheat and more likely to do more tasks. Games With A Purpose (GWAP) are a class of platforms that try to trade users enjoyment for useful input data. We will give more details about GWAPs later in this chapter, since this is an avenue we have chosen to explore in our work. Social development can also be an intrinsic motivation for workers. The possibility to meet new people, join a community and maybe make new friends is a strong incentive. Multi-player GWAPs are once again a good

example of this phenomenom. Personal development covers tasks from which workers will gain competences. A recent example is Duolingo, a website/app created by Luis Von Ahn. Duolingo presents itself as a website to learn languages, but is designed to benefit from users while they learn and use their input to automatically translate texts.

Workers can also have extrinsic motivations to complete tasks. The most common one is money: users get paid to complete tasks. Many websites provide a platform to connect requesters (that propose a task) and workers. One of the most famous of such websites is Amazon Mechanical Turk, created in 2005. When writing this thesis, we observerd that more than 200,000 tasks (also known as HIT, Human Intelligence Task) were available at the same time. Others similar platforms include Samasource, Microworkers, Crowdflower etc. InnoCentive is a similar platform in which requesters are companies and workers have to submit creative solutions to industrial problems. Although requesters offer a reward, the nature of the tasks proposed on InnoCentive provide an additional incentive to workers: action significance. Indeed some of the tasks could potentially benefit society, and this makes workers feel useful to their community which is a strong motivation. Another recent illustration of this phenomenon can be seen on `http://www.tomnod.com/nod/challenge/mh370_indian_ocean`. This website proposes workers to look for the vanished flight MH370 from Malaysian Airlines. (see Figure 2.5)

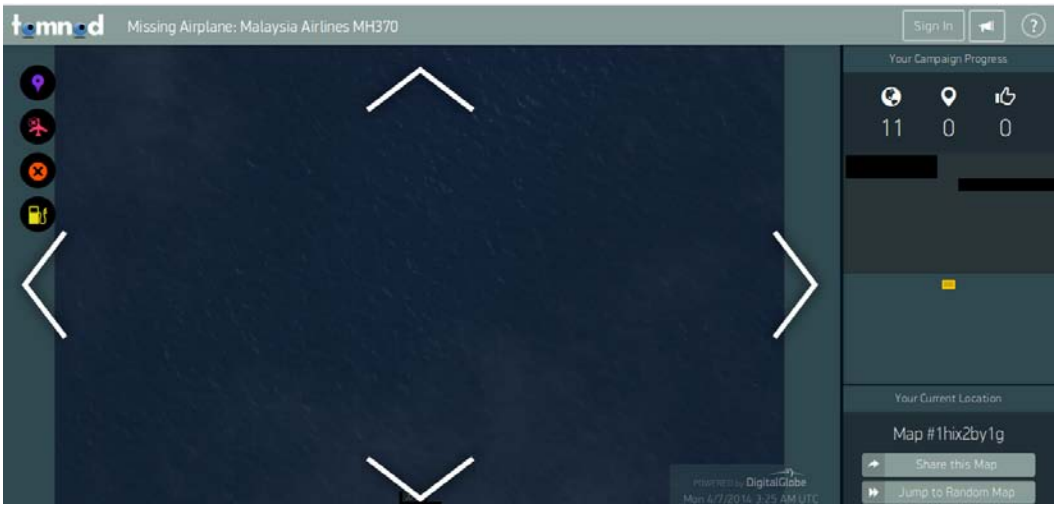


Figure 2.5: Interface of the *tomnod* website to localize the missing MH 370 plane.

Finally it is also possible that workers participate in tasks without even knowing it. This can occur for example when workers use an interface to consume content, like the zoomable wideo that we will describe in section 3.1 of this thesis. Another case of implicit motivation is the use of security measures, for which a previously introduced example is the ReCAPTCHA application, from Luis Von Ahn.

Since incentive is such a central challenge when creating a crowdsourcing/human computation task, one can wonder which incentive leads to the best results in terms of quality

and informativeness of traces. The most obvious fact is that incentives are task-dependent. It is not always possible to design an interface for which users will implicitly solve problems, as well as creating a GWAP that will help solving a particular problem. Payment is often the simpler solution, since it allows collecting a large amount of data rather quickly. Conversely, since workers are paid to do a task, they will be more likely to try to cheat the system to earn more easily. Mao et al. [Mao 2013] published a comparison of the work done by volunteers against turkers. They selected a task proposed on *Zooniverse*, a crowdsourcing website where citizens volunteer do tedious tasks to help science projects. They proposed the same task on Amazon Mechanical Turk, and observed a similar quality of traces. They also tried different methods to pay the workers: per task, for time spent, and per annotation. They observed that workers were less precise when paid per task, and in average spent less time to perform the tasks. Conversely, when workers were paid per time (for a fixed time spent on doing the task), both their precision and their recall rates increased, but the time they spent per task was also higher. Araujo [Araujo 2013] studied traces from the website *99 designs* where requesters post artistic challenges (e.g. creating logos) and workers submit their proposal. At the end of the challenge, the requester chooses the best answer and rewards the worker. It can also happen that the requester is not happy with the results, in which case nobody receives the reward. Araujo found that the number of workers was a key parameter to the quality of the winning design. For a given fixed price, the more workers participate in the challenge, the better the winner is. A somehow surprising result is that for a fixed number of participants, increasing the reward does not lead to a better quality for the winning design.

Quality Control. Controlling the quality of workers inputs is really the most important challenge in crowdsourcing and human computation. It is indeed necessary to discard as much noisy data as possible, in order for the algorithms based on this data to perform as well as possible. Typically noisy data can come from three different sources [Oleson 2011]: *scammers*, *insufficient attention* and *incompetent workers*. *Scammers* voluntarily try to cheat the system (e.g. to maximize their profit while minimizing their effort). *Incompetent workers* do not understand the task description and therefore produce a lot of noisy data. Finally workers with *insufficient attention* understand the task but get tired or confused and do some random mistakes. There are many policies that can help reduce the number of noisy inputs. These policies can be applied either before, during, or after the task.

Before the task, it is very important to make sure that the workers fully understand what they have to do. This helps limiting the amount of noise due to *incompetent workers*. First, it helps to have a tutorial explaining all commands, and showing examples of both good and bad inputs. Gottlieb et al [Gottlieb 2012] explain how, in the difficult context of video geo-location, several rounds of testing convinced them to setup a tutorial that showed workers how to perform the task on an example. In addition to having a tutorial, it is also important to have gold standard data, i.e. tasks for which the answer is already known. Such data can be used to display messages to workers when they do a wrong annotation on

these tasks. Oleson et al [Oleson 2011] introduce the concept of programmatic gold, which consists in generating gold standard from crowd-generated data.

There are also several mechanisms that can limit noisy data to be produced by workers. First, mechanisms that rely on an agreement from several humans working independently and simultaneously on the same task considerably limit this issue. Both input and output agreements have been formally defined by Luis Von Ahn [Von Ahn 2008a] in the context of Games With A Purpose. Output agreement consists in accepting an answer if two workers agree (independently and simultaneously) on this answer. This process is used for example in the ESP game [Von Ahn 2004]. In input agreement, two workers are presented (independently and simultaneously) with one input each, and their goal is to find out whether this input is the same or not (figure 2.7). An example of this process can be found in the game Tag-A-Tune [Law 2007]. This agreement concept is interesting because it takes a conjunction of incidents for noisy data to appear. Either two spammers have to find a way to cheat together (which may be difficult), or two workers have to make the same mistake at the same time which is highly unlikely. Similarly, one can also devise tasks that involve some workers validating or correcting another worker's tasks. In the Natural Language Processing community, Bernstein et al. [Bernstein 2010] even formalized a "crowd programming pattern": *Find-Fix-Verify*. This pattern consists in having one worker look for candidate errors (the context is the research and correction of spelling mistakes), then have a second worker correct the mistakes and finally have a third worker validate it. An example of the use of this pattern in visual content analysis has been given by Su et al [Su 2012], in the context of ground truth creation object detection. For a given label, one worker has to create bounding boxes around this object in some images, then a second worker can tweak the bounding boxes position and size so that they fit the object perfectly, and finally a third worker validates the boxes. Another way to ensure data quality is to make workers annotate data for which the answer is already known, thus providing an easy way to detect *scammers* and *incompetent workers*. This method is also known as Ground Truth Seeding.

Finally, the most evident moment to detect noisy inputs is after the data collection. First a statistical analysis of the data can help discarding bad inputs. In their experiments, Mao et al [Mao 2013] determined that is was not normal that workers spent less than 5 seconds on a task or produced more than 8 annotations per task, which helped them discarding bad workers (of course these values are task-dependent). In the same spirit, when a task is assigned to more than one worker it makes sense to use a majority vote or robust estimators (such as median) to find a suitable answer while eliminating outliers. More elaborated noise correction algorithms have also been proposed. Ipeirotis et al. [Ipeirotis 2010] use an EM-based algorithm to find both the errors of the workers and a measure of the workers' reliability.

To add a little bit of perspective to this problem, it is worthy mentioning the work of Grier [Grier 2011] who argues in his paper that human computation is not a new idea,

Figure 2.6: Interface of the ESP game [Von Ahn 2004]

and that similar paradigms have been going on for centuries which just did not involve any computer. The author analyzes the example of the Mathematical Tables Project, an organization that went on from 1938 to 1948 and produced tables of mathematical functions. The team that managed this organization claimed their work was mistake free, thanks to careful planning and constant effort to prevent any error. Grier lists four lessons that should benefit the current human computation community. First, there should always be an estimate of the result of a task that is assigned to a human. Second, there have to be multiple policies put into place to detect errors. Also, a result should always be computed at least twice by different methods. And finally, all possible sources of errors should be identified before starting a campaign.

We will not go through all examples of crowdsourcing and human computation in the multimedia analysis community because of the tediousness of the task. However, we will detail two categories of algorithms that are interesting to study in order to establish some context for this thesis's contributions.

2.1.3 A step towards our contributions

.

In this section we focus our related work analysis on two main subjects, that are of interest for the rest of the dissertation. First we study how to infer Regions of Interest from an analysis of users' behavior, in order to contextualize the work we present in Part I. We also focus on Games With A Purpose, as we will present in Part II our contribution in this domain.

Crowdsourcing User interest. Finding Regions of Interest is a difficult problem that involves basic semantic understanding of a content. We will later present in chapter 3 our contributions in this domain, which are based on crowdsourcing. We hereby review the related works.

A first basic idea is to extract ROIs by directly analyzing regions gazed by a user. In their work, Ukita et. al use an eye-mark recorder system to track user gaze. Important objects, including moving objects, can be detected and tracked by analyzing the gaze of the users [Ukita 2005]. Shamma et. al proposed a similar approach: they gather information about what is being watched from a user community to understand the content of the video [Shamma 2007].

These works are clearly related to the research topics on *implicit feedback*. Inspired by the definitions from Diane Kelly [Kelly 2003], *implicit feedback* can be defined as information about users, their needs and content preferences that is obtained unobtrusively, by monitoring their behavior and interactions with systems and content.

Bringing this idea into the multimedia community, Syeda-Mahmood and Ponceleon collect implicit feedback from the users by analyzing the playback interactions in the temporal domain (play, fast-forward, pause). They use it to infer the most interesting temporal segments from the video [Syeda-Mahmood 2001]. This pioneer work inspired many subsequent papers among which we can mention the one from [Gkonela 2014]. In this paper a heuristic user modeling is used as a substitute for the initial HMM (hidden Markov model) from Syeda-Mahmood and Ponceleon.

In our group, we also contributed formal models to understand implicit feedback in multimedia use cases: Plesca et al. [Plesca 2008a] used MDP models (Markov Decision Processes) and POMDP (Partially Observable Markov Decision Processes) to interpret interaction sequences (e.g. clickstreams). Moreover, from a metadata perspective, implicit feedback signals can be converted into implicit descriptors (e.g. user interest descriptor) as shown in [Plesca 2008b].

Xie et al [Xie 2005] have also studied mobile image browsers and the (implicit) power of the associated interactions. They found that on small displays, users tend to use more zooming and scrolling actions in order to view interesting regions in detail. From this fact, they designed a specific ROI extraction model (figure 2.7) from still images and so-called user interest maps. We use a similar idea to infer ROIs from a zoomable video player's browsing traces in our own contributions.

GWAP. In a similar fashion, games can be used to implicitly collect useful data for visual content understanding. The idea of using games with the purpose of collecting useful data for computer vision has been brought first by Luis von Ahn and his ESP game [Von Ahn 2004]. In that game, two players look at the same image and try to agree on a description of the image by typing words. They score points when they manage to type the same word, and in this case the word becomes part of the tags describing the image. This game has been initially devised to address the problem of constructing ground truth

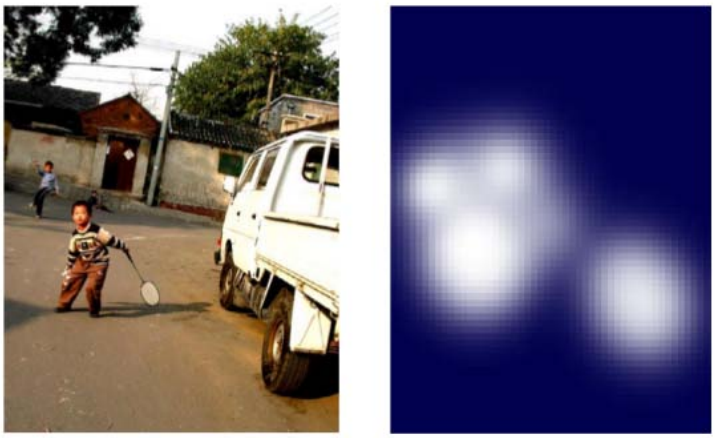


Figure 2.7: A result from [Xie 2005]: a user interest map obtained through the analysis of zooming interactions on a mobile device.

database for training computer vision algorithms. It is worth noticing that the feedback from the gamers is also implicit to some extent: they play for fun and not for labeling images. In the same spirit, Peekaboom [Von Ahn 2006], a subsequent and complementary game, goes a step further since it consists in locating objects (labeled by ESP) in a given image. Two players are again paired randomly: while one player reveals parts of the image, the other (who initially sees nothing from the image) has to guess the correct associated label. In chapter 6 we introduce the Ask'nSeek game, that can help collecting data that is equivalent to both ESP and Peekaboom traces.

In 2009, Ho et al. postulated that the cooperative nature of the ESP game has a number of limitations, including the generation of less specific or diverse labeling results, and proposed a competitive game for image annotation: KissKissBan [Ho 2009]. Their game uses a *couple*, whose objective is the same as the players in the ESP Game (i.e., to guess what the partner is typing), but introduces the role of *blocker*, a third party who has 7 seconds to provide a list of blocked words, which contains the words he thinks couples might match on. They show that the resulting have higher entropy than the ones produced by the ESP game (used as baseline for comparison), and are, therefore, more diverse.

More recently, Steggink and Snoek [Steggink 2011] presented the Name-It-Game, an interactive region-based image annotation game, whose labels are semantically enhanced by means of the WordNet ontology. Name-It is a two-player game in which players switch roles (either *revealer* or *guesser*) after each turn. The revealer is shown an image and a list of words, from which he selects an object name, chooses the definition (obtained via WordNet) that best describes the sense in which that word is used in that particular image, and outlines the object of interest using a combination of polygonal and freehand segmentation, in order to progressively reveal an object in an image to the guesser. The

guesser has to guess the name of the object (or a synonym) and may ask for hints during the guessing process.

In another recent effort, Ni et al. [Feng 2012] have designed P-HOG (Purposive Hidden-Object-Game), a single-player game in which the goal is to locate an object that has been artificially embedded (i.e., hidden) within an image by drawing a bounding box around it.

More recently a game called *Flash the fish* has been introduced in [Di Salvo 2013]. This game enables an easy and fast acquisition of video annotations. The users' clicks (or points) allow to locate fishes in the video but are not accurate and dense enough to properly segment the animals in the video clips. An interesting idea that we also followed in our own GWAP Ask'nSeek [Carlier 2012], is then to combine the clickstream from a game (e.g. *Flash the fish*) with content analysis (simple object segmentation in [Di Salvo 2013]). This is what we call combining "usage analysis" with "content analysis" in this thesis. The second half of this chapter investigates how content analysis can complement human interactions.

2.2 When Content Analysis assists humans

In the previous section, we discussed how experts may supervise automatic content analysis based on machine learning. We also showed how to enroll and manage a crowd of annotators to perform *human computation*, explicitly or implicitly. We have reviewed some tools with dedicated interfaces which ease the injection of high-level (i.e. semantic) expertise while annotating a visual content. The ergonomics of such interfaces are important in all situations but they especially matter in crowdsourcing scenarios, when non-experts may be enrolled as annotators. Recently many content-aware visual interfaces have appeared, particularly in the Human-Computer Interaction community. These works demonstrate that content analysis may enrich the interfaces, improve their ergonomics and eventually assist the interacting users by moderating their cognitive involvement. Content-aware annotation tools emerged in this context, leading to a very interesting closed loop:

- content-aware interfaces integrate content analysis to assist human annotators,
- human annotators assist content analysis through their interactions.

This is why human-in-the-loop approaches are now emerging both in computer vision and multimedia leading to new opportunities and technical challenges [Grauman 2014]. In this second half of our related work chapter, we give an overview of this flourishing topic. We first discuss influential content-aware visual interfaces (section 2.2.1) before focusing on content-aware annotation tools (section 2.2.2). The latter is strongly related to interactive segmentation models that we address in section 2.2.3. Finally, we review the most important humans-in-the-loop approaches with a particular focus on active learning techniques in section 2.2.4.

2.2.1 Content-Aware Video Interfaces

In this paragraph, we highlight several content-aware interfaces for interactive video. Analysis of video content to detect and track objects is highlighted by Goldman et al. [Goldman 2008] as enabler for new interaction paradigms to browse and manipulate video. Examples include dynamic video annotation and video navigation by direct manipulation. Direct manipulation, also discussed by Dragicevic et al. [Dragicevic 2008], allows control of video playback and access of nearby frames by directly clicking and dragging on moving objects in the image space. In this context, moving ROIs must be first detected and tracked by computer vision techniques. Figure 2.8 illustrates the idea of direct video manipulation.

Figure 2.8: Direct video manipulation [Dragicevic 2008]

Content analysis also enables hypervideo links [Shipman 2008], where links associated to moving video objects allows a quick access to additional content.

Many new interactions are made possible by ROI detection, a fundamental problem in content analysis. ROI are commonly detected using visual attention models. The works by Goferman et al. [Goferman 2012], Han et al. [Han 2006] and Itti et al. [Itti 1998] are representative in ROI detection on still images.

ROI detection can be applied to the temporal dimension; for example, it can consist in looking for interesting space-time regions within a video. Detection of interesting or important video segments (or shots) enables new video interactions such as those proposed in the *Smart Player* [Cheng 2009]. The Smart Player adjusts the video playback speed based on a predicted level of interest for each video shots. Both inter-frame motion estimations and shot boundary detections are used to support automated fast-forwarding during less interesting shots.

Vliegendhart et al. proposed the LikeLines interface [Vliegendhart 2012] which, in addition to the classical video interface, displays a heatmap of temporal regions of interest. Users can use this LikeLine to browse the video, and their interactions are used to refine the LikeLine which was initialized with content analysis.

In one of our recent work [Riegler 2014], we combined a tool to annotate videos with a zoomable video player and the LikeLine interface. The resulting interface is a powerful tool to create (for example) tutorial videos, that will then benefit from the LikeLine mechanism.

Figure 2.9: The Smart Player Interface [Cheng 2009] (left) illustrated by scenic car driving (right): a typical driver would go slowly to watch beautiful landscapes and faster when the scenery is boring.

2.2.2 Content-Aware Annotation Interfaces

The previous examples illustrate how a smart use of content analysis can improve interfaces ergonomics or create new interactions. In the image and video annotation community, interfaces are also more and more content-aware as we will explain in this section.

Table 2.2 confronts the tools -cited in the previous section- for image and video analysis and more recent interfaces that include content analysis. This non-exhaustive table is nevertheless quite representative of the current state of the art.

	Image annotation	Video annotation
Without Content Analysis	Caliph [Lux 2003, Lux 2009]	ViPER-GT [Doermann 2000]
	PhotoStuff [Halaschek-Wiener 2005]	Anvil [Kipp 2010]
	LabelMe [Russell 2008])	LabelMeVideo [Yuen 2009]
Content-Aware Interfaces	M-ontoMat-Annotizer [Petridis 2006]	SVAS [Schallauer 2008]
	Markup-SVG [Kim 2011]	PerLa [Kavasidis 2013]

Table 2.2: Annotation Interfaces with/without content analysis

As a general comment, one common feature of many annotation interfaces is a segmentation tool, which allows a quick annotation of candidates segments or regions. Considering the difficulty already mentioned earlier of the segmentation problem, many methods can be used for such a task. The tools usually either propose candidate segmentations among which the annotator can choose and annotate a segment; or it can also integrate a sursegmentation of the visual content (e.g. a superpixels partition).

The MarkupSVG tool [Kim 2011] is interesting because of its use of the SVG format as well as content-aware annotation features. It includes several Segmentation Assist Action Modules, based on active contours or on Conditional Random Fields (which work at the superpixel level, using superpixels segmentation from [Comaniciu 2002] or [Shi 2000]). Modules can be easily integrated in the MarkupSVG interface, as can be seen on the screen-

shot of this interface in figure 2.10.

Figure 2.10: Markup SVG - an online content-aware annotation tool [Kim 2011]

The M-ontoMat-Annotizer [Petridis 2006] is an older tool which integrates a Visual Descriptor Extractor (namely the aceToolbox) that can extract descriptors in regions annotated by the user and bind them to semantic descriptors at the segment level.

The Semantic Video Annotation Suite (SVAS) [Schallauer 2008] developped at the Joanneum Research Institute in Austria integrates video manipulation tools and particularly a shot and keyframe detector. The integrated Semantic Video Annotation Tool (SVAT) assists annotators to locate specific regions in a keyframe with automatic image segmentation. It also provides a matching service to automate the detection of similar video objects throughout the entire video content. The power of the tool relies on the complementarity between content analysis algorithms (e.g. that detects sometimes too many keyframes) and the users' interactions (who can manually add or delete keyframes).

The recent PerLa [Kavasidis 2013] (for video annotation) also identifies candidates segments thanks to active contours algorithms. In addition PerLa embeds mechanisms for the propagation of annotations that are considerably more sophisticated than the polygones propagation techniques reported in section 2.1.

2.2.3 Interactive Segmentation

The annotation tools we have reviewed in the last section integrate visual content analysis algorithms that assist the annotator. We have seen in particular that automatic segmentation functionalities were very useful. The development of content-aware annotation tools is therefore closely related to interactive segmentation techniques (for images as well as videos and 3D content). The most popular interaction in interactive segmentation is undoubtedly the scribble. It consists in asking users to draw continuous lines that constraint the algorithms to distinguish between an object's foreground and the background.

Figure 2.11 illustrates the use of such scribbles in the iCoseg interface for co-segmentation from [Batra 2011]. Blue scribbles are indicating foreground objects, whereas red scribbles indicate background elements.

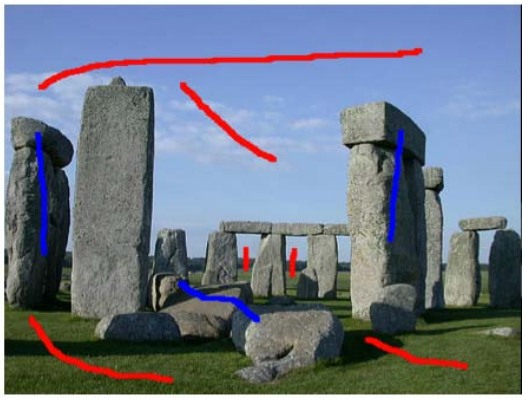


Figure 2.11: Scribbles in ICoSeg [Batra 2011]

Even for such simple interactions, there are many questions that arise for a user: How many scribbles are necessary? Are there optimal locations for scribbles? Should background or foreground scribbles be favored? Only users' experience and expertise can allow them to interact efficiently with scribble-based interactive segmentation algorithms.

The foundational proposal in this domain was based on *graph cuts* [Boykov 2001]. The algorithm considers every pixel as a node in a graph, connected to their spatial neighbors by an edge whose weight depends on the visual similarity between pixels. In addition, every pixel node is also connected to two special terminal nodes, each of them representing an *object* or *background* label. Segmenting an object is equivalent to finding the *min cut* of the graph, that is, those edges that once disconnected minimizes an energy function defined on the two resulting sub-graphs. The user is expected to manually label some pixels (through scribbles) to make the operation possible.

Other types of scribbles are sometimes used. In [Yang 2010], the authors proposed a new type of scribbles providing soft segmentation constraints. This input type allows a user to draw some "strokes" to indicate the region that the boundary should pass through.

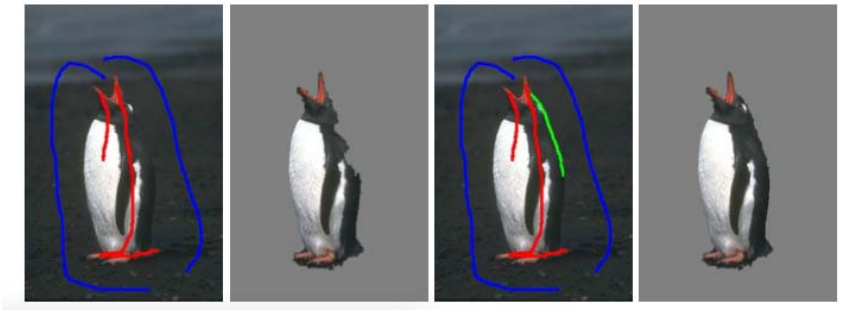


Figure 2.12: Comparison between classical scribbles (left) and injection of soft strokes in green (right) [Yang 2010]

The graph cuts approach has been later expanded by other authors. A relevant contribution was *GrabCut* [Rother 2004], an adaptation of the algorithm to color images which applies an iterative minimization that reduces the user interaction. In this case the algorithm relies on an initial labelling of background pixel by drawing a bounding box around the object. The resulting segmentation can be edited using additional scribbles. A more recent work [Gulshan 2010] has introduced geometrical constraints in the resulting shapes so that the final segments adjust better to some priors.

Other solutions [Wang 2005] [Noma 2012] [McGuinness 2013] [Lee 2014] avoid the computational load of a pixel-to-pixel segmentation by working with unsupervised image segmentations performed offline. These solutions strongly rely on the region boundaries defined by the segmentation algorithm, as they are assumed to capture all the semantics in the image. These initial image partitions are usually configured to generate over-segmentations from the semantic point of view, so that all object contours are included, but also many additional ones which are to be removed through the interaction process. The process of mapping user interaction to regions in the partitions is trivial, so that the pixel labels are assigned to their corresponding region. For example, in [Wang 2005] a video is preprocessed with a mean shift segmentation to later expand user scribbles with the min-cut algorithm.

Hierarchical image partitions have been repeatedly used in this domain because they provide an expansion criterion for the seeds [Salembier 2000] [Adamek 2006] [Arbelaez 2008] [Giro-i Nieto 2013]. These image representations correspond to a tree whose leaves are the regions of a fine partition of the image and the root correspond to the whole image. Every node in the tree correspond to a new region resulting from the merge of two regions from a lower level. When one or multiple labels are assigned to a leaf node, the merging history contained in the tree provides a decision criterion about their expansion to their neighboring regions.

The comparative study in [McGuinness 2010] indicated similar accuracy labels for GrabCut [Rother 2004] and hierarchical solutions [Salembier 2000] [Adamek 2006], but a faster response for the latter ones.

All these works that focused on image interactive segmentation have later been generalized to video segmentation [Price 2009, Chen 2012] and to 3D content segmentation [Price 2011, Malmberg 2012].

On the related topic of object co-segmentation, which consists in segmenting the same object in multiple images that feature this object, it is worth describing iCoseg [Batra 2010]. In this paper, the authors also allow users to draw scribbles on images to annotate background and foreground. The scribbles on one image are used to co-segment all images that show the same object. In addition, the authors use an active learning formulation that allows the system to automatically detect the areas that would lead to the most informative scribbles, and propose it to the users. This active recommendation system is illustrated in figure 2.13 where a candidate region for a next scribble is suggested to the user. This region is identified as the most discriminant one by the computer vision algorithm.

Figure 2.13: Next scribble region suggestion in ICoSeg [Batra 2011]

This algorithm belongs to a class of approaches known as active learning: the human annotation is guided by the algorithm's suggestions, making the algorithm active in its learning process. Other recent works have performed interactive segmentation while following active learning guidelines [Wang 2013a].

In the next section we give a more detailed overview of the utilization of active learning techniques in works that emphasize on the collaboration between a content analysis system and users.

2.2.4 Active Learning and Humans-in-the-loop approaches

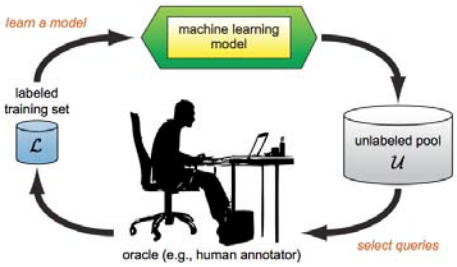


Figure 2.14: The pool-based active learning loop from [Settles 2010]

As suggested by figure 2.14, the ultimate level of interaction between automatic content analysis and human annotations takes the form of a closed loop. The fundamental idea of active learning techniques is to try to obtain the best possible inference performances (on a test set, i.e. new, unlabelled data) for a supervised learning system that can be modelled as:

$$\mathbf{f} \in \mathcal{X} \rightarrow \boxed{\text{prediction model h}} \rightarrow \hat{y} = h(\mathbf{f}) \in \mathcal{Y}$$

but with a reduced learning set $\{(\mathbf{f}_i, y_i)\}_{i=1...n}$ (i.e. with n as small as possible). To achieve this goal, the learning data is actively and sequentially chosen by the algorithm which then requests to a human expert (or oracle, see figure 2.14) the annotation of this training data. It has been shown in section 2.1.2 that gathering annotated data (by crowdsourcing from example) can be a tedious and costly process. It is not efficient (and sometimes counterproductive) to use redundant annotation data when an oracle can provide more insightful data at the same cost. It seems natural that the algorithm which is actually building the classifier should determine the annotations to request from an expert. The algorithms usually determine the next query by optimizing one of many possible criteria. These criteria, which have for the most part a solid mathematical background, are explained in more details in Burr Settles survey [Settles 2010].

The general intuitive idea is that a good annotation should reduce the potential ambiguities of the prediction model. There are several strategies that can guide the choice of the next query. The *uncertainty sampling* technique consists in querying annotations $y_j \in \mathcal{Y}$ of data $\mathbf{f}_j$ located around frontiers between different categories, as predicted by h. In *pool-based sampling*, algorithms define and evaluate an informativeness (or utility) measure to choose from a pool of unlabeled data which one would bring the most useful (or informative) annotation.

There are many more strategies identified by [Settles 2010]:

- *Query by commitee*: this approach involves maintaining a committee of prediction models $\{h_1, h_2, \dots h_C\}$ all trained on the current labeled data $\{(\mathbf{f}_i, y_i)\}_{i=1...n}$ but representing complementary hypotheses. Each prediction model is allowed to vote for incorporating a new unlabeled data picked out from a pool of K query candidates $\mathbf{f}_{n+1} \dots \mathbf{f}_{n+1K}$. The most informative query is considered to be the candidate about which they most disagree.

- *Expected model change*: in this strategy, the selected instance is the one that would imply the greatest change of the model if we knew its label (e.g the greatest impact on the current parameters of h if its is parametric).

- *Expected error reduction*: as clearly suggested by its name, the selected instance is the one that best reduces the expected prediction error (EPE) of h on new data (e.g. minimizing $E_{\mathbf{f},y}[l(h(\mathbf{f}), y)]$ for a given loss l).

- *Variance reduction*: in this case, the instance minimizes the variance term hidden (along with the bias) in the expected prediction error EPE.

- *Density-weighted methods*: in this strategy, minimizing the generalization error (i.e. EPE) or reducing its variance related portion is only done for representative instances according to a modeled distribution $p(\mathcal{X})$ of $\mathbf{f}$ features in input space $\mathcal{X}$. Outlying instances lying in low density areas are then avoided.

All those approaches have an interesting potential to limit the human involvement during annotation and to only trigger queries to humans where it is most crucial. Besides these multiple strategies, Settles also discusses practical questions about how to use non-experts (or even noisy experts) in active learning. Key questions are: when should the algorithm decide to query for the potentially noisy label of *new* unlabeled instances? When should it query for repeated labels to dnoise an *existing* uncertain training instance? How does annotation quality vary over time? etc.

Several recent studies implement active learning techniques to build annotated visual datasets. Kristen Grauman, in her very recent publication [Vijayanarasimhan 2014], extends a CVPR'11 publication where she presented, with Vijayanarasimhan, an active learning approach for live learning of object detectors. The basic idea is to use active learning to train window-based detectors based on SVMs. This is actually challenging since the object extents (i.e. the windows sizes) are unknown in the unlabeled instances. In other words the active learning is done at the level of an object and not at the level of entire image. An immediate interpretation of the previously introduced active learning principles would lead to evaluate all possible windows within an image in order to choose the most uncertain (following the *uncertainty sampling* idea). This is impossible since it would lead to exhaustively evaluating a prohibitively large unlabeled pool of instances. Instead, a *jumping window* method speeds up the candidate windows generation. This generation is further improved using a hashing scheme and the active selection is eventually performed with a SVM margin criterion (which encodes the uncertainty). We clearly see that content analysis (i.e. the current margin for any instance) assists the subsequent human interventions in a closed loop (figure 2.14).

The same authors also investigated active learning for video annotations [Vijayanarasimhan 2012]. Manually segmenting and labeling video objects is tedious and we already underlined the need for propagation techniques to alleviate many difficulties. In [Vijayanarasimhan 2012], active learning is used to actively select the keyframe on which pixel level annotations will be manually done before propagation. The raised research questions are the following: how many keyframes should be optimally labelled? Which keyframes should be annotated? Following the previous active learning methods, the authors typically look for the most useful frames such that some expected labeling errors are minimized. Their method keeps the number of selected frames low and ensures that after propagation minimal human intervention is required.

In [Kovashka 2011], the same group from Austin introduces new possibilities for actively teaching object recognition systems. The key idea is that different queries can help: why should we systematically ask questions like "what is the category of this image ?", "what is this object ?" In [Vijayanarasimhan 2014], we already saw that querying at object level (and not at image level) can be an interesting functionality. More generally, the queries in active learning shouldn't be seen as monolithic or indivisible. Kovashka et al. [Kovashka 2011] propose to actively select image annotation requests among object category labels *and*

attribute labels. By object labels, they mean usual object names like *house, phone, dog, etc.* whereas attribute labels might include emphwooden, furry, red, etc.. Using both types of query they try to reduce total uncertainty for multi-class object predictions. During one active learning loop a human may be asked to name an object and then may be asked to state whether a particular attribute is present in the next cycle. It is worth noticing that accounting for dependencies between labels on the same image is not straightforward yet useful in active learning.

In the same spirit, selecting the best *type* of active query in a humans-in-the-loop approach for object recognition has been investigated in depth within visipedia project. Visipedia (see `http://www.vision.caltech.edu/visipedia/`) is a joint project between Pietro Perona's Vision Group at Caltech and Serge Belongie's Vision Group at UCSD. Visipedia, short for "Visual Encyclopedia", is an augmented version of Wikipedia, where pictures are first-class citizens alongside text. The goals of Visipedia include creation of hyperlinked, interactive images embedded in Wikipedia articles, scalable representations of visual knowledge, largescale machine vision datasets, and visual search capabilities. Toward achieving these goals, Visipedia advocates interaction and collaboration between machine vision and human users and experts.

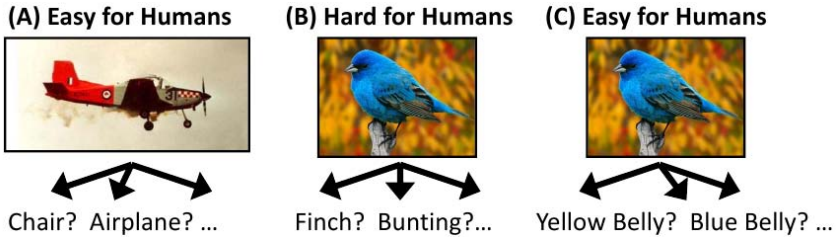


Figure 2.15: Fine-grained object recognition [Branson 2010]

In one of the visipedia sub-projects, object recognition is investigated under its so-called fine-grained variant. Fine-grained object recognition consists in differentiating different sub-categories in a same class. In the works we describe here, the authors study the case of birds species. In this example, object recognition would consist in recognizing that the object is a bird, and fine- grained recognition would consist in recogning to what specie the bird belongs to (figure 2.15). Fine-grained recognition is a very hard problem, since the objects the algorithms must differentiate look alike: they have the same shape, the same attributes, sometimes even the same color and texture.

Distinguishing between *finch* and *bunting* is difficult for most people (see part B of figure 2.15), but is doable by people with the appropriate expertise. The key idea is that answering alternative questions of part C (*yellow belly ? vs. blue belly ?* is also feasible for non-experts and can definitely help a computer recognition system. In [Branson 2010], the proposed classification method can be seen as a visual version of the 20 questions game, where questions based on simple visual attributes are posed interactively (`http://www.20q.net`).

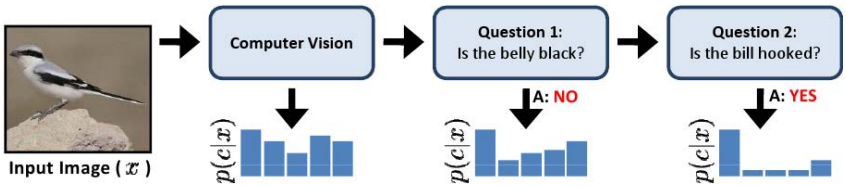


Figure 2.16: [Branson 2010] Visual recognition with humans in the loop

For Branson et al. the goal is to identify the true class while minimizing the number of questions asked, using the visual content of the image. Incorporating user input drives up recognition accuracy to levels that are good enough for practical applications; at the same time, computer vision reduces the amount of human interaction required. Following active learning rules, the resulting humans-in-the-loop system is able to handle difficult, large multi-class problems with tightly-related categories. In figure 2.16, the conditional distribution over fine-grained classes is illustrated. The optimized sequence of questions obviously allows to reduce the distribution entropy and eventually converges towards a reliable recognition. This work has been integrated in [Branson 2014] with an interactive labeling of deformable-part model presented in [Branson 2011].

In the visipedia project, the quality of the crowdsourced labels has also be studied. An online algorithm has been proposed in [Welinder 2010] to determine the most likely value of the expected labels from multiple noisy annotations. As a by-product the algorithm produces an estimate of annotator expertise and reliability. It actively selects which images to label based on the uncertainty of their estimated ground truth values, and the desired level of confidence.

To conclude this section in which we reviewed closed loop techniques where content analysis helps making the optimal decisions about human interventions (e.g. annotations, responses to well chosen queries), we can get back to the key message highlighting that humans become assistants to computer algorithms and, reciprocally, computer algorithms improve human efficiency. By mingling the power of crowdsourcing and the potential of automatically computed informations (edge detection, trimaps computation, object candidate production), Vittayakorn and Hays [Vittayakorn 2011] defend powerful annotation scoring functions that practically bridge the gap between both halves of our related work chapter.

2.3 Summary and Conclusion

In this chapter we have reviewed part of the state of the art in visual content understanding. We have focused our attention on techniques that involve humans at some point in the process. Involving humans in such tasks is probably the only way to go beyond the semantic gap that still remains a challenge in many domains.

First we have explained how annotation interfaces allow the gathering of large visual

content datasets augmented with ground truth solutions to visual content understanding problems. We have shown how these datasets can then be used, in the supervised machine learning paradigm, as training data for content analysis to learn how to associate high level information to low-level visual features. We have also described the emergence of new communities that study the problems inherent to the process of outsourcing tasks to a crowd of users. The challenges of motivating users as well as controlling their inputs quality have been explored. And we have finally briefly introduced a class of algorithms that try to infer knowledge from implicit users interactions.

Then we have shown how the traditional content analysis techniques could come handy in helping users annotating content. We have described annotation tools that diminish the need for users contributions by either providing assistance (e.g. candidate segments to be annotated) or expanding users inputs (e.g. segments propagation in PerLa). We also reviewed algorithms for interactive segmentation and co-segmentation that try to minimize the amount of interactions needed from users while obtaining the best possible segments. And we have concluded by describing one of the latest tendencies in visual content understanding, that takes advantage of the active learning paradigm to request only optimal annotations from users.

All these works perfectly contextualize our own contributions. In the same spirit, we have studied how to infer semantic knowledge (Regions of Interest) from implicit user interactions. This set of contributions constitute part I of this dissertation. The regions of interest are then used to feed several applications such as video retargeting or video mashups, and to build new interfaces that can also be used to gather mroe information about the contents (zoomable video player, video query interface).

In the part II of this thesis, we study problems that require a higher level of semantic information: object detection and image segmentation. We are interested in particular in comparing the performances of different classes of users, such as experts and non-experts. We also introduce a Game With A Purpose through which users provide information about images. We study how this tool for collecting implicit users interactions can be used to perform image segmentation and object detection, and we address this fundamental question: what are the benefits and drawbacks of implicit interactive techniques?

Part I

Regions of Interest Detection from the Crowd

Regions of Interest Detection from the Crowd: Overview

Detecting the Regions of Interest (referred to as ROI from now on) of a visual content usually constitutes the first step towards understanding their visual content. This problem can be mapped to the saliency detection problem, in which algorithms try to model the human vision system to detect regions (within images or videos) that will attract the human eye's attention. ROI and saliency detection typically produce outputs that are similar in nature. A saliency map (resp. an interest map) is a black and white image that indicate for each pixel its level of saliency (resp. of interest). The main difference is that an interest map has more semantic meaning than a saliency map, since the notion of interest implies that the content has been understood – even partially – by humans.

In this part, we argue that ROIs can be inferred through an analysis of how humans interact with visual information (images or videos) through specific interfaces. We also show that an automatic analysis of the image or video brings additional and complementary information, which makes the combination of usage analysis with content analysis very relevant in this context.

Our contributions are divided into two chapters. In chapter 3 we focus on the interfaces and algorithms to produce interest maps, both in 2D and in 3D. In chapter 4, we present and evaluate some applications that benefit from interest maps.

Chapter 3

Crowdsourcing Regions Of Interest

Contents

Detecting regions of interest (ROI) is a fundamental step for visual content understanding. First it reduces the complexity of the task since understanding a visual content's ROI should be sufficient to understand the content itself. In addition the detected ROI contain enough semantic information to be used in an application, as will be shown in chapter 4.

ROI detection is a difficult problem because it involves - at least partially - a semantic understanding of the scene. It also implies a knowledge of the consumer's (the human who watches the visual content) motivations. How to know what users would find interesting if we do not know why they watch the content in the first place?

Since it is very hard to perform ROI detection automatically, a natural idea is therefore to involve humans in the process. The task of determining ROI is rather simple to a human. The only issue is that the notion of interest is subjective, and very much dependent of each individual. This means that ROI determined by one user may not be valid for the majority of users.

This is why in this chapter we will describe how to find ROI by engaging a crowd of users. The contributions of many different users (a process known as "human computation") allows us to produce a more accurate list of ROI. In this chapter, we also present how to obtain the contributions from users in an implicit manner, through the use of interfaces. We use a Zoomable Video Player in sections 3.1.2 and 3.2.1, and we argue in section 3.3 that the collective actions of shooting videos at a public event reveals the humans interest on the scene.

3.1 Crowdsourcing ROI using Zoomable Video Player

3.1.1 Zoomable Video

Zoomable video is a term coined by Ngo et. al [Quang Minh Khiem 2010] to refer a video that is encoded and stored at multiple resolutions and supports dynamic cropping and random access into the spatial region in the video. Besides normal operation in the temporal dimension such as play, pause, fast forward, a zoomable video supports two new types of operation: zoom and pan. Users view a zoomable video with a display size that is smaller than the maximum resolution of the video (see figure 3.1). Zooming allows users to magnify a selected region in the video without loss of resolution (up to a maximum zoom level) and panning allows user to move the zoomed in region spatially around the video. Zooming and panning can occur even when the video is playing.

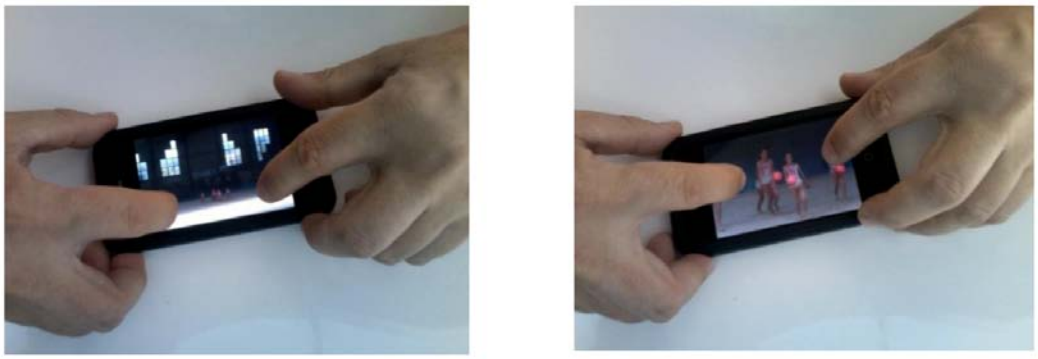


Figure 3.1: A use case scenario for the zoomable video player.

Zoomable videos are very helpful in situations where display devices are computationally constrained by the ability to decode and render high resolution video. Zoomable videos are also useful in cases where devices have access to streamed video over a low bandwidth network interface, resulting in the inability to stream bandwidth intensive high resolution video. Traditionally, computational and bandwidth constraints have been handled by scaling down the video temporally, spatially and in quality. Such an approach would result in loss of information, despite the fact that the capture devices were able to record the video at very high resolution. Zoomable video provides an alternative where users can select regions of interest from a low resolution video and view these regions at higher resolution. Such a scheme can satisfy both bandwidth and computational constraints in a more scalable fashion.

A zoomable video system can be built using a bit-stream switching architecture. The high resolution video is encoded at multiple resolutions. The lowest resolution video is first accessed and displayed to the user. When a user wishes to view a specific region at a higher resolution, user's intent is represented as a rectangular viewport. This viewport is first mapped to a higher resolution layer. Then a region corresponding to the specified viewport

is cropped from the higher resolution layer, and presented to the user.

The key requirement is the ability for dynamic cropping, which can be a challenge in encoded video. Methods to handle dynamic cropping and other issues related to streaming of zoomable video have been presented in [Quang Minh Khiem 2010].

The fundamental idea of our work is that users, when provided with a zoomable video player, will use it to visualize the regions they are the most interested in. In other words by interacting with the player, users choose to restrict their viewing window to a smaller and more interesting sub-region of the frame, thus pointing out the ROI. This idea is similar in essence to the works introduced in section 2.1.3.

3.1.2 Zoomable Video Player and Data Collection

We first describe how we collected a first generation of traces (in 2010) of viewports selected by users via our zoomable video player.

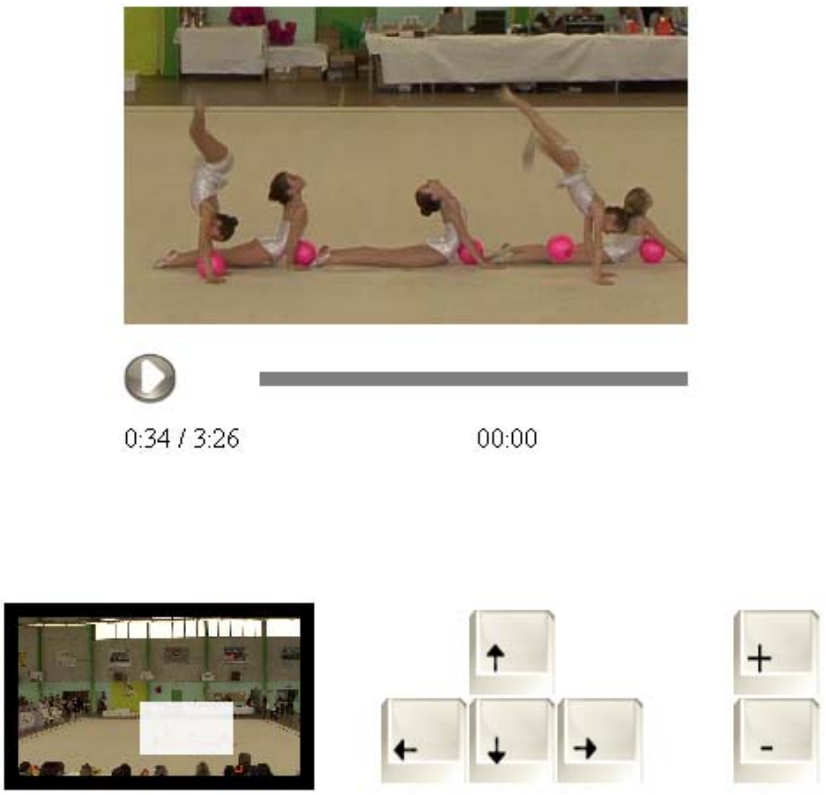


Figure 3.2: User interface of the zoomable video player

Figure 3.2 shows a screenshot of the zoomable video player. A video display of size 320 × 180 can be found on top. The small view display simulates the scenario where a video is watched under resource constraints (low bandwidth, or small screen size, or low

decoding capability). The lower left corner of the interface contains a thumbnail window of size 160×90. The thumbnail always display a scaled down version of the source video. The user's viewport is shown in white whenever users zoom in, to provide the users with the context. A viewport is a rectangular window that materializes the region currently visualized by a user. The lower right corner shows some control buttons where the users can click to zoom and pan, as an alternative to using the mouse.

Video sequences in our system are of a HD resolution (1920×1080). By default (zoom level 0), the video is scaled down for playback in the video display. At this zoom level, the user can view the whole video, although with a lower level of detail. Five levels of zoom are supported (levels 1 to 5). Viewing a region at a higher zoom level is equivalent to viewing a cropped region of size 320×180 from a higher resolution version of the video. For instance, at zoom level 5, users see a 320×180 region from the original source video. At zoom level 4, users see a 640×360 region from the original HD video scaled down to fit the 320×180 window. Users can use either the $+$ or $-$ buttons or the scroll wheel on the mouse to zoom in and out.

Users, after zooming in, can drag the mouse on the video display to pan. They may also use the arrow buttons for finer grain control of the panning. Users can pan anywhere within the content of the original HD video.

Despite using user interfaces that are similar to those used by other zooming interfaces (e.g., Google Earth), we found in our initial deployment that many users tend to play around with the different possibilities of the user interface. To prevent this, we have asked the users that use our system to watch an instructional video, showing them on how to use the interface. After watching the video, we force our users to go through a small practice session, in which they are asked to complete a step-by-step tutorial on how to use the user interface. The practice session must be completed before users can start watching any zoomable video.

The system logs all mouse and keyboard operations performed by the users, along with their timestamp, when they watch the zoomable video (the interactions from practice sessions are not logged). From these logs, one can replay the users' zoom and pan action, as well as determine the viewport of each user on each frame.

3.1.3 Region of Interest Modeling

We now describe how we detect and model the ROI from the logs of users' interactions with a zoomable video player. We first construct a user interest map by aggregating the selected viewing regions from multiple viewers. This user interest map, or heat map, might yield multiple hotspots – regions that are popularly viewed by the users. The challenge then is to extract ROIs based on these hotspots. With the video retargeting application in mind (section 4.1), we also want to extract some viewports of interest (see second row of figure 3.3). Viewports of interest are regions that contain one or several ROIs and that are candidates to appear in the retargeted video.

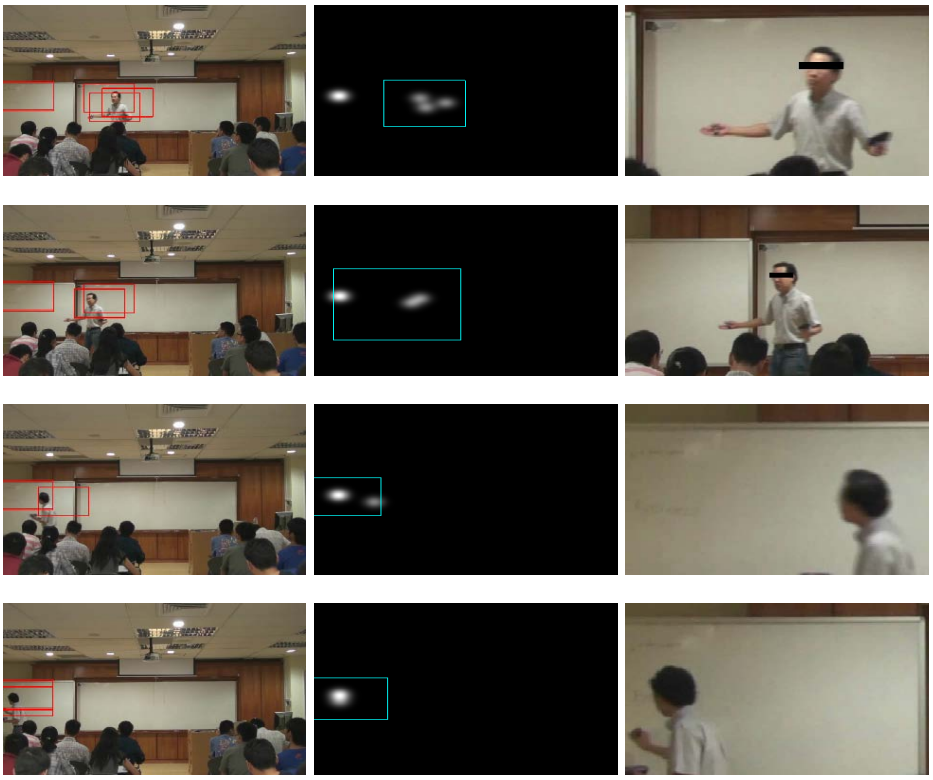


Figure 3.3: Approach overview: four frames and a few viewports (first column), heatmaps and detected viewport of interest (second column), retargeted frames including reframing techniques (last column). (Note that the lecturer's eye are redacted for anonymous review).

User interest map construction

During playback, for any frame, each user chooses a rectangular viewing window (i.e., viewport) via zoom and pan. We observed that users naturally center the window (e.g. the focus) on their preferred regions. Thus each pixel from a viewing window does not have the same interest level. The closer a pixel is to the window's center, the higher should its contribution to the user interest map be. In line with previous work [Han 2006], we model the pixel-wise interest levels using gaussian probability density functions (pdf). Figure 3.4 illustrates our method. A viewing window centered on $\mu = (u, v)^T$ is shown superposed on a frame. We also plot the gaussian pdf with the same center and a covariance matrix Σ that fits well with the viewing window. Theoretically the matrix Σ is chosen such that the ellipse whose equation is $d_M(\mathbf{x}, \mu)^2 = (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu) = 13.8$ is tangent to the viewing window VW[1]. As a consequence, we have a gaussian weight at each pixel and a gaussian elliptical footprint for the associated viewing window (Figure 3.4). Now we have to aggregate the data from multiple viewers to produce our user interest map. We simply blend the footprints by accumulating gaussian weights and normalize the result by dividing

[1]The threshold comes from the Mahalanobis distance $d_M^2 \sim \chi_2^2$ such that $\forall \mathbf{x} \in \text{VW} \quad P\{d_M(\mathbf{x}, \mu)^2) \leq 13.8\} \geq 0.999$

it by the number of windows involved for each pixel.

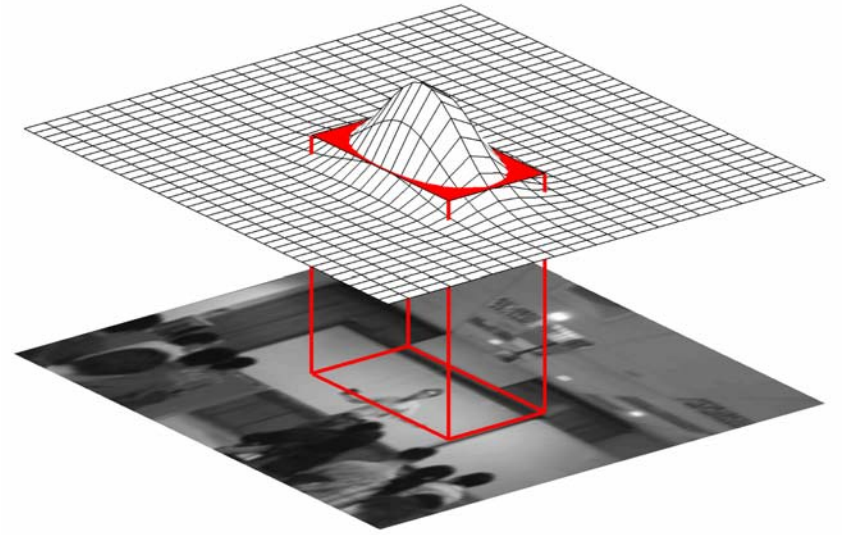


Figure 3.4: Gaussian weights within a viewing window

Middle column of Figure 3.3 shows typical heat maps: the dark areas represent less popular regions in the video frame, whereas brighter areas have cumulated many votes and highlight preferred regions of interest. The clustering of hotspots indicates that users tend to agree on the most interesting regions, with only a slight variations on zoom factor and focus. Less important ROIs may differ from viewer to viewer. Another heat map example is given by Figure 3.5, extracted from another test video (depicting a magic trick). At this stage the reader should ignore the green annotations, to be explained later.

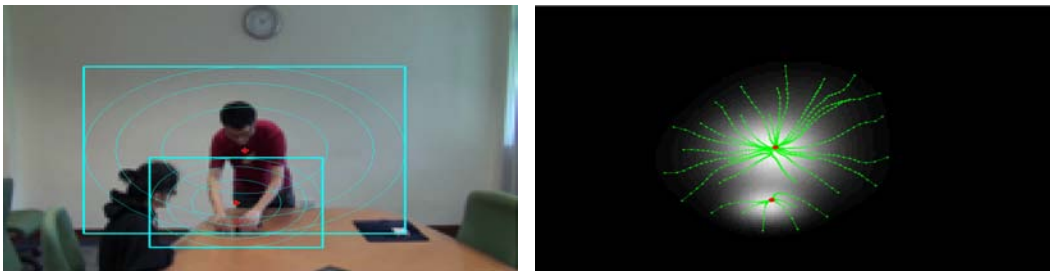


Figure 3.5: Viewports for a magic trick video (left) and multiple starts of Mean Shift, converging to two modes (right), on the corresponding user interest map.

Modeling ROIs as a Gaussian Mixture Model

The next step is to extract ROIs. There are many possible techniques to detect and model ROIs of a heat map. Huang et al. [Huang 2009] cite three main approaches: a simple binary mask, a set of focus points (FOA - focus of attention), an importance map viewed

as a probability density function using a mixture distribution. We favor the letter in this work, and in the whole chapter.

We used Gaussian Mixture Models (GMM) to model interest maps because of their multiple good properties:

- Easy **aggregation** of different contributions.
- Natural emergence of the concept of **popularity** for a ROI.
- Approximation of any interest distribution.
- Extension to higher dimensions (see section 3.3 for an example of the use of 3D GMM).
- **Simplification** (reduction of the number of gaussians in the mixture).

GMM simplification is one of the key aspects of our work, since we aggregate contributions of multiple users and extract a few (typically, from 1 to 4) ROIs. The main drawback of GMM is the complexity of the estimation of its multiple parameters: we explain later in this section how we proceed.

Keeping in mind our video retargeting application (in section 4.1.2 we will see that modeling the ROI dynamics using a mixture model proves to be rather simple), we aim at modeling the heat map at frame t as a GMM involving K ROIs. We assume that the interest location variable $\mathbf{x} \in \mathbb{R}^2$ leading to a heat map follows a GMM pdf:

$$p(\mathbf{x}|\theta_t) = \sum_{j=1}^{K} \omega_{t,j} \;\; p(\mathbf{x}|\mu_{t,j}, \Sigma_{t,j}) \tag{3.1}$$

where $\omega_{t,j}$ are the relative importance weights of the K ROIs considered at frame t and subject to $\omega_{t,j} > 0$ and $\sum_{j=1}^{K} \omega_{t,j} = 1$. Each ROI's density is a normal probability distribution :

$$p(\mathbf{x}|\mu_{t,j}, \Sigma_{t,j}) = \frac{\exp\left[-\frac{1}{2}(\mathbf{x}-\mu_{t,j})^T {\Sigma_{t,j}}^{-1}(\mathbf{x}-\mu_{t,j})\right]}{(2\pi)\,|\Sigma_{t,j}|^{1/2}} \tag{3.2}$$

where $\mu_{t,j}$ and $\Sigma_{t,j}$ are the moments of the j^{th} gaussian ROI at frame t. The global set of parameters for frame t is $\theta_t = \{\omega_{t,1} \ldots \omega_{t,K}, \mu_{t,1} \ldots \mu_{t,K}, \Sigma_{t,1} \ldots \Sigma_{t,K}\}$. One may think that it is straightforward to estimate θ_t by EM (Expectation Maximization) but a major question that remains is the correct value for K (the number of ROIs). This can be formally expressed as a model selection problem [Huang 2009] but we actually show that a non parametric clustering technique and a smart method for the covariance matrix estimation fits better our need.

Parameters estimation and ROI detection

We use the *Mean Shift Clustering* technique [Comaniciu 2002] to automatically find the K modes (peaks of the heat map) and to subsequently assign pixel locations to the associ-

ated K clusters. The basic *Mean Shift* procedure starts from a pixel point and converges towards a stationary point of the density function (see the green traces corresponding to multiple starts on the figure 3.5 (bottom)). The set of all pixels that converge to the same mode defines a cluster. The method does not require prior knowledge except one parameter: the kernel bandwidth. A value of $\xi_d = 60$ pixels was selected since it is the right distance to separate significantly different viewing windows in full HD images.

Once we have the modes $\mu_{t,j}$, we also need good covariance matrices that produce ROIs well centered around the modes and small enough (in order to get a good attention window). We estimate each $\Sigma_{t,j}$ with the Minimum Covariance Determinant (MCD) estimator. This estimator is basically a robust estimator that allows to cope with outlying or spread data. Its principle consists in finding a covariance matrix able to capture a maximum amount of interest while having the smallest area. As previously mentioned, the area is proportional to the covariance determinant that is minimized by our MCD estimation for $\Sigma_{t,j}$. Moreover, the weigths $\omega_{t,j}$ are taken proportional to the clusters' sizes. The GMM parameters set θ_t is now fully defined. Figure 3.5 (top) shows two gaussians centered on the two modes (red points). Each gaussian is represented by a set of iso-values, the biggest one being tangent to the detected rectangular ROI.

3.1.4 Evaluation

We choose to evaluate our ROIs through the video retargeting use case. Indeed the visual quality and interestingness of a retargeted video - computed as described in section 4.1 - depends a lot on the quality of the ROIs. In other words, validating our video retargeting algorithm will also validate the quality of our ROIs. We present such an evaluation in section 4.1.4.

Nevertheless, we can say a few words about the zoomable video player introduced earlier (section 3.1.2). We have tested the player in multiple user studies ([Carlier 2010b, Quang Minh Khiem 2010, Carlier 2010a, Carlier 2011a]). The videos we have used during these studies share important features: they are HD videos (1920×1080 pixels) shot with fixed cameras. We worked on 5 different sets of videos:

- *Lectures*: they feature a teacher who comments slides and writes on a board (e.g. figure 3.3).
- *Magic tricks*: they feature a magician along with his assistant who performs simple tricks (e.g. figure 3.5).
- *Gymnastic*: this video features a group of performers who dance on a mat (e.g. figure 3.2).
- *Longjump*: these videos feature the end of a runway and a sand pit where an athlete is jumping (e.g. figure 3.12).

- *Coffeelounge*: this video features a group of persons interacting in a coffee lounge (e.g. 3.7).

We learned two principal facts from these studies. First our zoomable video player (figure 3.2) offers too many ways of doing the same interaction. Very few users utilize the inferior components of the interface (the thumbnail and the buttons). Users tend to interact directly on the video frame, very much like they would interact on an image. In the subsequent version (see section 3.2) of the zoomable video player, we removed these components.

A second interesting result that came out of the studies is the difficulty to interact with a video that displays a lot of movement. Typically, the *lectures* videos have a very static content: the teacher seldom moves, and often stays at the same position to write on the board for example. Users dis not have any problems to interact on these videos. Conversely, the sports videos (such as *Gymnastic* or *Longjump*) feature moving objects of interest, that force users to interact heavily in order to keep the ROIs in their viewports.

This issue is at the heart of the work presented in the next section, in which we introduce interactions based on content analysis to help users visualize moving objects.

3.1.5 Summary and Conclusion

We have introduced an algorithm to produce user interest maps from a zoomable video player's browsing history. Since users tend to naturally zoom into interesting regions within the videos, this aspect can be exploited, allowing us to use the aggregation of users' viewports to build interest maps. We then detect the hotspots of this interest map using the Mean-Shift algorithm, to obtain a set of ROIs. The ROIs are modelled by a mixture of Gaussians where each Gaussian is associated to an ROI.

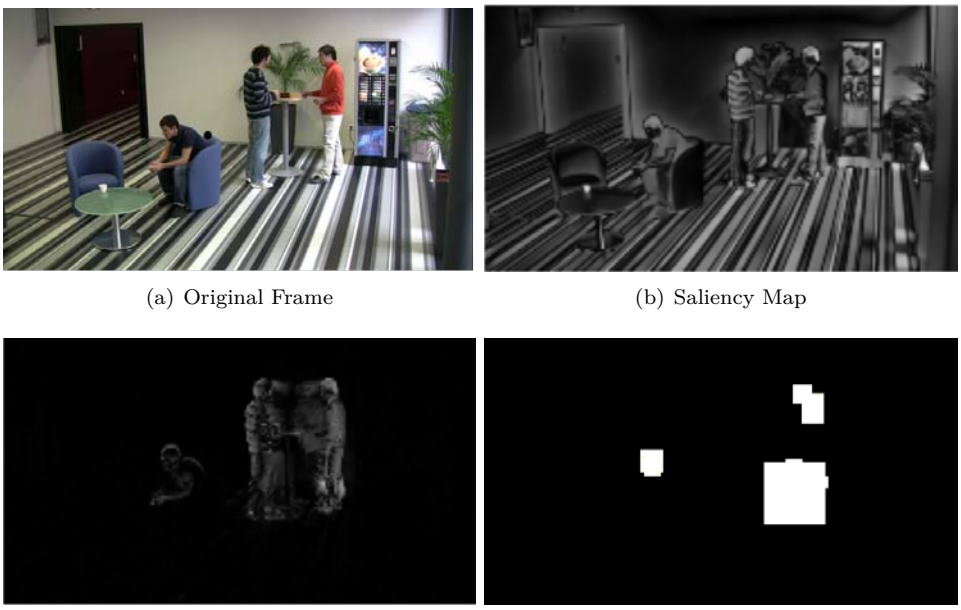
In section 4.1 we will explain how we use the resulting ROIs to produce a retargeted video. This work was published at ACM Multimedia 2010 [Carlier 2010a].

Note that the ROI detection presented here is only based on an analysis of users' inputs. It seems a good idea to include some automatically inferred information about the video. This is the purpose of the next section, in which we will bias users' interactions by introducing content-based recommended viewports.

3.2 Content-Aware Zoomable Video Player

In this section, we introduce a new zoomable video player that features recommendations. The recommendations aim at providing easier users' interactions, and are computed using content analysis. In a second phase, recommendations are recomputed to take into account interest maps obtained from users' interactions. We show in our evaluations that the new zoomable video interface with recommendations is helpful to users, and that the combination of content analysis with user interest maps gives the best recommendations.

3.2.1 Recommended Viewports in Zoomable Video



(a) Original Frame (b) Saliency Map

(c) Motion Map (d) Face Map

Figure 3.6: Content Analysis

This section gives an introduction to the notion of recommended viewport in zoomable video. We describe how a user would interact with the zoomable video using recommended viewport, and defer the discussion of how the recommended viewport is computed to the next section.

Recommended Viewport A key feature in our proposed interaction is the *recommended viewport*, which corresponds to interesting objects or events in the video, and is a region in the video that the user is likely to zoom into. Hovering the mouse over a recommended viewport reveals a white semi-transparent rectangular box (see figure 3.7). The user can zoom into the region by left-clicking the recommended viewport with the mouse button. When the mouse cursor hovers over one or more recommended viewports that overlap, only the most important recommended viewport is shown.

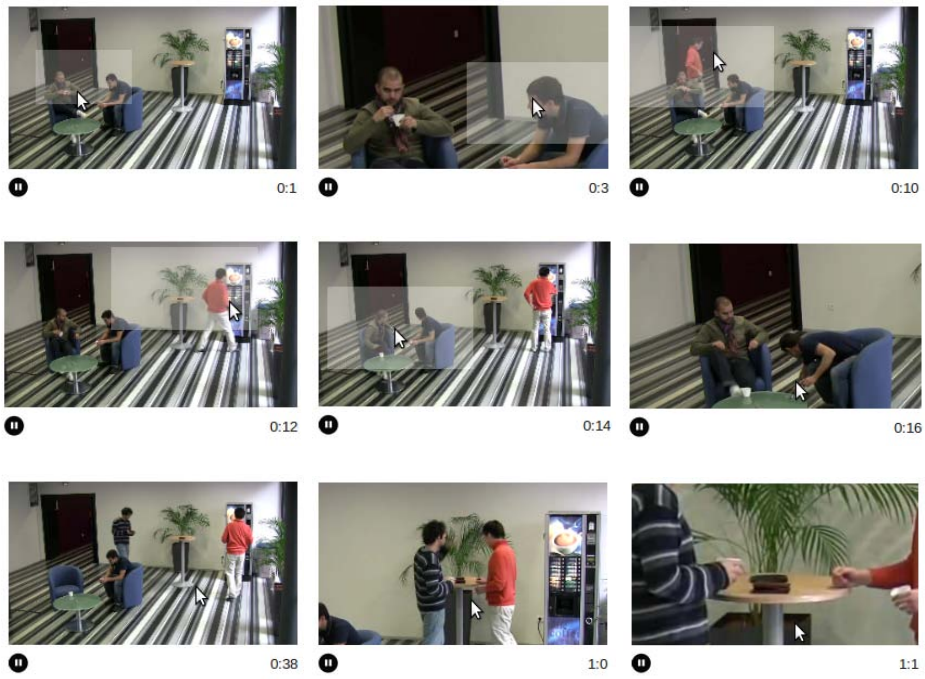


Figure 3.7: Zoomable Video Interface with Recommended Viewports

Users can left click outside of a recommended viewport to zoom in. In this case, the new viewport of the user is placed such that the center of the viewport is the coordinate of the mouse click. Right-clicking anywhere of the video zooms out.

The recommended viewport automatically moves to track a moving object of interest. If the user's current viewport is one that matches the recommended viewport (by clicking inside a recommended viewport), the user viewport pans automatically along with the recommended viewport.

One design decision we make is to limit the number of recommended viewports per zoom level to three. While using recommended viewport helps users to place their viewport easily with a single mouse click, it also restricts the viewport placement. Presenting too many recommended viewports to the users is not only too inflexible, but can also be confusing. We choose the value of three since we observe that in a typical video there are rarely more than three events of interest at the same time. Of course, this number can be configured to be higher depending on the content.

Implementation We implemented our interface on a Web browser using HTML5. The webpage is minimalist in design, and shows only a video canvas of size 320×180 and a play/pause button along with the current playback time. Videos of resolution 1920×1080 are loaded along with a JSON file that contains the recommended viewports. A Javascript crops and scale the video for display in the canvas, as well as highlights the recommended viewport according to the JSON file (when hovered by the mouse).

A critical ingredient to the success of our approach is the quality of the recommended viewports. In the next two sections, we detail how we combine content analysis and crowd-sourcing to compute the recommended viewports.

3.2.2 Content-based Analysis

The recommended viewports are determined based on a sequence of content analysis steps. We use saliency, and motion in the frame as the two main criteria to determine the possible regions of interest (ROI), which then serve as the recommended viewports. In cases where a face can be detected, face position is used as an additional criterion. Saliency is determined using the visual saliency features described in [Montabone 2010]. These features have shown good results for human detection, and hence the saliency is biased towards the presence of human or human-like objects in the video. Figure 3.6(b) shows an example saliency map.

Motion saliency is based on a moving average of frame differences and is similar to the work in [Wang 2009]. A single channel disparity image of two successive frames is added to a moving average and used in place of the current frame. As a result the long-term motion pattern is available in a single frame. Figure 3.6(c) shows what a motion saliency map looks like.

Face detection, for both frontal and profile faces, is done using the Viola-Jones face detector [Viola 2001]. We track faces across frames using a hue histogram of the detected face, with the CAMShift algorithm. Figure 3.6(d) shows the result of face detection.

Importance Maps Each video frame is now represented by three maps, one for each criteria mentioned above. The pixel values are single channel quantities ranging between 0 and 1. The frames representing the three criteria are linearly combined to give an *importance map* I. Dittrich et al [Dittrich 2013] explain in their work that linear combination is not the only possibility for the combination, but is the most often used.

The weights assigned for linear combination may be obtained in many ways. One approach is to weigh all criteria equally. Alternatively, one may analyze each video and determine the weights empirically. In our experiments, we use empirically derived weights 0.7 for motion, 0.2 for saliency, and 0.1 for faces. Finally, a more formal approach would be to use a training-testing model with least squares approximation for the solution. Let $S_{i,j}$, $M_{i,j}$, $F_{i,j}$ represent the saliency, motion, and face measures for the pixels at i, j in their respective maps. Then $I(i, j)$ is $\alpha S_{i,j} + \beta M_{i,j} + \gamma F_{i,j}$. Further, if we have N ROIs generated by human subjects as described in [Ngo 2011], and the probability p_i of a user choosing the i^{th} ROI is known, then the problem of finding weights may be written as the solution to a linear system of equations $Ax = P$. Here $x^T = [\alpha, \beta, \gamma]$, $P^T = [p_1, p_2, ..., p_N]$ and A is a $N \times 3$ matrix. Each element $a_{i,1}$ in the first column of A represents the cumulative sum of saliency of all pixels in the i^{th} ROI. Similarly the second and the third columns are for motion and face values. The least squares solution to $Ax = P$ is computed by the classic Moore-Penrose pseudo inverse.

Clustering of Important Regions The next step is to cluster pixel elements in the

(a) Bad Cut (b) Good Cut

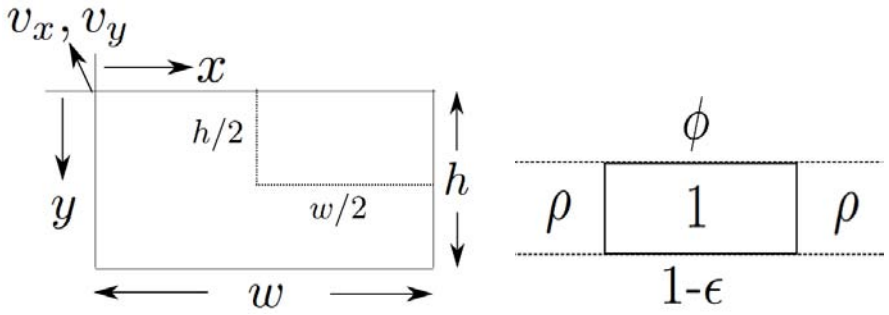
Figure 3.8: Frames Showing a Bad Cut and a Good Cut

importance map to reveal regions that are candidate ROIs. The pixel elements in the importance map of every image are selected based on a threshold. Selected pixels are clustered using Mean-Shift ([Comaniciu 2002]) clustering algorithm. We chose a 60 pixels bandwidth in our implementation since it allows to detect significantly different viewports in 1920×1080 videos. Once clustering is performed, the candidate ROIs are determined at each zoom level. All ROIs at a particular zoom level have the same dimensions, i.e., the dimension of the corresponding viewport at that zoom level.

A ROI is a good candidate if it encloses as many cluster points as possible without leaving out cluster points. Hence we need a measure to quantify the extent to which a viewport cuts a cluster into two or more parts. We define a term called the *cut of a viewport*. If a viewport does not cut a cluster, then its cut is very high. The concept of using a cut has been proposed in earlier work [Itti 1998, Liu 2006, El-Alfy 2007], but we use the notion of cut in conjunction with clustering to minimize cuts to the top portion of an object. Such an approach is better suited to aesthetically present human-like objects without cutting body parts. Figure 3.8(a) shows a case where the viewport cuts the clusters into two parts. There is a cluster region enclosed within the viewport, and the remaining cluster points fall outside the viewport. The region within the viewport encloses the lower part of two human subjects whose presence may be identified by the structure of the clusters shown. Hence, this viewport should be penalized. On the other hand figure 3.8(b) shows a viewport cutting a cluster such that the upper portion of the human subjects is enclosed, and the lower portions are cut out of the scene. This viewport should be treated favourably in comparison to the previous case. Hence the *cut of a viewport* should also account for how the cut is performed.

We now formally define how the cut is computed. For a frame f, let C_f represent all the cluster centroids in f. Let $E(c)$, where $c \in C_f$, be the set of pixels clustered around centroid c. Then $CUT(V_f, f)$, the cut of viewport V_f with top left coordinate (v_x, v_y) and height h (See also Fig 3.9(a)), is a measure for the extent to which $E(c)$ is fully contained within V_f:

$$CUT(V_f, f) = \frac{\sum_{c \in C_f} \sum_{p \in E(c)} W(p, V_f)}{\sum_{c \in C_f} |E(c)|}$$



(a) Coordinates (b) Cut Weights

Figure 3.9: Viewport Coordinate System and Weight Assignment for a Cut

where $W(p, V_f)$ with p at coordinate (x, y) is given by

$$W(p, V_f) = \begin{cases} 1 & \text{if} p \in V_f \\ 1 - \varepsilon & \text{if} y > v_y + h \\ \phi & \text{if} y < v_y \\ \rho & \text{otherwise} \end{cases}$$

and $1 > 1 - \varepsilon >> \rho > \phi$ (See also Fig 3.9(b)). $CUT(V_f, f)$ reaches a maximum value of one when V_f contains all cluster points.

Our goal is to find the best viewports that maximize $CUT(V_f, f)$. This step is achieved by evaluating all candidate viewports and selecting a few that have the highest value of cut.

Tubing: Finding Recommended Viewports over Time Viewports change in every frame. When users select a recommended viewport in a frame, the same viewport may not be optimal when recommended in the next frame. The system has to switch to a nearest viewport, causing a virtual camera shake. To minimize the irritation caused by frequent and abrupt change in viewport position, we compute an optimal strategy to switch viewports while maintaining a smooth, linear transition of the viewport position. Such a linear change manifests as a virtual camera pan.

To compute a virtual camera pan, we first designate some frames in the video as key frames. All frames falling between two key frames constitute a *shot*. The viewport is allowed to linearly change position within the shot. We expect the viewport at the beginning of a shot to smoothly change to a viewport at the end of the shot.

In our implementation, we use one key frame every 20 frames. There is a trade off involved in the choice of inter key frame distance. More key frames would lead to less stable viewport (more shaky) and fewer key frames would lead to less optimal recommended viewports (the reason for which will become clear later).

There are multiple candidate viewports at different zoom levels at the beginning and at the end of a shot. We can choose different combination of the starting viewport and ending

viewport. Each of these combinations result in a spatial-temporal trajectory of viewports across the frames in between the two key frames. We refer to this trajectory as a *tube*. Since we linearly interpolate the viewports in a tube, a viewport in an intermediate frame may not be the optimal viewport for that frame (as determined by the cut).

Our goal is therefore to find a good tube that gives good overall viewport quality across all frames in the tube. To this end, we define four metrics to evaluate the quality of the tube. Given a tube $\mathcal{T} =< V_f, V_{f+1}, \ldots V_{f+N} >$ we first consider the cut metric, which is used to prevent violation of aesthetic rules, especially for human body. We define the *cut of a tube* as the sum of all cuts of viewports in the tube. Second, we consider the importance of the tube, and define the *heat* of a tube as the sum of all importance value in every viewport $V_{i,i\in f\ldots f+N}$ in the tube, i.e., the pixel values in the importance map of each frame that falls in the viewport. Third, we consider the *temporal coherence* of a tube. We aim at preserving the motion of foreground objects within a tube. We proceed as in [Beleznai 2006] and track our clusters over time by *mode seeking*. A given cluster is tracked across l frames creating a temporal chain of cluster centroids $< c_j, c_{j+1}, \ldots c_{j+l} >_{id}$ that starts at frame j. Each chain is assigned an unique id, and every cluster whose centroid is part of this chain is labelled with that same id. With this simple method clusters might split or merge as we track, creating new chains. This is not an issue as it is not required to precisely track and segment objects to ensure temporal consistency [Wang 2009]. The temporal coherence of a tube can then be computed. For every pair of viewports $V_{f'}$ and $V_{f'+i}$ in the tube, a score proportional to i is added to the tube's coherence value $COH(\mathcal{T})$ every time clusters with the same id can be found in both $V_{f'}$ and $V_{f'+i}$[2] The coherence is then normalized to range into $[0, 1]$. Finally, we consider the spatial displacement of the tube. We define the *regularity* of the tube as the measure for rate of change of the viewport positions within the tube. We compute regularity as

$$e^{-\frac{\left\|v_f - v_{f+N}\right\|_2^2}{N^2}},$$

where v_f and v_{f+N} are the initial and final viewport positions respectively, and N is the number of frames in the tube $\mathcal{T}$. This metric penalizes large displacement of viewports with an exponential weighting function.

To find the set of good tubes between two key frames, all possible tubes are computed and assigned a score by simply summing the four metrics. The top k tubes with the highest scores are chosen and form the recommended viewports between the two key frames (we use $k = 3$ in our implementation). An example of the result of creating tubes is shown as viewport sequences in figure 3.10.

[2] $COH(\mathcal{T}) \propto \sum_{i=f}^{f+N} \sum_{c\in V_i \cap C_i} \sum_{k=i+1}^{f+N} \phi(c,k)$ where $\phi(c,k) = k - f$ if the cluster centroid c has an id that can be found among the ids present in V_k.

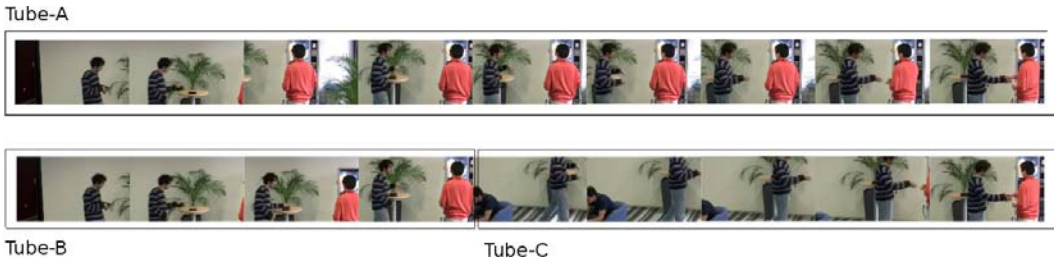


Figure 3.10: Viewport sequences showing Tubes. Tube-A is a long tube showing a scene where a person in motion is followed. Tube-B and Tube-C are other candidate Tubes rejected by the Tubing algorithm. Tube-B is short and has a low Regularity measure, while Tube-C is not only short but also has a lower cut value that manifests as a not so aesthetic framing

3.2.3 Combining Content-Based Analysis and Crowdsourcing

Using the approach detailed in the previous section, we have viewports to recommend to users (step 3 of Fig 3.11). We then start collecting user interaction traces with our zoomable interface. The goal is to find an approach to refine the content based viewport recommendation with the user selected viewports.

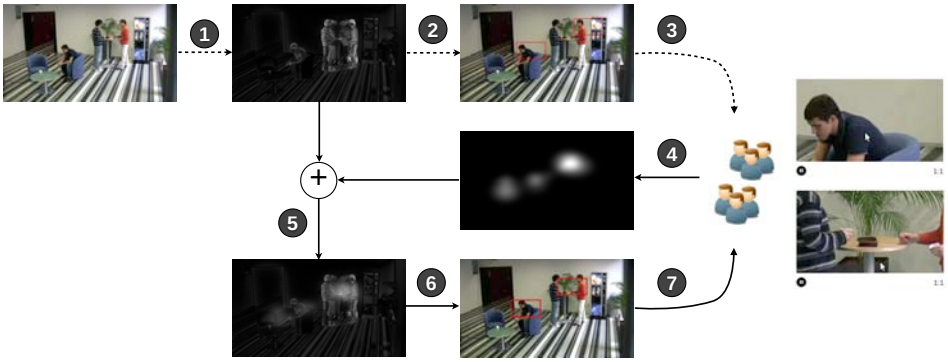


Figure 3.11: Overview of Approach

While zooming with the interface, each user selects a viewport at a given frame f. This viewport V_f is a rectangle whose top left corner is positioned at (v_x, x_y) and is of size (w, h). The viewport crops the important region the user is interested in. Since the level of interest of each cropped pixel differs with respect to its position within the viewport, as in [Carlier 2010a] we assume that users naturally center the viewport on the most interesting area. We then model the interest within a viewport as a gaussian pdf being centered at $\mu = (v_x + w/2, v_y + h/2)$ with a covariance Σ constrained to the dimensions of the viewport: $\Sigma \propto \begin{pmatrix} w^2/4 & 0 \\ 0 & h^2/4 \end{pmatrix}$ and $\int_{V_f} \mathcal{N}(\mu, \Sigma) = 0.99$.

A user interest map UIM_f associated with frame f is then crowdsourced by accumulating the interest levels from multiple users. If we collect K viewport traces from

K users who have zoomed on f, a gaussian mixture model can be computed such that $UIM_f^K = 1/K \sum_{k=1}^{K} \mathcal{N}(\mu_k, \Sigma_k)$.

The first image in Figure 3.12 shows $K = 16$ viewports selected by users while watching a long jump video with our interface. The second image shows the associated user interest map generated using the gaussian mixture model described earlier. In this example, users focus on the sand pit because they were asked to estimate the length of the jump.

With this simple formulation, the user interest map computation stabilizes after only 10 or 15 users. We observed that the KL-divergence $KL(UIM_f^{K+1}||UIM_f^K)$ is negligible for $K \geq 15$. Note that in this paper we do not consider any sequential estimation of the user interest map where a weight could be used to de-emphasize old traces in a time-varying context.

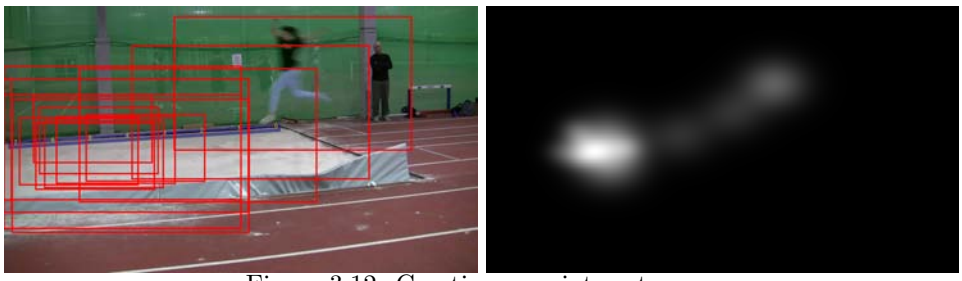


Figure 3.12: Creating user interest maps

As shown on step 5 of Fig 3.11, we then use the implicit feedback from users as another modality in the computation of importance map. We merge user interest maps with content-based importance maps by assigning them an equal weight. How to properly weight remains an open question: we plan to study the performances of different (either static or dynamic) weighting strategies in our future work. Yet experiments presented in the next section show that this simple strategy already demonstrates significant success. Indeed by applying the algorithms presented in section 3.2.2 to this new importance map, we obtain updated recommended viewports that match intention of the users better.

3.2.4 Evaluation

Video Sequences. We use two types of videos to assess our work. The *longjump* videos are about 30 second long and feature an athlete running and jumping in a sand pit.

The *coffeelounge* video is longer and semantically more complex. Fig 3.7 summarizes the action taking place in a coffee lounge. At the beginning of the video, two people are sitting in the foreground on blue sofas (times 0:01 and 0:03). A new person in orange sweater is then entering the scene, and loses his keys while removing his wallet from his pocket (time 0:10), before reaching the coffee machine and staying there (time 0:12). At the same time, one of the two seated people picks up the keys (time 0:16) and hands it over to his friend who leaves the scene. After this theft, a fourth person arrives and goes

to the coffee machine (time 0:38), after leaving his wallet on the same table as the person in orange (see also Fig 3.10). At the end of the video, the two people from time 1:00 (Fig 3.7) take their wallets and leave the scene, leaving the thief alone in the room.

Interaction Techniques. We built four variants of user interfaces for zoomable video. The version we proposed section is denoted as RC+U, which stands for Recommendation based on Content and Usage.

To study the effect of combining usage analysis with content analysis, we setup a version of the user interface that uses only recommended viewport computed using content analysis, without considering user access pattern. We call this version RC (Recommendation based on Content). This version is equivalent to the output of Step 3, after the process described in Section 3.2.2.

The third variant of the user interface we setup is called NR, which stands for No Recommendation. The purpose is to study the effects of presenting recommended viewport to the users. All interaction elements in this user interface remains the same, except that the recommended viewports are removed. This interface is equivalent to the one introduced in the previous section 3.1.

Finally, we setup a variant of the interface which we refer to as NZ, standing for No Zoom. We use NZ in one of our control experiments. NZ does not allow any zooming or panning.

Methodology. We evaluate the three successive versions (NR, RC and RC+U) of our interface by conducting the following user study. In our experiments, the user traces from RC are used to compute the recommended viewports of RC+U.

We assign tasks to users where zooming may be useful. We ask them to estimate the jump length in the *longjump* videos and we ask them if there are key thefts and/or wallets thefts in *coffeelounge*.

First we provide a first group of users with NZ, which plays only the low resolution (320×180 px) version of the video sequences without any interaction. We want to evaluate how well users answer the questions without zooming.

The core of the user study involves comparing the three versions of our interface: NR, RC and RC+U using three independent set of 20 users Users0, Users1 and Users2.

Except for NZ where no interaction is available, we always start our user studies with a learning phase. We first demonstrate the features of the interface, and then observe how users interact with our training video (one of the *longjump* videos). We never continue the study without explicitly reminding the user of interactions he/she did not try.

Then we explain to users what task they have to complete while viewing the next video clips. We always present the clips in the same order: *longjump* and *coffeelounge*. We collect a user's answers at the end of each clip. We let users watch the video as many times as they want and we record every interaction into a database. In average the test lasted between 8 and 10 minutes for each participant.

Participants. We collected traces from 16 females and 54 males (total 70 participants),

with an age ranging between 19 and 55 years old. Among these users, 10 were presented NZ and 20 each were presented other interfaces.

Number of Interactions. We analyze the traces to count the number of user interactions in each of NR, RC, and RC+U. Figure 3.13 shows the results. We first observed that there is no significant difference in the number of zooms, but the number of pans when using NR is on average almost twice as much as RC+U. Since pans are used mostly to position the viewport correctly, this results show that the recommended viewport is useful in reducing the number of interactions. The number of pans for RC+U is also less than RC, indicating that the quality of recommended viewports for RC+U is better, as users pan less using RC+U. This point will be further elaborated in the next result.

Note that *coffeelounge* lasts 1 minute 22 seconds while *longjump2* (one of the *longjump* videos) is only 35 seconds long, and this explains the difference in the number of panning for each video.

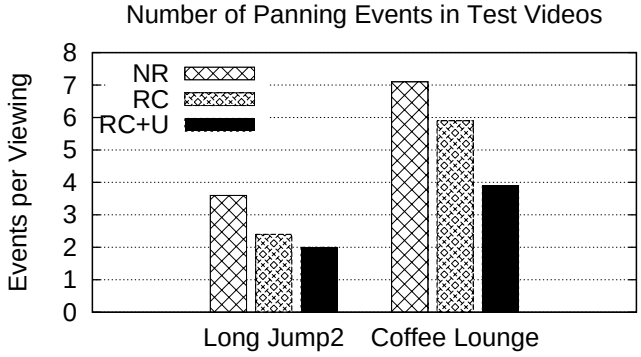


Figure 3.13: Number of panning events per view session

Number of Recommended Viewport Selected. To further compare the recommended viewports between RC and RC+U, we analyze the trace at one particular event in the *coffeelounge* video, to understand how the recommended viewport are clicked.

When Users1 zoom on the *coffeelounge* video (with RC), the recommended viewports are clicked 45% of the time. The remaining 55% clicks are outside the recommended viewports (Table 3.1). The first row of Table 3.2 gives insights into the distribution of zoom levels of those 45% clicks. It shows that users rarely zoom to the maximum level (i.e. close-ups). This result highlights the importance of the relationship between content semantic and participants' tasks. In this task, automatically detected close-ups are not useful enough to successfully complete the task.

However, RC+U exhibits better performance regarding the number of recommended viewports clicked like we see on the second row of Table 3.1. The ratio of clicks on recommended viewports and outside shows that recommended viewports are more relevant. This underlines the interest of combining content analysis with crowdsourcing to learn better ROIs. Moreover, learning better ROIs affects the distribution of zoom levels, summarized

by Table 3.2. As shown below, emergence of task-relevant close-ups encourages users to zoom more when needed.

We observe the same phenomenon on *longjump2*. Recommended viewports in RC are not really selected by users (only 18%, see first row of Table 3.1). But once combined with user maps, new recommended viewports are twice as much clicked by users (40%).

Video	**longjump2**	**coffeelounge**
RC	18%	45%
RC+U	40%	55%

Table 3.1: Percentage of clicks in recommended viewports for RC and RC+U

Interface	960×540 **px**	640×360 **px**	320×180 **px**
RC	73.2%	25.6%	1.2%
RC+U	24.6%	42.5%	32.9%

Table 3.2: Size of the recommended viewports clicked by users on *coffeelounge*

In summary, we found that integrating user interest maps to improve the relevance of recommended viewports yields a better recommendation. Users more often used the recommended viewports resulting in lower number of panning events.

Understanding Video Content. Previous results show that recommended viewports from RC+U are selected more often than the ones from RC, but does it mean that it helps them understand the content better? The answer, as shown in this paragraph, is yes.

Table 3.3 presents users' answers to the questions based on the task specific to *coffeelounge*. As a reminder, we asked users whether or not a key was stolen (the correct answer is yes), and whether or not a wallet was stolen (the correct answer is no). Whereas it is quite easy to spot the theft of the key even without zooming, there is an ambiguity with regard to the wallets (it may appear as if they were exchanged). Zooming in to the region of the wallets help resolve this ambiguity.

We noticed that 70% users who see the video at a low resolution (NZ) spot the key theft, whereas only 50% identify that no wallets have been stolen. We actually observed during the study that users tried to guess the answer because they could not see accurately, and indeed the answers were equally distributed as yes and no. This result provides us with a lower bound to compare interfaces NR, RC and RC+U.

The percentage of good answers is higher with NR thanks to the zooming functionality, especially for the wallets question (70% of good answers). However results are disappointing for RC. Guiding users with recommendations is potentially double-edged: since they follow the recommendations (Table 3.1), the quality of their answers is correlated to the relevance of the recommended viewports with respect to the task. About the specific wallets task, it is understandable that content analysis alone fails to detect the region around the wallets as one of the prominent ROIs.

We get the best answers with RC+U, which combines content analysis and crowdsourc-

ing. Indeed we observe in Fig 3.14 that crowdsourcing complements content analysis to create a new recommended viewport located on the wallets.

Interface	NZ	NR	RC	RC+U
Is there a key stolen ?	70%	75%	70%	**85%**
Is there a wallet stolen ?	50%	70%	50%	**75%**

Table 3.3: Percentage of right answers to the questions

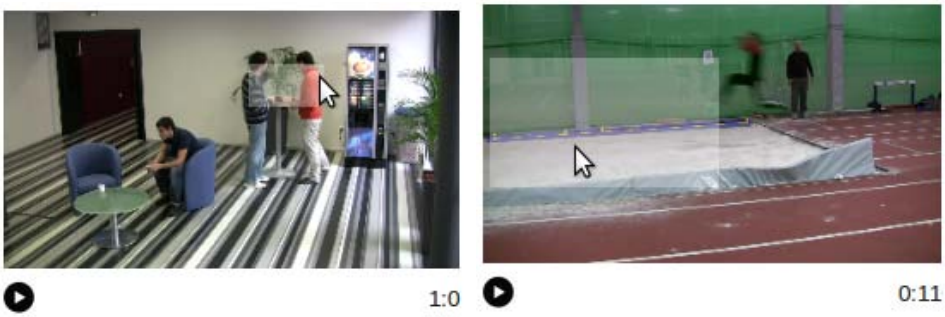


Figure 3.14: Example of ROIs emerging in RC+U: close-up on the wallets in *coffeelounge* and sand pit in *longjump2*.

We do not discuss the answers to the task for *longjump2* because the task is not discriminant enough to create differences in the quality of answers when different interfaces are used. As the users have a very specific task in the longjump2 video, i.e to identify the length of the jump, the length of the jump is localized to the sand-pit. Hence users directly zoom into the region enclosing the sand-pit irrespective of the interface used. However consistent with *coffeelounge* task, we observed the emergence of a recommended viewport particularly suited to the task (see Fig 3.14). Conversely some recommended viewports have a low interest with respect to the assigned task and have not be selected by Users1. As a consequence, they disappeared in RC+U making our video interface adaptive and user-centric.

3.2.5 Summary and Conclusion

In this section we have introduced the novel concept of recommended viewports in the context of zoomable video. The recommended viewports provide an assistance to the users, by indicating them which candidate regions they might be interested in zooming into. We have shown how to combine the content analysis (in our case, a combination of saliency, motion, and face detection) with an analysis of the interface usages (as introduced in the section 3.1) to improve the quality of the recommended viewports. This work was published in [Carlier 2011a].

In this work, content analysis serves two purposes. First, it provides a temporary solution to the problem of ROI detection when we do not have traces of users' interactions yet.

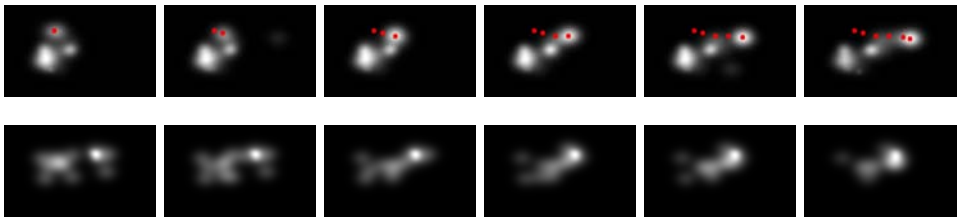


Figure 3.15: User interest maps computed on the same video at the same time interval through two different interfaces. On top, the interface with recommended viewports described in section 3.2.1 was used ; below, the interface described in section 3.1.2 was used.

Second, it helps biasing users interactions (through recommended viewports) which eventually positively affects the quality of the resulting user interest maps. For example, when an ROI is moving, our algorithm based on content analysis (including motion detection) will probably generate a recommended viewport that automatically follows the ROI. In that case, the resulting user interest maps will benefit from the recommendations because users tend to watch the ROI through the recommended viewport. With the zoomable video player described in section 3.1, users would have to manually pan in order to follow the ROI which would result in an imprecise user interest maps (figure 3.15, bottom row). Thanks to the recommended viewports, the resulting user interest maps is more insightful: ROIs are well focused and appear clearly (figure 3.15, top row).

In the next section, we consider the problem of estimating a 3D user interest map from videos shot by multiple users.

3.3 Crowdsourcing 3D Interest Maps using multiple synchroneous video streams

In the previous section, we have explained how to find ROIs using a combination of content analysis and users behavior analysis. It was reasonable to use content analysis since in the use cases we considered, the ROIs were often salient, and/or moving, and/or people (recognizable via face detection). We also took advantage of the assumption that users often zoom in on a region when they are interested in it. Zoomable video is thus a good ROI detector.

In the same spirit than zoomable video, there are other actions a user can make to manifest their interest towards a visual content's region. Taking a picture is a good example of such actions: one usually takes a picture to remember a happy moment, or capture a beautiful landscape, etc. In any case, the picture reveals the interest of the photographer who made the effort to grab a camera and shoot a scene.

In this section, we consider the common use case of public performances where many people in the audience are filming the scene (figure 3.16). We generalize our previous contributions to build 3D interest maps that model the crowd's level of interest in the 3D space. These 3D interest maps are computed based on usage analysis (the users shooting videos from the scene) and on content analysis, as we will now explain in details.

3.3.1 What is a 3D Interest Map?

Understanding and predicting user interest is a key issue in many multimedia applications. In order to efficiently compress, store, retrieve, transmit, recommend, display any media item, it is helpful to estimate users' level of interest in it.

Beyond saliency. In some situations, user interest and attention are guided by a task to be performed, in other free-viewing situations a user may subjectively gaze at most interesting items. In the latter sense, a saliency map associated with an image often serves as a gaze predictor and can be interpreted as a 2D map with a type of "interest level" (preferably normalized, between 0 and 1) assigned to each pixel. Unfortunately 2D saliency models still fall short of predicting semantically interesting objects perfectly. With a 3D content in mind, one can formally define a 3D saliency map by generalization and determine saliency at a given 3D voxel by how different this voxel is from its surround in color, orientation, depth etc. Similar to many 2D saliency models, this generalized 3D saliency estimation does not integrate any semantics and is a poor interest predictor for a 3D scene like the one depicted in Figure 3.17. In this example, a semantic definition of the interest (in 3D) would probably assign the highest interest level to the voxels located on the soloist in front of the band, an intermediate interest value to voxels located on others musicians and low interest to voxels located on the background buildings. We devised our 3D interest maps with this objective in mind.

Our 3D interest maps are innovative for several reasons. First, we infer 3D interest

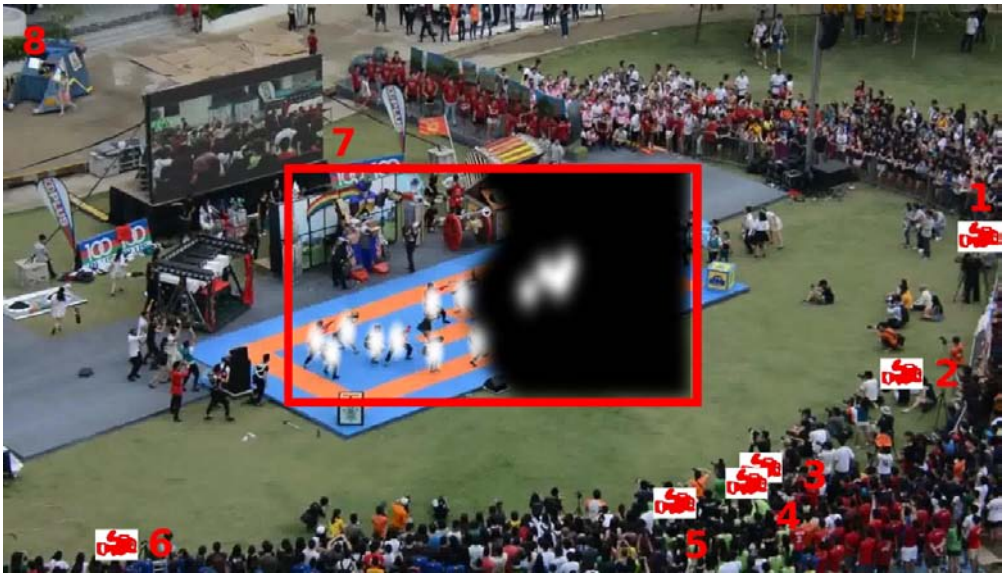


Figure 3.16: A dance performance from RagAndFlag dataset. Cameras 1 to 6 are located around the stage at ground level. Camera 7 shoots this picture (zoom position 8) and occasionally zoom into the red rectangle (zoom position 7). A 3D interest map is drawn in white in the 3D space and also in the 2D image space after re-projection of the 3D map (with black background).

Figure 3.17: Central focus assumption on the **BrassBand** dataset.

information from the multiple view geometry of simultaneous video recordings: it is an original 3D generalization of 2D saliency maps. Indeed, a 3D interest map is richer than several saliency maps computed from the original video recordings: as shown later, the correspondences between salient areas seen in range views are naturally established when re-projecting the components of the 3D interest maps. Another important difference with traditional saliency models is that we do not need to predict where the spectators would gaze at – we can simply observe it. We can see a user's camera as a third eye that he focuses towards the parts of the scene that he is most interested in. Our 3D interest levels are somehow "crowdsourced" from many videographers and we assume that the semantically important portions of the scene can be statistically revealed by analyzing the selected view frustums (originated by zoom, pan and framing decisions). If many videographers focus

towards "the most interesting" musician (e.g., the soloist) shown in Figure 3.17, a bit of semantically motivated interest is revealed.

A formal approach to 3D interest maps. We now formally introduce the notion of 3D interest map. We define a 3D interest map with respect to a given time instant, and hence we consider a set of J images corresponding to synchronized video frames[3] taken from J cameras capturing the same scene from different angles at this instant. We assume that the camera projection matrices $\mathtt{P}^j$, $j = 1..J$, with respect to some Euclidean projective representation, are given.

We also assume that we have the 2D regions of interest (ROI) in the J range views at our disposal. Note that the simplest user's ROI in a view can be drawn from the central fixation assumption [Tatler 2007], in which it corresponds to an elliptic region centered at the principal point (*i.e.*, at the point where the optical axis meets the image plane). Such elliptic regions are shown in blue in Figure 3.17 for two cameras. Even if this assumption will be relaxed later, it makes sense as it seems natural that the user tends to shot by steering the camera optical axis towards the scene hot spots.

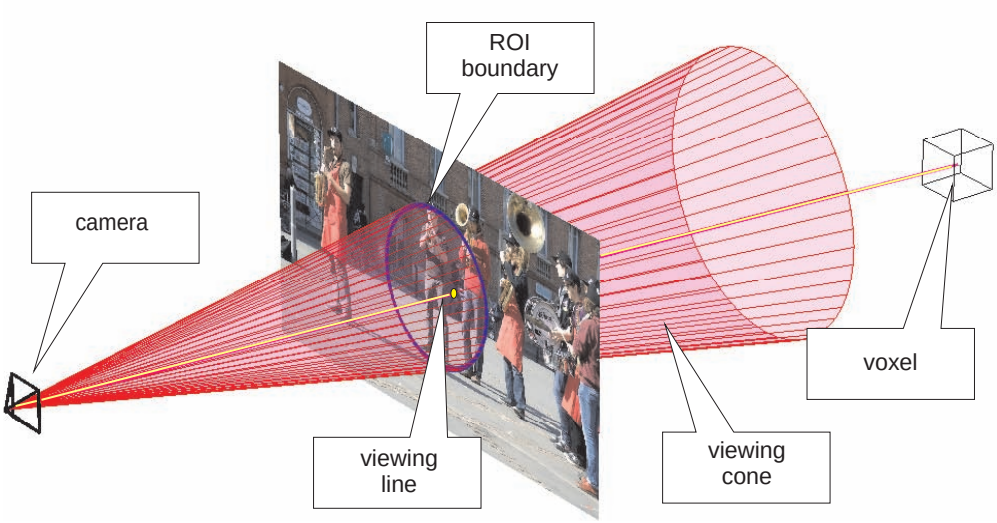


Figure 3.18: Viewing cone back-projecting a 2D ROI (in blue) and viewing line back-projecting a 2D point (in yellow); see text.

The key ingredient for defining a 3D interest map is the notion of viewing cones. A *viewing cone* is the back-projection of a 2D ROI, *i.e.*, the generalized cone in 3D space that is the assemblage of 3D lines passing through the camera center and every pixel on the 2D ROI boundary (see figure 3.18). For the k-th elliptic ROI ($k = 1..K$) in view j, referred to as E_k^j, the corresponding viewing cone can be defined in the projective 3D space by its

[3]Both the synchronization and the calibration are supposed to be, at least approximately, available for the J cameras.

order-4 symmetric homogeneous matrix [Hartley 2003, p199]

$$\Lambda(E_k^j) = (\mathtt{P}^j)^T \mathtt{E}_{jk} \mathtt{P}^j \tag{3.3}$$

where $\mathtt{E}_k^j$ is the order-3 symmetric homogeneous matrix of the ellipse [Hartley 2003, p30] describing the boundary of E_k^j, and $\mathtt{P}^j$ is the projection matrix of camera j. In the situation illustrated by Figure 3.17, we have one elliptic ROI centered in each view with $J = K$. In the more general situation, multiple ROIs in a view are allowed (as explained in the next section).

We now partition the 3D scene into a set of voxels $\mathcal{S} = \{v_i\}_{i \in I}$. The intersection of viewing cones in 3D (as suggested by Figure 3.20) reveals a region where the voxels are seen by many cameras and can be considered as interesting for that reason. Hence, we can introduce a *measure of interest* of a voxel v_i by looking at how often it is seen. If we let $\mathcal{E}$ be the set of all 2D ROIs and $\mathcal{E}(v_i) \subset \mathcal{E}$ be a subset that depends on v_i, this 3D interest level is related to the 2D ROIs through the following definition:

$$Int(v_i) = \frac{1}{|\mathcal{E}|} \sum_{E \in \mathcal{E}(v_i)} \frac{\mathrm{Vol}(\Lambda(E) \cap v_i)}{\mathrm{Vol}(v_i)} \tag{3.4}$$

where $\mathrm{Vol}(a)$ stands for the 3D volume of the considered 3D regions in argument a. As the quantity $\mathrm{Vol}(\Lambda(E) \cap v_i)$ computes the volume of intersection of v_i with the viewing cone through E, the measure (3.4) is maximum when $\mathcal{E}(v_i) = \mathcal{E}$ and v_i is *entirely* include in all ROIs of $\mathcal{E}$, giving $Int(v_i) = 1$. It is minimum when v_i is not included in any ROI, giving $Int(v_i) = 0$. The intermediate values measure how often a given voxel is partially or completely seen.

We will say that the measure of interest (3.4) is (only) *geometrically consistent* when $\mathcal{E}(v_i) = \mathcal{E}$ and is (both geometrically and) *photometrically consistent* when $\mathcal{E}(v_i) \subset \mathcal{E}$ is the subset in which v_i is (at least partially) visible. In our context, a voxel v_i is said to be *visible with respect to a set of ROIs* $\mathcal{E}' \subset \mathcal{E}$ if there are pixels in each ROI of $\mathcal{E}'$ such that (i) all their back-projections as 3D *viewing lines* cut v_i (see figure 3.18) and (ii) the neighborhoods of all these pixels are photometrically consistent in the views. It is straight forward to compute a geometrically consistent measure of interest, by intersecting all the cones associated with $\mathcal{E}$; it is the information brought by the users and no sophisticated computer vision algorithm is required. Nevertheless, due to many inter-voxels occlusions, only a subset of viewing cones associated with $\mathcal{E}$ are really seeing a visible voxel, which motivates the notion of photometrically consistent measure of interest. In Figure 3.21, the red voxel that is located at the intersection of the frustums of the three cameras is only visible (i.e., not occluded) in one frustum. Even if seen by all cameras, this red voxel is clearly not the most interesting since in fact only one user is seeing it. The method to determine $\mathcal{E}(v_i)$, the subset of 2D ROIs in which v_i is visible, will be tackled in the next section.

At this step, it is natural to consider the distribution of interest among the voxels through an histogram and subsequently introduce a normalized interest measure derived from (3.4) as

$$\widetilde{Int}(v_i) = Int(v_i)/\sum_{i \in I} Int(v_i). \tag{3.5}$$

Définition 1 *We call a **3D interest map** the limit form of the 3D histogram with voxels as bins (diagonal length $\Delta l \to 0$) with respect to the normalized measure of interest (3.5).*

In other words, if the voxels decrease in size and become infinitesimally small, the normalized measure of interest (3.5) behaves like a continuous probability density function.

Finally, we will model our 3D interest map (in our implementation and due to its generality) as a 3D Gaussian Mixture Model (3D GMM) with a satisfying number G of mixed components:

$$\widetilde{Int}(v) = \sum_{g=1}^{G} w_g \mathcal{N}(v; \boldsymbol{\mu}_g, \Sigma_g) \tag{3.6}$$

where v is now a 3D point (seen as an infinitesimally small voxel), $\mathcal{N}$ is the 3D Gaussian density, and w_g, $\boldsymbol{\mu}_g$, Σ_g are the weights and parameters of the 3D mixture. We detail the estimation of these parameters in the next section.

A GMM is flexible enough to approximate closely any multi-modal density of 3D interest. Using GMM, the estimated 3D interest map can also be re-projected in the image spaces to play the role of J dependent 2D interest maps associated with the J cameras (see also the right part of the red rectangle in Figure 3.16).

Figure 3.19: 2D elliptic ROIs, as computed in Step (1) of our algorithm, on the **RagAndFlag** dataset.

3.3.2 3D Interest Map Computation

In this section, we present a method to build a 3D interest map using several synchronized videos shooting the same scene. The proposed algorithm (see Algorithm 3.1) is broken down in two successive estimations: a Gaussian mixture is first computed to produce a geometrically consistent 3D interest map (Steps (1)-(3)). The intuition behind this first

step is to triangulate 3D interesting regions (Figure 3.20) from automatically detected 2D regions of interest (Figure 3.18). At this step, the 3D interest map (i.e. the Gaussian mixture) is coarse. The map is then refined into a final photometrically consistent one (Steps (4)-(6)). These last steps are made possible thanks to the constraints brought the coarse intermediate 3D interest map.

Algorithm 3.1: Computation of a GMM-based 3D interest map.
(1) Compute K_j ellipses of interest E_k^j for each view j
(2) Compute the intersection of visual cones
(3) Estimate a geometrically consistent 3D GMM
(4) Re-project it to get 2D GMM in image spaces
(5) Compute 2D Masks serving as PMVS input
(6) Estimate the final photometrically consistent 3D GMM from PMVS output

Detection of Ellipses (1). In this step, we look for 2D regions that are likely to contain interesting objects. These regions will then be used, as an implementation of Equation 3.4, to produce the cones as explained in the previous section. Many algorithms exist for such a task, from saliency maps to object detectors (face or body detectors would fit our use case).

At this stage, we do not need a high precision on these ellipses of interest, since they are primarily a way of reducing the complexity of the following steps. We therefore choose to use off-the-shelf OpenCV blob tracker, which uses standard algorithms for foreground detection (we use a temporal window of 15 frames) followed by blobs detection and tracking, considering the connected components of the foreground mask. In other words, we rely mostly on the apparent motion to detect the ellipses of interest, which is coherent with our use case of live performances. An example of the output of this algorithm can be seen on Figure 3.19. Note that the detected ellipses are not matched between views.

It is possible that the apparent motion is too low to detect ellipses based on the previously explained algorithm. In that case, we build on our assumption that users naturally tend to focus their camera towards objects of interest. We therefore consider an ellipse of area A_c centered on the camera frame, as shown on Figure 3.17. If the cumulated area of the ellipses of interest detected during Step (1) of our algorithm is less than $A_c/2$, then we empirically consider that there is insufficient apparent motion on the scene and switch to the central focus assumption.

Intersection of Visual Cones (2). At this stage, only a geometrically consistent 3D interest map can be computed. Indeed, we have no information about the visibility of a detected 2D ellipse number k in view j ($\mathtt{E}_k^j$ in Section 3.3.1's formalism) in other views $j' \neq j$. We use Equation (3.4) with $\mathcal{E}(v_i) = \mathcal{E}$ for defining a first coarse 3D interest map. In other words, the basic idea is to intersect cone-pairs $\Lambda(\mathtt{E}_{k_1}^j)$ and $\Lambda(\mathtt{E}_{k_2}^{j'})$ for all pairs of views (j, j') and for all pairs of 2D ROIs $(\mathtt{E}_{k_1}^j, \mathtt{E}_{k_2}^{j'})$ in these views.

The algorithm for intersecting two visual cones, related to cameras j and j' and 2D ROIs $\mathtt{E}_{k_1}^j$ and $\mathtt{E}_{k_2}^{j'}$, is as follows:

1. In image j, generate M random points inside the ellipse $\mathtt{E}_{k_1}^{j}$ and back-project each of these points p as a 3D line $\mathtt{L}_{p}^{j}$, through p and the camera center.

2. Compute the two intersection points where line $\mathtt{L}_{p}^{j}$ meets the viewing cone $\mathtt{\Lambda}(\mathtt{E}_{k_2}^{j'})$ associated with image j'. This step can be achieved very efficiently when formulated with projective geometry[4].

3. If such intersection points exist, discretize the line segment, whose endpoints are these two points, into T points so that image j yields MT 3D points.

4. Repeat 1-3 by switching the roles of cameras j and j'.

Now, given a sequence of J views, we can apply the above algorithm for each of the $\frac{1}{2}J(J-1)$ distinct image-pairs and each pair of detected ellipses that can be formed in the image-pairs. Indeed, Figure 3.19 shows a pair of images for which 5 (left image) and 4 ellipses were detected, which means that, in this particular example, 20 cones intersections are computed. Note that all cones do not necessarily intersect, so 20 cones intersections is an upper bound. The result of this step is a cloud of 3D points with denser regions where the interest is locally high.

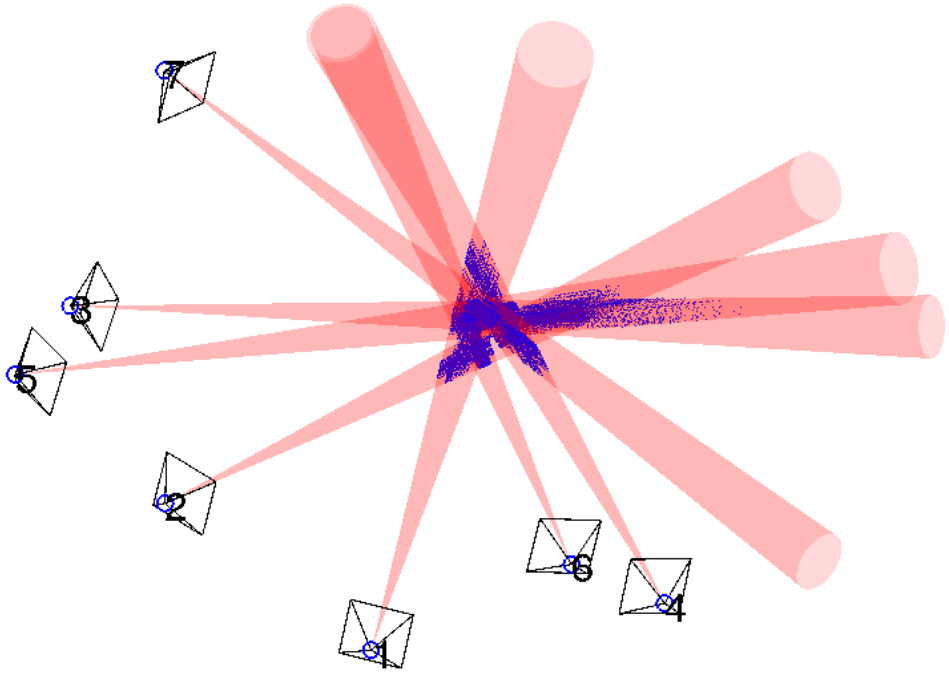


Figure 3.20: Intersection of visual cones (BrassBand dataset).

[4]Let $\mathbf{x} \in \mathbb{R}^3$ be the homogeneous vector of an image point x. The 3D line, back-projection of x, can be given by a 3D point function $\mathbf{X} : \mathbb{R} \to \mathbb{R}^4$ defined by $\mathbf{X}(\mu) = ((\mu\mathbf{x} - \mathbf{p}_4)^T \mathtt{M}^{-T}, 1)^T$, using the projection matrix decomposition $\mathtt{P} = [\mathtt{M} \mid \mathbf{p}_4]$ ([Hartley 2003, p162]). The line cuts a cone $\mathtt{\Lambda}$ at two points $\mathbf{X}(\mu_1)$ and $\mathbf{X}(\mu_2)$ where μ_1 and μ_2 are the two real roots of the quadratic polynomial $\mathbf{X}(\mu)^T \mathtt{\Lambda} \mathbf{X}(\mu)$.

Figure 3.20 illustrates this step on our BrassBand dataset, for which there is few movement on the scene. As a consequence, cones are generated based on the central focus assumption. Thus, there is one cone per camera. We can see on the figure the 3D points (in blue) that are generated in the cones intersection.

3D GMM Estimation (3). Based on the obtained 3D data points, we now estimate the first interest density (a 3D GMM) given by Equation (3.6). Since we do not know the number G of Gaussian modes and we have no initial parameter estimates, it is delicate to adopt a standard Expectation-Maximization technique. We use Mean-Shift density estimator [Comaniciu 2002] that is devised to iteratively detect the unknown distribution modes along with the data points that belong to the cluster associated with each mode. The hard assignment between data points and modes given by Mean-Shift then allows us to trivially estimate each weight w_g (ratio of points assigned to the jth cluster vs. total number of data points) and each parameter: both $\boldsymbol{\mu}_g$ and Σ_g are estimated with maximum likelihood formulas.

Mean-Shift clustering is only controlled by one scale parameter: the bandwidth. Theoretically, the bandwidth compromises between bias and variance of the estimator. In practice, it can be understood as the distance allowed between peaks of the distribution. At the current level (Step (3) in algorithm 3.1), however, the data points located at the intersection of the visual cones (see also Figure 3.20) do not fit with any recognizable shape (e.g. human performers). It is discriminative enough to select a relatively large bandwidth to detect the modes of our first 3D interest map: we typically choose $h = 2m$ in our experiments.

Re-projection to 2D GMM (4). Using a GMM in 3D allows the 3D interest maps to be re-projected in the J views. Such re-projection helps with the visualization and the evaluation of the 3D interest maps in the experimental section. See the second row of Figure 3.22 for an example. Note that the 3D multi-modal distribution also appears as a multi-modal density in 2D. Flandin and Chaumette [Flandin 2002] define a model providing, for every 3D point of the scene, the probability that it belongs to an object of interest. They introduce several propagating rules for Gaussian uncertainties, such as subspace projection and back-projection of a distribution from 2D to 3D. Their 4th rule can be reused in our case to characterize the 2D GMM in image spaces resulting from the perspective projection of our 3D interest maps. Modeling the interest with Gaussian allows us to switch back and forth between 3D and 2D.

PMVS Masks Computation (5). We now aim at making the 3D interest map more discriminative. As said before, we used Equation (3.4) with $\mathcal{E}(v_i) = \mathcal{E}$, leading to a first coarse estimate. To refine it, we must determine whether a voxel is visible or not in a view. In Figure 3.21, we already noticed (cf. Section 3.3.1) that the red voxel that is located at the intersection of the viewing cones is the most interesting according to a *geometrically consistent measure of interest* but it should not be considered as the most interesting with a *photometrically consistent measure*: its red color is not occluded in only one view. The

blue voxel is more interesting (seen from several views with photometrically consistent blue projections). In our implementation, this consistence is brought by a constrained 3D reconstruction. This can be achieved very efficiently using multi-view stereo software such as PMVS [Furukawa 2010] that is devised to propagate photometric consistency.

Given the extreme difficulty of wide-baseline multiple-view correspondences in our set-up, we use the re-projected GMM to produce 2D masks that will constrain and guide the 3D reconstruction algorithm. For each image, we binarize the 2D GMM (i.e. the 3D Interest maps re-projection) in order for the mask to contain 95% of the information from the GMM. Results of this step can be visualized on the third row in Figure 3.22. As shown in the experiments, PMVS outputs denser 3D reconstructed points in the most interesting regions (e.g. around the human performers) thanks to the 2D masks guidance. The role of the "crowdsourced" masks is quantitatively evaluated in Section 3.3.3.

Final Estimation from PMVS Output (6). The PMVS software takes as input the camera projection matrices along with the masks computed during the previous step, and outputs a set of reconstructed 3D points. These points respect the photometric consistency. Note that we need to provide a parameter to PMVS, called the level, which specifies the resolution at which the input images will be processed. We set the level to 4, which means the image resolution considered is 120×68. This choice significantly speeds-up the computation as well as limits the influence of a bad synchronization on the quality of our results. Similarly to Step (4) of the algorithm, we then estimate the parameters of a 3D GMM to model the final 3D interest map. In our public event use case, if we ideally aim at estimating a separate mode for each dancer or singer, the bandwidth for the Mean-Shift clustering should be selected as $h = 60cm$ to typically separate two humans standing next to each other. The re-projection of our refined 3D interest map is shown on the fourth row in Figure 3.22.

3.3.3 Evaluation

To evaluate our proposed 3D interest maps, we first introduce three useful datasets and briefly explain our experimental set-up. The interest maps obtained from our experiments are then compared against saliency maps in the 2D image space (a comparison in 3D is difficult to visualize).

3.3.3.1 Dataset

We present in this subsection the data we have worked on and that we use to evaluate our approach.

RagAndFlag. This dataset consists of videos filmed with seven cameras simultaneously (see Figure 3.16). This dataset presents a challenging scenario where the cameras have very diverse viewing angles and the videos depict a highly dynamic outdoor scene with variable lightning and about 50 moving objects (dancers). We use this dataset to test our mashup application.

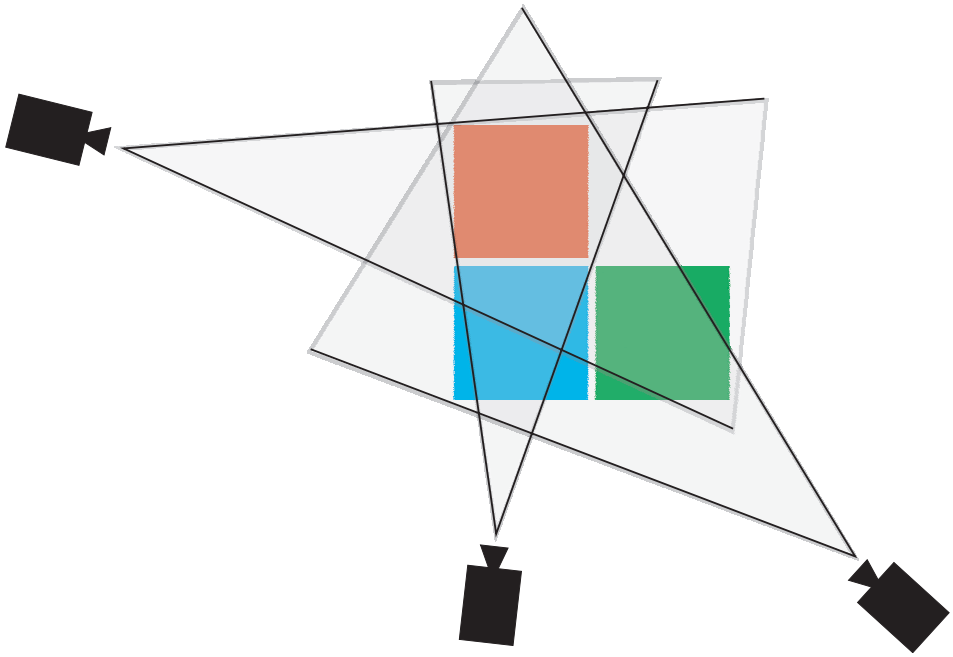


Figure 3.21: The red voxel has the highest geometrically consistent measure of interest (included in the three viewing cones) but a low photometrically consistent measure (only visible in one).

Fukuoka. This dataset consists of five video clips of a 5-minute dance show. These videos are interesting because they have been captured by fixed cameras, are of high resolution, and because the scene has features that make the camera parameters estimation easier (the floor is composed of colored concentric circles). The scene is complicated though, due to the fact that the performers are very close from each other, they are all wear the same uniform, and they move and occlude each other frequently. The five cameras in this dataset are all located in the same area, just in front of the scene which makes it a difficult scenario for our algorithm.

BrassBand. This dataset depicts a more static scene with eight musicians close to each other, moving occasionally. Seven cameras are spread around the scene: three are fixed on a tripod and four are handheld (shot with smartphones) with little movement.

3.3.3.2 3D Interest Map Accuracy

In this section, we evaluate the accuracy of the reconstructed 3D interest maps.

We start by providing the reader with the parameters we use in our experiments. We use a radius of one fifth of the focal length to produce the cones-based 3D interest map. We also set the parameters $M = T = 10$.

We calibrate the cameras for each dataset using specific patterns (orange squares in **RagAndFlag**, concentric circles in **BrassBand** and **Fukuoka**) on the planar stages. Know-

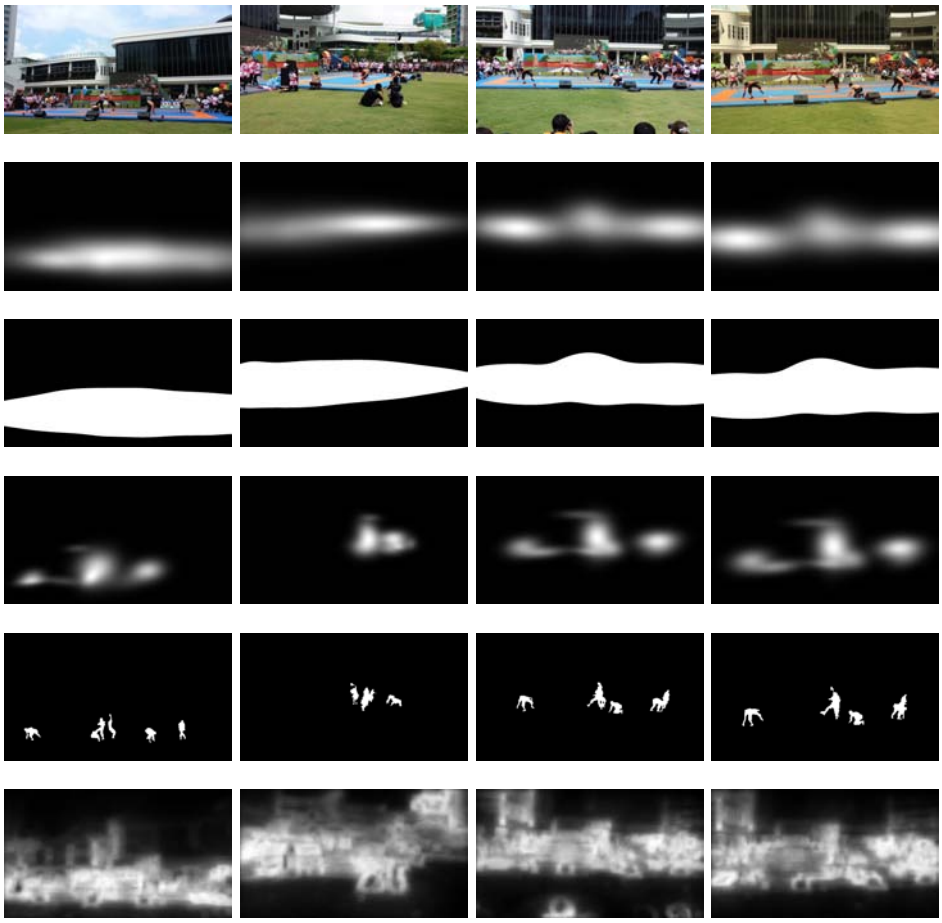


Figure 3.22: Results on the **RagAndFlag** dataset. For each row, from top to bottom: (i) original images; (ii) re-projected 3D interest maps computed by intersecting viewpoint frustums (*cones maps*); (iii) masks used as an input to PMVS; (iv) our final results; (v) manually generated ground truth binary masks; and (vi) state-of-the-art saliency map, computed using Goferman's method [Goferman 2012]

ing the plane-to-image homography associated to a given plane in the scene, we can (under standard assumptions: square pixels, and centered principal point) compute the focal length [Hartley 2003] and then the camera pose [Sturm 2000]. The average re-projection error of our calibration is computed to be 2.2 pixels on 1920×1080 images, which is acceptable for our application.

We synchronize the videos using an algorithm from Shrestha et. al [Shrestha 2010] and observe an average error 7.25 frames (0.25 sec).

Examples of our results on the **RagAndFlag** dataset can be seen in Figure 3.22. In this figure, original images are introduced in the top row; the output of Steps (1)-(3) from our algorithm (i.e. a geometrically consistent version of the 3D interest maps computed from the cones intersection) is shown in the second row. This version will be referred as *cones maps* in the following paragraphs. The third row shows the masks that are inferred

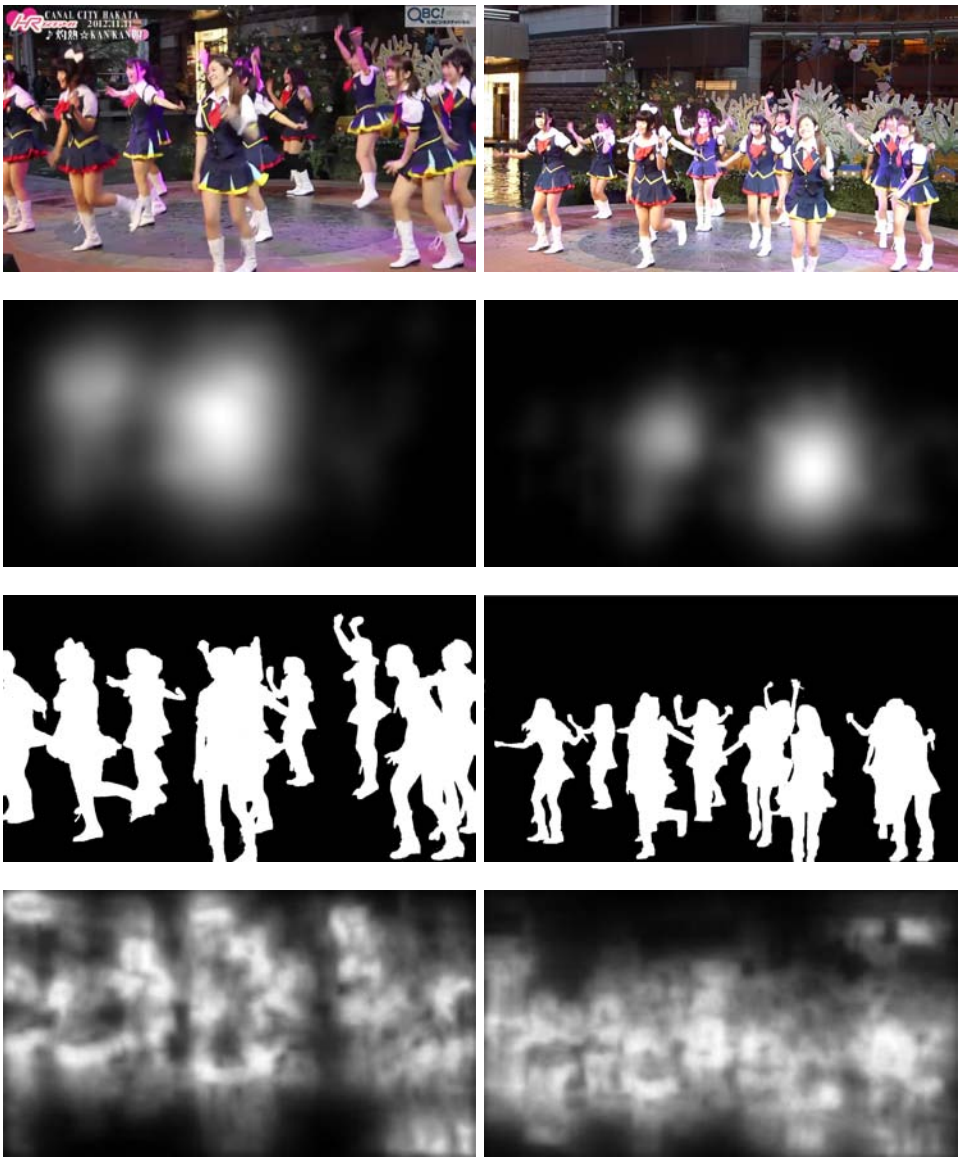


Figure 3.23: Results on the **Fukuoka** dataset: top row, original images ; second row, our 3D interest maps ; third row, state-of-the-art saliency map, as computed by Goferman ([Goferman 2012]) ; and fourth row, manually generated ground truth binary masks

from the cones maps, as described in Step (4) of our algorithm. The forth row shows the final output of our algorithm, the 3D interest maps. We manually created ground truth masks to evaluate the 3D interest maps, and these ground truth masks are exposed on the 5th row. Finally results from a state-of-the-art saliency algorithm by Goferman et. al [Goferman 2012] are shown on the 6th row.

Similar results are presented for **BrassBand** in Figure 3.24. Due to space limitations we only present original images, our final 3D interest maps, Goferman saliency, and the

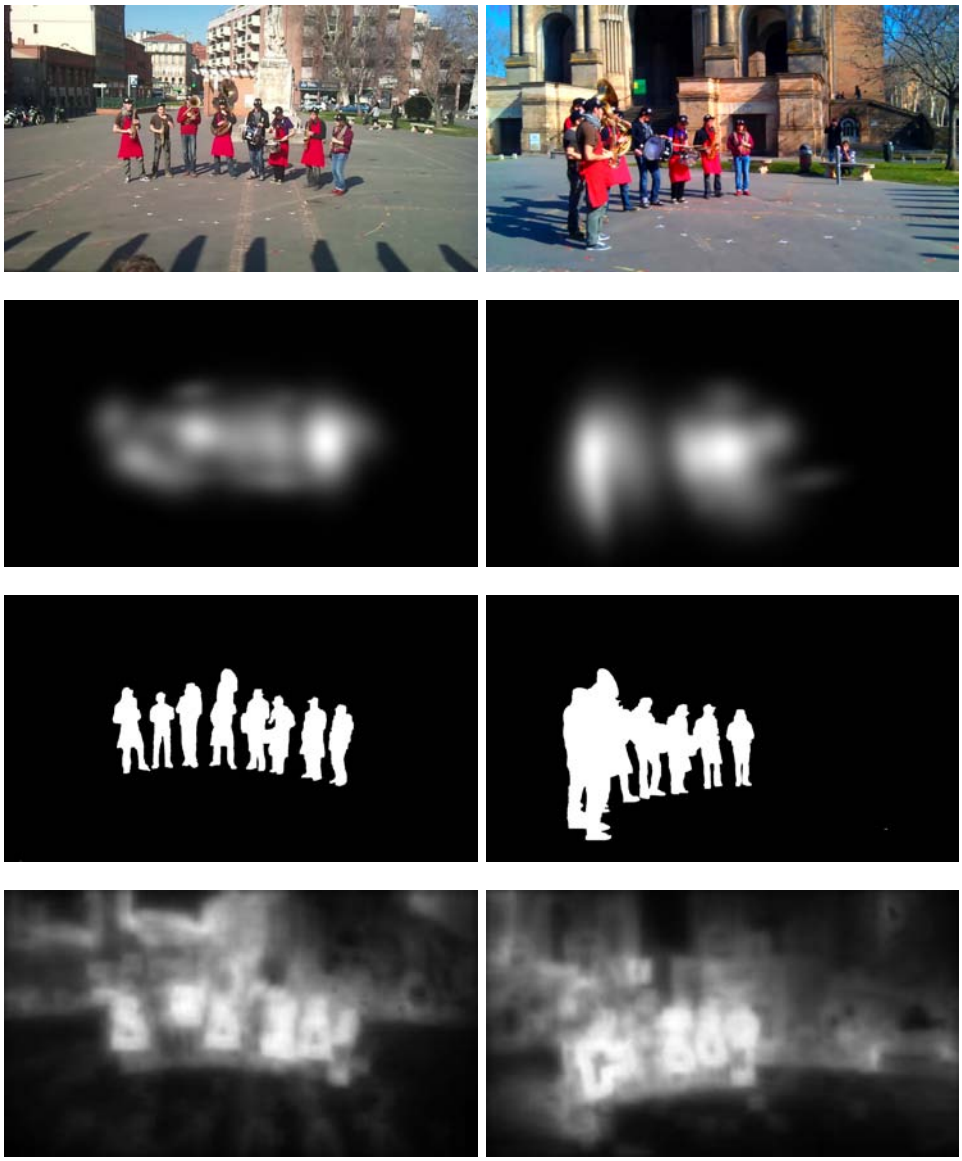


Figure 3.24: Results on the **BrassBand** dataset. For each row, from top to bottom: (i) original images; (ii) our 3D interest maps; (iii) state-of-the-art saliency map, as computed by Goferman [Goferman 2012]; and (iv) manually generated ground truth binary masks.

ground truth masks in these figures.

We can observe on all these figures that our 3D interest maps are very precise: almost all regions that are highlighted as salient are recognized as part of some objects of interest.

On the other hand, our results sometimes do not recall all objects of interest. This is especially true for the **Fukuoka** dataset, in which some of the performers are not recognized as salient by our algorithm, due to the particular configuration of cameras poses that prevents full reconstruction of the scene. Indeed the cameras are very close from each

other, which results in very similar viewpoints and in performers being occluded in all the views. The results on **BrassBand** and **RagAndFlag** are better since the cameras in these two datasets are spread all over the scene, which means that the performers are always visible in some images.

We can see also on Figure 3.22 that cones maps (2nd row) are more precise than Goferman's saliency maps (last row). This is the reason why we do not simply use state-of-the-art saliency maps and binarize it to produce the 2D masks. Though the recall from saliency maps is quite high, the precision is insufficient to efficiently serve as a mask for PMVS.

In order to numerically estimate the quality of our 3D interest maps, we compare these projections against manually created masks and also against Goferman's algorithm [Goferman 2012], which is top-ranked in the benchmark from [Borji 2012]. Before commenting on the results, please note that a salient region do not necessary imply that it is interesting or important, two attributes that require an understanding of the semantic of the content and the users' intention and are difficult to infer from the visual content alone. So the good predictions by Goferman's detector shown in the second row of the **BrassBand** results come from a lucky coincidence: the musicians are semantically important and their appearance is contrasted enough to make them salient.

In order to quantitatively evaluate our results, we follow the methodology from [Borji 2012] and estimate a precision-recall (PR) curve by varying a threshold on the interest values and generating a binary saliency map for each value of the threshold. These experiments have been conducted on four types of interest maps: our final 3D interest maps, Goferman saliency, cones maps (see above) and a version of our interest maps we call *No Masks*. *No Masks* is computed by applying the PMVS algorithm to our data without inputting the masks based on the cones intersection, and estimating 3D interest maps from the set of points obtained from PMVS. We evaluate this version to prove the importance of Steps (1) to (5) in our algorithm.

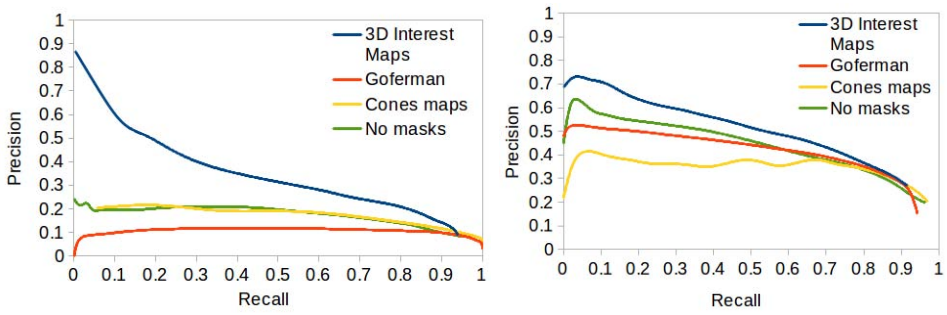


Figure 3.25: Averaged precision-recall curves on **RagAndFlag** (top) and **BrassBand** (bottom)

We created ground truth masks for 12 instants in our datasets, and resulting PR curves are displayed in Figure 3.26. Our 3D interest maps spectacularly outperforms Goferman's algorithm on the **RagAndFlag** dataset, which can be explained by the small size of the

regions of interest (see Figure 3.16: the crowd is far from the scene) and the high variety in contrast of many background elements (the mat, the buildings, etc.). Our 3D interest maps is also better than Goferman in the **BrassBand** dataset, but the saliency detection is more performant in this case. Indeed the objects of interest are very salient in the images from **BrassBand**, since most of the musicians are wearing red colors.

Results on *cones* and *no masks* are also not as good as our 3D interest maps, which validates the importance of both steps in our approach. It is interesting, however, to note that cones maps is more precise than Goferman on the curves. This validates our choice to use cones maps to compute the PMVS masks (Step (5) in our algorithm) instead of just using a saliency map, because the more precise the masks are, the less noisy the reconstructed points are and therefore the more precise our 3D interest maps are. In addition, it takes less than a second to compute cones maps whereas Goferman's saliency takes 5 seconds per image on GPU.

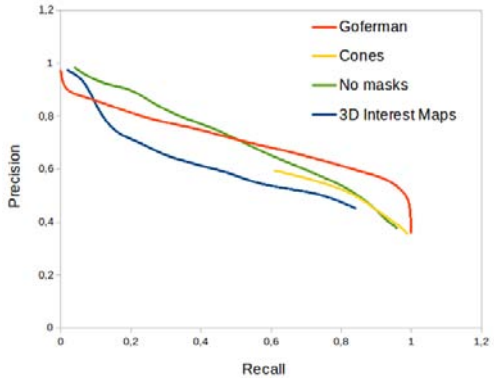


Figure 3.26: Averaged precision-recall curves on **Fukuoka**

The results on the **Fukuoka** dataset are less good. The precision of *cones* is low whereas its recall is quite high. In fact the *cones* interest maps occupies almost the entire frame. This happens because the cameras are located very close from each other, and therefore the cones intersect on a large 3D regions. This camera positioning also makes the 3D reconstruction very difficult since a lot of ROI are occluding each other: this explains the low recall for our final 3D interest maps.

3.3.4 Summary and Conclusion

In this section we have extended the concept of interest maps to the 3D space, using the same Gaussian mixture model to model the ROIs. The algorithm relies on a combination of an analysis of users' behavior (what do they film?) materialized by the videos that are produced, and of content analysis through the 3D reconstruction software that we use.

In this particular case, crowdsourcing provides a first solution which, though imprecise, allows for content analysis to perform better and to be more efficient. Content analysis is also a way to refine the user interest maps. The resulting user interest maps are comparable

to the ones produced by state-of-the-art saliency techniques.

The main challenges and limitations of our work are the strong assumptions about estimation of the camera poses and synchronization between them. In our experiments, cameras were calibrated from a single view in order to avoid to match features between views, as we benefited from the existence of geometric patterns lying on the stage ground. In many man-made environments, it might be worthwhile to calibrate the camera by automatically detecting vanishing points corresponding to orthogonal directions. Multiple tags and projected patterns (e.g. multiple tags or projections on the walls behind the stage) also have the potential for solving this problem.

This work (to appear in [Carlier 2014]) ends the chapter on crowd-based ROI detection. In the next chapter we study several applications of user interest maps that benefit from the rich semantic information contained within the maps.

Chapter 4

User Interest Maps Applications

Contents

In this chapter, we describe how some applications can benefit from the ROIs that we built using the techniques described in chapter 3. We focus on three different applications.

First we present how the problem of video retargeting can be interestingly solved using interest maps. Video retargeting refers to the problem of modifying a video so that it fits a given display size. It can involve changing the aspect ratio, or simply decreasing the resolution of the video. In section 4.1, we focus on a branch of retargeting techniques that use cropping techniques to fit the video to it new size. Our algorithm builds upon the user interest maps created in section 3.1 in order to generate an automatically zooming and panning video retargeting.

Then we show how the 3D interest maps can help build better video mashups. A video mashup is the act of taking a set of simultaneously recorded videos and generating a new video by selecting and editing together sequences from the original videos. In addition to selecting, at a given time instant, the best video to be displayed, some mashup algorithms also select a subregion of the frame to zoom into. Our 3D interest maps, created in section 3.3, can help this process in two different ways: first it indicates the Regions of Interest (ROI) to zoom into, and second it also allows to map ROIs from one video to another, allowing for smooth transitions. We describe our method in section 4.2.

Finally we introduce in section 4.3 an experimental application of inter-video navigation via ROI querying. A user of this interface is given the ability to select an ROI in the video they are watching, and the system finds and displays another synchroneous video that best shows the selected ROI. This application relies on the 3D interest maps built in section 3.3.

4.1 Video Retargeting

In this section we explain our algorithm for retargeting a video based on user interest maps computed from a zoomable video player browsing traces (see section 3.1). We start with a review of the related work in video retargeting and cinematographic techniques for automatic video generation, then explain our approach and how we evaluate our results.

4.1.1 Related work

To automatically retarget a video, Liu et al. [Liu 2006] analyzes a candidate set of cropping windows and chooses an optimal cropping window that minimizes a distortion function to crop the video before it is scaled. The cropping window can change with motion in the video and is therefore adaptive. In the context of ROI detection and tracking for stored video playback, there have been attempts to model the ROI [Fan 2003] based on the amount of motion. Such models could help determine the ROI, without human interaction with a display device. Multi-scale cropping [El-Alfy 2007] dealt with automated tracking of multiple ROIs defined by motion change. The aim was to minimize the number of ROI trajectories within the video while covering all the ROIs.

To retarget general movies, careful handling is required for camera motion: the initial cinematography must be preserved and motion artifacts must be avoided [Liu 2006]. With our videos captured from a fixed HD camera, aesthetic reframing and transitions are however required. Cinematography is an art and only informal rules are described to film various scenes [Arijon 1991]. These informal rules rather led to more heuristic than formal models in computer science. For instance, researchers in virtual reality and game design employed cinematographic rules for real-time positioning of the virtual camera [He 1996]. Automating the film-making process for computer animation with a virtual cinematography system [Li 2005] is also a challenge. Declarative language like Declarative Camera Control Language have been devised for that purpose. Formal attempts were made by MPEG7 standard (ISO/IEC 15938-5:2003) to specify tools for describing video editing segment, shots and different types of transition. In a computer vision context, Doubek et al. [Doubek 2004] also explore the use of some basic cinematographic rules for selecting the best view available in a camera network. They then crop the selected views with a virtual zoom and interpolate novel views to finally produce one attractive video stream from many coming simultaneously from the network. Automatic video retargeting can benefit from cinematography. Gleicher et al. [Gleicher 2008] propose to improve apparent camera movements in order to better follow cinematographic conventions. For example, they use virtual pans to better show moving objects.

4.1.2 Generating and Editing Shots

Thanks to our GMM-based models, ROI can be detected for each frame. The next question is how to deal with the temporal dimension. In this section, we first model the ROI

dynamics. Then we produce shots highlighting the most preferred areas. Finally, these shots will be enhanced (section 4.1.3) using reframing techniques in the last stage of our approach.

Graph-based dynamics. Regions of interest are different from frame to frame. The viewers usually follow moving attentional objects or focus on particular areas due to their semantic content. Let us consider the magic trick video (figure 3.5): at a given moment the viewer is orally encouraged to pay attention to a dice or to a card in order to understand the trick. We observed that most of the viewers actually follow such instructions and zoom into the expected visual area. Such a scenario naturally leads to time-varying ROIs both in position and shape. Additionally, split, merge and delete occur within our set of gaussian components as a natural consequence of ROI splitting, merging and disappearing.

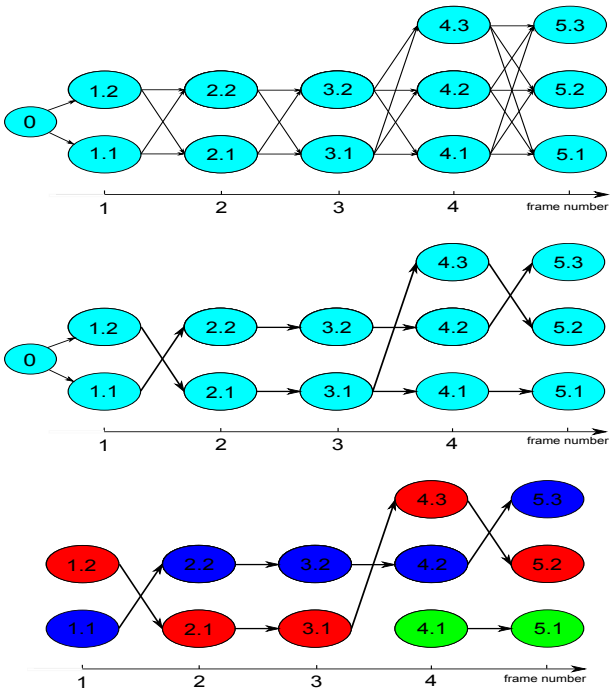


Figure 4.1: ROI dynamics graphs: full graph (top), MST graph (middle) and MST after cuts (bottom)

In order to model all these variations, we use a graph-based approach. Figure 4.1 (top row) shows the beginning of a video sequence where the j^{th} detected ROI at frame t is a graph node denoted $\langle t, j \rangle$. Only five frames are shown and there are only two ROIs by frame except for the 4^{th} and a 5^{th} where a third one appears. Since we will later need a tree structure for the graph, we introduce a virtual frame 0. More precisely, our set of ROIs is represented as a weighted directed graph where an edge is formed between every pair of ROI in subsequent frames (eg. $[\langle t, j \rangle, \langle t+1, i \rangle]$). The edge set is noted E. The weight w_d of an edge between two ROIs is initially set as the euclidean distance between their modes:

$w_d(\langle t,j\rangle, \langle t+1,i\rangle)$ is the distance between the ROI $\langle t,j\rangle$ (located on mode $\mu_{t,j}$) and the ROI $\langle t+1,i\rangle$ (located on $\mu_{t+1,i}$):

$$w_d(\langle t,j\rangle, \langle t+1,i\rangle) = ||\mu_{t,j} - \mu_{t+1,i}||. \tag{4.1}$$

Moreover, we add a few more attributes to label the ROI nodes and to assess the ROI variations. Each ROI node $\langle t,j\rangle$ can be labeled with the *associated weight* $\omega_{t,j}$. The *weight variation* between two subsequent ROIs can also be measured by :

$$w_\omega(\langle t,j\rangle, \langle t+1,i\rangle) = |\omega_{t,j} - \omega_{t+1,i}|. \tag{4.2}$$

Similarly, each node $\langle t,j\rangle$ can be associated with the *area* associated with the covariance matrix $\pi\sqrt{\det(\Sigma_{t,j})}$. The *area variation* between two subsequent ROIs can be evaluated by :

$$w_a(\langle t,j\rangle, \langle t+1,i\rangle) = \pi.|\sqrt{\det(\Sigma_{t,j})} - \sqrt{\det(\Sigma_{t+1,i})}|. \tag{4.3}$$

Shot segmentation. In order to group related and subsequent ROIs into shots[1], we seek a partition of the set of nodes where some consistency measure in a subset is high. Our approach consists in two steps :

- compute a Minimum Spanning Tree according to w_d weights,
- cut some edges to produce consistent candidate shots.

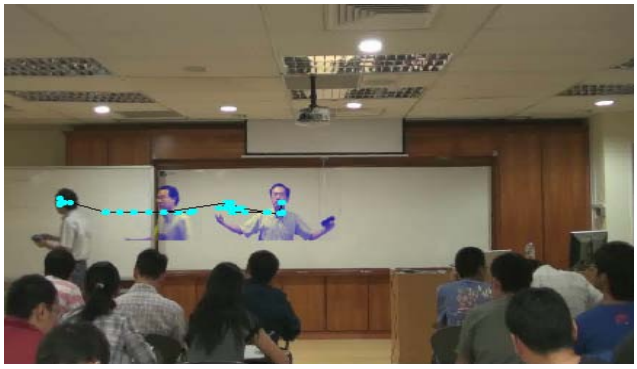


Figure 4.2: Minimum Spanning Tree (MST)

A Minimum Spanning Tree (MST) is an acyclic subset T of edges selected from the edge set E of the initial graph. For now, let us consider that our graph is undirected. The MST edges T connect all the ROI nodes such that their total weight is minimum :

$$w_d(T) = \sum_{[\langle t,j\rangle, \langle t+1,i\rangle] \in T} w_d(\langle t,j\rangle, \langle t+1,i\rangle). \tag{4.4}$$

[1] A shot is a single stream of images, uninterrupted by editing.

The second row of figure 4.1 presents an MST of the full graph. Removed edges are those with the highest weight values. To get it, we apply a simplified version of Prim's algorithm that takes the particular structure of our graph into account (no edge between ROIs at the same frame). Figure 4.2 also presents an MST on our lecture video. The figure suggests the motion of the speaker walking leftwards in the scene by using ghost effects. All the nodes of the graph of ROIs are plotted in cyan. The MST edges are displayed in black (same color coding as the one used in figure 4.1). Long edges can be observed on figure 4.2 when the speaker moves quickly leftwards. Long edges are characterized by an important w_d weight.

The middle graph of figure 4.1 (MST) can be interpreted as follows. One moving attentional object is present in the frames 1-5. Our ROIs (MST nodes) are able to track it along the tree. For example, the object is tracked by ROIs $\langle 1.2 \rangle$, $\langle 2.1 \rangle$, $\langle 3.1 \rangle$, $\langle 4.3 \rangle$ and $\langle 5.2 \rangle$ from which we deduce a single stream of cropped images (focused on the object): a tracking *shot*. In order to produce it automatically, the system has to decide to cut the edge between node $\langle 3.1 \rangle \rightarrow \langle 4.1 \rangle$. In other words, producing shots can be viewed as splitting the MST tree.

A tree can be cut into two disjoint set by simply removing an edge. The third row of figure 4.1 shows a MST after the cuts. Once again, removed edges are those with the highest weight. Then we form three shots (in red, blue and green). Similarily, figure 4.3 presents the MST of figure 4.2 after the cuts. In our application we cut the edges with large w_d weights. The threshold we use is also the bandwidth parameter of Mean-Shift clustering technique that aims at distinguishing separate ROIs:

$$\text{Cut}[\langle t, j \rangle, \langle t+1, i \rangle] \text{ if } w_d(\langle t, j \rangle, \langle t+1, i \rangle) > \xi_d \tag{4.5}$$

Figure 4.3 presents the 5 shots we obtained from our lecture video. Shot 1, 3 and 4 present the teacher speaking in the middle of the classroom in the first part of the sequence. Shot 2 is focused on the whiteboard. Shot 5 tracks the speaker motion. Figure 4.4 shows the shots on the timeline.

As a refinement, for each shot, we also identify a few candidate positions for possible new cuts to be used in the next step. For instance, shot 1 can be further refined at frame 37 and 51. These possible new cuts are identified by high values of $w_a(\langle t, j \rangle, \langle t+1, i \rangle)$ or $w_\omega(\langle t, j \rangle, \langle t+1, i \rangle)$.

Compositing Shots. The final result we obtain from the graph is a sequence of shots to be edited. For each temporal interval between two possible cuts and among possible shots, we select the best one given a selection criterion. In figure 4.4, the considered intervals are $[1, 6], [6, 37]$ etc. Each interval may be covered by several shots. The simplest selection criterion may be the average weight of the shot divided by its duration. In other words, this strategy selects the most popular shots. Other more complex strategies may be imagined. For example, the selection may favor shots with higher motion.

Figure 4.3: Shots after the MST cuts

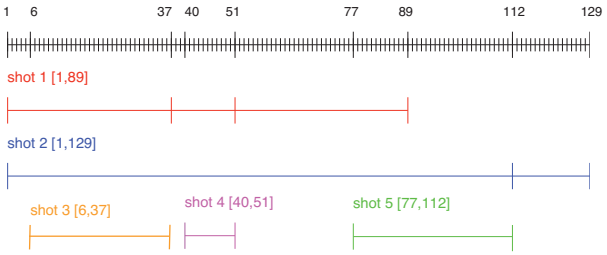


Figure 4.4: Timeline [1,129] and possible shots

4.1.3 Reframing Techniques

We now have a sequence of shots that represents a reframed version of our input video. The quality of this video is suboptimal. User studies (section 4.1.4) show that this automatically produced version is not pleasant to watch since it is too shaky. This is mainly due to annoying visual jumps coming from translational and scale noise.

Therefore we use reframing techniques aiming at (1) stabilizing shots to correct for this noise (lack of aesthetics) and (2) reestablishing shots to establish the scene context before moving to close shots (loss of important context information).

Simple Guidelines. Before detailing our reframing techniques, let us introduce some good editing practices that we followed. We start by systematically keeping the initial aspect ratio (16/9) for the generated framing. Shots may be produced at different scales. An *extreme long shot* (ELS) is a framing close to the initial full HD format (ranging from 1920×1080 to 1600×900). The *long shot* (LS) is a smaller framing (width from 1600 to 1100) useful to make a rather stable shot that can easily cover movement without reframing. The *medium shot* (MS) is the most popular and its width ranges from 1100 to 700. Finally, *close-up* (CU: width from 700 to 400) and *extreme close-up* (ECU: width < 400) are smaller framings for showing details. Regarding the duration of the shots, we do not impose strong constraints since the decision to extend a shot can be as effective as the decision to cut it. We simply ensure that handled shots include at least 10 frames. Generally speaking we try

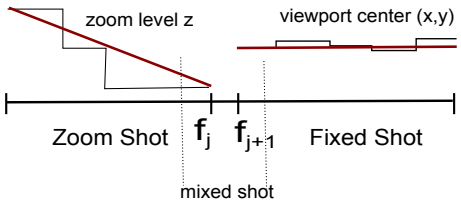


Figure 4.5: Reframing on two shots

to produce a retargeted video that maintains a clear and continuous action. Therefore we follow the general style of *continuity editing* [Arijon 1991].

Bottom-up Reframing. At the *bottom* level (individual shot-level) we stabilize each shot to ensure its spatial continuity. This stabilization is a two-step process. First we classify the shot into *fixed*, *zooming* or *dolly* types of shots. In a fixed shot, the focus does not change. A zooming shot results only from an optical zoom into an object. In a dolly shot, the camera is moving as if it was mounted on a train track, parallel to the scene (pan-like motion). Some *mixed* shots exhibit combinations of these types.

In the second step, each shot is stabilized according to its type. For example, in the zoom shot of figure 4.5, the zoom level of each frame of a zoom shot is interpolated using the values of the first and last frame. For a detected fixed shot, since there are small variations of the focus, we modify the center of each frame using the average center.

Once we stabilize each shot individually, we work at the *upper* level (inter-shot level) in order to smooth out the transitions between shots. In our example, if there is a discontinuity of the viewport center of frames f_j and f_{j+1}, we create an additional short *mixed* shot that interpolates both the zoom and position of the center.

Additionally, also at the upper level, we apply another technique called *reestablishing shot* (RS). A RS is often a medium or medium-long shot. It usually follows a close-up and is used to help the viewer better understand the context. In other words, it reminds the audience the position of the viewport inside the scene. The figure 3.3 (third row) shows a RS. The first shot shows a close-up on the speaker and the third is a close-up on a very different ROI (the left board). Our system automatically produced a RS in between, upon the detection of a ROI (the left hotspot of the third shot, representing the left board) that was not visible during the first shot (left hotspot). Therefore the user is presented with a *medium* RS before the second close-up.

4.1.4 Results

To evaluate our proposed crowd-sourced video retargeting, we posted several videos for users to watch through our zoomable video player (see figure 3.2). The videos are recorded in high definition using a fixed camera. The video sequences posted has a range of different content, ranging from lecture, sports, and magic tricks. Within a period of two weeks, we collected traces from 53 user sessions (one session for each viewed video). A total of 11183

interaction events are logged.

Retargeted versions of the video are created using the algorithms described in the previous sections. In this section, we present results for three video sequences. The first two are magic tricks videos, which we refer to as the *card trick* and the *dice trick* video respectively. The third video sequence is a video of a rhythmic gymnastics movement (introduced by figure 3.2 and 4.7), which we refer to as the *gymnastic* video. We first describe the retargeted videos and compare our results with saliency-based methods. An analysis of the different shot types, along with their distributions is then performed. We show that the distribution of shot types in the retargeted video is consistent with those of the collected traces. Then, we carried out a user study to assess the quality of the retargeted shots (4.1.5).

Retargeted Video. We now describe two sample videos that our method produced. We qualitatively compare our results with results generated by using visual attention models. Resulting videos are available at
`http://www.youtube.com/user/AutoZap`.

Figure 4.8 shows a selected set of original (HD) video frames and their corresponding retargeted frames from the *dice trick* video. In this trick, the magician is challenged to find the correct dice number. He gives instructions to the girl: put the dice in the black box (2 first rows), show the dice number to the camera (rows 3-5), close the box and put it down on the table (row 6). He returns to the table (row 7) and continues (row 8-9). One should notice the close-up on the dice number (row 5). Another relevant close-up (row 8) emerged and focused on the box: the critical object of interest for understanding the trick. Just after, the reestablishing shot helps to put the user into context for the rest of the trick.

We first highlight some shots of interest region in our video that is hard to be reproduced by existing approaches. The close-up shot on the dice number (row 5) in the retargeted video sequence is synchronized with the instructions of the magician ("show the dice to the audience"). Doing this automatically with content-based approach would require speech recognition to understand what the magician has said and object recognition to identify the dice within the scene. Both are hard problems that are challenging to solve. This example stresses the importance of leveraging user behavior to generate retargeted shots, since users naturally follows the instruction of the magician and zoom in to see the dice.

Figure 4.6 shows an example of salient regions in the *dice trick* video, computed using Matlab SaliencyToolbox[Walther 2006], which is based on visual attention model, particularly, color, intensity, and orientation. The left shows a frame from the video with salient regions enclosed in yellow. The right shows the heatmap indicating the salient regions. This example shows that the most salient region in the video, according to the visual attention model is the back of the magician, followed by the hand, a deck of cards on the table (which is not even used in this trick), and finally the dice. This example illustrate that visual attention model is not sufficient for detecting interest regions in the video sequence.

The interest regions detected using visual attention model on the *gymnastic* video is shown in Figure 4.7 to further illustrate the efficacy of our approach. The *gymnastic* video

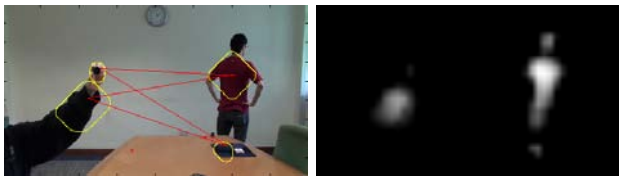


Figure 4.6: Outputs from the saliency toolbox.

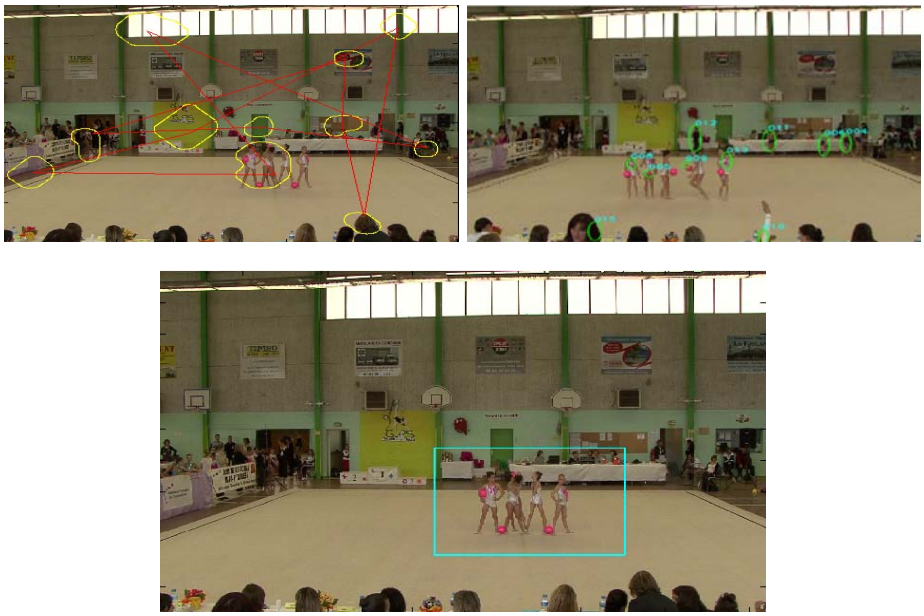


Figure 4.7: Visual Attention Models vs. our ROI

is a wide angle shot of a gymnastic contest, with five gymnasts performing on the floor, surrounded by audience, judges, and officials. The top image shows the results obtained by the SaliencyToolbox. The salient regions detected include doors, windows, notice boards, posters, and some audience. The most interesting part of this video, however, is the gymnasts. While the SaliencyToolbox managed to detect the gymnasts as salient regions, this detection is only possible after the option which gives skin colors more weight is turned on. On the other hand, our method correctly detect the gymnasts as the ROI (the middle figure). The bottom figure shows another frame from the same sequence, with moving regions highlighted, as detected using the blob tracking algorithm in OpenCV. Visual attention model that assumes motion as a salient feature would have overestimated the number of interesting regions in the frame. In this example, besides the gymnasts, motions of audience and other gymnasts standing-by are also detected.

Properties of Retargeted Shots. Having described two of our retargeted video sequences, we now present some of their properties. Table 4.1 summarizes the distribution

of shot types as captured by our logs. Shot types are classified into classes, ranging from *extreme long shot* (ELS) to *extreme close-up* (ECU). The variety of observed shot types is a clear indication that users effectively used the zoom and scroll video interface for a better viewing experience. Shot distributions are not very similar among the three video sequences, since they are strongly related to the content. The *Card Trick* exhibits numerous *close-ups* (CU) and *extreme close-up* (ECU), since the user is explicitly and orally encouraged to gaze at playing cards. These results confirm the importance of the semantic of the contents. This statement is also verified by a detailed analysis of the *gymnastic* video: users who were relatives of one of the gymnasts tended to track her by using close-ups.

shot	ELS	LS	MS	CU	ECU
Dice Trick	17%	26%	29%	19%	9%
Card Trick	13%	25%	13%	24%	25%
Gymnastic	8%	23%	26%	31%	12%

Table 4.1: Statistics on shot types

Table 4.2 compares various versions of retargeted shots for the dice video sequence. The first row ("User Traces") shows the previously discussed shot types ratios (in %). The following rows present shot distributions for various versions of our retargeted video. The second row ("Final") is our final retargeted version that includes reframing techniques (Section 4.1.3) and the MCD estimator (see Section 3.1.3). The third row ("No RT") is the retargeted version without reframing techniques produced with MCD. The fourth row ("ML") corresponds to an alternative version without reframing techniques produced with Maximum Likelihood (ML) covariance estimator instead of MCD.

The shot distributions of various versions provide some insights into how our method works. First, we see that the shot distribution for "Final" and "User Traces" are slightly different. Less ELS exist for "Final" due to the retargeting that favors LS and MS. Indeed, the retargeted version shows that for the dice video, all ELS shots can be effectively replaced by smaller shots. We will see in the next section that the user studies proved the effectiveness of this editing, since most of them are satisfied with the quality. The number of ECUs doubles. Second, using ML (commonly used in GMMs models), tend to produce bigger shots. This result justifies our preference of MCD over ML for estimating the covariance matrices. The final retargeted *dice trick* includes more CUs and ECUs since it focuses on a small object of interest (the dice), which are not found if ML is used. A remarkable result is 54% LS, $27\% = 13 + 14$ close-ups of the "No RT" version. We can understand the cause by watching the produced video: the focus changes rapidly from close-ups to the LS. This is smoothed out by the "Final' version around MS thanks to reframing techniques.

4.1.5 Evaluation

Recall that the goal of video retargeting is to produce a visually pleasant, yet informative video for a small display. To assess if our results have achieved the two goals of video

Shots from	ELS	LS	MS	CU	ECU
User Traces	17%	26%	29%	19%	9%
Final	0%	32%	38%	12%	18%
No RT	1%	54%	18%	13%	14%
ML	27%	32%	20%	19%	2%

Table 4.2: Statistics on shot types for different retargetings.

retargeting, we conducted a user study, comparing retargeted video from our method with two other video sequences produced from the same video.

Video Sequences. We choose the *dice trick* video as the clip for user study. As a ground truth, we compare our retargeted video (denoted crowdsourced) to a version of retargeted video produced by an expert user (denoted expert). This ground truth retargeted video is produced by having the expert, who is well aware of the video content and user interface, to watch the video using our zoomable video player. His choices of viewports are used as input to produce the ground truth. On the opposite, we use either one of unretargeted video (i.e., always in zoom level 0, denoted nozoom), a retargeted video generated from a chosen user trace (user), and a retargeted video produced after Section 4.1.2 without reframing technique (noRT). The chosen user trace is a typical trace (zooming and panning to regions similar to our retargeted video), except for a short time when the user is not able to find the dice and has to zoom out and in again to refocus on the dice in the video.

Methodology. We show three videos in random order to each participant. The expert and crowdsourced versions are always shown, whereas the third video is randomly chosen from one of the remaining: nozoom, user or noRT. Users are asked to watch the video clips and to rate the editing quality of the video. A user may watch the video as many times as he wants. We assess the quality of the retargeted video by asking our participants the following questions: (i) Is the editing quality of each video reasonable? (ii) Does the video manage to convey the important information to understand the events in the video? The participants are asked to rate each video in a rating of 1 (poorest) to 5 (best), as well as to make qualitative comments on the video.

Participants. The user study involves 48 participants (28 Male, 20 Female). 16% of the participants have a prior experience in video editing or cinematography. 18 participants are presented noRT and user versions, 12 are presented nozoom.

Results. Respectively 87% and 79% of the participants find the editing quality of expert and crowdsourced reasonable. Only 21% of users find the third video reasonable. Among the possible third video, nozoom is found to be most reasonable (41%), while noRT is the least reasonable (5%). 22% of users find the user video reasonable.

The expert and crowdsourced versions of the retargeted video have an average rating of 3.6 each. The rating for the third videos averaged to be only 2.35 (2.11 for noRT, 2.33 for user, 2.75 for nozoom). There is no significant differences between the ratings made by users with experience in video editing and cinematography. These users rated expert 3.5, crowdsourced 3.3, and the third sets of video 2 on average.

Figure 4.8: Retargeted video: selected frames (top) and retargeted frames (bottom)

When asked if the video managed to convey useful information, expert scores the highest with affirmative answers from 79% of users, followed closely by crowdsourced. Interestingly, 66% of the participants who watched user thinks that it still manages to convey useful information. Surprisingly, none of the 12 users who watch nozoom found it useful. The last two results validate the importance of retargeting, and validates the usefulness of user traces in choosing important regions to watch, even when it is done by only one single user. Only 38% of users found that noRT managed to be useful.

The results above shows that our crowdsourced version is only slightly worse than the video retargeted manually by an expert user.

4.1.6 Summary and Conclusion

In this section, we have proposed an automated approach to retarget a high-resolution video for display on small screens by exploiting user interest maps. This approach relies on: (i) historical user traces to select regions that effectively convey the message of the video; and (ii) cinematography rules to improve the visual quality of the retargeted video. Our usage-based approach is a natural complement to previous retargeting methods based on content analysis. A user study with 48 participants showed that the retargeted video produced manually by an expert is only slightly better than those produced automatically using our approach.

4.2 Video Mashup

In this section we demonstrate the richness of 3D interest maps, by using the information obtained in section 3.3 to build video mashups. We start by reviewing the related work, then explain how we generate a video mashup based on the 3D interest maps and finally evaluate our results.

4.2.1 Definition and Related Work

The goal of video mashup is to automatically create a temporally continuous video of an event by splicing together different clips from simultaneously recorded videos of the event, usually shot from a diverse set of angles. A video mashup system aims to replace a human director, and therefore a successful video mashup system should produce videos that are interesting to watch, logically coherent, and cover the important scene that happens during the event.

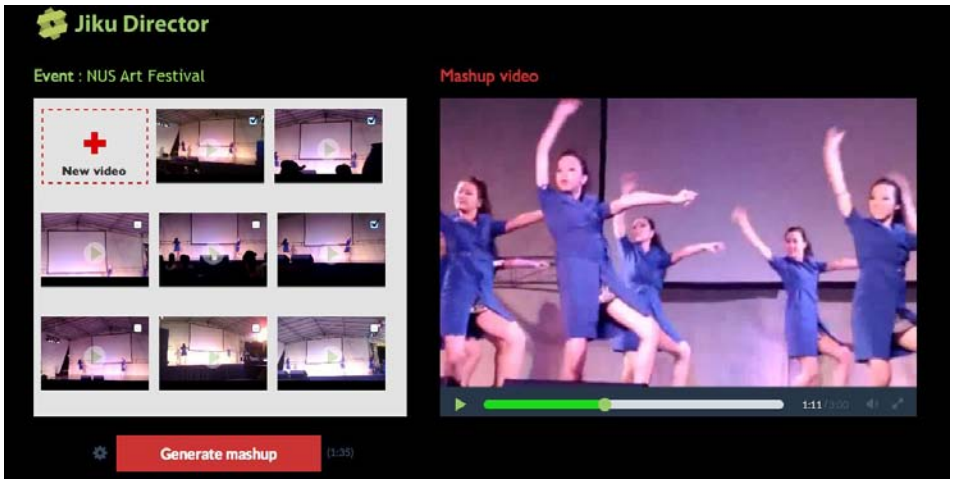


Figure 4.9: Screenshot of the Jiku Director.

One of the state-of-the-art systems for video mashup is Jiku Director [Nguyen 2013] (figure 4.9), which focuses on mashing up videos captured with handheld mobile devices. Jiku Director uses an online algorithm [Saini 2012] to produce the mashup, following a shot-change pattern learn from professional, human-edited, video. The system also filters out bad quality videos through simple feature analysis and try to increase the diversity of shorts in the mashup by not selecting clips from the same devices when possible.

Jiku Director, however, is limited in two ways. First, it always uses the video clips recorded by users "as-is" without any cropping and scaling, nor introducing any camera movement. As a result, the camera shots in the resulting video is limited to the movement made by the user while recording the video. Furthermore, since the videos are recorded by audiences of an event, the shots are most of type of long/medium shots. The resulting

mashup is lacking in diversity of shot types. Second, when Jiku Director introduces a cut, its only consideration is the shot angle (viewing direction of the camera) and shot length (distance of the camera from the stage), without considering the subjects or narrative.

One way to overcome the first limitation is to analyze salient features from each video independently and model the importance of the content. Jiku Director could then introduce virtual camera movement (zooming effects and dolly shots) by cropping out the region-of-interests (ROIs) from the videos. This approach is similar to that adopted by Carlier [Carlier 2010a]. This approach, however, could not overcome the second limitation, as there is no way to associate an object of interest that appears in one video clip with another object of interest that appears in another clips.

The proposed 3D interest map allows us to identify such association for objects that appears in the same time instance across two video clips. Furthermore, using the objects' bounding boxes and interest levels, we can add camera movements to pan from one object to another, and to zoom in for close-up shots of the objects, producing better mashups. The rest of this section describes how we make use of 3D interest map in designing a new video mashup algorithm.

4.2.2 Computing Bounding Boxes

The first step of our mashup algorithm is to compute bounding boxes that contains the most interesting regions of the videos. Since our 3D interest maps are modeled by GMM, one bounding box for each Gaussian in the mixture can be produced easily.

The mashup application, however, needs stable bounding boxes that are tracked over time. The algorithm described in Section 3.3.2, however, applies at every time instant without considering temporal consistency. Therefore, to produce bounding boxes for the mashup applications, we change the last step of our algorithm (clustering a 3D point cloud) and compute the bounding boxes in the 4D space (3D and time). This is actually equivalent to building a 4D user interest maps! We augment the 3D points with a fourth coordinate that corresponds to their frame index, and apply the Mean-Shift algorithm to this 4D point cloud. We compute a bounding box for each cluster that contains all points of the cluster for its entire duration. The bounding boxes are stable over time by construction and can be mapped in all videos. An example of this output can be seen on Figure 4.10. Note that all bounding boxes are constrained to be of the same aspect ratio as the original video.

4.2.3 Mashup Heuristics

With the set of bounding boxes, we now present a simple algorithm for video mashups to illustrate the power of 3D interest map.

Note that 3D interest map cannot be computed online, but as we will later compare our results with Jiku Director's, which is an online algorithm, we choose to design our mashup algorithm to be online as well, with no knowledge of future frames beyond the current

Figure 4.10: Bounding boxes computed from our 3D interest maps. Bounding boxes of the same color are corresponding to the same object of interest.

selected shots.

Our algorithm take, as input, the set of object of interests computed from 3D interest map. For each object appearing in each frame, we have its bounding box, x-y coordinate and depth of its centroid in each frame. The output of our algorithm will be a sequence of *shots* (t, j, B, m), where t is the beginning time of the shot, j is the camera to choose the shot from, B is the set of regions of interest to use in this shot, and m is the virtual camera movement in this shot.

We use the same decision matrix used by Jiku Director to decide Camera j in each shot. Once j is fixed, we determine t, B, and m as follows. We first define a *scene* as a sequence of shots with logical coherence. For instance, a scene can start with a wide angle shot, followed by a zoom in shot to an object x, followed by a close up shot of x. We limit the length of the scene to containing no more than N_s shots (we use $N_s = 3$).

To determine the current shot starting at time t, the algorithm first needs to decide if the current shot belongs to the same scene as previous shot, or starts a new scene. If current scene has reached its limit in number of shots, we simply start a new scene. Otherwise, we try to see if the current shot fits.

Let L_{max} be the maximum possible shot length (we use $L_{max} = 7s$). To determine the current shot, we look at the bounding boxes and objects for the next L_{max} time in the video shot by camera j.

We prioritize the bounding box of each object by their importance in the scene. Each object i has a *camera independent importance*, $I(i,t)$, which is computed from $\sum_j v_i^j(t)/d_i^j(t)$, where $v_i^j(t)$ is 1 if object i appears in Camera j at time t; 0 otherwise, and $d_i^j(t)$ is the depth of the object i from the camera plane of camera j at time t, normalized to 0 and 1. The intuition here is that an object that appears frequently in the collective video clips and closer to the camera is the more important than those that appears fewer number of times/far away from cameras.

In each video clip, we compute the *camera dependent importance* of an object, $I_j(i,t)$ as $I(i,t)/d_i^j(t)$. The importance of the bounding box is then the sum of the camera dependent importance of all objects in the bounding box. Note that even though each bounding box is associated with an object, other objects may appear in the bounding box. Naturally, a

bounding box that includes multiple important objects becomes important.

Now, we look at the most important bounding boxes in j at each time instance for the next L_{max} seconds. If at the start of the L_{max} seconds, the same bounding boxes is used as the previous shot, we fit the current shot into the same scene, and set B to these bounding boxes. We may stop the current shot at the time when these bounding boxes disappear. If a bounding box different from that of previous shot become the most important, we introduce a new scene, setting B to the new most important bounding box.

It remains to determine m for the current shot. The choice of m can be either: wide angle shot, close up, move the camera (zoom in, zoom out, or pan). The algorithm follows the following principles: (i) there should not be frequent camera movement, and (ii) the choice of m should leads to "fluid" shots: two consecutive close up shots should have the same object; zooming into an object should be followed by either a close up shot that include that object or a pan shot moving from that object to something else; a zoom out shot should be followed by a wide-angle shot; and a pan shot moving from an source object to a target object should be followed by a zoom out shot from the target, or a close up shot of the target. In figure 4.11, a video transition from our heuristics illustrates how the 3D informations can be used for continuity editing.

Figure 4.11: A video transition keeping the same object(s) of interest in focus between two successive shots from two different cameras.

4.2.4 Evaluation

We now present the methodology and results of our user study to evaluate the quality of videos produced from our mashup algorithm.

Dataset. We pick the Rag&Flag dataset as the input to our algorithm, as it has the most dynamic content. We use a 1-minute sequence from the 7-minute video, to limit the length of the user study, and let users be able to watch the videos several times.

Benchmark. We choose to compare our output with two other methods. The first is the output from the MoviMash algorithm [Saini 2012]. As MoviMash does not generate virtual camera movements, we denoted the output from this video as NoVC. We also included the output from a version of our algorithm that makes use of 2D saliency maps to generate zoom and pan shots. We use the same saliency detector than the one from step (1) in algorithm 1, by consistency. We track the obtained ROIs with the same technique than the one from

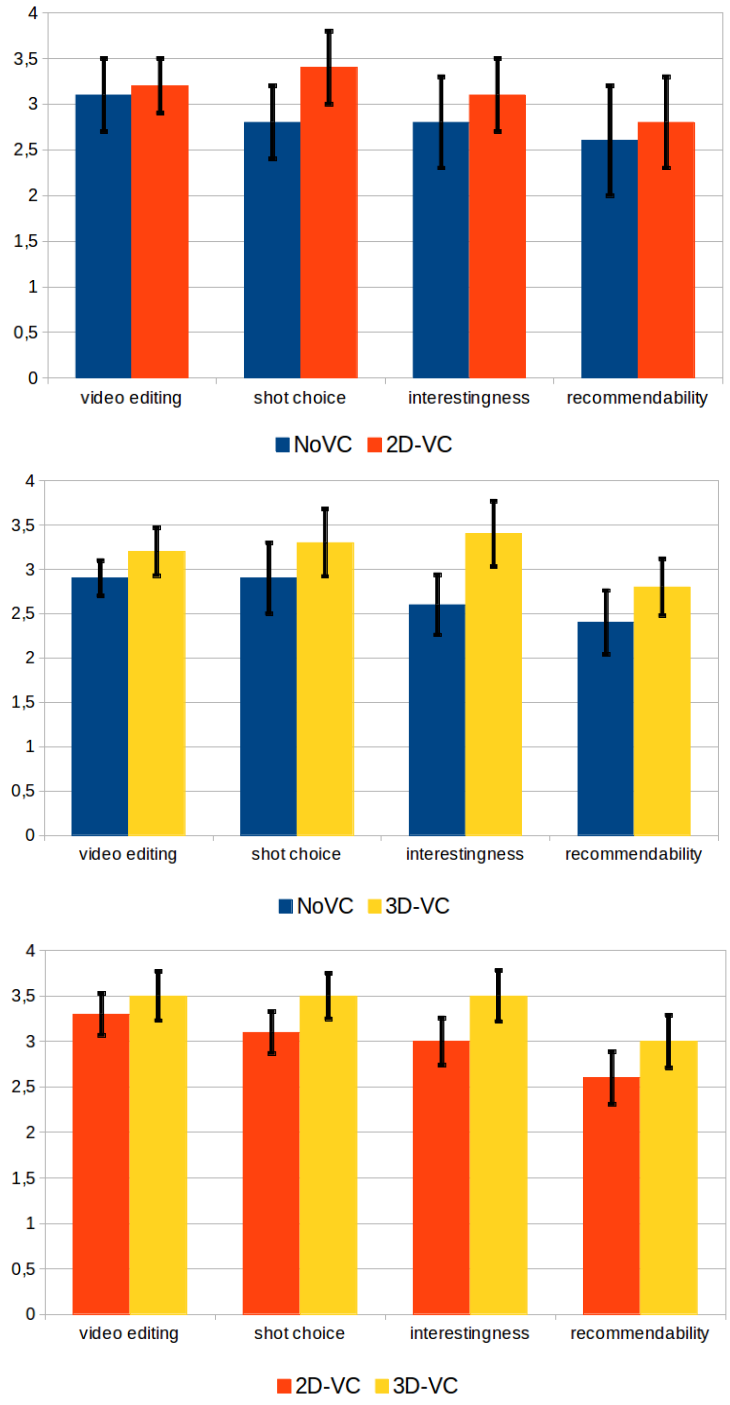


Figure 4.12: Average responses of the users for the three scenarios: NoVC vs. 2D-VC, NoVC vs. 3D-VC and 2D-VC vs. 3D-VC

paragraph 4.2.2 except that it is applied in reduced (2D and time) dimension. The difference between this 2D version and ours is that: (i) the saliency information is computed for each video individually (without considering 3D scene information from other videos), and (ii)

no interest information across different videos is available. We denoted the output from this video as 2D-VC. The output from our algorithm using 3D interest map is denoted as 3D-VC. Note that the three versions NoVC, 2D-VC, 3D-VC can be downloaded as supplemental material.

Users. We recruited 30 users (10 female, 20 male, age between 21-43) to participate in the user studies. All users but one have limited experience with video editing and production.

Methodology. The user study is conducted online. Users are presented with the instructions and are told that "the main purpose of the user study is to rate the videos according to the quality of video editing". We created two Web pages for our purpose, each of them displaying two videos. On one of the page, NoVC is displayed along with either 2D-VC or 3D-VC (the choice is random). On the other page, 2D-VC, and 3D-VC are displayed. The order of display of the videos is random on both Web pages, to eliminate bias. The users are allowed to replay the video as many times as they want. A list of rating criteria are presented to the user below the video, where they are asked to rate the videos with a rating from 1 (worst) to 5 (best) according to the criteria of (i) the quality of video editing, (ii) the choice of camera shot, (iii) the interestingness of the video, and (iv) the likelihood that the user would recommend the video to a friend.

Results. The average responses of the users, along with their 95% confidence interval, are plotted in Figure 4.12.

The first result that is very clear is that users prefer the versions with virtual camera movements (2D-VC, 3D-VC) rather than the output from MoviMash (2D-VC). The choice of camera shot (ii) is the criteria where the difference is the most obvious, which is understandable as variety in the choice of shots makes the video more enjoyable to watch.

The interestingness of the video (iii), meaning the capacity of a video to display the interesting subjects in a scene, is also significantly higher in 3D-VC than in NoVC. This proves that the 3D interest maps indeed help the mashup algorithm displaying the interesting regions in the videos. Note that 2D-VC also outperforms NoVC with respect to this criteria but that the difference is less significant. Indeed since the 2D interest maps used to compute this version are mostly based on apparent motion, the 2D-VC mashup has often multiple choices of moving regions to display and does not necessarily focus on the prominent region of interest.

The right diagram of figure 4.12 seems to indicate that 3D-VC performs better than 2D-VC, however the results can not be proven statistically significant. Indeed, to non-expert users, the two videos can look similar because the shots dynamics are the same (zoom in, zoom out, pan).

Therefore we conducted another, much shorter, user study. We asked a different pool of 45 users to look at three pairs of very short videos, each depicting a transition between 2 shots (e.g. figure 4.11). Each pair consisted in the transition computed by 3D-VC and by 2D-VC. We displayed the three pairs in random order, and asked users which transition

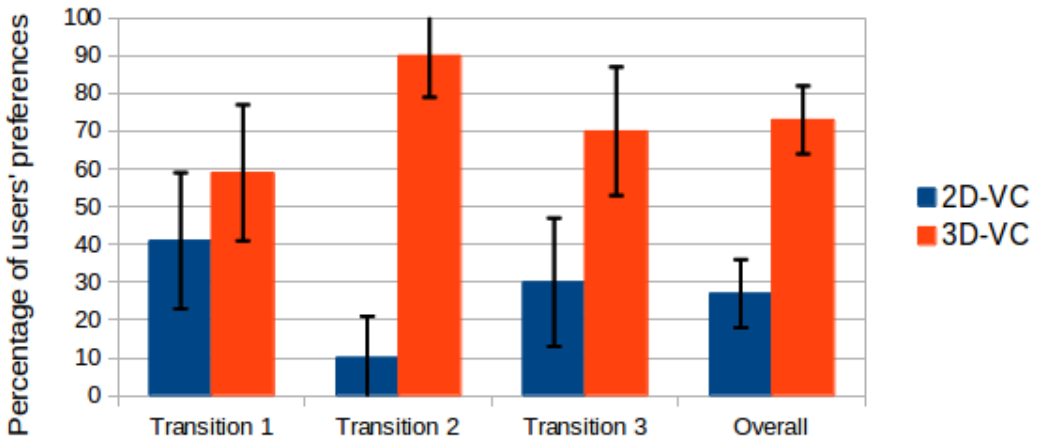


Figure 4.13: Second user study (99% confidence intervals)

they preferred. Unlike the results presented in figure 4.12, this study allows to clearly (statistically significantly) state that the transitions based on the 3D interest maps are better than the ones based on the 2D interest maps (see figure 4.13).

4.2.5 Summary and Conclusion

In this section we have shown how to generate a video mashup based on 3D user interest maps. The originality of this approach is that it allows for transitions from one video to another that are focused on the same regions of interest. Two user studies, with 30 and 40 users respectively, have proven that viewers prefer a mashup generated from the 3D interest maps.

In the next section, we introduce another application derived from the 3D user interest maps: navigating between videos by region of interest querying.

4.3 Querying multiple video streams

In this section, we still consider the use case of public events that are filmed by many users. This use case produces a set of synchronous videos that reveal 3D ROIs. We explain here how the 3D interest maps can be used to navigate between the videos.

Figure 4.14: Images of three synchronized videos from the **Fukuoka** dataset (on the left), and an example of 2 possible queries (on the right) materialized by a red rectangle and a blue circle on the image.

The idea of this work is introduced in Figure 4.14. The three images on the left show frames captured by devices recording the same scene at the same time. Those frames were extracted from a dataset of five videos depicting a chant-and-dance performance. At that particular moment, the leader of the band (wearing a white bow on her hair) is singing alone, and the other performers are dancing by her side. We can build a 3D interest map from these three images, as described in section 3.3. The rightmost image in Figure 4.14 shows a user interface on the mobile phone. In this figure, the user is viewing the stage from the right side, and is not able to get a good view of the performance. The user can issue a query by either tapping on the band leader on his screen (the blue circle) or click and drag a rectangle surrounding the band leader on his screen (the red rectangle). The query is sent to the server, which uses the 3D user interest map to find the video that will best answer the query.

In this section, we introduce the term *viewpoint entropy*, and explain how the problem of finding the video that best displays a Region of Interest can be reformulated as maximizing the *viewpoint entropy*. We then present a preliminary evaluation of our algorithm.

4.3.1 Viewpoint entropy

To simplify the explanation, we will present our algorithm in the context of a set of J images I_j, $j = 1...J$, corresponding to video frames taken from the same time instance from J cameras filming the same scene from different angles. We denote the cameras as $\mathcal{C}_j$ and we know the projection matrices of the cameras.

Let I_q be the image currently viewed by the user, onto which the user will specify a region of interest (ROI) R_q. We call I_q the query image and R_q the query region.

For now, we assume that we have a set of K 3D shapes, representing the interesting objects in the scene (see section 3.3). We back-project R_q onto 3D space, forming a query volume V_q that is a generalized viewing cone through R_q. We then compute the intersection

between V_q and the 3D shapes.

After this step, the algorithm selects a subset of 3D shapes $\mathcal{O}_q$ that intersects with V_q (we consider V_q intersects with a 3D shapes if more than 40% of a shape is within V_q). Note that, it is possible to select a shape corresponding to an object that does not appear in I_q.

The set $\mathcal{O}_q$ represents 3D shapes selected by the user through R_q, and ideally, would correspond to the set of objects of interest in the scene that the user is interested in. What remains is for the algorithm to return an image that depicts these objects in the "best" way. To compute this, we use the notation of *viewpoint entropy*, as inspired by [Vázquez 2001].

For each image I_j, we compute its viewpoint entropy. We adapt the notion of viewpoint entropy by Vazquez et al. to handle a finite set of 3D shapes and the restricted region of background visible from I_j. The viewpoint entropy $E(I_j)$ represents the amount of visual information about the selected 3D shapes $\mathcal{O}_q$ in I_j.

Let $\mathcal{A}_o$ be the projected area of shape o on I_j, normalized to between 0 and 1 with respected to the area of I_j. We define $\mathcal{A}_{bg}$ as the normalized area in I_j that is not covered by any shape (i.e., the background).

We define the viewpoint entropy as

$$E(I_j) = -\mathcal{A}_{bg} \log_2 \mathcal{A}_{bg} - \sum_{o \in \mathcal{O}_q} \mathcal{A}_o \log_2 \mathcal{A}_o \tag{4.6}$$

A image that depicts all the requested shapes with the same relative projected area would have the highest entropy. Since the relative projected areas $\mathcal{A}_i$ form a probability distribution, the relative visibility of the background at maximum entropy ($\log_2(|\mathcal{O}_q| + 1)$) should be also comparable to the visibility of each shape. In practice, we do not reach this upper bound and simply maximize $E(I_j)$ over j.

We return the image with the highest entropy as the result of our query. The system then switch the video stream to the one corresponding to the resulting image.

As clearly mentioned by Vazquez et al. the intervention of $\mathcal{A}_0$ helps to handle various zoom levels (or various distances between the cameras and the scene) among the J candidate images. The use of the background visibility level gives nearer shapes a higher entropy. In Figure 4.15, a larger projection of the requested yellow shape increases the entropy of the best viewpoint (by limiting the information brought by the background).

Figure 4.15 shows a small example illustrating this process. The small red square inside the image represents the query R_q region. The corresponding query volume V_q intersects two shapes, shown in blue and yellow, out of three 3D shapes. The image with the highest entropy (0.75) is selected out of two candidate images.

4.3.2 Evaluation

We evaluate our query application through a user study, since only users can tell if the answer to a query matches what they want to see. Our user study has the following set up.

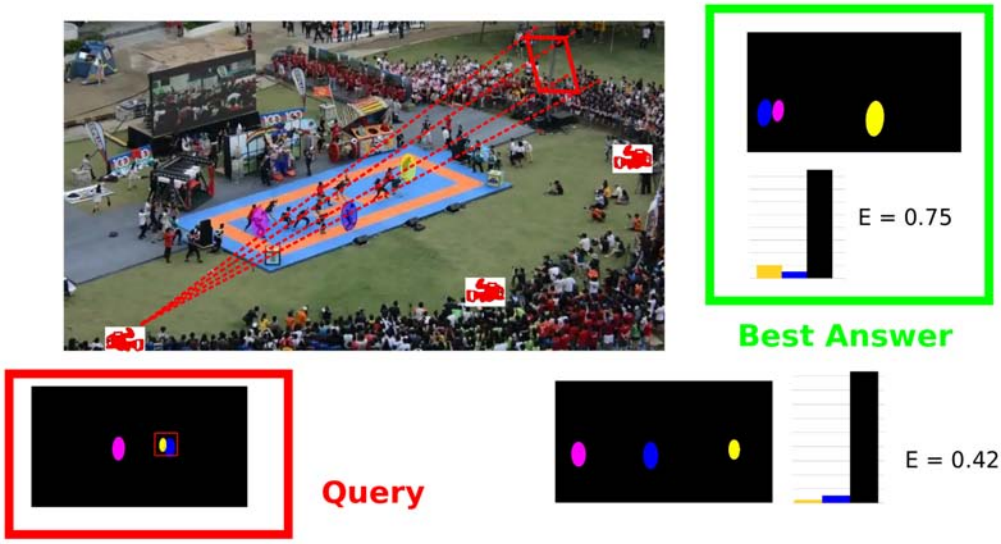


Figure 4.15: A 3D query volume V_q intersecting $K_q = 2$ shapes and $J = 2$ candidate viewpoints.

Images We selected 15 time instants from our three datasets (described in section 3.3.3): **Fukuoka**, **Jiku** and **BrassBand**.

Methodology For each sequence, one of the images is considered being captured by a user who wants to query the system. On this image, we define five spatial regions corresponding to the actual query we want to evaluate. For each query we ask users which of the remaining videos they think is the best answer to the query. The order of presentation of the videos is randomized, as well as the order of the sequences and the order of the queries, in order to avoid any bias.

We set-up a web based interface, displaying the query (as a red rectangle on one of the images) and the entire set of videos. Users were asked to look at the query, identify the objects of interest that were queried and find among the set of videos which one they think shows best the queried objects of interest. They can then select the video that answers the query by clicking on it, and then move on to the next query.

Participants. The study involves 35 participants with ages ranging from 20 to 60, and averaging 30. Among these users, 13 were females and 22 were males. The 35 participants each answered to 75 queries, which makes a total of 2625 answers.

Results. We say that an user is satisfied by the answer to a query if the answer given by our algorithm is the same as chosen by the user during the user study. This hypothesis assumes that users would be satisfied with one answer only, which is not true in general. For example, in the *BrassBand* dataset, some of the cameras were located pretty close to each other, which means that in some cases users would have been satisfied by both videos as an answer to a query.

Dataset	Level 3	
Number of answers	1	2
Fukuoka	30%	63%
Jiku	31%	49%
BrassBand	36%	63%

Table 4.3: Percentage of users that would be satisfied with the result after the 1st and 2nd answer on the three datasets.

Dataset	1st answer	2nd answer
Fukuoka	61%	80%
Jiku	45%	72%
BrassBand	56%	79%

Table 4.4: Percentage of the users it was possible to please after the 1st and 2nd answer on the three datasets.

Table 4.3 shows the average percentage of users that are satisfied by the top answer returned by our algorithm, and by the top two answers returned by our algorithm. The results are good for the *Fukuoka* and *BrassBand* datasets: one third of the users are satisfied with the top answer and two thirds of users are satisfied by the top two answers. These results should be compared to the ones introduced in Table 4.4, which shows the maximum percentage of users that it is possible to satisfy with one answer, and with two.

The remaining users chose different views, which does not necessarily mean they would not have been pleased with the answer from our system. In fact, in the case of the **Fukuoka** and **BrassBand** datasets, there are often multiple views that would give acceptable answers to a query, as some cameras are close from each other.

4.3.3 Summary and conclusion

In this section we have shown how to apply the concept of viewpoint entropy to an application of region of interest (ROI) querying. Preliminary evaluations have shown promising results, but they need to be extended to a more extensive real-world setup to determine its real effectiveness. We already developed a web interface to navigate within a set of videos, and we plan in future work to evaluate this interface in a user study.

Part II

Crowd-based semantic analysis of images

Crowd-based semantic analysis of images: Overview

We have shown in part I how to detect ROI using implicit interactions from a crowd of users. We have seen how the detected ROI benefit both from user interest maps and content analysis techniques. We have modeled the user interest maps as Gaussian mixtures, and have demonstrated a set of applications that take advantage of our detected ROI.

The main drawback of all the work we have presented in part I is the limited amount of semantics that is contained in our ROI. Indeed the interactions through which users contributed to build our user interest maps did not allow them to introduce any semantic label for example.

In this part, we are adressing the problems of object detection and image segmentation. Unlike ROI detection, these two problems imply the inference of a lot of semantic information in order to achieve good results. Whereas in part I our algorithms involved a low level of semantic and users' attention, in part II we study the level of cognitive commitment needed from users in order to bring information embedding a higher semantic level. Ideally we would like to implicitly collect semantic labels (e.g., textual of audio labels). But is it possible to implicitly collect that much semantic information?

We start by introducing a new interactive segmentation interface called Click'n'Cut. In the corresponding chapter 5, we show how a combination of light interactions (a few clicks) from users with content analysis techniques can lead to high quality figure-ground segmentation. Then in chapter 6 we introduce our Game With A Purpose, Ask'nSeek, through which we collect rich annotations that we use to perform object detection and image segmentation.

Finally in chapter 7, we use the previously introduced Click'n'Cut and Ask'nSeek systems to answer these fundamental questions: what is the loss induced by assigning a task to a crowd of paid workers instead of experts? And what is the loss induced by assigning the same task to players through a game? How to minimze such losses using content analysis?

Chapter 5

A baseline for Interactive Segmentation

Contents

In this chapter, we adress the problem of interactive segmentation of images via the development of an interface called Click'n'Cut. In this scenario and unlike the work introduced in part I, interactions are fully explicit. A user (or a crowd of users) is presented with a semantic label describing an object and must segment the object using the interactions proposed by the interface. Users can be experts (in which case we can compare our interface to existing state-of-the-art interactive segmentation interfaces such as [Rother 2004]) or paid workers who do not know anything about image segmentation. In our case, semantic labels are assumed known (a possible way to get the labels is to use the ESP game [Von Ahn 2004] or an equivalent platform) but we could devise an alternative interface in which users would enter semantic labels and then interactively segment the corresponding object.

In this work, we had two main requirements for the interface's design. First we wanted to keep users' interactions at a minimum. Annotations in interactive segmentation are usually scribbles (see section 2.2.3); we decided to use points instead. Points have the advantage to be produced quicker than scribbles (which can take a few seconds to draw), which allows users to rapidly shape an acceptable segmentation mask. We also wanted to obtain an interface as reactive as possible. The challenge was then, as we have already discussed in section 2.2.1, to design a content analysis algorithm that was both fast and efficient to expand users' interactions.

In this chapter we evaluate segmentation performances of expert users with Click'n'Cut, in order to obtain a baseline for interactive segmentation. We will then, in the subsequent chapters, compare the performances of a crowd of paid workers against expert users. Finally we will compare to this baseline the quality of a segmentation obtained through a new Game With A Purpose: Ask'nSeek.

5.1 Click'n'Cut: an interface for Interactive figure-ground Segmentation.

In this section we describe our web interface Click'n'Cut for interactive object segmentation.

5.1.1 Presentation of the interface

Figure 5.1 shows a screenshot of the Click'n'Cut interface. The interface consists in displaying the image that we wish to segment, along with a set of basic interactions (on the bottom-right of the screen) and a reminder of how the interface works (on the top-right part of the screen). There is also a description of the object to segment on the top of the screen, right above the image. On the figure, the object to be segmented is the cat.

The fundamental interactions available to users are the left and right clicks. A left click on the image indicates a *foreground* point (in green) whereas a right click on the image indicates a *background* point (in red). After each click the current version of the

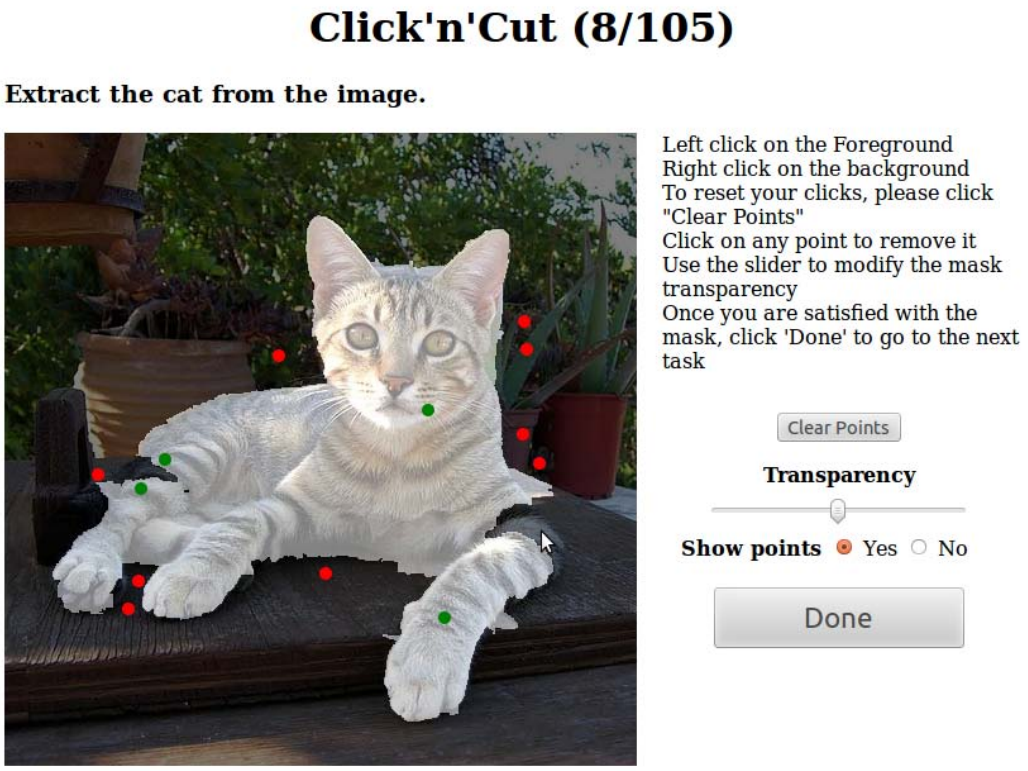


Figure 5.1: Screenshot of the Click'n'Cut interface

segmentation is updated and displayed over the image with an alpha value of 0.5 by default. At any time the user can choose to modify the alpha using the *Transparency* slider to either get a better look at the image or to better see the current Foreground mask.

A user can also remove a bad click: just clicking on it again makes it disappear. The *Clear points* button removes the entire set of clicks that have been made by the worker. Finally, once satisfied with the result, the user can go on to the next task by clicking the *Done* button.

The user can also choose not to display the points (annotations), in order to have a clearer view of the current state of the segmentation. The radio buttons "Show points" serve this purpose.

The number of interactions available to users is voluntarily kept low: we wanted to obtain an easy-to-use interface with intuitive interactions. We therefore did not implement any tutorial (we will see later in this chapter that it was not a problem for experts; unfortunately it became an issue with paid workers as we will show in section 7.1.2).

The interface is implemented using HTML5 features and JavaScript on the client side, and in Java on the server side. The server side handles the computation of the current best mask as well as the persistance of workers' interactions in a database. The mask that is produced by users via Click'n'Cut is a binary image in which a pixel value is set to 1 (resp.

0) if the pixels belongs (resp. does not belong) to the object. The algorithm we present in section 5.2 could be modified to produce probability maps, or tri-maps for example.

The interface is deployed at the following adress: `http://carlier.perso.enseeiht.fr/demos/clickncut.html`.

5.1.2 Content-Aware interface

On the server side, an algorithm (introduced in section 5.2) binds the clicks to a pool of segments that are used to build the object's mask. The very simple idea behind the algorithm is that when users label a pixel as foreground (by left-clicking on it), all the pixels that belong to the same segment (obtained through content analysis) are also labeled as foreground. Segments can similarly be labeled as background.

We use different granularities for the segments in our algorithm. Large segments allow to infer a lot of information from a single click but the segmentation quality greatly depends on the segments quality. Small segments are less informative since they expand the users annotation to less pixels, however the information propagation is less likely to be wrong than with large segments.

Superpixels. The term of superpixels has appeared at the beginning of the 2000s. It could be defined as an intermediate level of abstraction between pixels and objects in a visual content (usually, authors refer to superpixels for images and supervoxels for videos). Superpixels can be seen as an oversegmentation of an image, in which each segment regroups pixels that exhibit a strong colorimetric and/or textural similarity. Originally superpixels were strongly related to computer vision techniques that used Markov Models. These methods typically build graphs to solve problems, and using superpixels (instead of pixels) as nodes for these graphs greatly decreased the complexity and computation time of these algorithms. Nowadays however, superpixels are widely used as a preliminary step in many algorithms.

Categorizing this class of algorithms is not easy, since superpixels are almost always introduced with a specific application in mind. Superpixels attributes are then very dependent of the desired application.

Superpixels can either be very compact to form a regular grid (e.g., [Achanta 2012]) or have free form to better fit objects boundaries (e.g., [Felzenszwalb 2004]). The number of desired superpixels can be a parameter (e.g., [Achanta 2012, Shi 2000]) or can be dependent on some parameters (e.g., [Felzenszwalb 2004, Comaniciu 2002]).

Most of the superpixels algorithms share a common objective though: they aim at offering the possibility to describe any object on the image as a set of superpixels. This property is desirable to us; in fact a very simple approach for an interactive segmentation could be to designate each superpixel as foreground or as background (using foreground clicks and background clicks, like in Click'n'Cut). Because this approach would still require a lot of interactions from users (since an object can be composed of hundreds of superpixels), we also choose to use object candidates in our algorithm.

Object Candidates.

The concept of object candidates has been made popular by the PASCAL Visual Object Challenge [Everingham 2010]. In the object detection challenge, a class of algorithms emerged as the best performers: the algorithms using deformable-part models. The idea is to decompose an object into smaller objects called parts whose spatial arrangement can vary, and to try to detect the object by finding its parts. In this context, it is really important to be able to detect regions in an image that are good candidates to be objects. The problem of object detection is then reformulated as checking whether the candidates fit a deformable-part model.

Many methods have been proposed to build such object candidates. Carreira et al. [Carreira 2010] generate candidates using a Parametric version of the Min-Cuts algorithm: they run Min-Cuts with different set of parameters to obtain a pool of candidates. Then they rank the candidates by applying a classifier trained using 34 low-level features that determines whether the candidate has a high probability of being an object or not.

Van de Sande et al. [Van de Sande 2011] build a hierarchy of segments (obtained with an initial superpixel segmentation from [Felzenszwalb 2004]) using a greedy algorithm. All the intermediate segments in the hierarchy are then considered as candidate objects. Arbelaez et al. [Arbeláez 2014] use a similar approach that differs on the superpixel segmentation as well as on the algorithm used to rank the objects.

In our work we use object candidates along with the underlying superpixel segmentation in conjunction with users' clicks. The clicks are used to select the best candidate segment or combination of segments, as we will explain in the next section.

5.2 Interactive Segmentation

To compute the best mask with respect to a set of f foreground points and b background points, we adopt the following algorithm.

For each mask $m \in MCG$, where MCG is the set of masks computed using the method from [Arbeláez 2014] (see some examples on figure 5.2: masks can be of various size and nature), we start by computing two scores fg_m and bg_m. fg_m (resp. bg_m) is the number of foreground (resp. background) points that are correct with respect to m. For example on figure 5.2 the foreground point (in green) is correct with respect to the masks 1 and 4.

Then if there exists a mask m^* for which $fg_{m^*} = f$ and $bg_{m^*} = b$ then m^* is the best possible mask and this is the mask that will be shown to the worker.

Else, it means that no mask is correct with respect to all clicks. In that case, we build a set of masks $M^* = \{m \in MCG, bg_m = b \text{ and } fg_m > 0\}$. This means that M^* contains the masks that have not been defined as background and for which there is at least one foreground point. The union of all masks that belong to M^* form the best mask that is displayed to the worker.

For example, in the example depicted on figure 5.2, the first mask is the best mask with

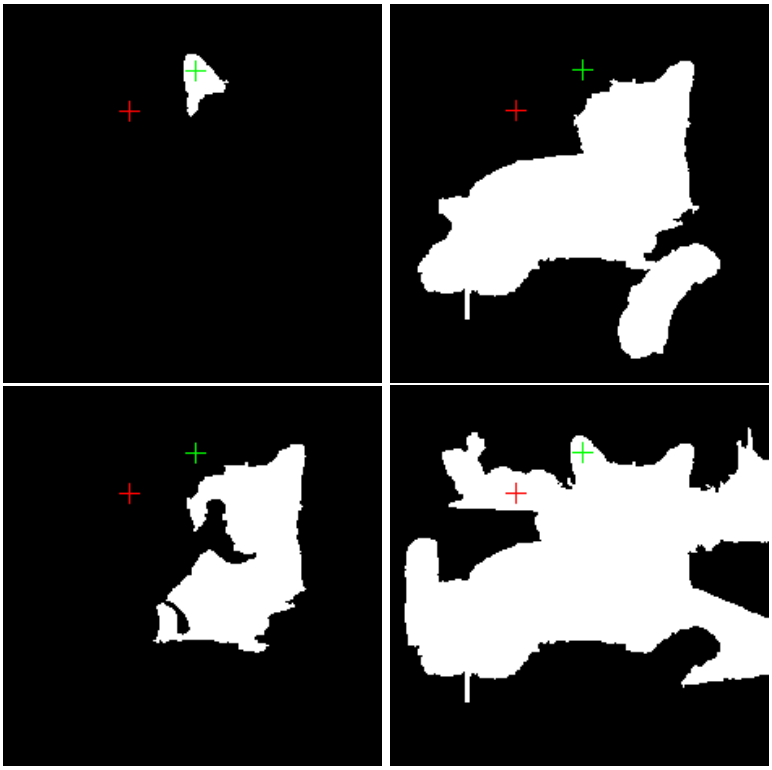


Figure 5.2: Examples of MCG candidates for the image shown in figure 5.1

respect to the two clicks (foreground in green, and background in red) since it is the only one that is consistent with the two clicks. If the user was to label a pixel on the cat's head as foreground, then no mask would be consistent with the three clicks. The best mask would therefore be the combination of the three first masks which would all be consistent with the background click and at least one foreground click. The fourth mask on the figure is clearly identified as too big since it contains a background click.

This simple algorithm presents multiple advantages. First it is very quick to compute the best mask which makes the interface pleasant to interact with, since the response time is low. As we will show in the next section, the interface is also very efficient in term of time needed to converge towards a good segmentation. The major drawback of this algorithm however is that it relies on the users capacity to produce accurate clicks. Just one wrong background click can dramatically reduce the segmentation quality.

5.3 Experiments

5.3.1 Protocol

Dataset. The experiments have been conducted on a pool of 101 images. 96 images are taken from a subset of the Berkeley Segmentation Dataset [Martin 2001], and have been

augmented with a description of 100 objects that appear on the images (2 images have three associated objects each). This dataset has originally been proposed by [McGuinness 2010] in their work on interactive segmentation. In order to study and control the quality of user traces we have added 5 images from the PASCAL VOC dataset along with a description of one object per image.

Therefore our image set is composed of 101 images, and there are 105 tasks (objects to segment) to perform. 5 tasks serve as a gold standard.

The tasks can be divided into four main categories:

- **Unique and salient object** (see middle image in figure 5.4). Only one instance of the object to segment is present on the image, and is significantly different than the background.
- **Very small or very big object** (see right image in figure 5.5). These tasks feature objects which are either very small (with respect to the image size), or very large (they cover almost entirely the image).
- **Object similar to the background**. The object to segment has similar color and texture than the background. An example of this can be seen on figure 5.5, in which the puma's back and legs are hardly recognizable from the background.
- **Multiple instances of the object**. The object to segment belongs to a class for which multiple other representatives appear on the image. The giraffe in figure 5.5 illustrates this category.

Participants. We asked experts (PhD students as well as Professors) from different computer vision labs to interactively segment the objects using Click'n'Cut. We asked the experts to try to reach for the best possible segmentation. Users performed the segmentation of the entire set of 105 tasks, which took them between 30 minutes to 2 hours.

A total of 15 experts (11 Males, 4 Females) participated to this study, with ages ranging from 19 to 55.

5.3.2 Results

Our interactive segmentation tool was compared with the top two best configurations proposed in [McGuinness 2010]: GrabCut [Rother 2004] and hierarchical partition with BPTs [Salembier 2000] [Adamek 2006]. The different solutions are assessed in terms of the accuracy vs user time trade-off, where segmentation accuracy is measured with the Jaccard Index (overlap score) $J = \frac{P \cap GT}{P \cup GT}$ between the Predicted (P) and Ground Truth (GT) masks. The graph in Figure 5.3 plots the average Jaccard obtained with the amount of time users spend creating their annotations. Our experiments indicate that Click'n'Cut (used by experts) converges more rapidly than the two graph-based approaches, but also that accuracy saturates sooner in our proposal.

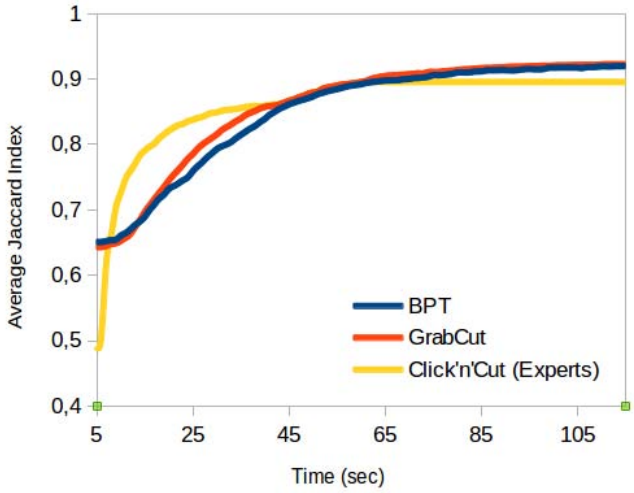


Figure 5.3: Average Jaccard vs User time.

The quick convergence towards a good segmentation is a natural consequence of the requirements we expressed at the beginning of the chapter. It takes our expert users a few clicks to reach a good segmentation result (the 0.8 Jaccard is reached in average in less than 20 seconds), and also the system provides feedback more quickly than the BPT and GrabCut approaches to which we compare ourselves. The combination of slower interactions (scribbles take longer than clicks to produce) as well as slower feedback (our algorithm's response is instantaneous compared to GrabCut for example) explains the relative slowness of state-of-the-art methods compared to our interface.

Figure 5.4 shows the three images for which the expert users obtain the best results (with an average Jaccard Index of respectively 0.97, 0.98 and 0.98). We can see that the utilization of Click'n'Cut can lead to almost perfect segmentation masks.

On the other hand, using clicks instead of scribbles also shows its limitation: our expert users converge towards a 0.90 average Jaccard Index, whereas BPT and GrabCut's users reach a 0.93 average Jaccard Index on the same data (figure 5.3). This fact can be understood by looking at the left image in figure 5.4. The mask boundaries are accurate, but there remain holes inside the mask that need to be filled. This is much easier to achieve with scribbles than with clicks and that can partly explain the performance difference between BPT and GrabCut's method and our method.

Figure 5.5 also gives an interesting insight about the limitations of our approach. The figure depicts three images for which experts fail to reach an acceptable segmentation (with an average Jaccard Index of respectively 0.72, 0.70 and 0.71). On the left, the puma is difficult to segment because of the underlying superpixel segmentation used in the MCG candidates. Indeed, a big part of the puma is missing in the segmented mask because it is part of a large superpixel that covers both the puma and the background. In other words, in

Figure 5.4: Original image and best mask obtained on the three tasks for which experts produced the best segmentation in average.

this particular case, experts fail because the content analysis algorithm makes it impossible to reach a good segmentation. The giraffe and camel objects are difficult because they are small objects, and some of their parts (mostly, the animals' legs) are so small that they are either difficult to click on, or not differentiated from the background in the superpixel segmentation.

In fact, our results clearly show that our algorithm is very performant for the first category of tasks (**unique and salient object**) and that two categories are problematic: **very small object** and **object similar to the background**. Our underlying content analysis algorithm (MCG candidates) performs less good on images of these categories which limits segmentation quality.

Table 5.1 compares many different algorithms (described in section 2.2.3) performances. We can see that Click'n'Cut allows users to achieve a very interesting compromise between segmentation quality and interaction time.

5.4 Summary and Conclusion

In this short chapter we have introduced a new interface for interactive segmentation called Click'n'Cut. This interface admits very simple interactions -left clicks on foreground, right clicks on background- and allows expert users to converge in about 30 seconds to a high performance segmentation (0.9 of average Jaccard Index). The results presented in this chapter are pending publication in CrowdMM 2014.

In this chapter we have temporarily set aside the notions of human computation and implicit crowdsourcing, but we will return to it in the two next chapters. We will first

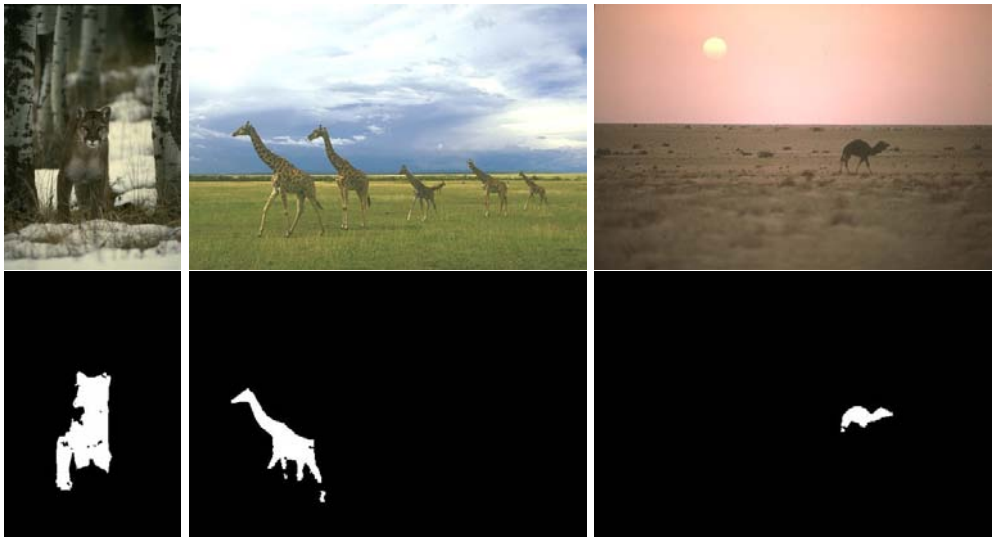


Figure 5.5: Original image and best mask obtained on the three tasks for which experts produced the worst segmentation in average.

introduce in chapter 6 a new Game With A Purpose called Ask'nSeek, which has the potential to generate traces that are similar to Click'n'Cut. We will describe in this next chapter another application for these traces: object detection.

Click'n'Cut will be used again in chapter 7, but not by experts anymore. We will investigate whether a crowd of paid workers can achieve similar results than experts when using our interface. We will also analyze experts traces more in depth.

The results that we have introduced in this chapter will be used as a baseline in order to evaluate the performance of paid workers on Click'n'Cut, as well as players on Ask'nSeek.

	Dataset	User / image	Users type	Input	Avg. Time (sec.)	Best Jaccard
[Chen 2014]	KITTI	9	Experts	Tight Polygon	60	(Used as GT)
		?	Crowd	Tight Polygon	?	0.85-0.87
[Lin 2014]	Microsoft COCO	1	Crowd	Tight Polygon	79	(Used as GT)
[Jain 2013]	IIS+MSRC+CoSeg	5	Crowd	Box	7	-
		5	Crowd	Sloppy contour	20	-
		5	Crowd	Polygon	54	0.51-0.76
[McGuinness 2010]	BSDS (DCU subset)	20	Experts	Scribble	60-85	0.93
Click'n'Cut	BSDS (DCU subset)	15	Experts	Click	32	0.90

Table 5.1: Comparison of Click'n'Cut with similar approaches including average user time and best Jaccard.

Chapter 6

Going implicit using a Game With A Purpose

Contents

In this chapter, we introduce a new Game With A Purpose called Ask'nSeek. This game involves two players interacting with each other on the same image, thus implicitly producing very rich annotations on this image. The game is interesting because it produces different types of data such as free text (describing objects on the image), clicks and spatial relations between clicks and objects.

The data gathered through Ask'nSeek's playing sessions can be virtually used to help any computer vision algorithm. In this chapter, we formulate a model that extends annotations from the game to a set of points obtained through content analysis. This model outputs labelled bounding boxes, which makes it an object detection algorithm.

The data can also be used (as we will show in chapter 7) for object segmentation using the same algorithm that has been presented in the previous chapter, at section 5.2.

We first detail Ask'nSeek's gameplay, then explain our model for object detection and finally describe our experiments to validate the model.

6.1 Ask'nSeek: a game for object detection and image segmentation

6.1.1 Presentation of the gameplay

Ask'nSeek is a web-based two players game. The two players have non-symmetric roles; the first player, called the master, must hide a target region in an image and from that point should guide the second player, called the seeker, to discover it. The seeker can see the same image as the master but does not know where the target region has been hidden. His goal is to find the region by clicking inside it.

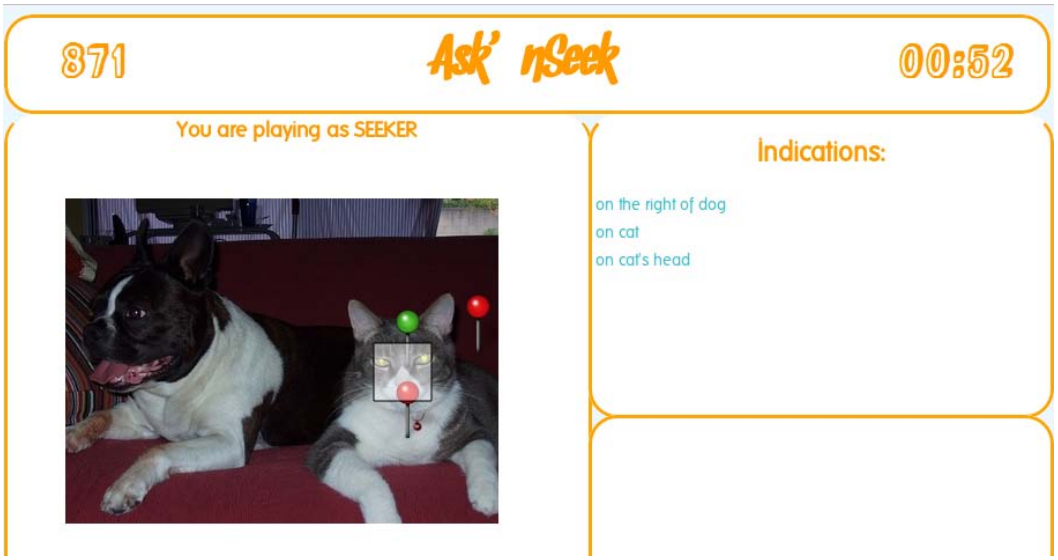


Figure 6.1: Example of the game from the seeker's point of view.

To do so, the seeker can ask the master for clues. More specifically, the seeker can type the name of the objects that he sees on the image, and ask the master where the region is relatively to these objects. Once prompted by the object requested by the seeker, the master can answer 7 possible answers : *on the left of*, *on the right of*, *above*, *below*, *on*, *partially on*, *can not relate to*. Therefore, the seeker can iteratively narrow down the possible locations for the hidden region on the image.

The game is cooperative: the goal of the master is to help the seeker find the target's location within the allocated time (that is displayed on the upper right part of the interface).

The gameplay is illustrated on figure 6.1. In the game featured on this example, the seeker first asked the master an indication relative to the dog, and the master answered that the region is "on the right of the dog". The seeker clicked on the image but not on the region, so he asked for a second clue, relative to the cat. The master answered that the region is "on the cat", and the seeker once again did not find the region. He finally got the indication that the region is "on the cat's head", and clicked on the right location. Once he clicks inside the region, the actual location of the region chosen by the master is prompted to the seeker (before finding it, he could not see it). In other terms, the square only appears on the image when the seeker managed to click inside it.

6.1.2 Nature of the data

The data collected on Ask'nSeek comes from the interaction between the 2 players, the master and the seeker. It is important to understand that the input from the 2 players is different, and brings complementary information about the image.

The seeker provides two kinds of information : textual information, that he types when describing an object, and spatial information that he brings when clicking on the image to find the hidden region.

The master on the other hand, provides two kinds of spatial information. The first information is the location of the hidden region, and the second one is the spatial relation between the object designated by the seeker and the hidden region. Because the seeker takes this information into account when clicking on the image, the spatial relation is also connecting the seeker's tags and clicks.

We now detail all the types of data provided by the players.

Textual information. The tags that are typed by the seeker are of various forms. Because the duration of the games is limited, players are tempted to write short tags. Indeed most of the tags are often composed of one word, which typically refers to an object of the image or to an object's part. This can be seen on figure 6.2, in which tags and their number of occurences are displayed on the right of the associated image. We can see that the words "tree", "plant" and "tiger" appear as one-word terms, as well as "face", "tail", or "tongue" which are meronyms of the object "tiger". Analysis of such patterns can help performing the task of *image labelling*.

It also happens that the tags are a group of words, which can be either very informative

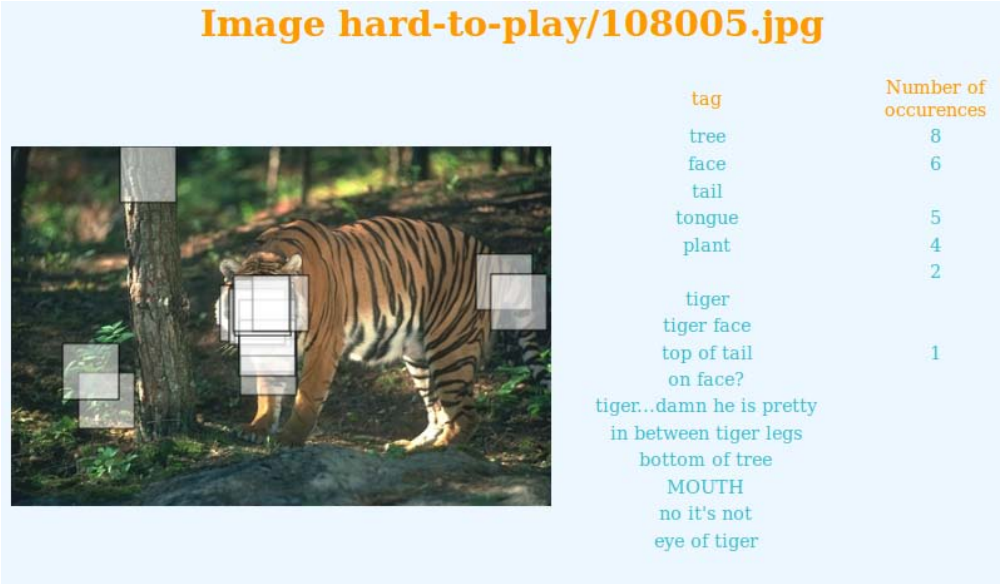


Figure 6.2: All games that have been played on one particular image. On the left, the location of hidden regions is superimposed on the image. On the right, the tags that have been typed by the seekers, along with their number of occurences.

because they connect two one-word objects ("eye of tiger", "tiger face") or either complicated to parse ("in between tiger legs"). The analysis of these traces can lead to learn a *hierarchy of objects*. On the example, it means that we can retrieve from the traces that tigers have a face and an eye.

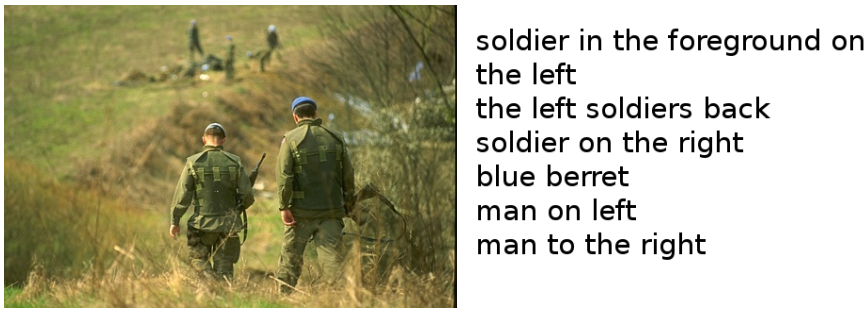


Figure 6.3: On the left, image that illustrates the problematic of multiple instances of the same object and on the right the tags collected during games associated to the image.

Also when people start using groups of words to designate an object, it usually means that there are multiple instances of this object. For example figure 6.3 depicts some labels that appeared on the traces associated to the image on the left. The label "soldier in the foreground on the left" suggests there are probably soldiers in the background and in the foreground, and probably more than one soldier in the foreground. Therefore, there is an interesting relation between the tags' length and the *number of instances* of an object.

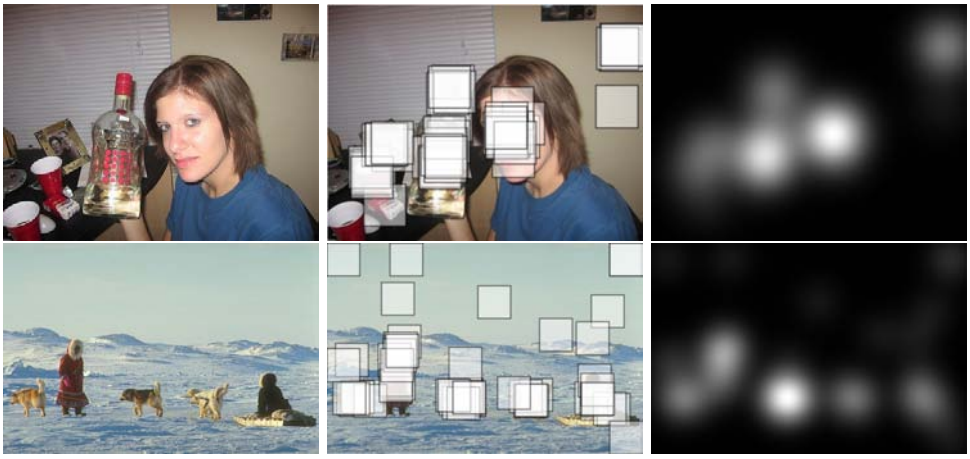


Figure 6.4: Saliency map computed from the hidden regions locations.

Finally, considering all different tags of an image and their frequencies can provide meaningful insight to solve problems like *image categorization* and *objects cooccurences*.

Hidden regions. Hidden regions are the regions that are positioned by the master during Ask'nSeek. Figure 6.4 displays on the middle column the location of all regions hidden by all masters on two images.

It is interesting to notice that the positions of those squares is not random at all, but often located on objects of interest. This can be explained by the nature of the game : because it is cooperative, the master wants the seeker to find the region, so he places it on rather obvious regions.

This makes the hidden regions good candidates to build **saliency** maps. To illustrate this idea, we associate a gaussian footprint to each hidden region and display the associated mixture of gaussians on the right column of figure 6.4. The result looks very convincing and would deserve to be investigated more thoroughly.

Spatial relations and clicks. All clicks from the seeker are associated to a spatial relation (*above*, *below*,*on the left of*,*on the right of*,*on*,*partially on*) and to a tag.

- *Above,below,left,right.* Figure 6.5 illustrates how we can use these clicks in order to perform *object detection*. Red points are all clicks collected above the cat, whereas blue points are clicks on the right of the cat. The lines correspond to the bounding boxes we can get for the detection of the object "cat" if we trust completely the traces (dotted line), or if consider there can be mistakes (full line). To handle the mistakes in this case, we use the median point (as opposed to the extreme point for the dotted line) to obtain a more robust estimation of the cat's position.

- *On.* Figure 6.6 shows all the "on" points obtained on one image. The distribution of the points can help to *segment* some of the objects (hut, man, trees, stick) . The right image shows "on" points obtained on objects (head, skirt, butt, feet) that are part of

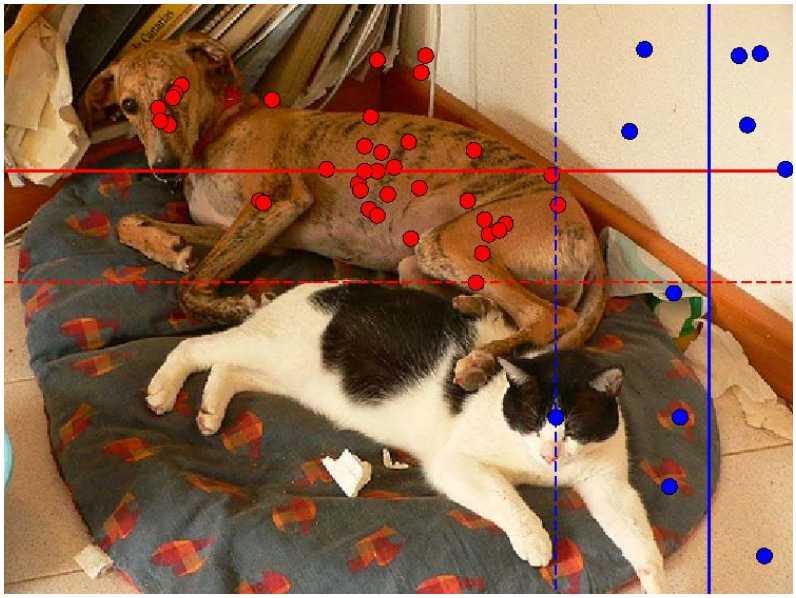


Figure 6.5: Clicks above (in red) and on the right (in blue) of the cat.

Figure 6.6: "On" clicks collected on one image.

Figure 6.7: In green, all points that have been clicked "partially on" the urn.

a bigger one ("man").

- *Partially On.* Figure 6.7 shows all the "partially on" points clicked on the object "urn". It seems clear that the concept of "partially on" click was not understood the same by all players. Some click exactly on the border of the object, some click near the border either inside or outside the object. These traces can nevertheless be used as a complementary information for object segmentation, in conjunction with foreground points ("on" clicks) and background points ("above", "below", "left", and "right" clicks).

The power of Ask'nSeek traces relies on their diversity and their complementarity. We can use them to perform image segmentation of one object, but taking into account that for each image we have information on multiple objects (see figure 6.6), we can go further and target *image parsing*.

The originality of the game logs, that bring in addition to clicks and tags a spatial relation, can help to learn *spatial relations* between objects, multiple instances of the same object, and parts of objects.

Examples from the data. In order to show the potential of Ask'nSeek traces we had all users play the same image, featuring a cat and a dog and extracted from the PASCAL dataset. We recorded traces of 99 games on this same image. Figure 6.12 shows the position of the hidden region of all games. It is very interesting to note that the regions completely cover the two main objets of the image, namely the cat and the dog. The highest density of regions is located on the head of both animals.

What is also interesting is that the labels "cat" and "dog" are the two most cited labels, with respectively 67 and 66 occurences each. Note that this is without counting the occurence of the word "cat" and "dog" in others tags, like "cat's head" or "dog's leg".

Figure 6.8: Seeker clicks relative to the dog, categorized in foreground (yellow) and background(red).

Figure 6.8 shows all clicks from all seekers relative to the object "dog". Yellow clicks are "on the dog" where as red clicks are "above", "below", "on the left of" or "on the right of" the dog. The first thing that comes to sight is that there are mistakes in the clicks. For example, some yellow clicks are a little bit outside the dog. There is also one of the background click that is located on the dog. One of the biggest challenges of the processing of the traces is to be able to detect and eliminate these outliers.

6.2 Object detection

As suggested in the figures 6.5 and 6.8, some objects are more cited than others by the players (in this case: the cat and the dog). For these most cited prominent objects, we gather enough spatial information from the traces to be able to detect their location on the image. In this section we describe the probabilistic model and machine learning strategies that we use to perform this detection. Due to the relatively small number of training data points collected during game play (figures 6.5 and 6.8 are misleading in this regard, because the corresponding image has been played more than 100 times), our strategy lies within the "semi-supervised clustering with constraints" framework. The idea is to cluster a set of points, and to associate a cluster to each object. More specifically, we use: (i) the *cluster assumption* ("If points are in the same cluster, they are likely to be of the same class" [Chapelle 2006]) and the *semi-supervised smoothness assumption* ("If two points in a high-density region are close, then so should be the corresponding outputs" [Chapelle 2006]) when we take into account the "on" points collected from game logs as well as the points produced from a superpixel segmentation algorithm; and (ii) the *low-density separation* assumption ("The decision boundary should lie in a low-density region" [Chapelle 2006]) when we factor in the "partially on" points from game logs.

6.2.1 A probabilistic generative model

Our model aims at labeling an image using data from two main sources: game logs and output of suitable computer vision algorithms, e.g., bottom-up visual saliency maps and superpixel-based segmentation.

The game logs contain labels as well as 'on', 'partially on' and 'left-right-above-below' relations. Our goal is to label each superpixel with a well-chosen label from a set of candidate labels $[\mathcal{L}_1, ...\mathcal{L}_K]$. Examples of labels include foreground objects (e.g., dog, bus) as well as other semantically meaningful regions within the image (e.g., sky, road).

We formally introduce an observable set X_{cv} of 2D data points that come from applying an automatic, unsupervised **c**omputer **v**ision technique to the input image. The way these points are produced will vary depending on the underlying technique, e.g., by applying density sampling techniques to a saliency map of the image (thereby producing a smaller set of points, whose density is proportional to the saliency of the surrounding region). These points lie on the image space $\Omega = [1, N_{row}] \times [1, N_{col}] \subset \mathbb{R}^2$. Intuitively our labeling task is to assign the points from X_{cv} to the correct labels, *if feasible*. Of course, since some of the salient regions may not map to any of the K most cited labels, the points located in such regions cannot be assigned any label. This is why we extend our set $\mathcal{L}$ of candidate labels with a dummy label ϕ which denotes that a given point is not labeled.

$$\mathcal{L} = [\mathcal{L}_1, ...\mathcal{L}_K] \cup \phi = \{\mathcal{L}_l\}_{l=1...K+1} \tag{6.1}$$

Similarly we denote X^l_{on} (resp. X^l_{pon}) the 2D points that have been clicked during the games and identified to be "on" (resp. "partially on") the object label $\mathcal{L}_l$ (or l for the sake of simplicity). So, we handle, as input data, a set $\mathcal{X}$ of 2D points :

$$\mathcal{X} = X_{cv} \cup X_{on} \cup X_{pon} \tag{6.2}$$

As input data we also have all of the "left-right-above-below" relations collected from the game logs. We follow the notations from the TAS model [Heitz 2008]. The collected relations are indicator variables $R_{i,l,r}$ that indicate whether a point $x_i \in \mathcal{X}$ and label l have relationship r. Arbitrarily $r = 1$ (resp. $r = 2$, etc.) stands for x_i being "on the left" (resp. "on the right") of object label l. The set of collected relations is denoted $\mathcal{R}$:

$$\mathcal{R} = \{R_{i,l,r}\}_{x_i \in \mathcal{X}, r=1..4, l=1..K} \tag{6.3}$$

So far, we have observable labels plus data and constraints from $\mathcal{L}$, $\mathcal{X}$ and $\mathcal{R}$ respectively. Our probabilistic model also includes unobservable (hidden) variables to be estimated for labeling purposes. First, we need variables corresponding to the assignments of data points from $\mathcal{X}$ to labels from $\mathcal{L}$. As explained later, we will handle soft (or probabilistic) assignments in our Expectation-Maximization algorithm. Secondly, we introduce an unobservable set of unknown model parameters θ. Our model is a clustering model such that our data

point set $\mathcal{X}$ must be partitioned into $K+1 = Card(\mathcal{L})$ clusters where K clusters are associated with the K labels $[\mathcal{L}_1, ...\mathcal{L}_K]$ and the $(K+1)^{th}$ one is the dummy (or "no label") cluster. We look for a mixture model (parametrized by θ) that explains how our observed data points $\mathcal{X}$ have been generated from labeled clusters while respecting the constraints $\mathcal{R}$. In other words, a mixture model explicitly says that a point $x \in \mathcal{X}$ could be statistically generated in $K+1$ possible ways leading to a mixture distribution :

$$P_{mix}(x|\theta) = \sum_{l=1}^{K} \pi_l \ \mathcal{N}(x|\mu_l, \Sigma_l) + \frac{\pi_{K+1}}{v} \tag{6.4}$$

where K Gaussian distributions (one per label) are mixed with an uniform distribution ($1/v$ is a constant to be defined) under the control of the mixing proportions $\pi_l, \forall l \in 1 \dots K+1$. The parameters θ defining the model are the K 2D locations μ_l of labeled Gaussian clusters, the K 2×2 covariance matrices Σ_l and the $K+1$ mixing weights π_l that sum to 1.

Ideally, in such generative model (with a well chosen vector of parameter θ), the data points from X_{cv} would be correctly clustered and labeled when they are located within the K Gaussian components. Moreover, when the points are more likely explained by the uniform distribution of the mixture (i.e., they are far apart from the Gaussian clusters) they are not labeled. Learning such a mixture model would be straightforward if we knew the correct label $y_i \in \mathcal{L}$ for each data point $x_i \in \mathcal{X}$. In practice, we don't know these assignments and want to estimate them for a current (resp. initial) model parameter θ^c (resp. θ^0). In this context, we will use "soft assignments"

$$\widehat{p}(l|i) = P(y_i = l|x_i, \theta^c) \tag{6.5}$$

A set of soft assignments is denoted $\gamma = \{\widehat{p}(l|i) \ l \in 1 \dots K+1, x_i \in \mathcal{X}\}$ or γ^c to reinforce its dependence on the current model θ^c.

6.2.2 Parameter estimation outline

We estimate our model parameters using the common *maximum likelihood estimate* (ML). Given labels $\mathcal{L}$ and observable data $\mathcal{X}$ and $\mathcal{R}$, the ML estimate is:

$$\theta_{ML}, \gamma_{ML} = \arg\max_{\theta,\gamma} P(\mathcal{X}, \mathcal{R}|\theta, \gamma) \tag{6.6}$$

Equation 6.6 expresses that we look for both the mixture (i.e., the Gaussian clusters) and the assignments that best explain our data. This principled estimation is very intuitive and threefold: (i) we compute an initial estimate of the model involving small clusters and reliable assignments; (ii) we make these clusters grow by propagating the initial assignments and maximizing the likelihood (equation 6.6); and (iii) the optimization algorithm stops when the clusters boundaries reach their expected positions, eventually leading to a local maximum of the likelihood.

These steps are illustrated in Figure 6.9. In that figure, red circles represent X_{on} points, gray crosses denote X_{cv} points, and blue circles represent X_{pon} points. The green rectangle represent the "bounding boxes" obtained by taking into account left-right-above-below spatial relations (see also figure 6.5). The figure only shows X_{cv} points included in bounding boxes. The red ellipses denote individual Gaussian distributions, whose center is marked with a red pin. The clusters are initialized with reliable assignments (X_{on} points), grow to include as many X_{cv} points as possible, and finally stop growing when the constraints imposed by the X_{pon} points is reached.

A more detailed explanation of each of the three steps follows.

Initialization. It is straightforward to use "on" points from X_{on} to start building an initial model θ^0. Indeed, the set of "on" points collected from the game logs can be broken down into subsets of points located "on" each object label l : $X_{on} = \cup_{l=1}^{K} X_{on}^l$. Hence if we consider the n_l points $x_i^l \in X_{on}^l$ that have been spotted by the gamers as being "on" the object (label) l, it is easy to estimate both an initial location μ_l^0 and an initial dispersion Σ_l^0 from the standard ML estimates : $\mu_l^0 \leftarrow \frac{1}{n_l}\sum_{i=1}^{n_l} x_i^l$ and $\Sigma_l^0 \leftarrow \frac{1}{n_l}\sum_{i=1}^{n_l}(x_i^l - \mu_l^0)(x_i^l - \mu_l^0)^T$. In the same spirit, the initial set of soft assignments γ^0 can be partially fixed given X_{on} and the set of observed relationships $\mathcal{R}$. Obviously, knowing that points from X_{on}^l are located "on" the object label l can be translated in reliable (i.e. fixed) assignments $\widehat{p}(l|i) = P(y_i = l|x_i) = 1 \; \forall x_i^l \in X_{on}^l$. In addition the constraints from $\mathcal{R}$ naturally lead to bounding boxes $\mathcal{BB}^l$ for each label l such that many impossible assignments can be also stated $\widehat{p}(l|i) = P(y_i = l|x_i) = 0 \; \forall x_i^l \notin \mathcal{BB}^l$. A particular case also occurs when a point $x_i \in X_{cv}$ is out of *all* the bounding boxes. In this case we are certain to a "no-label" point: $\widehat{p}(K+1|i) = P(y_i = K+1|x_i) = 1 \; \forall x_i \notin \cup_{l=1,K}\mathcal{BB}^l$. All other assignments can be selected as equiprobable to form a sufficiently good set γ^0. Eventually, γ^0 also leads to the choice of the initial mixing proportions to complete the choice of θ^0 as detailed later.

Clusters growing. Since the initial Gaussian clusters are concentrated on the "on" points from the game logs, we then make them grow using an *expectation-maximization* algorithm. The first *Expectation* step improves the initial assignments γ^0 by first propagating the reliable assignments (already established $\forall x_i^l \in X_{on}^l$) to neighboring points from X_{cv}. During this E-step that produces a new set γ^c of soft assignments, the current model setting θ^0 is fixed. In the subsequent *Maximization* step, the likelihood is then maximized with respect to θ given the current assignments γ^c. A new model is produced and passed to the next E-step as a current θ^c and so forth. These iterations (which are further detailed below) are made to monotonically increase the likelihood until convergence.

Stopping the growth. By construction, the expected image labeling will be deduced from the K (one per label) Gaussian clusters that are in the process of growing. Therefore, we have to stop the growth of the clusters in the image space when the clusters of labeled points reach the boundaries of the associated object. The "partially on" points (from X_{pon}) can be used to do so. In the next subsection, we will see that the factorization of our likelihood (equation 6.6) integrates a simple term that naturally minimizes the distances between the

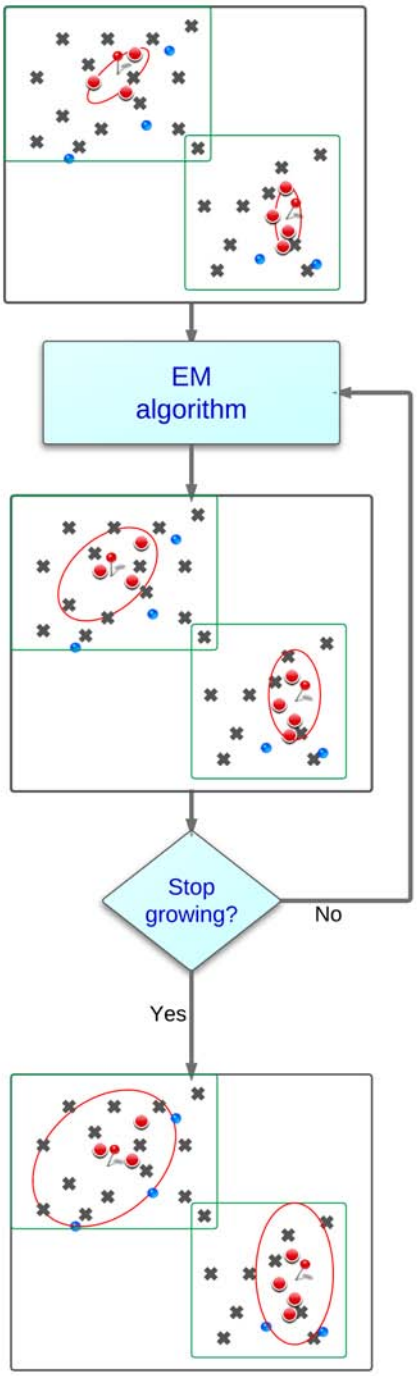


Figure 6.9: Parameter estimation using EM algorithm.

"partially on" points (X^l_{pon}) for a given label l and the boundary of the associated cluster. In our set-up, this boundary is chosen by thresholding the density $\mathcal{N}(x|\mu_l, \Sigma_l)$ or the associated squared Mahalanobis distance d^2_M. We formally define the elliptical Gaussian footprint S_l

of the l^{th} cluster as $S_l = \{x \in \Omega | d^2_M(x) = (x-\mu_l)^T \Sigma_l^{-1}(x-\mu_l) \leq \xi\}$. The threshold ξ is selected such that $P\left(d^2_M(x) \leq \xi \mid x \sim \mathcal{N}(\mu_l, \Sigma_l)\right) = 0.95$. Hence the boundary ∂S_l of the l^{th} cluster is basically $\partial S_l = \{x | d^2_M(x) = \xi\}$. Stopping the growth of the l^{th} cluster should be done when its boundary ∂S_l fits well with the locations of the "partially on" points (X^l_{pon}).

6.2.3 EM estimation

This subsection provides further technical details about our actual EM algorithm. First, we need to formally decompose the likelihood from equation (6.6) into three factors, as follows:

$$P(\mathcal{X}, \mathcal{R}|\theta, \gamma) = P(X_{cv} \cup X_{on}|\theta) P(X_{pon}|\theta) P(\mathcal{R}|\gamma) \tag{6.7}$$

The first factor is the conditional probability of generating the observed data $X_{cv} \cup X_{on}$ given the labels and the model parameters. The second factor is the conditional probability of observing the "partially on" points at the boundaries of the Gaussian clusters (deduced from the model θ). The third factor assigns a higher probability to the assignments γ that do not violate the observed relationships $\mathcal{R}$. More precisely, the first factor uses the common independence assumption for density estimation:

$$P(X_{cv} \cup X_{on}|\theta) = \prod_{x_i \in X_{cv} \cup X_{on}} P_{mix}(x_i|\theta) \tag{6.8}$$

The second factor $P(X_{pon}|\theta)$ is factorized over the labels as:

$$\prod_{l=1\ldots K} P(X^l_{pon}|\theta) = \prod_{l=1\ldots K} \frac{1}{Z} \exp\left(-\sum_{x_i \in X^l_{pon}} d^2(x_i, \partial S_l)\right) \tag{6.9}$$

where Z is a normalizer and $d^2(x_i, \partial S_l)$ measures the distance between a point x_i being "partially on" the object label l and the boundary ∂S_l of the l^{th} cluster as defined in the previous section (see **Stopping the growth** paragraph).

The third factor is currently defined as a 0-1 value (although it could be devised in a more elaborated way). In our current implementation we only manipulate soft assignments that respect, by construction, the observed relationships $\mathcal{R}$. As explained before (see **Initialization** paragraph), by forcing, for each label l, the assignments $\widehat{p}(l|i) = P(y_i = l|x_i) = 1 \; \forall x^l_i \in X^l_{on}$ ("on" points are assigned) as well as $\widehat{p}(l|i) = P(y_i = l|x_i) = 0 \; \forall x^l_i \notin \mathcal{BB}^l$ (labels cannot be found out of bounding boxes) and $\widehat{p}(K+1|i) = P(y_i = K+1|x_i \in X_{cv}) = 1 \; \forall x_i \notin \cup_{l=1,K} \mathcal{BB}^l$ (impossible labelings are set) we get, by definition, $P(\mathcal{R}|\gamma) = 1$.

The EM algorithm is then implemented as follows:

1. **Initialization**. $k \leftarrow 0$

   - Start initializing the mixture $\theta^{(k)}$ and the initial assignments $\gamma^{(k)}$ as stated earlier (see **Initialization** in section 6.2.2)

- Initialize the mixing proportions

  $n \leftarrow \text{Card}(\mathcal{X})$ and $\widehat{n_l}^{(k)} \leftarrow \sum_{i=1}^{n} \widehat{p}(l|i)^{(k)} \quad \forall l = 1, K$

  $n_{no-label} \leftarrow \text{Card}\{x_i \in \mathcal{X} | \widehat{p}(K+1|i)^{(k)} = 1\}$

  $\pi_{K+1}^{(k)} \leftarrow max(\varepsilon, n_{no-label}/n))$ with a small $\varepsilon \approx 0.05$

  $\forall l = 1, K \quad \widehat{\pi_l}^{(k)} \leftarrow (1 - \pi_{K+1}^{(k)}) \frac{\widehat{n_l}^{(k)}}{\sum_{l=1}^{K} \widehat{n_l}^{(k)}}$

2. **Repeat**.

   $k \leftarrow k + 1$

   (E) Evaluate the posterior assignment probabilities $\gamma^{(k)}$ based on the current model setting $\theta^{(k-1)}$, $\forall x_i \in \mathcal{X}$, $\forall l \in 1 \dots K+1$

   $\widehat{p}(l|i)^{(k)} \leftarrow P(y_i = l | x_i, \theta^{(k-1)}) = \ell(x_i)/P_{mix}(x_i|\theta^{(k-1)})$ with

   $\ell(x_i) = \widehat{\pi_l}^{(k-1)} \mathcal{N}(x_i | \mu_l^{(k-1)}, \Sigma_l^{(k-1)})$ if $l \in 1, K$ and

   $\ell(x_i) = \widehat{\pi_{K+1}}^{(k-1)})/v$ if $l = K+1$

   where the normalizer v is $v = Area(\Omega \setminus \cup_{l=1}^{K} S_l)$ [1].

   Update $\pi_{K+1}^{(k)}, \widehat{n_l}^{(k)}$ and then $\widehat{\pi_l}^{(k)}$ consistently.

   (M) Update the model $\theta^{(k)}$ given $\gamma^{(k)}$:

   The baseline approach consists in using a gradient descent for each Gaussian cluster l given the current estimates $\widehat{\mu_l}^{(k-1)}$ and $\widehat{\Sigma_l}^{(k-1)}$:

   $$\widehat{\mu_l}^{(k)}, \widehat{\Sigma_l}^{(k)} = \arg\max \mu_l, \Sigma_l P(\mathcal{X}, \mathcal{R} | \mu_l, \Sigma_l, \gamma^{(k)}) \tag{6.10}$$

   We use a simpler approach. In standard GMM estimation by EM we could update the model directly for each label l as:
   $\widetilde{\mu_l}^{(k)} \leftarrow \frac{1}{\widehat{n_l}^{(k)}} \sum_{i=1}^{n} \widehat{p}(l|i)^{(k)} x_i$
   $\widetilde{\Sigma_l}^{(k)} \leftarrow \frac{1}{\widehat{n_l}^{(k)}} \sum_{i=1}^{n} \widehat{p}(l|i)^{(k)} (x_i - \widetilde{\mu_l}^{(k)})(x_i - \widetilde{\mu_l}^{(k)})^T$
   However these updates (done for all l) may only maximize part of our likelihood (approximately the first factor) and may be not satisfying (especially when we stop the cluster growths thanks to the second factor). In practice, instead of a gradient descent, we look for the best cluster location between $\widehat{\mu_l}^{(k-1)}$ and $\widetilde{\mu_l}^{(k)}$ and do similarly for the covariance matrices. For each label l, we optimize $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ such that $\mu_l^{\alpha} = \alpha \widehat{\mu_l}^{(k-1)} + (1-\alpha)\widetilde{\mu_l}^{(k)}$ and $\Sigma_l^{\beta} = \beta \widehat{\Sigma_l}^{(k-1)} + (1-\beta)\widetilde{\Sigma_l}^{(k)}$ actually improve the likelihood:

   $$\alpha^*, \beta^* = \arg\max \alpha, \beta P(\mathcal{X}, \mathcal{R} | \mu_l^{\alpha}, \Sigma_l^{\beta}, \gamma^{(k)}) \tag{6.11}$$

[1] It formally means that the dummy (no-label) uniform density is $x \mapsto \mathbb{1}_{\Omega \setminus \cup_l S_l}(x)/v$ and that $\widehat{p}(K+1|i)^{(k)} = 0$ as soon as x_i is in any cluster support S_l

$\theta^{(k)}$ is updated with K new clusters locations $\widehat{\mu_l}^{(k)} \leftarrow \mu_l^{\alpha^*}$ and covariances $\widehat{\Sigma_l}^{(k)} \leftarrow \Sigma_l^{\beta^*}$, produced by K optimizations (equation 6.11 for $l = 1, ., K$) that make the likelihood monotonically increasing.

Until $P(\mathcal{X}, \mathcal{R}|\theta^{(k)}, \gamma^{(k)})$ **converge**.

6.3 Experiments

6.3.1 Protocol

In this section, we explain how we collected game logs. We used the same pool of images than on the previous chapter (section 5.3), i.e. 101 images from both the Berkeley dataset and from the PASCAL dataset.

Players have to watch a tutorial video before playing for the first time. The tutorial only explains how the game is played, nothing being said about the use of the players' traces.

Because the game is not widely known yet, we launched 2 successive campaigns of experiments. For each campaign we indicated to potential players synchronized times during which they could be connected, so that there would be enough players to be paired up. Every player was aware of the following guidelines.

Guidelines. You will start each game adopting one of the two possible roles: the master or the seeker.

If you are the master: your role is to hide a target anywhere inside the image that you will see on the screen. After that, the seeker will ask for clues related to the objects on the image, and you will be asked to provide information about the position of the target related to the object. The possibilities are: on the left of, on the right of, below, above, partially on, on, or not related to.

If you are the seeker: your role is to find the target that the master has hidden as soon as possible. In order to do so, you have to ask for clues to the master following this procedure:

You will be asked to type the name of one of the objects in the image, which will be sent to the master. The master will reply with a hint indicating what is the position of the hidden target related to the object that you chose. After this, you can try to guess the position of the target by clicking on the image, following the indications of the master. If you find it, the game finishes. If you miss, you ask for another clue.

The game goes on until either the seeker finds the target or the time runs out.

Ask'nSeek is a cooperative game; this means that both the master and the seeker have the same goal: locating the target as fast as possible.

Numbers: A total number of 3,250 games have been started during the 2 campaigns of experiments. Among these games, there are 240 games that can not be included in the results, because one of the players left the game before the end of the game. A total of 162 users have played at least 5 games.

114 out of the remaining 3,010 games finished without the hidden region being found by the seeker. This means that 96% of the games ended ud with a victory of the players. The mean duration of a game that has been won is 29 seconds. Adding up the games that went until the end of the timer (the losses), the average duration of a game is about 33 seconds.

We collected more than 5,000 seeker clicks for a total of 9,063 indications. Indeed, a click can hold more than one indication. For example in figure 6.1, the winning click from the seeker is at the same time "on the right of the dog", "on the cat" and "on the cat's head". This particular click brings more than just one indication.

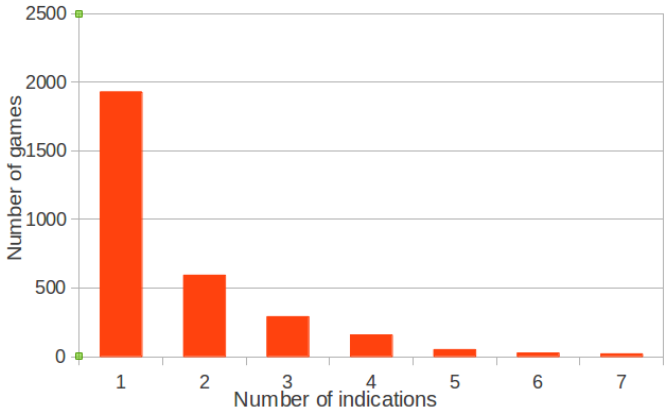


Figure 6.10: Distribution of the number of indications per game.

Figure 6.10 shows the distribution of the number of indications per game. A large number of the games played (almost two thirds) only had one indication. In average there was 1.6 indications per game (we will comment these numbers more in chapter 7). This is understandable, as the best strategy to win is to place the hidden region on the most salient part of an image. For example, as shown on figures 6.4 and 6.12, it is very intuitive to place a region on the head of any animal or human present on an image. In the case there are multiple animals, like the cat and the dog for example, then it usually takes more than only one indication to find the region.

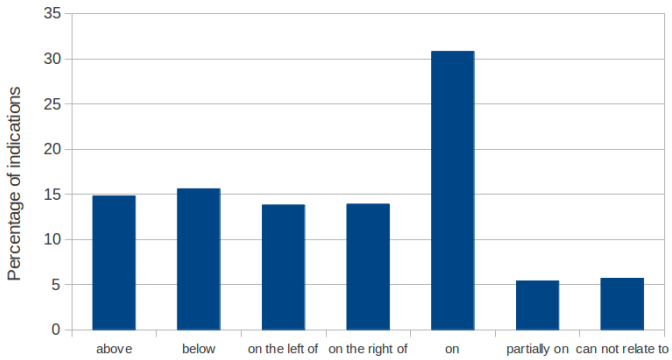


Figure 6.11: Occurences of spatial relations.

Figure 6.11 shows the distribution of the spatial relations indicated by the master to the seeker. It is interesting to note that the *on* spatial relation is by far the most used. This is a direct consequence of the fact presented above : the master tends to place the region on the most salient object. Because the seeker also tends to ask a clue relative to the most salient object, the *on* spatial relation comes very often. The second interesting aspect to note is that the *above*, *below*, *on the left of*, and *on the right of* are almost equally distributed. This probably means that the position of the salient objects in the dataset is equally distributed on the images.

Figure 6.12: Location of the hidden regions on the cat and dog image.

6.3.2 Results

In order to test our detection model, we manually process the textual traces. The natural language processing algorithms that would be necessary to perform this step automatically are beyond the scope of this thesis. In addition, we test our detection algorithm on images for which the object is designated by a somtimes complicated description. The description can be as simple as "hat" and as complicated "topmost fish on the center-right of the image". Finally by manually processing the textual labels, we can evaluate the effectiveness of our model only.

In order to manually annotate the labels, we categorized each label relatively to the target object as:

- *Object*: when the label designates the target object.
- *Part*: when the label designates a part of the target object.
- *Other*: when the label designates a different object than the target.
- *Ambiguous*: when the label is not precise or clear enough to be considered a member of one of the three previous categories.

For our detection algorithm, we use mainly the points categorized as *Object*. We also include the *On* points from the *Part* category: points that are on an object's part are obviously on the object itself.

In order to build X_{cv}, we use a saliency map as computed by [Itti 1998] as well as a map that characterizes the probability of the object's presence on the image, based on the users clicks. We describe how we build these maps in section 7.2.2. In a nutshell, these maps are a simple manner to introduce content analysis techniques (in this case, superpixels) to expand the information brought by players clicks based on colorimetric cues.

Finally we evaluate our model using a criteria defined in the PASCAL Challenge. We say an object is detected if the Jaccard Index between the bounding box obtained from our model and the Ground Truth bounding box is superior to 0.5.

With this definition our algorithm detects 58% of the objects. The average Recall (i.e., the percentage of the object that falls into the bounding box) of our detection algorithm is 0.83. The Precision is not applicable in our case, as we always generate only one bounding box for each object.

Figure 6.13: Results of our detection algorithm (green bounding box). Green points are clicks on the object, red points are above the object, blue points are partially on the object and yellow points are on the right of the object.

Figure 6.13 shows examples for which our algorithm works well. The red bounding box is built from the *above*, *below*, *left* and *right* points. We can observe in these two examples that there are no (or very few) errors in the traces which leads to very good results.

Figure 6.13 shows examples for which our algorithm fails to detect the object. On the left, the lady bug is detected (with a Recall of 1) but the bounding box is too big compared to the actual size of the bug. The resulting jaccard index is therefore inferior to 0.5 which means (in the PASCAL sense) that the object is not detected. On the right image, the object that should have been detected is the right zebra. Unfortunately, most players clicks are concentrated on the two others zebras. The two only clicks related to the right zebra are incorrect, which results in an incorrect detection. This scenario tends to happen when multiple instances of the same object are depicted together on the image.

These figures have been obtained on a particular dataset that was designed to benchmark segmentation algorithms [McGuinness 2010]. In particular we have the ground truth for

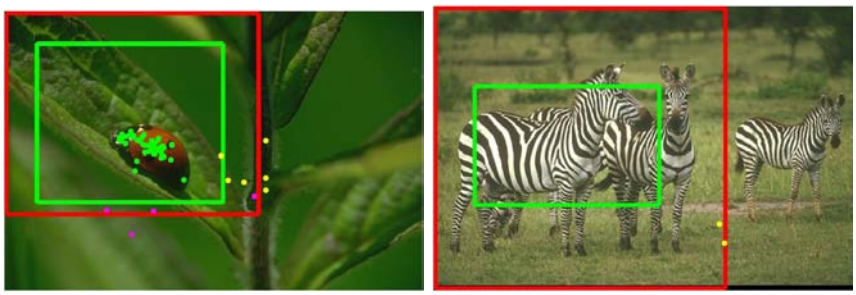


Figure 6.14: Results of our detection algorithm (green bounding box). Green points are clicks on the object, magenta points are below the object and yellow points are on the right of the object.

only one object per image, which does not allow us to test our model in the case of multiple objects.

Figure 6.15: Results of our detection algorithm on images from the PASCAL VOC dataset.

Figure 6.15 presents the results of our detection algorithm on two images from the PASCAL VOC dataset ([Everingham 2010]). These results (published in [Carlier 2012]) show that our algorithm can obtain very good results when detecting concurrent objects.

6.4 Summary and Conclusion

In this chapter, we have presented a new Game With A Purpose called Ask'nSeek. This game allows to implicitly collect textual and spatial data from users.

We described an algorithm for object detection and evaluated it on the same dataset than Click'n'Cut. We found that our object detection algorithm performs well, even if on some occasions the users did not provide enough inputs to obtain a satisfying result.

The contributions presented in this chapter have been published in [Carlier 2012]. We have also shown in [Salvador 2013] how to use Ask'nSeek traces to perform figure-ground segmentation, following the same method presented in chapter 5. In the next chapter, we will compare these results to those obtained on Click'n'Cut in order to answer the interesting

question: Is there a loss induced by the gamification of an interface?

Chapter 7

Crowdsourcing and Gamification losses

Contents

In this chapter we compare the results and necessary effort required to obtain acceptable results with the interface/GWAP described in the two previous chapters.

We have shown in chapter 5 that Click'n'Cut is an efficient interface for interactive segmentation. If properly used, it can lead expert users to produce a very good segmentation in a matter of seconds, significantly quicker than state-of-the-art algorithms.

The only problem is that experts are expensive. We can not have expert users annotate thousands of images. Therefore we investigated the ability of non-expert workers (picked on Microworkers.com) to use Click'n'Cut and obtain similar results. Though less effective, the workers are also less expensive than experts.

Going further, can we achieve a good segmentation without paying any workers to do so? How informative are inputs from Ask'nSeek players?

In this chapter we try to answer these questions and explore avenues to reach the ultimate goal of perfect segmentation with (implicit or explicit) crowdsourcing.

7.1 A comparative analysis of the user clicks on Click'n'Cut and Ask'nSeek

In this section we have a closer look at the data we have gathered during our experiments. First, we start with a quick reminder of the three experiments we conducted, and that we will study in this section.

- **Click'n'Cut - Experts**: 15 Computer Vision researchers (Ph.D. students and professors) interacted with the Click'n'Cut interface on the 105 tasks (see section 5.3).
- **Click'n'Cut - Paid Workers**: 20 workers from the platform `Microworkers.com` interacted with the Click'n'Cut interface on the 105 tasks.
- **Ask'nSeek - Players**: 162 players (mostly students) played the Ask'nSeek game on the number of images they wanted to (see section 6.3.1).

7.1.1 Preliminary figures

Table 7.1 presents a comparison of figures for the three experiments. The first main comment is that the workers produced a lot of clicks. In average the workers clicked more that twice as many times as the experts on the same images (they were 20 against 15), and ten times more than the Ask'nSeek players.

	Click'n'Cut	Click'n'Cut	Ask'nSeek
	Experts	Paid workers	Players
# Users	15	20	162
# Clicks (per image, all users included)	234.4 168 FG 66.4 BG	544.6 345.8 FG 198.8 BG	51.4 29 FG 21 BG 1.4 Part. On
# Errors	4%	35%	7%

Table 7.1: Comparison of the number of clicks and error rates in the different setups.

Another interesting difference between the different groups of users is the ratio between foreground (FG) and background (BG) clicks. Expert users on Click'n'Cut mostly produce foreground clicks (72% of the times). Paid workers also use more foreground clicks but the ratio is 63%/37%. Finally Ask'nSeek players tend to produce 57% of foreground clicks.

The most spectacular number is the percentage of errors in the clicks. The percentage of errors is computed as the number of clicks that are badly categorized, i.e. foreground clicks that are in fact on the background and vice versa. We did not consider the *Partially On* clicks in this percentage, as 1) they represent a minority of clicks 2) they are a specificity of Ask'nSeek, therefore not comparable to Click'n'Cut. We have previously emphasized in chapter 5 that we wanted to keep the interface as simple as possible, and that we had not implemented any tutorial for Click'n'Cut. The low error rate on expert traces validates this

choice, but it seems clear that we should have introduced such a tutorial for paid workers: it could have helped reducing their error rate.

Q&A. Two fundamental questions are immediately raised by these numbers.

Why is there so few clicks in Ask'nSeek compared to Click'n'Cut ?

There are several reasons that explain this difference. First, it takes 2 players to produce a click in Ask'nSeek (the master and the seeker) whereas only one user is necessary in Click'n'Cut. Second, the users in Ask'nSeek played an average of 30 games (i.e. images) each whereas Click'n'Cut users performed the entire set of 105 tasks. Also, Click'n'Cut users were given the possibility to do as many clicks as they wanted to. Ask'nSeek players are limited by the 2 minutes timer (which includes the time to type labels, and exchange indications). The game stops when the seeker finds the target, which occurs after an average of 1.6 indications per game. Finally, Click'n'Cut users are focused on one object for each image: the object they have to segment. In Ask'nSeek, the players use all the objects they can see in the image.

In other words it takes two players to play an average of 30 games that usually produce 1.6 clicks each, and the clicks are not even related to the target object for certain!

Why is there so much errors performed by paid workers on Click'n'Cut and so few by Ask'nSeek players?

As we will see in the next paragraph, the high error rate of the paid workers on Click'n'Cut is partly due to a subset of the workers who performed particularly bad. Ask'nSeek players error rate is much more homogeneous. The fact that Ask'nSeek is a game naturally limits the impact of some of the usual sources of errors in crowdsourcing. [Oleson 2011] listed the spammers, the incompetent workers and workers' insufficient attention as the major sources of errors. Being a game, Ask'nSeek is relatively safe from spammers (the game has nothing to offer except for enjoyment; if players do not like the game, they are free to leave). The players' attention is kept at a certain level by the non-repetitiveness of the task. Unlike Click'n'Cut where the task to perform is always the same, Ask'nSeek players regularly switch roles (from master to seeker) and since the players' pairing is random, players interact with different people over time. The major source of errors in Ask'nSeek is the misunderstanding between the master and the seeker. Misunderstanding can arise from an imprecise requested object from the seeker (e.g. "fish" in an image where there are three fishes), or from the master not knowing a word used by the seeker.

7.1.2 A deeper look at clicks and errors in Click'n'Cut

In this paragraph, we take a closer look at user traces on Click'n'Cut to try to understand the **Crowdsourcing loss** on Click'n'Cut.

Figure 7.1 shows an analysis of the expert users' clicks and errors. The numbers are averaged on all tasks. Expert users produce in average from 7 to 26 clicks per image and it is interesting to note that for each expert user, the proportion of foreground/background clicks is fairly similar.

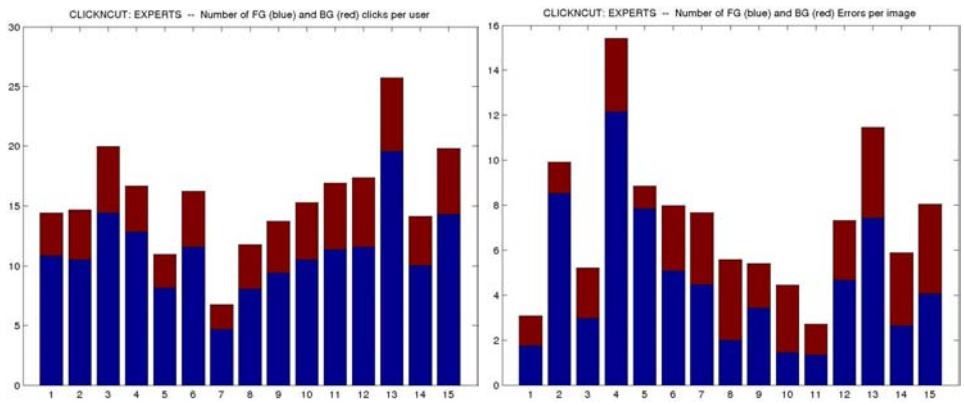


Figure 7.1: Number of foreground/background clicks (left) and percentage of foreground/background errors (right) per expert user on Click'n'Cut

The right of figure 7.1 presents the percentage of errors on foreground clicks (in blue) and on background clicks (in red) that are stacked together. Note that these percentages do not take into account the number of foreground and background clicks, which means that the mean of the two percentages is not equivalent to the total error rate. It is interesting to note that the expert's highest source of errors seems to be the foreground clicks.

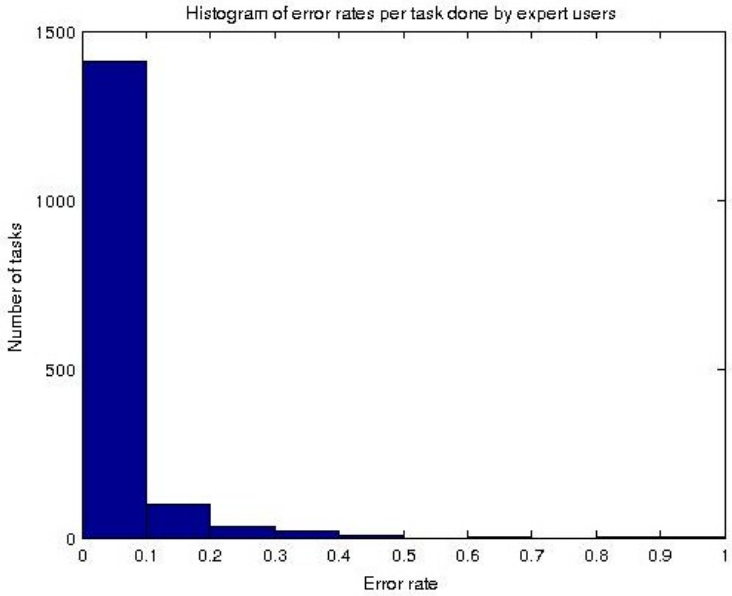


Figure 7.2: Histogram of the distribution of error rates per task.

To further understand this phenomenom, let us consider the following numbers. Expert users have produced 24,611 clicks on $15 * 105 = 1,575$ tasks, and among those clicks there were 1,042 wrong ones. The 10 tasks (out of 1,575) for which the most errors were made

account for a total of 372 errors, i.e. more than one third of the errors. This error rate distribution is visible on figure 7.2, on which we can see that a very large majority of the tasks had a very low error rate.

Figure 7.3: Two of the tasks that produced a lot of errors. Descriptions associated to the tasks are: 'Extract just the man's hat. Do not include the rest of the man or any or any other objects.' (left) and 'Extract the topmost fish on the center-right of the image.' (right)

Figure 7.3 presents two tasks that have created a lot of errors from the experts. On the left, only the man's hat should have been segmented. Two experts segmented the man, which created 100 errors (one tenth of the total number). On the right, the description of the fish to segment ("topmost fish on the center-right") was also misunderstood by two experts.

It is interesting to note that these errors are due to insufficient attention from the experts. Even experts can make mistakes! This suggests we should always have more than one expert performing a task (typically in our traces, there were never more than two experts who misunderstood a task at the same time).

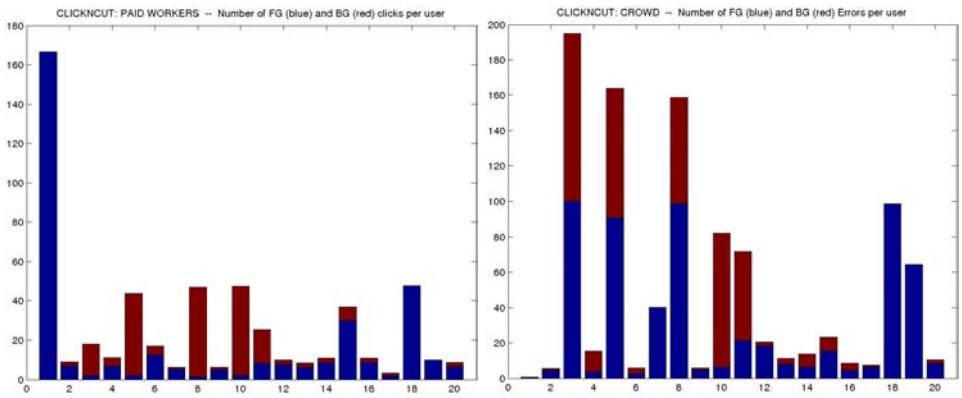


Figure 7.4: Number of foreground/background clicks (left) and percentage of foreground/background errors (right) per paid worker on Click'n'Cut

Figure 7.4 shows the same plot than Figure 7.1 but for paid workers.

The first very obvious fact is that unlike expert users, paid workers have a very het-

erogeneous way of interacting with Click'n'Cut. 5 workers out of 20 produced a majority of background clicks, whereas we previously observed that all expert users clicked a higher number of foreground clicks.

The distribution of the number of clicks is also clearly biaised by one user, who produced an average of 160 clicks per image (only foreground clicks). The right plot of figure 7.4 also shows that this particular user (user # 1) made very few errors. We should be careful with the data from this user since it can affect our results a lot without being statistically significant.

The biggest difference between paid workers and experts is visible on the right of figure 7.4. There are workers who display an exceptional error rate. Here is a tentative description of the different types of paid workers (illustrated in figure 7.5):

- Worker # 1, a.k.a. "The painter" produced only foreground clicks, with an exceptional amount of clicks and an error rate almost equal to 0%. In fact we suspect that this user misunderstood the interface and believed he had to paint the object with green clicks.

- Workers # 3 and 5, a.k.a. "The mirrors", have such a high error rate that by inverting their contributions (considering their background clicks as foreground, and vice versa), they would actually display a very low error rate! We can only assume that they misunderstood the task as well, mixing up foreground and background clicks.

- Worker # 8 and 10, a.k.a. "The border guards" produced almost exclusively background clicks located on the border of the objects.

- Worker # 18, a.k.a. "The surrounder" produced only foreground clicks, and has almost 100% errors. He tried to surrounder the object with foreground clicks, which would have produced a good result on LabelMe [Russell 2008].

- Worker # 19, a.k.a. "The spammer", randomly placed foreground clicks over the image so that he would get paid. This worker did the entire set of tasks in less than 5 minutes, whereas it takes from 30 to 60 minutes to a serious user.

- Remaining workers, a.k.a. "The experts", only placed a few well-positioned clicks, and made a few mistakes due to insufficient attention. These workers exhibit statistics (in number of clicks and error rate) that are comparable to expert users.

The main lesson of this decomposition is that except for worker # 19 who was just a spammer, the highest number of errors come from users who did not understand the job properly. This could have been avoided, or at least limited, with a proper tutorial on gold standard images that would have taught workers what is a good click and what is a bad click. Nevertheless we must use the data we collected, and we show in the next paragraph how to limit the impact of errors.

Figure 7.5: 6 types of paid workers: "The painter", "The mirror", "The border guard", "The surrounder", "The spammer" and "The expert".

7.1.3 Filtering errors

First, we will not focus too much on filtering errors from the experts, since they do not significantly affect the final results.

We start by studying the errors from paid workers. The figures we introduced in the previous paragraph were computed on the entire set of images. We presented these figures to help the reader understand the nature of the data we are dealing with, but we can not use the knowledge we have acquired (for example the classification of the workers) to process the data.

The only data we can use to filter workers are the traces on the gold standard images, i.e., the 5 PASCAL images we have introduced to serve as a control dataset. Figure 7.6 displays the error rate per user on the gold standard dataset (in blue) and on the test dataset (in red).

The good news is there is an obvious correlation between error rates on the gold standard and on the test set. Of course the correlation is not perfect; for example worker # 7 made no mistakes on the gold standard set, but on the contrary made more than 30% of mistakes on the test set.

Nevertheless, we can filter a decent amount of errors by just removing the workers that are above a threshold of error rate on the gold standard. In figure 7.7, we vary the threshold that we use to filter users based on the gold standard images. The blue curve (resp. red curve) represents the error rate of the remaining users on the gold standard (resp. on the test set).

In the next section we will therefore test 2 thresholds based on this graph. First we will keep only the users who did less than 50% errors on the gold standard: that makes the total error rate less around 20% on the gold standard. We will also try the smaller threshold of 20%, which makes the total error rate less than 10% on the gold standard (and on the test

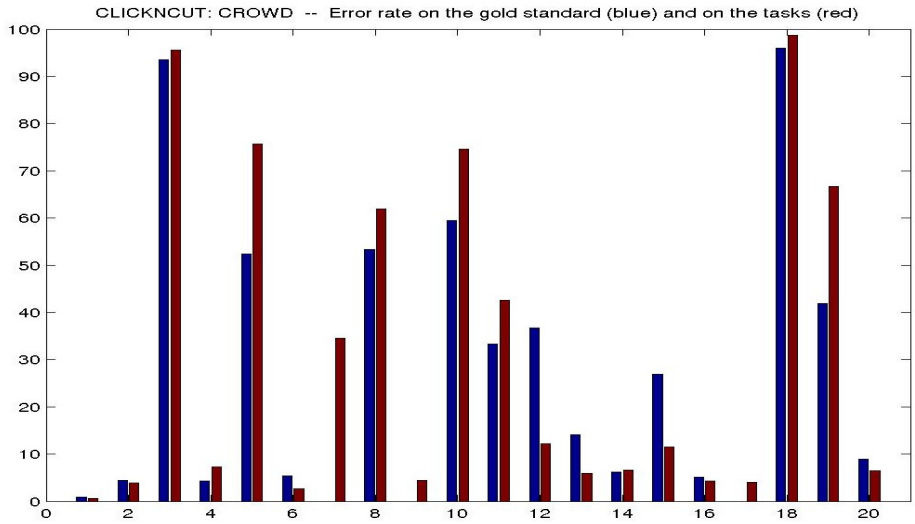


Figure 7.6: Error rate on the gold standard image (in blue) and on the tasks (in red) for each worker

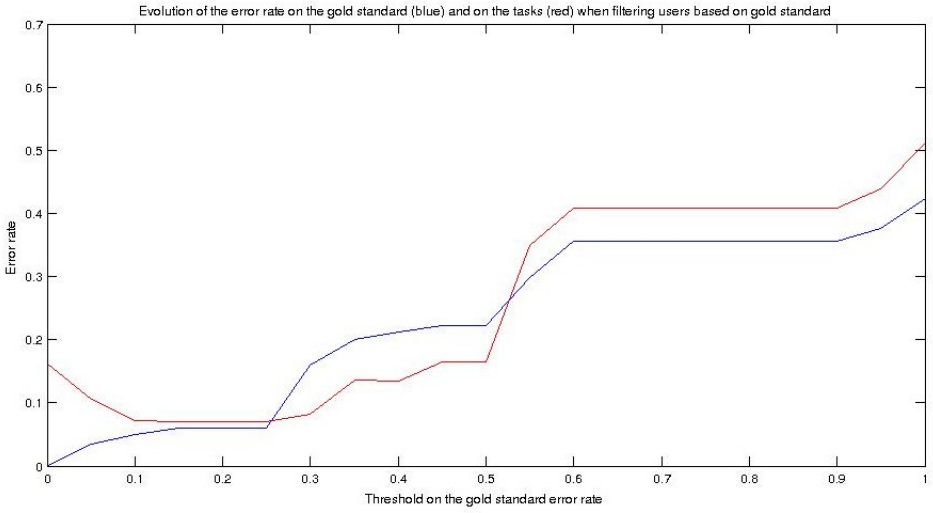


Figure 7.7: Evolution of the overall error rate on gold standard images (in blue) and on the tasks (in red) when filtering users based on a threshold on the gold standard error rate.

data as well).

7.2 Segmentation results

In this section, we comment the results on the figure-ground segmentation task for the three groups of users: the experts (who serve as a baseline, introduced in chapter 5), the paid workers and the players (on Ask'nSeek). We are particularly interested in comparing the

paid workers performance against the experts, in order to estimate a "crowdsourcing loss". In the next section we will elaborate more on a "gamification loss" that could explain the performance difference between the Ask'nSeek players and the Click'n'Cut experts.

7.2.1 Analysis of the results

	Click'n'Cut Experts	Click'n'Cut Paid workers	Ask'nSeek Players
All users	0.90	0.14	**0.44**
Users with less than 50% errors on GS	0.90	0.63	0.43
Users with less than 20% errors on GS	0.90	**0.82**	0.40

Table 7.2: Average Jaccard Index on the test dataset in the three experiments.

Table 7.2 presents the results on the three experiments: Click'n'Cut with experts, Click'n'Cut with paid workers, and finally Ask'nSeek. We use the Jaccard index as a measure of the segmentation precision (the Jaccard Index is defined as $J = \frac{P \cap GT}{P \cup GT}$ between the Predicted (P) and Ground Truth (GT) masks).

The experts results are computed for each expert separately, and then averaged over all experts. The results thus mean that one expert will obtain an average of 0.90 jaccard on each task. Note that filtering experts based on their gold standard performances does not make a lot of sense, since all experts have an error rate below 10% on the gold standard.

There are many things to be said about these numbers. First and not surprisingly, the experts obtain the best segmentation score. This is understandable because they are fully aware that they are performing segmentation (unlike Ask'nSeek players), and they already know what is a good segmentation and what are the main difficulties to obtain it. In other words, their experience help them to focus on more meaningful regions to click on than paid workers for example.

Paid workers' results are very dependent on the filtering based on the gold standard images. The results range from 0.14 without filtering (which is very bad) to 0.82 when keeping users with a low error rate on gold standard images, which is a pretty good result!

Finally, Ask'nSeek results are very low compared to Click'n'Cut experiments. We will elaborate more on Ask'nSeek results in section 7.3

7.2.2 Improving the paid workers results on Click'n'Cut

In this paragraph, we try to improve the results from paid workers without filtering their contributions based on the gold standard.

The figures shown on table 7.2 establishes that the method described in section 5.1 for computing the best mask is not robust enough to noisy clicks to produce optimal quality results. In addition, filtering workers based on their gold standard error rate does not take advantage from users like "the mirrors" described in section 7.1. It would be nice to be able to automatically invert this data and use it as the other good users.

To test this idea, we have implemented a very simple algorithm. Since we have a lot of clicks from the workers, we can try to paint a superpixel segmentation with these clicks. At that point, it seems a better idea to use superpixels than object candidates (like MCG candidates that we have used until now in our algorithm) because we have a lot of clicks. We do not need to propagate the users annotation to a lot of neighbouring pixels, therefore superpixels are a better choice than candidates since they are usually of smaller size: this will limit the impact of bad clicks. Each superpixel is labelled with a number between 0 (background) and 1 (foreground). To compute these labels we leverage each worker contribution by a measure of the worker's confidence, based on this worker's performance on the gold standard images. For example, if a worker w has a 5% error rate on the gold standard images (see figure 7.6), the measure of confidence c_w for this worker will be 0.95. A foreground (resp. background) click brings a contribution to the superpixel of c_w (resp. $1 - c_w$).

To limit the influence of the superpixel segmentation, we perform the computation on several different superpixel segmentations and average the respective results. In our implementation we used Felzenszwalb's algorithm [Felzenszwalb 2004] with different parameters (k varying from 100 to 500).

Figure 7.8: Probability map of object's presence, based on workers clicks.

Figure 7.8 illustrates the performance of this simple algorithm. The object to be segmented is the brightest region, and we can see traces from noisy clicks when regions in the background are bright as well.

Thresholding these maps (any value between 0.6 and 0.7 gives good results) and performing a simple hole filling algorithm allows obtaining a final Jaccard Index of 0.83 for the paid workers. The gain is slight but it is more elegant to handle all users identically, and not just discard some.

7.2.3 The crowdsourcing loss

The results introduced earlier reveal several facts that could be characterized as crowdsourcing loss, i.e. a loss induced by having a task performed by paid workers instead of

experts.

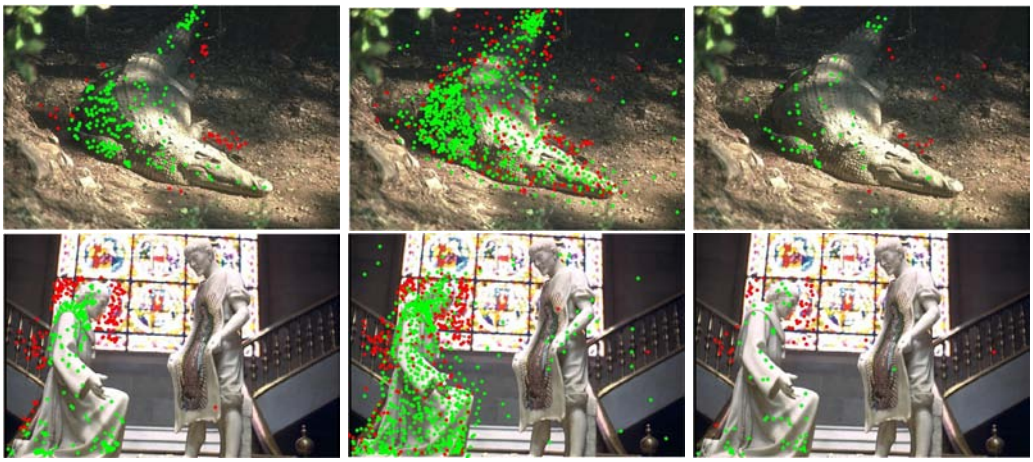


Figure 7.9: Clicks distribution (foreground in green, background in red) from the experts (left), from the paid workers (middle), and from the paid workers categorized as experts in section 7.1.2.

First, the data is obviously more noisy with paid workers than with experts. This can be visualized in figure 7.9: green and red points are mingled in the central column and well separated on the left column. This is caused by many factors: spammers, workers who do not understand the task, a lower attention level, etc. It only reinforces the message from the related work section: experiments should be carefully designed, there should be several ways to detect errors (for example we can detect spammers both using gold standard images and an analysis of the time spent to annotate images), etc.

Second, the best workers from the crowd are still less performant than experts. In our results section we have presented the results for a filtered crowd (also shown on the right column of figure 7.9), which altogether performs less good than the average expert. There are of course several reasons for that (noisy data, lack of knowledge in segmentation) but this is a fact. In addition, filtering the crowd also leads to a loss of information. Figure 7.9 shows that there are areas in which users (expert and crowd) tend to click more. In our experience, these high density areas are located on regions where the MCG object candidates fail to follow the object boundaries. As such, high density areas convey a lot of information: they can help improving the MCG object candidates for example. Filtering users, as we can see on the figure, makes these high density areas disappear.

Nevertheless, there is something to be said about a crowdsourcing gain. We do not exploit this very well for the moment, but something that is very clear from the classification of paid workers is the diversity and the richness of the clicks collected on our experiments. We have clicks that completely paint the object, we have clicks on the object's boundaries, we have clicks surrounding the object, etc. This information produced by paid workers is much richer that the one obtained from experts.

7.3 Understanding the gamification loss

In this section we try to understand why the Ask'nSeek results in segmentation are so poor compared to the results obtained through the use of Click'n'Cut.

First it is important to note that the results are not so poor. An average Jaccard Index of 0.44 is already better than the top performers on the PASCAL segmentation challenge, who consider only 20 classes of objects. In the dataset we have considered, the objects are more complex and therefore more adapted to evaluate interactive segmentation. Since the interaction is completely implicit to the users, it is natural that the results are less good on Ask'nSeek than on Click'n'Cur.

A first very simple reason for Ask'nSeek's performance is the number of clicks gathered through the game. We have already commented on this matter in section 7.1 so we will not say much more about it in this section. However it is important to state here that one of the limitations of our current approach is the difficulty of text processing. Even by doing it manually there are many labels that remain hard to categorize, either because they are not precise enough or because they are not understandable enough. It would be interesting to study simple ways (e.g., autocompletion, limited vocabulary, and so forth) to make the natural language processing more straightforward.

Figure 7.10: Clicks distribution (foreground in green, background in red) from the Ask'nSeek players.

The second reason of Ask'nSeek's poor performances is the distribution of the seekers' clicks. Figure 7.10 introduces the foreground and background clicks gathered through Ask'nSeek on two difficult images in our dataset. On the left image, the entire soldier that stands on the right should be segmented. We can see that all the foreground clicks are focused on the soldier's head. We can make a similar observation on the right image: the foreground clicks are concentrated into the duck's head whereas the entire duck should be segmented in our case.

What is even more interesting is the distribution of the background clicks. We can see that on Ask'nSeek traces, the background clicks are mostly located on other objects which is understandable: the seekers predict that the target region is often located on a salient object, so their clicks are focused on objects (other soldiers, or other ducks in figure

7.10). When we look at figure 7.9, we can see that background points obtained through Click'n'Cut are almost always located near the object's boundaries. This is probably the key reason that explains the performance loss in Ask'nSeek.

What is interesting is that figure 6.8, which presents the clicks distribution on an image that has been played 99 times, shows that after a high number of games the clicks are spread on the entire object. This suggests that having images played a lot could asymptotically improve the performance of Ask'nSeek traces in segmentation. We have simulated this phenomenon in [Salvador 2013] and obtained a higher average Jaccard Index when augmenting the number of games per image.

In other words, the gamification loss is a direct consequence of the nature of the game itself: the players know that the most efficient stategy to win in Ask'nSeek is to place the target region on an object, and preferably on a salient part of the object (e.g., human's face). But then how can we limit the impact of such behaviour?

The answer is simple: we need to modify the gameplay so that the players will not be always biased towards the same regions of an image. One possibility is to bias the master towards placing the target regions in different areas of the image. Based on the previous games' target position (and some content analysis to determine, for example, which position could bring the highest amount of information), we could suggest (or forbid) some areas to the master. Modifying the scoring system would also be appropppriate to further encourage the diversity of the regions locations. These ideas are clearly following the trend that we have discussed in the related work chapter at section 2.2.4: it falls into the active learning paradigm.

7.4 Summary and conclusion

In this chapter we have compared the segmentation results in three different setups: Click'n'Cut used by experts, Click'n'Cut used by a crowd of paid workers and Ask'nSeek played by many users. Not surprisingly, the experts who use Click'n'Cut exhibit the best results. The crowd of paid workers produce very noisy inputs, but we have shown how a simple filtering method based on gold standard images can bring acceptable results. Finally, results obtained through Ask'nSeek are poor and significantly worse than the results obtained through Click'n'Cut.

In addition we have tried to define the loss induced by having a crowd of paid workers perform a segmentation (by comparison to experts). We have seen that the paid workers are less efficient than expert users in positioning their clicks in meaningful areas. However, this loss could be compensated by a better of use of the diversity of workers' profiles that actually produce a high amount of clicks that are wrong, but that are still informative.

Conversely, Ask'nSeek clicks are not very informative because they are very redundant. Due to the nature of the games, players are biased towards positioning their clicks on objects which is not the best strategy for our segmentation algorithm. The challenge we will have to

face in future work will be to modify the gameplay in order to encourage a higher diversity of clicks' positions. We should emphasize however that Ask'nSeek performances should not be judged only on the particular setup that we have introduced in this chapter. The traces collected through Ask'nSeek contain rich information for harder tasks like image parsing since we collect spatial and textual information about many objects of an image at the same time (see figure 6.15). The work that has been presented in this chapter is still in progress, and is in fact only partially published. There are many avenues to explore that we will detail in the conclusion of this thesis.

Chapter 8

Conclusion

8.1 Contributions

In this thesis we have presented three main contributions.

First we conducted a detailed review of the state of the art in computer vision and multimedia, in which we explained how in the last ten years a general trend shifted the traditional "all automatic" paradigm in visual content understanding and placed the human at the center of algorithms. We therefore studied both how human input could benefit to content analysis algorithms, as well as how content analysis could assist humans in producing valuable inputs. This survey first allows to position our twofold contribution in the literature: crowdsourcing user interest (Part I) and crowd-based semantic image segmentation (Part II). It also identifies main emerging avenues for further developments of humans-in-the-loop approaches including the most formal ones (e.g., active learning).

In part I, we have then presented how to compute user interest maps for a video, and a set of simultaneous video recordings. We explained how humans, when interacting with visual content on particular interfaces, can implicitly reveal the regions of interest of the visual content. We also showed that user inputs could be combined with content analysis in order to obtain better interest maps. For example, we showed how zooming on a video (through a zoomable video player) can implicitly designate the zoomed region as more interesting than the rest of the frame. We have then listed and detailed a few applications of user interest maps, such as video retargeting or video mashups. Some of the interfaces that we have developed for this work are available on my home page. The basic zoomable video player (section 3.1) is available on `carlier.perso.enseeiht.fr/demos/zoomable-video/v1.2/coffeelounge.html`. Then the zoomable video player with recommended viewports can be tried on `carlier.perso.enseeiht.fr/demos/zoomable-video/v1.1/coffeelounge.html`. I have also contributed to including ROI detection software into an existing video mashup system (Jiku Director, `http://liubei.ddns.comp.nus.edu.sg/jiku/jiku-director.html`, to appear in the technical demos of ACM MM'14). The interfaces have been designed in HTML5 and javascript.

Finally, we studied in the second part of this thesis problems that require a higher level of semantics: object detection and figure-ground segmentation. We introduced a new interface, called Click'n'Cut, to perform interactive segmentation. This interface takes points (i.e. clicks) as user inputs and allows, thanks to candidates provided by content

analysis, to converge towards a very good segmentation result in a matter of seconds. We have then described a Game With A Purpose called Ask'nSeek which produces traces that can virtually be used for all image understanding problems. We studied how the Ask'nSeek traces can be used for object detection as well as object segmentation. Finally we have compared performances of a crowd of paid workers using Click'n'Cut, against a crowd of gamers playing Ask'nSeek for the task of object segmentation. In that sense, we evaluated both a crowdsourcing loss and a gamification loss by comparison with our baseline (the semantic segmentation of images by experts). We found that even if a game allows to gather traces for free, the very unique nature of the game biases the traces. As a consequence, it is necessary to include adaptable gameplay mechanisms to push players to provide original and valuable information, which is closely related to the active learning paradigm.

Current versions of Ask'nSeek (`carlier.perso.enseeiht.fr/demos/askandseek.html`) and Click'n'Cut (`carlier.perso.enseeiht.fr/demos/clickncut.html`) are also accessible online. There is also a beta version of Ask'nSeek for Android platforms available on Google Play (`https://play.google.com/store/apps/details?id=fr.enseeiht.ubee.askandseek`).

The results presented in this thesis have almost all been obtained thanks to user studies. In a way this thesis itself can be seen as the result of an intensive 3 years crowdsourcing campaign.

We have enrolled 70 participants in studies about our original zoomable video player, introduced in section 3.1 [Carlier 2010b, Carlier 2010a, Quang Minh Khiem 2010]. 48 additional users participated to the evaluation of the retargeting algorithm presented in section 4.1. We have also recruited 70 users to evaluate our zoomable video player with recommended viewports (section 3.2, [Carlier 2011a, Carlier 2011b]). In addition, there were 75 users who evaluated the mashup video built thanks to 3D interest maps (section 4.2, to appear in ACM Multimedia 2014) and 30 more to perform the preliminary evaluation of the query system introduced in section 4.3 (unpublished yet).

As for the Click'n'Cut and Ask'nSeek interfaces, there were 40 players who tried the original version of Ask'nSeek ([Carlier 2012]) followed by 50 players who tried the second version of the game ([Salvador 2013]). Finally, 162 players have tested the current version of Ask'nSeek (section 6.3.1). We have released the traces gathered during these studies to the french community (to appear in CORESA 2014). As introduced in section 5.3 and 7.1, there were also 35 users who segmented objects using Click'n'Cut (to appear in CrowdMM 2014).

That makes a total of 580 users who participated in the user studies that allowed us to evaluate our work through this thesis.

8.2 Publications and other work

Most of these contributions have been already published. The publications related to this thesis are as follows:

- A. Carlier, L. Calvet, D. T. Dung, W.T. Ooi, P. Gurdjos, V. Charvillat: 3D interest maps from simultaneous video recordings to appear in ACM MM'2014
- D. T. Dung, A. Carlier, W.T. Ooi, V. Charvillat Jiku Director 2.0: A Mobile Video Mashup System with Zoom and Pan Using Motion Maps to appear as a ACM MM'2014 demo paper
- A. Carlier, A. Salvador, X. Giro i Nieto, O. Marques, V. Charvillat: Click'n'Cut: Crowdsourced Interactive Segmentation with Object Candidates to appear in CrowdMM'2014
- M. Riegler, M. Lux, V. Charvillat, A. Carlier, R. Vliegendhart, M. Larson: VideoJot: A Multifunctional Video Annotation Tool ICMR'2014
- S. Zhao, W.T. Ooi, A. Carlier, G. Morin, V. Charvillat: Bandwidth Adaptation for 3D Mesh Preview Streaming TOMCCAP 10 (1s)
- A. Carlier, V. Charvillat Un jeu, des images, des clics et du texte : collecte implicite de données visuelles et sémantiques to appear in CORESA'2014 (french multimedia conference)
- S. Zhao, W.T. Ooi, A. Carlier, G. Morin, V. Charvillat: 3D Mesh Preview Streaming MMSys'2013, 178-189
- A. Salvador, A. Carlier, X. Giro i Nieto, O. Marques, V. Charvillat : Crowdsourced Object Segmentation with a Game CrowdMM'2013
- W.T. Ooi, O. Marques, V. Charvillat, A. Carlier : Pushing the Envelope: Solving Hard Multimedia Problems with Crowdsourcing IEEE COMSOC MMTC E-Letter, January 2013.
- A. Carlier, V. Charvillat, O. Marques: Ask'nSeek, A New Game for Object Detection and Labeling ECCV Workshops(1) 2012, 249-258
- T.P. Nghiem, A. Carlier, G. Morin, V. Charvillat: Enhancing online 3D products through crowdsourcing CrowdMM'2012, 47-52
- A. Carlier, R. Guntur, V. Charvillat, W.T. Ooi: Combining Content-based Analysis and Crowdsourcing to Improve User Interaction with Zoomable Video ACM MM'2011, 43-52
- A. Carlier, A. Shafiei, J. Badie, S. Bensiali, W.T. Ooi: COZI: Crowdsourced and Content-based Zoomable Video Player ACM MM'11, 829-830

- A. Carlier, V. Charvillat: A propos d'interactions qui permettent d'analyser une vidéo ORASIS'2011 (french computer vision workshop)
- A. Carlier, V. Charvillat, W.T. Ooi, R. Grigoras et G. Morin: Crowdsourced Automatic Zoom and Scroll for Video Retargeting ACM MM'2010, 201-210
- A. Carlier, R. Guntur, W.T. Ooi: Towards Characterizing Users' Interaction with Zoomable Video SAPMIA'2010, 21-24
- N.Q.M. Khiem, R. Guntur, A. Carlier, W.T. Ooi: Supporting Zoomable Video Streams via Dynamic Region-of-interest Cropping MMSys'2010, 259-270

As suggested by this list, I had the opportunity to collaborate with researchers from NUS (Singapore), FAU (USA), UPC (Barcelona, Spain), Universität Klagenfurt (Austria) and many contributions are shared between our group in Toulouse and Wei Tsang Ooi, Oge Marques and Xavier Giro-i-Nieto. I wish to thank them once again for this prolific collaboration.

In this collaborative context, I also had the opportunity to work on closely related subjects that were part of other PhD students' work: Shanghong Zhao (from NUS), and Thi Phuong Nghiem (from our group). This is why I have put aside a set of contributions about 3D content from my thesis. In [Nghiem 2012] we showed how to crowdsource spatial relations between a 3D model and its semantic description. This allows to determine keyviews to visualize the most interesting parts of 3D objects. Recommended views can be crowdsourced in a very similar manner than the techniques proposed in the first part of this thesis (see chapter 3). In addition we studied in [Zhao 2013, Zhao 2014] ways of displaying a 3D content to a user and adapting this visualization (a camera path) to streaming constraints.

8.3 Future work

The major perspectives for the work presented in this thesis can be divided into three categories: short term (in the next few months), medium term (in the next two years) and long term.

Short term perspectives.

- Part I: the 3D interest maps can probably lead to many future work. To begin with, we should test the video query (section 4.3) on a real setup. Now that the interface is implemented, we need to study how users interact with this interface and whether the query results really match users' intentions.

  Regarding the 3D interest maps computation, we would like to develop an approach in which we would not use the PMVS software (for 3D reconstruction). This software is not very adapted to our problem, since it provides a point cloud reconstruction whereas we only need a very coarse reconstruction. Alternatively, we may also rethink

our work following uncalibrated approaches. The authors of Crowdcam ([Arpa 2013] exploit sparse feature flows inbetween neighbouring images/cameras and structure these visual data using distances and angles. In our case, the idea would be to replace the 3D space - where interest levels are estimated - with descriptor spaces where relationships between videos crowd- sourced from different cameras could also be established.

- Part II: our next efforts will focus on trying to automatically find and filter out errors, in order to improve the detection and segmentation results. We would also like to develop a video segmentation tool similar to Click'n'Cut. About Ask'nSeek there are many uses for the traces that we haven't investigated in details yet, such as trying to compute saliency maps from the target regions.

Medium term.

- Part I: one of the steps to compute the 3D interest maps is the detection of 2D ROI in each video. We are using off-the-shelf content analysis algorithms at the moment, and one other avenue could be to use the methods introduced in sections 3.1 and 3.2. We can augment the query system introduced in section 4.3 with a zoomable video interaction, and compute user interest maps out of the gathered traces. Better 2D interest maps would help produce better 3D interest maps.

  The idea that has driven the entire chapter 3 is that users, through the use of interfaces, can implicitly reveal information about the content. We could apply this same idea to our query system: when users watch a video among the entire set of videos, they implicitly say that they are satisfied with the current video and do not need to query another better view. In that sense we could devise a measure of the interest of each video (over time), that we could then use to improve video mashups.

- Part II: one of the major issues we currently have with Ask'nSeek is the text processing. It should be one of our mid-term goals to perform this step automatically. The next step will be to perform segmentation of several objects at the same time using Ask'nSeek traces.

  There is another avenue that we partially explored in [Carlier 2012]. Ask'nSeek traces can be very useful to filter false positives from content analysis algorithms. One idea could be to relax some constraints on the content analysis algorithm, which would augment the Recall (higher chance to obtain a true positive) but in the mean time decrease the Precision (higher rate of false positives). We would then use Ask'nSeek traces to filter out the false positives and increase the Precision. This approach could be used in object detection and in object segmentation for example.

Long term

Regarding Ask'nSeek, our analysis on section 7.3 encourages us to aim at defining an active learning setup for the game. We have seen that Ask'nSeek traces are biased towards

the most prominent objects (and object parts) on the images. We could devise a system that would determine where to place the target region on an image in order to gain as much information as possible from the game logs. Then we could bias the master's choice of the target region (e.g., by granting more points if the master follows the system's advice) in order to gather more informative traces through the Ask'nSeek game. This proposal makes an ambitious topic that we could share, again, with the UPC and FAU teams.

Concerning the part I (in which all the work we have presented was developed in collaboration with Wei Tsang Ooi at NUS), the long term research problem that we target is the conception of multimedia interactive systems for which we can control the predictability. Predicting users interactions with a content can greatly assist the adaptive streaming of the content, as well as improve the Quality of Experience. Predicting users interactions is however greatly dependent on the degrees of freedom provided by the interactions. For example a "standard" zoomable video player (as described in section 3.1) is highly unpredictable, whereas a zoomable video player with recommmended viewports (section 3.2) considerably reduces the users' freedom and therefore generates more predictable interactions. In the future, we want to explore the conception of systems that could adapt their ergonomics (to network constraints, for example) in order to adapt their predictability which would help optimizing the Quality of Experience.

Bibliography

[Achanta 2012] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua et Sabine Susstrunk. *SLIC superpixels compared to state-of-the-art superpixel methods*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 34, no. 11, pages 2274–2282, 2012. (Cited in page 112.)

[Adamek 2006] Tomasz Adamek. *Using contour information and segmentation for object registration, modeling and retrieval*. PhD thesis, Dublin City University, 2006. (Cited in pages 27 et 115.)

[Araujo 2013] Ricardo Matsumura Araujo. *99designs: An Analysis of Creative Competition in Crowdsourced Design*. In First AAAI Conference on Human Computation and Crowdsourcing, 2013. (Cited in page 17.)

[Arbelaez 2008] P. Arbelaez et L. Cohen. *Constrained image segmentation from hierarchical boundaries*. In CVPR'08, pages 1–8, 2008. (Cited in page 27.)

[Arbeláez 2014] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques et Jitendra Malik. *Multiscale Combinatorial Grouping*. In WCVPR 2014-W. on Computer Vision and Human Computation, 2014. (Cited in page 113.)

[Arijon 1991] D. Arijon. Grammar of the film language. Silman-James Press, 1991. (Cited in pages 81 et 86.)

[Arpa 2013] Aydin Arpa, Luca Ballan, Rahul Sukthankar, Gabriel Taubin, Marc Pollefeys et Ramesh Raskar. *CrowdCam: Instantaneous Navigation of Crowd Images Using Angled Graph*. In 3DTV-Conference, 2013 International Conference on, pages 422–429. IEEE, 2013. (Cited in page 161.)

[Ayache 1991] Nicholas Ayache. Artificial vision for mobile robots - stereo vision and multisensory perception. MIT Press, 1991. (Cited in page 2.)

[Batra 2010] Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo et Tsuhan Chen. *iCoseg: Interactive co-segmentation with intelligent scribble guidance*. In Proceedings of CVPR'10, pages 3169–3176, 2010. (Cited in page 27.)

[Batra 2011] Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo et Tsuhan Chen. *Interactively co-segmentating topically related images with intelligent scribble guidance*. International journal of computer vision, vol. 93, no. 3, pages 273–292, 2011. (Cited in pages 25, 26 et 28.)

[Beleznai 2006] Csaba Beleznai, Bernhard Frühstück et Horst Bischof. *Human tracking by fast mean shift mode seeking*. Journal of Multimedia, vol. 1, no. 1, pages 1–8, 2006. (Cited in page 54.)

[Bernstein 2010] Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell et Katrina Panovich. *Soylent: a word processor with a crowd inside.* In Proceedings of the 23nd annual ACM symposium on User interface software and technology, pages 313–322. ACM, 2010. (Cited in page 18.)

[Biederman 1987] Irving Biederman. *Recognition-by-components: a theory of human image understanding.* Psychological review, vol. 94, no. 2, page 115, 1987. (Cited in page 11.)

[Borji 2012] Ali Borji, Dicky N Sihite et Laurent Itti. *Salient object detection: A benchmark.* In Computer Vision–ECCV 2012, pages 414–429. Springer, 2012. (Cited in page 75.)

[Boykov 2001] Y.Y. Boykov et M.-P. Jolly. *Interactive graph cuts for optimal boundary map; region segmentation of objects in N-D images.* In ICCV'01, pages 105–112 vol.1, 2001. (Cited in page 26.)

[Branson 2010] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona et Serge Belongie. *Visual recognition with humans in the loop.* In Computer Vision–ECCV 2010, pages 438–451. Springer, 2010. (Cited in pages 31 et 32.)

[Branson 2011] Steve Branson, Pietro Perona et Serge Belongie. *Strong supervision from weak annotation: Interactive training of deformable part models.* In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 1832–1839. IEEE, 2011. (Cited in page 32.)

[Branson 2014] Steve Branson, Grant Van Horn, Catherine Wah, Pietro Perona et Serge Belongie. *The Ignorant Led by the Blind: A Hybrid Human–Machine Vision System for Fine-Grained Categorization.* International Journal of Computer Vision, pages 1–27, 2014. (Cited in pages 4 et 32.)

[Canny 1986] John Canny. *A computational approach to edge detection.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, no. 6, pages 679–698, 1986. (Cited in page 1.)

[Carlier 2010a] Axel Carlier, Vincent Charvillat, Wei Tsang Ooi, Romulus Grigoras et Geraldine Morin. *Crowdsourced automatic zoom and scroll for video retargeting.* In Proceedings of the international conference on Multimedia, pages 201–210. ACM, 2010. (Cited in pages 47, 48, 55, 94 et 158.)

[Carlier 2010b] Axel Carlier, Ravindra Guntur et Wei Tsang Ooi. *Towards characterizing users' interaction with zoomable video.* In Proceedings of the 2010 ACM workshop on Social, adaptive and personalized multimedia interaction and access, pages 21–24. ACM, 2010. (Cited in pages 47 et 158.)

[Carlier 2011a] Axel Carlier, Ravindra Guntur, Vincent Charvillat et Wei Tsang Ooi. *Combining content-based analysis and crowdsourcing to improve user interaction with zoomable video.* In Proceedings of the 19th ACM international conference on Multimedia, pages 43–52. ACM, 2011. (Cited in pages 47, 60 et 158.)

[Carlier 2011b] Axel Carlier, Arash Shafiei, Julien Badie, Salim Bensiali et Wei Tsang Ooi. *COZI: crowdsourced and content-based zoomable video player.* In Proceedings of the 19th ACM international conference on Multimedia, pages 829–830. ACM, 2011. (Cited in page 158.)

[Carlier 2012] Axel Carlier, Oge Marques et Vincent Charvillat. *Ask'nSeek: A new game for object detection and labeling.* In Computer Vision–ECCV 2012. Workshops and Demonstrations, pages 249–258. Springer, 2012. (Cited in pages 22, 139, 158 et 161.)

[Carlier 2014] Axel Carlier, Lilian Calvet, Duong Nguyen, Wei Tsang Ooi, Pierre Gurdjos et Vincent Charvillat. *3D Interest Maps From Simultaneous Video Recordings.* In (to appear in) ACM Multimedia. ACM, 2014. (Cited in page 77.)

[Carreira 2010] Joao Carreira et Cristian Sminchisescu. *Constrained parametric min-cuts for automatic object segmentation.* In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 3241–3248. IEEE, 2010. (Cited in page 113.)

[Chakravarthy 2006] Ajay Chakravarthy, Fabio Ciravegna et Vitaveska Lanfranchi. *AKTiveMedia: Cross-media document annotation and enrichment.* In Poster Proceedings of 5th International Semantic Web Conference (ISWC), Athens, GA, USA, 2006. (Cited in page 12.)

[Chapelle 2006] O. Chapelle, B. Schölkopf et A. Zien, editeurs. Semi-supervised learning. MIT Press, Cambridge, 2006. (Cited in page 128.)

[Chen 2012] Ding-Jie Chen, Hwann-Tzong Chen et Long-Wen Chang. *Video object cosegmentation.* In Proceedings of the 20th ACM international conference on Multimedia, pages 805–808. ACM, 2012. (Cited in page 27.)

[Chen 2014] Liang-Chieh Chen, Sanja Fidler, Alan L Yuille et Raquel Urtasun. *Beat the MTurkers: Automatic Image Labeling from Weak 3D Supervision.* In CVPR, 2014. (Cited in page 119.)

[Cheng 2009] Kai-Yin Cheng, Sheng-Jie Luo, Bing-Yu Chen et Hao-Hua Chu. *SmartPlayer: User-Centric Video Fast-Forwarding.* In Proceedings of CHI'09, 2009. (Cited in pages 23 et 24.)

[Comaniciu 2002] Dorin Comaniciu et Peter Meer. *Mean Shift: A Robust Approach Toward Feature Space Analysis.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 5, pages 603–619, 2002. (Cited in pages 24, 46, 52, 69 et 112.)

[Dasiopoulou 2011] Stamatia Dasiopoulou, Eirini Giannakidou, Georgios Litos, Polyxeni Malasioti et Yiannis Kompatsiaris. *A survey of semantic image and video annotation tools*. In Knowledge-driven multimedia information extraction and ontology evolution, pages 196–239. Springer, 2011. (Cited in page 12.)

[Di Salvo 2013] R Di Salvo, D Giordano et I Kavasidis. *A crowdsourcing approach to support video annotation*. In Proceedings of the International Workshop on Video and Image Ground Truth in Computer Vision Applications, page 8. ACM, 2013. (Cited in page 22.)

[Dittrich 2013] Torben Dittrich, Stephan Kopf, Philipp Schaber, Benjamin Guthier et Wolfgang Effelsberg. *Saliency detection for stereoscopic video*. In Proceedings of the 4th ACM Multimedia Systems Conference, pages 12–23. ACM, 2013. (Cited in page 51.)

[Doermann 2000] David Doermann et David Mihalcik. *Tools and techniques for video performance evaluation*. In Pattern Recognition, International Conference on, volume 4, pages 4167–4167. IEEE Computer Society, 2000. (Cited in pages 12 et 24.)

[Doubek 2004] Petr Doubek, Indra Geys, Tomas Svoboda et Luc Van Gool. *Cinematographic Rules Applied to a Camera Network*. In Proc. of Omnivis, The fifth Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras, pages 17–30, 2004. (Cited in page 81.)

[Dragicevic 2008] Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowitcz, Derek Nowrouzezahrai, Ravin Balakrishnan et Karan Singh. *Video browsing by direct manipulation*. In Proceedings of CHI'08, pages 237–246, Florence, Italy, 2008. (Cited in page 23.)

[El-Alfy 2007] Hazem El-Alfy, David Jacobs et Larry Davis. *Multi-scale video cropping*. In Proceedings of the 15th international conference on Multimedia, pages 97–106. ACM, 2007. (Cited in pages 52 et 81.)

[Everingham 2010] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn et A. Zisserman. *The Pascal Visual Object Classes (VOC) Challenge*. International Journal of Computer Vision, vol. 88, no. 2, pages 303–338, Juin 2010. (Cited in pages 3, 8, 11, 113 et 139.)

[Fan 2003] Xin Fan, Xing Xie, He-Qin Zhou et Wei-Ying Ma. *Looking into video frames on small displays*. In Proceedings of MULTIMEDIA '03, pages 247–250, Berkeley, CA, USA, 2003. ACM. (Cited in page 81.)

[Fei-Fei 2004] L. Fei-Fei, R. Fergus et P. Perona. *Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories*. IEEE. CVPR 2004, Workshop on Generative-Model Based Vision., 2004. (Cited in page 11.)

[Felzenszwalb 2004] Pedro F Felzenszwalb et Daniel P Huttenlocher. *Efficient graph-based image segmentation*. IJCV, vol. 59, no. 2, 2004. (Cited in pages 112, 113 et 151.)

[Feng 2012] Jiashi Feng, Yuzhao Ni, Jian Dong, Zilei Wang et Shuicheng Yan. *Purposive hidden-object-game: embedding human computation in popular game*. Multimedia, IEEE Transactions on, vol. 14, no. 5, pages 1496–1507, 2012. (Cited in page 22.)

[Flandin 2002] Grégory Flandin et François Chaumette. *Visual data fusion for objects localization by active vision*. In Computer Vision-ECCV 2002, pages 312–326. Springer, 2002. (Cited in page 69.)

[Furukawa 2010] Yasutaka Furukawa et Jean Ponce. *Accurate, dense, and robust multiview stereopsis*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 32, no. 8, pages 1362–1376, 2010. (Cited in page 70.)

[Giro-i Nieto 2013] Xavier Giro-i Nieto, Manuel Martos, Eva Mohedano et Jordi Pont-Tuset. *From Global Image Annotation to Interactive Object Segmentation*. Multimedia Tools and Applications, vol. 46, no. 2, pages 155–174, 2013. (Cited in page 27.)

[Gkonela 2014] Chrysoula Gkonela et Konstantinos Chorianopoulos. *VideoSkip: event detection in social web videos with an implicit user heuristic*. Multimedia Tools and Applications, vol. 69, no. 2, pages 383–396, 2014. (Cited in page 20.)

[Gleicher 2008] Michael L. Gleicher et Feng Liu. *Re-cinematography: Improving the camerawork of casual video*. ACM Trans. Multimedia Comput. Commun. Appl., vol. 5, no. 1, pages 1–28, 2008. (Cited in page 81.)

[Goferman 2012] Stas Goferman, Lihi Zelnik-Manor et Ayellet Tal. *Context-aware saliency detection*. Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 34, no. 10, pages 1915–1926, 2012. (Cited in pages 23, 72, 73, 74 et 75.)

[Goldman 2008] Dan B. Goldman, Chris Gonterman, Brian Curless, David Salesin et Steven M. Seitz. *Video object annotation, navigation, and composition*. In Proceedings of UIST'08, pages 3–12, 2008. (Cited in page 23.)

[Gottlieb 2012] Luke Gottlieb, Jaeyoung Choi, Pascal Kelm, Thomas Sikora et Gerald Friedland. *Pushing the limits of mechanical turk: qualifying the crowd for video geo-location*. In Proceedings of the ACM multimedia 2012 workshop on Crowdsourcing for multimedia, pages 23–28. ACM, 2012. (Cited in page 17.)

[Grauman 2014] Kristen Grauman et Serge Belongie. *Editorial: Special Issue on Active and Interactive Methods in Computer Vision*. International Journal of Computer Vision, vol. 108, no. 1-2, pages 1–2, 2014. (Cited in page 22.)

[Grier 2011] David Alan Grier. *Error Identification and Correction in Human Computation: Lessons from the WPA.* In Human Computation, 2011. (Cited in page 18.)

[Gulshan 2010] V. Gulshan, C. Rother, Antonio Criminisi, A. Blake et A. Zisserman. *Geodesic star convexity for interactive image segmentation.* In CVPR'10, pages 3129–3136, 2010. (Cited in page 27.)

[Halaschek-Wiener 2005] Christian Halaschek-Wiener, Jennifer Golbeck, Andrew Schain, Michael Grove, Bijan Parsia et Jim Hendler. *Photostuff-an image annotation tool for the semantic web.* In Proceedings of the Poster Track, 4th International Semantic Web Conference (ISWC2005). Citeseer, 2005. (Cited in pages 12 et 24.)

[Han 2006] Junwei Han, King Ngi Ngan, Mingjing Li et HongJiang Zhang. *Unsupervised extraction of visual attention objects in color images.* IEEE Trans. Circuits Syst. Video Techn., vol. 16, no. 1, pages 141–145, 2006. (Cited in pages 23 et 44.)

[Hartley 2003] Richard Hartley et Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003. (Cited in pages 65, 68 et 72.)

[He 1996] Li-wei He, Michael F. Cohen et David H. Salesin. *The virtual cinematographer: a paradigm for automatic real-time camera control and directing.* In Proceedings of SIGGRAPH '96, pages 217–224. ACM, 1996. (Cited in page 81.)

[Heitz 2008] Geremy Heitz et Daphne Koller. *Learning Spatial Context: Using Stuff to Find Things.* In ECCV, pages 30–43, 2008. (Cited in page 129.)

[Ho 2009] Chien-Ju Ho, Tao-Hsuan Chang, Jong-Chuan Lee, Jane Yung-jen Hsu et Kuan-Ta Chen. *KissKissBan: a competitive human computation game for image annotation.* In Proceedings of the acm sigkdd workshop on human computation, pages 11–14. ACM, 2009. (Cited in page 21.)

[Howe 2006] Jeff Howe. *The rise of crowdsourcing.* Wired magazine, vol. 14, no. 6, pages 1–4, 2006. (Cited in page 14.)

[Huang 2009] Tz-Huan Huang, Kai-Yin Cheng et Yung-Yu Chuang. *A Collaborative Benchmark for Region of Interest Detection Algorithms.* In Proceedings of CVPR '09, Miami, FL, June 2009. (Cited in pages 45 et 46.)

[Ipeirotis 2010] Panagiotis G Ipeirotis, Foster Provost et Jing Wang. *Quality management on amazon mechanical turk.* In Proceedings of the ACM SIGKDD workshop on human computation, pages 64–67. ACM, 2010. (Cited in page 18.)

[Itti 1998] Laurent Itti, Christof Koch et Ernst Niebur. *A model of saliency-based visual attention for rapid scene analysis.* IEEE Transactions on pattern analysis and machine intelligence, vol. 20, no. 11, pages 1254–1259, 1998. (Cited in pages 8, 23, 52 et 138.)

[Jain 2013] Suyog Dutt Jain et Kristen Grauman. *Predicting Sufficient Annotation Strength for Interactive Foreground Segmentation.* In ICCV, 2013. (Cited in page 119.)

[Kaufmann 2011] Nicolas Kaufmann et Thimo Schulze. *Worker motivation in crowdsourcing and human computation.* Education, vol. 17, 2011. (Cited in page 15.)

[Kavasidis 2013] Isaak Kavasidis, Simone Palazzo, Roberto Di Salvo, Daniela Giordano et Concetto Spampinato. *An innovative web-based collaborative platform for video annotation.* Multimedia Tools and Applications, pages 1–20, 2013. (Cited in pages 24 et 25.)

[Kelly 2003] Diane Kelly et Jaime Teevan. *Implicit feedback for inferring user preference: a bibliography.* In ACM SIGIR Forum, volume 37, pages 18–28. ACM, 2003. (Cited in page 20.)

[Kim 2011] Edward Kim, Xiaolei Huang et Gang Tan. *Markup SVG—An Online Content-Aware Image Abstraction and Annotation Tool.* Trans. Multi., vol. 13, no. 5, pages 993–1006, Octobre 2011. (Cited in pages 24 et 25.)

[Kipp 2010] Michael Kipp. *Multimedia annotation, querying and analysis in ANVIL.* Multimedia information extraction, vol. 19, 2010. (Cited in pages 12 et 24.)

[Kovashka 2011] Adriana Kovashka, Sudheendra Vijayanarasimhan et Kristen Grauman. *Actively selecting annotations among objects and attributes.* In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 1403–1410. IEEE, 2011. (Cited in page 30.)

[Law 2007] Edith LM Law, Luis Von Ahn, Roger B Dannenberg et Mike Crawford. *TagATune: A Game for Music and Sound Annotation.* In ISMIR, volume 3, page 2, 2007. (Cited in page 18.)

[Lee 2014] Han S Lee, Jiwhan Kim, Sun Jeong Park et Junmo Kim. *Interactive Segmentation as Supervised Classification with Superpixels.* In WCVPR 2014-W. on Computer Vision and Human Computation, 2014. (Cited in page 27.)

[Li 2005] Tsai-Yen Li et Xiang-Yan Xiao. *An Interactive Camera Planning System for Automatic Cinematographer.* volume 0, pages 310–315, Los Alamitos, CA, USA, 2005. IEEE Computer Society. (Cited in page 81.)

[Lin 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár et C. Lawrence Zitnick. *Microsoft COCO: Common Objects in Context.* CoRR, 2014. (Cited in page 119.)

[Liu 2006] Feng Liu et Michael Gleicher. *Video retargeting: automating pan and scan.* In Proceedings of the 14th annual ACM international conference on Multimedia, pages 241–250. ACM, 2006. (Cited in pages 52 et 81.)

[Lux 2003] Mathias Lux, Jutta Becker et Harald Krottmaier. *Caliph & Emir: Semantic Annotation and Retrieval of Digital Photos.* In CAiSE Short Paper Proceedings, 2003. (Cited in pages 12 et 24.)

[Lux 2009] Mathias Lux. *Caliph & Emir: MPEG-7 photo annotation and retrieval.* In Proceedings of the 17th ACM international conference on Multimedia, pages 925–926. ACM, 2009. (Cited in pages 12 et 24.)

[Malmberg 2012] Filip Malmberg, Robin Strand, Joel Kullberg, Richard Nordenskjöld et Ewert Bengtsson. *Smart Paint-A New Interactive Segmentation Method Applied to MR Prostate Segmentation.* Medical Image Computing and Computer Assisted Intervention (MICCAI) Grand Challenge: Prostate MR Image Segmentation, vol. 8, 2012. (Cited in page 27.)

[Mao 2013] Andrew Mao, Ece Kamar, Yiling Chen, Eric Horvitz, Megan E Schwamb, Chris J Lintott et Arfon M Smith. *Volunteering Versus Work for Pay: Incentives and Tradeoffs in Crowdsourcing.* In First AAAI Conference on Human Computation and Crowdsourcing, 2013. (Cited in pages 17 et 18.)

[Martin 2001] D. Martin, C. Fowlkes, D. Tal et J. Malik. *A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics.* In ICCV, 2001. (Cited in page 114.)

[McGuinness 2010] Kevin McGuinness et Noel E. O'Connor. *A comparative evaluation of interactive segmentation algorithms.* Pattern Recognition, vol. 43, no. 2, 2010. (Cited in pages 27, 115, 119 et 138.)

[McGuinness 2013] K. McGuinness et N.E. O'Connor. *Improved graph cut segmentation by learning a contrast model on the fly.* In ICIP, 2013. (Cited in page 27.)

[Montabone 2010] Sebastian Montabone et Alvaro Soto. *Human detection using a mobile platform and novel features derived from a visual saliency mechanism.* Image and Vision Computing, vol. 28, no. 3, pages 391–402, 2010. (Cited in page 51.)

[Nghiem 2012] Thi Phuong Nghiem, Axel Carlier, Geraldine Morin et Vincent Charvillat. *Enhancing online 3D products through crowdsourcing.* In Proceedings of the ACM multimedia 2012 workshop on Crowdsourcing for multimedia, pages 47–52. ACM, 2012. (Cited in page 160.)

[Ngo 2011] Khiem Quang Minh Ngo, Ravindra Guntur et Wei Tsang Ooi. *Adaptive encoding of zoomable video streams based on user access pattern.* In Proceedings of the second annual ACM conference on Multimedia systems, pages 211–222. ACM, 2011. (Cited in page 51.)

[Nguyen 2013] Duong-Trung-Dung Nguyen, Mukesh Saini, Vu-Thanh Nguyen et Wei Tsang Ooi. *Jiku director: a mobile video mashup system.* In Proceedings of the 21st

ACM international conference on Multimedia, pages 477–478. ACM, 2013. (Cited in page 93.)

[Noma 2012] Alexandre Noma, Ana B.V. Graciano, Roberto M. Cesar Jr, Luis A. Consularo et Isabelle Bloch. *Interactive image segmentation by matching attributed relational graphs*. Pattern Recognition, vol. 45, no. 3, pages 1159 – 1179, 2012. (Cited in page 27.)

[Oleson 2011] David Oleson, Alexander Sorokin, Greg P Laughlin, Vaughn Hester, John Le et Lukas Biewald. *Programmatic Gold: Targeted and Scalable Quality Assurance in Crowdsourcing.* Human computation, vol. 11, page 11, 2011. (Cited in pages 17, 18 et 144.)

[Petridis 2006] Kosmas Petridis, Dionysios Anastasopoulos, Carsten Saathoff, Norman Timmermann, Yiannis Kompatsiaris et Steffen Staab. *M-ontomat-annotizer: Image annotation linking ontologies and multimedia low-level features*. In Knowledge-Based Intelligent Information and Engineering Systems, pages 633–640. Springer, 2006. (Cited in pages 24 et 25.)

[Plesca 2008a] Cezar Plesca, Vincent Charvillat et Romulus Grigoras. *Adapting content delivery to limited resources and inferred user interest*. International Journal of Digital Multimedia Broadcasting, vol. 2008, 2008. (Cited in page 20.)

[Plesca 2008b] Cezar Plesca, Vincent Charvillat et Romulus Grigoras. *User-aware adaptation by subjective metadata and inferred implicit descriptors*. In Multimedia Semantics The Role of Metadata, pages 127–147. Springer, 2008. (Cited in page 20.)

[Price 2009] Brian L Price, Bryan S Morse et Scott Cohen. *Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues*. In Computer Vision, 2009 IEEE 12th International Conference on, pages 779–786. IEEE, 2009. (Cited in page 27.)

[Price 2011] Brian L Price et Scott Cohen. *Stereocut: Consistent interactive object selection in stereo image pairs*. In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 1148–1155. IEEE, 2011. (Cited in page 27.)

[Quang Minh Khiem 2010] Ngo Quang Minh Khiem, Guntur Ravindra, Axel Carlier et Wei Tsang Ooi. *Supporting zoomable video streams with dynamic region-of-interest cropping*. In Proceedings of the first annual ACM SIGMM conference on Multimedia systems, pages 259–270. ACM, 2010. (Cited in pages 41, 42, 47 et 158.)

[Quinn 2011] Alexander J Quinn et Benjamin B Bederson. *Human computation: a survey and taxonomy of a growing field*. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 1403–1412. ACM, 2011. (Cited in page 15.)

[Riegler 2014] Michael Riegler, Mathias Lux, Vincent Charvillat, Axel Carlier, Raynor Vliegendhart et Martha Larson. *VideoJot: A Multifunctional Video Annotation Tool*. In Proceedings of International Conference on Multimedia Retrieval, page 534. ACM, 2014. (Cited in page 23.)

[Rother 2004] Carsten Rother, Vladimir Kolmogorov et Andrew Blake. *"GrabCut": interactive foreground extraction using iterated graph cuts*. ACM Trans. Graph., vol. 23, no. 3, Août 2004. (Cited in pages 27, 110 et 115.)

[Rowe 2005] Lawrence A. Rowe et Ramesh Jain. *ACM SIGMM Retreat Report on Future Directions in Multimedia Research*. ACM Trans. Multimedia Comput. Commun. Appl., vol. 1, no. 1, pages 3–13, Février 2005. (Cited in page 3.)

[Rowe 2013] Lawrence A. Rowe. *Looking Forward 10 Years to Multimedia Successes*. ACM Trans. Multimedia Comput. Commun. Appl., vol. 9, no. 1s, pages 37:1–37:7, Octobre 2013. (Cited in page 3.)

[Russell 2008] Bryan C Russell, Antonio Torralba, Kevin P Murphy et William T Freeman. *LabelMe: a database and web-based tool for image annotation*. International journal of computer vision, vol. 77, no. 1-3, pages 157–173, 2008. (Cited in pages 12, 24 et 147.)

[Saathoff 2008] Carsten Saathoff, Simon Schenk et Ansgar Scherp. *Kat: the k-space annotation tool*. In Poster Session, Int. Conf. on Semantic and Digital Media Technologies (SAMT), Koblenz, Germany, 2008. (Cited in page 12.)

[Saini 2012] Mukesh Kumar Saini, Raghudeep Gadde, Shuicheng Yan et Wei Tsang Ooi. *MoViMash: online mobile video mashup*. In Proceedings of the 20th ACM international conference on Multimedia, pages 139–148. ACM, 2012. (Cited in pages 93 et 96.)

[Salembier 2000] P. Salembier et L. Garrido. *Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval*. IEEE T. on Image Processing, vol. 9, no. 4, 2000. (Cited in pages 27 et 115.)

[Salvador 2013] Amaia Salvador, Axel Carlier, Xavier Giro-i Nieto, Oge Marques et Vincent Charvillat. *Crowdsourced object segmentation with a game*. In Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia, pages 15–20. ACM, 2013. (Cited in pages 139, 154 et 158.)

[Schallauer 2008] Peter Schallauer, Sandra Ober et Helmut Neuschmied. *Efficient semantic video annotation by object and shot re-detection*. In Posters and Demos Session, 2nd International Conference on Semantic and Digital Media Technologies (SAMT), Koblenz, Germany, 2008. (Cited in pages 24 et 25.)

[Settles 2010] Burr Settles. *Active learning literature survey.* University of Wisconsin, Madison, vol. 52, pages 55–66, 2010. (Cited in pages 28 et 29.)

[Shamma 2007] David A. Shamma, Ryan Shaw, Peter L. Shafton et Yiming Liu. *Watch what I watch: using community activity to understand content.* In Multimedia Information Retrieval, pages 275–284, 2007. (Cited in page 20.)

[Shi 2000] Jianbo Shi et Jitendra Malik. *Normalized cuts and image segmentation.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 22, no. 8, pages 888–905, 2000. (Cited in pages 24 et 112.)

[Shipman 2008] Frank Shipman, Andreas Girgensohn et Lynn Wilcox. *Authoring, viewing, and generating hypervideo: An overview of Hyper-Hitchcock.* ACM Trans. Multimedia Comput. Commun. Appl., vol. 5, no. 2, pages 1–19, 2008. (Cited in page 23.)

[Shrestha 2010] Prarthana Shrestha, Mauro Barbieri, Hans Weda et Dragan Sekulovski. *Synchronization of multiple camera videos using audio-visual features.* Multimedia, IEEE Transactions on, vol. 12, no. 1, pages 79–92, 2010. (Cited in page 72.)

[Spampinato 2012] Concetto Spampinato, Bas Boom et Jiyin He. Proceedings of the 1st international workshop on visual interfaces for ground truth collection in computer vision applications. ACM, 2012. (Cited in page 12.)

[Steggink 2011] Jeroen Steggink et Cees GM Snoek. *Adding semantics to image-region annotations with the name-it-game.* Multimedia systems, vol. 17, no. 5, pages 367–378, 2011. (Cited in page 21.)

[Sturm 2000] Peter Sturm. *Algorithms for plane-based pose estimation.* In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, volume 1, pages 706–711. IEEE, 2000. (Cited in page 72.)

[Su 2012] Hao Su, Jia Deng et Li Fei-Fei. *Crowdsourcing annotations for visual object detection.* In Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012. (Cited in page 18.)

[Syeda-Mahmood 2001] Tanveer Syeda-Mahmood et Dulce Ponceleon. *Learning video browsing behavior and its application in the generation of video previews.* In Proceedings of MULTIMEDIA'01, pages 119–128, Ottawa, Canada, 2001. (Cited in page 20.)

[Szeliski 2011] Richard Szeliski. Computer vision - algorithms and applications. Texts in Computer Science. Springer, 2011. (Cited in page 3.)

[Tatler 2007] Benjamin W Tatler. *The central fixation bias in scene viewing: Selecting an optimal viewing position independently of motor biases and image feature distributions.* Journal of Vision, vol. 7, no. 14, page 4, 2007. (Cited in page 64.)

[Ukita 2005] Norimichi Ukita, Tomohisa Ono et Masatsugu Kidode. *Region extraction of a gaze object using the gaze point and view image sequences*. In Proceedings of the 7th international conference on Multimodal interfaces, pages 129–136. ACM, 2005. (Cited in page 20.)

[Van de Sande 2011] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers et Arnold WM Smeulders. *Segmentation as selective search for object recognition*. In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 1879–1886. IEEE, 2011. (Cited in page 113.)

[Vázquez 2001] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert et Wolfgang Heidrich. *Viewpoint Selection Using Viewpoint Entropy*. In Proceedings of VMV 2001, pages 273–280, 2001. (Cited in page 101.)

[Vijayanarasimhan 2012] Sudheendra Vijayanarasimhan et Kristen Grauman. *Active frame selection for label propagation in videos*. In Computer Vision–ECCV 2012, pages 496–509. Springer, 2012. (Cited in page 30.)

[Vijayanarasimhan 2014] Sudheendra Vijayanarasimhan et Kristen Grauman. *Large-scale live active learning: Training object detectors with crawled data and crowds*. International Journal of Computer Vision, vol. 108, no. 1-2, pages 97–114, 2014. (Cited in page 30.)

[Vincent 1991] Luc Vincent et Pierre Soille. *Watersheds in digital spaces: an efficient algorithm based on immersion simulations*. IEEE transactions on pattern analysis and machine intelligence, vol. 13, no. 6, pages 583–598, 1991. (Cited in page 1.)

[Viola 2001] Paul Viola et Michael Jones. *Rapid object detection using a boosted cascade of simple features*. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, volume 1, pages I–511. IEEE, 2001. (Cited in page 51.)

[Vittayakorn 2011] Sirion Vittayakorn et James Hays. *Quality Assessment for Crowdsourced Object Annotations*. In BMVC, pages 1–11, 2011. (Cited in page 32.)

[Vliegendhart 2012] Raynor Vliegendhart, Martha Larson et Alan Hanjalic. *LikeLines: collecting timecode-level feedback for web videos through user interactions*. In Proceedings of the 20th ACM international conference on Multimedia, pages 1271–1272. ACM, 2012. (Cited in page 23.)

[Von Ahn 2004] Luis Von Ahn et Laura Dabbish. *Labeling images with a computer game*. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 319–326. ACM, 2004. (Cited in pages 18, 19, 20 et 110.)

[Von Ahn 2005] Luis Von Ahn. *Human Computation*. PhD thesis, Pittsburgh, PA, USA, 2005. AAI3205378. (Cited in page 14.)

[Von Ahn 2006] Luis Von Ahn, Ruoran Liu et Manuel Blum. *Peekaboom: a game for locating objects in images*. In Proceedings of the SIGCHI conference on Human Factors in computing systems, pages 55–64. ACM, 2006. (Cited in page 21.)

[Von Ahn 2008a] Luis Von Ahn et Laura Dabbish. *Designing games with a purpose*. Communications of the ACM, vol. 51, no. 8, pages 58–67, 2008. (Cited in page 18.)

[Von Ahn 2008b] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham et Manuel Blum. *recaptcha: Human-based character recognition via web security measures*. Science, vol. 321, no. 5895, pages 1465–1468, 2008. (Cited in page 14.)

[Vondrick 2013] Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz et Antonio Torralba. *Hoggles: Visualizing object detection features*. In Computer Vision (ICCV), 2013 IEEE International Conference on, pages 1–8. IEEE, 2013. (Cited in page 13.)

[Walther 2006] Dirk Walther et Christof Koch. *Modeling attention to salient proto-objects*. Neural Networks, vol. 19, pages 1395–1407, 2006. (Cited in page 87.)

[Wang 2005] Jue Wang, Pravin Bhat, R. Alex Colburn, Maneesh Agrawala et Michael F. Cohen. *Interactive video cutout*. ACM Trans. Graph., vol. 24, no. 3, pages 585–594, Juillet 2005. (Cited in page 27.)

[Wang 2009] Yu-Shuen Wang, Hongbo Fu, Olga Sorkine, Tong-Yee Lee et Hans-Peter Seidel. *Motion-aware temporal coherence for video resizing*. ACM Transactions on Graphics (TOG), vol. 28, no. 5, page 127, 2009. (Cited in pages 51 et 54.)

[Wang 2013a] Dan Wang, Canxiang Yan, Shiguang Shan et Xilin Chen. *Active learning for interactive segmentation with expected confidence change*. In Computer Vision–ACCV 2012, pages 790–802. Springer, 2013. (Cited in page 28.)

[Wang 2013b] Xiaoyu Wang, Ming Yang, Shenghuo Zhu et Yuanqing Lin. *Regionlets for generic object detection*. In Computer Vision (ICCV), 2013 IEEE International Conference on, pages 17–24. IEEE, 2013. (Cited in page 11.)

[Welinder 2010] Peter Welinder et Pietro Perona. *Online crowdsourcing: rating annotators and obtaining cost-effective labels*. In Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on, pages 25–32. IEEE, 2010. (Cited in page 32.)

[Xie 2005] Xing Xie, Hao Liu, Simon Goumaz et Wei-Ying Ma. *Learning user interest for image browsing on small-form-factor devices*. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 671–680. ACM, 2005. (Cited in pages 20 et 21.)

[Yang 2010] Wenxian Yang, Jianfei Cai, Jianmin Zheng et Jiebo Luo. *User-friendly interactive image segmentation through unified combinatorial user inputs.* Image Processing, IEEE Transactions on, vol. 19, no. 9, pages 2470–2479, 2010. (Cited in page 26.)

[Yuen 2009] Jenny Yuen, Bryan Russell, Ce Liu et Antonio Torralba. *Labelme video: Building a video database with human annotations.* In Computer Vision, 2009 IEEE 12th International Conference on, pages 1451–1458. IEEE, 2009. (Cited in pages 12 et 24.)

[Zhao 2013] Shanghong Zhao, Wei Tsang Ooi, Axel Carlier, Geraldine Morin et Vincent Charvillat. *3D mesh preview streaming.* In Proceedings of the 4th ACM Multimedia Systems Conference, pages 178–189. ACM, 2013. (Cited in page 160.)

[Zhao 2014] Shanghong Zhao, Wei Tsang Ooi, Axel Carlier, Geraldine Morin et Vincent Charvillat. *Bandwidth adaptation for 3D mesh preview streaming.* ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP), vol. 10, no. 1s, page 13, 2014. (Cited in page 160.)