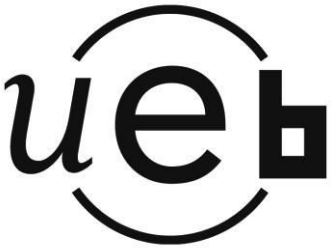
Real-time detection of surgical tools in 2D neurosurgical videos by modelling global shape and local appearance

David Bouget

ANNÉE 2015



THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Génie Biologique et Médical

Ecole doctorale Vie-Agro-Santé

présentée par

David BOUGET

Préparée à l'unité de recherche U1099 MediCIS - LTSI
Modélisation des connaissances et procédures chirurgicales
Science de la vie et de l'environnement

Real-time detection of surgical tools in 2D neurosurgical videos by modelling global shape and local appearance

Thèse soutenue à Rennes
le 27 Mai 2015

devant le jury composé de :

Michel DE MATHELIN
PU, ICube, Strasbourg / *rapporteur*

Danail STOYANOV
CR, University College London / *rapporteur*

Nassir NAVAB
PU, TUM Munich / *examinateur*

Guy CAZUGUEL
MCU, Télécom Bretagne / *examinateur*

Laurent RIFFAUD
PU-PH, CHU Rennes, MediCIS / *examinateur*

Pierre JANNIN
DR INSERM, MediCIS / *directeur de thèse*

Remerciements

Au cours de ces années de thèse, j'ai eu la chance de cotoyer et de travailler avec un certain nombre de personnes, et je voudrais les en remercier.
Pour commencer, mon directeur de thèse Pierre Jannin, tout d'abord pour avoir cru en moi pour cette thèse et avoir fait le nécessaire pour obtenir un financement. Ensuite, pour son aide indispensable apportée pendant ces trois années, et pour m'avoir laisser une grande liberté dans l'organisation de ma recherche.
Ensuite, merci à la société Carl Zeiss Meditec AG, et plus particulièrement au Docteur Marco Wilzbach pour avoir été le décisionnaire principal dans le financement de cette thèse et pour avoir bien voulu mettre à ma disposition un microscope OPMI Pentero. Travailler avec une entreprise privée fut intense, mais appréciable et très enrichissant. En particulier, merci au Docteur Stefan Saur pour son suivi et ses conseils techniques et méthodologiques.
Je remercie ensuite le Professeur Laurent Riffaud pour m'avoir fait l'honneur de présider mon jury de thèse, le Professeur Michel De Mathelin et le Docteur Danail Stoyanov pour avoir accepter de rapporter mon travail, et enfin Guy Cazuguel et le Professeur Nassir Navab pour avoir accepter de faire partie de mon jury de thèse.
Je voudrais aussi remercier les membres du service de Neurochirurgie de l'hôpital de Pontchaillou. Principalement pour leur aide précieuse concernant les termes techniques médicaux mais aussi pour m'avoir permis d'accéder aux données in-vivo, primordiales pour ce travail. Je remercie les docteurs Laurent Riffaud, Claire Haegelen, Pierre-Louis Henaux, et Pierre-Jean Le Reste.
Concernant les collègues de travail, je vais commencer par remercier les anciens, les membres de l'équipe Visages, avec lesquels j'ai effectué mon stage de master jusqu'à la séparation de l'équipe. Je remercie tout particulièrement Olivier Commowick pour sa disponibilité et son aide précieuse à la fois au niveau méthodologique mais aussi technique; ainsi que Pierre M., Aymeric, Guillaume, Alex, Camille, Clément.
Je remercie bien évidemment tous les anciens membres passés par l'équipe Medicis: Florent, mon ancien co-bureau et mentor dans la recherche qui m'a appris à faire de la recherche efficace, et sportif de haut niveau!; Bacem (alias k-mel wali!); Tristan (alias Tryphon), Fabien, co-bureau d'une année, chercheur motivé mais un peu fou-fou; Cédric; Darko, parler anglais pendant les pauses ca fait du bien; Brivael; Germain; les stagiaires...
Bien entendu, je remercie aussi les membres actuels de l'équipe: Noémie, Bernard, Fred, Yulong, Arnaud, Olga, et Clément. On est pas beaucoup, mais suffisament pour passer des bons moments, surtout pendant les pauses café; l'acquisition des canapés aura été une bonne chose! Je ne parle même pas de tous les délires autour de Kaamelott, Joueur du Grenier, Golden Moustache, et humour du genre. Je conçois que ça puisse être fatiguant à force, surtout pour ceux qui ne connaissent pas toutes les références, et encore moins par coeur.
Une mention toute particulière pour les membres du département *Computer Vision and Multimodal Imaging* de l'institut d'informatique Max-Planck de Saarbruck. Je remercie le Professeur Schiele de m'avoir permis de venir travailler chez eux un été.

Et je remercie particulèrement le Dr.Rodrigo Benenson et Mohamed Omran avec qui j'ai appris énormément et sans qui cette thèse n'aurait pu être aussi complète. Pour finir, je remercie Raymond Viard, professeur de Mathématiques à Acadomia, pour m'avoir suivi de la Terminale jusqu'à la fin de l'école d'ingénieurs. Et enfin Ludo, pour les séances de sport salvatrices du vendredi soir, sans lesquelles j'aurais probablement perdu toute santé mentale!

Contents

List of Figures

List of Tables

Glossary

2D 2-dimensional. 26, 32, 45, 114, 129–131, 161, 162, 164, 165, 214

3D 3-dimensional. 45, 130, 161, 162, 165, 214

ACDF Anterior Cervical Discectomy and Fusion. 5

BB Bounding Box. 20

CACAI Context-Aware Computer-Assisted Intervention. 8, 12, 105, 160, 164

CAS Computer-Assisted Surgery. 3, 8, 165, 214

DBS Deep Brain Stimulation. 5

dVSS daVinci Surgical System. 26–28, 33, 162

FoV Field-of-View. 32, 52, 141, 153

FPPI False Positives Per Image. xiii, 21, 87, 88, 117, 118, 126, 132, 163, 214

HOG Histogram Of Gradient. 17, 29, 41, 74, 89, 161, 186

HSV Hue Saturation Value. 29, 41, 140, 148

IOU Intersection Over Union. 155

LAMR Log-Average Miss-Rate. xiii, xvii, 21, 45, 87, 105, 116, 123, 124, 126

LBP Local Binary Patterns. 17

LF Low Frequency. 143

MIS Minimally Invasive Surgery. 5, 26, 28, 162, 214

MRI Magnetic Resonance Imaging. 5

NMS Non-Maximum Suppression. 17, 21, 30, 64, 67, 71, 80, 82, 90, 104, 105, 124, 126, 186, 192

OR Operating Room. 4–8, 12, 38, 105, 143–146, 160–162, 164, 165, 214

PACS Picture Archiving and Communication System. 7

SPM Surgical Process Modeling. 7, 8, 12, 164, 214

SVM Support Vector Machine. xii, 76, 77, 81–83, 91, 111, 114, 115, 162

ToF Time-of-Flight. 26, 38

UHF Ultra-High Frequency. 142–144

Part I

Introduction and related work

Chapter 1

Introduction

Contents

1.1 Working context

The PhD was funded by Carl Zeiss Meditec AG, a subsidiary of Carl Zeiss AG. ZEISS is world renowned for manufacturing high quality optical systems, industrial measurements and medical devices. Figure 1.1 illustrates manufactured medical devices for neurosurgery, eye-surgery, gynaecology, or oncology. Amongst the many devices, surgical microscopes are of interest as used as part of the clinical routine in many hospitals throughout the world.

Figure 1.1: Products of Carl Zeiss Meditec AG.

The work reported by this thesis took place within the Medicis team (Modeling surgical knowledge and processes, https://medicis.univ-rennes1.fr/), belonging to the UMR 1099-LTSI (http://www.ltsi.univ-rennes1.fr/), INSERM (National institute of health and scientific research, http://www.inserm.fr), and University of Rennes 1 (http://www.univ-rennes1.fr).
Medicis team research activities relate to the development of information processing algorithms and Computer-Assisted Surgery (CAS) systems in the neurosurgical

medical context, for a better understanding of brain-related diseases and the conception of the Operating Room (OR) of the future. Two main thematics emerge regarding the creation of surgical assistance systems and knowledge-based procedural models.

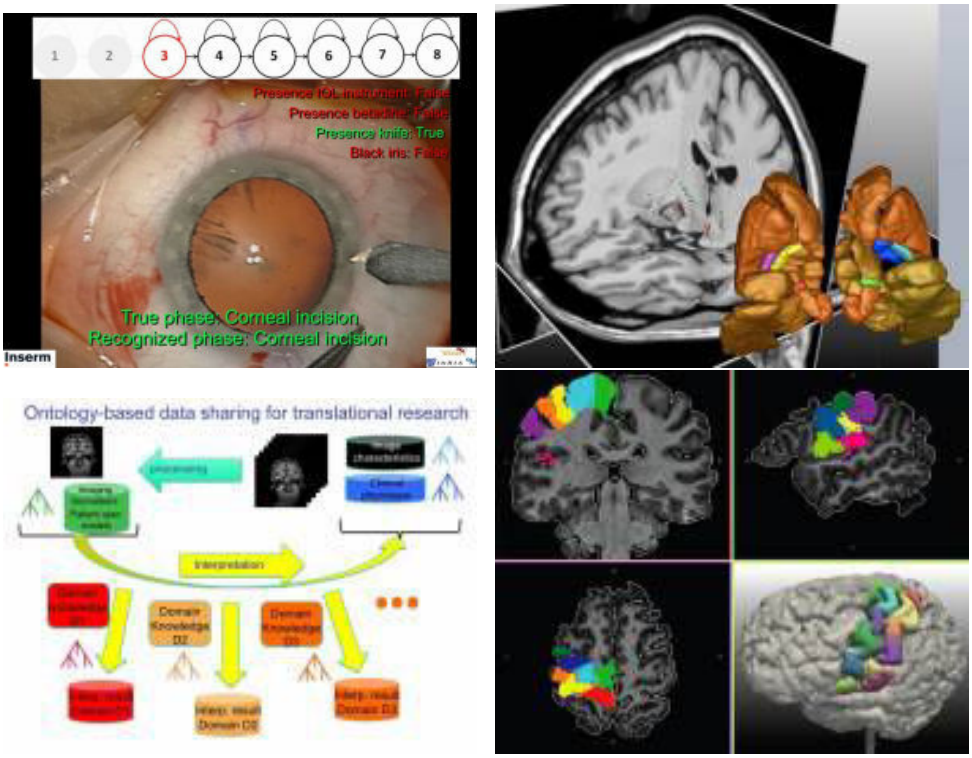


Figure 1.2: Research work examples pursued by the Medicis team (from the team website). Top row highlights the first thematic, second row represents the second thematic.

The first thematic refers to numerical approaches for building surgical knowledge and process model based on data fusion. Data fusion is the process of extracting and integrating relevant information from different sources of data to help the decision-making process. The data fusion challenge can be addressed from two aspects of patient data.
First, data representing planned surgical or interventional procedures. Secondly, data representing the patient himself through multimodal images and clinical scores. Registration of different views of the same phenomenon (i.e. of the same patient) is used to build specific models, whereas generic models are created through registration of data and information coming from a homogeneous population (i.e. different but similar patients).
The long-term objective is to develop methods able to generate predictive patient outcome models, generate surgical procedure models from patient-specific models,

compare surgical practice between surgeons with different expertise levels or from different surgical centers. For instance, in case of Deep Brain Stimulation (DBS) used to treat Parkinson disease, predicting patient motor improvement and clinical side-effects is highly valuable. Similarly, predicting the sequence of surgical steps and options in case of cervical spinal surgery (e.g. Anterior Cervical Discectomy and Fusion (ACDF)) is studied.

The second thematic focuses on ontological approaches for symbolic models. A common conceptualization for manipulated entities (e.g. anatomical structures, their specific role, surgical instruments) allows to articulate generic and specific models. This is crucial in the surgical and interventional decision making process. Symbolic knowledge modeling allows to express a consensus about a vocabulary and shared semantics, to exploit a formal representation in various contexts, and to express the knowledge in such a way it can be processed by both humans and automated systems.
Two major yet different topics are being studied, relying on ontology and other semantic web technologies. First, research to build a suitable architecture to share images and processing tools, more specifically in the neurosurgical context. Secondly, research on semantic annotation of brain anatomical structures in Magnetic Resonance Imaging (MRI) images, in order to demonstrate how ontologies can be used as knowledge source supporting knowledge-based image annotation software.

1.2 Scientific and medical context

Despite modern-life technological advances and tremendous progress made in surgical techniques including Minimally Invasive Surgery (MIS), today's OR is facing many challenges remaining yet to be addressed. In 2004, a workshop entitled "OR2020: Operating Room of the Future" gathered a hundred of experts including physicians, engineers, and medical staff around a complex topic: identifying existing OR challenges in order to define the general characteristics and integrated systems of the OR of the future [Cleary 2004].
Nowadays, patient safety in the OR remains a sizable issue as preventable medical errors occur frequently enough to cost tens of thousands of human lives per year in the USA only [Kohn 2000]. Surgeons, equally to any human being, are prone to making errors. Reducing the amount of errors can be provided by capable surgical systems able to at least perform intelligent patient monitor and at best understand the surgical workflow to prevent error making.
From the expected population growth, notably for the senior age group, a significant increase in demands for surgical services will arise. An actual alarming observation is however conflicting with the aforementioned estimation: there are personnel shortages in almost every component of the medical array from nurses, to surgeons or technicians. Two ways exist to cope with high demand and low supply situations: either medical staff training time are shorten in order provide operative medical

personnel as soon as possible, or the work-load of the remaining personnel is increased. Pursuing the objective to ensure the best patient outcome possible, none of them are correctly addressing the issue as less-qualified or overworked personnel are both patient endangering factors. Strategies need to be developed to meet this expanding workload while maintaining the best patient outcome possible.
Better OR management is also a major concern as being the most cost-intensive sector in the hospital. The optimization of workflow processes through the identification of inefficient, ineffective, and redundant procedures for scheduling and supply management is of particular concern. More-efficient allocations of medical resources is one response to the expanding workload problem while increasing patient outcome either by having rested medical staff or having enough medical devices available for each and every surgical procedure.
While some medical devices are already integrated with computer technology such as patient monitors, surgical robots and imaging equipment, they usually are not fully incorporated with clinical management and operational systems. Using advanced technology seamlessly integrated into the OR should provide better patient care.
Designing such OR of the future should provide a safer, smarter, time-efficient, and affordable medical care. Many challenges can be partly addressed through the development and integration of well-thought health-care specific systems

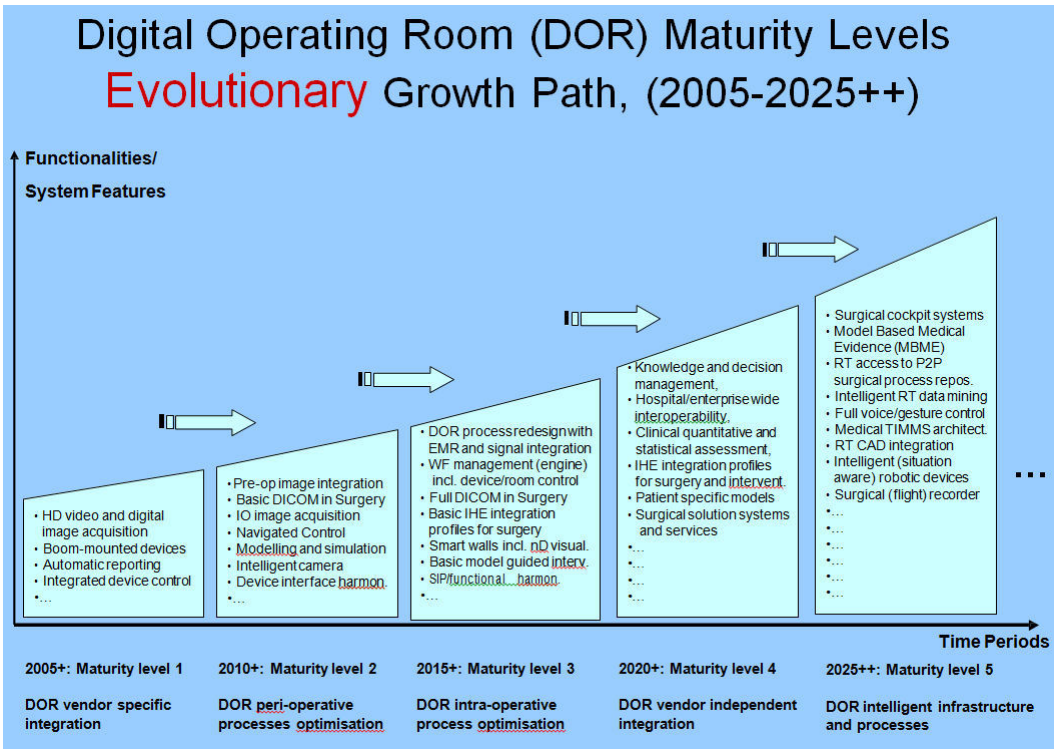


Figure 1.3: Digital operating room maturity levels (according to H.Lemke).

Redesigning healthcare infrastructures towards the integration of such systems is left facing the challenge to handle the complexity of the OR technology. Through the proliferation of diverse array of medical devices and information technology in today's OR, technical and organizational issues arise regarding inter-operability standards. Heinz Lemke[1] detailed expected OR maturity levels (illustrated figure 1.3) towards the achievement of full digital integration and efficient use of technology within the OR. Fully integrated operating rooms have already emerged, such as the OR1[2] enabling efficient operation, simple data recording, and unhindered data exchange within and without the OR.
As described by the fifth and last maturity level, the final objective is to fuse intelligent infrastructures and processes into a surgical cockpit system. Similarly to an airplane cockpit system, this system would be able to monitor in real-time everything happening in the OR and store all the information into black-box like devices. Medical errors could then be analyzed in retrospect and prevented through better digital OR designs. Per-operatively, such systems could centralize and display information coming from every sensor used in the OR, but could also assist the surgeon through robotic tasks or provide procedural advice thanks to its context-awareness. Warnings could be sent to the surgeon or the medical team when the system considers that the surgical procedure is going off-course. Having an easy and standardized access to data from all medical devices is the first core element for OR systems. To that end, the handling, transmitting, storing and creation of information in medical imaging has been standardized into the DICOM standard, which stands for Digital Imaging and COmmunications in Medecine. DICOM has been widely adopted by hospitals and enables the integration of scanners, servers, workstations, printers, and network hardware into a Picture Archiving and Communication System (PACS). Aside from medical data access and exchange, the second core element for surgical cockpit systems lies in the context-awareness, in other terms the capacity to process, analyze and understand in real-time everything happening in the OR. For the system to be able to discern if the surgical procedure is following its course accordingly or not, a typical road-map for the procedure should be provided. To that end, the Surgical Process Modeling (SPM) methology has been introduced in order to model and analyze the surgical practice through a standard terminology [Lalys 2014]. The term Surgical Process (SP) has been defined as "a set of one or more linked activities that collectively realize a surgical objective within the context of an organizational structure" [Neumuth 2007]. A SPM has been defined as "a simplified pattern of an SP that reflects a predefined subset of interest of the SP in a formal or semi-formal representation" [Neumuth 2007]. The formalization of terms employed to describe a SP is usually coming from expert consensus and represented in the form of ontologies [Gibaud 2014]. Generic surgical procedure models (gSPMs) can be learned from a collection of SP through data-mining and learning techniques in order to

[1] http://news.iscas.co/interoperability-standards-for-medical-device-integration-in-the-or-and-issues
[2] https://www.karlstorz.com/de/en/karl-storz-or1.htm

represent all possible transitions within SPs. The SPM methodology is organized around the concept of granularity levels, defining the level of abstraction at which the surgical procedure can be described. As proposed by Lalys et al. [Lalys 2014], the highest granularity level corresponds to the procedure itself. The procedure is composed of a list of phases defining the major types of events occurring during surgery. Each phase can be further subdivided into several steps representing a sequence of activities used to achieve a surgical objective. An activity represents a physical task and is commonly and minimally defined with a triple: an action verb, a surgical tool, and an anatomical structure. Eventually, the activity can be represented as a list of motions representing a surgical task involving only one hand trajectory without any semantics, being the lowest granularity level.

The emergence of CAS systems represents a glimpse of what may be ahead with surgical cockpit systems. Pre-operatively, they provide an access to patient-specific information through multi-modality images, enabling the preparation and simulation of a surgical scenario. During surgery, pre-operative data are registered in real-time and displayed on visualization interfaces to assist the surgeon in his decision-making process. They can also be aware of the current surgical situation in order to detect risk situations and adapt assistance accordingly [Sudra 2007, Katić 2014]. Post-operatively, CAS systems can be used for surgical procedure analysis, either for surgeon training, or for surgical practice comparison [Forestier 2013].
While the first generation of CAS systems mainly focused on providing the surgeon with access to medical information, new generations of Context-Aware Computer-Assisted Intervention (CACAI) systems integrate the SPM methodology for real-time understanding of surgical procedures. One important deadlock is the ability to detect in real-time what is happening in the OR and to match the result of this detection with the surgical knowledge formalized into the SPM in order to produce an estimation regarding the current position along the surgical workflow. Sensor-based approaches have been increasingly adopted to acquire relevant information about surgery at a low-granularity level without disturbing the flow of the intervention. In order to identify surgical activities, many image-based analysis approaches have been investigated [Lalys 2013]. Among the three elements minimally representing a surgical activity, performing the detection of the anatomical structure is the less discriminating one. However, being able to detect surgical tools will enable to fill the other two elements: the tool class, and the action verb corresponding to a semantic formalism of the trajectory described by a tool. As such, surgical tool detection is a key solution to help activity recognition.

1.3 Manuscript organization

The manuscript is organized in three main parts. The first one, from which this introduction is part of, aims to heighten reader awareness on the research topic tackled

in this work as long as provide a deeper understanding on automatic image-based surgical instrument detection through an extended review of the literature. The second part of the manuscript presents the contributions from this work, through an in-depth description of the exploited data, a deep explanation of proposed surgical tool detectors and an exhaustive presentation of the validation methodology as long as the corresponding results obtained. The last part of the manuscript introduces perspective work towards a robust integration of surgical instrument detection into real-time in-vivo medical applications, while presenting some preliminary qualitative results on phantom data. A global discussion and conclusion over the addressed topics are provided to wrap-up the manuscript.

Chapter 2

Surgical tool detection: from computer vision to medical field

Contents

2.1 Introduction

As presented in the previous chapter, the operating theater has been undergoing significant transformations over the past years. Operating rooms have evolved into highly complex and technologically rich environments with the incorporation of many sensors or medical devices. Computer technologies are now essential and increasingly used throughout the course of surgical interventions to process the important load of information.

Automatic understanding of surgical procedures represents the core issue for many per-operative surgeon guidance and assistance systems, or post-operative systems for example performing surgical videos fast browsing [Lalys 2013]. At the heart of CACAI systems, leveraging videos acquired from various OR sensors with computer technologies presents a strong challenge. Surgeon gestures describing the best a surgery itself, accurate detection of surgical tools throughout surgical procedures, combined with the surgical knowledge from the SPM methodology, is a key element necessary for CACAI systems.

The term of *computer vision* is employed to cover this technological field where numerical or symbolic information are produced in the form of decisions from processing, analyzing and understanding images. Detecting surgical tools in videos is bound to the wide topic of *object detection*. Related to both computer vision and image processing fields, it deals with detecting instances of semantic objects such as humans, buildings, or faces in digital images and videos. It encompasses object localization retrieving positions in the image, image segmentation for object classes without clear structures, and object categorization assigning the right label to each identified object instance.

To help create models of such object classes, *supervised learning* is often used, being a machine learning task of inferring a function from labeled training data (i.e. examples). Each training data is a pair consisting of an input object typically represented as a vector and a desired output value also called the supervisory signal. The algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.
Before entering operating rooms, the object detection topic has been widely addressed for applications in many areas including video surveillance, robotics, and automated car driving. As such, instead of directly diving into surgical tool detection, we start by investigating one of the most covered object detection instance: pedestrian detection (section 2.2). Then, we propose a methodological review of the literature focusing on surgical tool detection (section 2.3). To end this chapter, we discuss about both object detection instances in section 2.4 and conclude in section 2.5.

2.2 Objects detection in computer vision: pedestrian detection instance

Computer vision algorithms have been used to tackle many different challenging object detection issues. Amongst them, the pedestrian detection represents a canonical instance because of its many possible applications in car safety, surveillance, or robotics. Strong of more than 10 years of extensive studies, it is a well defined problem with established benchmarks and validation methodologies, and has served as a playground to explore different ideas for object detection. As such, we propose to scan the literature on pedestrian detection, which will later serve as a baseline for the surgical tool detection task.
We do not intend to perform an in-depth exhaustive review of the literature on this topic, but rather try to highlight the most important points. Information have been gathered from three main review papers on the subject [Dollar 2009b, Dollár 2011, Benenson 2014] and we encourage the reader to refer to those papers for more information on cited algorithms.
Any pedestrian detector is build around a specific detection strategy and has been validated over a specific data-set following a specific methodology. In order to understand and analyze pedestrian detector performance, a detailed identification and description of those three elementary bricks is paramount. The remainder of the section is organized accordingly, starting by a presentation of existing validation data-sets in section 2.2.1, then introducing detection methods in section 2.2.2, and finally describing validation methodologies in section 2.2.3.

2.2.1 Validation data-sets

Many public pedestrian validation data-sets have been published over the years; each time adding challenge and improving precision in results thanks to fine-grain annotations, hence being catalysts for progress. In order to understand their main characteristics, a detailed presentation of only some of the most commonly used, or most known data-sets, is performed. An overview of the considered data-sets is shown table 2.1, and illustrations from the Caltech-USA data-set[1] are reported figure 2.1.
Data-sets containing cropped pedestrian window only are known as 'classification' data-sets. They are primarily used for train/test binary classification scenarios (see upper part of the table). On the other hand, data-sets containing pedestrians in their original full images are known as 'detection' data-sets. They allow for the design and testing of full-image detection schemes (see bottom part of the table).
Below, data-sets are presented and compared based on the data collection strategy (in section 2.2.1.1) and the data annotation strategy (in section 2.2.1.2).

Figure 2.1: Caltech-USA data-set examples. Annotations: green bounding boxes for full pedestrians and dotted yellow bounding boxes for visible pedestrian parts.

2.2.1.1 Data collection

The data collection describes data acquisition strategies of raw video materials and their transformation into final benchmark data-sets. Each data-set can be further described by its quantified size (i.e. amount of data), by extended object location,

[1] http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/

Table 2.1: Non-exhaustive list of existing validation data-sets for the pedestrian detection task.

	Imaging setup		Training set		Testing set		Annotations		Properties		
	Photo	Mobile	# Positive	# Negative	# Positive	# Negative	# Training	# Testing	Video seq.	Temporal occ.	Occlusion labels
MIT [Papageorgiou 2000]	✔		-	-	-	-	924	-			
Daimler-CB [Geiger 2012]		✔	-	15k	-	10k	2.4k	1.6k			
NICTA [Overett 2008]		✔	-	5.2k	-	50k	18.7k	6.9k			
INRIA [Dalal 2005]	✔		614	1218	453	566	1208	566			
ETH [Ess 2009a]		✔	499	-	1804	-	2388	12k	✔		
TUD-Brussels [Wojek 2009]		✔	1092	218	508	-	1776	1498			
Daimler-DB [Enzweiler 2009]		✔	-	6.7k	21.8k	-	15.6k	56.5k	✔	✔	
Caltech-USA [Dollar 2009b]		✔	67k	61k	65k	56k	192k	155k	✔	✔	✔

scale, or shape statistics, or by content-specific properties. Below, we describe each element in details following the same order.

Data acquisition

The data acquisition process defines the imaging setup used to obtain the data, as long as strategies followed to extract still images from videos when necessary. Pedestrians have been mainly labeled in photographs [Papageorgiou 2000, Dalal 2005], in images taken from mobile device setup such as a robot or vehicle [Dollar 2009b, Ess 2009a], and occasionally in surveillance videos (in other data-sets). Generally, data coming from photographs suffer from selection bias due to pre-emptive manual selection, while surveillance data suffer from restricted background. As such, those means of recording rarely serve as basis for detection data-sets.

Continuous data collection from a mobile recording setup largely eliminates selection bias while providing moderate diversity from urban scenes, unless staged. In Caltech-USA, the largest benchmark data-set, a driver was asked to drive normally through neighborhoods of the large metropolitan area of Los Angeles.

Consequently, hours of video can be collected, potentially leading to millions of images. In order to remove side-effects coming from moving recording vehicles such as pitching, videos are stabilized. Finally, cropped video segments are extracted, representing a better alternative to stand-alone images because of pedestrian temporal consistency in the sequences. Additionally, video segments enable the use of new visual cues such as optical flow.

Data-set size

The size represents the amount of data composing the data-set, usually in the form of a number of images or video sequences. The amount of images in each data-set has largely increased over time, starting from a thousand images in MIT and leading up to hundreds of thousand images in Caltech-USA, being two orders of magnitude larger than most data-sets.

We can also note that nearly all data-sets are distributed into a training and a testing set, with an equal amount of images in both sets when possible (e.g. Caltech-USA).

Towards direct and unbiased comparison of pedestrian detection methods, a clear image distribution between training and testing sets is necessary.

Data-set statistics
Not reported in the table, an in-depth statistical data-set analysis is informative and serves to establish the requirements of a real world system, helping to identify constraints that can be used to improve automatic detection systems. For pedestrians, aside from instance counting and averaging per image, distributions of the following three elements have to be considered: scale, location, and occlusion. Depending on those statistics, some data-sets will be considered more challenging than others. For example, pedestrian image size can be grouped into three scales: near, medium and far [Dollár 2011]. Many detectors being designed for the near scale detection, they perform quite poorly at the medium scale.

Data-set properties
Additional properties can be used to provide at first glance a deeper level of details for a data-set without having to access its content. For example, it can be notified if the data-set is made of still images only or if complete video sequences are also provided. In case of video sequence data-sets, a property can be added regarding the availability of temporal correspondences between frames as part of annotations. Similarly, the annotation level of details can be summarized with one property such as the availability of occlusion labels.

2.2.1.2 Data annotation

The data annotation process, also known as 'ground truthing', consists in manually referencing positions of pedestrians in each and every frame of a data-set. For real outdoor recordings, ground truth positions can not be retrieved automatically as pedestrians are not equipped with any kind of tracking system.
Performing the data ground-truthing is a time-consuming but mandatory step for validation, but also for training purposes. Even with large amount of data, where potentially hundreds of hours might be required (e.g. 400h of effort for Caltech-USA), the annotation process must be done cautiously. Luckily, some appropriate annotation tools exist performing automatic position prediction in-between two frames manually annotated using cubic interpolation for example [Dollár 2011].
The most used annotation type is a bounding box (BB) indicating the full extent of a pedestrian, which might involve estimating hidden parts of occluded pedestrians. In such cases, the favored strategy is to add a second bounding box to delineate the visible part of the pedestrian only.
For each bounding box, a label is assigned: 'Person' for individual pedestrian, 'People' for a large group where it would be almost impossible to label individuals, or 'truncated' for occluded pedestrian [Everingham 2010]. Furthermore, within image sequences, the temporal correspondence between bounding boxes belonging to the same pedestrian can be saved (e.g. Caltech-USA).

The amount of annotated pedestrians per data-set has been increasing at the same pace as the data-set size, and is reported in the fourth set of columns of table 2.1.

2.2.2 Pedestrian detection methods

Research in pedestrian detection has been very profuse and quite diverse, with more than 40 proposed detection methods, also called detectors, in the last ten years. These detectors share many conceptual elements as they are based on the computation of image features used within a learning strategy to create object models, sometimes refined with the help of contextual information. Then, detectors apply learned models following a sliding window paradigm entailing dense multi-scale scanning followed by a Non-Maximum Suppression (NMS) procedure as part of the detection details specification.
As expressed by Dollar et al. [Dollár 2009c]: "The performance of object detection systems is determined by two key factors: the learning algorithm and the feature representation". This summarizes the two algorithmic challenges needing to be addressed in order to develop a robust, accurate and optimally fast object detection system.
Below, we introduce each principal component of existing detectors, including the feature representation (section 2.2.2.1), the learning strategy (section 2.2.2.2), the use of contextual information (section 2.2.2.3), and the detection details specification (section 2.2.2.4), all of which while providing some detector examples.

2.2.2.1 Feature representation

Features computed over the input image and aggregated into specific representations serve as basis for object-specific model learning and classification. While many different features or combination of features can be used, nearly all modern detectors rely on some form of Histogram Of Gradient (HOG) [Lin 2008, Dollár 2010, Benenson 2013]. Nevertheless, gradients can be directly leveraged [Sabzmeydani 2007], or grayscale features by computing Haar wavelets [Viola 2004]. Obviously, color features over various spaces have been experimented, mainly in the CIE-LUV color space [Dollár 2012, Mathias 2013]. Detectors can also utilize texture features such as Local Binary Patterns (LBP) [Ojala 2002] and co-occurence [Schwartz 2009], or self-similarity [Walk 2010] features. Finally, some detectors integrate motion features, such as the optical flow [Park 2013].
The results of such linear or non-linear transformations of the input image are called *image channels*, which notion can be traced back to the earliest days of computer vision. Given an input image, a corresponding channel is a registered map of the original image, where the output pixels are computed from corresponding patches of the input pixels, thus preserving overall image layout [Dollár 2009c]. An efficient way of computing channel features, called *integral channel features*, has been proposed by Dollar et al.[Dollár 2009c]. Features are extracted from each image channel using sums over local rectangular regions. Integral channel features com-

bine the richness and diversity of information from use of image channels with the computational efficiency of the Viola and Jones detection framework.
A number of papers have utilized integral channel features for different applications such as object recognition [Laptev 2006, Tu 2005], pedestrian detection [Dollár 2007], edge detection [Dollar 2006], and local region matching [Babenko 2007].

2.2.2.2 Learning strategy

As part of any machine learning approach, the learning strategy aims to leverage information provided by extracted image features to create object models. Support Vector Machines (SVMs) have been a popular choice since the original work of Dalal et al. [Dalal 2005]. While many linear kernels have been considered; nonlinear kernels are less common with the exception of the fast histogram intersection kernel [Maji 2008].
An important part of recent detectors has been relying on Decision Forests (DF) to perform the model learning [Zhang 2014, Benenson 2013]. This family of boosted classifiers has been introduced in [Viola 2004] and is of peculiar interest because of the automatic feature selection. They share with SVM classifiers many advantages such as a relative speed, theoretical guarantees, extensibility, and good performance.
Deformable Part Based (DPM) detectors, originally motivated for pedestrian detection [Felzenszwalb 2010], have been explored in many variants [Yan 2014, Park 2010].
Finally, deep architectures such as convolutional neural networks, using a mix of unsupervised and supervised training, have been considered (ConvNet [Sermanet 2013]). Another line of work proposes deep architectures to jointly model parts and occlusions [Ouyang 2013].

2.2.2.3 Exploiting context

Pedestrian detectors following a sliding window strategy, the content inside successive windows is used to score potential detections. However, when appearance alone is not enough, drawing on the context of the window provides additional information to disambiguate object classes. Pedestrian detection methods integrate for example context information into classifier inputs in the form of additional features [Wolf 2006].
Context can be defined as information relevant to the detection task not directly due to the physical appearance of the object [Wolf 2006]. Five classes of contextual relations between an object and its surroundings have been proposed by [Galleguillos 2010], with three of particular interest. Semantic context represents object co-occurrence, spatial context represents the relative position of the object in the image, and scale context represents the relative scale of the object within images.
To integrate contextual information, every element of an image can be thought as

a belonging of either the "things" or "stuff" meta-category [Heitz 2008]. "Things" represents compact objects with distinct shape properties (e.g. pedestrian), while "Stuff" includes elements with less structure or shape, and which can not be segmented with bounding boxes (e.g. road, grass, buildings). Both categories are represented using labeled pixels, and such a detection problem is formulated as a labeling problem (referred to as *semantic labelling*), see illustration figure 2.2.

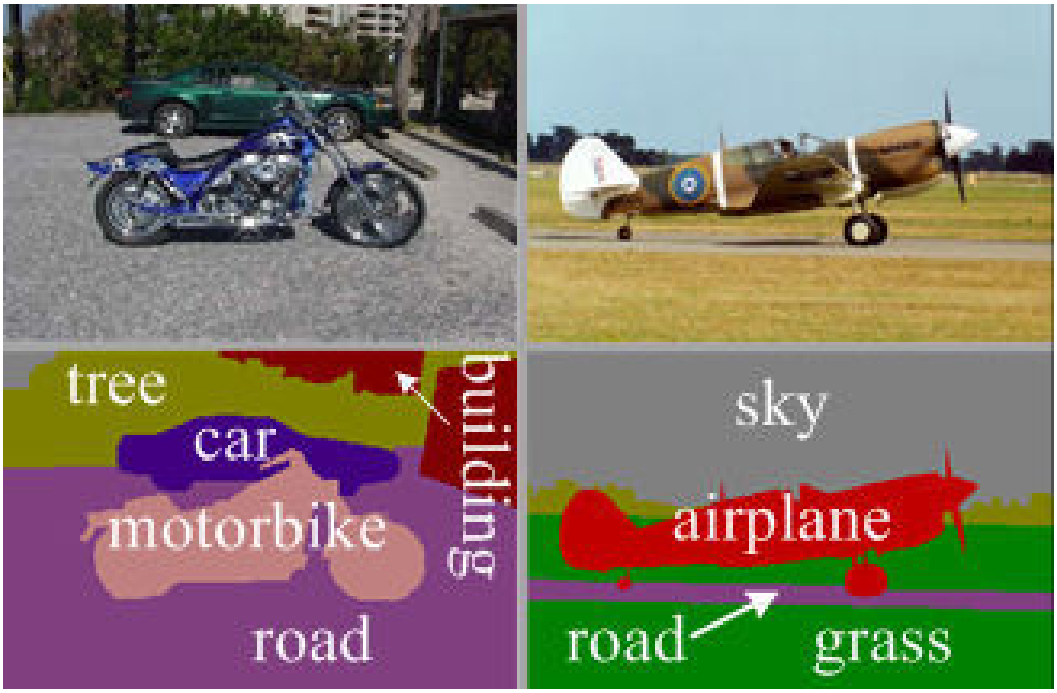


Figure 2.2: Semantic labelling illustration from the MSRC-21 data-set.

2.2.2.4 Detection details specification

Detection details determine parameters regarding the search space during the sliding window procedure and the post-processing of candidate detections. For multiscale detection, the standard approach has been to recursively scan multiple scales at multiple image resolutions (i.e. octaves), usually around 10-14 scales per octave. Typically, both high and low resolution candidate windows are resampled to a common size before extracting features. Recently, it has been noticed that training different models for different resolutions systematically improves performance since the detector has access to the full information available at each window size [Park 2010, Benenson 2013, Yan 2013]. Although training time might be increased, this approach does not impact computational cost at test time [Benenson 2012].
Regarding the Non-Maximum Suppression (NMS) approach, two dominant schemes can be identified: mean shift mode estimation [Dalal 2006] and pairwise max suppression [Felzenszwalb 2008]. The latter discards the less confident of every pair

of detections that overlap sufficiently (according to the PASCAL overlap criterion [Everingham 2010]), thus requiring only one parameter. In addition, a pairwise max variant has been proposed allowing a detection to match any sub-region of another detection, thus resulting in improved performance [Dollár 2009c].

2.2.3 Validation methodology

In order to quantify pedestrian detector performance and perform rankings in a realistic, unbiased and informative manner, a proper and well defined validation methodology is required. As a side-note, this methodology is also referred to as *evaluation methodology* in pedestrian detection studies, such as in this work from Dollar et al. [Dollár 2011]. However, we chose throughout this manuscript to follow the terminology proposed by Jannin et al. [Jannin 2006]. *Verification* consists in assessing that a method is built according to its specifications, *validation* consists in assessing that a method actually fulfills the purpose for which it was intented, and *evaluation* consists in assessing that the method is accepted by the end-users and is reliable for a specific purpose.
To describe such methodology, we first detail the object model learning strategy employed as long as validation protocols 2.2.3.1. Then metrics used to compute the validation are presented in section 2.2.3.2.

2.2.3.1 Specification phase

The model validation strategy employed to create object models is the first element needing to be specified to perform detection method validation. As a starter, we assume disposing of well-referenced data-sets with separate training and testing splits (resp. named d0 and d1), each one sub-dividable into different sequences approximately of the same size.
The following four training/testing scenarios have been proposed and encouraged by Dollar et al. [Dollár 2011]:

- Train on external data, test on d0.
- Train on external data, test on d1.
- Train on d0, test on d1.
- Perform k-fold cross validation, k being the number of sequences within either d0 or d1. For example, in each phase k-1 sub-sequences of d0 are used for training and the k^{th} for testing. Then, results are merged before being reported.

The first two scenarios allow a validation of existing and pre-trained pedestrian detectors, while the other two involve the use of a new data-set.
The second element to specify is the strategy used to collect and process detector results. A classic protocol to assess detector performance is to perform a *full image* validation. Over an image, a detector returns for each detection a Bounding Box

(BB) coupled with a confidence score, obtained after multi-scale detection and NMS procedure. The validation is then performed using the final list of detected bounding boxes (BB_{dt}) and the corresponding list of ground-truth/reference bounding boxes (BB_{gt}). A greedy matching is performed to resolve the assignment ambiguity, prioritizing detections with highest confidence scores. Thus, each BB_{dt} and BB_{gt} can be possibly matched at most once.
Occasionally, only portions of a data-set can be used for the validation by filtering ground truth or detector responses. For example, ground truth filtering can be used to remove ambiguous regions where pedestrian locations are unknown (e.g. crowds). Both ground truth and detector responses filtering are necessary to consider performing limited validations such as for pedestrian in a restricted scale range.

2.2.3.2 Computation phase

To compute and report detector performance, the standard performance metrics [Makhoul 1999] such as recall and precision values are computed but not exploited directly through recall/precision curves. Instead, miss rates are plotted against False Positives Per Image (FPPI) using log-log plots, by varying the threshold on detection confidence score. This approach is favored as typically there is an upper limit on the acceptable FPPI rate which is independent of pedestrian density. To summarize detector performance, a single reference value representing the entire curve can be used: the Log-Average Miss-Rate (LAMR). This value is computed by averaging miss rates at nine FPPI rates evenly spaced in the $[10^{-2}; 10^{0}]$ log-space range. In case of curves ending before reaching the given FPPI rate, the minimum miss rate is used. Curves being usually linear in the given FPPI range, the LAMR is generally similar to the performance at 10^{-1} FPPI, while providing more stable and informative assessment of performance.
Regarding the computation of such metrics, the following approach is used. After the per-image detection assignment, unmatched BB_{dt} are considered to be false positives and unmatched BB_{gt} count as false negatives. In order to identify true positives, a detected bounding box (BB_{dt}) and a ground truth bounding box (BB_{gt}) form a potential match if they overlap sufficiently. The criterion employed, based upon the PASCAL measure [Everingham 2010], states that their area of overlap must exceed 50%. For pedestrians, results are insensitive to the exact threshold as long as it is below 60% [Dollár 2011].

2.3 Surgical instrument detection

Applying computer vision techniques to medical data is still fairly new. As such, surgical field and data acquisition type ranges are rather large. Consequently, to perform this review of the literature regarding surgical tool detection, we did not limit the scope to the neuro-surgical field nor to surgical microscope data. We were interested in any work performing image or video based surgical tool detection, whichever the surgical context. Approaches relying on external markers are briefly mentioned as they will be addressed in details farther in the manuscript (see chapter 8).

2.3.1 Review Introduction

In order to ease the review and the discussions, we propose to describe and classify each study under three main categories, similarly to the ones proposed for pedestrian detection approaches: validation data-sets, detection methods, and validation methodologies. Each category has been subdivided into their corresponding most representative elements, as illustrated by the diagram shown in figure 2.3. The review is organized according to the diagram and each component is explained in details in the following subsections.
Reporting one extremely large table containing all the information gathered throughout each paper is not feasible. For clarity, we created one table for each main category, thus allowing more detailed information. Also for a space saving purpose, reported values larger than 1 000 are represented as 1k.
For each paper, when multiple data-sets, methods, or validation methodologies are introduced, as many table entries have been added.

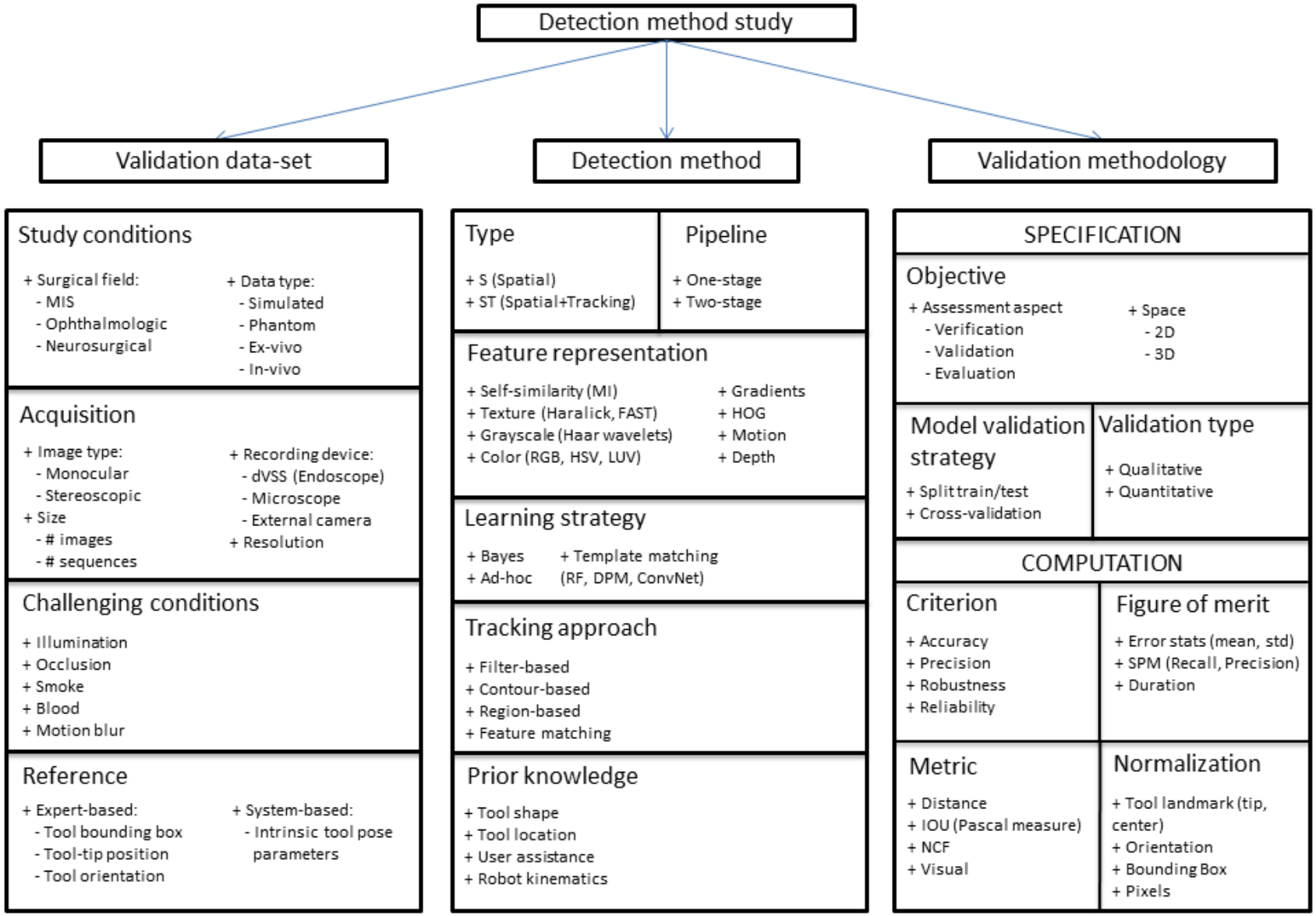


Figure 2.3: Tool detection literature review: overview graph.

2.3.2 Validation data-sets

To describe a data-set, we propose to rely on four types of information: the study conditions in which data have been acquired, the amount of data and its type, the range of challenging visual conditions covered by the data, and the type of annotation/reference provided. Almost each and every study has been relying on its own data-set, introduced with more or less details in the publications. Amongst those data-sets, only few are available online for consultation. As such, some table information may be missing or inaccurate depending on the level of details introduced in the corresponding publication and the online availability of the data-set. Table 2.2 provides an overview of used surgical tool data-sets.

Table 2.2: Overview of surgical tool validation data-sets considered in the literature review.

	Study conditions							Data acquisition						Data reference				Challenging conditions				
	Surgical field			Data type					Image type			Size		Manual								
	MIS	Opth.	Neuro.	Simulated	Phantom	Ex-vivo	In-vivo	Recording device	Monocular	Stereoscopic	Resolution	# Images	# Seq.	Bounding Box	Tip position	Pose parameters	System	Illumination	Occlusion	Smoke	Shadows	Motion blur
[Allan 2013]	✔ ✔ ✔ ✔			✔	✔	✔	✔	- - - -	✔	✔	720 × 288	- 100 500 97	15 1 1 6	✔			✔ ✔ ✔					
[Burschka 2005]	✔	✔			✔ ✔			- -		✔ ✔	- -	<500 -	1 1				✔ ✔					✔
[Doignon 2007]	✔ ✔				✔		✔	End. End.	✔ ✔		- -	30 52	1 1					✔					
[Haase 2013]	✔ ✔					✔ ✔		End. ToF End. ToF		✔ ✔	640×480 64 × 50 640×480 64 × 50	30 4	2 4		✔ ✔				✔			
[Kumar 2013b]	✔						✔	dVSS	✔			3k2	16	✔				✔	✔	✔	✔	✔
[McKenna 2005]	✔						✔	End.	✔		720×576	>835	2	✔								
[Pezzementi 2009]	✔ ✔	✔		✔	✔		✔	PC Mic. dVSS		✔ ✔ ✔	640 × 480	>100 >65 >175	1 1 1									
[Reiter 2010]	✔						✔	-				>137	1					✔				✔
[Reiter 2012a]	✔						✔	dVSS		✔	-	>1k6	5	✚				✔				
[Reiter 2012b]	✔						✔	dVSS		✔	-	>1k6	5	✚				✔				
[Richa 2011]		✔ ✔			✔		✔	Mic. Mic.		✔ ✔	1k6 × 1k2	1k1 -	2 2					✔	✔			✔
[Speidel 2006]	✔						✔	End.	✔		-	>60	1	✔								
[Speidel 2008]	✔ ✔					✔ ✔		End. End.		✔ ✔	768 × 576	560 100	6 100	✔	✔ ✔			✔	✔			
[Speidel 2014]	✔						✔	End.		✔	320×240	542	6		✔				✔	✔		
[Speidela 2013]	✔ ✔ ✔			✔		✔	✔	End. End. End.		✔ ✔ ✔	- - -	151 80 76	4 1 1				✔ ✔					
[Sznitman 2012]	✔	✔					✔	Mic. -	✔ ✔		640×480 -	1k5 1k	4 1	✔	✔			✔	✔		✔	✔
[Sznitman 2013]		✔ ✔			✔		✔	Ext.	✔ ✔		1k6 × 1k2	>400 850	1 1		✔ ✔	✔ ✔		✔ ✔		✔	✔ ✔	✔ ✔
[Sznitman 2014]	✔		✔				✔ ✔	- -	✔ ✔		640×360 320×240	1390 472	2 1	✔ ✔								✔ ✔
[Voros 2007]	✔ ✔					✔	✔	End. End	✔ ✔		200 × 100	6 45	6 1		✔			✔	✔	✔		
[Wolf 2011]	✔ ✔			✔	✔			- End.	✔ ✔		- 768×576	1050 1050	1 1			✔ ✔	✔ ✔					

2.3.2.1 Study conditions

The study conditions component aims at describing the surgical context of each study through characteristics including the assessment scenario, location, and environment, as described in Jannin et al. [Jannin 2008]. To cover the identification of the surgical context, we report the surgical field as assessment scenario and the data type as assessment environment.

Surgical specialty

Surgical tool detection has been applied to different surgical specialties, but minimally invasive surgeries including endoscopic and laparoscopic procedures have been mainly focused. Surgical examples are cholecystectomy [McKenna 2005], nephrectomy [Reiter 2010], hysterectomy [Kumar 2013b], pelvic [Sznitman 2014]. For the most part performed on humans, some data have been collected on porcine experiments [Pezzementi 2009, Reiter 2012a].

Far behind, ophthalmology is the second most studied surgical field, especially with retinal micro-surgeries [Burschka 2005, Pezzementi 2009, Richa 2011, Sznitman 2012, Sznitman 2013, Sznitman 2014]. Finally, one study only from Sznitman et al. [Sznitman 2014] has performed tool detection over neurosurgical data.

Data type

The data type component is used to position the data-set along the control versus clinical realism continuum [Jannin 2008]. Four data acquisition types can be identified: simulated, phantom, ex-vivo, and in-vivo. The simulated type represents one end of the continuum with full control and no clinical realism while the in-vivo type represents the other end of the continuum with no control and full clinical realism.

Simulated data are virtual and obtained from fully controlled environment where tool models and surgical backgrounds can be mixed at will. A simple black tool mask moved on top of a surgical background [Wolf 2011], rendered tool models undergoing translation and articulation against a black background [Pezzementi 2009] or a virtual endoscope moving over a liver with homogeneous texture [Speidela 2013] are representative examples.

Phantom data are a real-world equivalency to simulated data where real surgical instruments are usually moved in front of phantom surgical backgrounds. They can describe very simple setups such as a real tool moving in front of a white background [Allan 2013] or in front of a real surgical background image [Wolf 2011]. Phantom backgrounds can be a little bit more complex, such as a half-sphere painted to resemble the retinal surface in an ophthalmic surgical context [Sznitman 2013]. Even real phantom organ models can be used to be as close to real clinical conditions as possible [Speidel 2008].

For ex-vivo data, few cases have been reported, with experiments on lamb liver tissue sample [Allan 2013], anatomical structures [Speidela 2013], or cadavers [Voros 2007].

In-vivo data are the most represented, nearly used in every study. More details about the range of challenging situations captured in such data are reported in

section 2.3.2.4.

2.3.2.2 Data acquisition

The second component aims at describing the data acquisition strategy through the type of recording device employed, the image type and resolution, and the data-set size either as number of images or number of video sequences.

Recording device
Data recording devices are intrinsically connected to studied surgical fields as consumer cameras can not be brought into operating rooms easily and even less positioned at will due to patient safety concerns. The recording device element is not relevant for simulated data-set where only a computer is needed (noted as *PC* in the table).
For MIS, the go-to surgical device commonly used is the daVinci Surgical System (dVSS), equipped with an endoscope as recording device (noted as *End.* or *dVSS* in the table). In ophthalmology and neuro-surgery, most surgeries are performed under a surgical microscope capable of recording videos (noted as *Mic.* in the table). Sznitman et al. [Sznitman 2013] managed to couple a consumer camera to a microscope (noted as *Ext.* in the table). While only Haase et al. [Haase 2013] proposed a different recording setup, using a 3D endoscope coupled with a Time-of-Flight (ToF) camera.

Image type
Depending on hardware capabilities of recording devices, two image types are available: monocular and stereoscopic.
Monocular represents single images, enabling to retrieve 2-dimensional (2D) positions in the image referential only and is the most represented image type [Sznitman 2012, Sznitman 2013, Sznitman 2014, Kumar 2013b, Wolf 2011, Voros 2007, Doignon 2007].
Stereoscopic represents a pair of monocular images, namely a left one and a right one, enabling to retrieve depth estimates using epipolar geometry. They have been used in [Pezzementi 2009, Reiter 2012a, Richa 2011, Burschka 2005, Speidel 2014, Speidel 2008].

Image resolution
A high variability can be noted in reported image resolutions, ranging from low resolution (e.g. 200×100 pixels [Voros 2007]) to high resolution (e.g. 1600×1200 pixels [Richa 2011, Sznitman 2013]) images.

Data-set size
The amount of data, represented as a number of images, can be expressed in different levels of magnitude: small, medium and large, with respect to the largest data-set used in the literature [Kumar 2013b].

Small data-sets contain less than a hundred images [Speidel 2006, Doignon 2007, Haase 2013], while medium data-sets range from a hundred to a thousand images [McKenna 2005, Allan 2013, Sznitman 2013]. Finally, large data-sets incorporate more than a thousand images [Richa 2011, Sznitman 2012, Reiter 2012a, Kumar 2013b].
The variability within a data-set is also representative and is represented by the number of video sequences from which images originated. Most of the studies, especially early works in the field, present a data-set made of one video sequence only [Burschka 2005, Doignon 2007, Pezzementi 2009, Reiter 2010, Wolf 2011, Sznitman 2013]. However, recent works are proposing more than 5-6 sequences thus introducing a little bit more diversity in the data pool [Allan 2013, Reiter 2012a, Kumar 2013b, Speidel 2014].

2.3.2.3 Data reference creation

The final objective being to estimate surgical tool positions in images, having a reference for said positions, assumed to be close to the correct result [Jannin 2006], is necessary. They can be obtained in two ways: either manually or automatically. The manual approach being widely employed as not requiring the installation of additional sensors.
The favored approach to obtain automatic annotations is to use an Optotrack optical localizer [Burschka 2005, Wolf 2011, Allan 2013]. For simulated data-sets, tool pose parameters are inherently known by the computer setting up the simulation [Wolf 2011, Speidela 2013].
Regarding manual annotations, most of the time tool-tip positions are involved [Sznitman 2012, Wolf 2011, Sznitman 2013, Voros 2007, Haase 2013, Speidel 2008, Speidel 2014], along with bounding boxes around surgical tools [Sznitman 2012, Sznitman 2014, Kumar 2013b, Wolf 2011, Speidel 2006] or parts of surgical tools [Reiter 2012b] (represented in the table by ✚). Occasionally, variants of bounding boxes are used [McKenna 2005], or extended pose parameters such as tool orientation, length, or entry point in the image [Sznitman 2013].

Tools statistics
In addition to the number of different video sequences, a second level of variability involves the number of different surgical tools and their occurrences in the data-set. Those information being only scarcely accessible, they are not displayed in the table. However, we propose to report here what we managed to gather.
Regarding surgical tools diversity, studies were mostly focusing on tubular-shaped tools such as a cylindrical needle-holder [Doignon 2004], forceps [Pezzementi 2009], a large needle driver [Reiter 2012a], or standard tools from dVSS [Kumar 2013b]. Many times, only the generic term of "endoscopic tools" or "tools" is mentioned [Haase 2013].
Some data-sets only feature one surgical instrument, especially in phantom conditions [Doignon 2007, Wolf 2011]. Two simultaneous tools are widely featured,

especially in a context of MIS [McKenna 2005]. More than two surgical tools is very unlikely, mostly because of the nature of minimally invasive surgeries performed using a dVSS and displaying a maximum of two tools at the same time. Only Speidel et al. [Speidel 2014] proposed a data-set with up to three tools simultaneously visible for a total of four different tools.
Unfortunately, tool occurrences, overlapping occurrences, orientation, number of simultaneous tool distributions, or any kind of extended statistics, are not available in any paper.

2.3.2.4 Challenging conditions

The third and last level of data variability corresponds to the range of challenging conditions captured. Data-sets may cover a wide range of appearance and lighting scenarios [Reiter 2012a, Sznitman 2012, Reiter 2010], include occlusions [Speidel 2014], rapid appearance changes [Reiter 2010], smoke [Kumar 2013b, Sznitman 2013, Speidel 2014], specular reflections [Kumar 2013b], shadow [Sznitman 2012, Sznitman 2013], blur [Sznitman 2012, Kumar 2013b] or blood spatter [Haase 2013]. While sometimes data-sets are explicitly not covering any challenging situations [McKenna 2005].

2.3.3 Tool detection methods

While pedestrian detectors share many elements and typically follow a similar workflow, it is less clear for surgical tool detectors. One fundamental difference comes from the use of external markers to ease the detection, for example color tags [Tonet 2007], light emitting diodes [Krupa 2003], or RFID tags [Miyawaki 2009]. In this review, we wanted to focus on approaches not requiring to apply physical modifications to surgical tools, thus leaving aside marker-based approaches for later consideration (see chapter 8).
From here, existing tool detectors may be grouped into two types: 'S' detectors performing full-image analysis and 'ST' detectors performing sub-image analysis by leveraging detection locations from previous frames. Both of them needing a full-image analysis scheme, the latter only to initialize or re-initialize the tracking procedure. Additionally, we inquire about prior knowledge used as core element of detectors' algorithmic design, which may lead to detectors not being suitable for other surgical instruments or surgical contexts. Below, we present each component, including the feature representation, the spatial pipeline form, the learning strategy, tracking solutions, the use of prior knowledge, and conclude with optimization strategies regarding detection speed. Table 2.3 gives an overview of each surgical tool detection method.

Table 2.3: Overview of surgical tool detection methods considered in the literature review.

		Features								Learning			Pipeline		Prior knowledge				Tracking approach			
	Type	Gradients	HOG	Grayscale	Color	Texture	Self-Similarity	Motion	Depth	Ad-hoc	Probabilistic	Template matching	One-stage	Two-Stage	Tool shape	Tool location	User assist.	Kinematics	Filtering	Contour	Region-based	Feature matching
[Allan 2013]	S		✔		✔	✔						✔		✔	✔	✔						
[Burschka 2005]	S ST	✔	✔		✔					✔		✔	✔	✔	✔			✔	✔			
[Haase 2013]	S	✔			✔				✔	✔			✔		✔	✔						
[Kumar 2013b]	ST		✔			✔		✔				✔	✔								✔	✔
[McKenna 2005]	ST				✔						✔			✔	✔	✔			✔			
[Pezzementi 2009]	ST				✔	✔					✔			✔	✔							
[Reiter 2010]	ST				✔	✔					✔		✔		✔		✔					✔
[Reiter 2012a]	ST				✔	✔					✔		✔					✔	✔			
[Reiter 2012b]	S	✔			✔	✔					✔	✔		✔	✔	✔		✔				
[Richa 2011]	ST						✔	✔				✔									✔	
[Speidel 2006]	ST				✔						✔		✔			✔				✔		
[Speidel 2008]	S				✔				✔	✔	✔			✔	✔							
[Speidel 2014]	ST				✔			✔	✔		✔		✔			✔			✔			
[Sznitman 2012]	ST	✔										✔	✔								✔	
[Sznitman 2013]	ST S			✔ ✔	✔ ✔							✔ ✔	✔ ✔								✔	
[Sznitman 2014]	S	✔										✔		✔	✔							
[Voros 2007]	S	✔		✔						✔			✔		✔		✔					
[Wolf 2011]	ST	✔		✔						✔			✔		✔		✔			✔		

2.3.3.1 Feature representation

The first set of columns in table 2.3 indicates feature types used by each detector. Features are reported as general category of image content extracted and not as particular instantiation. It appears nearly all detectors extract a combination of some form of color features, mostly from the classic RGB space [McKenna 2005, Reiter 2010, Sznitman 2013] and the Hue Saturation Value (HSV) space [Speidel 2006, Speidel 2008, Pezzementi 2009]. Unconventional color spaces such as CIE, XYZ, O2, O3 have been proposed in Allan et al. [Allan 2013]. Fairly few detectors rely on variants of HOG features [Burschka 2005, Allan 2013, Kumar 2013b], preferring to leverage gradients directly such as image derivatives [Wolf 2011] or through Sobel filtering [Haase 2013]. Texture features are also a frequent cue for tool detection. Allan et al. [Allan 2013] proposed to employ interest point extractors (e.g. SIFT, Color SIFT) to represent 2D texture around local patches in the image, while Reiter et al. [Reiter 2010] chose to rely on FAST corners. In [Pezzementi 2009], the texture of the image is measured via co-occurrence matrices and represented using a sub-set of Haralick coefficients.
Amongst the least represented categories, detectors can utilize grayscale (e.g. Haar wavelets [Sznitman 2013], Hu moments [Voros 2007]), self-similarity (e.g. mutual information [Richa 2011]) and depth [Speidel 2008, Haase 2013, Speidel 2014] features.
Lastly, another important cue for human perception, which has almost never been incorporated successfully, is motion. In a recent study from Speidel et al. [Speidel 2014], a motion disparity map is built and fused with traditional cues.

On the other hand, Kumar et al. [Kumar 2013b] proposed to compute dense optical flow for tracking purposes.

2.3.3.2 Detection pipeline

Overall, out of the 20 considered tool detectors, we can identify two families of pipeline architecture employed to perform spatial detection: one-stage [Kumar 2013b, Sznitman 2012] and two-stage [Pezzementi 2009, Allan 2013]. One-stage surgical tool detectors are very similar to standard pedestrian detectors in a sense, where tool pose is directly estimated from input image feature channels. This category covers detectors computing tool model responses over the input image, potentially in a multi-scale search with candidate detections refined by a NMS step [Kumar 2013b]. In addition, it includes detectors performing multiple steps always solely over input image channels. For example, a four-step method of mathematical morphology operations in [Voros 2007], or a three-step method encompassing clustering and Hough fitting operations in [Haase 2013].
Two-stage surgical tool detectors can be assimilated to pedestrian detectors exploiting contextual information. The first stage consists in a multi-class pixel-wise classification (a.k.a. *semantic labelling*) of the input image using the feature channels. Usually two classes are modelled, one to represent tool pixels and one to represent background pixels [McKenna 2005]. Three classes [Allan 2013] or more [Reiter 2012a] have also been tested. The second stage corresponds to the tool pose estimation, similar to the aforementioned one-stage, the main difference being the use of semantic label maps (i.e. result of the first stage) as input instead of the image feature channels. Usually performed in this order, context can also be used as second stage, in order to refine candidate detections provided by a pose estimation first stage [Reiter 2012b].

2.3.3.3 Learning strategy

Throughout the studies, we have identified three groups of learning strategies employed to perform spatial detection: probabilistic classifiers such as Naive Bayes, a meta-group of template matching strategies such as cascade classifiers, and ad-hoc strategies through empirically defined thresholds.
Largely employed for the multi-class pixel-wise classification, simple probabilistic classifiers are based on applying Bayes' theorem with naive independence assumptions between the features. Aside from the Naive Bayes classifier itself [Speidel 2006, Speidel 2008], many related variants have been proposed. For example, probability density function [Speidel 2014], likelihood posterior probability [McKenna 2005, Reiter 2010], or Gaussian Mixture Models (GMMs) [Pezzementi 2009, Reiter 2012b].
The second category relates to object model learning usually intended for one-stage approaches. Kumar et al. [Kumar 2013b] used a Latent SVM (LSVM) to perform the pose estimation using a tool model, while Burschka et al. [Burschka 2005]

used a standard template matching approach. An interesting template matching strategy using an Active Testing approach has been proposed by Sznitman et al. [Sznitman 2013]. In addition, some approaches proposed to perform part-based modelling for example using a star-structured model [Kumar 2013b]. Conversely, Allan et al. [Allan 2013] proposed to use Random Forests to perform the pixel-wise classification, and Sznitman et al. [Sznitman 2014] the Gradient Boosting framework.
We can mention a third type of approaches based on empirically defined thresholds even though they do not represent learning strategies strictly speaking. In most cases, such thresholds are necessary for mathematical morphology operations [Voros 2007], Hough line-fitting [Doignon 2004] or connected components identification [Haase 2013]. Sometimes, empirically defined thresholds are not even needed, approaches being entirely dependent on relative maximum values [Wolf 2011].
Whilst all aforementioned strategies require models to be trained prior to performing tool detection, an online learning strategy has been proposed by Reiter et al. [Reiter 2010].

2.3.3.4 Tracking approach

Roughly half of the studied tool detectors couple the spatial pose estimation with a tracking approach in order to constrain the search space by re-using the knowledge of previous detections. All of the most common tracking categories have been represented in the literature, and are introduced by order of importance.
Filtering tracking has been the preferred category of approaches, regrouping Particle filter useful for sampling the underlying state-space distribution [McKenna 2005, Speidel 2014] and Kalman filter [Burschka 2005, Reiter 2012a].
Variants of *Region-based tracking*, based on the minimization of a similarity measure, have also been widely considered. For example, the Sum of Squares Difference (SSD) was adopted as similarity measure in [Sznitman 2012], Richa et al. preferred to rely on the Mutual Information (MI) [Richa 2011], while Sznitman et al. proposed to use the Sum of Conditional Variance (SCV) as objective function [Sznitman 2013]. Such tracker can also be based on extracting dense optical flow [Kumar 2013b].
Coming after, we can find *contour tracking* performing detection of object boundary, with CONDENSATION being the most representative algorithm [Speidel 2006, Wolf 2011].
Feature matching has been examined, using for instance the Normalized Cross Correlation (NCC) over FAST corners [Reiter 2010], or the Kanade-Lucas-Tomasi (KLT) point feature tracker [Kumar 2013b].
An interesting line of work, proposed by Kumar at al. [Kumar 2013b] dwells in an optimal fusion between outputs from various trackers. Hence combining, for example, feature-based trackers robustness to small motion and region-based trackers robustness to significant motion.

2.3.3.5 Prior knowledge

In order to constrain the detection search space, thus facilitating the task, many approaches rely on a set of assumptions, or prior knowledge (fourth set of columns table 2.3). Such knowledge having different forms and aspects, we chose four categories for its representation: assumption over tool location within the image, assumption over tool shape, user assistance, robotic assistance. A fifth befitting category has already been discarded from the review: markers (refer to the review introduction section 2.3.1). In the following, an overview of each category and some examples are provided.

Tool shape constraints

Assumptions over the shape have been widely employed to design detectors. Low-level considerations regarding tools as simple tubular shapes [Speidel 2008, Sznitman 2014] or solid cylinders with a tip alongside the center-line [Allan 2013, Burschka 2005, Haase 2013] have been used.

Similarly, rough estimates of a tool shape have been expressed, either being represented by two edges symmetrically spaced from an axis-line containing the tip [Voros 2007] or by two parallel side segments and a tip lying in-between [McKenna 2005].

On the other hand, highly detailed shapes with joint configurations can be leveraged from a rendering software [Reiter 2012b, Pezzementi 2009].

Tool location constraints

The second most common type of assumption relates to tool appearance and disappearance from the Field-of-View (FoV). In other words, surgical tool intersection conditions with image boundaries.

Surgical instruments are expected to enter the scene (i.e. FoV) from image boundaries [McKenna 2005, Allan 2013, Haase 2013], thus being visible on image edges [Sznitman 2013].

Sometimes the constraint is expressed within the processing algorithm, where the corresponding initialization is performed by looking exclusively at image border areas [Speidel 2006], or by choosing candidate regions close to image boundaries [Doignon 2004].

User assistance

Instead of relying on generic assumptions over a tool shape or its location within the image, some methods request a manual help from the user.

For MIS, the knowledge of the instrument insertion point in the cavity greatly constraints the search space to a limited beam. The insertion point can be selected by the surgeon using a vocal mouse [Voros 2007] or after computation requiring manual selection of 2D instrument boundaries in a sequence of images [Wolf 2011].

In case of online learning algorithms, a user may also have to indicate to the system which image portions are containing surgical tools needing to be subsequently

identified [Reiter 2010].

Robotic assistance
Size and kinematic complexity of dVSS robots leads to relatively inaccurate tool pose estimations supplied by internal encoders. At the same time, such inaccurate positions can be seen at good estimates to constraint the search.
Robot kinematics data can be used as input to the detector [Burschka 2005], or to render on-the-fly tool models with a limited set of joint configurations (i.e. different tool poses) [Reiter 2012b]. Lastly, robot kinematics can be used in a post-processing manner to reject erroneous detections or fill the gap of missed ones [Reiter 2012a].

2.3.3.6 Optimization strategies

For integration in real-time in-vivo medical applications, highly accurate tool detectors are key, but not at the detriment of the processing speed. Finding the optimal position along the speed versus accuracy trade-off is usually difficult. Computer hardware specifications, code optimization and image resolution have a significant impact on speed performances. Because of such variability, we chose not to integrate this information into the table as a direct speed comparison between detectors would not yield much sense. However, we propose to report interesting optimization strategies mentioned by authors to increase the computational speed.
For detectors using a sliding window approach, the most popular ad-hoc optimization implementation is to reduce the number of pixels to process. It can be achieved by performing spatial down-sampling (by a factor 2 to 4) over input images [Voros 2007, Pezzementi 2009], or by processing every fourth line and column [Speidel 2006].
When processing video inputs, not every frame needs to be processed, assuming a recording speed between 25 and 30 Hz, because of the limited motion of surgical tools within consecutive frames. Speidel et al. [Speidela 2013] proposed to process every fifth frame, while Reiter et al. [Reiter 2010] processed every third frame.
Limiting cascade classifiers (such as Random Forests) parameters, especially tree length and depth, to a minimum has been proposed to increase computational speed. Early stopping scheme [Sznitman 2014] or manual limitation [Allan 2013] have also been proposed towards this effect.
Finally, for brute-force approaches requiring to process large amounts of pose-specific models, a coarse-to-fine approach can remedy the huge processing time issue [Reiter 2012b].

2.3.4 Validation methodology

As mentioned in the corresponding section for pedestrian detectors, a well described validation methodology is necessary to quantify detector performance and perform rankings in a realistic, unbiased and informative manner. To do so, we propose to investigate existing tool detection validation methodologies through their specifica-

tion phase (high-level) and computation phase (low-level). In the former, we explore the assessment objective, the validation type and the model validation technique. In the latter, we examine validation criterion and its estimation by focusing on figures of merit, validation metrics, and normalization steps, using the same terminology as specified by Jannin et al. [Jannin 2006]. Table 2.4 illustrates collected information about the validation methodologies. In the following, both categories are presented in details.

Table 2.4: Overview of surgical tool detection validation methodologies considered in the literature review.

	Specification							Computation										
	Aspect			Model		Type		FOM			Normalization				Metric			
	Type	2D	3D	Split	Cross-Validation	Qualitative	Quantitative	Error stats.	SPM	Duration	Landmark	Orientation	Bounding Box	Pixels	Distance	IOU	NCF	Visual
[Allan 2013]	Verif. Valid. Valid. Valid. Valid.	 ✔ 	✔ ✔ ✔ ✔	 	 ✔ ✔ 	 ✔ 	✔ ✔ ✔ ✔	M,S 	 R,P,PE R,P,PE 	 	✔ ✔ End	✔ ✔ 	 ✔ 	 ✔ 	✔ ✔	 ✔ ✔ 	 	 ✔
[Burschka 2005]	Valid. Valid.	✔ 	 ✔	 	 	✔ ✔	 	 	 	 	✔ Tip	 	 	 	 	 	 	✔ ✔
[Haase 2013]	Valid. Valid.	✔ 	 ✔	 	 	 	✔ ✔	M,O M,O	 	 	Tip Tip	 	 	 	✔ ✔	 	 	
[Kumar 2013b]	Valid.	✔			✔		✔		A				✔			✔		
[McKenna 2005]	Valid.	✔					✔	M				✔			✔			
[Pezzementi 2009]	Valid. Valid.	✔ ✔					✔ ✔	 M	P,R,PE 		 ✔	 ✔		✔ 	 ✔	✔ 	 	
[Reiter 2010]	Valid. Valid.	✔ ✔				 ✔	✔ 			✔	 End						✔ 	 ✔
[Reiter 2012a]	Valid.	✔		✔			✔		A		Center							✔
[Reiter 2012b]	Valid.	✔					✔		R		✔	✔						✔
[Richa 2011]	Valid. Eval.	✔ ✔				✔ 	 ✔				✔ Tip				 ✔			✔
[Speidel 2006]	Valid.	✔				✔					✔							✔
[Speidel 2008]	Valid. Valid.	✔ ✔					✔ ✔	M			Tip Tip				✔ ✔			
[Speidel 2014]	Valid. Valid.	✔ ✔					✔ ✔	 M	R,P		 Tip			✔ 	 ✔	✔ 		
[Sznitman 2012]	Valid. Valid. Valid.	✔ ✔ ✔		✔ ✔	 ✔ 		✔ ✔ ✔	 M	R R 		Tip Tip 				✔ ✔ 		 ✔	
[Sznitman 2013]	Verif. Valid. Valid. Valid.	✔ ✔ ✔ ✔		 ✔ ✔ ✔		✔ 	 ✔ ✔ ✔	 M,S 	 TPR,FPR,P 		 Tip Tip Tip	 ✔ 			 ✔ ✔ 		 ✔	✔
[Sznitman 2014]	Valid. Valid. Valid.	✔ ✔ ✔		✔ ✔ ✔			✔ ✔ ✔	 M,S M,S	R 		Center ✔ Center	 ✔ 			✔ ✔ ✔			
[Voros 2007]	Valid. Valid.	✔ ✔				✔ 	 ✔	 M	 		Tip Tip				 ✔			✔
[Wolf 2011]	Valid. Valid. Valid.	✔ ✔	 ✔ 				✔ ✔ ✔	M,S M,S M			 Tip	✔ ✔ 			✔ ✔ ✔			

2.3.4.1 Specification phase

The specification phase of the assessment methodology defines the conditions in which the assessment is being performed with a clearly formulated assessment objective and type. Most of the studies performing validation, as detailed in the next paragraph, the term of validation is used to refer to the methodology and relative components.

Assessment objective

In each and every existing work, the validation performance assessment has been put forth, consisting in assessing that the tool detection method actually fulfills the purpose for which it was intended [Jannin 2006]. Most of the time, the quality of the detections is being studied [Sznitman 2012, Kumar 2013b, Wolf 2011, Voros 2007]. Less frequently, the quality of the tracking component is investigated [Sznitman 2013, Reiter 2010].

From times to times, two other performance assessment aspects are considered: verification consisting in assessing that a method is built correctly and evaluation consisting in assessing that a method is valuable [Jannin 2006]. Verification assessment has been used to guarantee a proper behavior of the method [Allan 2013], or to get some insights about method strengths and weaknesses [Sznitman 2013]. Evaluation assessment has been performed for practical value demonstration in an eye surgery proximity detection task context [Richa 2011].

The vast majority of assessment have been carried out in the 2-dimensional space, and only a few in the 3-dimensional one [Wolf 2011, Haase 2013, Burschka 2005].

Validation type

Every study can be commonly assessed under two forms, described by the following qualifiers: qualitative and quantitative. The former returns insights after visual observation of a phenomena. The latter corresponds to a systematic empirical investigation of observable phenomena through the computation of statistical or numerical values.

The vast majority of studies report detector performances in a quantitative way, explained in details in the following section. Regarding qualitative assessment, it can be expressed in numerous ways such as images with overlaid detection results [Speidel 2006] or plots showing the evolution of one parameter within the image referential [Richa 2011].

Model validation strategy

The model validation strategy is crucial for assessing the external validity of the model: the extent to which the results of a study can be generalized to other surgical contexts or tools. In a prediction problem, the model training is performed over a data-set of known data, and the model is tested against a data-set of unknown data.

Standard data-set splitting has been used, where the first half of every sequence is collected into the train split and the other halves represent the test split [Sznitman 2012, Sznitman 2013, Sznitman 2014]. More robust validations, using a cross-validation strategy, have been employed; in a leave-one-out manner [Sznitman 2012], or in a 10-fold way [Kumar 2013b].

Sometimes, the data-set separation into train/test sets is unclear and the same images may appear in both sets [Speidel 2006, Speidel 2008]. Or even no train/test sets are required for online learning algorithms [Reiter 2010].

Finally, tool detectors not relying on a model learning strategy do not require separate train and test splits [Voros 2007].

2.3.4.2 Computation phase

The computation phase of the validation methodology expresses how the estimation of a validation criterion is being performed. Three elements describe the quantification of a validation criterion: a comparison method (i.e. metric), information on which the computation is performed (i.e. reference), and a figure of merit (i.e. quality index).

Criterion

A validation criterion aims at characterizing different properties of a method such as its accuracy, precision, robustness, or reliability. This information is not reported in the table as every study has been exclusively focusing on both accuracy and precision. Some attempts have been made to retrospectively study the robustness, but it was not the intended objective for the study and as such can not be considered as the validation criterion. Using in-vivo data, with full clinical realism but no control, it is difficult to target the validation of either robustness or reliability.

Reference

The first element necessary for the validation computation is the type of information on which the measure is performed. In general, the computation is not directly performed between detection results and corresponding references. Many different information can be contained in the reference, for example the tool location, its orientation or its tip position, but not all the information can be compared simultaneously. As such, a normalization step is performed to transform reference and results information into a meaningful and equivalent representation for processing.
The favored reference, used in every study, is a landmark on the tool: either the tip [Sznitman 2012, Sznitman 2013, Speidel 2008, Voros 2007], the center [Sznitman 2014, Reiter 2012a], or the end [Reiter 2010]. While sometimes, the overall tool pose not limited to a specific landmark is used [Pezzementi 2009, Speidel 2006].
The second most common reference is the orientation of the tool shaft [Wolf 2011, McKenna 2005]. While few works exploited tool bounding boxes, either as direct reference in Kumar et al. [Kumar 2013b], or by deriving pixel-wise tool label maps [Speidel 2014, Pezzementi 2009].

Validation metrics

The metric is a comparison function measuring a distance between the normalized results of the method and the corresponding normalized reference. Previously used metrics can be regrouped in four categories: simple distance, IOU, NCF and visual criterion.
The distance, often computed as the Euclidean distance, is favored when the

reference is a single value (e.g. tool orientation) or a point (e.g. tip position). The metric, usually used for simple computation, can also be used in a thresholding fashion to separate true positive from false positive detections. For example, a detection is considered accurate for a distance error under 10 pixels [Sznitman 2013], or recall values are reported following an evolving distance threshold [Sznitman 2012, Sznitman 2014].
The Intersection Over Union (IOU) criterion metric has been used with the standard 50% overlap threshold between bounding boxes in [Kumar 2013b]. A variant has also been proposed, where the criterion is not employed with bounding boxes but rather over the full image in a pixel-wise fashion [Pezzementi 2009, Speidel 2014].
Both aforementioned metrics operate towards spatial detection performance. A metric dedicated to the tracking aspect has been proposed by [Sznitman 2012] and [Reiter 2010]. The Number of Consecutive Frames (NCF) until the tracker loses the tracked detection is considered.
Finally, the visual metric does not rely on a specific use of reference, the evaluator being the only judge with his own subjective opinion. For example, where the tool center-line must be within the tool shaft [Reiter 2012a], or where the tool-tip location must be accurate with according joint configurations [Reiter 2012b].

Figures of merit
The figure of merit, or quality index, is used to obtain a statistical measure of the distribution of local discrepancies computed using the validation metric. Three figure of merit types have been identified: standard statistics, standard performance measures [Makhoul 1999], and duration.
Standard statistics relate to error computation, most of the time of pixel values. Examples are mean (M) error [Wolf 2011, McKenna 2005], standard deviation (S) of the error [Sznitman 2014, Speidela 2013, Kumar 2013b], or order statistics (O) of the error [Haase 2013].
Standard performance measures, also expressed as information retrieval metrics, cover the calculation of true positive, true negative, false positive, false negative, and all entailing measurements such as recall (R) [Sznitman 2012, Sznitman 2014], precision (P) [Allan 2013], accuracy (A) [Kumar 2013b] and probability of error (PE) [Pezzementi 2009].
The duration has only been used once to report in seconds an elapsed time [Reiter 2010].

2.4 Discussion

2.4.1 Data-sets

While many pedestrian data-sets have been collected over the years, only a handful were commonly used and nowadays the Caltech-USA data-set is the predominant benchmark because of its large and challenging data. Conversely, almost each surgical tool detection study has been relying on its own data-set and as of today, no

surgical tool benchmark has been chosen by the community.
However, the value of benchmarks is undeniable according to Benenson et al. [Benenson 2014] because individual papers simply show a narrow view over the state of the art on a data-set. Having an official benchmark greatly eases the author's effort to put their results into context, thus also providing reviewers easy access to state of the art results.
The lack of surgical tool data-sets online availability has also been a major hindrance, compared to the well-referenced accessibility to pedestrian ones. Recently, more and more authors have been making their data-sets freely available (e.g. [Kumar 2013b, Sznitman 2014]), which represents a step in the good direction.

2.4.1.1 Data acquisition

Generally, acquiring videos in order to put together a pedestrian data-set can be considered as a quite easy task to achieve because necessitating daily life street recordings only. For Caltech-USA, a vehicle was equipped with a camera and the driver was asked to drive normally through regular traffic in different neighborhoods of a big city. On the other hand, having access to in-vivo surgical recordings is quite harder, as it requires to enter the OR which is a much more restricted and regulated environment than urban streets. As a result, every surgical tool data-set has been covering either one of those three surgical fields: MIS, ophthalmology and neuro-surgery, where a device capable of performing video recording is used as part of the clinical routine (e.g. endoscope, surgical microscope).
Adding extra-sensors into the operating theater has been considered (e.g. consumer cameras [Sznitman 2014], ToF camera [Haase 2013]), yet remaining an unpopular solution due to many regulations. Being able to leverage data from already existing sensors used in multiple hospitals seems more critical than the creation of a better setup with ideal recording conditions and specific to one hospital. The choice of adequate additional sensors is also not clear, especially to record the surgical field-of-view where surgical tools are in action. For example, ToF camera devices exhibit a low signal-to-noise ratio due to multiple error sources, such as temporal noise or systematic offsets [Haase 2013]. Already existing surgical endoscopes and microscopes are seemingly the fittest devices for the task, and the recording quality can be improved with new hardware generations (i.e. from SD to HD cameras).

Regarding data quantity, for pedestrian there is no limit to the amount of data that can be collected by continuous acquisition from a mobile recording setup, hence the large and diverse Caltech-USA data-set. Unfortunately, depending on the surgical field chosen for a data-set to represent, or the surgical procedure itself (e.g. cholecystecomy), data may be scarce. In the partnered hospital, only one intervention could be performed per week for this specific type of surgery. As such, if not enough videos are collected, the data-set will not be representative enough of the surgery and will not cover a wide range of background and tool variations. On the opposite, collecting enough surgical recordings can be a very tedious

process over many years. For surgical tool data-sets, a trade-off has to be found between data quantity and diversity since a lack in either category is deterrent for data-driven generic approaches. As of now, the richest data-set proposed contains around three thousand images taken across sixteen different videos [Kumar 2013b]. When gathering data, automatic or random selection processes should be favored to manual ones in order to reduce to a minimum any selection bias. As such, video segment selection for subsequent split into images is preferable to stand-alone images selection [Dollár 2011]. Aside from minimizing selection bias, selecting video sequences also provides an additional pool of information to process. Temporal correspondences between tool instances within a sequence can be used to analyze trajectories, and even temporal features can be leveraged in the model learning process (e.g. optical flow). Not to mention the inability for tracking systems to exploit a data-set made of stand-alone images only. Similarly, depending on camera hardware used for the recording, either monocular or stereoscopic images are obtained. While the former category is represented in the vast majority of pedestrian and surgical data-sets, only the latter provides an access to additional features such as depth maps. As a result, depending on the type of data constituting the data-set, some features may not be accessible for computation.

For pedestrian, data taken from photographs or surveillance videos rarely serve as a basis for data-sets due to selection bias and restricted backgrounds, mobile recording device usually being favored [Dollár 2011]. Surgical data acquired in simulated or phantom environments where the control is higher than the clinical realism have proven useful in order to verify that a method is built correctly or in order to precisely understand a method behavior in a challenging environment where for example tools are crossing each other under varying illumination. On the other hand, in-vivo videos should be considered for reference data-set creation as they provide full clinical realism with moderately diverse backgrounds depending on surgical fields and tool appearance variations (e.g. occlusions, motion blur, smoke).
From the amount of visual variations (i.e. challenging conditions) observed, different levels of difficulty can be associated to data-sets. In the medical field, data-sets can be either too easy if only one tool is fully visible or highly challenging when multiple tools are intersecting each other throughout important illumination variations. The data collection strategy should not focus on collecting or avoiding specific challenging conditions such as occlusions, illumination variations, or motion blur. With enough data, all the "challenging conditions" will be present in the data-set in realistic proportions. In addition, strategies exist to access sub-sets covering specific ranges of challenging conditions by relying on the information gathered through the annotation process. For surgical tool data-sets, it is hard to judge about the level of difficulty as the precise number of occurrence for each type of challenging condition is not referenced anywhere.

2.4.1.2 Annotations

Every pedestrian data-set is provided with its corresponding set of annotations, which are bounding boxes around single pedestrian or crowd regions of interest. Such annotations present a two-fold usefulness: they are necessary to extract object windows for training and they serve as reference for the validation process. Suffice to say that a data-set published without annotations has close to no utility, should it be for pedestrian or surgical tool detection. Having erroneous annotations can also be a major handicap as many learning approaches will fail to generalize when similar samples are placed in both positive and negative pools. Annotating the data, also abusively called ground truthing, has to be manually performed because of the impossibility to access the 'real' ground truth. Not only would it require every pedestrian to be GPS-tracked, but also cameras with their pose parameters in order to retrieve pedestrian locations in 2D images. For surgical tools, optical trackers can be used to retrieve the 3D pose and then 2D tool locations can be computed assuming a good camera calibration. Rather tedious and time-consuming, depending on the quantity of images to annotate and the amount of labels to place, the data annotation process must be done carefully to optimally benefit from the data-set.

Bounding boxes have been the favored choice of annotations for pedestrian, fitting well enough with their appearances. In order to be uttermost accurate, visible and occluded parts of pedestrian are separately annotated. Some labels are added to better describe the content of the bounding box and specify for example if a pedestrian is isolated or amongst a crowd. All those annotations put together enable the computation of a large set of pedestrian statistics for a data-set. In addition, they also allow for reference filtering in order to exclude portions of a data-set during the validation, such as removing pedestrian belonging to crowd regions much more difficult to detect. On the opposite, annotations performed on surgical tool data-sets are much more lackluster, where most of the time only tool-tip positions exist. Bounding boxes have been used without the distinction between visible and occluded tool parts. However, surgical tools undergo tremendous amounts of in-plane rotations. As such, bounding boxes can contain a lot of surgical background due to said rotations and a tighter geometry seems necessary to delineate surgical instrument contours (e.g. polygons). Annotations in a pixel-wise fashion, along the line of what is presented in [Speidel 2008] seems more fitting. On the down-side, it is not possible to exclude difficult portions of a data-set as attribute labels are not provided to describe tool appearances (e.g. blurry, occluded).

2.4.2 Detection methods

First of all, we evoked that surgical tool detection can be achieved by modifying the tool physical appearance using external markers or by adding hardware sensors, encoders or external optical systems. While the former mainly presents issues with

manufacturing, bio-compatibility and interference with other medical devices; the latter requires extensive hardware integration and still has limitations in accuracy and integration into the operating theater. Further insights about this category of methods for surgical tool detection are presented and discussed in chapter 8.
Most pedestrian and surgical tool detectors are relying on image-based analysis, thus recovering object position and orientation directly in the viewing reference. Albeit pedestrian detectors focus on individual images, many tool detectors try to make use of previous frames through tracking. According to Benenson et al. [Benenson 2014], when starting from a strong spatial detector, using extra information such as tracking or temporal features improves performance. Nevertheless, it is not clear how much tracking can improve per-frame detection itself. Additionally, tracking is reliant on a spatial detector for initialization and occasionally re-initialization procedures. All of it suggests the necessity to possess an efficient and reliable spatial detector.

2.4.2.1 Feature representation

Feature representation is a highly versatile component as variants or combinations of more than eight different feature families can be extracted from images. While spatial features (e.g. color, HOG) have been present since the beginning of detectors, stereo and temporal cues (e.g. depth map, optical flow) remain yet to be fully exploited.
For pedestrian detectors, every feature representation is based around HOG, usually combined with color to form a HOG+LUV representation. Top performance are reached with the use of ten feature channels, despite some approaches having considered up to an order of magnitude more channels. Conversely, it is worth noticing that HOG features are barely used for surgical tool detection, where color is heavily favored through multiple spaces. Choosing the right color space can be driven by the object to detect, for example HSV is stated to be better suited than RGB for surgical tool detection [Speidel 2006]. HSV color space offers a separation between the chromaticity and the luminance component, therefore more robust to illumination changes.
Previously mentioned, features available for extraction are bound to the data acquisition device and the detection strategy. Indeed, depth features cannot be computed if only monocular images are retrieved, and motion features will not be extracted in case of purely spatial detectors.
As stated in [Benenson 2014], the most popular approach for quality improvement is to increase and diversify the features computed over the input image. Having richer and higher dimensional representations tends to ease the classification task, enabling improved results. However, developing a more profound understanding of what makes good features good and how to design them is still needed. Up to now, improvements were made through extensive trial and error.
As a side note, many pedestrian detectors have been favoring integral feature channels, which is also starting to be the case for surgical tool detectors [Reiter 2012a].

It seems to be the way to go as it offers easier and faster access to feature values over different configurations of rectangles, compared to standard feature channels.

2.4.2.2 Detection strategy

For pedestrian detectors, three main families of learning techniques have been exploited (i.e. Decision Forests, Deformable Part Models, Deep Networks), each one providing extremely close results to the others [Benenson 2014], indicating the choice of the learning technique is not a dominant one. Interestingly enough, they have only been scarcely used for tool detectors where many threshold-based approaches have been favored, even though inducing too much bias and not enough reproducibility in the way thresholds are defined. According to Sznitman et al. [Sznitman 2014], building classifiers to evaluate the presence of surgical instruments appear to be the most promising solution for both in-vivo detection and tracking, and as such data-driven learning strategies should always be favored.
A major objective for every learning approach is to generalize from train to test set, usually represented by the model accuracy versus generalization trade-off. Learning with too much accuracy is called "over-fitting", where even noise or insignificant details are learned (i.e. similar to learning something by heart). Conversely, not enough accuracy in the learning leads to over-generalization and the model is triggered too much (i.e. many false alarms). Models that can generalize well are compulsory for detectors to be able to identify surgical tools throughout various surgical procedures coming with slight to moderate background and condition variations.
An alternative for surgical tool model creation is to use a robot renderer with a CAD model to construct tool templates according to specific kinematic joint configurations [Reiter 2012b]. This is stated to be desirable because collecting training data becomes easier than if it had to come from videos, thus enabling larger collection with less effort. Advantages of this type of data generation have been shown successfully in [Shotton 2013].
Choosing appropriate object parts to model can also prove to be a tricky matter, especially because of object occlusions. Pedestrian are ordinarily fully modelled, then sub-models can be derived to better handle occlusion cases [Mathias 2013]. For surgical tools, the most important part to model is also the most characteristic landmark for tool differentiation, namely the tip region. Unfortunately, this is also the part most likely subject to appearance modifications. Additionally, tool tips can be cumbersome to model when made of many parts, which is the case for articulated surgical instruments. Nevertheless, modeling the tip remains the most viable tactic because undoubtedly always visible in the field-of-view and very specific for each instrument, relatively to the tool end or tool body.
A parallel can be made between pedestrian detectors exploiting context and two-stage tool detectors performing a pixel-wise classification as first stage. Both detectors rely on integrating *semantic labelling* results in the pose estimation step. While many classes can be modelled towards pedestrian detection such as road, building, or sky, their number is highly reduced for surgical tool detection. Indeed, given

the nature of surgical instruments being gray-ish metallic objects, usually only two classes are necessary: one to model tool pixels and one to model background pixels. Evidently, the more classes the less accurate semantic labels maps, as differences between classes will be more and more subtle. Leveraging contextual information seems to be a promising concept and has been largely employed for surgical tool detectors.

2.4.2.3 Prior knowledge

Pedestrian detection methods are fully data-driven, the model learning being performed from a set of training samples. Thus, methods' design is unaffected in any ways by outdoor backgrounds or pedestrian appearance conditions, everything resting on the shoulders of the training strategy.
Contrariwise, many surgical tool detectors try to reduce the complexity by adding some sort of prior knowledge. Purely looking at both computer vision and machine learning aspects, using such assumptions or external assistance can be seen as a weakness. Surgical tool detection approaches will fail to be generic as shape or location constraints for one surgical instrument do not necessary apply for another surgical instrument. Thus, it will not be possible for one detector to detect other type of tool or even to detect the same tool within another surgical context with a different background.
Nevertheless, when looking at the other end of the scope, namely in-vivo surgical applications, those same assumptions can be seen as a strength. For routine surgical applications heavily relying on surgical tool detection, the higher the performance, the safer for the patient. In that regard, adding as much prior knowledge as possible will increase to a maximum detector performance.

2.4.2.4 Optimization strategies

The end goal for pedestrian or surgical tool detectors is an integration within higher-level systems, be it for robotics, surveillance, care for the elderly or disabled, or context-aware surgical applications. As such, in addition to high performance, a processing speed matching the recording device speed is also required for such real-time applications.
Pedestrian detectors can be straightforwardly ranked and compared based on their speed performance thanks to benchmark data-sets and standardized search space (e.g. scales per octave). Unfortunately, surgical tool detector speed comparison cannot be performed because of too much variations in input image resolution and search space.
The biggest impact on processing speed comes from hardware specifications and code optimization. When dealing with image-based processing, an extensive use of the GPU should be done to benefit from parallel computing. As such, every new hardware generation is accompanied by a huge speed boost. Implementation strategies will also have an impact on the processing speed, but a lesser one. For instance,

cascade-based classifiers such as random forests can be speed-up by either limiting each tree depth, using some sort of soft-cascading [Benenson 2012] or early stopping scheme [Sznitman 2014]. Already mentioned, performing feature computation over integral channels also provides a speed gain. The most popular speed-up strategy is to perform down-sampling over input images or to use larger strides at run-time, for example processing every fourth line and column.
Aside from GPU implementation and code optimization, most of the strategies enable a speed-up gain always at the cost accuracy. As such, it is necessary to find the right balance within the speed versus accuracy trade-off.

2.4.3 Validation methodology

The purpose of the validation methodology is to quantify and rank detectors performance in a realistic, unbiased and informative manner. According to Dollar et al. [Dollár 2011], there is in general no single 'correct' validation methodology, yet proposing a proper one is crucial and surprisingly tricky. To ensure consistent and reproducible comparisons, using the exact same code is mandatory, as opposed to a re-implementation where elements may be missing.
Analogically to data-sets, a validation methodology of reference is used for pedestrian detectors while each surgical tool detector has been validated according to its own methodology. Currently, it is hard to judge or compare tool detectors looking only at their individual results provided out of a reference context.

2.4.3.1 Specification phase

For pedestrian detectors, a clear set of model validation strategies has been proposed by Dollar et al. [Dollár 2011]. Described scenarios, relying on two distinct train and test image sets, can be used either for the validation of existing and pre-trained detectors, or for training and validation of a new detector, depending on train and test set belonging to a same data-set. For surgical tool detectors, only few studies use similar protocols with train/test splits, such as k-fold cross-validations. One limitation arises from a train/test splits strategy employed where the beginning of an image sequence represents the training set and following images of the same sequence represent the testing set. This is a borderline strategy as the learned model can over-fit the data, thus potentially not working in other video sequences of a same surgical type. However, many times there is no clear explanation about image distributions and images from the training set may also be included in the test set. The only exception being for online learning approaches where model validation techniques do not apply.
In addition to quantitative results, some surgical tool detection studies conjointly report qualitative assessment. While it can be interesting to get insights about detection success/failure modes, this is heavily observer-biased and not at all reproducible. Detector comparison and ranking can only be performed through quantitative assessment.

Pedestrian detector validation is mainly being performed in the 2D space since single monocular input images are processed. Some surgical tool detectors have been validated in the 3-dimensional (3D) space, but not only stereoscopic input images are needed to obtain 3D tool detections but a real ground-truth obtained via an automatic system is also necessary to perform the validation. In general, those conditions are difficultly fulfilled, explaining the favored 2D space validation.

2.4.3.2 Computation phase

Unsurprisingly, the choice of the metric, the reference and the figure of merit are all entangled and highly correlated to the specification phase. In case of a 3D validation, the Intersection Over Union criterion metric can not be used, mainly because 3D reference bounding boxes do not exist. Similarly, the number of consecutive frames metric will only be used to assess tracking performance.
Many tool detector validation methodologies have been performed only over one tool landmark (e.g. tool-tip). While a detector could achieve high performance regarding only a specific tool landmark location, the overall tool pose could be inaccurate most of the time. It feels necessary to couple a tool landmark validation with a global pose validation, using for example bounding boxes or any kind of larger geometry. That said, choosing the according tool detector metric set should be driven by the final medical application in which it will be used. In specific cases of tool positioning, only an accurate tool-tip location is mandatory, thus justifying the choice to perform exclusively an assessment of tool-tip location performance.
Every pedestrian detector has been validated using the same set of parameters: the Intersection Over Union criterion metric, bounding boxes as reference, and recall, precision, and LAMR as figures of merit. Given the nature of pedestrian to detect, rectangular bounding boxes fit rather well with their geometry, hence reported performance results are highly informative. Using the same strategy for surgical tools, where important in-plane rotations are occurring, will provide far less precise results. As already mentioned, for surgical tools, bounding boxes should be replaced by a geometry tighter to a tool shape (e.g. polygons). Another strategy, for object validated under this metric set, is to modulate the overlap threshold depending on the class of the object. For pedestrians and cyclists, an overlap of 50% is required, while for cars the overlap should be at least of 70% [Geiger 2012]. Depending on the nature of the object to detect, and the relative size of the bounding geometry, it is necessary to trade lightly with the overlap threshold.
Aside from global tool pose validation, it is worth mentioning that intermediate steps performance are also being assessed. For example, the pixel-wise classification (i.e. *semantic labelling*) is studied with per-class and per-pixel accuracy computation. Regarding surgical tool trackers, the validation is only performed through the computation of the number of consecutive frames they are able to track. However, the added-value of the tracker itself is almost never quantified. Comparing detector performance with and without the use of the tracking layer could be of interest.

2.5 Conclusion and problematic of the thesis

In this chapter, we presented a methodological literature review on image-based object detection for two different instances: pedestrian and surgical tool. For structuring the review, existing works have been classified and presented following three major components: validation data-sets, detection methods, and validation methodology. Each component introducing a sub-set of elements to provide a more detailed level of description for each existing work.
When comparing pedestrian detection and surgical tool detection existing works, the most eye-catching conclusion to draw is the lack of standardization for the latter one. For pedestrian, a limited number of well-thought, well-detailed and well-referenced benchmark data-sets is employed. Conjointly with very standardized validation methodologies, performing comparisons and rankings between detectors is fairly easy. For the surgical tool instance, each work has been referencing to a different data-set, and even though validation methodologies are relying on the same concepts (as for pedestrian), a direct performance comparison between detectors is simply impossible. Regarding detection techniques, the pool of machine learning algorithms is rather limited, with a large majority of tool detectors tending to rely on prior knowledge to ease the process, compared to pedestrian ones.
Such a difference between the two object detection instances shows the relative novelty of the domain that has not been as extensively studied as the pedestrian one. As a consequence, many insights can be taken from the pedestrian detection task as to build upon better tool detectors and better comparisons between them.

From this review of the literature, the problematic of the thesis was centered on four main aspects. First, neurosurgical data being barely used as material for tool detection, the necessity arose to create a new data-set, as robust and diverse as possible with the potential to be a benchmark. Secondly, developing new image-based tool detector approaches was of interest, using microscope videos as input and as few assumptions as possible (i.e. limited to no prior knowledge). Thirdly, for use in real medical application, solutions had to be found to obtain real-time tool detection, either through algorithmic concepts or programming strategies. Finally, exhaustive validation methodologies have been investigated for meaningful detector performance comparison and optimal assessment of tool pose estimation quality.

Part II

Automatic detection of surgical tools in real-time

Chapter 3

Data-sets and annotation protocol presentation

Contents

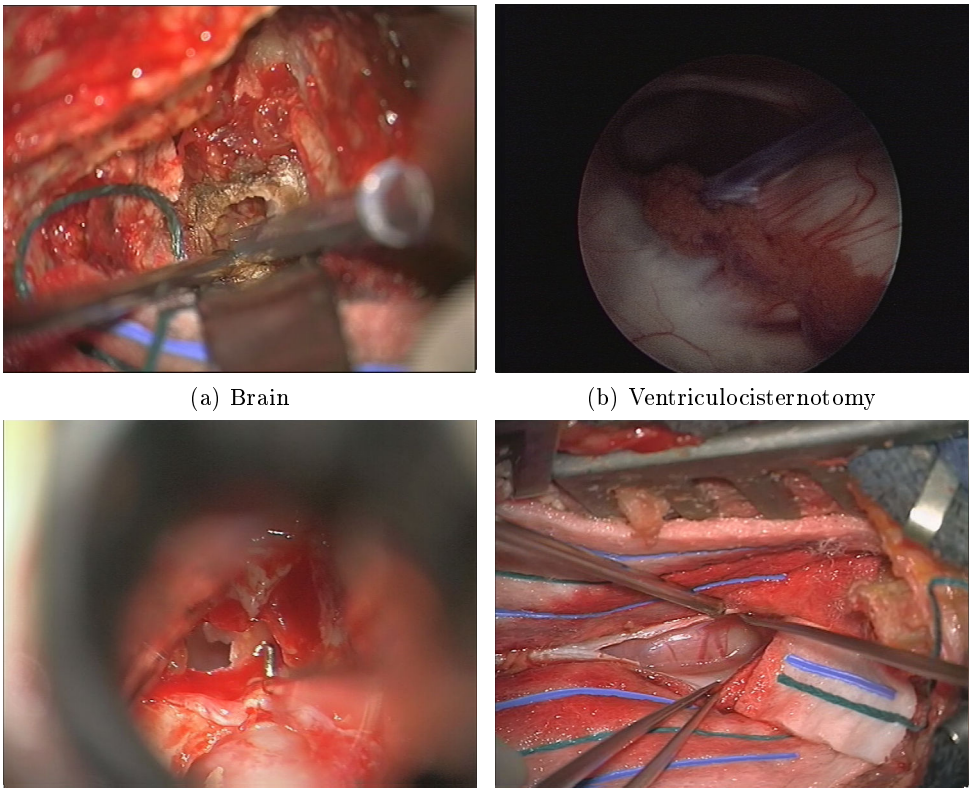
3.1 Introduction

As previously explained, data-driven approaches for object detection require large and diverse data-sets to efficiently learn object models. Surgical tool detection being a fairly new topic of interest, no benchmark data-sets exist yet, each study being evaluated upon its own proposed data-set. In addition, only one previous work has been focusing on the neurosurgical field, with a data-set exhibiting weaknesses such as a rather low diversity, and not precise enough annotations. As such, we created a new in-vivo data-set covering the neurosurgical field, tackling as best as possible issues regarding size, diversity, annotation quality and covered challenging conditions. All studied algorithms presented in the manuscript were validated upon this new data-set.

This chapter starts by presenting in section 3.2 all the in-vivo data gathered from CHU Rennes hospital and available for the data-set creation. In section 3.3, we introduce in details our proposed data-set and its creation strategy. Then, the data

annotation protocol followed to manually create the reference (i.e. ground truth) for the data-set is described in section 3.4. Finally, a discussion regarding both the data-set creation and the data annotation protocol is provided section 3.5.

3.2 Available data



(a) Brain (b) Ventriculocisternotomy

(c) Pituitary (d) Spine

Figure 3.1: Example images taken from collected neurosurgical data.

In-vivo surgical recordings (i.e. videos) were captured at the neurosurgical department of the University of Rennes hospital (CHU Pontchaillou). Recording devices used were OPMI Pentero (Carl Zeiss Meditec AG) microscopes, with an initial image resolution of 720×576 pixels and at 25 frames per second (fps). Four intervention types were covered by the data: pituitary surgeries, brain tumor removal surgeries, spine tumor removal surgeries, and ventriculocisternotomy surgeries.

Pituitary surgeries consist in an incision made in the back wall of the nose to remove tumors within the pituitary gland.
Ventriculocisternotomy surgeries consist in an opening allowing cerebro-spinal fluid to drain through a shunt from the ventricles of the brain into the cisterna magna.
Brain tumor removal surgeries consist in the removal of a tumor either at the

basis of the cranium, or on the upper part of the brain.
Spine tumor removal surgeries, also known as spinal cord surgeries, consist in a dorsal sagittal incision to remove tumors such as meningioma or schwannoma (a.k.a neuroma).

Table 3.1 gives an overview of available data, where multiple video sequences have been acquired for a single intervention (i.e. surgical procedure). Storage size restriction prevents the recording of a procedure in its entirety within a single object and as such multiple video sequences are necessary. In addition, surgeons often choose to record only specific moments of the surgery, thus resulting in multiple video sequences for the same intervention.

Table 3.1: Available in-vivo surgical recordings acquired in the neurosurgical department of the University of Rennes hospital.

Surgery Type	# Interventions	# Video seq.
Brain tumor removal	60	530
Spine tumor removal	25	140
Pituitary	29	117
Ventriculocisternotomy	12	12

3.3 The *NeuroSurgicalTools* Data-set

We named our proposed data-set the *NeuroSurgicalTools* data-set. In section 3.3.1, we start by presenting the data collection strategy followed to assemble the data-set. Then, we provide in-depth tool statistics in section 3.3.2 and highlight some challenging conditions appearing throughout the data-set in section 3.3.3.

3.3.1 Data collection strategy

The flowchart representing the data collection process is provided in figure 3.2, where I represents a number of interventions, S a number of video sequences and N a number of images.

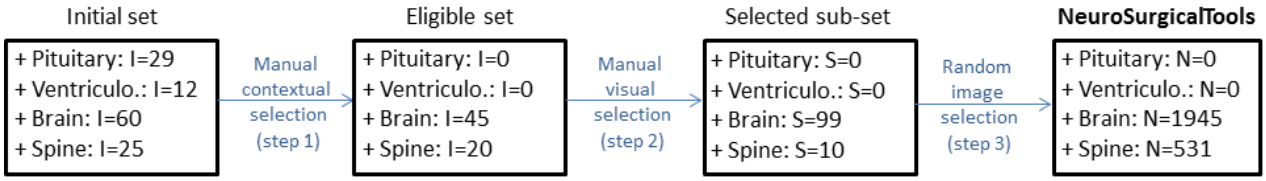


Figure 3.2: Flowchart describing the data collection strategy to create the *NeuroSurgicalTools* data-set.

From the initial pool of surgical data, we excluded videos not exploitable for

image-based surgical tool detection, step 1 of the flowchart. Pituitary recordings were put aside because of inherent surgical conditions where the in-focus field-of-view is extremely narrow, thus making the most part of surgical instruments extremely blurry. Ventriculocisternotomy recordings were removed due to a lack of tool diversity as only one is appearing throughout the twelve videos. Finally, as illustrated in figure 3.3, other videos were rejected because of bad recording conditions. Visibly darker videos exhibiting a lot of noise indicate a recording problem from the microscope, and as such are not representative for the task. Similarly, videos showcasing surgeons' fingers occluding more than 70% of the FoV are of no interest since surgical instruments are not visible.

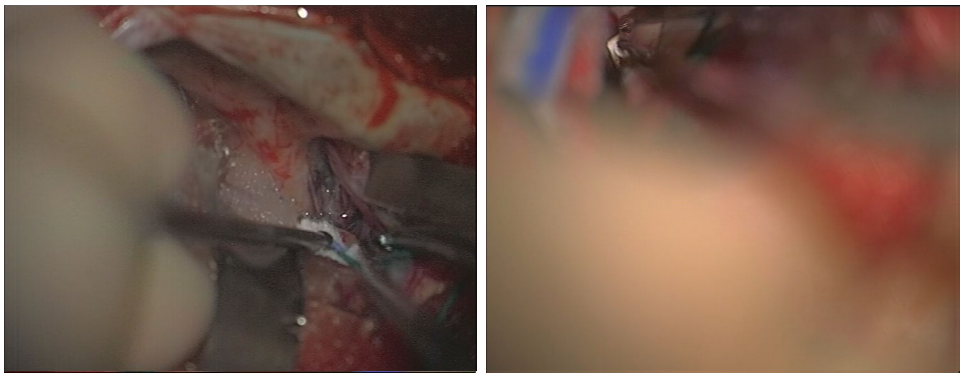


Figure 3.3: Example excluded data because of bad video quality (left) or large occlusion (right).

Within the remaining set of brain and spine tumor removal procedures, we manually selected 14 different videos with the targeted objective to obtain enough diversity in terms of background and instrument representation (step 2 of the flowchart). From each one, we cropped a video sub-sequence of varying length, at least showing in action one of the two most common surgical instruments used in neurosurgery (i.e. suction tube and bipolar forceps). To further limit the manual selection bias, we did not try to avoid challenging conditions for image-based detection, such as cases of tool occlusion, tool overlap, or coming from lightning.
In order to remove side-effects from interlaced videos that are appearing on still images (see figure 3.4), each video segment has been re-encoded for a final video resolution of 612×460 pixels. This stabilization simplifies the annotation process while also providing 'cleaner' images for tool model creation.

After video de-interlacing, each sequence has been sampled at 1Hz, and 2476 images were randomly selected to form the final data-set (step 3 of the flowchart). The *NeuroSurgicalTools* data-set has been further split into training and testing sets to fit with validation methodologies. Figure 3.5 shows representative images of the data-set. The data-set is freely available online [1].

[1] https://medicis.univ-rennes1.fr/software

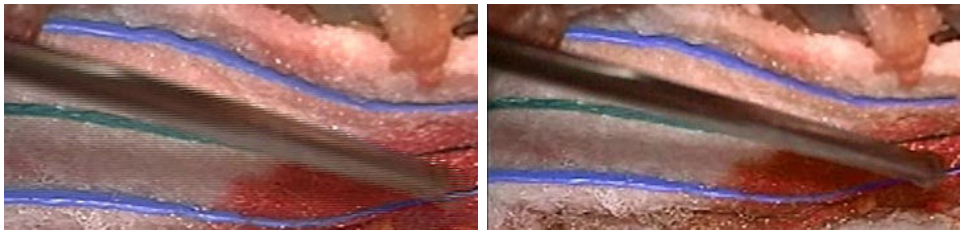


Figure 3.4: Interlacing effect on still images. Original suction tube (left) and de-interlaced suction tube (right).

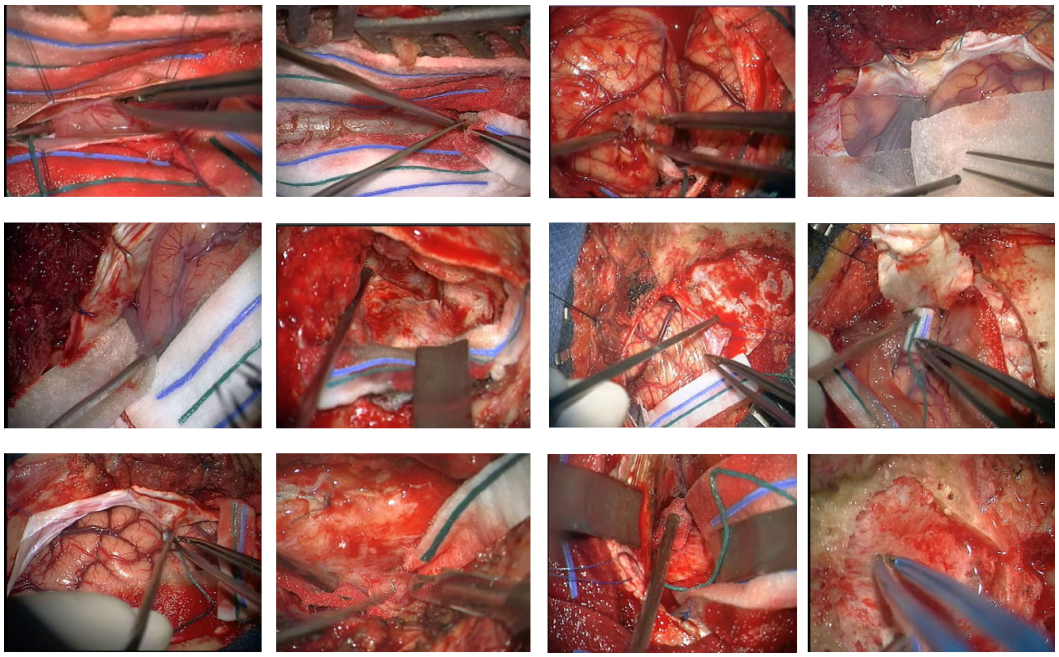


Figure 3.5: Example images from our *NeuroSurgicalTools* data-set.

3.3.2 Dataset statistics

A summary of the data-set is provided from table 3.2 to table 3.5. The distribution of images from each surgical sequence is provided in table 3.2. In those images, seven different surgical tools are appearing for a total of 3819 occurrences (see table 3.3). At most one instance of each instrument category is visible in an image, at the exception to retractors which can be up to three simultaneously. Two surgical instruments are heavily featured in the data-set: the suction tube and the bipolar forceps.
Proportions of the number of tools per image are reported in table 3.4 and 3.5. Not considering the retractors, almost 50% of images are displaying two surgical instruments simultaneously, only 10% exhibit three surgical tools at the same time, and about 27% of the frames have no instruments at all.

Table 3.2: Number of images per surgical sequence.

Train set		Test set	
Background	221	Background	256
Seq 1	225	Seq 7	180
Seq 2	300	Seq 8	140
Seq 3	80	Seq 9	224
Seq 4	32	Seq 10	80
Seq 5	180	Seq 11	139
Seq 6	183	Seq 12	40
-	-	Seq 13	70
-	-	Seq 14	126
Total	1221	Total	1255

Table 3.3: Tool occurrence per image split.

Surgical tool	Train&Test splits	
Suction Tube	900	844
Bipolar Forceps	538	460
Retractors	157	140
Hook	163	88
Scalpel	55	130
Pliers	81	79
Scissors	33	30
Others	0	121
Total = 3819	1927	1892

Table 3.4: Distribution of tools per image (with retractors).

Tools/image	0	1	2	3	4
Number	647	293	1041	356	139

Table 3.5: Distribution of tools per image (without retractors).

Tools/image	0	1	2	3	4
Number	667	341	1173	295	0

Below, we analyze distributions of surgical instrument scale, in-plane orientation and location over the whole *NeuroSurgicalTools* data-set (i.e. both train and test splits). We report results for the two most represented instruments: the suction tube and the bipolar forceps. In addition, only those instruments have been annotated with an isosceles triangle (see section 3.4) from which statistics are computed.

3.3.2.1 Scale statistics

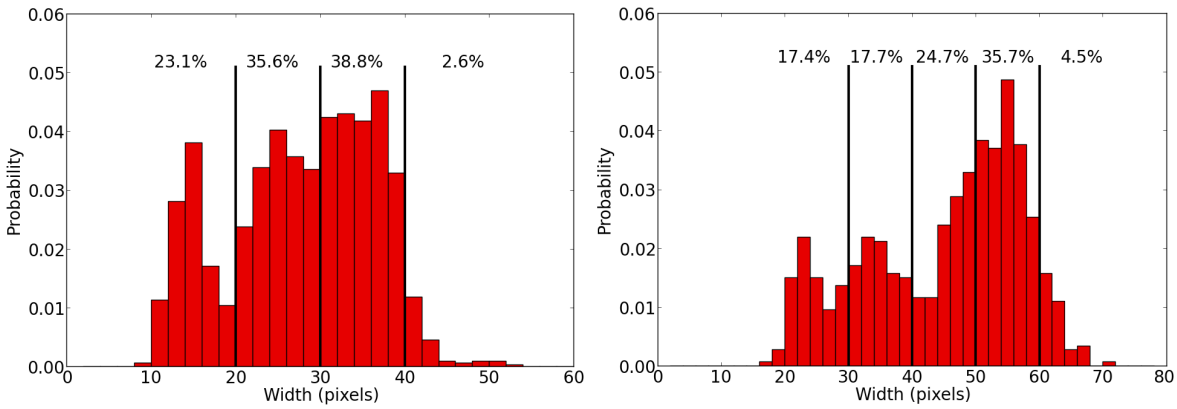


Figure 3.6: Tool shaft width distributions computer over the *NeuroSurgicalTools* dataset. Suction tube (left) and bipolar forceps (right).

Surgical videos being recorded with different microscope parameters, especially the zoom value, surgical tools appear at different scales. In figure 3.6, we report histograms of tool shaft widths for the suction tube and the bipolar forceps. The vast majority of suction tubes (i.e. around 75%) appear with a shaft width between 20 and 40 pixels. The bipolar forceps is a larger tool, mostly with a shaft width between 40 and 60 pixels (around 60%).

3.3.2.2 In-plane orientation statistics

During surgeries, surgical tools undergo in-plane rotations in a range mainly constrained by the surgeon's dexterity. In figure 3.7, we report for each of the two aforementioned tool categories their orientation distributions. For reference, we consider orientation 0° to represent a surgical tool horizontally aligned with its tip facing the left border. Orientation ranges do no overlap between the two tools, implying all the surgeons from the dataset have the same hand dexterity as suction tube and bipolar forceps are often used concurrently. Given orientation ranges for the bipolar forceps between [0°, 30°] and [320°, 360°[, we can assume all surgeons to be right-handed as this tool is consistently used by the dominant hand. Similarly, suction tubes in the range [150°, 270°] are indicating left hand manipulation. Relatively to a vertical image-centred axis, a symmetry can be noticed between instruments use, suggesting an optimal placement of surgeons' hands.

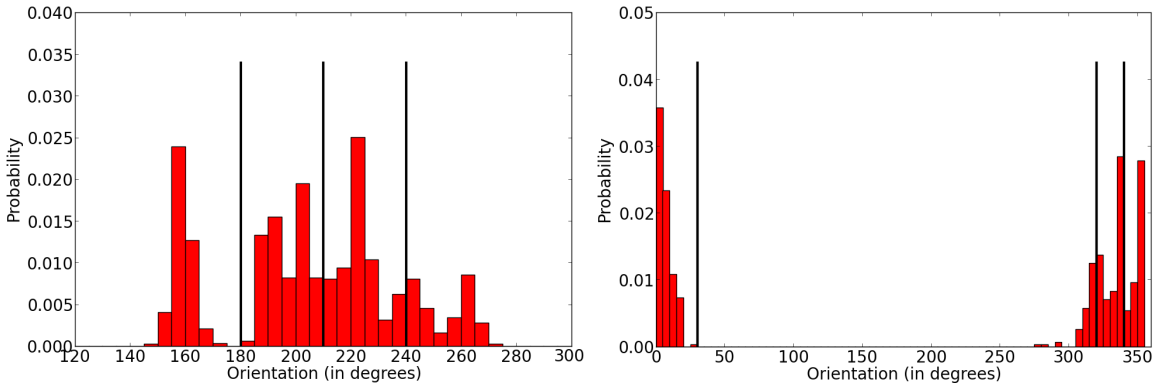


Figure 3.7: In-plane orientation distributions computer over the *NeuroSurgicalTools* data-set. Suction tube (left) and bipolar forceps (right).

Table 3.6: Proportions of suction tube appearance per orientation range.

Orientation range	[140°, 180°]	]180°, 210°]	]210°, 240°]	]240°, 280°]
Proportion	21.7%	32.6%	31.1%	14.6%

Table 3.7: Proportions of bipolar forceps appearance per orientation range.

Orientation range	[0°, 30°]	[260°, 320°]	]320°, 340°]	]340°, 360°[
Proportion	36.8%	11.1%	23.9%	28.2%

3.3.2.3 Position statistics

Viewpoints as long as operating conditions constrain surgical tools to appear only in certain regions of the image. Tool-tip locations over the data-set are computed using annotated isosceles triangles, and resulting heat maps are plotted as shown in figure 3.8. As can be seen, tool-tips are mainly located in the center of the image, although being widely spread. This was to be expected as surgical microscopes are centered over anatomical structures of interest (i.e. on which the surgeon is working).

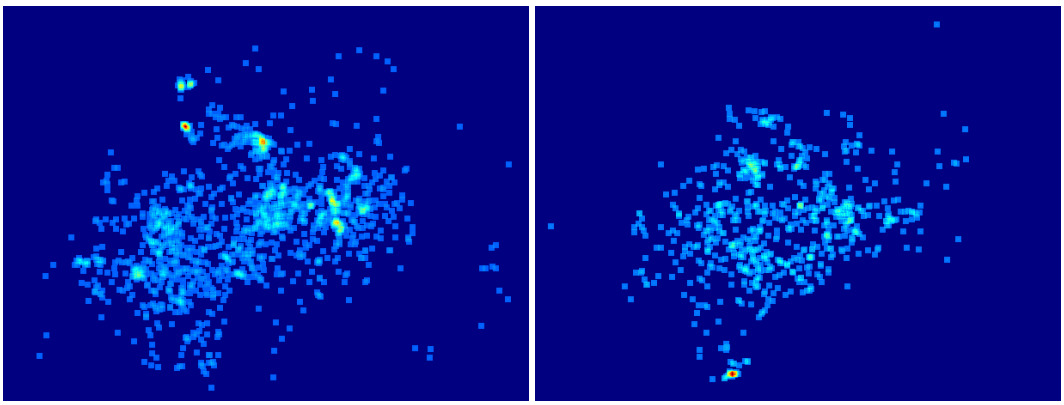
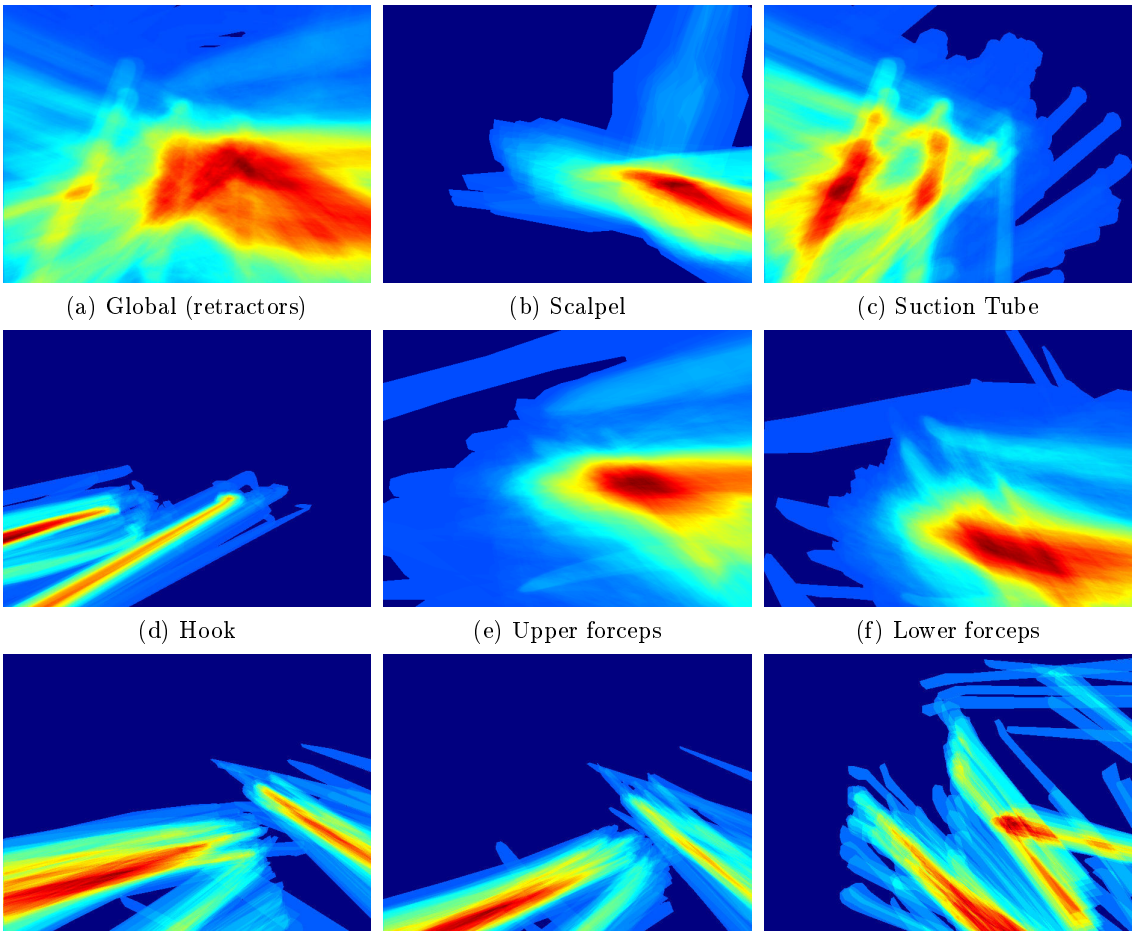


Figure 3.8: Tool-tip locations over the whole *NeuroSurgicalTools* data-set, for the suction tube (left) and the upper part of the bipolar forceps (right).

In addition, global locations over the data-set are also studied and resulting heat maps are shown in figure 3.9. We report in figure 3.9a a mixed heat map accumulating every surgical tool location at the exception of the retractors. In case of multiple-parts instruments, we report one heat map per instrument part (e.g. upper and lower part of a bipolar forceps).
Commonly, surgical instruments enter the field of view from the bottom part of the image, in an upwards direction. This is highly correlated to operating conditions, as the surgeon is usually standing in front of the surgical field. We can also notice an impact of surgeon's handedness on surgical instruments location. For example, the suction tube, hook, and pliers are used by the surgeon's left hand. Conversely, the bipolar forceps, scalpel, and curette are being used by the surgeon's right hand.



(a) Global (retractors) (b) Scalpel (c) Suction Tube

(d) Hook (e) Upper forceps (f) Lower forceps

(g) Upper pliers (h) Lower pliers (i) Curette

Figure 3.9: Cumulative location per tool class over the whole *NeuroSurgicalTools* data-set, represented as heat-maps.

3.3.3 Challenging conditions

Throughout the data-set, surgical instruments appear under a wide range of adverse (or challenging) conditions. Those conditions have been neither annotated nor labelled, as such we propose to highlight each one with a couple images in figure 3.10. The challenging conditions accounted for are the following: tool-tips hidden under anatomical structures, motion blur, occlusions, tools overlapping each other, presence of blood partially covering a tool, and the reflection of one tool into another one.

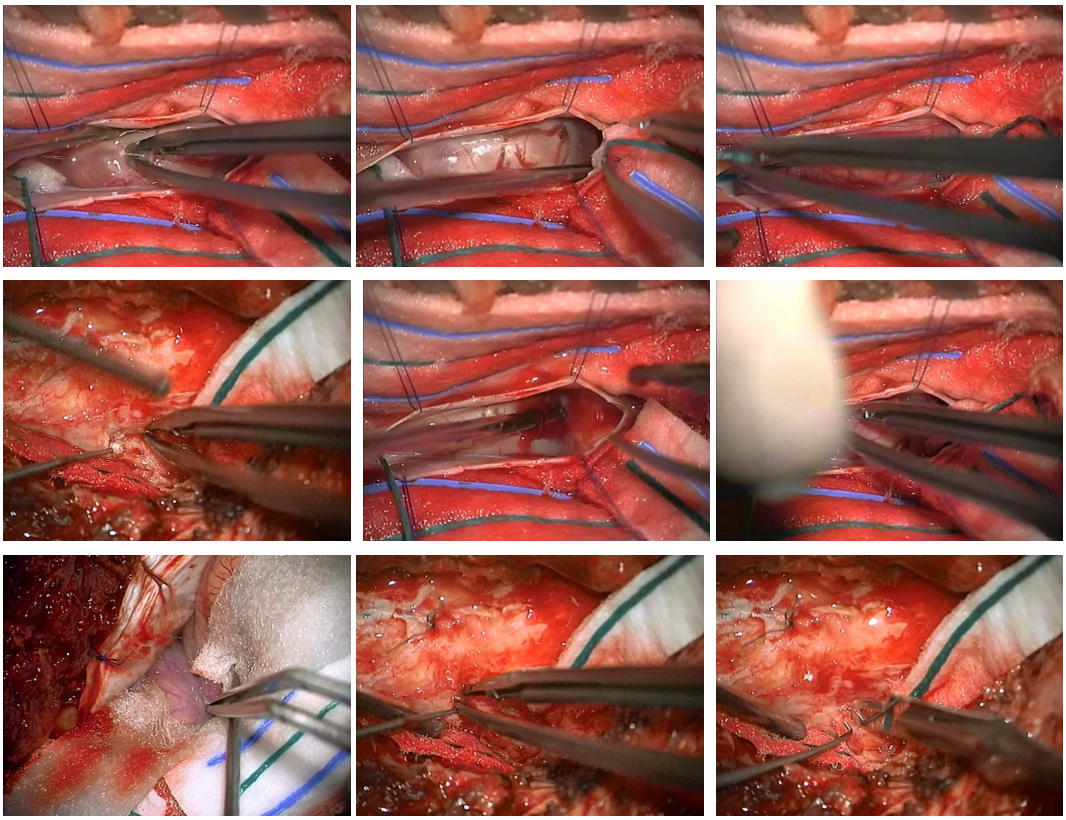


Figure 3.10: Challenging conditions identified throughout the *NeuroSurgicalTools* data-set.

3.4 Data annotation protocol

For the annotation process, we decided to use a web-browser open-source software named LabelMe [Russell 2007]. For more information, please refer to the annex B. Annotations were done manually in each image of the aforementioned data-set by a domain expert. Three types of annotations, illustrated in figure 3.11, have been performed:

- A bounding polygon and a class label.
- An isosceles triangle which first point is located on the tool-tip and the two other points are on each side of the tool body.
- Attribute labels (e.g. blur).

The first kind of annotations has been done on each and every tool appearing in the data-set. Whereas the second one has been done on the suction tube and the bipolar forceps only. This triangle is encoding the tool orientation, its width and its tip position, which are necessary information for automatic data processing.
The attribute labelling is necessary for sub-set selection, especially to remove blurry

images from training samples. However, in case of partial occlusion only one bounding polygon has been placed over the extended tool position. We did not added one annotation on the visible tool part only.

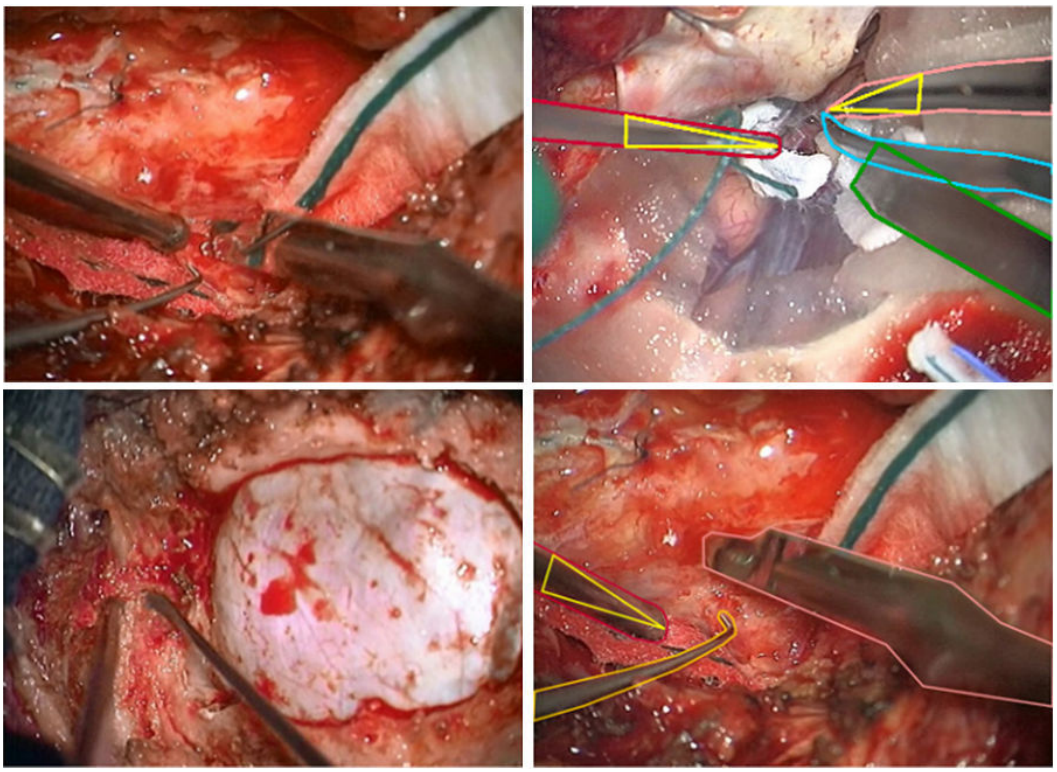


Figure 3.11: Example data-set frames (left column) and annotations (right column). Isosceles triangles are represented in yellow, other colors represent bounding polygons.

3.5 Discussion

3.5.1 Data collection

With our data collection strategy, we aimed to create a data-set diverse-enough in terms of surgical background and instrument representation, as representative as possible to the surgical reality, and with as little selective bias as possible. As illustrated by the numerous challenging conditions, the data-set presents a lot of diversity regarding instrument appearance, especially for the suction tube and the bipolar forceps. However, as highlighted by the statistics, the data-set is fairly unbalanced as other tools are less represented. While representative of the surgical reality where both the aforementioned instruments are mainly used, increasing the data-set in size to obtain enough samples for the different categories of instrument is desirable.

Also pointed out by the statistics, all of the fourteen selected surgical interventions used to create our data-set have been performed by right-handed surgeons. Consequently, surgical instruments are constrained to appear in restricted areas of the image, thus impacting the diversity. In order to fully cover the range of tool appearance possibilities while respecting the surgical reality, video recordings of surgical interventions performed by left-handed surgeons should be added to the data-set.
Lastly, our proposed data-set is only representative of surgical interventions performed at the neurosurgical department from the Rennes University hospital. While many surgical instruments are standard, surgical practice may differ from one hospital to another or from one country to another. For completeness, adding data acquired in other hospitals throughout the world would be of interest to further increase the diversity in tool and background appearance. Be it as it may, the proposed *NeuroSurgicalTools* data-set is diversified enough to train and validate tool detection approaches.

3.5.2 Data annotation

The data annotation quality is preponderant for every ensuing usage of the data-set. Image regions annotated as instrument with a bounding polygon and an isosceles triangle serve as input to the tool model creation process. Similarly, validation methodologies are exclusively focusing on the overlap between obtained candidate detections and the annotated references (i.e. bounding polygons). Consequently, annotations must be done with extreme caution as to obtain high quality tool models and in the end the most meaningful validation results possible. Depending on the data-set size, the annotation process can be cumbersome and time-consuming. In order to obtain high quality annotations and prevent one single annotator to botch up the process, crowdsourcing solutions have been proposed [Maier-Hein 2014]. Data are made available online and many different persons can participate in the annotation effort. In addition to alleviating the time spent annotating for each person, in also tends to prove a limited inter-/intra-annotator variability.
Aside from annotation quality based on user performance, in other terms how close polygons are placed around surgical instruments, the level of details can also be used as a good assessment. We identify five different levels of annotation details as described in table 3.8, each level improving in precision over the previous one. With the first level, each instrument in the image is solely annotated with a bounding polygon, enough to perform pixel-wise classification techniques. In the second level, an isosceles triangle is added enabling compensation in orientation and scale in tool model creation techniques and more detailed performance results by analyzing tool-tip position and tool orientation. The third level provides details regarding cases of occlusion with multiple polygons for one instrument (e.g. one polygon over the visible part and another one estimated over the extended instrument). The fourth level adds information to assess the quality of a tracking approach via temporal correspondences between bounding polygons in consecutive images, requiring image sequences. Finally, semantic attributes can be added for each instrument (fifth

level), further helping to describe the content visible by mentioning the presence of specular reflection, motion blur, or blood stain. The different levels of annotation details enable the creation of specific sub-sets in order to gain a better understanding of a tool detector behavior, for example focusing on occluded tools.

Table 3.8: Data annotation levels of detail.

	Single polygon	Isosceles triangle	Multiple polygons	Temporal corr.	Semantic attributes
Level 1	✔				
Level 2	✔	✔			
Level 3	✔	✔	✔		
Level 4	✔	✔	✔	✔	
Level 5	✔	✔	✔	✔	✔

Chapter 4

Surgical tool detection: Methods

Contents

4.1 Introduction

Towards tool detection from in-vivo surgical recordings, we chose to focus on approaches performing purely spatial image-based analysis, thus not relying on any kind of tracking or external marker support. Many state-of-the-art pedestrian detectors share a similar inability when it comes to surgical tool detection: how to deal with in-plane rotations. Extensive studies have been performed in order to detect in real-time pedestrian at multiple scales in videos, however extracted features are not invariant to rotation. Regarding surgical tool detectors, many different strategies have been followed either regarding the tool modelling task or the pose estimation task applying the model over input images. However, performing comparison between tool detectors is almost irrelevant due to the lack of validation data-set and methodology standardization. While it is hard to identify the best detection strategy to adopt, most tool detectors exhibit the use of prior knowledge to facilitate the pose estimation task. However, such strategy might impede a detector's ability to transfer to one surgical field or tool to another.
In section 4.2, we describe a first detection approach to transfer from real-word pedestrian to surgical instruments: the *adapted SquaresChnFtrs*. A second approach, the core contribution of this manuscript: the *ShapeDetector*, combining pedestrian and surgical tool detector strengths is presented in section 4.3.

4.2 First approach: *adapted SquaresChnFtrs*

4.2.1 Introduction

Dealing with in-plane rotations does not represent an issue for pedestrian detection in real life street scene recordings because gravity applies. Pedestrian most usually appear standing or walking, thus representing a uniformity in the pedestrian model with the head on top and feet at the bottom. Regarding surgical tools, surgeons hover them over anatomical structures and as such they can appear under different in-plane orientations or tilt inclinations. As a first solution to tackle the tool detection problem, we considered more fitting to start building upon an already performing detector from the literature, rather than creating a new detector from scratch. As such, finding the right pedestrian detector amongst the wide variety of state-of-the-art detectors was paramount. One approach stood out when performing a search with the three following criterion: low detection miss-rate, high speed and online source code availability; the *SquaresChnFtrs* detector from Benenson et al. [Benenson 2013]. Consequently, our first contribution is an adaptation of the *SquaresChnFtrs* detector to make it perform in a context of surgical tool detection. The *SquaresChnFtrs* detector belongs to the category of *one-stage* approaches, and follows a typical sliding window paradigm. At train time, image features are extracted and used to learn an object specific model. At test time, the object specific model is applied over input image features in a dense multiscale sliding window scanning followed by a NMS procedure. In section 4.2.2, we describe in details each

component of the vanilla *SquaresChnFtrs*, including the feature representation, the model learning strategy, the sliding window scanning, and optimization strategies. Next, in section 4.2.3, we describe our proposed adaptation enabling surgical tool detection.

4.2.2 SquaresChnFtrs

4.2.2.1 Feature representation

The feature representation, inspired by the *Integral Channel Features* from Dollar et al. [Dollár 2009a], combines three types of channels: color, gradient magnitude, and gradient histograms for a total of 10 channels. An illustration is given figure 4.1, when computed over a pedestrian input image.

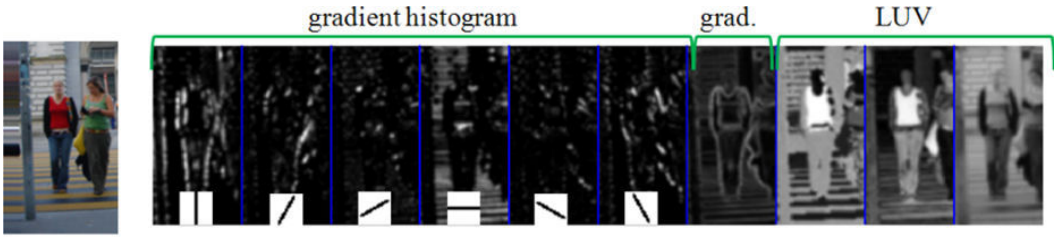


Figure 4.1: Example input image (left) and computed channels (taken from [Dollár 2009a]).

Color channels are extracted from the LUV color space because of its simplicity to compute and its attempt to perform perceptual color uniformity. Three channels are necessary for the representation, one for each color space component.

Gradient histogram channels are weighted histograms where bin index is determined by gradient angle and weight by gradient magnitude. Six channels are necessary for the representation, to model six different quantized orientations.

Gradient magnitude computed along gradient histograms is additionally stored in a separate channel.

Integral channels
Integral images (or summed area tables) have been introduced for object detection by the Viola and Jones detection framework [Viola 2001]. Using integral channel features representation, only three floating point operations are needed to compute a sum of pixel values within a rectangular region of a channel. This represents a fast and efficient way of evaluating features within the model window during the sliding window process. Creating integral images is quite simple and can be done in one pass considering equation 4.1 where I is the input image, S the integral image, x and y being pixel location.

$$S(x,y) = I(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1) \tag{4.1}$$

An input image and its corresponding integral version is illustrated figure 4.2[1]. Assuming A, B, C, and D the four corners of a rectangular region, with a clock-wise enumeration starting from the upper left corner. The corresponding features value over the region is obtained using equation 4.2.

$$Value = S(A) + S(D) - S(B) - S(C) \tag{4.2}$$

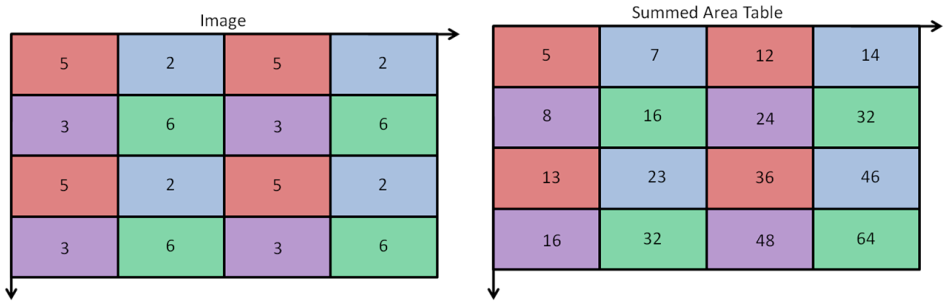


Figure 4.2: Integral image computation example. Input image (left) and integral image (right).

4.2.2.2 Model learning strategy

The model learning strategy employed belongs to the category of cascade classifier (e.g. Random Forests) and is very similar to the one proposed in the Viola and Jones framework. Low level features are built from summing over rectangular regions (i.e. pooling), then using boosting these rectangular regions are selected and assembled in a set of weak classifiers [Benenson 2013]. This set of weak classifiers, also called a strong classifier, refined by bootstrapping is the representation of the learned model.

Pooling

The feature pool is the set of rectangles used to construct the weak classifiers. Rectangle candidates are not obtained through careful design (i.e. regular pattern) but are randomly selected within the model window. Both the channel index and the rectangle size are arbitrary determined (enforcing a minimal area of 25 pixels). Following this pooling strategy, a total of 30000 rectangles was extracted and used as input to the boosting.

[1] https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/

Boosting
From the feature pool, a set of level-two decision trees (three stump classifiers per tree) was constructed and then linearly weighted using Adaboost. The final strong classifier consists of 2000 weak classifiers.

Bootstrapping
To increase the variability and difficulty of the training set, supplementary hard negatives are obtained via bootstrapping. The training starts with a set of 5000 random negative samples and then bootstraps twice, each time adding 5000 additional hard negative samples.

4.2.2.3 Sliding window scanning

At run-time, a full image detection is performed following a sliding window approach (see figure 4.3). On every image pixel, starting from the upper left corner, the model (i.e. cascade classifier) is applied over the corresponding image sub-region, corresponding to the model window size. As a result, a score representing the confidence to have the object present in the image sub-region is returned for the pixel location. In the end, each pixel location is associated with a confidence score. In order to limit the pool of candidate locations only to the most promising ones, a score thresholding is applied. The threshold value is empirically defined from score values range, depending on the classifier parameters. A further selection is performed via NMS to suppress multiple nearby detections.

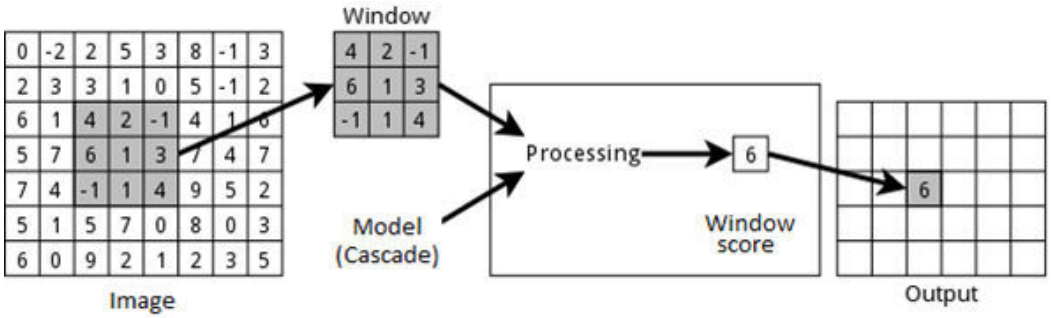
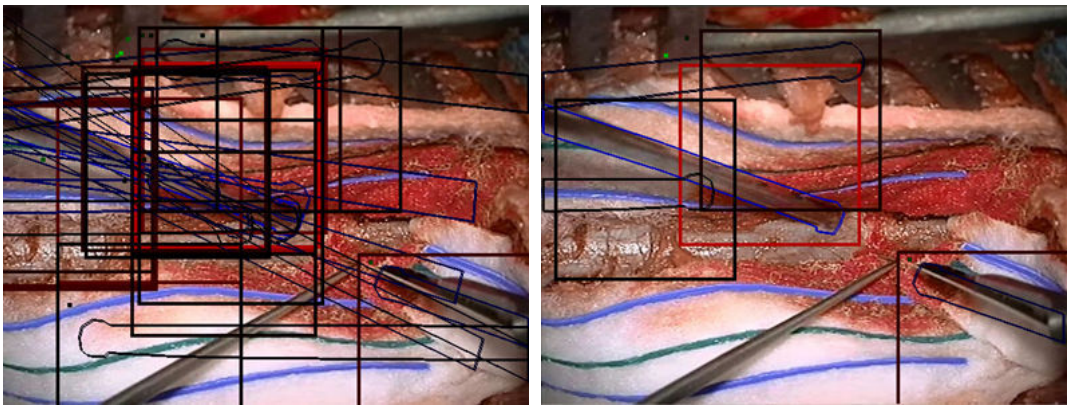


Figure 4.3: Sliding window process representation.

Non-Maximum Suppression
The NMS procedure belongs to the category of pairwise max suppression [Felzenszwalb 2008]. The less confident of every pair of detections that overlap sufficiently is suppressed. This simplified NMS procedure only requires one single parameter: the overlap threshold, set to 50% in practice. The overlap value is computed using the Intersection Over Union criterion. In figure 4.4 the NMS impact is illustrated .



(a) Without NMS (b) NMS

Figure 4.4: Illustration of the Non-Maximum Suppression procedure.

4.2.2.4 Optimization strategies

On every pixel location, all of the 2000 weak classifiers have to be evaluated before a confidence score can be returned. To accelerate the detections, the use of an early stopping scheme has been proposed: a soft-cascade [Zhang 2007]. The soft cascade aborts the evaluation of non-promising detections if the score of a given stage drops below a learned threshold. Thus, the number of weak classifiers needing to be evaluated at run-time can be reduced from 2000 (i.e. full cascade) to 50-100, which is eventually speeding up the object detection task. The soft cascade has to be manually tuned according to the initial cascade. Figure 4.5 illustrates side-by-side stage thresholds from the original cascade and the manually set soft cascade.

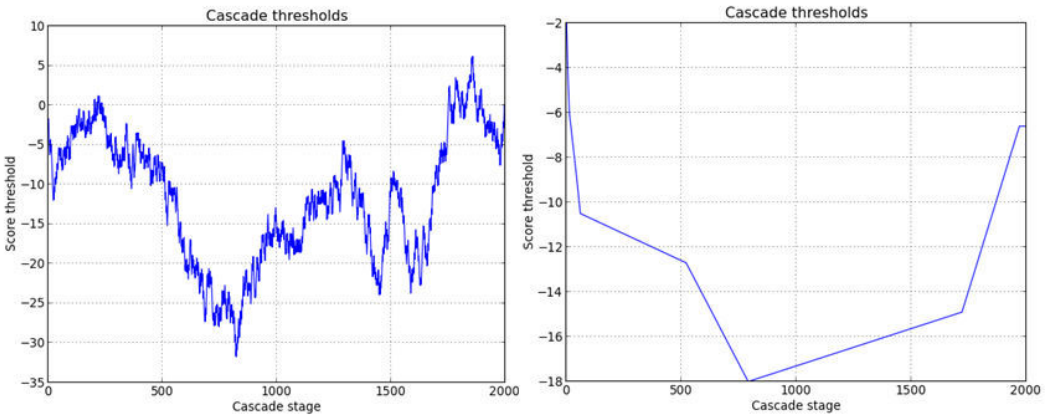


Figure 4.5: Illustration of an original cascade (left) and its corresponding soft cascade (right).

4.2.3 *Adapted SquaresChnFtrs*

To be able to accurately detect surgical instruments using the *SquaresChnFtrs* detector, it is necessary to take care of orientation related issues. First, we start by establishing a reference point for in-plane rotation (section 4.2.3.1). Then, we propose to replace bounding boxes by bounding polygons as output to obtain tighter detections (section 4.2.3.2). Finally, in sections 4.2.3.3 and 4.2.3.4, we present the creation of orientation-specific models and their aggregation into a multi-orientation bundle for run-time detection.

4.2.3.1 Orientation reference

Throughout the manuscript, the orientation of surgical instruments will be mentioned many times, as such we establish here what we consider to be the orientation of reference. We propose to define orientation 0° as representing the surgical instrument horizontally aligned with its tip facing the left side of the image, as shown in figure 4.6.

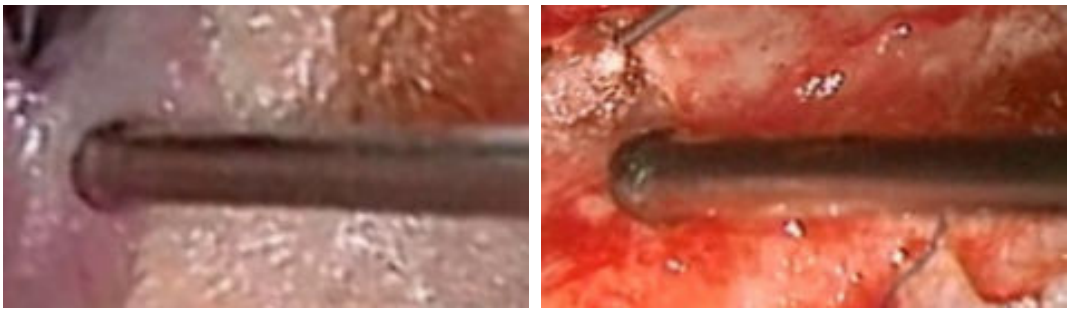


Figure 4.6: Suction tubes registered at orientation 0°.

4.2.3.2 Bounding polygons

Detections are usually being stored and displayed under the form of a rectangle, having the same size as the model window (i.e. size of training samples). However, because of in-plane rotation, such rectangles are not tight enough around tools. Not only does it affect the visual display, but it is also an hindrance for the NMS procedure. Consequently, we propose to use a geometry more fitting to instruments shape: a polygon. Figure 4.7 illustrates the difference between both geometries.

4.2.3.3 Orientation-specific model

In order to create an object model, training samples have to be aligned to compensate for translation and scale. For surgical instruments, a third compensation regarding the orientation has to be performed. To give more freedom to the model, generated training images are not strictly represented at the desired orientation, but

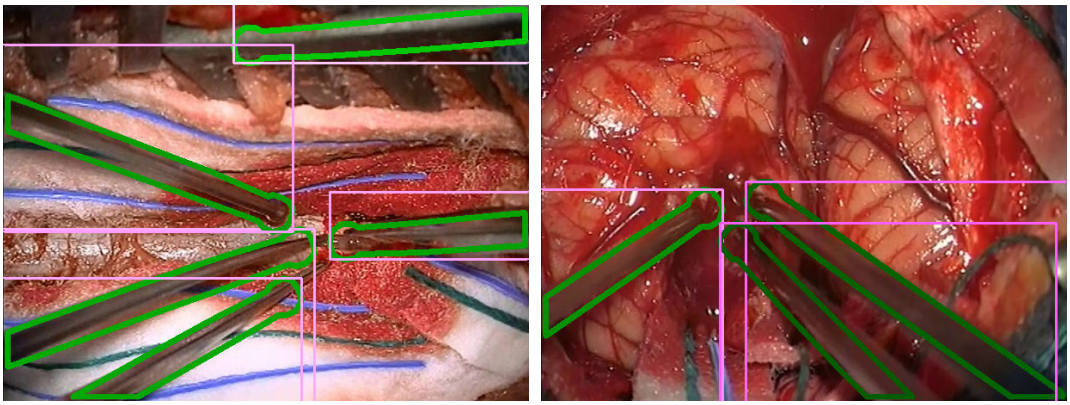
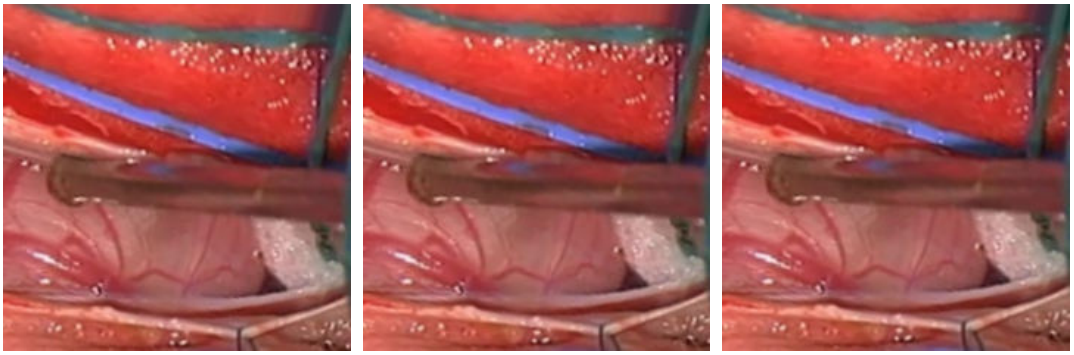
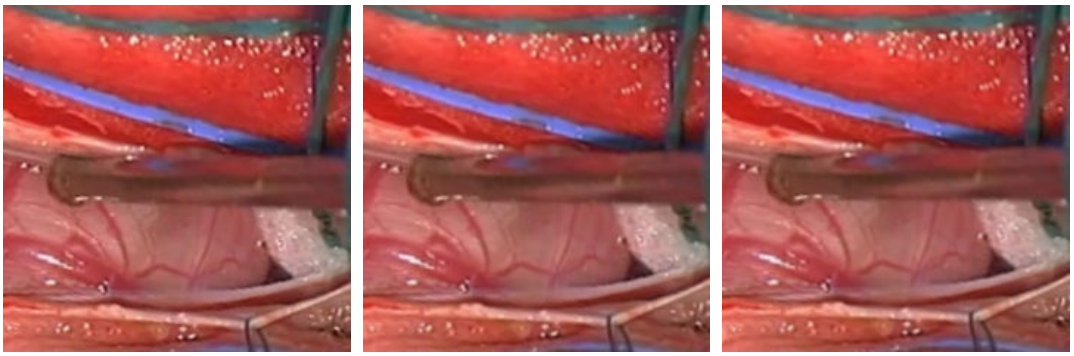


Figure 4.7: Bounding boxes (pink) versus bounding polygons (green) displayed over detected surgical tools.

are also spanning over a limited orientation range. We propose to learn a model for every 5°, as such six training samples are generated for each input sample in the range -3/+2° (see figure 4.8).



(a) Orientation -3° (b) Orientation -2° (c) Orientation -1°



(d) Orientation 0° (e) Orientation 1° (f) Orientation 2°

Figure 4.8: Suction tube training image examples for a model creation at orientation 0°.

In addition to the cascade classifier, an orientation-specific model is associated with a bounding polygon and its tip location. We used the SVG inkscape software [2] to manually create polygons, using as reference a training sample at the specific orientation. The polygon has to be perfectly placed within the model window in order to generate accurate detections during the sliding window process, as illustrated in figure 4.9.

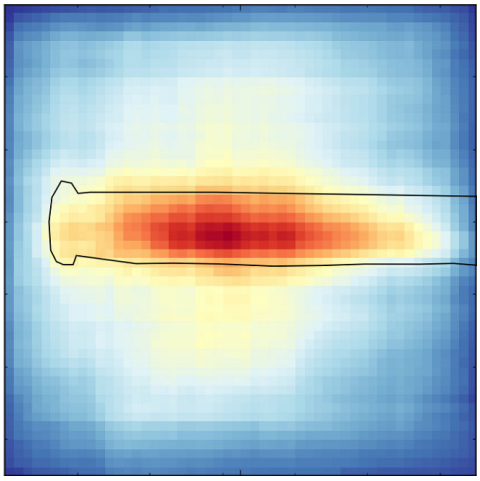


Figure 4.9: Bounding polygon (in black) placed over a suction tube model for the *Adapted SquaresChnFtrs*. The model size is 256×256 pixels.

4.2.3.4 Multi-orientation detection

Similarly to a multi-scale detection at run-time, we propose to perform a multi-orientation detection. While creating one orientation-specific model only and rotating input images multiple times could be a solution, the computational cost behind features re-computation would be far too important, especially regarding real-time detection. As such, we propose to transfer the time consuming effort from run-time to train-time by creating multiple models, each one covering a specific orientation. At test time, features are extracted once and input images are left untouched, while every orientation-specific model is processed one after the other in a sliding window fashion. For each orientation-specific model, a set of orientation-specific candidate detections is collected. In the end, multiple sets of orientation-specific candidate detections have been gathered and are sent to the NMS procedure.
A tool multi-orientation model is created as a bundle of multiple orientation-specific models. In the study, a bundle is made of 72 single orientation models as illustrated figure 4.10, with an orientation step of 5° between each.

[2] https://inkscape.org/fr/

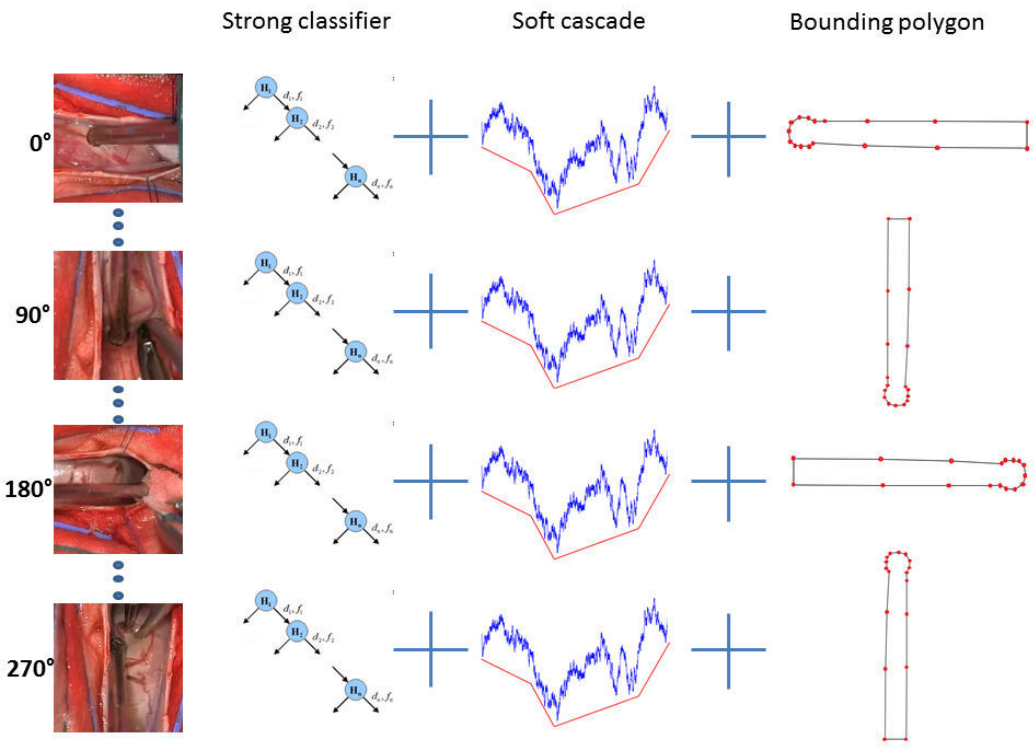


Figure 4.10: Illustration of a multi-orientation model for surgical instrument detection via the *adapted SquaresChnFtrs* approach. From left to right: example training image at the given tool orientation, representation of the cascade classifier, SoftCascade scheme, and bounding polygon.

4.3 Second approach: *ShapeDetector*

4.3.1 Introduction

Adapting a state-of-the-art pedestrian detector to make it perform in a surgical context was a first step, as representing a simple solution. While it provides encouraging results, reported in chapter 7, better performance can be achieved. Contextual information, presented as a strong component for pedestrian and tool detectors, is not handled yet. Moreover, although surgical tools usually do not have a distinctive color due to reflections, illumination variations, and grey tissue or texture, they do exhibit a distinctive local structure. Those reasons were motivations to integrate context within a tool detector pipeline.
Consequently, our second contribution belongs to the category of *two-stage* approaches, and is named: *ShapeDetector*. The first stage of the pipeline performs local appearance decisions by classifying each pixel into "tool" or "background" categories (steps 1 and 2 in figure 4.11). The second stage enforces the global shape by evaluating a tool-specific shape template (steps 3 in figure 4.11).

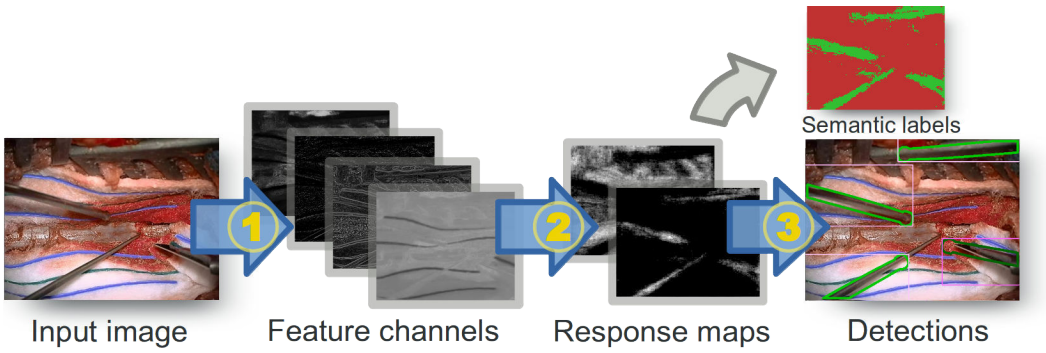


Figure 4.11: Overview of two-stage approaches pipeline. Step 1 computes a set of integral feature channel from the input image. Step 2 performs the pixel-wise classification (i.e. semantic labelling) for two classes: tool and background. Step 3 represents the pose estimation process using SVM shape models. Either as many response maps as classes, or a single map of semantic labels, are eligible as input for the pose estimation.

In the following, we first present the semantic labelling methodology, using the *SquaresChnFtrs* detector framework (section 4.3.2). Then, we describe multiple shape template creation processes (section 4.3.3). Finally, we introduce the second stage performing the pose estimation on top of the semantic labels, also in a sliding window fashion (section 4.3.4).

4.3.2 First stage: semantic labelling

In order to perform the multi-class pixel-wise classification, we propose to re-use the *SquaresChnFtrs* detector framework almost identically. We suggest to model two classes of pixels: the ones belonging to the surgical background and the ones belonging to surgical instruments. As such, one classifier is learned for each class, avoiding relying on a single sensitive threshold from a single classifier. In addition, learning as many classifiers as classes to model enables the possibility to perform a pixel-wise classification for more than two classes.
The principal structural change occurring to the *SquaresChnFtrs* framework is related to the extraction of training samples. As we move from global object modelling to pixel-wise modelling, the number of training samples does not correspond to the number of training images exhibiting the corresponding object. Moreover, the pooling strategy cannot occur within the model window, non-existing for pixel modelling, and a new strategy is necessary (section 4.3.2.1). Compared to the initial *SquaresChnFtrs* framework, a wider set of low-level feature channels are taken into account (section 4.3.2.2). The classifier itself is learned following the exact same pattern as used in the initial *SquaresChnFtrs* framework.

4.3.2.1 Training samples extraction

Training samples used for object detection can not be used here. A new set of training samples has to be generated for this task. For each training image, the corresponding training image for the semantic labelling is created by associating the right label to every pixel of the image (see figure 4.12).

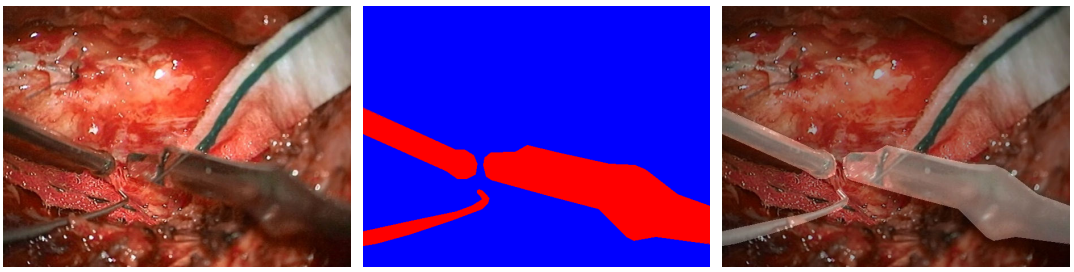


Figure 4.12: Original image (left), corresponding semantic labelling training image (middle), and overlaid semantic labelling onto the input image (right).

Using this set of semantic labelling training images, training samples (or patches) are obtained using a randomly perturbed regular grid. In order to obtain balanced sets for each class, an estimate number of samples is computed from available samples in each class. Otherwise, classes occurring frequently in the data-set or having a large spatial extent such as the surgical background tend to be favored. The sampling grid granularity is adapted automatically to fit the samples requirements. In case of unlabeled pixels in training images, they are ignored from the extraction process, as to avoid the risk of adding confusing examples.

4.3.2.2 Feature representation

In the initial *SquaresChnFtrs* framework, three types of feature channels were considered. For the pixel-wise classification, four different types of feature channels are used, with some extensions to the color feature type.

Gradient features are identical to the HOG feature representation introduced in the initial framework. Six channels corresponding to six different gradient orientations are considered, as well as a gradient magnitude channel.

Color features are represented either through the basic color representation consisting of three LUV channels, or an alternative to LUV: Color Names (CN) [Shahbaz Khan 2012]. The set of color attributes corresponds to linguistic color labels commonly assigned by humans to colors occurring in the world. Eleven name color channels are proposed: black, blue, brown, grey, green, orange, pink, purple, red, white and yellow.

Normalized location features are also considered by simple linear mapping between (x,y) pixel coordinates and real-valued interval [0,1]x[0,1].

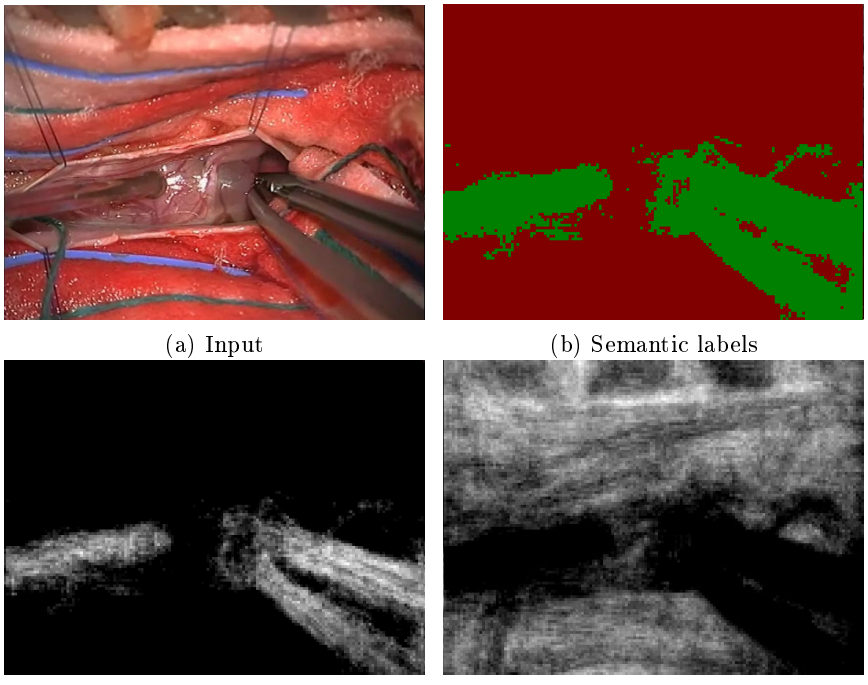
Linear filters are applied to the image, as proposed by Shotton et al. [Shotton 2006]: Gaussian filters at three different scales applied to the three CIElab channels, x- and y- derivatives of Gaussians at two different scales applied to the luminance channel, and Laplacian of Gaussians at four different scales, for a total of seventeen filter channels.

4.3.2.3 Output normalization

As a result from the pixel-wise classification over the input image, classifier outputs for each class are obtained. The distributions of outputs per class being approximated fairly well using Gaussians, outputs can be mapped to the desired range of [0,255] (see Response maps in figure 4.11). For clarity, we propose to define two terminologies: semantic scores and semantic labels.

Semantic scores represent the initial set of classifier outputs in the range [0,255], with as many response maps as classes modelled. Illustrated by figure 4.13c and 4.13d.

Semantic labels represent a single response map, obtained after post-processing the initial set of response maps. A pixel-wise argmax over each response map is operated, and the maximum score across all classes (i.e. maps) determines the label of a pixel. Illustrated by figure 4.13b, where green represents pixels being labelled as tool and red represents pixels being labelled as background.



(a) Input (b) Semantic labels

(c) Semantic scores (tool class) (d) Semantic scores (background class)

Figure 4.13: Semantic labels versus semantic scores illustration.

4.3.3 Shape-template model creation

In this section, two different shape template creation strategies are introduced. Conversely to one-stage approaches, as it is for the *adapted SquaresChnFtrs* detector, a model is not derived from original images. Here, semantic labelling outputs are used as input to the shape template creation process. A shared similarity emerges from the necessity for all positive samples (i.e. showing a surgical tool) to be aligned beforehand, to compensate for translation, scale and rotation. Thanks to this compensation, a model can be learned at any orientation or scale desired. For easier training, a model can be up-/down- sampled or rotated in a post-processing fashion. As such, only one set of training samples needs to be created in order to obtain shape templates at various scales and orientations for the surgical instrument of interest. Below, we start by mentioning an hand-crafted model strategy (section 4.3.3.1). Then, the main shape template creation approach, using a SVM and completely data-driven, is described in section 4.3.3.2. Finally, we report in section 4.3.3.3 necessary elements to build the final *ShapeDetector* model for ensuing processing by the second stage of the pipeline.

4.3.3.1 Fixed template

The first proposed template creation strategy is simple and ad-hoc, using only a polygon example of the tool of interest as input. In an rectangular envelope around the polygon, a weight of +1 is set to pixels within a 5-pixel distance of the polygon edges. A weight of -1 is given to pixel ranging from a 6-pixel to a 20-pixel distance from the polygon edges. Any other pixel within the rectangle is attributed a weight of 0. Figure 4.14 illustrates such shape templates for a suction tube and a bipolar forceps.

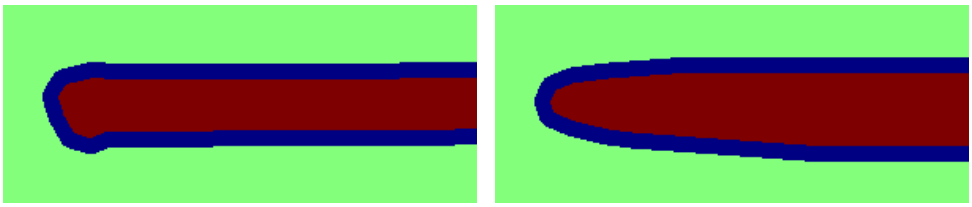


Figure 4.14: Fixed shape template examples for a suction tube (left) and a bipolar forceps (right). Weights color code: red is +1, blue -1, and green 0.

4.3.3.2 Data-driven SVM template

For this data-driven approach, we propose to learn the shape template model by using a linear Support Vector Machine. Surgical instruments having a spatial coherence, we consider regularizing the SVM training by adding a two-dimensional spatial smoothness prior to also enforce a spatial coherence within the shape model.

Below, we explain in details the SVM regularization process and the creation of positive and negative samples for the SVM training.

SVM regularization

The regularization term is an important element for SVM training. Since we know that we are operating a two-dimensional domain, we consider modifying the vanilla SVM equation (4.3) (see [Burges 1998]). We include a regularization term M (equation 4.4) that promotes a 2d spatial smoothness prior [Lehmann 2011].

$$\min_{w \in \mathbb{R}^m} \; w^T w + \frac{C}{|T|} \sum_{t \in T} L\left(y_t, < x_t, w >\right) \tag{4.3}$$

$$\min_{w \in \mathbb{R}^m} \; w^T M w + \frac{C}{|T|} \sum_{t \in T} L\left(y_t, < x_t, w >\right) \tag{4.4}$$

where $T = \{x_t, y_t\}_{t=1}^{|T|}$ are instance-label pairs, $L : \{0, 1\} \times \mathbb{R} \rightarrow \mathbb{R}_0^+$ is the loss function and C is a penalty parameter. The matrix M can be decomposed as shown in equation 4.5. The regularization matrix R encodes the 2d spatial structure.

$$M = R^T \cdot R \tag{4.5}$$

In equations 4.6, we develop the link between the standard SVM formulation and the one using regularization via R. It can be seen that the 2d prior can be encoded via a simple transformation of the input data (via R^{-1}), allowing the use of unmodified SVM training code. At test time, we use the resulting w, without needing to change the input data.

$$\begin{gathered}
w^T M w + \tfrac{C}{|T|} \sum_{t \in T} L\left(y_t, < x_t, w >\right) \\
w^T (R^T R) w + \tfrac{C}{|T|} \sum_{t \in T} L\left(y_t, < x_t, (R^{-1} R) w >\right) \\
(Rw)^T (Rw) + \tfrac{C}{|T|} \sum_{t \in T} L\left(y_t, < x_t, R^{-1}(Rw) >\right) \\
\tilde{w}^T \tilde{w} + \tfrac{C}{|T|} \sum_{t \in T} L\left(y_t, < x_t, R^{-1} \tilde{w} >\right)
\end{gathered}$$

$$w = R^{-1} \cdot \tilde{w} \tag{4.6}$$

The 2d spatial smoothness in the regularization matrix is performed by enforcing 4-connex pixels to have close values. In case of a 4-pixel image (represented by a, b, c, and d), the regularization matrix to use is represented in equation 4.7. For comparison, equations 4.8 and 4.9 illustrate the regularization term with and without 2d spatial smoothness.

$$w = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}, \; \tilde{w} = R \cdot w = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{4.7}$$

$$\begin{aligned} \tilde{w}^T \cdot \tilde{w} &= a^2 + b^2 + c^2 + d^2 + \\ &\quad (a-b)^2 + (a-c)^2 + (b-d)^2 + (c-d)^2 \end{aligned} \tag{4.8}$$

$$w^T \cdot w = a^2 + b^2 + c^2 + d^2 \tag{4.9}$$

Training positive samples

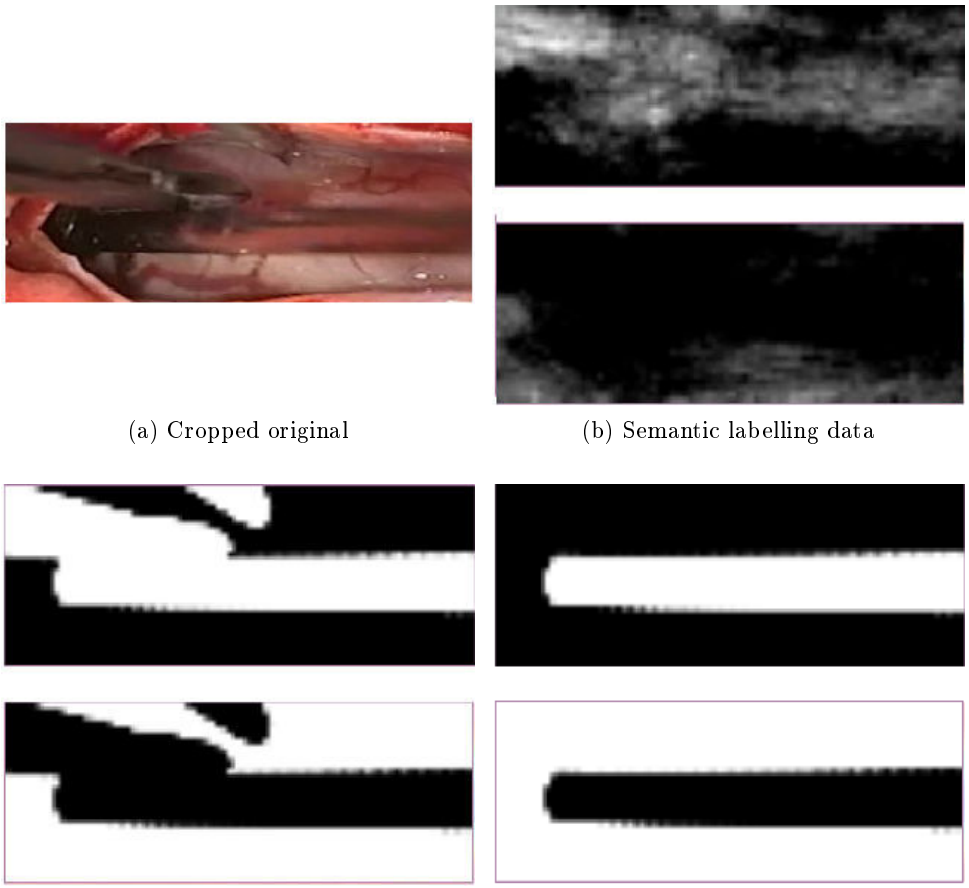
From the set of annotations provided with the data-set described in chapter 3, it is possible to generate different training samples. We consider three alternatives on how exactly we choose to generate the training samples, from the same original image (see figure 4.15a):

1. *SquaresChnFtrs* semantic labelling maps (see figure 4.15b).
2. Annotation masks of all surgical instruments (see figure 4.15c).
3. Annotation masks of a single surgical instrument (see figure 4.15d).

Semantic labelling maps from alternative (1) best represent the data the classifier will receive a test time as they are the direct results from the first stage. However, they are somewhat noisy, making it more difficult to accurately learn the shape of a surgical instrument.

To overcome this issue, we propose with alternative (2) to create binary masks using surgical instrument annotations. Those masks can be considered as a perfect/ideal semantic labelling result where all the noise has disappeared. To generate tool class training masks, every annotated surgical instrument is assigned the white color and the rest of the image is in black. Corresponding background class training masks are obtained as being the opposite/complementary images.

For alternative (3), we also want to take advantage of an ideal semantic labelling case, yet learning the surgical instrument shape only and not modeling surrounding instruments. As such, only the tool of interest is in white in tool class training masks and the rest of the image, including neighboring surgical instruments, is in black.



(a) Cropped original (b) Semantic labelling data

(c) All instruments annotation (d) Single instrument annotation

Figure 4.15: SVM positive samples for the suction tube. Top rows correspond to the tool class and bottom rows to the background class.

Obviously, training samples can be generated either to represent semantic scores (i.e. one mask per class) or to represent semantic labels (i.e. only one overall mask). For the former, background class samples are generated as specified above for alternatives (2) and (3). For the latter, no background class samples are generated for alternatives (2) and (3).

Training negative samples

Negative training samples are created from scratch in a random manner, following one of the three distribution sampling alternatives:

1. Binary sampling: uniform binary distribution where pixels can only have the value 0 or the value 255 (see figure 4.16a).

2. Grayscale sampling: uniform distribution where pixels can have a value in the range [0, 255] (see figure 4.16b).

3. Gaussian sampling: Gaussian distribution where pixels can have a value in the range [0, 1], then re-scaled in the range [0, 255] (see figures 4.16c and 4.16d).

(a) Binary distribution (b) Grayscale distribution

(c) Gaussian distribution (Tool) (d) Gaussian distribution (Background)

Figure 4.16: SVM negative inputs depending on the sampling strategy.

Compared to the naive alternative (1), alternative (2) better matches with positive samples, while alternative (3) is meant to close this gap even further by modelling pixels distribution from real semantic labelling results.
Negative samples should always have the same size as positive samples, be it for the image resolution but also for the number of classes, in case of semantic scores modelling. For alternatives (1) and (2), the tool class sample is randomly sampled, and the background class sample is obtained as the opposite image. For alternative (3), each class is modelled by its own Gaussian distribution, as such each class sample is randomly obtained from the corresponding distribution.
Selecting cropped images as negative samples could also have been a solution, however the pooling strategy is hard to put in place. In semantic labelling images, large portions are totally uniform aside from some noise, thus many negative samples randomly extracted could be almost identical and not discriminating enough for the training procedure.

4.3.3.3 *ShapeDetector* model

To be processed by the second stage of the pipeline, a global *ShapeDetector* model needs to be created. The following elements/parameters are included in the global model, displayed figure 4.17:

- Shape-template model: core element representing the modelled tool shape, which will be evaluated on each image pixel during the sliding window process.
- Bounding polygon: generic geometrical shape representing the tool, which is mandatory for the NMS procedure and results displaying.

- Tool-tip location: element required to perform the corresponding validation.
- Tool orientation/scale: two parameters necessary to perform detection at multiple orientations and scales.
- Tool class: imperative information as a first step towards the classification process when different tool models are ran in parallel.

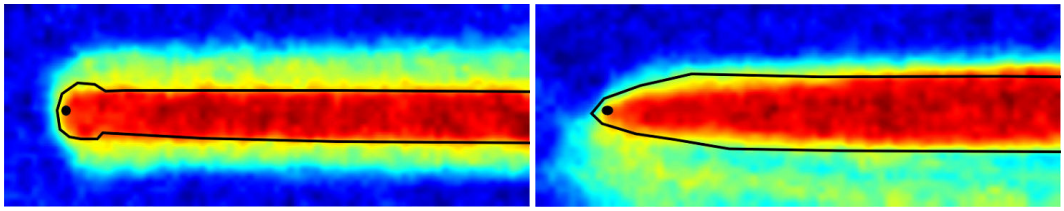


Figure 4.17: Illustration of the final *ShapeDetector* model for a suction tube (left) and a the upper part of a bipolar forceps (right). The bounding polygon and tool-tip location are represented in black, other colors are representing the SVM shape model.

4.3.4 Second stage: pose estimation

In the second stage of the pipeline, we propose to perform tool pose estimation by capturing the global shape of specific tools, using rigid templates. For each surgical instrument, a shape model has been learned over a set of normalized pose images. This learning-driven approach makes no assumption about the texture or the shape of the objects.
We chose not to use semantic labels as additional feature channels, as such the pose estimation is performed on top of semantic labelling results in a sliding window fashion. By conducing an exhaustive search, we can detect an arbitrary number of surgical instruments, at any position and orientation in the image. Each trained shape template is transformed for each desired scale and orientation (similar to [Benenson 2012]) at the very beginning (see section 4.3.4.1). This speeds test time computation up, since it avoids the need to recompute the semantic labelling at different scales and orientations (i.e. features extraction and pixel-wise classification). Assuming a restricted depth range, semantic labelling can be applied over the image at a single scale, which is common practice in street scene labelling [Ess 2009b].
Each shape-template is approximated piece-wise via a set of squares (see section 4.3.4.3), thus enabling the use of semantic labels integral images when evaluating the correlation of each scale/orientation specific template. Using integral channels makes the computation cost of the sliding window independent of the template scale, meaning that searching for small instruments costs as much as looking for large ones.

Each candidate detection consists of a score, a bounding polygon on the hypothesized object and a tool-tip position, coming from the set of information contained in the *ShapeDetector* model. Finally, the candidates are filtered using a greedy NMS procedure based on their score, polygonal representation overlap, and orientation (see section 4.3.4.2).

4.3.4.1 Search space setup

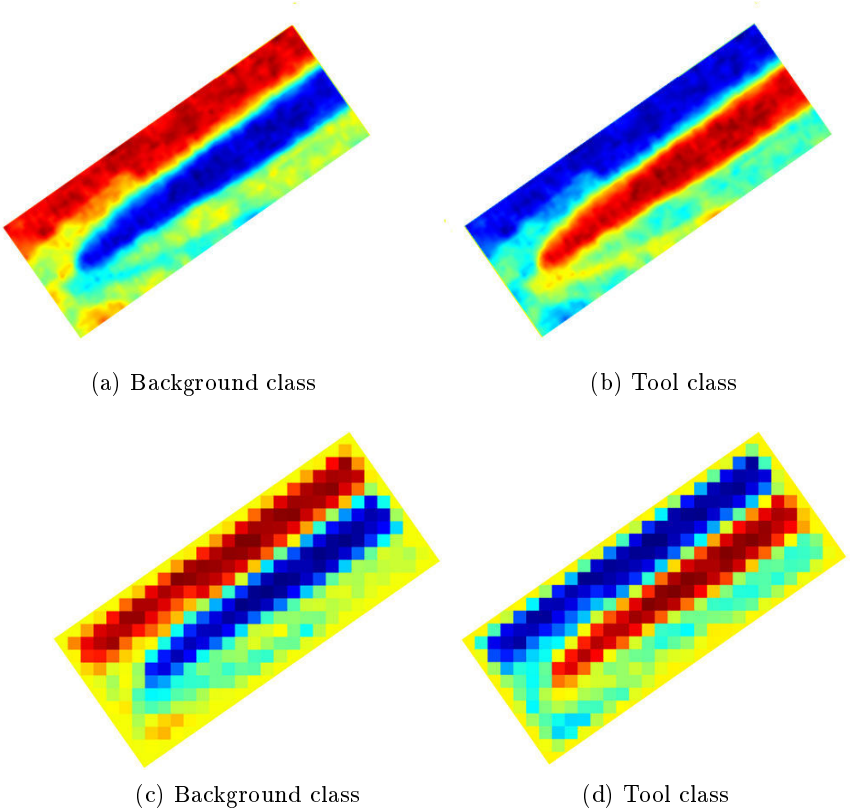
Before starting the sliding window process, an exhaustive set of shape templates is created from the initially learned model, to cover every scale and orientation specified by the search space.
The model scale is based on one parameter only, a proportionality coefficient compared to the initial model. This enables detecting surgical tools at different scales, corresponding to different microscope zoom values. However, the scale parameter is impacting the tool shaft width only, and it is not possible to specifically extend the model to look for larger or shorter tools. The model orientation is represented by one parameter, being an angle value in our orientation referential.

4.3.4.2 Non-Maximum Suppression

To eliminate spurious detection hypotheses, a form of greedy NMS is applied, suppressing multiple nearby detections. The NMS procedure removes the less confident of every pair of detections that overlap sufficiently according the Intersection Over Union criterion [Everingham 2010], but only if the difference in orientation is higher than an empirically defined threshold. Our NMS is thus represented by two parameters: the overlap threshold and the orientation difference threshold. By setting the latter threshold to 0, the simplified NMS procedure as presented in [Dollár 2011] is retrieved.

4.3.4.3 Model piece-wise approximation

To be combined with integral images, each shape-template is approximated piecewise via a set of squares (see figure 4.18). Such approximation also reduces the number of operations required to evaluate the model on a given image location, thus making the overall process faster.
To perform the approximation, the initial SVM model window is sub-divided into 15×15 pixel squares, after addition of extra padding to avoid uneven square size division. A new weight is set for each piece, computed by averaging SVM values within the square. For the number of pieces created to be stable across the various scales processed, the model scale coefficient is applied to the square size.



(a) Background class (b) Tool class

(c) Background class (d) Tool class

Figure 4.18: Visual effect of the piece-wise approximation over the SVM model of a bipolar forceps. Top row displays the original model and bottom row the approximated model.

4.4 Conclusion

In this chapter, we presented two novel image-based approaches for surgical tool detection. The first one, the *adapted SquaresChnFtrs*, is a direct extension of an efficient, robust, and real-time state-of-the-art object detector. Following a one-stage framework, the method is able to compensate for tools undergoing in-plane rotations. The second approach, the *ShapeDetector*, is a novel approach following a two-stage pipeline framework and re-using the *SquareChnFtrs* framework to perform the pixel-wise classification. The pose estimation over semantic labelling result maps is performed using data-driven SVM tool shape templates in an exhaustive sliding window fashion. The approach does not rely in its design on prior knowledge regarding the number of surgical tools, their shape, or position in the image. Through the use of bounding polygons instead of bounding boxes to represent candidate detections, both approaches enable to retrieve tool pose more accurately. Detection performances obtained using the aforementioned techniques are reported in chapter 7.

Chapter 5

Surgical tool detection: Validation studies

Contents

5.1 Introduction

As stated in chapter 2, there is in general no single correct strategy to assess detector performance. However, a proper validation methodology should aim to quantify detector performance in a realistic, unbiased, and informative manner. Additionally, detectors should be ranked and compared with one another, to the possible extent. At the very least, a detector performance should be put in perspective with competitive baseline approaches from the same research field. In this chapter, we propose our own validation methodology, closely mimicking the best acknowledged methodologies in computer vision.
In section 5.2, we detail the specification phase of the validation methodology, specifying the conditions in which the validation was performed. Then, we describe a set

of computational metrics in section 5.3. Finally, in section 5.4, we present multiple baseline approaches.

5.2 Specification phase

Detectors are trained using our *NeuroSurgicalTools* data-set (presented chapter 3) under the model validation technique presented in section 5.2.1. Validation is performed following the full image analysis described in section 5.2.2, and results are reported as described in section 5.2.3.

5.2.1 Model validation technique

As a remainder, the *NeuroSurgicalTools* data-set has been equitably split into training and testing sets. For our model validation technique, we chose one of the four training/testing scenarios proposed by Dollar et al. [Dollár 2011], which is named *scenario cal1* in their paper.
In other words, the *NeuroSurgicalTools* training set is used to train detectors while detection results are collected over the *NeuroSurgicalTools* testing set.

5.2.2 Full image analysis

Similarly to the scheme laid out in the review paper from Dollar et al. [Dollár 2011], we chose to perform validation following the single frame analysis protocol. In general, an object detector takes an image as input and returns for each detection a Bounding Geometry (BG) at the image location (e.g. bounding box, bounding polygon) coupled to a confidence score. The validation is therefore being performed using the final list of detections (BG_{dt}) and the corresponding list of ground truth (BG_{gt}).
Each BG_{dt} and BG_{gt} may be matched at most once, using a greedy matching strategy to resolve the assignment ambiguity. Detections with highest confidence scores are prioritized and matched first. In case of a BG_{dt} matching multiple BG_{gt}, the match with highest overlap value is kept using the PASCAL measure described in section 5.3.1.
In this work, we aim at detecting surgical tools, in other words distinguishing surgical instruments from the surgical background, and leave aside the problem of tool categorization. When validating the detection of a specific tool, we ignore all false positives on other annotated tools. Only unmatched BG_{dt} of the specific tool class are accounted for as False Positives. This is similar to the protocol used for pedestrian detection [Dollár 2009a], where regions with "crowds" triggering false positives for pedestrians are ignored. False positives on other tools are considered part of the fine-grained tool classification left for future work.

5.2.3 Results reporting

To report and compare detector performance, we do not rely on standard recall/precision curves. Instead, we prefer to plot (using log plots) miss rate against FPPI by varying the threshold on detection score.
To summarize detector performance, we use the LAMR computed by averaging miss rate at nine FPPI rates evenly spaced (in the log-space) in the range $[10^{-2}, 10^{0}]$. In case of curves ending before reaching a given FPPI rate, the minimum miss rate achieved is used [Dollár 2011].
As a side-note, our *adapted SquaresChnFtrs* first proposed detector is reported under the name of *SquaresChnFtrs* in plots of the Results chapter (chapter 7) for better visualization.

5.3 Validation metrics

The validation criterion considered in this work aims at characterizing accuracy and precision properties of detection methods. The three elements describing the quantification of the validation criterion (i.e. metric, reference, figure of merit) are presented altogether in the following under the term of validation metric.
We consider three validation metrics to assess a detector pose estimation performance: a first one regarding the overall bounding geometry (section 5.3.1), a second one regarding the tool orientation (section 5.3.2), and a third one regarding the tool-tip position (section 5.3.3). We also propose in section 5.3.4 another metric dedicated to the performance of the *semantic labelling* stage of our proposed *ShapeDetector*.

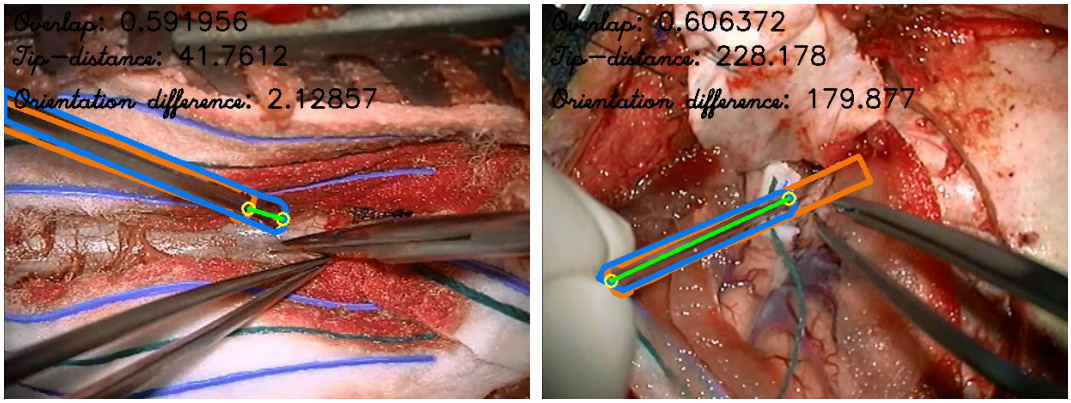


Figure 5.1: Visual representation of validation metrics computation between the blue BG_{gt} and the orange BG_{dt}. Tool-tips are symbolized by a yellow circle and the green line represents the tip to tip distance.

Figure 5.1 illustrates for two different images the metrics validating the pose estimation. Metrics are computed between the blue BG_{gt} and the orange BG_{dt},

each tool-tip being symbolized by a yellow circle. The green line represents the distance between both tip locations.

5.3.1 Polygon overlap

The *polygon overlap* metric proposes to study the tool global position, which is the go-to metric for pedestrian detectors. Due to surgical instruments' elongated shapes, we decide to use bounding polygons cropped to image borders, instead of standard bounding boxes. We use the traditional Intersection Over Union criterion [Everingham 2014] to count false positives and false negatives (see equation 5.1). Since a small difference in orientation between two elongated polygons leads to small overlapping areas, we consider true detections the ones with an overlap of at least 25% with the ground truth, instead of the traditional 50% threshold. We will show in chapter 7 detector performance stability until a 60% area threshold.

$$Overlap = \frac{area(BG_{dt} \cap BG_{gt})}{area(BG_{dt} \cup BG_{gt})} > 25\% \tag{5.1}$$

5.3.2 Orientation difference

Given many in-plane tool rotations during surgeries, we compute for every true detection BG_{dt} obtained at a fixed rate of 10^{-1} FPPI the error in the orientation estimation with the corresponding reference BG_{gt}. Results are plotted as probability versus orientation difference (in degrees).

5.3.3 Tool-tip distance

In some applications, the tool-tip position is more relevant than the tool-body pose estimation. We can thus measure the Euclidean distance between a detection BG_{dt} and its corresponding reference BG_{gt}. To ensure meaningful results, we compare methods at a fixed rate of 10^{-1} FPPI, and propose to disregard detections deviating by more than 45° from the ground truth. This measure is optimistic over the estimated accuracy given many false positives, but gives an upper bound on the tool-tip precision when detections are correct. In chapter 7, we will show the limited impact of the orientation threshold over reported tool-tip distances.

5.3.4 Segmentation quality

For reporting multi-class semantic labelling results, two metrics are widely used: per-pixel and per-class averages. Both those metrics are computed by performing a full image pixel-wise comparison between semantic labels (i.e. single map output) and ground truth labels.
For the per-pixel metric, the overall percentage of correctly-classified pixels is computed, all classes included. For the per-class metric, per-class percentages of correctly-classified pixels are measured, then averaged altogether.

5.4 Baseline

To understand the difficulty of detecting surgical tools from in-vivo surgery images, we re-used or implemented different baselines for comparison with our proposed methods. In sections 5.4.1, 5.4.2, and 5.4.3, semantic labelling, one-stage and two-stage baseline approaches are respectively introduced. For completeness, we also detail in section 5.4.4 multiple variants of our proposed *ShapeDetector* approach.

5.4.1 Semantic labelling: Darwin

We used the Darwin framework, introduced in [Gould 2012], as a baseline for the semantic labelling task.
The classifier approach is inspired by [Shotton 2006] and is based on boosted decision trees built on top of features comprising filter banks, HOG and RGB color. Features are heavily hand-crafted and choice of pooling regions is not randomly performed.

5.4.2 One-stage: Linemod

As a representative for one-stage approach baseline, we chose the Line-mod detector [Hinterstoisser 2010], based on a fast matching of oriented gradient templates. It is a well-known and open-source detector, already packaged and ready to use as part of the OpenCV libraries. This technique is supposed to perform well and thus serves as a good baseline because surgical instruments are mainly texture-less objects.

5.4.3 Two-stage: *Skeleton*

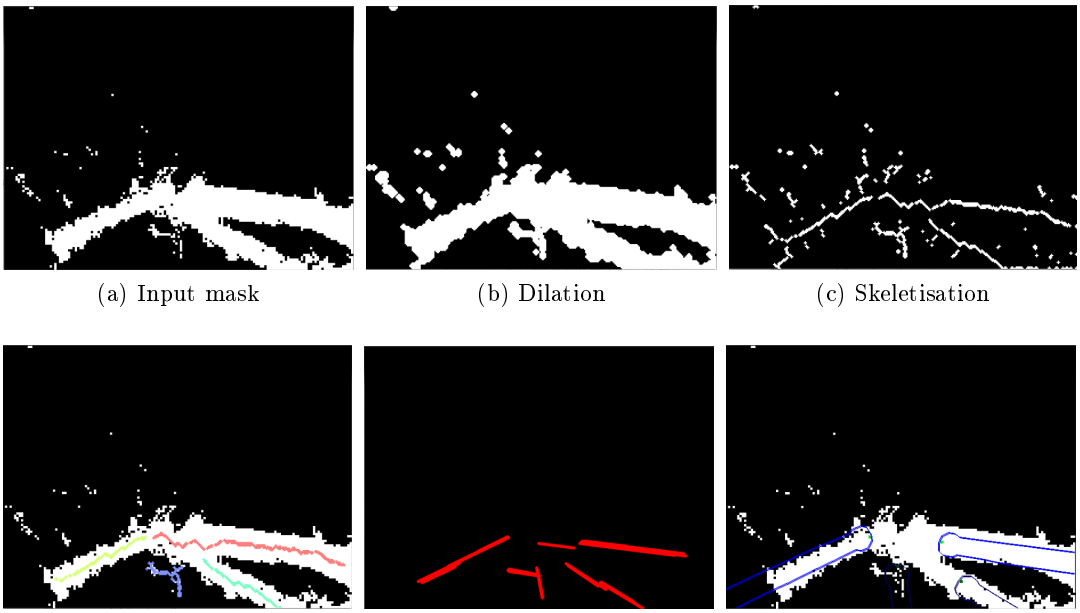
As a first two-stage baseline, we implemented a naive method based on mathematical morphology operations. Heavily hand-crafted for tool detection, this method exploits the geometry of surgical instruments by searching exclusively for tubular shapes. An overview of the *Skeleton* pipeline is shown figure 5.2.

Inputs

The semantic labels (i.e. single map output) coming from the first stage are used as input for this method. Pixels labelled as belonging to the background class are set to black and pixels labelled as belonging to the tool class are set to white, thus giving a binary black and white mask (illustrated figure 5.2a).

Process

The first mathematical morphology operation consists in a double dilation on the input mask, using a structuring element of size 5x5 (refer to figure 5.2b for the result of the dilation). The double dilation is necessary as to reduce labelling noise visible in the input mask as much as possible .
Tubular shapes needing to be identified and counted, we thus extract topological skeletons [Zhao 1991] in order to summarize the tool presence evidence. Indeed, as



(a) Input mask (b) Dilation (c) Skeletisation

(d) Connected components labelling (e) Hough transform (f) Output detections

Figure 5.2: Illustrated Skeleton approach workflow.

we consider tubular shapes only, they can be reduced to their barycentre lines or "skeletons" (see figure 5.2c). Assuming a minimal size for surgical instruments in the image can be set, an additional noise reduction step is performed. After computing connected components, only skeletons with a length longer than an empirical threshold are kept (illustrated figure 5.2d). From the set of remaining connected components, we estimate straight lines using the Hough transform (see figure 5.2e). Each line obtained, and longer than a specific threshold, is considered as a candidate detection and enriched with a bounding polygon and a score computed proportionally to the line length.

Post-processing

Finally, a greedy NMS iteration is performed based on candidate detection scores for a final set of detections presented figure 5.2f.

5.4.4 Two-stage: *ShapeDetector* variants

In section 4.3, we presented the methodology of our proposed two-stage pipeline: the *ShapeDetector*. Following the pipeline, various semantic labelling methods can be employed, two types of semantic labelling outputs can be leveraged, and two shape-template creation processes have been introduced. As such, many variants of the *ShapeDetector* can be instantiated, but we decided to limit ourselves to three representative ones.

FixedTemplate : The *SquaresChnFtrs* framework is used to perform the semantic labelling, combined with a fixed shape template model processed over semantic labels (i.e. single map).

DarwinDetector : The Darwin framework is used to perform the semantic labelling, combined with a fixed shape template model processed over semantic labels (i.e. single map).

ShapeDetector : The *SquaresChnFtrs* framework is used to perform the semantic labelling, combined with the data-driven SVM shape template model processed over semantic scores (i.e. multiple maps).

FixedTemplate and *DarwinDetector* instances can highlight the impact of employing different techniques to perform the pixel-wise classification. The impact of performing differently the shape template model creation will be underlined by *FixedTemplate* and *ShapeDetector* instances. While we consider *ShapeDetector* to be the name of our proposed two-stage detector pipeline, it also refers to the variant achieving best performance.

5.5 Conclusion

Although no single correct way to perform the quantification of a validation criterion exists, defining a proper validation methodology is crucial. To that extent, this chapter provides a detailed description of our proposed methodology designed with the aim to quantify and rank detection approaches in a realistic, unbiased, and informative manner. A set of competitive baselines has been re-used or implemented in order to provide perspective in detector performance. Obtained results following our proposed validation framework are reported in chapter 7.

Chapter 6

Towards real-time detection

Contents

6.1 Introduction

For in-vivo medical applications, while tool detectors must provide highly accurate results, they also should be able to run at a speed close to the recording device frame-rate (i.e. around $25\,Hz$-$30\,Hz$). In order to decrease the processing time, two strategies can be employed: code optimization and ad-hoc optimization strategies. The former relies on CPU and GPU performances to obtain the exact same detector results with better time performance. The speed gain is also being heavily correlated to computer hardware specifications: the more CPU/GPU CUDA cores, the faster

the detector. The latter usually obtains speed gain at the cost of detection accuracy and should be traded with more carefully.
In this chapter, we present a fast implementation (in C++ programming language) of our *ShapeDetector* detector. We especially focus on optimizing the pose estimation step of the pipeline (i.e. second stage) as the semantic labelling (i.e. first stage) is already performed on GPU thanks to the *SquaresChnFtrs* framework. We propose to start by giving some insights about good programming practice, which can be applied to any application (section 6.2). Then, we propose in sections 6.3 and 6.4 our code optimization choices for a CPU implementation and a GPU one. Finally, in section 6.5, we introduce some ad-hoc optimization strategies employed to further increase the processing speed.

6.2 Good programming practice

Before performing deep code optimization, which can be a cumbersome process be it in CPU or GPU, some generic good programming practice can be considered as a first layer of optimization. Usually, they can be applied without deep C++ programming knowledge and do not require extensive code modifications. All of the following programming advice have been employed towards *ShapeDetector* speed-up.

Compiler options
As a starter, your compiler is smart and can do a lot of code optimization on its own. As such, the following compiler options should always be prioritized: -march=native -O3.

Inline functions
At compile time, the compiler places a copy of the function body at each point in the code where the function is called. Using inline functions save the overhead of function invocation and return (i.e. register saving and restore) by avoiding a jump to a sub-routine.
Such functions are usually only declared in header files with the *inline* tag. Occasionally, they can be declared within source files, and have to placed before any functions invoking them (i.e. preferably at the beginning of the file).

Loop unrolling
Using a loop unrolling strategy optimizes a program's execution speed at the expense of its binary size (i.e. space versus speed trade-off). It reduces pointer arithmetic, end of loop tests, branch penalties and hide latencies (i.e. delay to read data from memory). As you can see in Figure 6.1, if/else condition tests can be skipped by increasing the pointer by 2 instead of 1.

Bottlenecks identification
Identifying instructions or code blocks requiring the most computational time is

Normal loop

```
int res= 0;
int x;
for(x = 0; x < 100; ++x)
{
    if( (x%2) == 0)
        res += x;
    else
        res += 2*x;
}
```

After loop unrolling

```
int res= 0;
int x;
for(x = 0; x < 100; x+=2)
{
    res += x;
    res += 2*(x+1);
}
```

Figure 6.1: Loop-unrolling example applied to remove unnecessary if/else condition tests.

paramount to be able to perform code optimization, such code blocks are called bottlenecks. While bottlenecks can probably be identified with experience (through educated guess), it is best in general no to assume their location and always measure speed using dedicated code profilers. As of today, the Intel VTunes code profiler is the best available for CPU bottlenecks identification, since originating from the same company manufacturing CPUs. To learn more about CPU code profilers, please refer to the appendix A.

6.3 CPU optimization

By default, every program/application is being ran on the CPU. As such, we introduce in details our proposed CPU optimization choices. We start by a presentation of CPU's hardware and their architecture in section 6.3.1. Then, in section 6.3.2, we explain how to perform CPU code modifications to make an efficient use of CPU architecture. Finally, in section 6.3.3, we introduce SIMD instructions enabling to perform parallel computing on a CPU.

6.3.1 CPU architecture

When working with a computer, numerical data can be stored within five different physical containers (see figure 6.2). External to a computer, data can be read from video sources (e.g. consumer cameras) or from hard drives/USB keys. Within the computer, data is stored into hard drive(s) for long-term periods.
For data to be processed by an application (i.e. some code), a transfer from permanent storage areas to temporary storage ones is required. In a standard pipeline, data is first loaded onto the RAM, then moved onto CPU caches, to finally end up into CPU registers where computations actually happen.

CPU caches

Designed to make instantly available data most often used by the CPU, two levels of

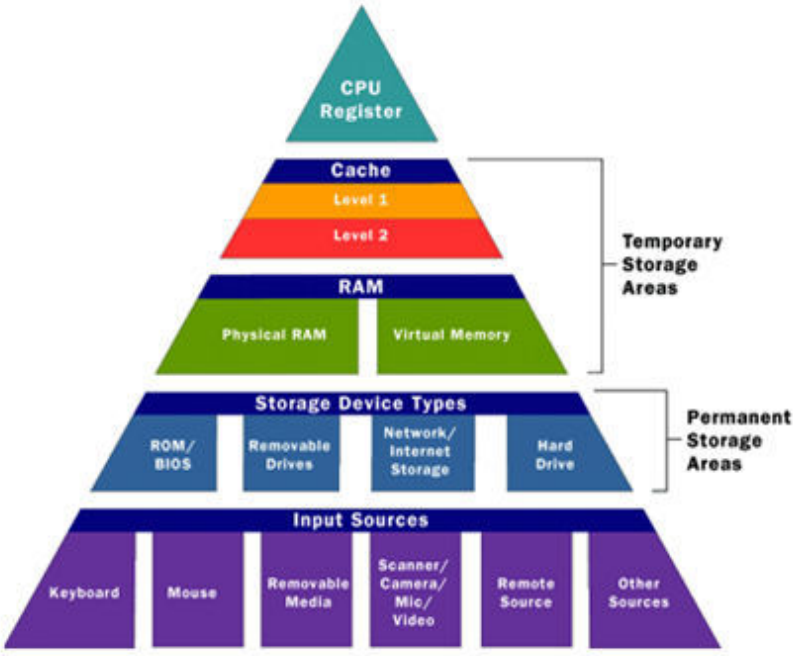


Figure 6.2: Illustration of computer and CPU architectures.

caches are available: level 1 and level 2. Their purpose is to serve as buffer between the global memory (i.e. RAM) and registers, thus reducing the average time to access data from the main memory.
Level 1 cache is a small amount of memory, usually ranging from 2Ko to 64Ko, directly located right onto the CPU. Level 2 cache is memory card located near the CPU, with a little bit more memory, usually ranging from 256Ko to 2Mo.

CPU registers
They are memory cells built onto the CPU and containing the Arithmetic and Logic Unit (ALU). They can contain between 32 bits and 256 bits of data, depending on the generation of CPU.
Data is loaded from a larger memory (i.e. caches) into registers where it is used for arithmetic or logic operations, or other machine instructions. CPU registers being at the top of the memory hierarchy, they provide the fastest way to access data.

6.3.2 Memory access and alignment

Having presented data access pattern within the CPU architecture, it appears crucial to properly store and access data (from the code side) in order to reduce as much as possible data transfer time from global memory to CPU caches (and eventually CPU registers).
In the following, we present how to organize memory access order (section 6.3.2.1),

the effect of having data structures aligned in memory (section 6.3.2.2), and our implementation choices towards this effect (section 6.3.2.3).

6.3.2.1 Memory access order

Following the locality of reference concept, phenomenon describing related storage locations frequently accessed, sequential locality occurs when data elements are arranged and accessed linearly, such as, traversing the elements in a one-dimensional array.
In other terms, data should be accessed in increasing addresses order as CPU caches optimize memory access in increasing sequential order. Data has to be fetched from global memory (i.e. RAM), placed within caches in order to be finally processed by CPU registers. However, fetching data is a time-consuming task, as such with sequential data access, the amount of data fetching is limited, thus providing optimized data access.
Most of the time for image processing algorithms, predicting in which order data will be accessed is easy. As such, data should be organized in memory in such a way that sequential processing is possible.

6.3.2.2 Data structure alignment

Modern computers read from/write to a memory address in word-sized chunks (e.g. 4 byte chunks on 32-bit systems). Data alignment and data structure padding correspond to two separate but related issues regarding the way data is arranged and accessed in computer memory.

Data alignment
Due to the way the CPU handles memory, putting the data at a memory offset equal to some multiple of the word size increases the system's performance; and is called data alignment. However, it may be necessary to insert some meaningless bytes between two data structures to ensure such alignment, which is called data padding.

Data padding
Usually, the compiler allocates individual data items as to respect data alignment, which is not possible for data structures containing members with different alignment requirements (e.g. a float and a char). To maintain proper alignment within a data structure, the translator usually inserts additional unnamed data members (see figure 6.3). In addition, a data structure as a whole may be padded with a final unnamed member to allow each member of an array of structures to be properly aligned.
Automatic padding being only performed when a structure member is followed by a member with a larger alignment requirement, re-ordering members in a structure will impact the amount of padding required to maintain alignment (see figure 6.3).

Using *pack()* instructions represents a lazy solution to tell the compiler to pack the members of a structure to a certain level of alignment (see figure 6.3). For example, using *pack(2)* aligns data members larger than a byte to a two-byte boundary so that any padding members are at most one byte long.

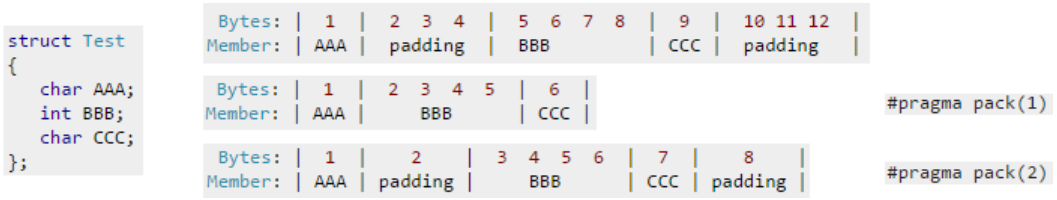


Figure 6.3: Illustration of different data structure paddings.

6.3.2.3 *ShapeDetector* containers re-factoring

In order to reduce data access time during the *ShapeDetector* process, data containers have been memory aligned and re-factored to enable a sequential data access as much as possible.
Standard OpenCV matrix containers have been substituted by Boost libraries containers enabling better control over memory alignment. Using Boost GIL (Generic Image Library) image containers, semantic labelling results (e.g. semantic scores) have been memory aligned on 32 bits .
Regarding the sliding window process, shape-template models are evaluated over each pixel location. However, while semantic labelling maps are scanned in an optimized and sequential order (i.e. double loop over rows then columns), shape-template models are quite large and cover more of the image than what can be cached. As such, multiple data transfers from RAM to caches are necessary to evaluate a model on every pixel location.
To overcome this issue, we propose to store a shape-template model within a structure where each square (from the piece-wise approximation) is represented by four pointers, one for each corner. The sliding window process is not performed over the image, only pointers within the structure are successively incremented, thus providing sequential access order and limiting jumps in memory.

6.3.3 SIMD

SIMD stands for Single Instruction Multiple Data, mimicking parallel computing on CPU registers as illustrated by figure 6.4[1]. Only data level parallelism is exploited but not concurrency: at a given moment the same operation is simultaneously (i.e. parallel) performed on multiple data, but in a single process (i.e. instruction). SIMD is well suited for algorithms requiring simple, repetitive calculations of large amounts of data.

[1] From https://www.kernel.org/pub/linux/kernel/people/geoff/cell/ps3-linux-docs/CellProgrammingTutorial/BasicsOfSIMDProgramming.html

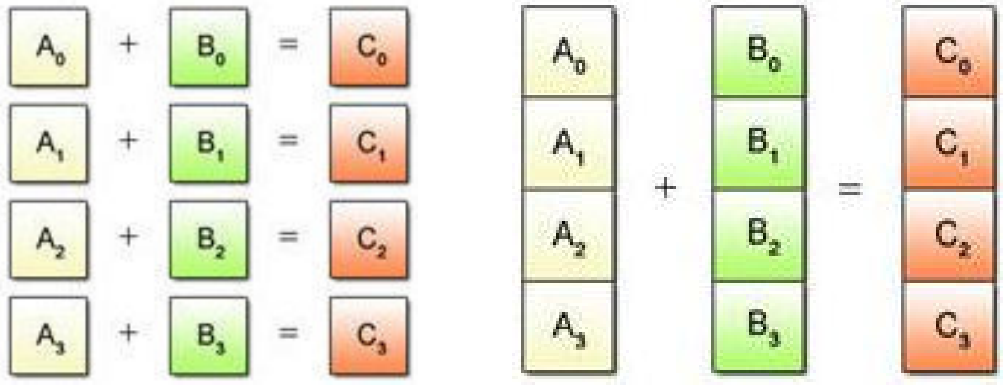


Figure 6.4: Scalar operations (left) versus SIMD operations (right).

6.3.3.1 SIMD overview

SIMD instructions are performed by Arithmetic and Logic Units (ALUs) within CPU registers, hence only arithmetic and logical operations can be executed (e.g. +, -, and, or) over standard scalar types (e.g. char, int, float, double). While multiple data can be processed with a single instruction, it is to note that operations cannot be mixed (as illustrated in figure 6.5).

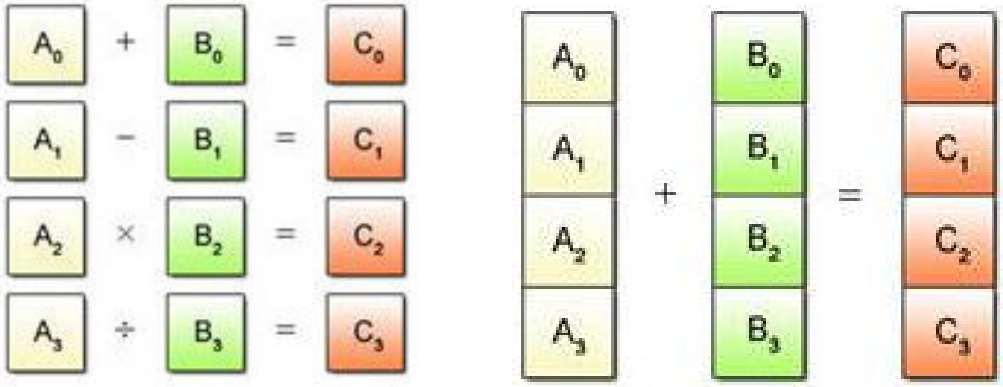


Figure 6.5: SIMD unprocessable pattern (left) and processable pattern (right).

SSE, SSE3 or AVX

CPU register size represents the upper limit of data held, which corresponds to the number of parallel instructions possible to perform. Depending on CPU architecture, register size is varying, but the two most common sizes are: 128 bits and 256 bits.

Multiple generations of SSE (Streaming SIMD Extensions) instructions have been developed to make use of 128-bit size registers. For example, 8 integer operations (16 bits) or 4 floating point operations (32 bits) can be performed simultaneously.

To manipulate 256-bit size registers, Advanced Vector Extensions (AVX) have been

created, adding new instructions to legacy 128-bit SSE ones. For example, 16 integer operations (16 bits) or 8 floating point operations (32 bits) can be performed simultaneously.

Vector types
While conventional data types used in C programming are scalar types (e.g. char, int, float), data types used for SIMD operations are called vector types. A vector can be interpreted as a series of scalars of the corresponding type, automatically aligned on 16-byte boundaries, and of similar length to the type of extensions used (e.g. SSE or AVX).
However, vector fields can not be accessed directly and union types have to be declared with a corresponding scalar container. Figure 6.6 illustrates an SSE vector type (m128) and an AVX vector type (m256) is case of float elements.

```
typedef union {
    m128 m;
    float v[4];
} v4sf; // gives access to __v4sf
```

```
typedef union {
    m256 m;
    float v[8];
} v8sf; // gives access to __v8sf
```

Figure 6.6: SIMD union types for SSE instructions (left) and AVX instructions (right).

6.3.3.2 SIMD implementation

In the second stage of the *ShapeDetector* pipeline, since integral images are employed, shape-template models are evaluated through the computation of a limited number of floating point operations. As such, we propose two SIMD implementations: one for 128-bit registers (i.e. SSE3) and one for 256-bit registers (i.e. AVX). We use an optimized memory access strategy, similar to the one described above, where iterators are created on each four corners of piece-wise shape-template model approximations, then slided along the semantic labelling results. As SIMD containers, we use *v2di* vector type for SSE instructions and *v4di* vector type for AVX instructions, respectively allowing to perform simultaneous operations on 4 float and 8 float values.
Here is the exhaustive list of SSE instructions required: _mm_loadu_si128, _mm_add_epi32, _mm_sub_epi32, _mm_load_ps1, _mm_mul_ps, _add_ps.
Here is the exhaustive list of AVX instructions required: _mm256_cvtepi32_ps, _mm256_add_ps, _mm256_sub_ps, _mm256_set1_ps, _mm256_mul_ps, _mm256_hadd_ps.

6.3.4 Multi-threading

OpenMP, which stands for Open Multi-Processing, is an API supporting multi-platform shared memory multiprocessing programming in C++. It consists of a

set of compiler directives, library rountines and environment variables that influence run-time behavior [2]. OpenMP can be easily used to multi-thread specific code blocks, using by default the maximum number of available cores.
For the *ShapeDetector*, the sliding window procedure is following a double loop, the first one over the different shape-template models (i.e. different scales and orientations) and the second loop corresponds to the full image evaluation of the shape-template model. Using OpenMP, the first loop is multi-threaded over available CPU cores. As such, in parallel, the same input image is being evaluated for different shape-template models.

6.4 GPU implementation

Having described an optimized CPU version of the *ShapeDetector*, we want to make use of the GPU to further increase computational speed. The GPU is especially well-suited to address problems that can be expressed as data-parallel computations with high arithmetic intensity. Because the same program is executed for each data element, over many data elements, the memory access latency can be hidden with calculations instead of big data caches.
We propose to present an overview of GPU architecture in section 6.4.1 and related GPU programming notions in section 6.4.2. Then, relying on CUDA libraries, we introduce design choices of our GPU implementation in section 6.4.3.

6.4.1 GPU architecture

Figure 6.7 illustrates differences in design between CPU and GPU architectures while highlighting GPU capabilities toward compute-intensive tasks[3]. GPU devices devote more transistors (i.e. green elements) to data processing than CPU's, where green elements correspond to ALUs (i.e. CPU registers), mainly versed into data caching and flow control.

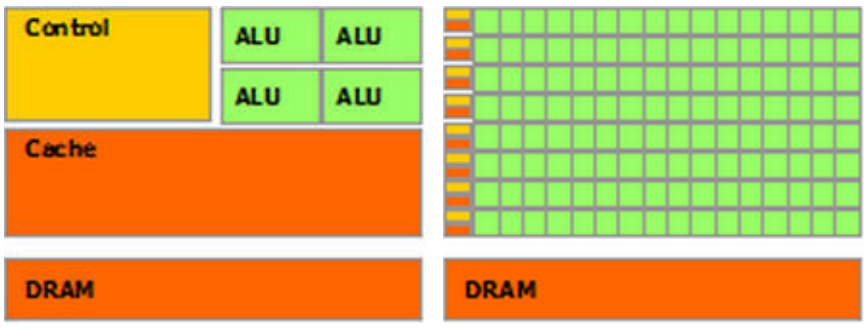


Figure 6.7: Representations of CPU architecture (left) versus GPU architecture (right).

[2] From http://en.wikipedia.org/wiki/OpenMP
[3] From http://docs.nvidia.com/cuda/cuda-c-programming-guide/

As illustrated by figure 6.8, a multi-dimensional grid (from 1D to 3D) is used to organize GPU thread blocks[4]. Since all threads of a block are expected to reside on the same processor core (and share limited memory resources), the number of threads per block is limited to 1024 on current GPUs.

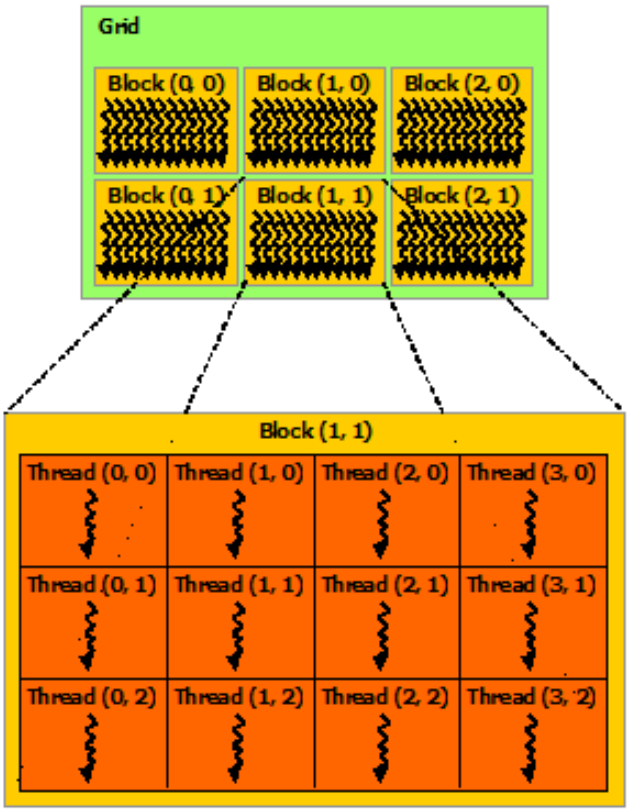


Figure 6.8: Illustration of grid/blocks/threads of a GPU.

6.4.2 GPU programming model

Host* versus *Device

The CUDA programming model assumes that CUDA threads are executed on a physically separate entity name *device* (i.e. the GPU) as a co-processor to the CPU running the main application, named *host*.

The model also assumes that both the *host* and the *device* maintain their own separate memory spaces. Similarly to header (*.hpp) and source files (*.cpp) managing and processing data for the *host*, specific *.cu code files manage memory spaces and process data for the *device*.

Kernels and execution configuration

As opposed to regular C functions executed only once, CUDA allows to define

[4] From http://docs.nvidia.com/cuda/cuda-c-programming-guide/

functions, called *kernels*, that are executed N times in parallel by N different CUDA threads. The number of CUDA threads executing a *kernel* function is specified using the execution configuration syntax inserted between the function name and the parenthesized argument list (represented in equation 6.1).

$$\_\_global\_\_ voidFunction <<< D_g, D_b, N_s, S >>> (parameters); \quad (6.1)$$

D_g specifies the dimension and size of the grid (in 3 dimensions) such as D_g.x * D_g.y * D_g.z equals the number of blocks being launched. D_b specifies the dimension and size of each block (in 3 dimensions) such as D_b.x * D_b.y * D_b.z equals the number of threads per block. N_s specifies the number of bytes in shared memory that is dynamically allocated per block in addition to the statically allocated memory. S is an optional argument specifying the associated CUDA stream.
Execution configuration arguments have to be carefully selected with respect to the GPU device compute capability. Likewise, function calls will fail if D_g or D_b are greater than maximum sizes allowed for the device.

Memory access
During their execution, CUDA threads can access data from the GPU device only via multiple memory spaces such as their local private memory, the thread block shared memory, or the overall memory. The latter memory type encompasses global, constant, and texture memory spaces, which are persistent across kernel launches by the same application. Data, once set as texture on the GPU device, can be accessed at any given time by any thread with almost any cost (time-wise).

6.4.3 CUDA implementation

To perform the detection process on the GPU, we opted to use the following grid and block dimensions:

- Block dimensions: D_b.x = 16, D_b.y = 16, D_b.z = 1.
- Grid dimensions: D_g.x = W/D_b.x, D_g.y = H/D_b.y, D_g.z = O/D_b.z. W and H represents the width and height of the input image. O corresponds to the total number of orientations to process.

The kernel method is separating the image into sub-images of 16×16 pixels, each processed on a different thread at a specific orientation. The complete correlation of the SVM model over one image pixel location to produce a detection score is performed in the same thread.

6.5 Ad-hoc optimization strategies

After CPU code optimization or GPU implementation, there is not much room left for speed improvement while maintaining exactly identical detector performance.

Although waiting for new hardware generations can be a possibility, the remaining solution to further increase computational speed is through the use of ad-hoc optimization strategies, potentially detrimental to detector accuracy.
Below, we report various parameters having an impact on the processing speed depending on the range of values selected by the user. They are applicable to both CPU and GPU implementations of the *ShapeDetector* as they are not directly impacting the computation process itself, but more-so relate to pre-/post-processing choices.

6.5.1 Data down-sampling

The first, and probably most obvious, strategy to speed-up detection is to reduce the size of input data to process. Instead of processing full images, spatial down-sampling by a factor 2 or 4 is performed beforehand. In order not to lose too much information, integral feature channels are shrunk after computing features over the original input image.

6.5.2 Search range

With the *ShapeDetector*, we perform at least multi-scale and multi-orientation surgical tool detection, with a potential round of multi-tool detection to try to perform classification and detection at the same time. For each initial tool model, a complete multi-scale/-orientation is necessary without the possibility to perform multi-threading or parallel computing. As such, using two tool models virtually doubles the processing speed of the detector.
While every space configuration could be tested, the computational time required would be tremendous. Conversely, an overly reduced search space might lead to far too inaccurate results. As such, orientation and scale search spaces have been parameterized as described below.

Orientation search range is defined by a unique parameter being the step between two consecutive orientations to search, obviously in the range $[0°, 360°[$.

Scale search range is defined by three parameters: the highest scale to detect, the lowest scale to detect, and the number of scales to search in-between. Scale is represented by one value, being a ratio when compared to the initially learned model size, assimilated to represent scale 1.

6.5.3 Candidate detections

The last category of user choices relates to the amount of candidate detections allowed. At the end of the sliding window approach, all the candidate detections are first sorted by their confidence score before being transferred to the NMS procedure. Evidently, post-processing a hundred detections is way faster than having to deal with a million ones.

To limit the pool of candidate detections, a first selection is performed by only keeping the ones with a confidence score higher than a specific threshold (i.e. detection score threshold). Detection scores being decimal values, potentially with a restrained order of magnitude differential, we propose to give an upper bound to the amount of candidate detections. A second threshold is then used to specify the number of best candidates to keep and transfer to the NMS procedure.

6.6 Discussion

Performing detection speed-up, either through code optimization and parallel computing or through the use of ad-hoc optimization strategies is becoming common. Obviously the main goal of object detectors is to obtain highly accurate results before being able to run fast. Nevertheless, the frame per second processing capacity is being mentioned side-by-side with the LAMR value in pedestrian detector literature review [Dollár 2011]. Conjointly with the use of integral channels, this indicates the importance of detection speed and a desire made by the community to find new ways for speed improvement.
Processing data in real-time is also crucial for integration into medical applications such as CACAI systems. At the very least, a detection algorithm should be able to process data at the same speed as the recording device to make a full use of available information. Depending on physical hardware constraints, investigating CPU code optimization, GPU code optimization and ad-hoc optimization strategies is necessary. Some medical devices are currently only equipped with a CPU and can not integrate a GPU because of space constraints. Waiting for high-end GPUs to enter the OR might take some time, thus investing other speed-up solutions is unavoidable. In addition to space constraints, GPUs require high power-supply and multiple fans to reduce the heat from an intense use, which might be an issue within sterile environment where dust can not be expelled anywhere.
Considering CPU code optimization through SIMD instructions or GPU implementation might be extreme, yet taking advantage of C++ programming advice and OpenMP libraries is a good start to notice speed differences. Without thinking about applications where high speed is mandatory, small speed increase enable the acquisition of more experimental results in a same amount of time. In our line of work, the sooner detection results are acquired and analyzed, the sooner the method can be modified and improved. Having to wait fifteen minutes for a new batch of detection results is not long-term time-efficient when the same results could be generated in five minutes with the use of OpenMP.
Of course, CPU code optimization and GPU implementation simply enable to obtain the same detection results in less time. When dealing with ad-hoc optimization strategies, the impact on detection accuracy should be assessed in order to find the best position along the speed versus accuracy trade-off.

Chapter 7

Surgical tool detection: Results and discussion

Contents

7.1 Introduction

Having presented core elements of this thesis work in previous chapters, as long as validation studies, in this chapter we focus on tool detector performance validation and comparison. First, we investigate *ShapeDetector* parameters impact on detector performance in section 7.2. Then in section 7.3, we report overall detection results for all the considered baseline methods (as introduced in section 5.4). In section 7.4, we study the processing speed for both the *adapted SquaresChnFtrs* and the *ShapeDetector*. Finally, in section 7.5 we provide discussions and future work directions.

7.2 *ShapeDetector* tuning process

As presented in section 4.3, the *ShapeDetector* framework is based upon a two-stage pipeline. Each stage using its own learning approach and specific set of parameters, investigating their impact on *ShapeDetector* performance is of interest. Independently optimizing parameters of each stage eventually leads to better detection results.
In section 7.2.1, we take a closer look to the pixel-wise classification process and focus on obtaining the best intermediate semantic labelling results. Then in section 7.2.2, we investigate the design space for the shape-template model creation.

7.2.1 Semantic labelling

Computed with the *segmentation quality* validation metric, we study the impact of the cascade classifier parameters on the semantic labelling quality (in section 7.2.1.1). Then, in section 7.2.1.2, we compare results with the considered semantic labelling baseline. Finally, in section 7.2.1.3, we provide images showcasing success and failure modes of the semantic labelling.

7.2.1.1 *ShapeDetector* semantic labelling: parameters tuning

Only a handful of parameters can have an impact on the semantic labelling quality (for the method introduced in section 4.3.2). Those parameters can be grouped

into two categories: related to the feature representation or to the cascade classifier learning. Table 7.1 reports the impact of different feature representation on the labelling accuracy. As expected, color and texture are strong cues, while position is a rather weak one.

Table 7.1: Feature representation impact on the labelling accuracy.

Feature channels	Accuracy
HOG alone	78.9%
HOG+LUV	84.9%
HOG+LUV+XY	85.1%
HOG+LUV+XY+FB	85.2%
HOG+CN+XY+FB	**85.7%**

For the HOG+CN+XY+FB feature channels combination, table 7.2 reports the impact of different cascade classifier learning configurations on the labelling accuracy. It indicates that larger model window size or increased number of weak classifiers has very little to no effect on the semantic labelling accuracy. The decision tree depth parameter is not studied as it can not be modified.

Table 7.2: Cascade classifier learning configuration impact on semantic labelling accuracy.

Model window size	# Weak classifiers	Accuracy
200	51×51	85.6%
500	31×31	85.4%
500	41×41	85.8%
500	$\mathbf{51 \times 51}$	**85.7%**
500	61×61	84.7%
750	51×51	85.4%
1000	51×51	85.4%

In the remainder of the chapter, all subsequent experiments using the *SquaresChnFtrs* semantic labelling are performed using 500 depth-2 decision trees, a 51×51 model window size, and HOG+CN+XY+FB as feature channels representation.

7.2.1.2 Semantic labelling performance comparison

As presented above, the *SquaresChnFtrs* semantic labelling is obtaining an accuracy of 85.7% with the selected parameter configuration. As a comparison, the semantic labelling baseline considered: Darwin, is obtaining a lesser accuracy with 73.4% only. Figure 7.1 illustrates semantic labels map obtained from both techniques over the same image. Tool regions are visibly better labelled with the *SquaresChnFtrs*

approach than with the Darwin one. The Darwin approach is under-labelling with many tool pixels missed, while by comparison the *SquaresChnFtrs* is slightly over-labelling.

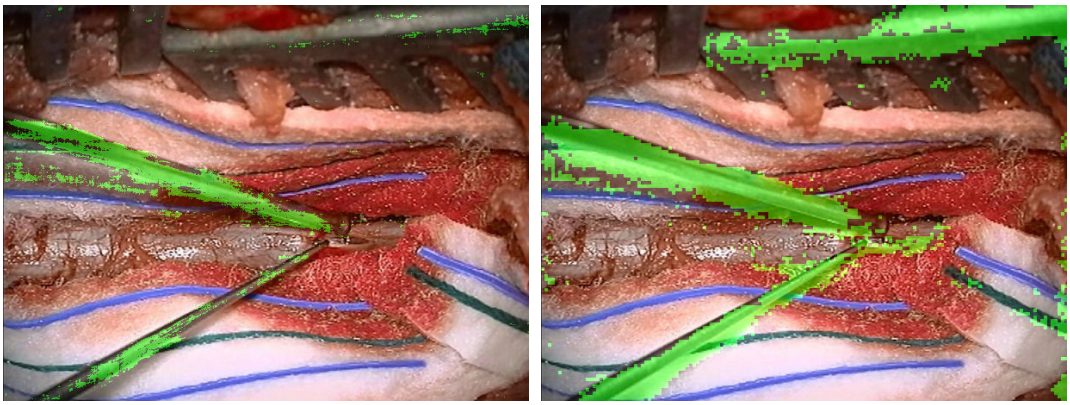


Figure 7.1: Semantic labels map obtained from Darwin (left) and the *SquaresChnFtrs* framework (right). Detected tool pixels are marked green.

7.2.1.3 Success and failure modes

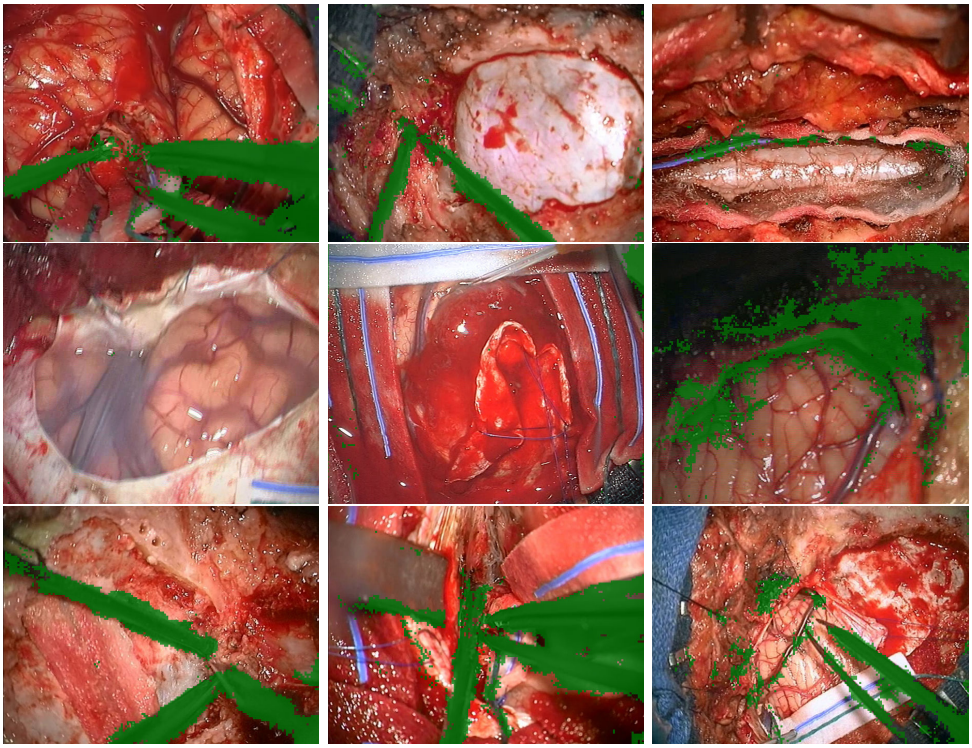


Figure 7.2: Example semantic labelling results obtained from the *SquaresChnFtrs* method (`HOG` + `CN` + `XY` + `FB` configuration). Detected tool pixels are marked green. The rightmost column shows some failure modes.

Figure 7.2 illustrates success and failure modes obtained with the *SquaresChnFtrs* semantic labelling. It can be noticed that either in success mode images or in failure ones, the semantic labelling is quite noisy. While most of surgical tools are marked in green, their boundaries are not well defined and tips are sometimes not completely covered.

7.2.2 SVM shape-template model creation

Creating an accurate surgical tool model is crucial for high detector performances. For the SVM model creation, introduced in section 4.3.3.2, many different training sample configurations as long as various design choices can be selected. Their impact on *ShapeDetector* performances are quantified using the *polygon overlap* validation metric.
First, SVM training sample impact is reported in section 7.2.2.1 for positive samples, and in section 7.2.2.2 for negative ones. Then, SVM design choice impact is reported in section 7.2.2.3 for the regularization term and in section 7.2.2.4 for other SVM parameters. Finally, a conclusion on the SVM shape-template model creation is provided section 7.2.2.5.

7.2.2.1 Positive training samples

Three strategies have been considered to generate positive samples for the SVM training (see section 4.3.3.2): (1) *SquaresChnFtrs* semantic labelling scores, (2) annotations of all surgical instruments, (3) annotations of a single surgical instrument. In figure 7.3, we report *ShapeDetector* performance for those three alternatives. Using single instrument annotations (red curve) leads to the best detection performances.

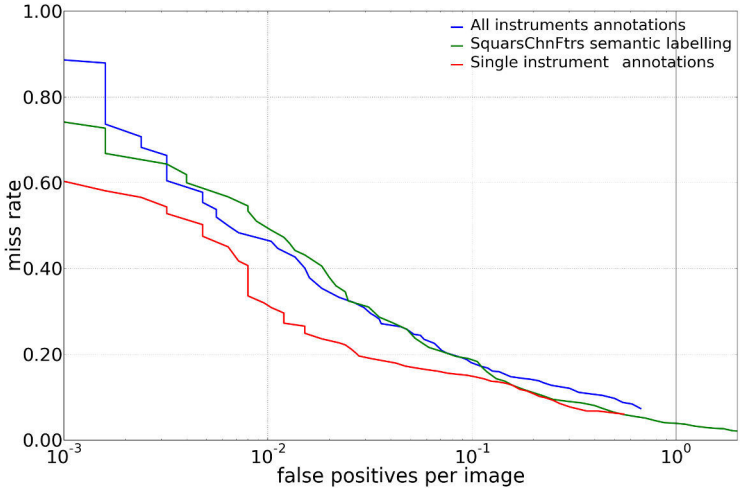
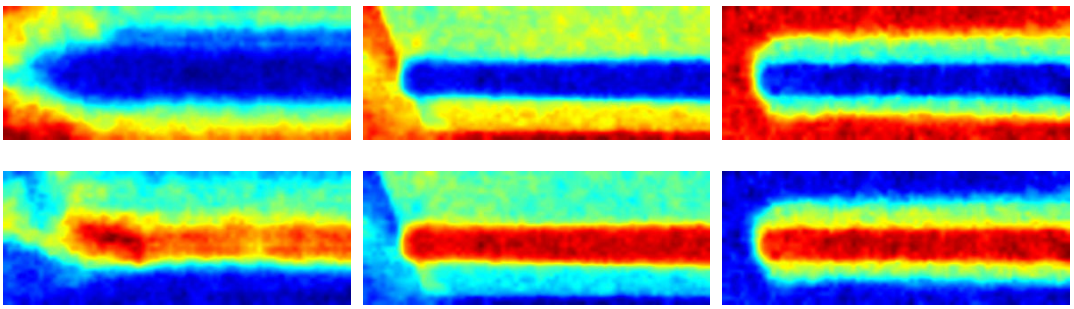
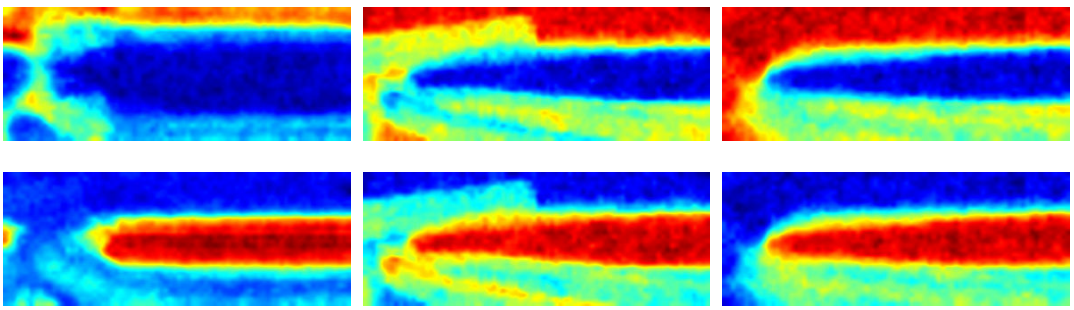


Figure 7.3: Detection results for each type of positive samples used to learn the SVM model. Obtained with the *polygon overlap* metric for a suction tube.

Figure 7.4 illustrates visual SVM model differences when learned with the three type of positive samples, for a suction tube instrument and the upper part of a bipolar forceps. As can be expected, the 'cleanest' model exhibiting best the tool shape is obtained from the third alternative. The tool shape obtained with the first alternative is extremely noisy and the tool is hardly recognizable. Indeed, in addition to semantic labelling results being quite noisy themselves, in the data-set the suction tube is sometimes partially occluded by other surgical instruments or being very close to them, explaining the not-so-well-defined shape.



(d) Semantic labelling data (e) All instruments annotations (f) Single instrument annotations

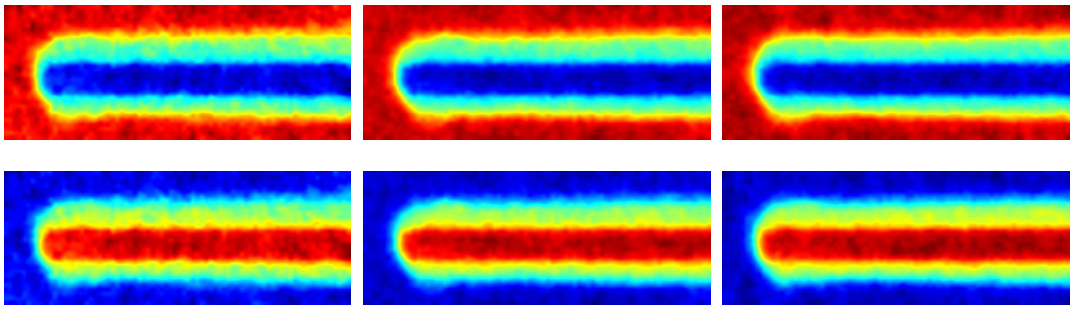


(j) Semantic labelling data (k) All instruments annotations (l) Single instrument annotations

Figure 7.4: SVM model per type of positive samples for a suction tube (top) and a bipolar forceps (bottom). Upper rows represent the background class and bottom rows the tool class.

7.2.2.2 Negative training samples

Three negative samples alternatives have been considered to train the SVM shape model (see section 4.3.3.2): (1) binary sampling, (2) grey-scale sampling, (3) Gaussian sampling.
Figure 7.5 displays the visual impact of SVM model alternatives. Variations in appearance are minimal across models, indicating a small impact of the negative sampling strategy on detection results.



(d) Binary sampling (e) Uniform sampling (f) Gaussian sampling

Figure 7.5: Suction tube SVM model per type of negative sampling strategy. Upper row represents the background class and bottom row the tool class.

Figure 7.6 reports *ShapeDetector* detection performance for those three alternatives. We can see that the same results are obtained without any noticeable different between negative sampling strategies.

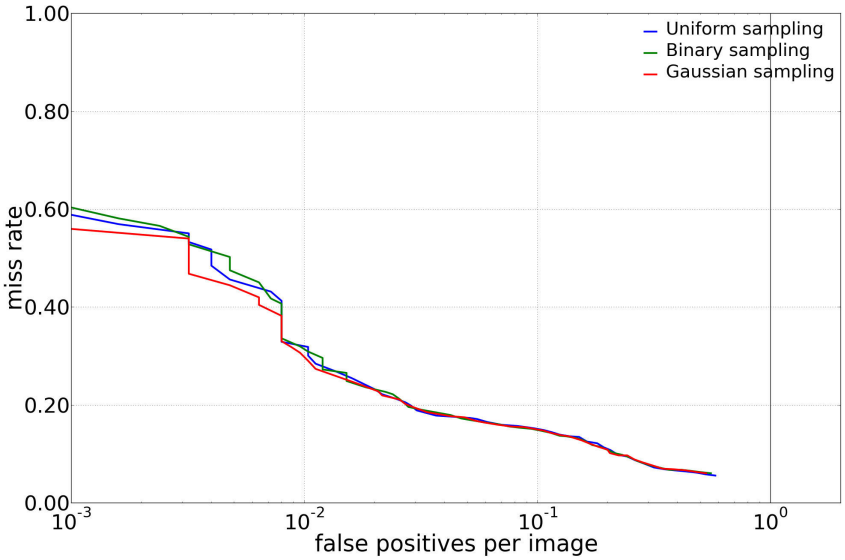


Figure 7.6: Detection results for each type of negative samples used to learn the SVM model. Obtained with the *polygon overlap* metric for a suction tube.

In table 7.3, we report the mean and variance parameters of the Gaussian used to perform the sampling with alternative (3). Gaussian distributions are very similar when using either single annotation (alternative (1)) or full annotations (alternative (2)) positive sample creation strategy, indicating a limited presence of other tools in the vicinity of one tool. Very different parameters are obtained from real semantic labelling positive samples and huge discrepancies can be noted when compared to a perfect semantic labelling represented by the full annotation positive sample creation strategy, corroborating the presence of noise.

Table 7.3: Gaussian parameters for different tools and positive image types: Single Annotation (SA), Full Annotation (FA) and *SquaresChnFtrs* labelling (Real). * indicates only the upper part of the tool is used.

	Suction tube			Bipolar Forceps			
	SA	FA	Real	SA*	SA	FA	Real
Mean (Tool class)	166	158	26	181	150	142	33
Var. (Tool class)	55	57	5	51	58	60	8
Mean (BG class)	89	97	41	74	105	113	55
Var. (BG class)	55	57	10	51	58	60	16

7.2.2.3 SVM regularization

We propose to investigate the impact of the SVM 2D spatial regularization term in two ways. First, its visual impact on shape-template models, and then its performance impact on *ShapeDetector* results.

Visual impact

Figure 7.7 displays side-by-side models obtained with and without SVM regularization for a suction tube and a bipolar forceps. Models learned with the use of the regularization term are noticeably smoother, indicating a proper behavior for the spatial regularization.

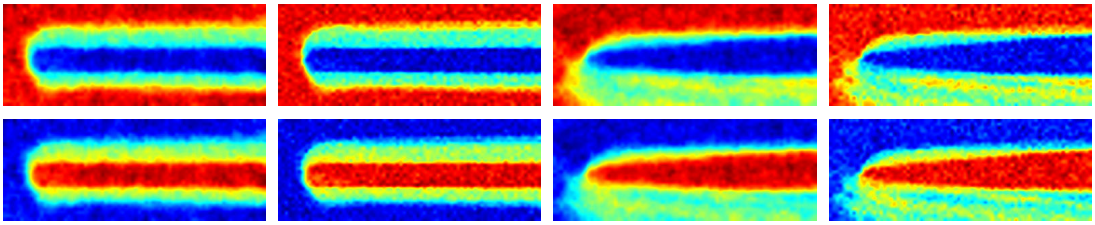


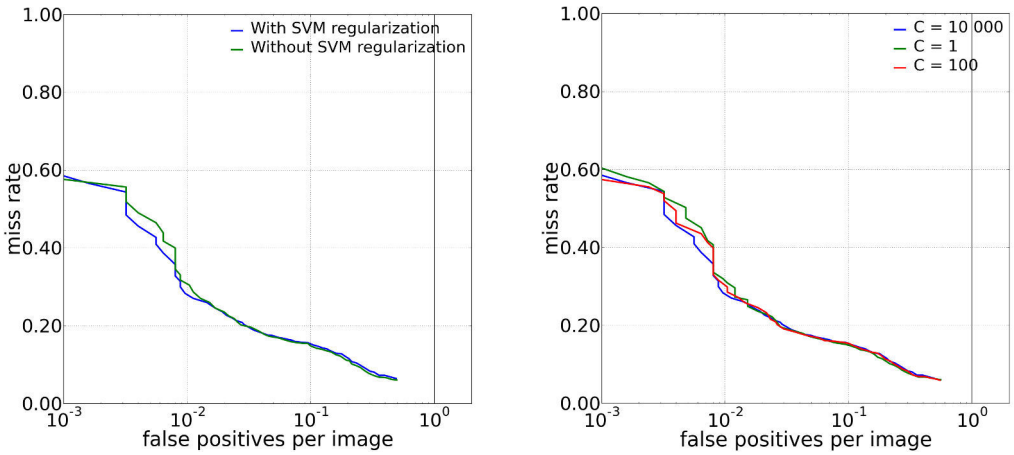
Figure 7.7: Visual impact of the SVM spatial regularization term. Odd columns display regularized SVM models and even columns unregularized ones. Upper row represents the background class and bottom row the tool class.

Impact on *ShapeDetector* performance

Figure 7.8a reports *ShapeDetector* performance obtained with two suction models, one with spatial regularization and one without. The SVM spatial smoothness prior is not improving the overall quality as results are almost identical.

7.2.2.4 SVM parameters

The last parameter studied is the value of the internal SVM regularization parameter C. As shown in figure 7.8b, varying this parameter has no impact on the detector performances.



(a) Impact of the SVM regularization.

(b) Impact of the SVM C parameter.

Figure 7.8: Impact of the SVM regularization (left) and the SVM C parameter (right) on *ShapeDetector* performance for the suction tube.

7.2.2.5 Conclusion

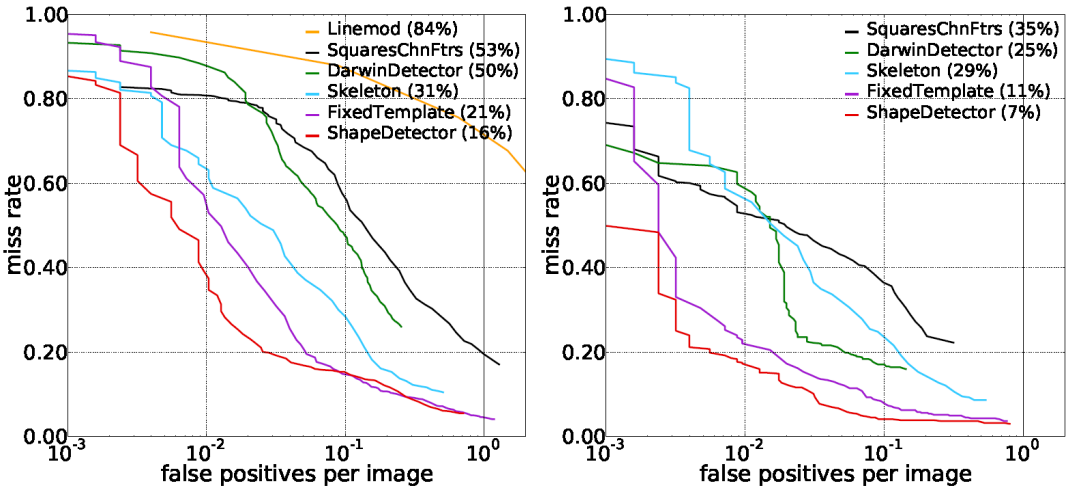
From those experiments, we identified which training samples and SVM design space have to be used to obtain the best detection results possible. In the remainder of the chapter, results are reported with SVM models learned using the following configuration: an SVM C value of 1, the spatial regularization term, a binary distribution for sampling negative examples, and single instruments annotations as positive examples. Shape template models are trained at a fixed size of 125×300 pixels. An exhaustive side-by-side display of SVM models over different design space configurations is available appendix C.

7.3 Detectors performance

Having presented how to optimize both stages of our *ShapeDetector*, we propose in this section to report and compare detectors performance for the two most common tools of the dataset: the suction tube and the bipolar forceps. To ensure a fair comparison, we match the parameters of each method as closely as possible in terms of training data, evaluated scales and orientations.
Considering the pose of an instrument to be represented by three parameters, a detailed performance validation for each is proposed. First, for the global position in section 7.3.1, then the tip location in section 7.3.2, and lastly the orientation in section 7.3.3.In addition, success and failure modes are illustrated in sections 7.3.4 and 7.3.6. To complete the comparison between considerd detectors, we propose in section 7.3.5 a side-by-side visual illustration of their results on a sub-set of images. At test time, detectors were evaluated using a 4-pixel stride in line and column

without image down-sampling, and a 5° orientation step (i.e. 72 orientations are evaluated). Only one tool shape model has been ran at a time.

7.3.1 Global position performance



(a) Suction tube (b) Bipolar forceps

Figure 7.9: Comparative detectors performance according to the *polygon overlap* metric. LAMR are reported in brackets.

For reference, global position performance results are obtained using the *polygon overlap* validation metric, as described in section 5.3.1. Figure 7.9 illustrates results where large differences in detection quality amongst the considered methods are noticeable. *Linemod* performs quite poorly in this domain, showing that using an off-the-shelf detector is not enough. Our *adapted SquaresChnFtrs* performs significantly better, most likely due to its more flexible model. Still, generic detectors (i.e. one-stage approaches) achieve a rather poor performance, reaching less than 50% recall at 10^{-1} false positive per image (for the suction tube). On the other hand, the hand-crafted *Skeleton* approach provides better results, indicating that pixel-wise segmentation is a strong cue. Finally, our *ShapeDetector* obtains the best results thanks to its data driven learning instead of hand-crafting features or shape cues. Using this metric, at 10^{-1} false positive per image, the miss-rate is reduced by a third with respect to the best generic detector. Which shows the utility of the proposed two-stage approach. The poor results of *DarwinDetector* compared to *FixedTemplate* indicates that high quality semantic labels is key for good detection.

Detector stability

Previous results are obtained using a 25% area threshold within the *polygon overlap* validation metric. As such, we propose to highlight the impact of selecting different overlap threshold, as illustrated in figure 7.10. The results obtained at the selected

value of 25% are similar to the ones obtained at the more standard 50% threshold, showing good performances stability. Our *ShapeDetector* approach obtains low log-average miss-rate for a large range of overlap thresholds and outperforms any of the other approaches.

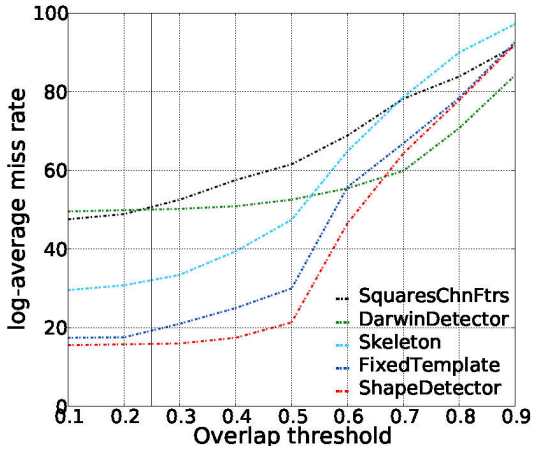
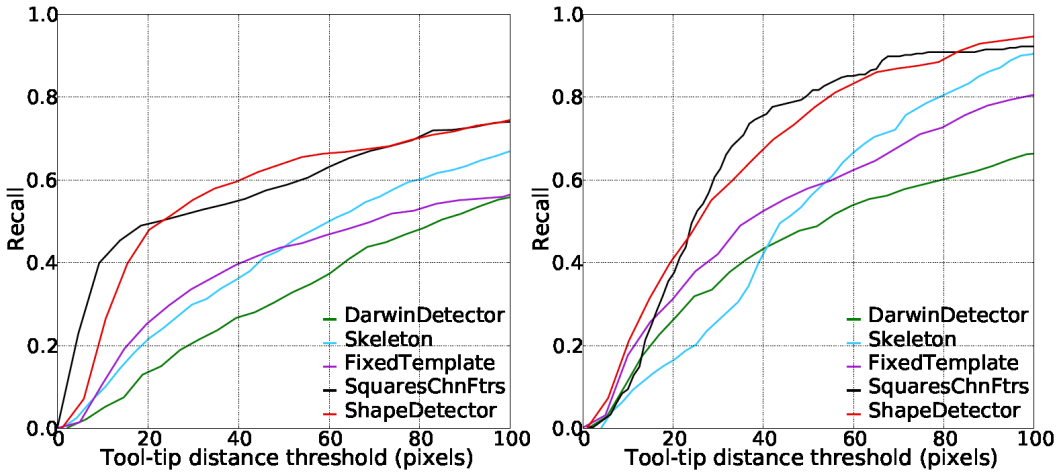


Figure 7.10: Log-Average Miss Rate as a function of the overlap threshold for the considered detectors, for a suction tube.

7.3.2 Tool-tip distance performance

The *Linemod* detector being extremely under-performing, the miss rate at 10^{-1} false positive per image mark is around 90%, which is far too low to perform an accurate validation over the remaining candidate detections. As such, for this study using the *tool-tip distance* validation metric, the *Linemod* detector is not considered. Results are reported in figure 7.11.



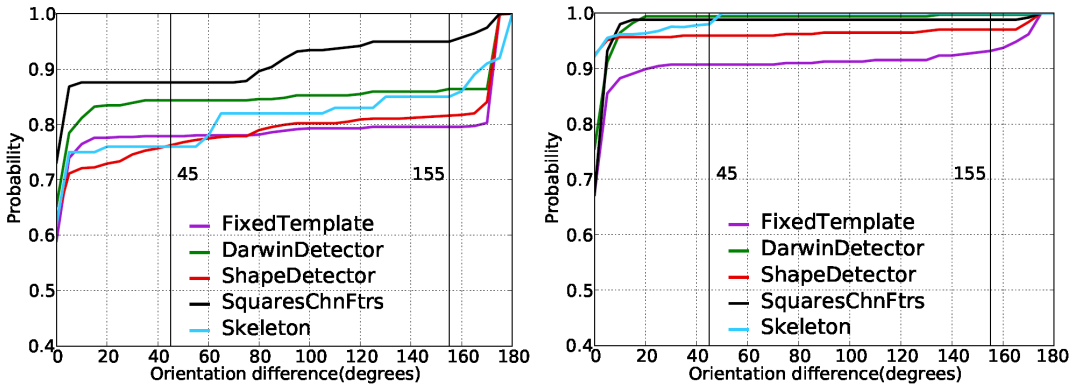
(a) Suction tube (b) Bipolar forceps

Figure 7.11: Comparative detectors performance according to the *tool-tip distance* metric, at the 10^{-1} FPPI mark.

Both our *adapted SquaresChnFtrs* and *ShapeDetector* have similar performances under this metric, with less than a 20-pixel error for a 50% recall for the suction tube. Between *FixedTemplate* and *DarwinDetector*, the 10% recall difference for a 40-pixel suction tube tip error indicates the impact of the semantic labelling quality around tool boundaries. A 20% recall improvement at a 20-pixel suction tube tip error can be noted between the *FixedTemplate* and the *ShapeDetector*, pointing out the benefits from sophisticated shape modelling towards the tool-tip estimation. With our proposed *ShapeDetector*, the bipolar forceps tip position is overall better estimated than for the suction tube. Aside from the tool tip occlusion issue, semantic labelling noise appears to be less influential for tools with a large enough tip region, the bipolar forceps being bigger than the suction tube at a similar microscope zoom value.

7.3.3 Orientation estimation performance

For the same reason as stated above, this study using the *orientation difference* validation metric does not consider the *Linemod* detector. Results are reported in figure 7.12.



(a) Suction tube

(b) Bipolar forceps

Figure 7.12: Comparative detectors performance according to the *orientation difference* metric, at the 10^{-1} FPPI mark.

All the compared approaches exhibit a similar behaviour regarding orientation accuracy for suction tube detections. The best orientation estimation (i.e. less than a 5° difference) is achieved for more than 60% of detections across the methods, given models being tested with a 5° orientation step. Curves are steady in-between a 45° difference and a 155° difference, indicating only few detections are completely erroneous orientation-wise. For the *ShapeDetector*, roughly 20% of detections have an orientation deviating by 170°-180° from the reference, indicating a well placed detection regarding its global position, only facing the opposite direction. Noisy semantic labelling results around tool-tip locations, region heavily focused by the shape model learning strategy, as long as occlusions can induce such a shift in orien-

tation. Regarding the bipolar forceps surgical instrument (illustrated figure 7.12b), such a confusion is orientation is far less important, happening only for 5% of candidate detections with the *ShapeDetector*. Bipolar forceps overall shape is diverse enough to compensate for tip labelling noise and provide a robust orientation estimation

7.3.4 Success modes

Figures 7.13 and 7.14 display success modes obtained with the *ShapeDetector* for a suction tube and a bipolar forceps model.

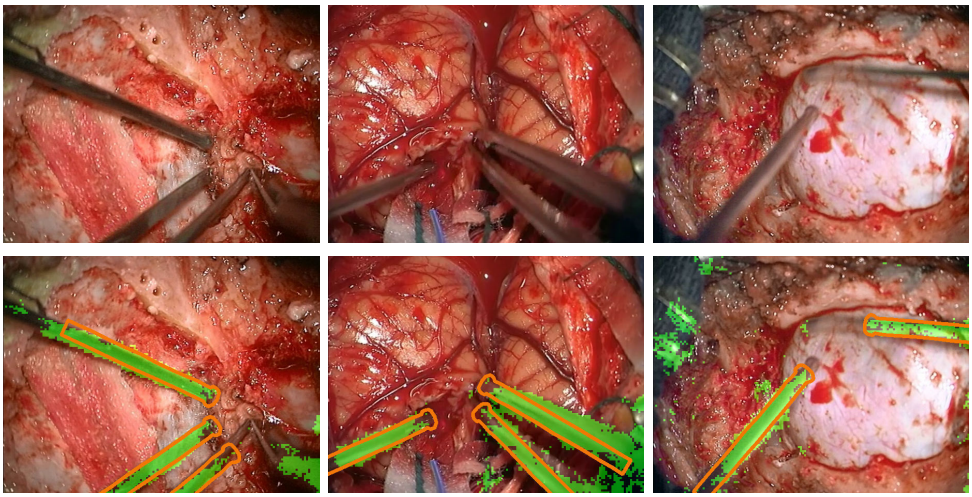


Figure 7.13: Detections using our *ShapeDetector* with a suction tube model. First row: original images; second row: our detections (semantic labelling labels overlaid in green).

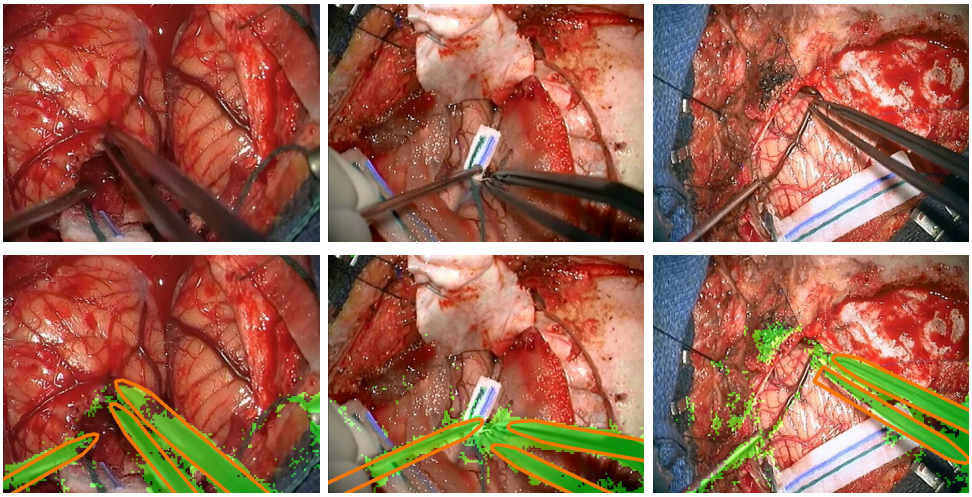


Figure 7.14: Detections using our *ShapeDetector* with a bipolar forceps model. First row: original images; second row: our detections (semantic labelling labels overlaid in green).

7.3.5 Comparative visual results

Figure 7.15 offers a side-by-side comparison of detections obtained with the various methods considered across a sub-set of images.

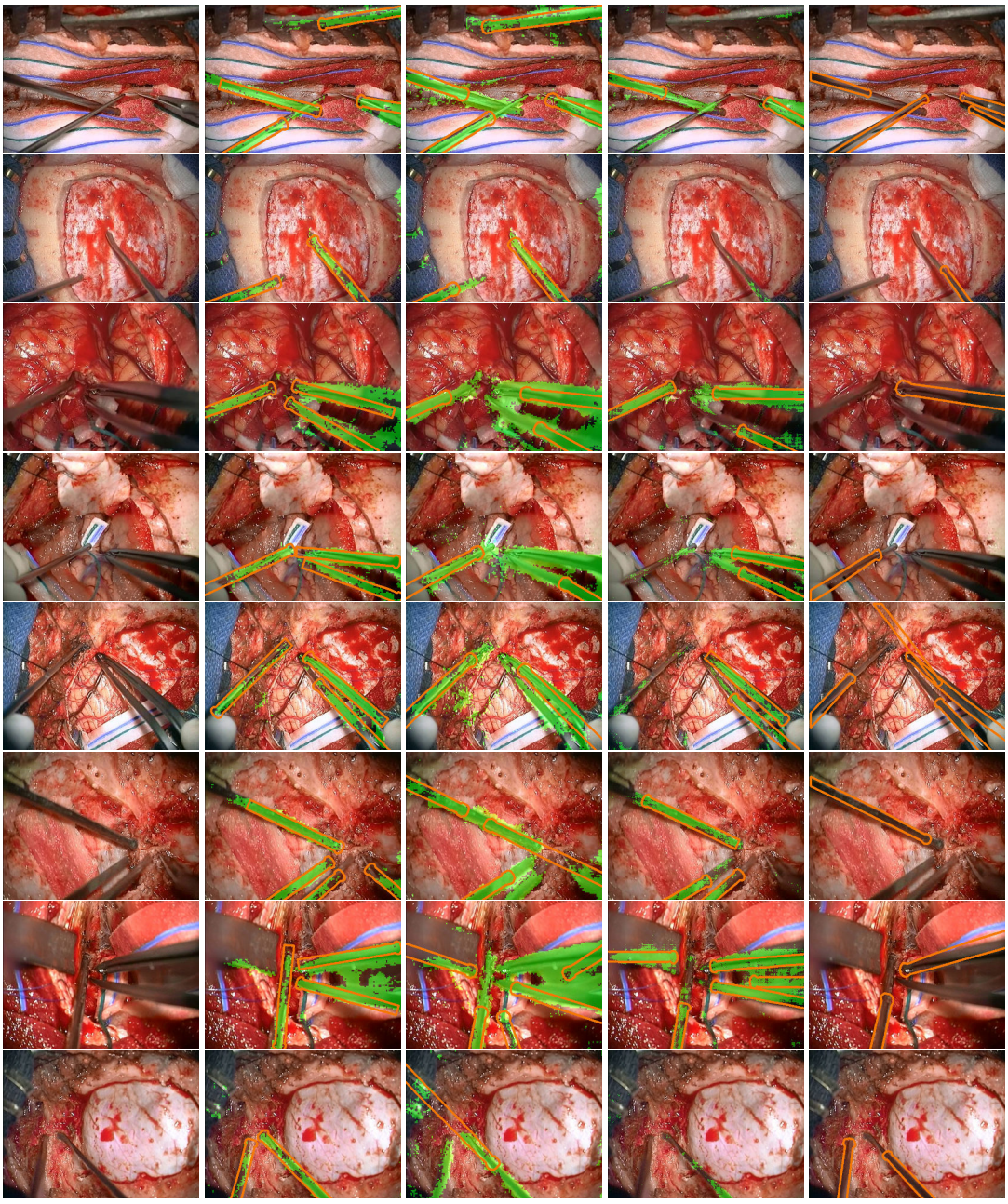


Figure 7.15: Detection examples using a suction tube model (with semantic labelling results overlaid in green when used).
From left to right: original image, *ShapeDetector*, *Skeleton*, *DarwinDetector*, *Adapted SquaresChnFtrs*.

7.3.6 Failure modes

Figure 7.16 illustrates failure modes obtained with a suction tube model for the *adapted SquaresChnFtrs* detector. Detections being performed on a square window from the surgical instrument model (see green boxes), the polygon model (see pink polygons) is fitting well-enough the surgical instrument within. However, surgical instruments being elongated shapes, the instrument model only covers a limited part of the visible instrument. As such, the polygon model can fit well within the model square but completely drift off over the remainder of the shape.

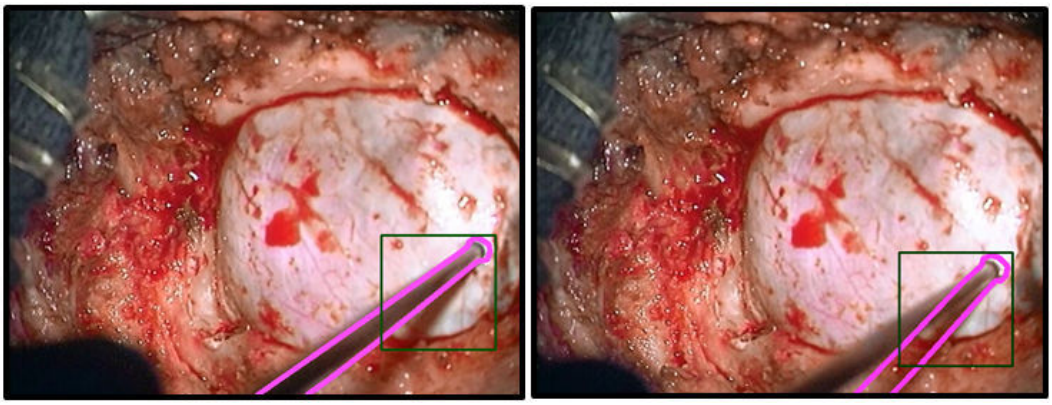


Figure 7.16: *Adapted SquaresChnFtrs* failure modes: polygon drift.

Another category of failure modes for the *ShapeDetector* approach are reported in figure 7.17. The detection polygon is well over the surgical instrument, however the orientation is reversed by 180° because either the tip is occluded or the semantic labelling is too noisy around the tip.

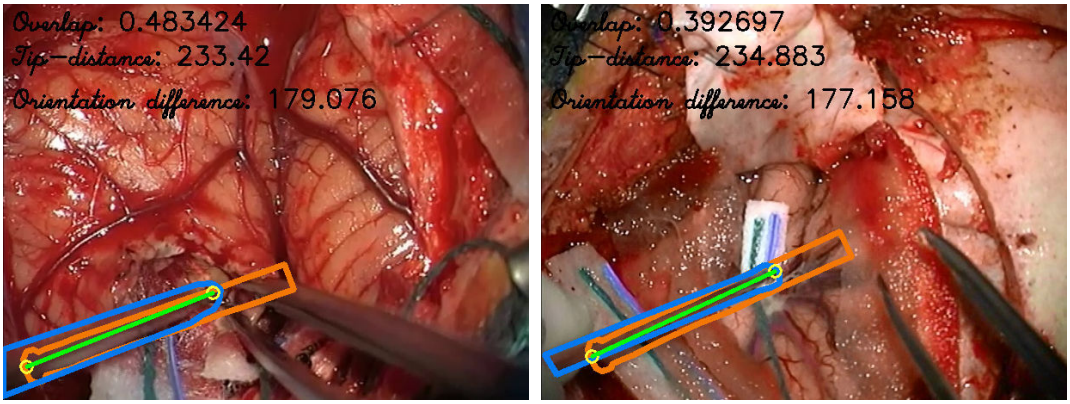


Figure 7.17: *ShapeDetector* failure modes: orientation inversion.

Figure 7.18 illustrates cases of missed detections with a suction tube model for our *ShapeDetector* approach.

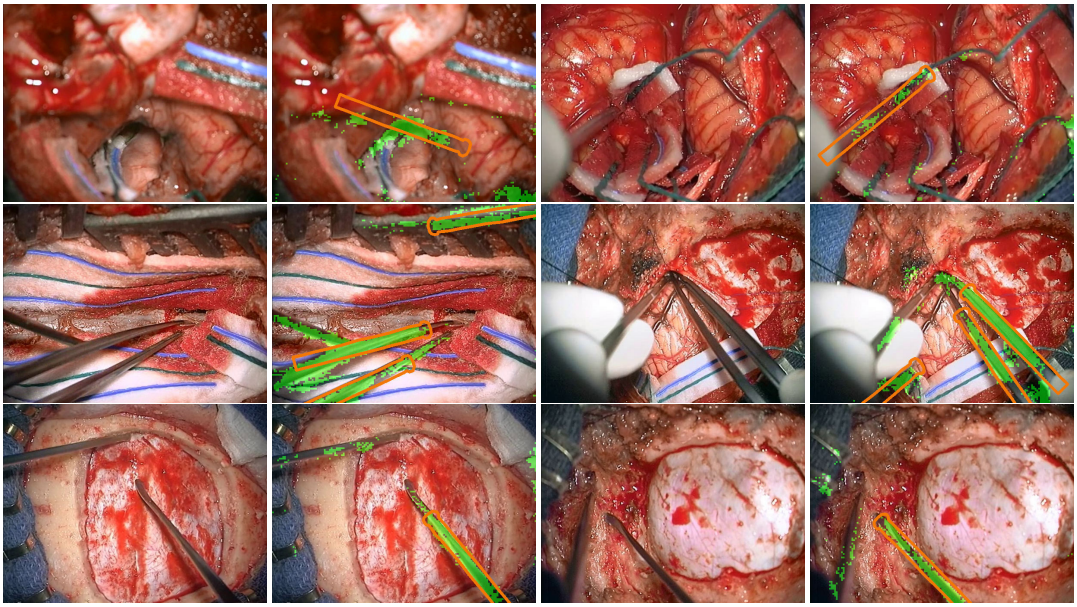


Figure 7.18: Failure cases using the *ShapeDetector* approach with a suction tube model. Odd columns show original images, even columns show detection results.

7.4 Detectors speed

For real-time use in medical applications, the computational speed of surgical tool detectors must be studied. In chapter 6, we presented two types of strategies to decrease computational time: using higher-level implementation techniques or using ad-hoc optimization strategies. While the former only affects detector speed, the latter also usually worsen detector performance.
Below, hardware specifications of the computer used to run the experiments are given in section 7.4.1. In section 7.4.2, speed improvements from CPU and GPU code implementation techniques are reported. Finally, in section 7.4.3, ad-hoc optimization strategy impact on both the computational speed and detector performance are studied.

7.4.1 Computer setup

Results were obtained using the following computer setup:
Model: DELL Precision T8600.
CPU: Intel Xeon E5-2620 v2 @2.10GHz.
GPU: NVIDIA GeForce Titan Black. 2880 cuda cores, 880 MHz Clock.

7.4.2 Speed gain from code design

The original *SquaresChnFtrs* pedestrian detector being already implemented on GPU, our *adapted SquaresChnFtrs* tool detector inherently benefits from its speed-up. As such, we investigate impacts of the classifier early stopping scheme (i.e. SoftCascade) on detector speed and performances (see section 7.4.2.1).
Regarding our proposed *ShapeDetector*, we report in section 7.4.2.2 speed-up values coming from the CPU code optimization and GPU parallel computing.

7.4.2.1 *Adapted SquaresChnFtrs*

Using a SoftCascade as early stopping scheme has been proposed to further speed-up the *SquaresChnFtrs* detector, as introduced in section 4.2.2.4. We report its impact on our *adapted SquaresChnFtrs* for two different processing configurations, as shown in table 7.4). The last column of the table summarizes performance with the LAMR value.
For a similar processing configuration, the speed-up is roughly tenfold when using the SoftCascade. As a counterpart, a small drop in detection performances can be noticed, with a ΔLAMR of 4 when processing 7 scales and 71 orientations.

Table 7.4: *Adapted SquaresChnFtrs* computational speed for different processing configurations. The LAMR value is used to report detector performance.

Scales	Orientation	Soft Cascade	Speed (Hz)	LAMR
1	71	No	3	/
1	71	Yes	28	60%
7	71	No	0.3	49%
7	71	Yes	3	53%

7.4.2.2 *ShapeDetector*

In table 7.5, we report processing times obtained with different *ShapeDetector* implementations. The semantic labelling step was already performed on the GPU since using the *SquaresChnFtrs* framework, and as such CPU speeds can not be reported in the table. From a non-optimized CPU implementation requiring 1.3 s per image, we managed to reach a computation time of only 180 ms with a non-optimized GPU implementation.

Table 7.5: *ShapeDetector* computational times with different CPU and GPU implementations, for one 612×460 image.

	Stage 1 (ms)	Stage 2 (ms)	*ShapeDetector* (ms)
CPU non-opti.	/	1200	1290
CPU opti.	/	600	690
CPU SIMD	/	200	290
GPU	90	90	180

7.4.3 Speed gain from ad-hoc optimization strategies

Ad-hoc optimization strategies can be used to further increase a detector's speed but at the cost of its performance. Below, results are reported for each category of ad-hoc strategies presented in section 6.5, when using a suction tube SVM model. The terminology used in the tables is the following: *stride* represents the spatial step used in row and column, # *Ori* represents the number of processed orientations, *Score thresh.* represents the minimum detection score, and # *Cand.* represents the maximum number of candidates sent to the NMS procedure. Computational speed in Hertz and LAMR values ares used to report speed and accuracy results for the detector.

7.4.3.1 Data down-sampling

Table 7.6 reports speed and accuracy results when performing different shrinking operations on input integral channels. We can notice a six time speed-up between the processing of the input image at the original scale and after down-sampling by a factor 4. The overall accuracy remaining unchanged with a similar LAMR value. As illustrated figure 7.19, some detections are however better estimated regarding the tool-tip position.

Table 7.6: Speed versus accuracy performance with different stride configurations.

	Stride	# Ori	Score thresh.	# Cand.	Speed (Hz)	LAMR
Config. 1	1	72	0.01	All	0.7	16%
Config. 2	2	72	0.01	All	2.5	16%
Config. 3	4	72	0.01	All	4.6	16%

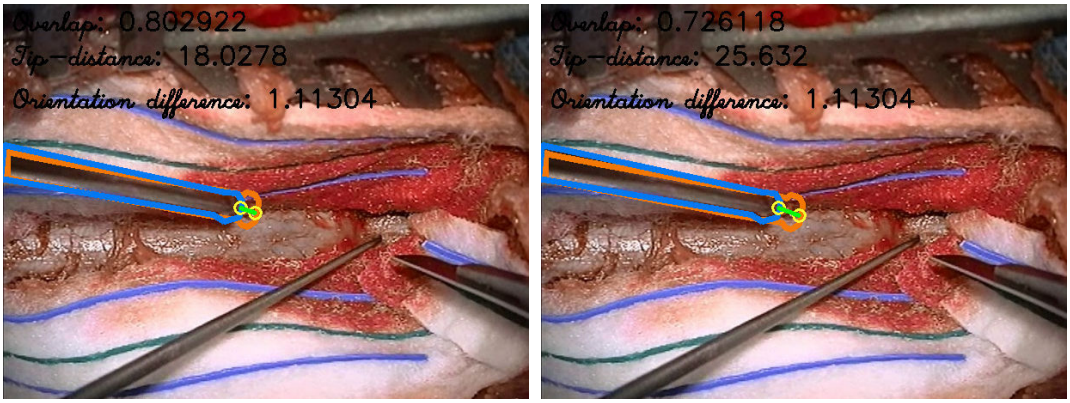


Figure 7.19: Detection results when processing input-size image (left) and a 4-time down-sampled image (right).

7.4.3.2 Search range

For the scale search range, the possibility exists to manually or automatically set the parameters when accessing the surgical microscope zoom value. As such, only the orientation search range impact is explored in table 7.7, where speed and accuracy values are reported. We additionally mention in brackets the orientation step used for an easier understanding of the table (in the *# Ori* column). An almost doubled speed is obtained when processing orientations every 5° compared to every 2°, for a 1% Δ LAMR decrease. For a triple speed-up, between processing orientations every 15° compared to every 2°, results are more impacted with a 7% Δ LAMR deterioration. Figure 7.20 illustrates detections obtained when using different orientation steps. The orientation is better estimated when using a smaller orientation step.

Table 7.7: Speed versus accuracy performance for different orientation search range configurations.

	Stride	# Ori	Score thresh.	# Cand.	Speed (Hz)	LAMR
Config. 1	4	24 (15°)	0.01	All	8.1	27%
Config. 2	4	36 (10°)	0.01	All	6.7	24%
Config. 3	4	48 (7.5°)	0.01	All	5.9	22%
Config. 4	4	72 (5°)	0.01	All	4.6	21%
Config. 5	4	180 (2°)	0.01	All	2.5	20%

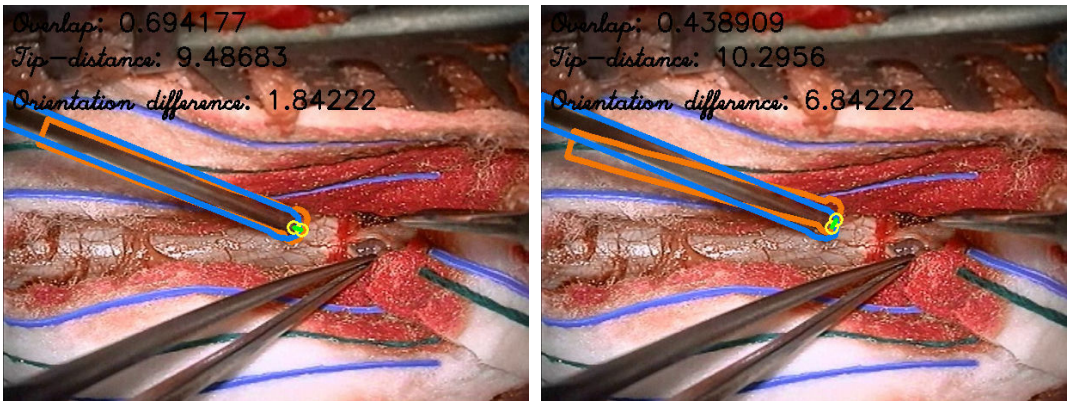


Figure 7.20: Detection results when scanning 180 orientations (left) and 24 orientations (right).

7.4.3.3 Candidate detections

As presented in section 6.5.3, two parameters regarding the selection of candidate detections can have an impact on the speed: the detection score threshold and the upper-bound on the number of detections. The speed-up provided comes from the NMS procedure taking a longer time with a larger pool of candidate detections to process. In table 7.8, we report speed versus accuracy performance for different configurations of those two parameters.

Table 7.8: Speed versus accuracy performance for different candidate detection selection configurations.

	Stride	# Ori	Score thresh.	# Cand.	Speed (Hz)	LAMR
Config. 1	4	72	0.01	All	4.6	21%
Config. 2	4	72	0.01	250	6.5	35%
Config. 3	4	72	0.01	500	6.3	28%
Config. 4	4	72	0.01	1000	5.6	24%
Config. 5	4	72	0.001	All	1.25	16%
Config. 6	4	72	0.0001	All	0.8	16%
Config. 7	4	72	0.0001	10000	1.8	16%

The detection score threshold is highly dependent on the shape model strategy chosen as final scores are partly computed from model weights. In our case, 0.001 and 0.0001 thresholds provide the same detection performance with a LAMR of 16% for a 1.5 speed difference. Using a reduced threshold of 0.01, the LAMR value drops to 21% while in fact results are comparable, until the 10^{-1} FPPI mark. Indeed, the LAMR value is computed in the $[10^{-2}{:}10^{0}]$ FPPI range, but with this reduced threshold, miss-rate versus FPPI curves are ending before reaching the 10^{-1} FPPI mark. As such, the last miss-rate value reached is used up to the 10^{0} mark. As illustrated figure 7.21, lowering the detection score threshold is increasing the number of candidate detections. Roughly 3000 candidate detections per image are obtained for a 0.01 threshold, while this number goes up to 100000 when using a 0.0001 score threshold. Decreasing the score threshold marginally improves detection results while providing up to a 5-time speed-up. Most of the candidate detections with a score in-between 0.01 and 0.001 are removed thanks to the NMS procedure for only a few number of false positive detections added.
The second parameter, an upper-bound on the maximum number of candidate detections, is very tricky to manually adjust as very dependent on the number of tools potentially appearing in the image. The speed-up provided is also very negligible with a 1.5× factor only, for a huge accuracy loss with a 14% Δ LAMR decrease. By keeping the best 250 candidate detections only, some true positives can be missed as shown figure 7.22. The suction being blurry in the showcase example because of fast motion, detection scores in this image area are lower than detection scores around the curette area. As such, detections over the suction tube are not transferred to the NMS procedure, thus increasing the miss-rate.

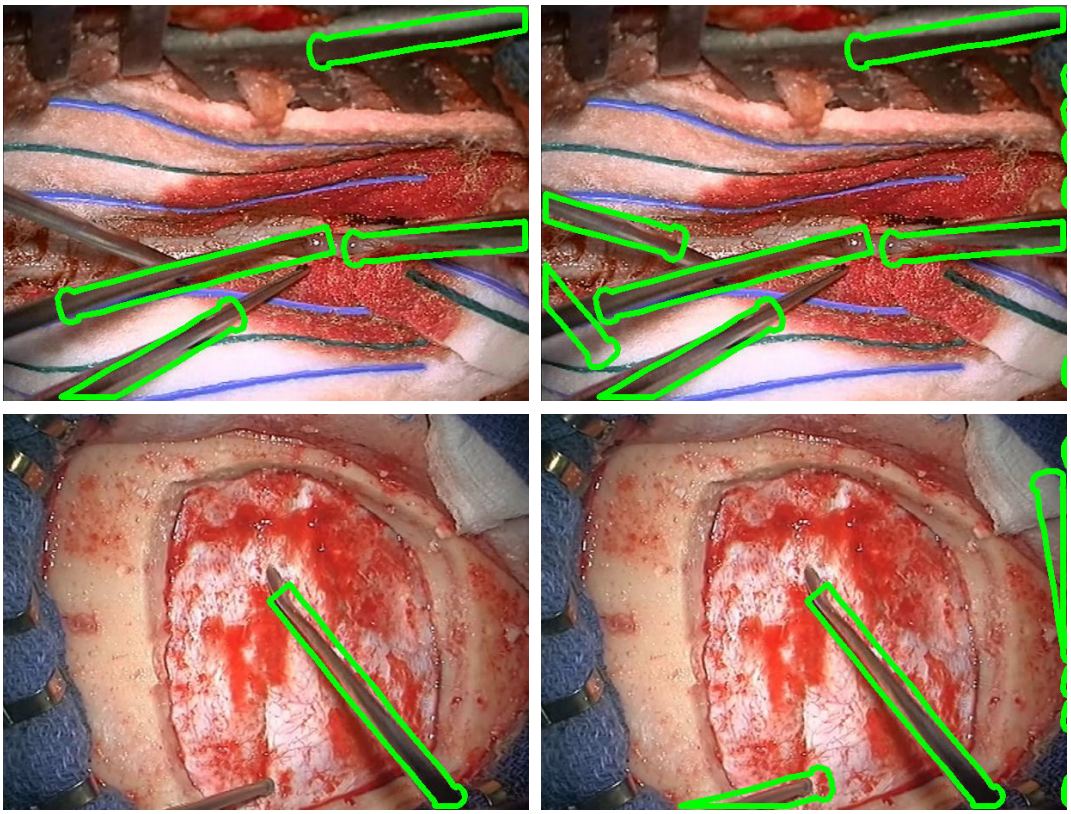


Figure 7.21: *ShapeDetector* results when using a detection score threshold of 0.01 (left) and 0.001 (right).

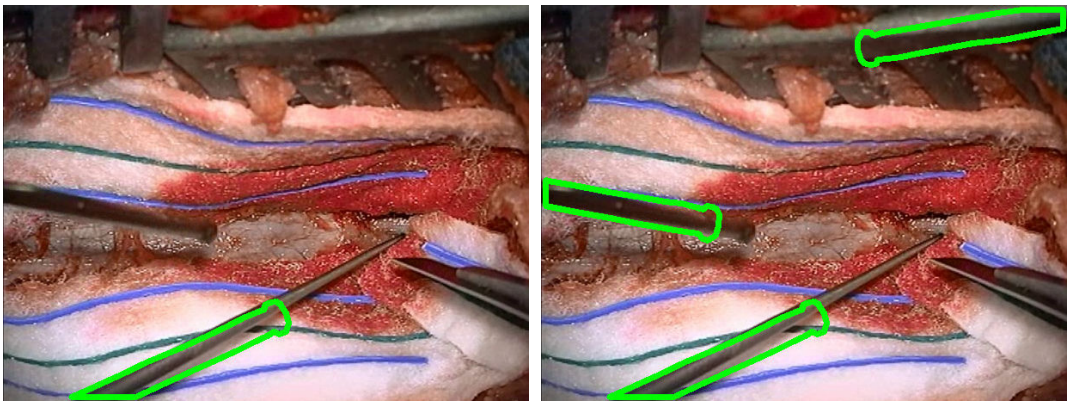


Figure 7.22: *ShapeDetector* results when limiting the number of candidate detections to 250 (left) and without any limitations (right).

7.5 Discussion

7.5.1 Detection bounding geometry

In every pedestrian detector as long as most of surgical tool detectors, detections are represented with square or rectangle windows. This representation is sufficient enough when dealing with real-life objects prone to gravity. However, surgical tools are subject to in-plane rotations and rectangles are covering too much of the image, which does not provide accurate tool pose estimation. Figure 7.23 illustrates surgical instrument detection results displayed with bounding boxes (in pink) and bounding polygons (in green). Bounding polygons represent a better alternative and enable more accurate tool detection.

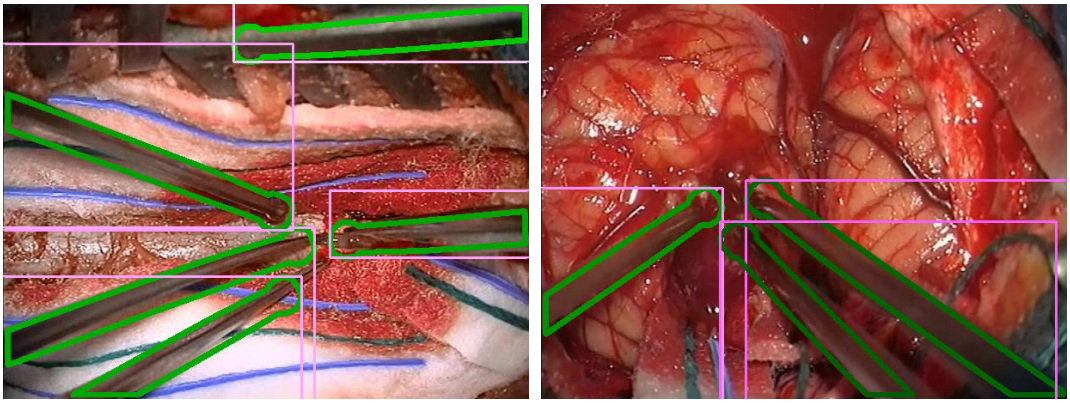


Figure 7.23: Surgical tool detections represented with bounding boxes (pink) and bounding polygons (green).

7.5.2 In-plane rotations

In computer vision images (e.g. for pedestrian), the orientation pose parameter is not taken into account as demonstrated by training images only having to compensate for scale and translation. However, surgical tools can appear through many in-plane orientation or tilt variations, and as such surgical tool detectors must deal with the orientation parameter.
The corresponding main issue with computer vision detectors is the feature representation not being rotation-invariant (e.g. gradient). A model learned for an object at one orientation is not able to detect the same object appearing at totally different orientation as both of their feature representations are widely different. Scanning a set of different orientations can be seen as a naive approach but is preferable to using rotation-invariant features, which would lead to a loss of information. A multi-orientation scanning is conceptually pretty similar to the multi-scale scanning consistently employed.

7.5.3 Semantic labelling

Compared to hand-crafted features from other semantic labelling methods such as Darwin, the *SquaresChnFtrs* semantic labelling is obtaining the best segmentation quality because of its flexible and data-driven approach. However, the *SquaresChnFtrs* semantic labelling is not perfect and resulting maps are relatively noisy and especially inaccurate around instruments boundaries including the tool-tip. Improving the quality of the labelling is essential to retrieve precise tool-tip position and to perform classification. In addition, maybe because of the few available samples, blurred tool pixels are incorrectly labelled.
Having chosen to model our problem using two classes only, one for tools and one for the surgical background, is providing good results. One class per surgical instrument would only draw confusion in the classifier as all the instruments exhibit a similar local appearance. Adding new classes to the pixel-wise classification could be to consider in case of colored surgical instruments such as the blue IOL instrument used for cataract surgeries.
A specific attention should be brought towards changing illumination conditions in the recorded videos, highly altering instruments' color and sometimes introducing shadows in the image. In those cases, new specific classes could be added to the pixel-wise classification to identify image regions with specular reflections or tool shadows.

7.5.4 SVM shape model

Even with hundreds of training images, learning an accurate tool-specific shape template through SVM training is difficult given the various ways to generate positive and negative samples. Using real semantic labelling results to generate SVM training samples seemed to be the most logical alternative as representing best the reality of data to process. However, the SVM learning process is hindered by the semantic labelling noise and resulting models are too noisy. Representing a perfect semantic labelling case, SVM positive samples created from data annotations serve as a good alternative. Generating random negative samples from scratch through Normal distributions is straightforward and easy to implement. Nevertheless, it appears to be an intelligent choice compared to random image cropping. Indeed, because of the nature of input images, especially when considered annotation masks, cropped negative samples would most of the time be completely filled in black or white. As such, they would not serve as viable input samples for SVM learning.
With our current implementation choices, enforcing 2D spatial smoothness in the SVM regularization has shown to be ineffective to induce any noticeable improvements in the detector performance. However, the resulting SVM models tend to be visibly smoother indicating a proper behavior of the regularization term. It might be due to the piece-wise approximation of shape templates used to gain computational speed already enforcing such spatial smoothness in a brute-force manner.
The shape learning strategy aims to modeling the way a surgical instrument is

appearing in the data, usually through many appearance variations such as illumination changes, occlusion, or x-/y-/z-axis tool rotations. As such, a model does not represent the physical reality of an instrument, as would a 3D auto-CAD model. For instruments exhibiting important appearance differences, for example when being viewed from the left side or right side, the created SVM shape might not be representative to the real instrument shape. In case of very different faces for a same instrument, multiple 2D SVM models should be created to avoid a big blurry/messy model which will never be good enough to be triggered at run-time.

7.5.5 Detection and classification

The proposed detection approaches are fully data-driven and do not rely on the use of any kind of prior knowledge to constrain the search space. As such, even if not thoroughly validated, we are confident in their ability to perform detections in another surgical context such as laparoscopy where surgical tools appear very similarly in the images.
Nevertheless, failure modes have been identified such as missed detections or erroneous detections especially regarding the orientation estimation. Some missed detections are the results from the presence of noise in the semantic labelling layer. Erroneous detections in orientation are exposing the limits of the pose estimation strategy coupled to the SVM model, most of the time due to tool-tips being occluded. In both cases, failure modes can be avoided by coupling a tracking approach, as we touch on in the next chapter. Additional features can also help, for example with stereoscopic videos and an access to depth information in order to identify blurred regions in the image. When the microscope is in-focus, those blurred regions represented regions where surgical tools are titled, due to the shallow depth-of-focus. Mixing a label map indicating pixels of tilted surgical tools with the label map from the *SquaresChnFtrs* where in-focus tools are well labelled can produce better overall semantic labelling maps, thus enabling a better pose estimation. Temporal features could have been investigated, however none have shown to be successfully effective in any other prior study. As such, we preferred to focus on spatial features instead of going into uncharted waters.
In the current *ShapeDetector* setup, while running multiple tool models simultaneously is possible even if not real-time, the classification issue remains. Indeed, SVM models can identify the difference between a tool and the surgical background, but are not built to learn how to differentiate shapes of two similar surgical instruments. As a result, the detection score over a suction tube with a suction tube SVM model can be hardly inferior to the one obtained with a bipolar forceps SVM model, as illustrated figure 7.24. Performing tool classification together with detection is not straightforward in the current architecture.

Our initial experiments indicate that only subtle cues enable to distinguish amongst tools (e.g. hook versus suction tube), and thus we believe that more discriminative features are needed to solve this fine-grained classification task. Potentially learning specific classifiers based on surgical instruments local edges and

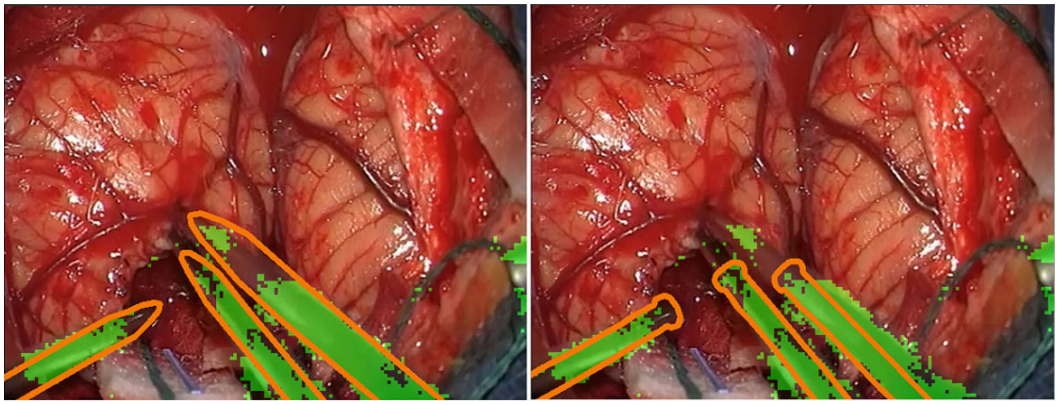


Figure 7.24: Classification issue for the *ShapeDetector* with concurrent bipolar forceps (left) and suction tube (right) models.

applied on top of the global shape detection could solve the classification issue.

7.5.6 Validation methodology

Assessing the performances of an object detection approach can be hard as a relevant validation methodology has to be defined. In this work, we proposed methods not relying on prior knowledge in their design and thus we chose to use standard computer vision metrics for the validation process.
To validate the pixel-wise classification performance, the per-class average metric is used. However, this is a flawed metric since it does not capture the aesthetic quality of the results and results that are equal quantitatively can be far from equal qualitatively. Nevertheless, developing metric assessing the visual quality of images is a work on its own, and as such we chose to rely on this go-to metric.
From our set of 2D detections, validating the estimation of the three pose parameters represented by the tool global position, tip location, and orientation, is the best achievable. Arguably using bounding polygons for the validation improves over previous work that considered only bounding box overlap [Kumar 2013a].
The first validation metric used, the *polygon overlap*, based upon the intersection over union criterion, is state-of-the-art and widely used for object detection in computer vision to assess of overall good positioning. Developed to be used with bounding boxes, we consider the intersection over union criterion to also fit well with bounding polygons with an adaptation regarding the overlap area threshold. Instead of a 50% overlap threshold traditionally used, we decreased it to 25% because of the nature of the elongated polygons. Small variations in orientation can substantially lower the overlap area, and the point of this metric is to assess of accurate location not correct orientation estimation. In retrospect, the traditional threshold could have been used since we observed performances stability until a 60% area threshold. This behavior has been mentioned for pedestrian detection where the evaluation is insensitive to the exact threshold as long as it is below 60% [Dollár 2011].

Having identified true detections from the use of this first metric, we proposed to further assess the pose estimation quality through the tool-tip distance and in-plane orientation difference metrics. Even though being relatively straightforward metrics, they have been previously used in similar work when using tracking approaches [Sznitman 2012] and in body pose estimation validation [Dantone 2014]. The choice to study tip position and orientation accuracy at a fixed rate of 10^{-1} FPPI gives an upper-bound of the pose estimation quality when detections are correct. Given detector behavior in the range $[10^{-2}, 10^{0}]$ FPPI, selecting a fixed rate of 10^{-1} FPPI for the study was the only acceptable possibility.
Obviously, the choice of the 10^{-1} FPPI fixed rate, as long as the obtained miss rates, orientation differences and tip-distance errors are hard to put in perspective without any medical application with specific objectives.

7.5.7 Processing speed

The speed-up provided by the Soft Cascade is hard to predict as extremely dependent on the cascade design itself. Methods spending more time on detection score computation than on feature computation are expected to benefit more from the Soft Cascade. For the *adapted SquaresChnFtrs*, using a Soft Cascade layout is particularly useful because features are computed only once while multiple cascades are evaluated. Being able to skip the evaluation of 1500 weak classifiers for each cascade is an enormous speed-up considering up to 72 different cascades (i.e. 72 orientations).
Regarding the *ShapeDetector*, the GPU implementation is enabling the fastest processing speed possible with respect to detection performance. Only specific GPU code optimization or the next hardware generation could bring further speed-up.

Detector processing time can be additionally decreased by limiting specific parameters range such as the orientation step or image strides, thus obtaining coarser detections quality-wise, known as the speed versus accuracy trade-off. Modifying parameters range or search range is user-dependent and can be easily done in-between consecutive processings as not impacting on the algorithmic structure of the detector.
For example, processing the full range of orientation [0°:5°:360°] is mandatory without any assumptions over tool position in the image. However, knowing the surgeon hand-dexterity ahead of a detection process, only a sub-range of orientations necessitate to be scanned, thus providing detection speed-up. Similarly, the scale search range can be reduced to a couple of scales, given the possibility to retrieve in real-time the microscope zoom value. Only a look-up-table is required to couple microscope zoom values with tool model scales.

7.5.8 Application usage

Many solutions investigated to solve the surgical instrument pose estimation problem required significant changes to operating room setup. Instead of relying on 2d video signals (as presented here), some methods require additional tags (e.g. RFID technology [Bardram 2011]) or specialized optics (e.g. Kinect-based systems [Lea 2012]). Only requiring the video feed from a surgical microscope, which is a standard medical equipment for most hospitals throughout the world, our proposed approach can directly be used in existing operating rooms. The surgical instrument detection task is a key element for in-vivo surgeon assistance within context-aware computer assisted intervention systems. Many other applications in the surgical field rely on tool detection from videos. Example applications are automatic indexation of surgical videos for faster browsing [Lalys 2012], or surgeons' technique comparisons to identify best practices [Neumuth 2009].
Even though not tested within a higher-level medical application, current results are encouraging since the approach is data-driven and not relying on prior knowledge. As such, we are confident in the possibility to achieve even higher performances by adding some assumptions when designing a targeted medical application.
Thanks to its architectural design and a GPU implementation, our *ShapeDetector* is currently running in-between 5-8Hz, which is close to fast enough for integration in real-time systems. The soft cap of 25Hz (i.e. recording device speed) should be reached with proper GPU code optimization or with the new generation of computer hardware.

Part III

Perspectives and applications

Chapter 8

Robustification process towards medical applications

Contents

8.1 Introduction

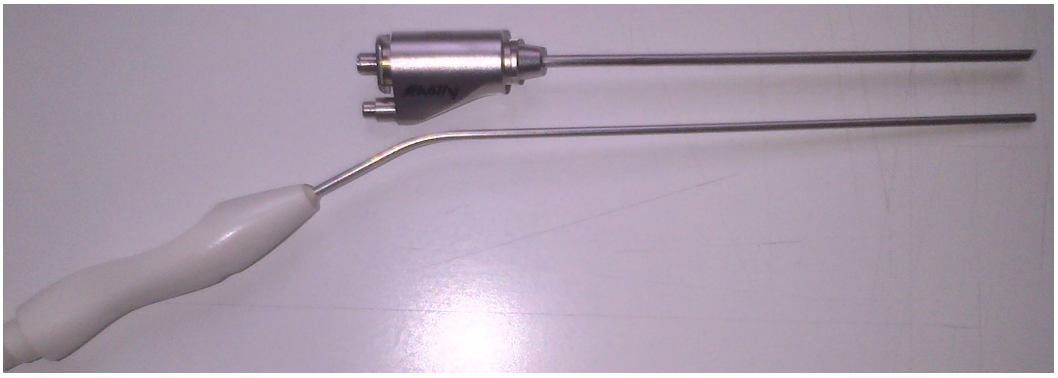
In previous chapters, we introduced an almost frame-rate spatial detector and showed interesting performance obtained over our data-set. Depending on the final medical application relying on tool detection, current results may or may not be

enough, especially regarding the inability to perform tool classification. Taking a look at pipeline robustification strategies might be necessary to achieve a specific goal within a targeted medical application. While it has already been proven that adding layers such as tracking to a robust spatial detector can only improve overall results, tool categorization is a trickier problem as image-based approaches can be helpless in case of limited visual differences between two surgical tools.
In this chapter, we start by investigating possible solutions to externally tackle the surgical tool classification problem and propose an adequate candidate solution (see section 8.2). Then, we present in section 8.3 a tracking solution to refine spatial detections.

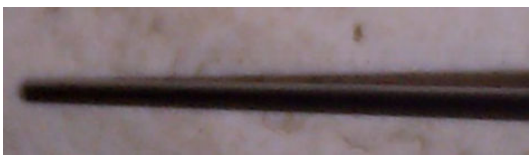
8.2 Tools classification through external markers

8.2.1 Introduction

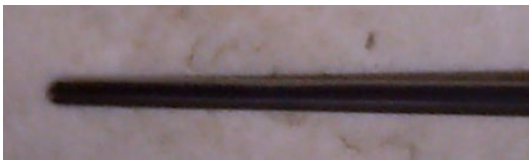
Performing simultaneous surgical tool detection and classification using video input only can prove difficult as many instruments share a similar color (e.g. gray-ish), overall shape (e.g. tubular structure), or texture. A striking example is illustrated by figure 8.1, where a suction tube and an endoscope are displayed side-by-side.



(a) Side-by-side endoscope and suction tube



(b) Suction tube



(c) Endoscope

Figure 8.1: Example of surgical instruments with almost no visual differences when in use.

Between both instruments, only subtle differences exist to perform categorization to the naked eye. Depending on the microscope recording setup (i.e. the field-of-view) and the surgical reality (i.e. how instruments are moved), it might be impossible to perform classification from still images as crucial visual landmarks

may not be visible.
In the literature review regarding surgical tool detection (see section 2.3), we excluded a category of approaches relying on external markers. Although we deemed such approaches unfit for the creation of robust spatial tool detectors, they represent a viable solution when video is not enough. In case of extremely similar surgical instruments, adding external markers will enable the otherwise impossible tool detection and categorization.
A succinct review of the literature regarding surgical tool detection from external markers is provided section 8.2.2. Then, in section 8.2.3, we describe our external marker solution and its corresponding identification technique. Finally, in sections 8.2.4 and 8.2.5, we present preliminary results regarding external marker correct identification and provide a discussion.

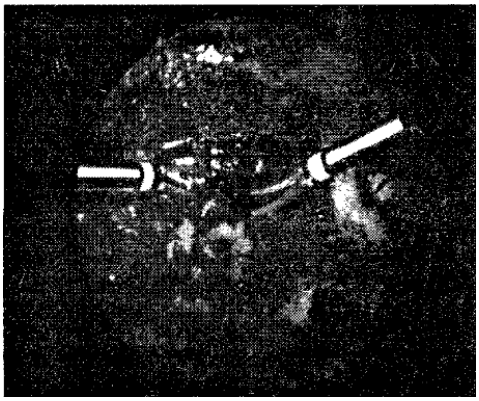
8.2.2 Review of the literature

Because operating rooms and hospitals in general represent a sensitive environment with many constraints and regulations, this review of the literature focuses on technologies that have been investigated in a medical context. Mainly two categories of markers have been extensively studied: color/shape markers retrieved from image-based analysis and RFID markers retrieved from external additional sensors.
In everyday life, barcodes represent the most known category of external markers, being used for various purposes especially in an industrial context (e.g. supermarkets, postal services).
Below, each category is detailed and discussed, starting by color/shape marker in section 8.2.2.1, then RFID markers in section 8.2.2.2 and finally barcode markers in section 8.2.2.3. A global conclusion regarding the possibility of usage within our application context is given in section 8.2.2.4.

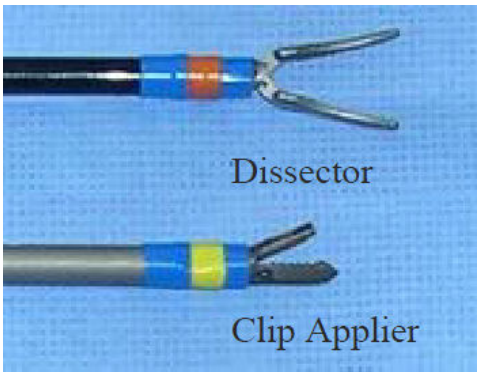
8.2.2.1 Color/shape markers

This first category can be summarized as external markers identified through low-level image-based analysis, thresholding for example. Such markers with a distinctive color or shape are usually placed somewhere along surgical instruments shaft. Figure 8.2 gives an overview of color/shape markers.

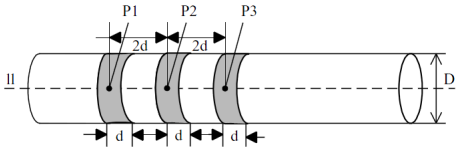
In the early work from Casals et al. [Casals 1996], a marker made of three elements has been proposed to identify the presence of surgical instruments during laparoscopic surgeries (illustrated by figure 8.2a). Two horizontal straight lines are placed along the surgical instrument axis in order to retrieve location and orientation. In addition, a ring mark located at the surgical instrument rod end enables 3D position retrieval from its diameter measurement. No color has been specified for the marker itself and basic mathematical morphology and filtering techniques are employed to perform the detection. In 2002, Zhang and al. [Zhang 2002] proposed to add on a surgical tool three black rings of a same size and spaced from each other by a known distance (see figure 8.2c). Marker locations



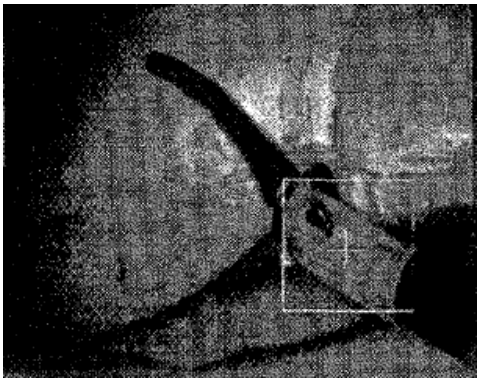
(a) Casals et al.



(b) Ko et al.



(c) Zhang et al.



(d) Wei et al.

Figure 8.2: Color/shape external marker placement examples throughout existing studies.

are retrieved using low-level image processing such as thresholding on grey-level images.

Instead of relying on markers' shape, Wei et al. [Wei 1997] proposed to effectively use color information to perform tool detection. A color distribution analysis based on the HSV color space is performed over laparoscopic data to identify the most suitable marker color. A final near-cyan plastic ring has been proposed (illustrated figure 8.2d), detected through thresholding on the Hue and Saturation color space channels. Similarly based on background color distribution analysis, Tonet and al. [Tonet 2007] added a cyan ring around the tool-tip. Focusing on cholecystectomy surgeries, Ko and al. [Ko 2005] proposed to add one specific color marker on each and every surgical instrument (see figure 8.2b).

Finally, bio-compatible color markers have been evoked for an in-vivo real-time tracking of surgical instruments in laparoscopic surgeries [Bouarfa 2012]. Exclusively relying on color information, this system is claimed to be robust to partial occlusion and smoke. In addition, performing a multi-instrument detection is possible, however as more and more different colors are used, the confusion increases between those colors. Generally, using colors with a maximum distance

from the surgical background in the hue space is preferred. Pink and red colors are too easily confused with the surgical background as opposed to blue and green colors.

To conclude, strengths lie in the image processing simplicity to accurately retrieve the location of the marker as long as in an easy design and placement over instruments. While adding color or shape information increases the visibility of one tool, those approaches are usually very sensitive to illumination variations and occlusions. Markers placed too close to the tool-tip can easily be occluded or hidden under anatomical structures. For markers positioned along the tool shaft (e.g. straight lines), visibility issues might arise due to surgical instruments undergoing in-plane rotations. For multiple parts instruments (e.g. pliers or forceps), placing the marker on the shared tubular part could be an issue as it might not be visible in the FoV of the recording device. For color markers, performing surgical background color distribution analysis is straightforward but illumination variations hinder the recognition process especially in case of low-level thresholding operations. In addition, when many different instruments have to be identified, finding enough color ranges to differentiate each instrument from the other ones as long as from the surgical background might be difficult to achieve.

8.2.2.2 RFID markers

Before an introduction of the related work on RFID markers use, we start by giving a quick overview about the RFID technology and its applications and barriers in health-care.

RFID technology overview
Radio-Frequency IDentification (RFID) is a wireless use of electromagnetic fields to transfer data for automatic object identification and tracking purposes. A RFID system commonly includes hardware components necessary to emit and receive signals (e.g. tags, readers, antennas) and a software component to process such signals.
Placed on surgical instruments, two types of RFID tags exist: passive or active, depending on powering techniques used. Passive tags, without battery power, can only communicate with the RFID reader when sitting in its electromagnetic field. On the other hand, active tags can broadcast a response signal towards the reader as they are self-supplied in power.
Regarding the software component, RFID readers scan tags and send the information to the system for further processing such as signal filtering for noise reduction of signal analysis. In health-care, RFID systems are commonly combined with other technologies such as Bluetooth or mobile devices. As for the tags, passive ones are primarily used for patient and drug identification and active ones for tracking purposes.

Applications and barriers in health-care
Over the past years, the use of RFID markers in health-care has grown rapidly.

From a recent literature review, performed by Yao et al. [Yao 2010], RFID use in health-care can be summarized into five categories:

- Tracking applications: aiming at large scale tracking of assets and equipment in hospitals. They can also be considered for vulnerable patients such as elderly people, dementia patients, children and newborn.
- Identification and verification: for drugs and sensitive medical supplies (e.g. blood bags) in order to alleviate drugs counterfeiting, theft and misuse.
- Sensing: for sensor-derived data collecting and integration with physical and chemical sensors (e.g. humidity sensor, temperature sensor, chemical sensor...).
- Interventions: for automating care with self pill-dispenser providing patients with their dose safely, or guiding pathway such as indoor navigational system form blind people.
- Alert and triggers: for preventing medical equipment such as sponges to remain inside a patient after the end of a surgical procedure. Or, for preventing a patient to ingest the wrong drug with RFID antenna wristbands.

Within hospital environments, due to the electromagnetic nature of the RFID technology, potential side effects have been investigated especially regarding potential interference between Ultra-High Frequency (UHF) and other devices. In [Van Der Togt 2008], 68 cases of interference were observed from 246 tests, ranging from minor effects (e.g. noise on computer monitors) to potential hazardous failures (e.g. stopping infusion pumps or ventilators) and occurred at distances from $1\,cm$ to $6\,m$ away from the interfering device. Conversely, Christe et al. [Christe 2008] reported no interference in 1600 tests on five devices at many different distances ranging from $0.3\,m$ to $1.8\,m$. As outlined by Houliston et al. [Houliston 2009], chances of interference grow with some specific factors such as higher output power of the RFID system, shorter distance between RFID reader and medical devices, and the presence of a tag on a device. In addition to interference, a total of six different categories of barriers have been enlightened:

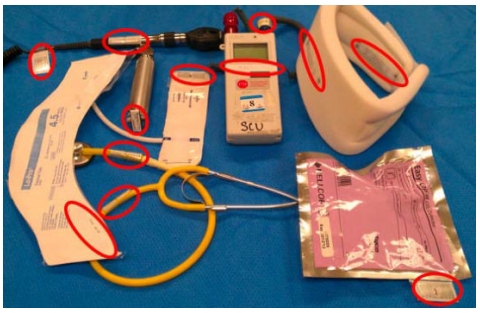
- Interference: electronic medical devices may be hindered in their common behavior in the presence of high-frequency RFID.
- Ineffectiveness: tag placement remains the main factor in incorrect identification. RFID tag readability is strongly dependent on factors such as angle of rotation and reading distance.
- Standardization: interoperability across providers is difficult because of the lack of standardization of the RFID protocols at hardware and software levels.
- Cost: infrastructure installation cost as long as integration cost are too expensive for a tracking system in an hospital.

- Other barriers: lack of organizational support, trust issue, security concern ...

Related work

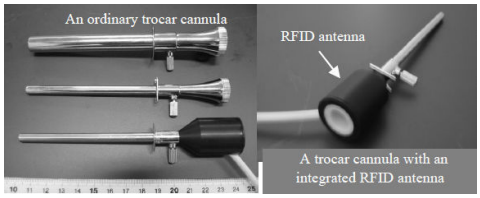
In this paragraph, we pay a closer attention to previous works targeting surgical instrument detection or tracking within the OR (illustrated in figure 8.3).

(a) Bardram et al.



(b) Parlak et al.

(c) Miyawaki et al.



(d) Miyawaki et al.

Figure 8.3: Different RFID use from existing works of the literature.

Bardram et al. [Bardram 2011] performed surgical phase recognition through RFID identification of surgical tools equipped with passive RFID tags. Tables and trolleys on which tools are placed had built-in RFID readers. As many medical equipment are metallic (e.g. instruments, tables), UHF RFID has been used. For detection, nurses were equipped with wireless palm-based RFID sensors (see figure 8.3a). This sensor is composed of a low-capability reader module, only able to detect one tag at a time in the low frequency range, meaning that a Low Frequency (LF) tag on instruments is also necessary. Hence, every surgical instrument was equipped with both LF and UHF tags. The setup has only been used in a simulated environment and is not yet suitable for deployment as many security, hygiene, and ergonomic issues still need to be addressed. In addition, small surgical instruments such as surgical needles are difficult to tag because of their small width.
Also in a context of surgical activity recognition, Parlak et al. [Parlak 2011] proposed to select RFID tags type and placement depending on instruments raw material and shape (see figure 8.3b). They identified long-term interaction instruments, attached

with two tags, one at the point of contact with the human body (i.e. to detect interaction) and one where the instrument is likely to remain exposed to the RF signal. And short-term interaction instruments only provided with a single tag as an accurate tracking is less crucial. Placing two tags on essential surgical instruments was generally helpful, however it can be hard to apply in practice, especially with small contact surfaces (i.e. small instruments). To reduce side-effects from human presence and system movements, the RFID antenna was mounted at the OR ceiling, facing the floor. Several types of UHF passive tags, including non-metallic ones, were studied yet not presenting significant impacts on detection performances. No interference between the UHF RFID system and patient monitors were experienced during the experiments.
Miyawaki et al. [Miyawaki 2009] developed a UHF RFID system specific to surgical instrument detection in endoscopic and laparoscopic surgeries. RFID readers were placed on the end of trocar cannulas (shown figure 8.3d) to detect moments of tool insertion and removal. Instruments were equipped with a passive ceramics-encapsulated disk-type tag, resistant to water and heat conditions. They designed a special device to easily connect and adapt an RFID tag to an instrument (see figure 8.3c). This special device is composed of a spring and a stopper such that the tag can be placed at any appropriate position along the shaft of the surgical instrument. The spring has the property to be low enough to give no resistance against a surgeon's hand during a procedure. Electromagnetic interference occurred mainly because of an electric knife used to destroy tissue with electricity, stop small vessels bleeding or cut through soft tissue. Even the smallest output power from the knife, when in cutting or blend modes, interrupted the RF communication.
Finally, Kranzfelder et al. [Kranzfelder 2014] studied the reliability of sensor-based real-time detection in the context of laparoscopic cholecystectomy for use under clinical conditions. Passive RFID transponders and UHF RFID antennas were selected to form the setup because of an easy integration into routine OR workflow. Some mismatches were experienced between RFID recordings and the reality. Indeed, even small levels of output power can interrupt RF communication.

Conclusion
RFID tags have proven to be quite effective for higher-level applications such as surgical phase or activity recognition. The positioning of the RFID antenna has to be well thought and done carefully in order to retrieve signals with maximal accuracy. Passive tags have been the preferred solution, but their placement on surgical instruments can be difficult and many instruments might not be eligible (e.g. needles).
While Ultra High Frequency appears to be more robust than Low Frequency, issues arise especially with interference, reflection and shielding. However, even if multiple studies have been conducted on the matter, it is still not 100% sure that RFID does not have any adverse impact on other medical devices. Be it as it may, the technology has been deemed safe-enough for use in in-vivo medical applications

within the OR.
Additionally, installing the RFID technology is extremely costly given the amount of element pieces necessary, which is an obstacle as only few hospitals can afford it.

8.2.2.3 Barcode marker

A barcode marker is an optical machine-readable representation of a pattern specific to the object to which it is attached. Two categories of barcode exist, depending on their pattern dimensionality: either 1D or 2D. Linear one-dimensional barcodes are systematically represented by parallel lines of varying widths and spacings. Evolved two-dimensional barcodes rely on rectangles, dots, hexagons and other geometric patterns. Figure 8.4 displays existing barcode representations.

(a) EAN (b) Pharmacode

(c) Aztec code (d) QR code

Figure 8.4: Examples of barcode markers. First row: one-dimensional, second row: two-dimensional.

Barcode recognition is mainstream in everyday life, especially commercially to automate supermarket check-out systems, keep track of mail or airline luggage, or for ticket offices. Originally barcodes were scanned by special optical scanners (i.e. barcode readers) but many image-based analysis computer software became available (e.g. ZBar, LeadTools, mobile phone applications). Some recent works performed standard 1D/2D barcode recognition using image processing techniques, mainly for mobile phone use [Gallo 2011, Rathod 2012].
Barcodes are also widely used in healthcare and hospital settings mainly for patient identification through identification wristband. However, they have not been used

within Operating Rooms in a tool detection/classification manner, probably because of two major constraints. First, the full extent of a barcode has to be visible in the surgical field-of-view for image-based analysis, which can be difficult because of barcode length or occlusions. Secondly, barcodes need to be wrapped around surgical instruments to be visible under various in-plane rotations, and identical at any given orientation. Another solution, not relying on image-based analysis, would be through the use of RFID-like systems, with body-worn barcode readers equipped for example on nurses.

8.2.2.4 Conclusion

Towards surgical instrument classification, three main categories of markers have been identified: color/shape markers, RFID markers and barcode markers; each one presenting strengths as long as weaknesses.
Color/shape markers are very easy to set-up and detect, however detection techniques rely on low-level image processing (e.g. threshold) and as such are extremely sensitive to illumination variations or occlusions. Moreover, the more surgical instruments to detect, the more confusion between close-range colors. Tested in in-vivo conditions, no side-effects were enlightened from the experiments.
On the other hand, RFID markers represent the new generation of markers for surgical applications as shown by the quantity of recent work focusing on the subject. While detection/classification performances have shown to be very significant, setting up a robust RFID system in an OR is complex because of interference with other medical devices and cost. Many studies have been conducted to ensure patient and staff safety in in-vivo conditions without reaching an unanimous consensus.
Finally, barcode markers have shown to be very effective and robust in industrial applications but have never really been used in an OR context, indirect use in hospitals only. Using traditional barcodes on surgical instruments seems difficult to set-up, especially because of in-plane rotations, tilt variations, and lighting conditions where standard identification techniques are likely to fail.

8.2.3 Marker detection

From this review of the literature, an off-the-shelf use of external markers does not seem possible; each category presenting advantages as long as drawbacks. The main conceptual design concern expressed for the *ShapeDetector* also applies here, which is to modify as little as possible the OR setup. Additionally, the marker technique should not have an impact on *ShapeDetector* detection results as only the classification part is handled by external markers not the detection itself.
Under these circumstances, the RFID technology is not fitting as requiring extensive modification of the OR setup to place antennas, in addition to be cumbersome and expensive.
Regarding color-based markers, while being extremely efficiently retrievable, they would greatly impact first stage of the *ShapeDetector* (i.e. the semantic labelling).

As many additional classes as marker colors will be necessary, overall performance will most probably suffer from it.
While barcodes represent an easy alternative with packaged and freely available detection algorithms, they will seemingly impact the semantic labelling in the same way as color markers could do. As illustrated in figure 8.5, too much of the original surgical instrument is occluded when using a standard barcode.
Eventually, we opted to design a marker fitting best our needs, presented in section 8.2.3.1. Then, we introduce in section 8.2.3.2 the corresponding image-based analysis technique to retrieve the marker from still images.

Figure 8.5: Illustration of an EAN barcode placed on a surgical instrument.

8.2.3.1 Chosen marker

As chosen solution, we propose to create a marker made of a set of black rings which would visually look like a barcode. Using black bars only, without their interlaced white counterparts such as in real barcodes, the hindrance on semantic labelling results is almost nonexistent. In addition, the control is total over the number of rings forming the marker, the spacing between rings, and the width of each ring, as illustrated in figure 8.6.
Such markers are to be placed around the shaft of surgical instruments, at a safe distance from the tool-tip to avoid occlusion issues, and are identical at any given in-plane orientation thanks to their circular pattern. As an alternative, such black rings can also be engraved directly onto the surgical instrument to avoid dealing with marker placement and withdrawal.

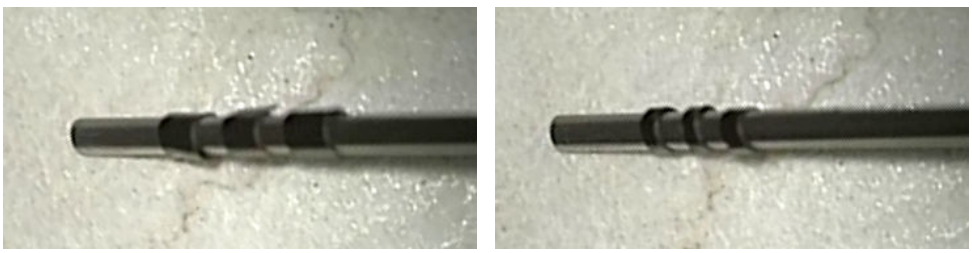


Figure 8.6: Illustrations of our proposed barcode-like marker, placed along the shaft of a suction tube.

8.2.3.2 Proposed method

As mentioned in the introduction, detecting markers is required to perform surgical tool classification only. As such, we rely on one assumption: barcodes can only be identified on already detected surgical tools.
A three-step approach is employed to detect and identify a barcode, thus providing a tool class for every candidate detections from the *shapeDetector*. Each detection is pre-processed for easier barcode recognition, which is performed following a similar strategy to the state-of-the-art technique [Gallo 2011]. From a detected barcode, its identification through pattern extraction is mandatory for matching in a barcode data-base to access the corresponding tool class. Each of the three components is described in details in the following.

Pre-processing
From a candidate detection, the sub-image containing the surgical tool is extracted and rotated to appear at orientation 0°. The resulting sub-image illustrated in figure 8.7 is used as input to the recognition step.

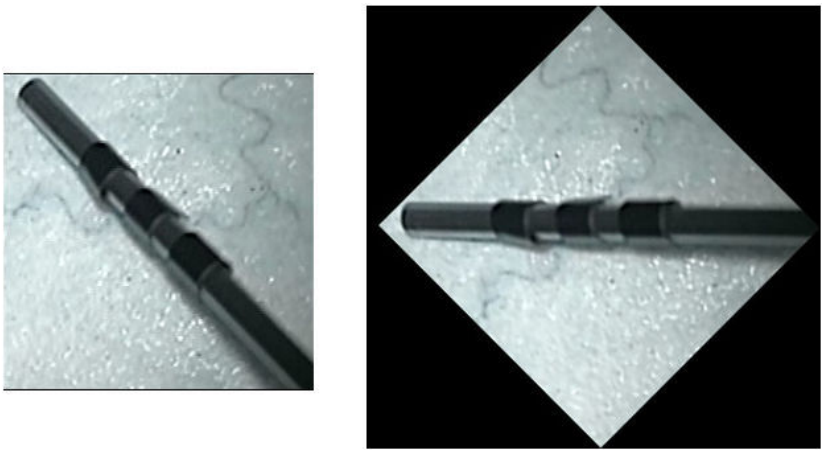


Figure 8.7: Marker recognition: pre-processing step. The found detection (left) is horizontally aligned (right).

Marker recognition
Assuming a barcode being present in the processed sub-image, an horizontal scanning is performed following a line starting from the estimated tool-tip position up to the sub-image border (see figure 8.8). After image color type conversion from RGB space to HSV space, the intensity value in each pixel along the virtual horizontal line is retrieved.
Knowing the barcode color (i.e. black), a threshold is applied to keep on intensity values lower than 100 only. Other intensity values having a high chance of belonging to the surgical instrument itself or the surgical background and as such are not relevant to perform the marker identification.

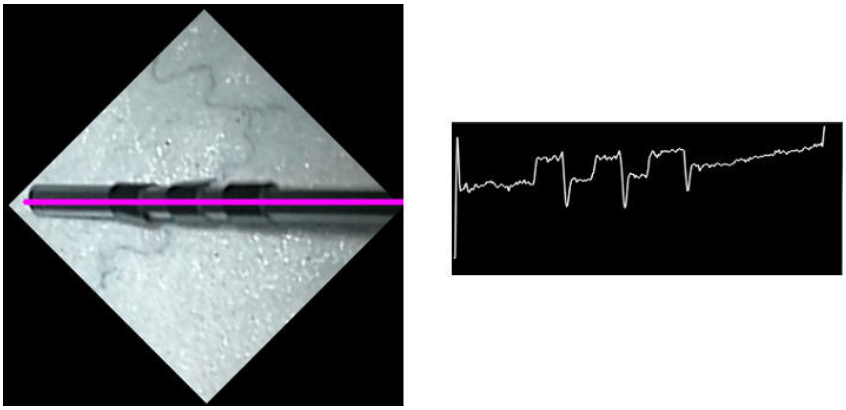


Figure 8.8: Illustration of the 1D signal extraction. The pink line in scanned (left) to produce the signal (right).

The line of intensity values retrieved can be considered as a one-dimensional signal. As such, standard signal processing techniques are applied with the objective to retrieve mandatory marker information such as the number of bars, their spacing, and their width. First, the signal is detrended to better visualize variations between background pixels' intensity and barcode pixels' intensity. Then, a low-pass filtering is performed in order to smooth the signal as high frequency variations are only noise from pixels' intensity variations. A limited number of 20 coefficients has been used for the filtering to restraint the delay between input and output signals. As identifying switches from surgical instrument to marker's bars is key for marker identification, the derivative of the filtered signal is computed. Finally, on this derivative signal, maximum positive and negative peaks are identified as they should correspond to in-bar and out-bar limitations. An overview of the four-step detection technique is provided in figure 8.9.

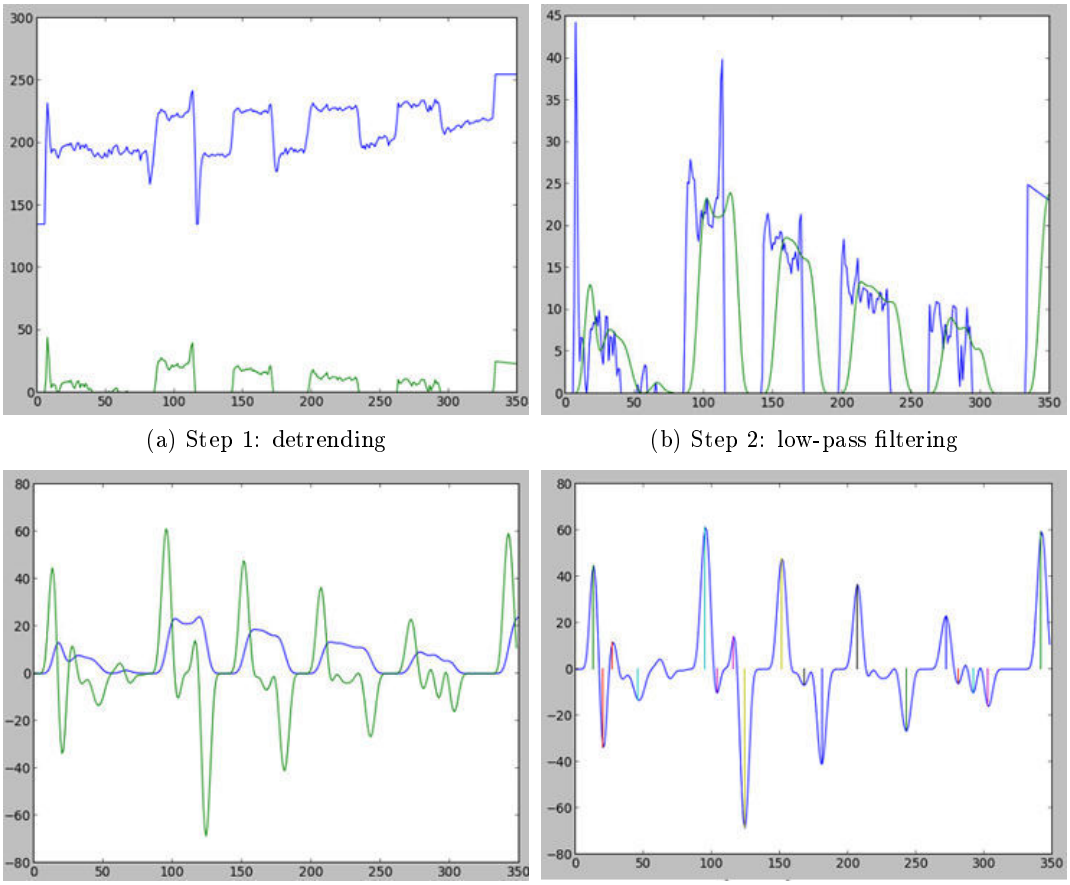
As output of the marker recognition step, a list of maximum positive and negative peaks with their respective amplitude values is created, necessary to perform barcode identification.

Marker identification

The goal of the marker identification step is to retrieve as accurately as possible the barcode pattern (i.e. number of bars, their width and spacing) in order to match it with known barcode patterns to identify the corresponding tool class.

To do so, a pairing between maximum positive and negative peaks is executed, based on two parameters: in-between pixel distance, and amplitude difference. The pairing process can be simplified by assuming limited ranges of in-between distances when looking to identify one specific pattern.

If the extracted barcode pattern can be matched with a known barcode pattern, the corresponding class label is given to the detection on which the barcode has been found and identified.



(a) Step 1: detrending

(b) Step 2: low-pass filtering

(c) Step 3: derivative

(d) Step 4: Peak detection

Figure 8.9: Illustration of the 1D signal processing steps. In the first three steps, input signals are represented in blue and output ones in green.

8.2.4 Results

To perform validation studies regarding marker detection, a dedicated new data-set has been acquired as the *NeuroSurgicalTool* data-set does not contain tools with markers. The following phantom setup was used for the acquisition: a suction tube covered with a marker has been moved over a fake surgical background make of a skull. Motions performed were not mimicking real in-vivo gestures but rather illustrated in-plane orientations and tilt variations.
Table 8.1 sums up different marker configurations employed, with the first bar of the marker always placed $7\,mm$ away from the tool-tip. Almost the same movements were performed for each configuration to enable a comparison as fair as possible, yet not identical since a robotized arm has not been used.
Two studies are proposed: a first one evaluating marker impact on *ShapeDetector* performance (section 8.2.4.1), secondly an evaluation of marker detection technique

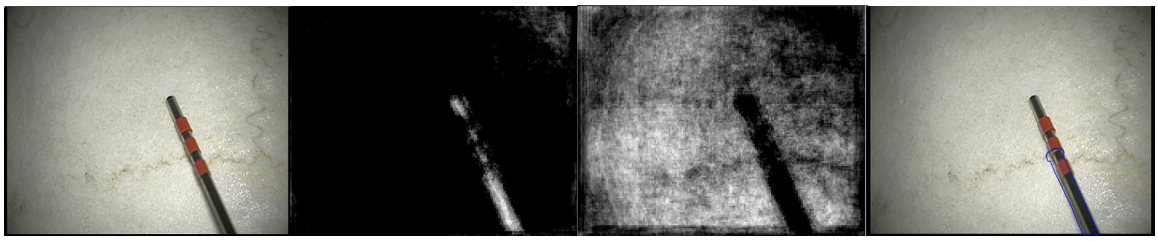
performance (section 8.2.4.2). No annotations were performed on the data-set, as such qualitative and visual results are provided only.

Table 8.1: Barcode marker employed with different ring configurations.

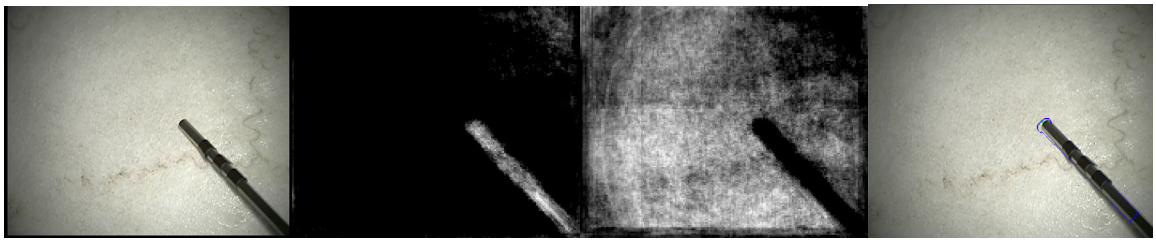
	Number	Width (mm)	Spacing (mm)	Color
Config. 1	3	3	3	Red
Config. 2	4	3	5	Red
Config. 3	3	3	2	Red
Config. 4	3	3	2	Black
Config. 5	4	3	2	Black
Config. 6	3	1	2	Black

8.2.4.1 Impact on *ShapeDetector* performance

As illustrated figure 8.10, the color choice is having an impact on the detector performance. Red color bars induce holes in semantic scores, especially for the tool class, leading to inaccurate detections over the tool starting after around the last bar instead of the tool-tip. Conversely, when using black bars, semantic scores are almost not altered and detections are as good as if no marker was on the surgical instrument. As such, our current setup necessitating to capture the entirety of a marker to perform its identification, using red bars is not compatible.



(a) Original (b) Tool class (c) Background class (d) Detection

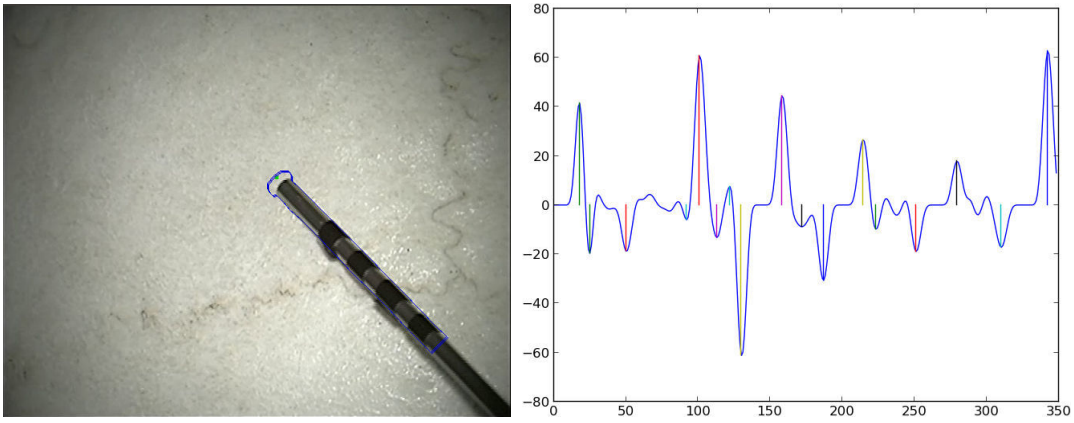


(e) Original (f) Tool class (g) Background class (h) Detection

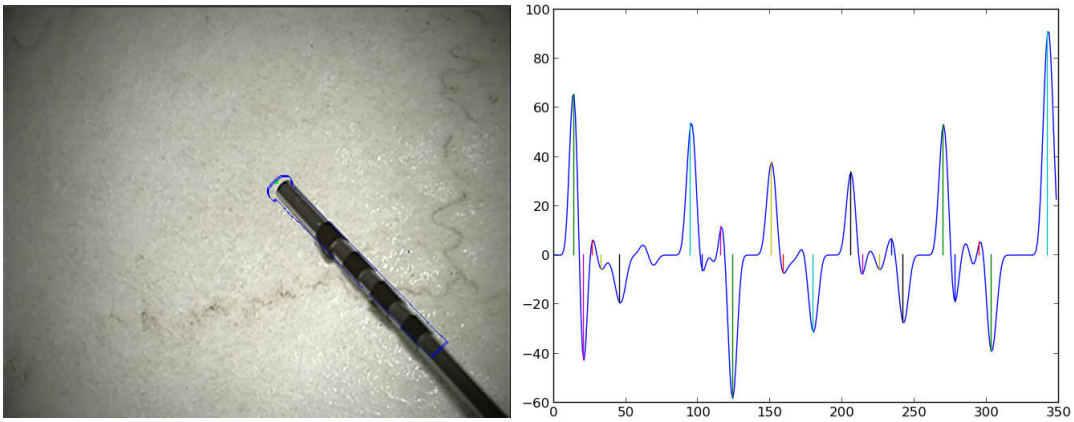
Figure 8.10: Semantic labelling results with a red barcode marker (top row) and a black barcode marker (bottom row).

8.2.4.2 Marker detection performances

In figure 8.11, we report marker identification success modes. The first couple of positive and negative peak is associated to the transition from the background to the surgical tool. Each of the other peak couple is corresponding to a black bar composing the marker. In general, good marker detection results are obtained when the surgical tool is close to being plane-aligned.



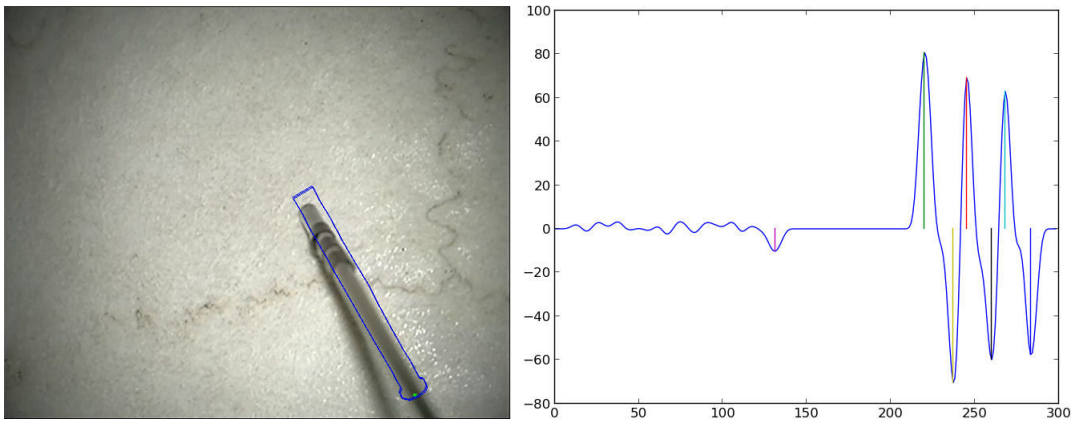
(a) Detection (b) Processed signal



(c) Detection (d) Processed signal

Figure 8.11: Success modes for the proposed marker detection technique.

However, when important tilt variations are applied, as illustrated figure 8.12, the detection is not well placed over the surgical instrument, and as such the processed signal can not be identified. Tilt variations are detrimental to well-positioned detections, but also to the barcode identification as proportions between bars are not preserved.



(a) Detection (b) Processed signal

Figure 8.12: Failure mode for the proposed marker detection technique.

8.2.5 Discussion

Given the nature of employed external markers, performing validation studies in in-vivo conditions does not appear feasible because of safety reasons. However, in a phantom context it is quite easy to add and remove different marker configurations in order to study the marker retrieval technique robustness. This approach is preferable to blending markers into in-vivo images as edges between markers and instruments would not be realistic.
While efficient to classify one surgical instrument from many others, such approach does not seem fit for each and every tool. For high microscope zoom values, only a small portion of a surgical instrument is visible in the FoV. As such, a marker made of four bars could not be visible to its full extent. Additionally, close marker configurations with only a one-bar difference can easily be confused in case of erroneous detection or high tilt variations. To avoid confusion cases, the solution is to focus on the identification of one specific barcode only at run-time.
Not to disturb the semantic labelling process and *ShapeDetector* results, the chosen color must be within a close color-range to modelled surgical instruments. At the same time, the color difference between the surgical instrument and the barcode must be high enough for the proposed signal processing technique to perform well. Obviously, having well placed candidate detections over surgical instruments is critical to ensure the best possible conditions to detect and recognize barcodes.

8.3 Tracking

8.3.1 Introduction

As mentioned in chapter 2, when starting from a strong spatial detector, the overall performance is necessarily enhanced from the addition of extra information such as tracking. Even if the added value on per-frame detection itself is not clear, we propose to discuss the interest in coupling detection and tracking.
In addition, the marker-based classification technique previously introduced has proven to be quite sensitive, especially to tilt variations. As such, adding a tracking layer could be a solution to propagate instruments' classes in case of missed identifications.
We chose not to perform a literature review regarding stand-alone tracking approaches or detectors coupling a tracking solution. Instead, we decided to rely on the go-to tracking technique: Kalman filter, because of its rather easy usage and off-the-shelf availability.
Below, we start by introducing our strategy to integrate Kalman filters within the *ShapeDetector* pipeline (section 8.3.2). Then, in section 8.3.3, we report preliminary and qualitative *ShapeDetector* performance from the use of a tracking layer.

8.3.2 Tracking integration

We start by presenting in section 8.3.2.1 our design choices for the Kalman filters. Then, the track creation process is described in section 8.3.2.2. Finally, the coupling of the tracking layer with our *ShapeDetector* is explained in section 8.3.2.3.

8.3.2.1 Kalman filter design

Regardless of targeted in-vivo medical applications, when dealing with image-based surgical instrument detection, the three following parameters must be retrieved accurately: global tool pose, tool orientation and tool-tip location. While those three parameters are intrinsically connected, we propose to model a track with two Kalman filters; the first one estimating the tool-tip location and the second one the tool orientation.

Position filter
Most *ShapeDetector* failing modes occur due to instruments being occluded by other instruments or anatomical structures, under high tilt values, or suffer from blurriness because of rapid motions. In such cases, the *ShapeDetector* is unable to provide candidate detections, hence a prediction regarding tool location is of interest. To do so, we propose a first Kalman filter in position and speed, using the tool-tip location as reference point, illustrated by equation 8.1.

$$Kalman_{position} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{8.1}$$

Orientation filter

We highlighted presence of noise is the semantic labelling results, especially around tool boundary and tip regions. Erroneous detections obtained in the opposite direction of the surgical tool, for example with an orientation shifted by 180°, represent an identified side-effect from such noise. To avoid orientation shift between consecutive frames, we propose to use a second Kalman filter in orientation, illustrated by equation 8.2.

$$Kalman_{orientation} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{8.2}$$

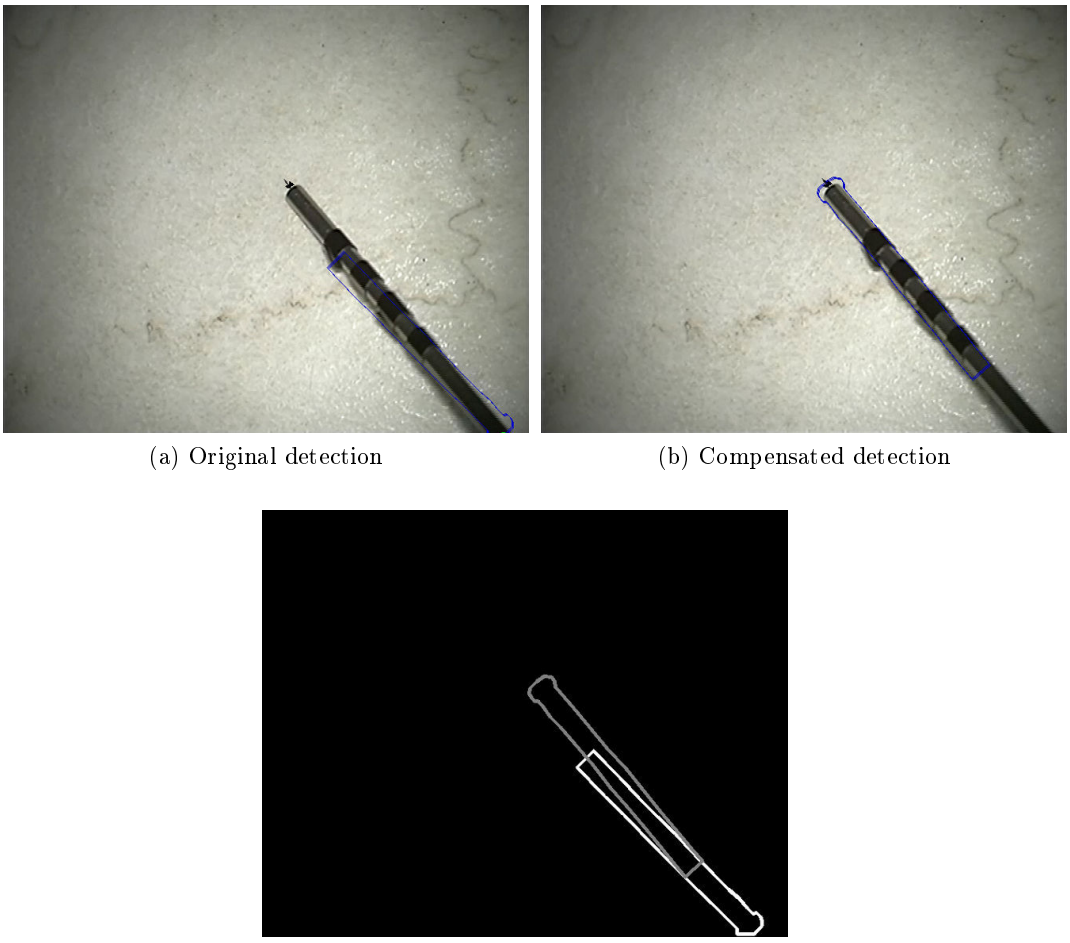
8.3.2.2 Tracks assignment

Starting with a set of initial detections resulting from the first frame processing, the corresponding set of tracks is created. Each track is associated with two Kalman filters initialized using pose parameters from its corresponding candidate detection. On following images, a matching strategy is employed to pair the set of tracks with the set of newly acquired detections. For each track, using both Kalman filters, an estimated detection location is computed , represented as a polygonal area. This estimated polygon is slightly enlarged to cover more space, as such being more permissive regarding detection inaccuracy. Ensuingly, the matching strategy between current detections and Kalman-based estimations is analogous to the validation methodology strategy. Pairing is done based on the Intersection Over Union (IOU) measurement where highest overlapping areas are favored. Ultimately, new tracks are created for unmatched detections, while tracks not having been paired are estimated using the strategy presented in the following section.

8.3.2.3 Compensation

In case of missing or erroneous detections, merging Kalman estimates is necessary to obtain new candidate detections. The new detection has the same polygonal shape and size as previous detections in the track, and is placed at the location given by the first Kalman filter, appearing under the orientation provided by the second Kalman filter.

When all tool pose parameters are correct aside from the orientation, typical situation in case of an orientation shift, the location of the detection is kept and an in-plane rotation of the polygonal shape is applied to match with the estimated orientation value from the Kalman filter. Such compensation is illustrated in figure 8.13.



(a) Original detection (b) Compensated detection

(c) Overlap

Figure 8.13: Illustration of the compensation process using Kalman filter estimates.

Track persistence

The longer the track has not been matched to a candidate detection, the more likely the track is dead and should be removed. As such, a persistence threshold is set, represented as a number of consecutive frames, after which the track is deleted if not matched.

8.3.3 Results

To evaluate the impact of adding a tracking process to the *ShapeDetector*, we do not have access to any reference. As such, performing quantitative studies is not possible and only visual results are provided.

Success modes are illustrated in figure 8.14. Tracked detections are displayed as black curves in the figure corresponding to successive tool-tip positions within the last 10 frames. Using Kalman filters enables to better handle one highlighted *ShapeDetector* issue: shift in orientation.

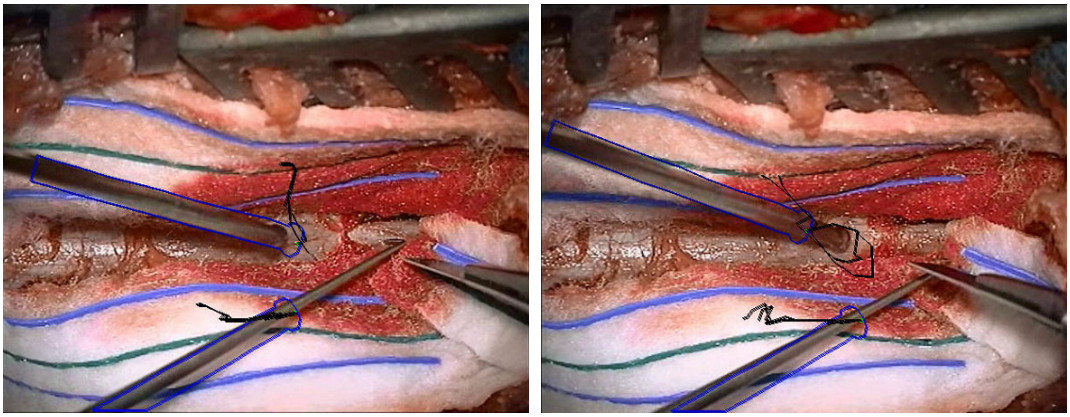


Figure 8.14: Success modes of the tracking layer on in-vivo data.

However, some unwanted side-effects also appear from the use of a tracking layer, as displayed in figure 8.15. The detection over the suction tube, on the left part of the image, is lost by the tracking system. As a result, both the previous track (in black) and a new track corresponding to the real detection (in blue) are present in the image, until the previous track is lost.

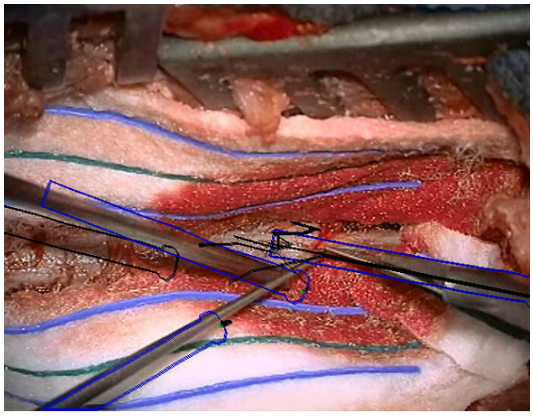


Figure 8.15: Failure mode of the tracking layer on in-vivo data.

Having surgical instrument tracks means having access to the trajectory followed by the instruments. This additional information can be useful for surgical activity recognition as it provides an access to the knowledge of the action performed, after proper motion pattern identification. In the end, we can have access to the surgical instrument position in the image, its class, and the corresponding action verb.

Chapter 9

General discussion - Conclusion

Contents

9.1 Discussion

To kick-off the last chapter, we propose a general discussion about the research performed in this thesis following the main categories introduced in the literature review in chapter 2. Many specific discussions having been provided throughout the manuscript in their corresponding chapters, we therefore try as much as possible not to be redundant.
The discussion is organized accordingly to the literature review, starting by the data acquisition process in section 9.1.1. In section 9.1.2, current detection methods and their future are discussed. Finally, in section 9.1.3 we address the validation methodology topic especially regarding medical application.

9.1.1 Data acquisition

Two-third of existing works addressing the surgical tool detection problem have been focusing on endoscopic videos in Minimally Invasive Surgical in-vivo setups while the remaining third has been exploiting surgical microscope videos in a retinal microsurgery context. Nevertheless, no official benchmark data-sets have been chosen by the community and each study has been relying on its own set of data. As such, the first contribution of the thesis was the creation of a new data-set, the *NeuroSurgicalTools* data-set, using microscope videos recorded during brain and spine tumor removal surgeries. The data-set has been put together with as few selective bias as possible to present all the makings to represent a benchmark solution, and has been made available online[1]. The data-set creation process followed pedestrian data-set creation hindsight since being a computer vision topic

[1] https://medicis.univ-rennes1.fr/software

heavily addressed in the past decade.

In this thesis work, we chose to rely on surgical microscopes as data recording devices. They present the advantage to already equip OR and using them to their full extent is a first step towards new generations of CACAI systems. In addition, information extracted from the OR in the form of videos recorded by surgical microscopes respect the constraints put forward by Bouarfa et al. [Bouarfa 2011] as they are discriminant, compact in size, easy to monitor and last but not least invariant to task distortion.
Surgical microscopes are set to observe with magnification the surgical field-of-view where instruments are manipulated over anatomical structures. In addition, video recording capabilities are available in any surgical microscope. As such, they provide an access to the best possible viewpoint to perform surgical tool detection through image-based analysis. Incoming hardware microscope updates from SD monocular camera to HD stereoscopic ones will in time provide additional resources for more robust and reliable tool detection.
Depending on surgeon skills, surgical choices and occasional adverse events, each surgical procedure is unique. Yet, when compared within a same surgical context such as spine tumor removal procedures in neurosurgery and observed from a microscope viewpoint, they appear to be reproducible. Variations in instrument appearance are visible across surgery types, especially between pituitary and spine surgeries for example. In the former, the surgical entry point is extremely narrow and instruments are highly tilted. In the latter, instruments are closer to being plane-aligned in the field-of-view. As such, features are consistent in their representation when extracted from videos showcasing procedures from a same surgical type. This invariance to task distortion is a key component for data-driven surgical tool detection approaches based on learning strategies.
Lastly, surgical microscopes do not disturb the OR setup as they are used as part of the clinical routine and the recording device is already embedded thus not requiring additional modifications. In addition, they do no require any control from surgical staff members aside from the activation or deactivation of the recording. Hence, a microscope is easy to monitor and data easy to record.

Surgical microscopes provide access to a unique viewpoint: the viewpoint of the operating surgeon. From this viewpoint, recorded videos display what the surgeon is seeing, which is the most important viewpoint providing on principle enough data to detect surgical tools used and by extension identify surgical activities. However, recording and analyzing videos from other viewpoints within the OR has been suggested. While surgical tools might not be directly identified from such viewpoints, they can provide other information complementary to the ones obtained from the standard viewpoint. For example, recording the surgeon him-/her-self can be used to retrieve body and arms posture, which could help identifying surgical tools in use relying on knowledge-based algorithms. Likewise, wide-angle cameras recording the operating theater in its entirety can be used to

monitor surgical devices being employed or the position and number of medical staff in the room. Manually adding consumer cameras in the OR prove to be troublesome as a standardized viewpoint must be found to ensure feature invariance. Moreover, issues related to occlusion and camera calibration must be addressed as medical staff could come and go in front of the camera, potentially preventing to retrieve essential information at a needed time. New solutions are starting to be commercialized to enable video recording from other viewpoints as long as answering the viewpoint standardization issue: cameras are embedded onto surgical lights or surgical glasses[2]. Adding other consumer cameras in the operating room remains a possibility, however using at their full potential already installed devices feels to be the priority.

Other global sensors could be added in order to have access to new feature types, such as Kinect devices or RFID tags placed on surgical tools. Similarly, new generation of cameras such as RGB-D cameras providing depth information or thermographic cameras can be used. Currently, such technology is costly, few hospitals are or can be equipped, and many safety regulations are not met by such sensors to be safely used within an OR.
As presented in the Introduction (see chapter 1), the medical community long-term objective comes down to the creation of surgical cockpit systems within fully digital operating rooms. In such technologically advanced environments, many different sensor types will be added with standardized and easy access. In time, a wider range of data to process will be available, complementary to current videos recorded by medical devices providing information from a unique viewpoint.
As long as data acquisition and processing comply with real-time capability, redundancy in input signals is a necessary situation for robust tool detection methods. As such, making full use of 2D/3D videos is the first stone for surgical cockpit systems, waiting for new sensor types to be efficiently integrated into OR as part of the surgical routine.

9.1.2 Detection methods

The core contribution of this thesis work was the creation of two spatial surgical tool detectors, inspired by the best state-of-the-art computer vision approaches. The first proposed detector, the *adapted SquaresChnFtrs*, belongs to the category of one-stage approaches where the pose estimation is directly performed on top of image feature channels. This detector is an adaptation of the vanilla Squares Channel Features pedestrian detector [Dollár 2009a] not compatible with surgical tool variations as extracted features such as HOG are not rotation-invariant. To deal with this issue, multi-orientation models are proposed and used similarly to multi-scale models. Instead of traditional bounding boxes to represent and display estimated tool positions, we choose to rely on a geometry tighter to instrument shape. Our proposed refined detection representation is a bounding polygon, fitting

[2] http://www.surgitel.com/surgicam

well with elongated shapes at various in-plane orientations.
The second proposed detector, the *ShapeDetector*, belongs to the category of two-stage approaches where the pose estimation is performed on top of a map of semantic labels obtained through pixel-wise classification. Surgical instruments exhibiting a similar local structure, we make use of the local classification to assign to every pixel a tool or background label. A data-driven approach through regularized SVM is used to learn specific shape-template models for each category of surgical instrument. With an exhaustive sliding-window approach, we can detect an arbitrary number of surgical instruments, at any scale or position in the image, without relying on a set of assumptions or prior knowledge. Through a specific non-maximum suppression design, we are able to detect instruments partly occluded or crossing each other.

Using 2D videos as input gives access to the minimal amount of information required to perform tool detection. All the parameters representing the instrument pose are not optimally estimated and many failure modes remain, for example due to high-tilt variations or motion blur. With the idea not to temper with an OR setup, surgical microscopes are still not used at their full extent. While many are exclusively equipped with a single camera, upgraded versions are capable of embedding two cameras thus enabling to record stereoscopic videos. Additional information can be leveraged from the processing of 3D videos such as depth maps and 3D tool shape models can be created. Identified 2D failure modes due to high tilt variations are likely to be better handled through this additional layer of information as the tilt can be quantified in 3D. Coupled to the fact that with time all surgical microscopes will be equipped with 3D technology, focus should be now directed towards tool detection from stereoscopic videos.
Nevertheless, either 2D or 3D image-based analysis are not enough to address the tool detection and classification problem, at least not under every visual appearance possibility when in-plane rotations, tilt variations, overlap, or motion blur are in play. Robust tool detection methods might not be using a single-modality input, but rather multiple modalities as a basis for feature extraction. For example, in a MIS context, internal information provided by the robotic dVSS are used either to constrain or refine image-based tool detection processes [Reiter 2012a]. Each modality on its own might not obtain high performance detection, but their combination can help overcome limitations from each.
The type of information to process and possible features to extract are bound by the type of input sensors. Even if many tool detection approaches are using the same combination of features and a similar family of learning strategy such as cascade classifiers, there is room for improvement. The impact of motion or depth fetures on a detector performance is still an uncharted territory. As of today, it is still not clear what makes a detector good as features behavior still remains to be fully understood and an increase in detector performance could come from here.

For integration in in-vivo medical applications, tool detectors must be able

to process input data in real-time, which corresponds to the frame-rate of the recording device, usually 25Hz for a microscope. For the *ShapeDetector*, we proposed two different implementations in order to increase the computational speed while keeping intact detection performance. The first fast version is relying on CPU processing power through code optimization and parallel computing on CPU registers through the use of SIMD instructions. The second fast version is making use of a GPU high effectiveness to perform parallel computing, especially relevant for image processing.
For one image at a resolution of 612×480 pixels, the initial processing time when looking for one model at one scale and 72 orientations was around $1.5\,s$. Under the same search space configuration, the *ShapeDetector* fast implementation on GPU is requiring $170\,ms$ for one image. Currently running in-between 5-8Hz on surgical microscope videos, the *ShapeDetector* is close to fast-enough for real-time use.
Detection approaches, as most of the algorithms performing some kind of image-processing, are inherently eligible for efficient parallel computing. Nevertheless, speed improvement while keeping untouched detection performance is a research topic on its own, as illustrated by the evolution from image channels to integral image channels.

9.1.3 Validation methodology

Validation studies were performed following the full-image analysis paradigm, using train and test splits of our proposed *NeuroSurgicalTools* data-set. The validation has been focusing on the following three tool pose parameters: overall position, tip location and orientation; each one through specific validation metric used over detection polygons. A set of competitive baseline methods, ranging from out-of-the box computer vision algorithms to specifically designed techniques, were used to compare performance results. The *ShapeDetector* obtained best performances regarding overall pose estimation while being tied for first place with the *adapted SquaresChnFtrs* regarding tool-tip location. Overall, with our *ShapeDetector* at 10^{-1} FPPI, we achieve a 15% miss-rate for the suction tube and a 5% miss-rate for the bipolar forceps. 50% of detected suction tubes have a tip position erroneous from less than 20 pixels and 80% have an orientation erroneous from less than 10°.
Illustrated by the stand-alone validation of the pixel-wise classification, semantic label maps are quite noisy, especially around tools boundaries. For two-stage detectors, heavily relying on semantic label maps, such noise has a negative impact regarding accurate tool-tip location.
Outside of any application task, we developed a data-driven approach for surgical tool detection from microscope videos. Even though not thoroughly validated in other surgical contexts, preliminary results in retinal microsurgery context are encouraging. Our detection approaches being as generic as possible without the use of any prior knowledge, we proposed to use a generic validation methodology to assess the estimation of the three tool pose parameters, almost identically to methodologies used for pedestrian validation.

Nevertheless, the level of expectation when examining detection performance results is not the same when performing detection validation and detection evaluation within a medical environment where patient lives are at risk. Current detection results can equally be good-enough for applications solely involving tool instances counting, or way too bad for applications requiring instrument tip location estimation at a mm spatial resolution. As such, drawing conclusions regarding our approach performance is highly versatile, and having the possibility to perform evaluation within a medical application is of interest.

Model validation strategies are straightforward with few variety, the two possible alternatives being the cross-validation and the train/test split. Obviously, it is necessary to specify the learning strategy to ensure results relevancy. Otherwise, any manual strategy could lead to results being driven by an over-fitting of the model, and not well generalized from the data. The validation methodology pertinence is intrinsically connected to the parameters selected regarding the validation metric, criterion, and the quality of the reference. Poorly performed and detailed data references preclude a precise understanding of a method strengths and weaknesses. Loose references will substantially and virtually increase results, especially in our work with tight polygonal geometry.
From highly detailed data references, the possibility emerges to perform targeted validation over specific sub-sets, for example covering instances of occluded tools or multiple overlap. Selective validation for different types of challenging conditions is of interest but requires large and diverse data-sets with carefully detailed references. Ideally, having accurate to the pixel references, annotated with semantic attributes and additional polygons in case of occlusion is what is needed. Such validation strategy is definitely necessary to develop robust detection systems able to perform well in in-vivo applications where many different situations can occur.

9.2 Conclusion

Even though becoming a richer environment filled with more and more devices and technology, OR equipment are still not used to their full extent and patient safety remains a major issue due to preventable medical errors. With the long-term objective to build safer and seamless digital operating rooms, the necessity to develop CACAI systems arisen to better integrate and standardize operating room technology. Being able to detect and recognize in real-time a surgical procedure up to its activities represents a central piece for such systems. Within the SPM terminology, a surgical activity is often minimally represented as a triple: action verb, surgical tool, and anatomical structure [Gibaud 2014]. To improve CACAI systems capabilities, an automatic and accurate detection of surgical instruments is essential. In this thesis, we proposed to focus on automatic and real-time surgical tool detection from 2D surgical microscope videos.

Using surgical microscope videos as data input for this work is extremely interesting because presenting many advantages. At first, data can be acquired easily without altering the clinical routine as surgical microscopes are already part of operating room standard setup. Secondly, recording videos from a microscope is a very simple and standardized process. Acquired data provide diverse and robust information necessary to tackle the tool detection problem either through 2D or 3D feature extraction.

We feel like pushing the limits of data extraction from current recording devices remains an optimal solution as there is still room for improvement in detection performance. Not all the feature representations, machine learning strategies, and various combinations of all of them have been investigated. Fully validating methods performing data extraction from existing surgical devices is a first keystone. As such, a clear assessment of what is working, how it is working and what kind of information are needed can help designing new recording systems. At the same time, the design of new devices, as long as some experimental studies be carried out in the OR to prepare what could be the next generation of sensors. However, tool detection methods should wait before trying to leverage data from unorthodox sensors as they are cumbersome to add in the OR, and still do not provide a standardized point of view nor a standardized access to data.

The second major obstacle for efficient comparison and ranking between surgical tool detection approaches, and to enable an healthy competition within the community is the lack of standardized validation methodologies and their corresponding benchmark data-sets. Provided with an easy access to both components, identifying strengths and weaknesses for each method within a specific surgical context or towards the detection of a specific surgical tool could be made possible. For even better performance result bench-marking, the exact same validation code should be employed and if possible all the methods ran from a same person/computer, as it is done for pedestrian detection [Dollár 2011]. In this optic, we already made available our created data-set as a possibility for other people to re-use it.

Bench-marking tool detection methods in a computer vision way is a good solution to perform validation and identify method flaws. Nevertheless, we are working on an application field where the end-goal is to integrate such algorithms into OR systems. As such, evaluating the added-value of tool detection techniques within CAS systems, either per-operatively or post-operatively, is of high importance. Having an evaluation objective would also help to design evaluation protocols or metrics to match with what is really necessary in the medical reality. Even if not for prevalent CAS systems, performing side-by-side validation and evaluation is paramount to faster achieve real-life usefulness for tool detection techniques.

Appendix A

CPU code profiling applications

In this appendix, we introduce different CPU code profiling software necessary to identify bottlenecks in CPU-based applications. In section A.1, we start by presenting the most powerful yet paying code profiling tool, then in sections A.2 and A.3 we suggest free alternatives.

A.1 Intel VTunes Amplifier

Intel VTunes Amplifier is a commercial application for software performance analysis for 32 and 64-bit x86 based machines, with both command line and GUI interfaces. It is available for both Linux and Microsoft Windows operating systems at `https://software.intel.com/en-us/intel-vtune-amplifier-xe`.
VTune Amplifier assists in various kinds of code profiling including stack sampling, thread profiling and hardware event sampling. The profiler result consists of details such as time spent in each sub routine which can be drilled down to the instruction level. The time taken by the instructions are indicative of any stalls in the pipeline during instruction execution. The tool can be also used to analyze thread performance.
In the following, we highlight Intel Vtunes use through multiple illustrations.

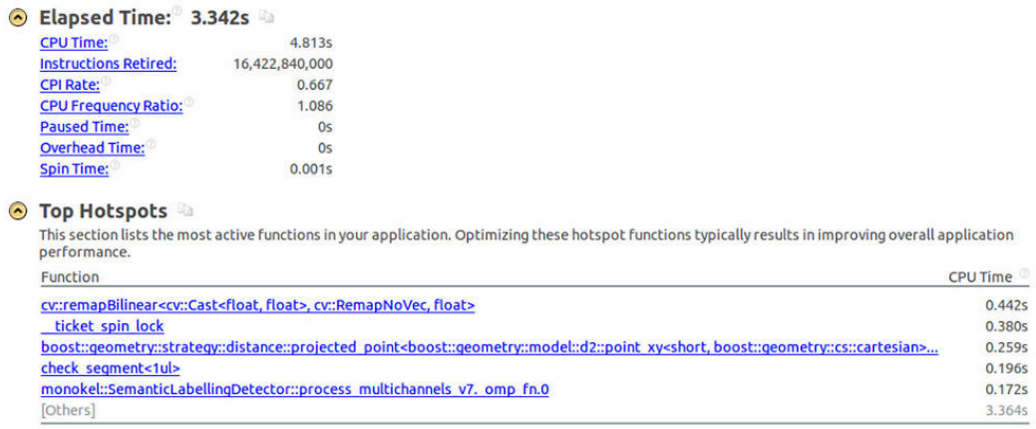


Figure A.1: Overall result view from Intel VTunes profiling on the *ShapeDetector* application.

Function / Call Stack	CPU Time by Utilization (Idle Poor Ok Ideal Over)	Instructions Retired	Overh... Ov..	Overh... Spi...	CPI Rate	CPU Freque...
cv::remapBilinear<cv::Cast<float, float>, cv::RemapN	441.695ms	2,438,100,000	0ms	0ms	0.415	1.094
__ticket_spin_lock	380.492ms	93,450,000	0ms	0ms	9.294	1.090
boost::geometry::strategy::distance::projected_poin	258.849ms	1,220,520,000	0ms	0ms	0.495	1.115
check_segment<1ul>	195.545ms	1,543,920,000	0ms	0ms	0.285	1.073
monokel::SemanticLabellingDetector::process_multi	172.343ms	533,190,000	0ms	0ms	0.718	1.061
boost::geometry::detail::distance::point_to_range<b	158.116ms	605,010,000	0ms	0ms	0.575	1.051
func@0x9370	110.471ms	69,090,000	0ms	0ms	3.617	1.080
__memcpy_ssse3_back	110.185ms	135,870,000	0ms	0ms	1.872	1.102
func@0x8fe0	107.893ms	67,410,000	0ms	0ms	3.626	1.082
Mean<float>	89.370ms	376,740,000	0ms	0ms	0.545	1.096
Mean<float>	88.988ms	376,320,000	0ms	0ms	0.532	1.074
boost::geometry::math::detail::equals<double, (bool)	76.289ms	262,500,000	0ms	0ms	0.688	1.130
advance<long int>	68.651ms	83,790,000	0ms	0ms	1.890	1.101
_mm256_hadd_ps	67.219ms	152,880,000	0ms	0ms	0.900	0.977
cv::warpAffineInvoker::operator()	64.354ms	406,140,000	0ms	0ms	0.346	1.044
_mm256_add_ps	64.259ms	174,720,000	0ms	0ms	0.768	0.997
func@0x9510	61.394ms	40,740,000	0ms	0ms	3.407	1.079
access<const monokel::v4di*&, const monokel::v4di*	51.464ms	109,830,000	0ms	0ms	1.052	1.071
advance<long int>	44.017ms	97,020,000	0ms	0ms	1.108	1.166
__do_page_fault	43.253ms	9,660,000	0ms	0ms	9.587	1.022
cv::Mat::Mat	42.394ms	123,270,000	0ms	0ms	0.831	1.154
boost::geometry::strategy::distance::pythagoras<bo	41.152ms	216,930,000	0ms	0ms	0.413	1.040
boost::geometry::detail::dot_product_maker<boost:	41.057ms	238,350,000	0ms	0ms	0.337	0.933
boost::geometry::detail::within::point_in_ring<boost:	40.675ms	305,340,000	0ms	0ms	0.324	1.161

Figure A.2: Extended result view from Intel VTunes profiling on the *ShapeDetector* application.

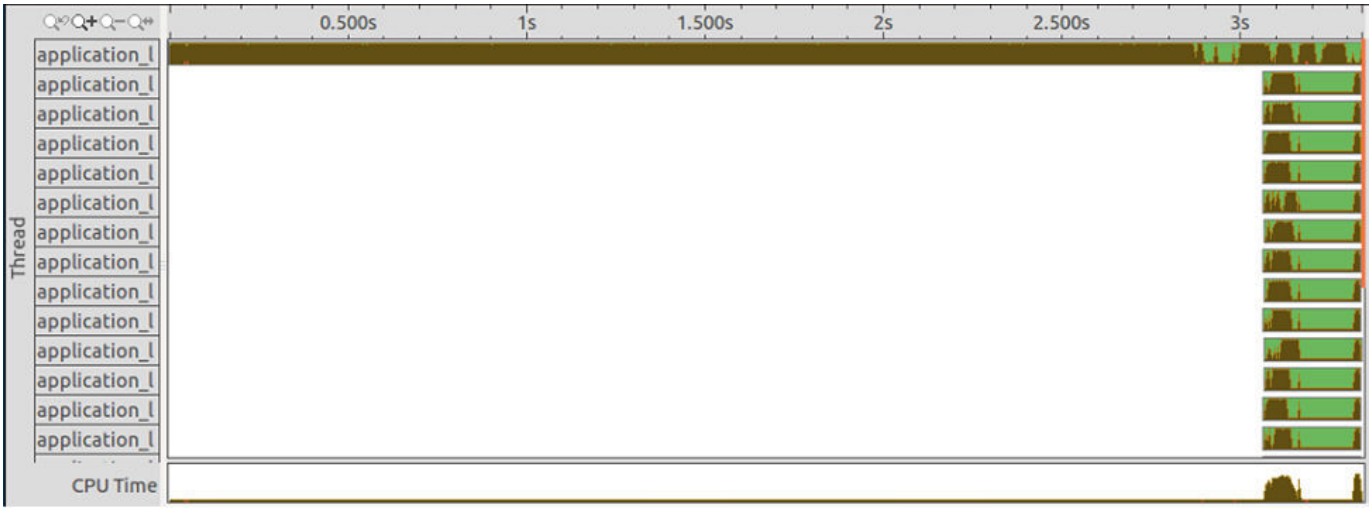


Figure A.3: Intel VTunes analysis of thread performance.

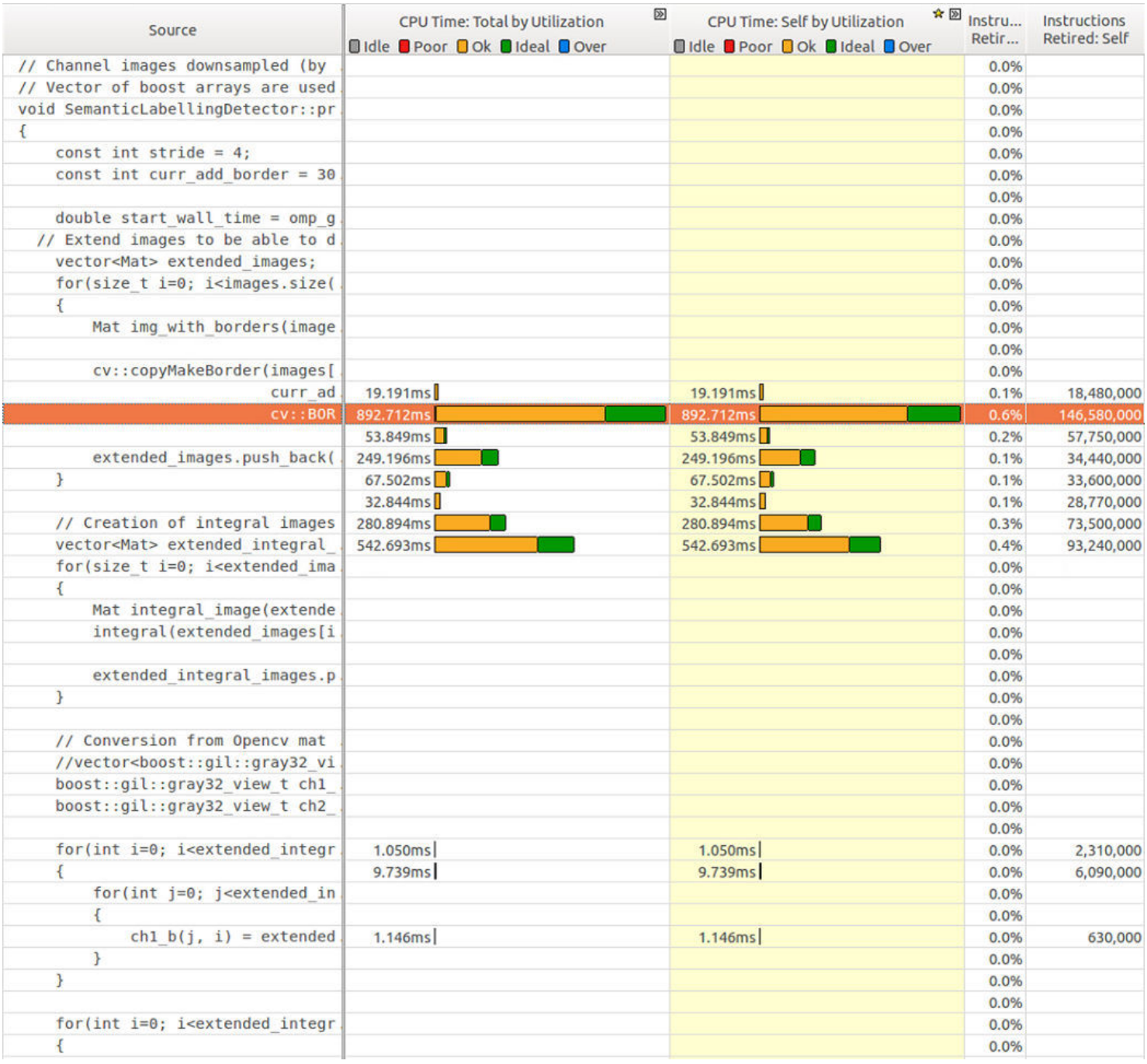


Figure A.4: Line per line result view from Intel VTunes profiling on the *ShapeDetector* application.

A.2 Valgrind and KCacheGrind

Valgrind is a programming tool for memory debugging, memory leak detection, and profiling. Valgrind was originally designed to be a free memory debugging tool for

Linux on x86, but has since evolved to become a generic framework for creating dynamic analysis tools such as checkers and profilers.
Part of the Valgrind toolchain, Callgrind is used for profiling, the application being transformed in an intermediate language and then ran in a virtual processor emulated by Valgrind. Thanks to a huge run-time overhead, the precision is really good and the data profiling is complete. An application running in Callgrind can be 10 to 50 times slower than normally. The output of Callgrind is not usable directly and KCachegrind must be used to display the informations about the profiling of the analyzed application, as shown figure A.5.

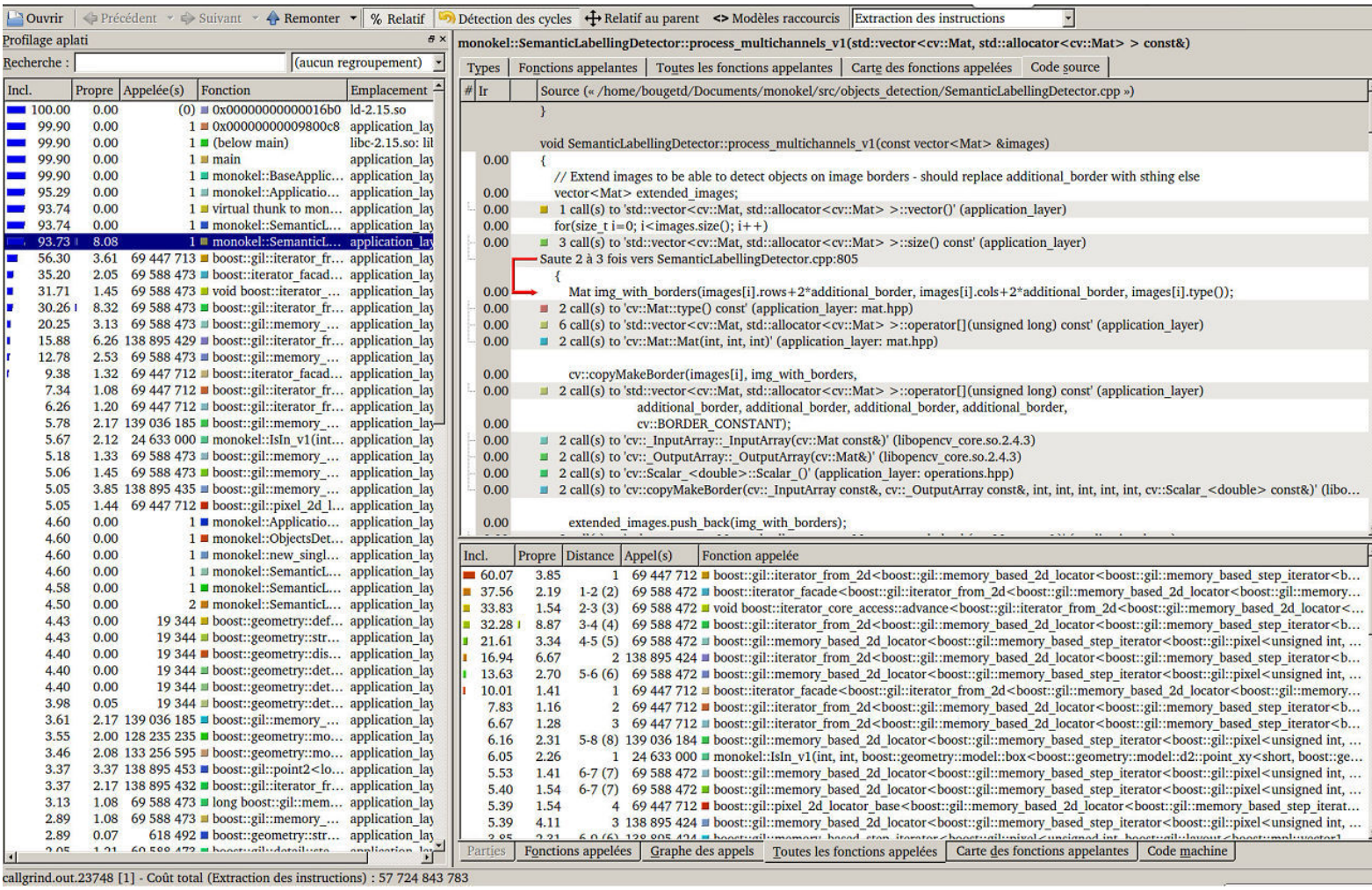


Figure A.5: *ShapeDetector* application profiling results using Callgrind (i.e. Valgrind) and displayed using KCacheGrind.

A.3 GPerfTools and KCacheGrind

GPerfTools, standing for Google Performance Tools, are offering a CPU profiler, a fast thread aware malloc implementation, a memory leak detector and a heap profiler. The GPerfTools profiler can profile multi-threaded applications. The run time overhead while profiling is very low and the applications run at ânative speedâ. KCachegrind can be used for analyzing the profiling data after converting it to a cachegrind compatible format.

Appendix B

LabelMe: data annotation tool

In this appendix, we present the LabelMe web-browser tool used to perform the data annotation process. LabelMe is a project created by the MIT Computer Science and Artificial Intelligence Laboratory (CSAIL), accessible at `http://labelme.csail.mit.edu/Release3.0/`.
The tool can be accessed anonymously or by logging in to a free account. To access the tool, users must have a compatible web browser with javascript support. To start performing the annotations, a new collection must be created and images added (.jpg format), 20 at a time maximum (as illustrated in figure B.1).

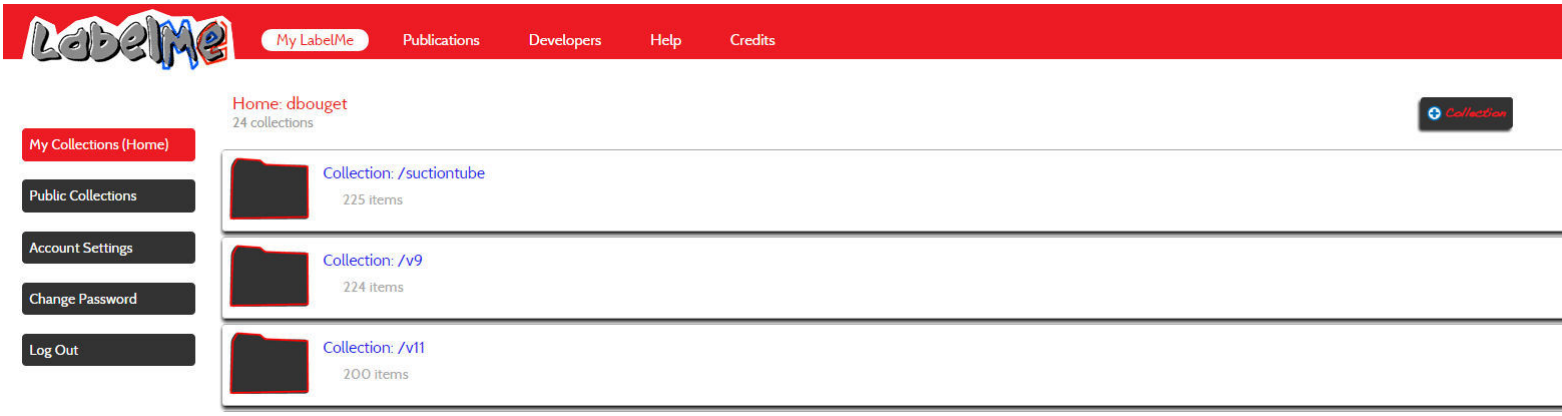


Figure B.1: LabelMe interface displaying collections created.

To start annotating, the user only has to click anywhere on the border of any object, and continue clicking along the outside edge until returning to the starting point. Once the polygon is closed, a bubble pops up on the screen which allows the user to enter a label for the object. The user can choose whatever label the user thinks best describes the object. The annotation step is illustrated in figure B.2.
Once every image of a collection has been processed, all the annotations can be downloaded in .xml files (one file per image).

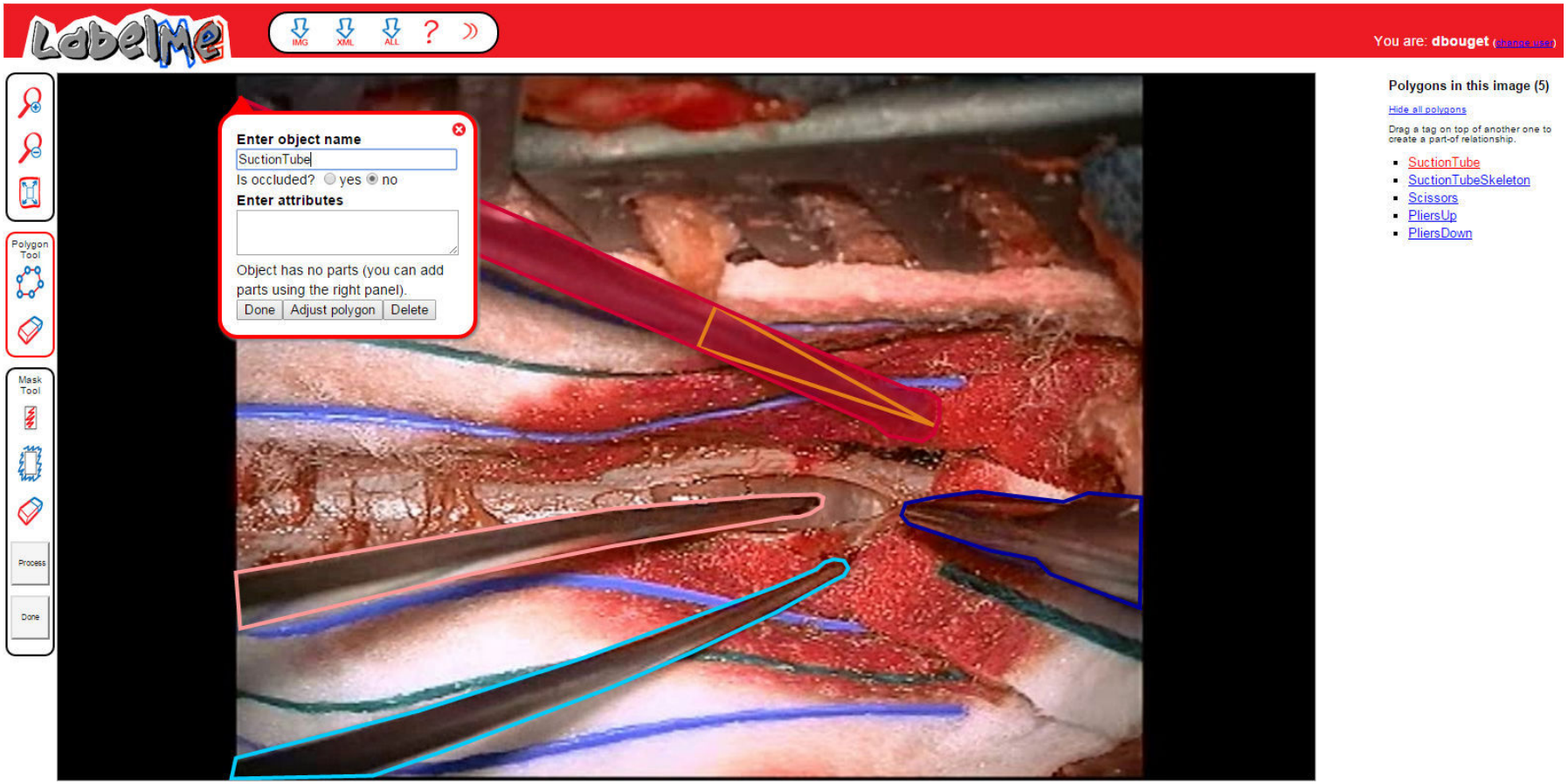


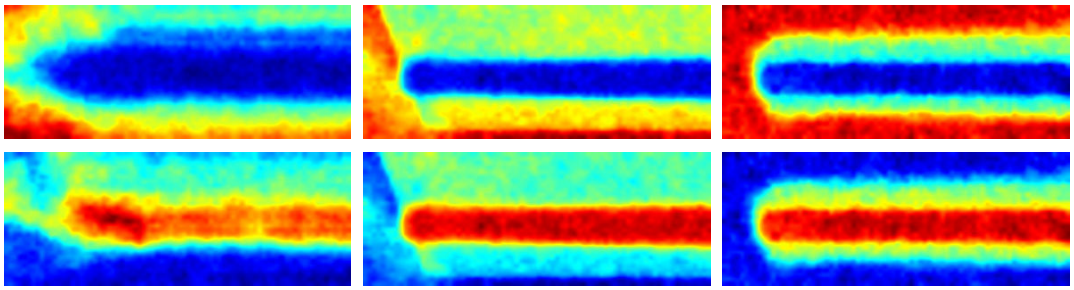
Figure B.2: Annotation process within the LabelMe interface.

Appendix C

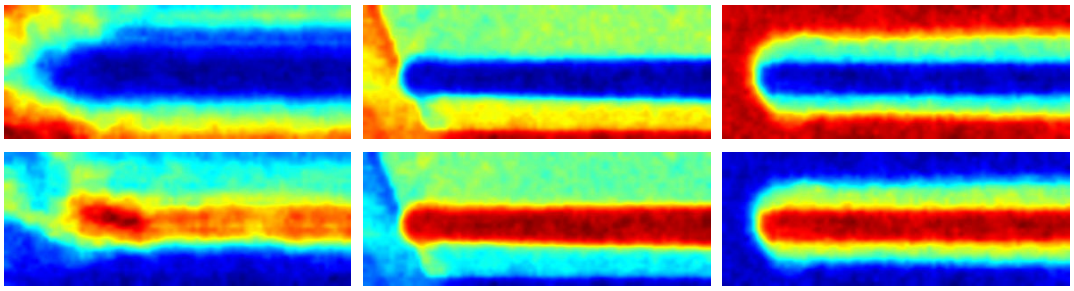
SVM model illustrations

In this appendix, we display side-by-side SVM models obtained through different configurations of the design space. In section C.1, we report models for the suction tube category and the bipolar forceps is illustrated in section C.2. Figures always display the background class on odd rows with the tool class being on even rows.

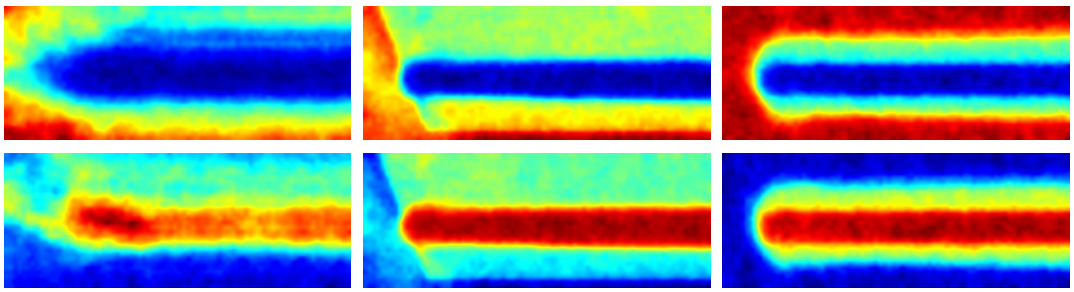
C.1 Suction tube



(e) Binary negative

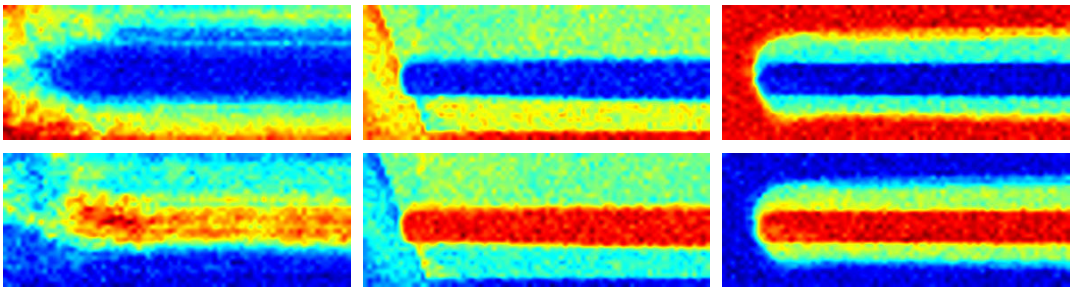


(k) Uniform negative

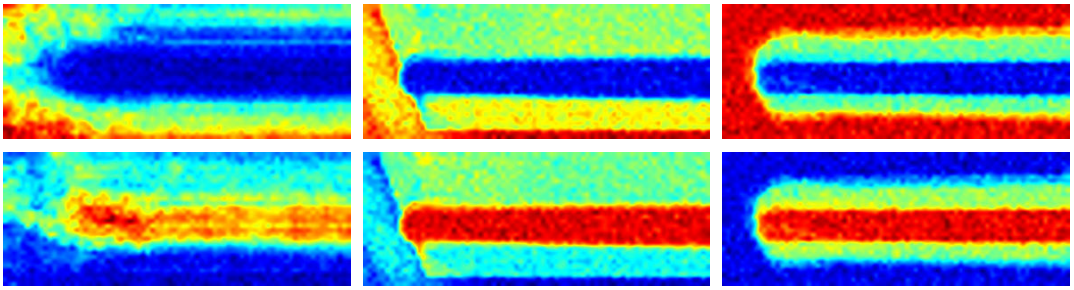


(q) Gaussian negative

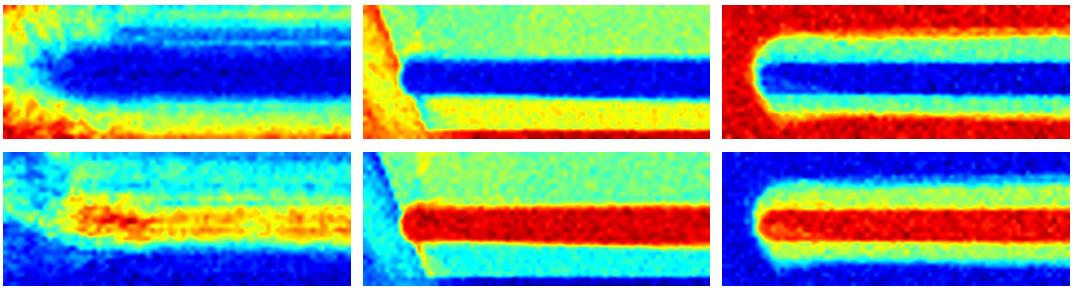
Figure C.1: Suction tube SVM models for various training configurations, with the use of the SVM spatial regularization term.



(e) Binary negative



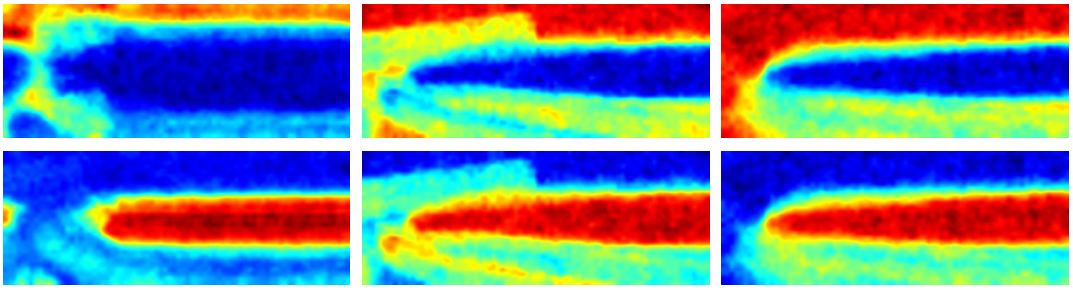
(k) Uniform negative



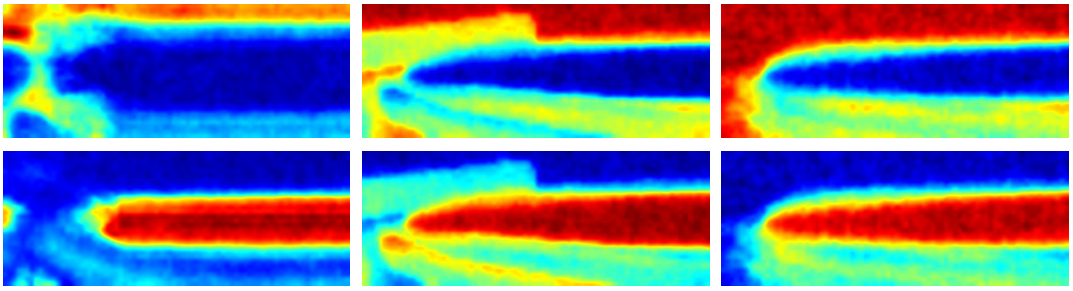
(q) Gaussian negative

Figure C.2: Suction tube SVM models for various training configurations, without the use of the SVM spatial regularization term.

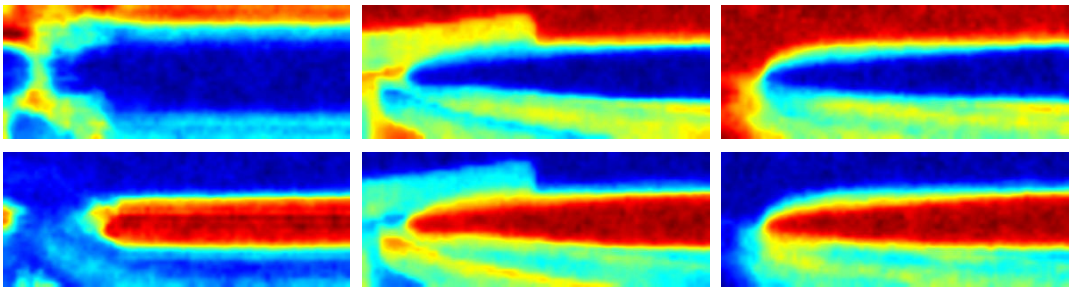
C.2 Bipolar forceps



(e) Binary negative

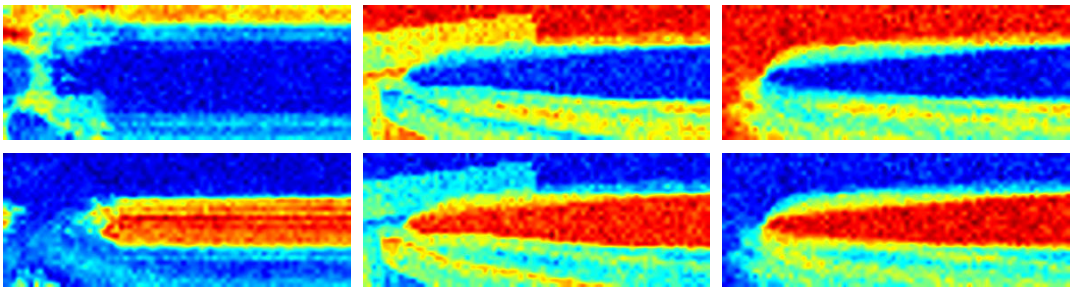


(k) Uniform negative

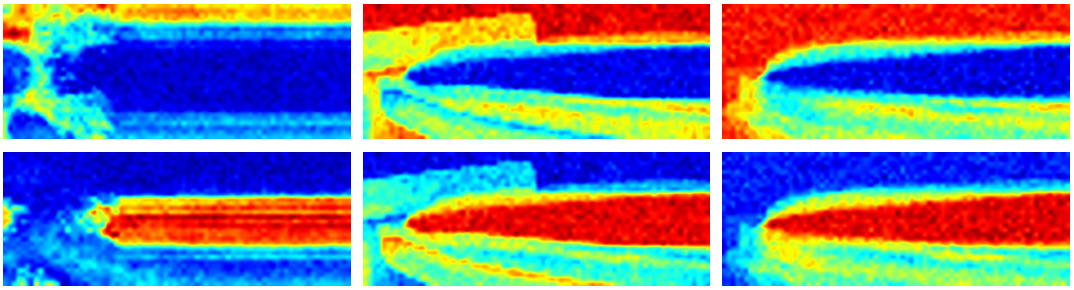


(q) Gaussian negative

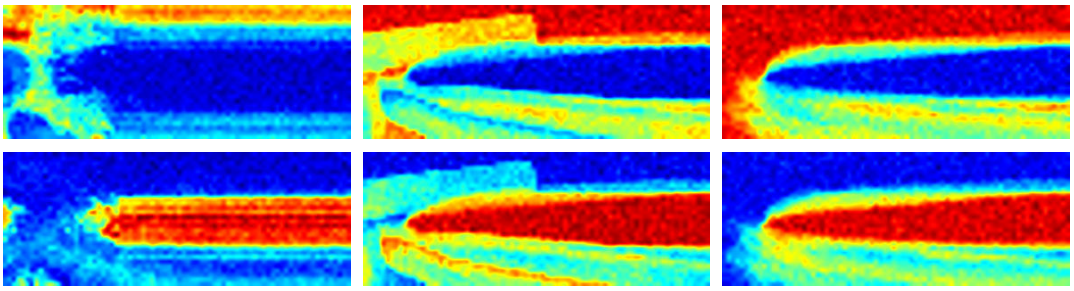
Figure C.3: Bipolar forceps SVM models for various training configurations, with the use of the SVM spatial regularization term.



(e) Binary negative



(k) Uniform negative



(q) Gaussian negative

Figure C.4: Bipolar forceps SVM models for various training configurations, without the use of the SVM spatial regularization term.

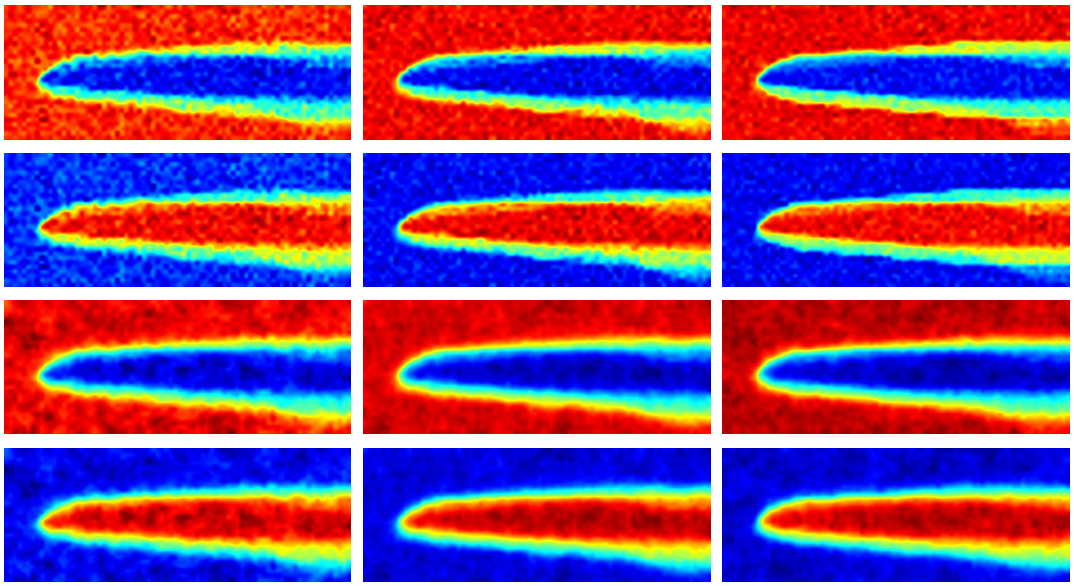


Figure C.5: Upper part of the bipolar forceps SVM models for various training configurations.

Appendix D

Résumé étendu de la thèse

Contents

D.1 Introduction

D.1.1 Contexte

En dépit d'avancées majeures concernant les technologies du quotidien et les progrès des techniques chirurgicales comme la chirurgie minimalement invasive (MIS), la salle d'opération fait toujours face à de nombreux challenges. En 2004, un groupe de travail s'est réuni autour de la thématique suivante: "La salle d'opération du futur, à l'horizon 2020.". L'objectif ciblé était l'identification des challenges actuels afin de concevoir les caractéristiques générales de la salle d'opération du futur, dans le but d'améliorer la prise en charge des patients.
Parmi les nombreux challenges identifiés se trouvent: un nombre toujours conséquent d'erreurs chirurgicales évitables [Kohn 2000], un nombre de personnels soignants toujours plus en baisse entrainant des praticiens surchargés de travail ou moins qualifiés (moins de temps de formation), une gestion non optimisé des blocs opératoires et des ressources médicales, et enfin une difficulté d'intégration et de standardisation des nouvelles technologies au sein du bloc opératoire. Une réponse à la plupart des challenges peut être apportée grâce au développement et à l'intégration de systèmes technologiques avancés et intelligents.
La conception de la salle d'opération du futur afin d'y intégrer des systèmes intelligents doit être à même de répondre aux problèmes de complexité liés à l'utilisation des nouvelles technologies. Cette salle d'opération devra ressembler à un cockpit chirurgical avec une intégration digitale complète et une utilisation efficace de la technologie, comme par exemple OR1[1]. De la même manière qu'un cockpit d'avion, ce cockpit chirurgical serait équipé d'un système de boite noire permettant d'analyser a posteriori les erreurs médicales afin d'en réduire leurs nombres. Pendant les opérations, il pourrait assister le chirurgien dans la réalisation de tâches chirurgicales robotisées, fournir des conseils à l'équipe soignante, ou émettre des alertes quand la chirurgie dévie de la procédure standard grâce à un raisonneur intégré.
Cette capacité de traitement, d'analyse et de compréhension des évènements se produisant au sein de la salle opératoire est primordiale. C'est dans ce but qu'est née la Modélisation des Processus Chirurgicaux permettant d'analyser la pratique chirurgicale via une terminologie unifiée [Lalys 2014], en se focalisant très souvent sur le chirurgien principal. Une procédure chirurgicale peut être décrite par plusieurs niveaux de granularité, partant d'une découpe en temps chirurgicaux majeurs et se

[1] https://www.karlstorz.com/de/en/karl-storz-or1.htm

terminant en une découpe très fine en gestes élémentaires. A mi-chemin sur l'échelle de granularité se trouve la représentation en activités chirurgicales; une activité étant synthétiquement représentée par un triplet: verbe d'action, outil chirurgical, et structure anatomique [Lalys 2013].
De nombreuses approches basées sur l'analyse d'image se sont attaquées à la reconnaissance automatique des activités. Pour cet objectif, être capable d'identifier les outils chirurgicaux est primordial car donnant accès à deux des trois éléments du triplet: l'outil chirurgical, et le verbe d'action provenant de l'analyse de trajectoire de l'outil.

D.1.2 État de l'art sur la détection des outils chirurgicaux

Une revue de la littérature a été effectuée concernant les méthodes de détection des outils chirurgicaux basées sur de l'analyse d'image. Les méthodes utilisant des marqueurs externes n'ont pas été incluses. Trois catégories sont proposées afin de décrire les travaux existants: les jeux de données de validation, les méthodes de détection, et les méthodologies de validation.

D.1.2.1 Jeux de données de validation

Aucun jeu de données de référence, ou benchmark, n'a été recensé et chaque étude s'est appuyée sur son propre jeu de données. La plupart des travaux se sont basés sur des données provenant de chirurgies minimalement invasives comme la cholécystectomie [McKenna 2005], ou l'hystérectomie [Kumar 2013b]. La deuxième spécialité chirurgicale la plus étudiée est l'ophtalmologie avec des microchirurgies de l'oeil [Pezzementi 2009, Sznitman 2012]. Un seul travail s'est porté sur des images provenant d'un contexte de neurochirurgie [Sznitman 2014].
Les annotations de la position des outils dans les images composant ces jeux de données ont souvent été du type boites englobantes [Speidel 2006, Sznitman 2012], ou du type point unique sur la position de l'extrémité des outils [Voros 2007, Haase 2013].

D.1.2.2 Méthodes de détection

Deux grandes familles de méthodes existent permettant d'effectuer la détection des outils chirurgicaux en temps-réel en utilisant des techniques d'analyse d'image. Les méthodes purement spatiales effectuant une analyse complète de l'image, et les méthodes réutilisant la position des outils détectés dans les images précédentes (type "tracking"). Les deux approches nécessitent une analyse de l'image entière, dans le but unique d'initialiser ou réinitialiser la procédure de tracking dans le second cas. Concernant les méthodes analysant l'image entière, trois classes se distinguent: les méthodes ad-hoc, les méthodes à un niveau, et les méthodes à deux niveaux.
Les méthodes ad-hoc reposent sur l'utilisation d'opérations de morphologie mathématique [Voros 2007], de seuillage, ou de la transformée de Hough [Doignon 2004].
Les méthodes à un niveau effectuent la détection de la position des outils directement à partir des caractéristiques extraites de l'image [Kumar 2013b]. Pour finir,

les méthodes à deux niveaux effectuent d'abord une première passe afin d'identifier les pixels appartenant aux outils et ceux appartenant à l'arrière-plan, pour ensuite effectuer l'estimation de la pose des outils sur le résultat de la première passe [Pezzementi 2009].
Afin de faciliter la tâche de détection, la plupart des méthodes font usage de contraintes sur l'espace de recherche. Ces contraintes sont représentées par un ensemble d'hypothèses soit par rapport à la forme de l'outil à identifier (une forme tubulaire [Speidel 2008]), soit concernant la position de l'outil dans les images (visible sur les bords [McKenna 2005]).

D.1.2.3 Méthodologies de validation

Au même titre que pour les jeux de données, aucune méthodologie de validation de référence n'existe. Un nombre conséquent de travaux se sont intéressés à la quantification de l'estimation de la position de l'extrémité des outils [Sznitman 2012]. Les autres paramètres de pose des outils validés ont été l'orientation [Wolf 2011], et la position globale de l'outil [Kumar 2013b].
Dans les cas de validation de l'orientation ou de la position de l'extrémité des outils, une simple distance Euclidienne est utilisée. Dans les cas de validation de la pose globale des outils, le critère d'Intersection Sur l'Union est pris en compte (mesure de PASCAL [Everingham 2010]).

D.1.2.4 Conclusion

La conclusion frappante que l'on peut tirer de cette revue de la littérature est le manque cruel de jeux de données et de méthodologie de référence pour valider les méthodes de détection d'outils chirurgicaux. A ce titre, il est difficile d'émettre un jugement concernant les méthodes existantes car ne pouvant être comparées et classées sur un pied d'égalité. Concernant les méthodes d'analyse en elles-mêmes, plusieurs types ont été étudiés chacun présentant avantages et inconvénients.

D.1.3 Problématique de la thèse

Devant l'intérêt croissant des études sur la détection des outils chirurgicaux dans des images/vidéos, cette thèse s'est focalisée sur quatre principaux aspects. Tout d'abord, les données de neurochirurgie étant peu ou pas utilisées, et devant la nécessité de créer des jeux de données robustes, le premier objectif a été la création d'un nouveau jeu de données. Deuxièmement, le coeur de cette thèse s'est porté sur la création de méthodes de détection des outils chirurgicaux dans des images en 2 dimensions sans utilisation de connaissances a priori. Ensuite, afin de pouvoir intégrer ce type de méthode dans un système opérationnel, nous nous sommes intéressés à l'optimisation de la vitesse de calcul. Pour finir, nous avons validé nos méthodes par rapport à des méthodes de référence, selon trois critères d'estimation de pose des outils: la position globale, la position de l'extrémité de l'outil et l'orientation de l'outil.

D.2 Données de validation

Les données utilisées dans cette thèse ont été acquises au sein du département de Neurochirurgie du CHU Pontchaillou à Rennes. Deux types de chirurgies ont plus particulièrement été privilégiées: les chirurgies de résection de tumeurs cérébrales et de tumeurs rachidiennes.
A partir des données à disposition, un sous-échantillon de 14 vidéos a été manuellement défini, représentatif des deux types de chirurgies et faisant apparaitre sept catégories d'instruments chirurgicaux. Notre jeu de données définitif, le *NeuroSurgicalTools*, est composé de 2476 images de résolution 612×460, extraites aléatoirement dans les 14 vidéos précédemment évoquées. La figure D.1 illustre différentes images se trouvant dans ce jeu de données, mis en accès libre[2]. Plusieurs conditions pouvant représenter un challenge pour la détection sont visibles: occlusions des outils, outils se chevauchant, illumination changeante, et flou entrainé par des mouvements rapides.

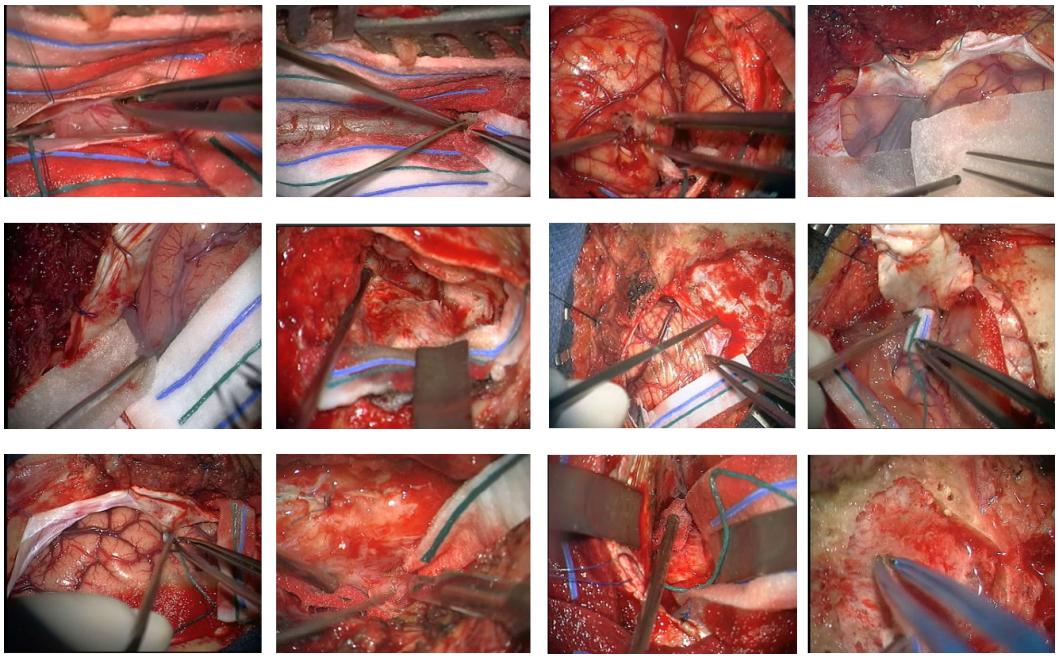


Figure D.1: Illustration des images composants notre *NeuroSurgicalToolsDataset*.

Afin d'obtenir les positions de référence des outils chirurgicaux dans ces images, un processus d'annotation a été suivi. Chaque outil a été délimité par au moins un polygone, ou plusieurs dans le cas d'instruments en plusieurs parties comme le forceps bipolaire. De plus, afin d'avoir accès aux informations d'orientation, de largeur et de position précise de l'extrémité de l'outil, un triangle isocèle a été ajouté pour les deux outils principaux.

[2] https://www.medicis.univ-rennes1.fr/software

D.3 Méthodes de détection

Dans cette thèse, deux approches différentes ont été proposées permettant de détecter des outils chirurgicaux dans des images 2D. Tout d'abord, le *SquaresChnFtrs adapté*, basé sur une des méthodes de détection de piétons les plus performantes de la littérature. Notre deuxième méthode, le *ShapeDetector*, est une approche à deux niveaux n'utilisant aucune contrainte ou hypothèse a priori sur le nombre d'outils, leurs positions, ou leurs formes afin de faciliter la détection.

D.3.1 Première approche: *SquaresChnFtrs adapté*

Cette première méthode se base sur le détecteur de piéton du même nom: le *SquaresChnFtrs*. Pendant l'entrainement, des caractéristiques image (features) sont extraites et utilisées pour la création d'un modèle de l'objet à détecter. Pendant la phase de test, le modèle crée est appliqué en chaque pixel de l'image en suivant une stratégie de fenêtre glissante (sliding window), suivi d'une passe de sélection des meilleures détections possible (NMS). Afin de détecter des outils chirurgicaux, pouvant apparaitre à diverse orientations, le détecteur *SquaresChnFtrs* a du être adapté.

Représentation des caractéristiques image Trois types de caractéristiques sont extraites de l'image, pour une représentation finale selon 10 canaux: 3 canaux de couleur dans l'espace LUV, 1 canal de magnitude des gradients, et 6 canaux d'histogrammes de gradient (HOG). De plus, chaque canal est représenté de manière intégrale, permettant ainsi de calculer la valeur des caractéristiques image dans une région rectangulaire indéfinie en seulement trois opérations élémentaires [Viola 2001].

Apprentissage du modèle d'objet La stratégie employée pour apprendre un modèle repose sur un classifieur en cascade: la forêt d'arbres décisionnels. Un ensemble de 2000 classifieurs faibles a été assemblé en utilisant Adaboost afin de créer un classifieur fort [Benenson 2013].

Apprentissage d'un modèle d'outil chirurgical Afin de créer un modèle d'outil chirurgical, les images d'entrainements doivent être compensées en position, taille, et en orientation dans le cas présent. Nous définissons l'orientation 0° comme représentant un outil aligné horizontalement et faisant face au bord gauche de l'image, comme illustré sur l'image D.2.

Un modèle est crée pour chaque 5° d'orientation, le tout assemblé dans un super modèle permettant de détecter un outil chirurgical à n'importe quelle orientation dans l'image avec une précision minimale de 5° dans l'estimation de l'orientation. De plus, pour gagner en précision dans l'estimation globale de la pose de l'outil, nous proposons l'utilisation de polygones englobants plutôt que de boites englobantes pour représenter les détections.

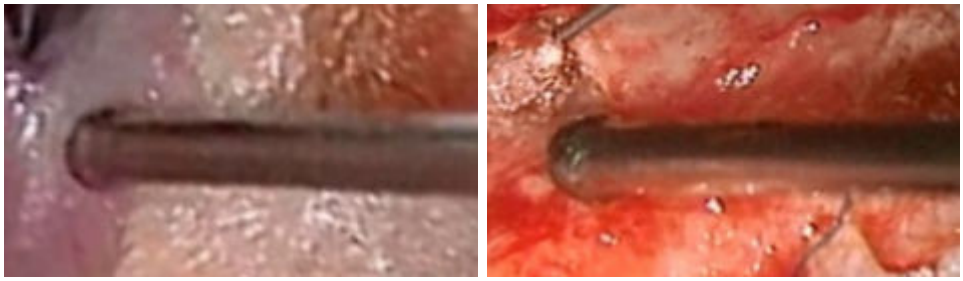


Figure D.2: Tubes d'aspirations apparaissant à l'orientation 0°.

D.3.2 Deuxième approche: *ShapeDetector*

Adapter une méthode de l'état de l'art dédiée à la détection de piétons a permis d'obtenir des résultats de détection intéressants (voir section D.6), cependant il est possible de faire mieux. Notre deuxième méthode suit le schéma des méthodes à deux niveaux, et permet l'utilisation d'informations de contexte pour améliorer la qualité des détections, comme illustré figure D.3.

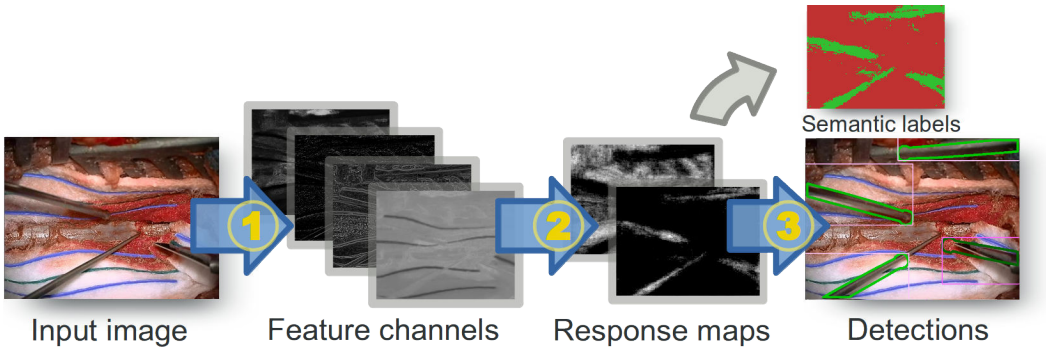


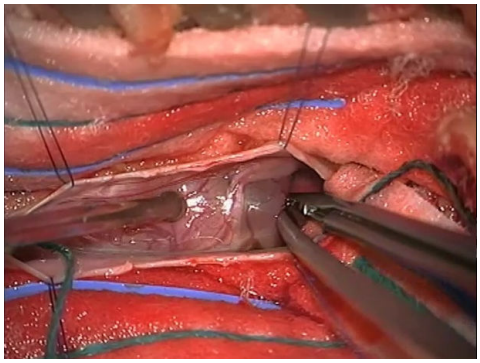
Figure D.3: Schéma décrivant le processus de notre *ShapeDetector*. Etape 1: calcul des canaux image sous forme intégrale. Etape 2: classification de chaque pixel de l'image en deux classes: outil et arrière-plan. Etape 3: estimation de la pose des outils à partir d'un modèle SVM.

D.3.2.1 Premier niveau: labélisation semantique

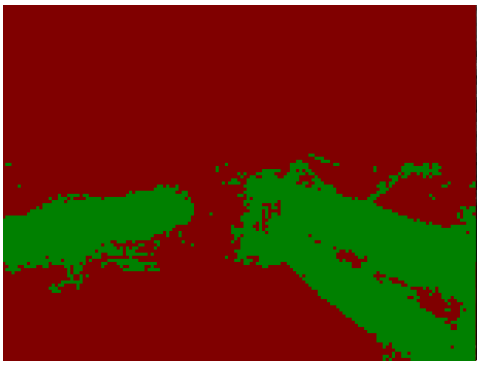
L'objectif de ce premier niveau est de traiter l'image afin d'en obtenir une carte affichant les pixels de l'image pouvant appartenir aux objets. La *SquaresChnFtrs* est réutilisé afin d'effectuer non-pas une décision sur un modèle global de l'outil mais en chaque pixel de l'image. Deux classifieurs différents sont appris, un pour modéliser les pixels appartenant à la classe outil, et un second pour modéliser les pixels appartenant à la classe arrière-plan.

Formalisme du résultat Le résultat de l'étape de labélisation sémantique est un ensemble de cartes de scores, une pour chaque classifieur. Pour faire référence à cette

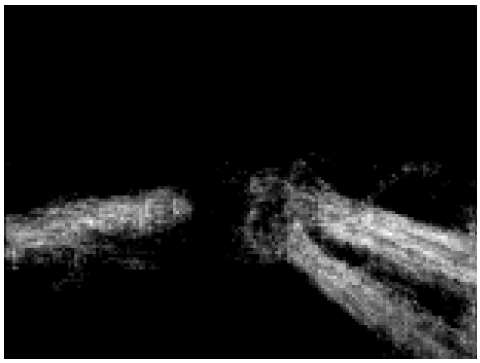
sortie multi-classes, nous utilisons le terme de scores sémantiques (cf. images D.4c et D.4d). Nous proposons aussi de fusionner cet ensemble de cartes en une image de labels. Pour ce faire, le label associé à chaque pixel est celui de la classe ayant le score sémantique le plus élevé. Nous utilisons le terme de labels sémantiques pour faire référence à cette sortie (cf. image D.4b).



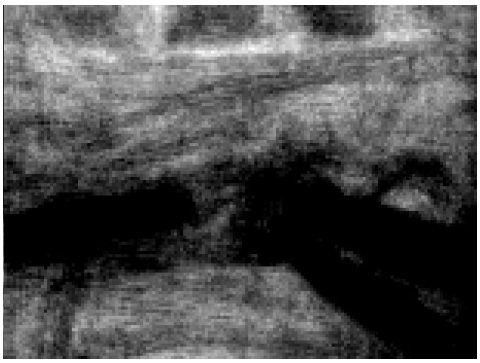
(a) Image d'entrée



(b) Labels sémantiques



(c) Scores sémantiques pour la classe outil



(d) Scores sémantiques pour la classe arrière-plan

Figure D.4: Illustration des scores et labels sémantiques.

D.3.2.2 Modèle d'outil SVM

Une machine à vecteurs de support (SVM) a été utilisée afin d'apprendre un modèle d'outil chirurgical. De plus, nous ajoutons un terme de régularisation spatiale permettant d'assurer une cohérence spatiale au sein du modèle. A partir de l'équation standard d'un SVM (équation (D.1) (cf. [Burges 1998]), nous ajoutons le terme de régularisation M (équation D.2) permettant d'appliquer notre lissage spatial en 2d [Lehmann 2011].

$$w^T \cdot w + C \cdot \sum_i L\left(y_i,\, x_i \cdot w\right) \tag{D.1}$$

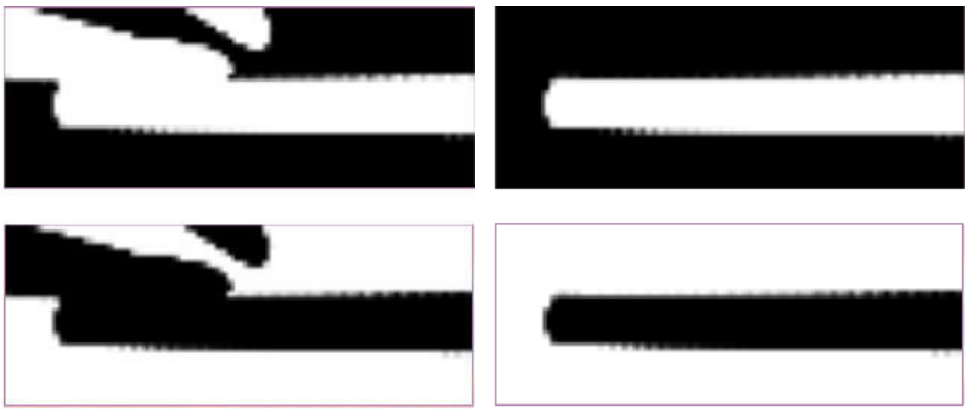
$$w^T \cdot M \cdot w + C \cdot \sum_i L\left(y_i,\, x_i \cdot w\right) \tag{D.2}$$

où $(x_i,\, y_i)$ sont des paires du type instance-label, $L\left(y_i,\, x_i \cdot w\right)$ est la fonction de coût et C est le paramètre de pénalité. La matrice M peut être décomposée comme représenté dans l'équation 4.5. La matrice de régularisation R encode la structure spatiale 2D.

$$M = R^T \cdot R \tag{D.3}$$

Echantillons d'entrainement positifs Trois alternatives ont été proposées, permettant de créer des échantillons positifs pour l'apprentissage du modèle SVM. Pour les alternatives (2) et (3), nous tirons profit des annotations effectuées sur le jeu de données.

1. Scores sémantiques du *SquaresChnFtrs*.
2. Masques d'annotations de tous les outils (cf. image D.5a).
3. Masques d'annotations de l'outil concerné uniquement (cf. image D.5b).



(a) All instruments annotation (b) Specific instrument annotation

Figure D.5: Echantillons positifs d'un tube d'aspiration pour l'apprentissage SVM. La ligne du haut représente la classe sémantique outil et celle du bas la classe arrière-plan.

Echantillons d'entrainement negatifs Les échantillons d'entrainement négatifs sont créés aléatoirement à partir d'une des trois stratégies suivantes:

1. Tirage binaire: distribution uniforme binaire où les pixels prennent soit la valeur 0 soit la valeur 255.

2. Tirage en niveaux de gris: distribution uniforme où les pixels prennent une valeur dans [0, 255].

3. Tirage gaussien: distribution gaussienne où les pixels prennent une valeur dans [0, 1], puis les valeurs sont remises à l'échelle dans [0, 255].

D.3.2.3 Modèle d'outil pour le *ShapeDetector*

Un modèle global décrivant plusieurs paramètres d'un outil chirurgical a besoin d'être crée pour pouvoir ensuite effectuer l'estimation de la pose dans le deuxième niveau du *ShapeDetector*. Les éléments suivants sont inclus, et représentés sur l'image D.6: le modèle SVM de l'outil, le polygone englobant décrivant la forme générique de l'outil, la position de l'extrémité de l'outil, l'orientation et échelle de l'outil, et enfin la catégorie de l'outil.

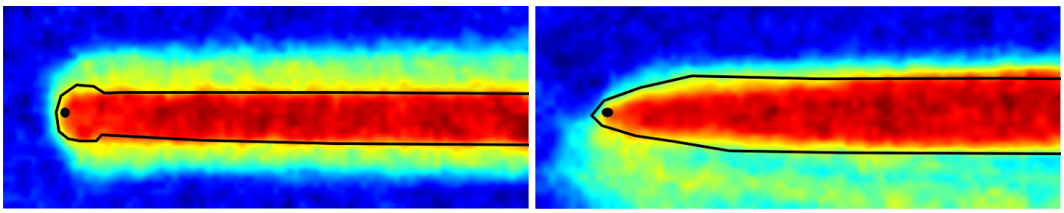
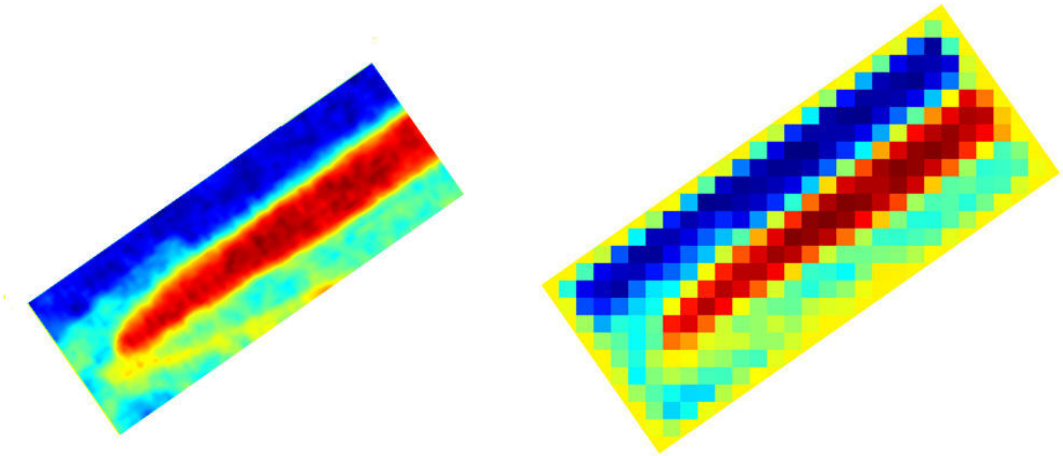


Figure D.6: Illustration du modèle général du *ShapeDetector* pour un tube d'aspiration (gauche) et la partie supérieure de la pince bipolaire (droite). En noir sont représentés le polygone englobant ainsi que la position de l'extrémité de l'outil. Les autres couleurs représentent le modèle SVM.

D.3.2.4 Second niveau: estimation de la pose des outils

L'estimation de la pose des outils dans l'image se fait en appliquant le modèle SVM suivant une stratégie de fenêtre glissante sur les scores sémantiques résultant du premier niveau.
Le modèle SVM d'un outil est appris pour une pose normalisée. Afin de détecter un nombre arbitraire d'outils, à des positions et orientations différentes, un jeu de modèles est dérivé du modèle de base pour correspondre aux paramètres de recherche spécifié par l'utilisateur. La génération d'un ensemble de modèles au lancement du processus permet de ne pas avoir à recalculer le premier niveau pour chaque échelle et orientation, permettant ainsi d'accélérer la vitesse d'acquisition des détections [Benenson 2012].
Chaque modèle SVM généré est découpé en blocs de 15×15 pixels et représenté comme un ensemble de carrés (cf. image D.7). Cette stratégie permet de tirer avantage des représentations intégrales des canaux de caractéristiques et d'obtenir un gain additionnel de vitesse.



(a) Modèle complet (b) Modèle par partie

Figure D.7: Effet visuel de l'approximation par parties du modèle SVM d'un forceps bipolaire, pour la classe outil.

D.4 Détections en temps-réel

Afin d'utiliser ce type de méthodes dans des systèmes opérationnels au sein de la salle opératoire, il est nécessaire de pouvoir traiter les données en temps-réel, c'est à dire entre 25 et 30Hz. Deux types de stratégies existent afin d'augmenter la vitesse de calcul d'un algorithme. Premièrement, optimiser le code, que ce soit sur le CPU ou le GPU, afin d'obtenir plus rapidement la même qualité de résultats. Deuxièmement, utiliser des stratégies d'optimisation ad-hoc ayant un effet négatif sur la qualité des résultats. Dans ce cas, il faut alors trouver le meilleur compromis entre qualité et vitesse.

D.4.1 Bonnes pratiques de programmation

Avant d'entrer dans une optimisation du code en profondeur, l'utilisation de "bonnes pratiques" de programmation représente une première couche d'optimisation de code permettant d'obtenir un gain de vitesse. Pour référence, les optimisations sont proposées pour un code écrit en C++.

- Déroulage de boucle: afin d'éviter des tests de condition du type if/else, et de réduire l'arithmétique sur les pointeurs.
- Fonctions inline: afin que le corps d'une fonction soit placé directement à l'endroit de son appel, évitant ainsi les sauts entre routines.
- Multi-threading: les libraires OpenMP peuvent être utilisées afin de répartir facilement le traitement des données sur les différents coeurs CPU à disposition.

- Identification des points bloquants "bottlenecks": pour augmenter la vitesse, il faut pouvoir identifier les bouts de code concentrant un temps important de calcul. Des logiciels spéciaux doivent être utilisés pour cette tâche, comme l'Intel VTunes Amplifier.

D.4.2 Implémentation CPU

Afin d'effectuer de manière efficace des calculs en parallèle sur le CPU, nous avons utilisé les instructions SIMD (Single Instruction Multiple Data). En un même temps CPU, la même opération (arithmétique ou logique) est effectuée simultanément pour différentes données.
Nous avons effectué deux implémentations différentes basées sur l'utilisation des instructions SIMD: SSE3 utilisant des registres CPU de 128 bits, et AVX utilisant des registres CPU de 256 bits. Dans le premier cas, seules quatre opérations sur des flottants peuvent être réalisées en même temps contre huit dans le second.

D.4.3 Implémentation GPU

L'utilisation des capacités de calcul du GPU apparait comme logique pour des algorithmes de traitement d'image où un même bout de code est exécuté à l'identique en chaque pixel d'une image. Nous avons opté pour une répartition comme suit:

- Taille d'un bloc: D_b.x = 16, D_b.y = 16, D_b.z = 1.
- Fonctions inline: D_g.x = Largeur/D_b.x, D_g.y = Hauteur/D_b.y, D_g.z = NbOrientations/D_b.z.

D.4.4 Méthodes d'optimisation ad-hoc

Pour ce dernier type de méthodes permettant d'accélérer la vitesse de calcul, il existe un impact sur la qualité des détections. Parmi les stratégies les plus prisées, on trouve le sous-échantillonnage de l'image de départ, la réduction des paramètres de recherche (nombre d'orientations moins important), et la limitation du nombre de détections candidates accélérant ainsi la procédure de NMS.

D.5 Méthodologie de validation

La littérature ne proposant pas de méthodologie de validation de référence, nous avons choisi d'utiliser une méthode similaire à celle utilisée par Dollar et al. [Dollár 2011] pour la détection des piétons. Nous proposons aussi un jeu de méthodes de référence (i.e. baseline) afin de comparer les performances de nos méthodes. Le protocole de validation (section D.5.1) définit la phase de spécification de la méthodologie et les métriques de validation définissent la façon dont le critère de validation est évalué (section D.5.2).

D.5.1 Protocole de validation

Le protocole de validation suivi a été établi par Dollar et al. [Dollár 2011]. Pour chaque image testée, l'ensemble de détections obtenues par notre méthode (BG_{dt}) est comparé aux détections de référence (BG_{gt}). Un appariement est effectué entre les éléments des deux ensembles en se basant sur la mesure de PASCAL [Everingham 2010].
Les différents modèles sont appris sur l'ensemble d'entrainement de notre jeu de données, et les résultats sont obtenus sur l'ensemble de test de ce même jeu de données.
Les résultats de validation sont affichés sous la forme de graphiques représentant le taux de détections manquées en ordonnée contre le nombre de faux positifs par image en abscisse. La moyenne du taux de détection manqué est calculée pour résumer les performances des détecteurs en une seule valeur.

D.5.2 Métriques de validation

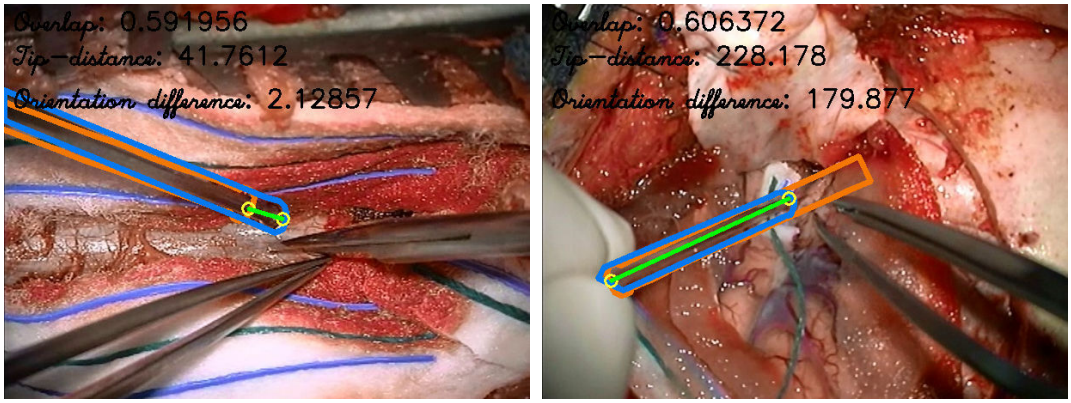


Figure D.8: Représentation visuelle des métriques de validation prises en compte et calculées entre la détection BG_{gt} bleue et la détection BG_{gt} orange. Les extrémités des outils sont représentées par des cercles jaunes et la ligne verte identifie la distance entre les deux extrémités.

Trois métriques sont utilisées pour valider trois paramètres d'estimation de pose des outils chirurgicaux (représentés figure D.8), et une quatrième métrique pour la qualité de la classification au niveau pixel:

- *Chevauchement de polygones*: le critère d'Intersection Sur l'Union est utilisé entre deux polygones. Chaque détection chevauchant sa référence de plus de 25% est considérée comme un vrai positif.

- *Différence en orientation*: la différence entre l'orientation de chaque vraie détection (au titre de la première métrique) et de sa référence est quantifiée.

- *Distance des extrémités*: la distance Euclidienne entre l'extrémité de chaque vraie détection (au titre de la première métrique) et de sa référence est calculée.

- *Qualité de la segmentation*: le pourcentage général de pixels correctement classifiés est calculé.

D.5.3 Méthodes comparées

Pour chaque catégorie d'approche considérée, au moins une méthode de référence de l'état de l'art est utilisée pour comparaison.

- Segmentation sémantique: la méthode Darwin est utilisée [Gould 2012].

- Approche à un niveau: le détecteur Linemod est utilisé.

- Approche à deux niveaux: la méthode *Skeleton*, créée à la main et se basant sur une succession d'opérations morphologiques.

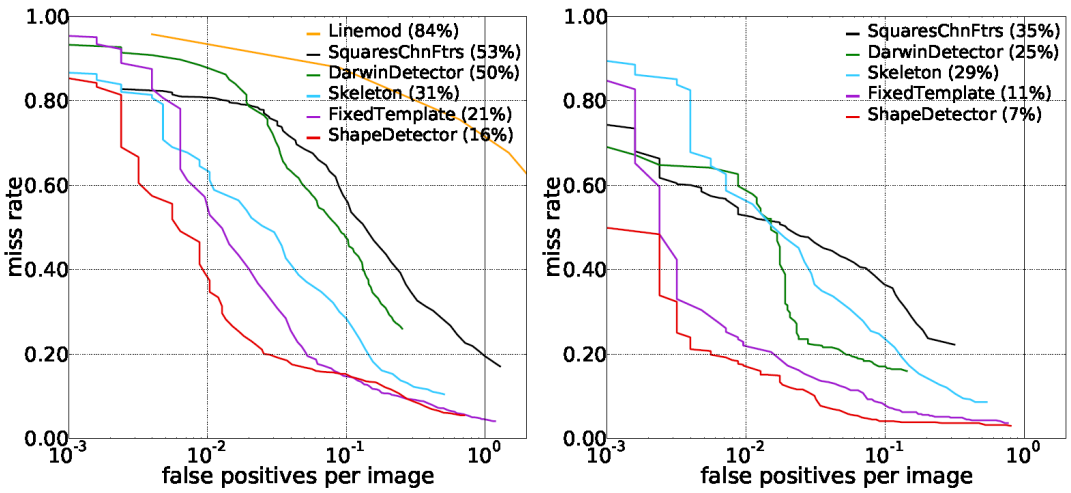
- Variantes du *ShapeDetector*: notre méthode permettant de créer plusieurs variantes, nous en considérons trois. La variante *FixedTemplate* appliquant un modèle fixe sur la segmentation du *SquaresChnFtrs*, la variante *DarwinDetector* appliquant un modèle fixe sur la segmentation Darwin, et la variante *ShapeDetector* appliquant un modèle SVM sur la segmentation *SquaresChnFtrs*.

D.6 Résultats

D.6.1 Qualité des détections

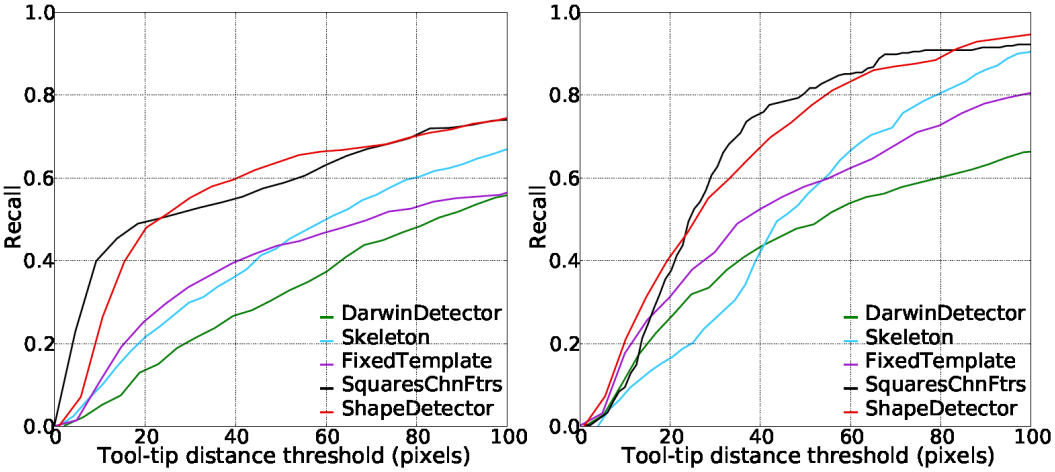
Les principaux résultats obtenus pour chaque détecteurs sont reportés sur la figure D.9. La méthode Linemod obtient des résultats assez faibles, démontrant l'incapacité d'une méthode générique de détection d'objets d'obtenir de bons résultats pour la détection des outils chirurgicaux. Notre *SquaresChnFtrs adapté* obtient en comparaison de meilleurs résultats, probablement grâce à son modèle flexible, mais les performances restent limitées avec moins de 50% de recall pour un taux de 10^{-1} faux positifs par image. Les méthodes à deux niveaux obtiennent de meilleurs résultats indiquant le fort impact de la segmentation sémantique. Finalement, notre *ShapeDetector* obtient les meilleurs résultats, divisant par deux le taux de détections ratées par rapport au meilleur détecteur générique. Les plus faibles performances du *DarwinDetector* par rapport au *FixedTemplate* indiquent la nécessité d'une segmentation de haute qualité afin d'obtenir les meilleures détections possibles.

Pour la validation de l'estimation de la position de l'extrémité des outils (figure D.10), le *SquaresChnFtrs adapté* et le *ShapeDetector* obtiennent des performances similaires.



(a) Suction tube (b) Bipolar forceps

Figure D.9: Performances des différentes méthodes selon la métrique *Chevauchement de polygones*.



(a) Tube (b) Forceps

Figure D.10: Performances des différentes méthodes selon la métrique *Distance des extrémités*. Evaluation réalisée à 10^{-1} faux-positifs par image.

Dans le cas du tube d'aspiration, une erreur inférieure à 20 pixels est obtenue pour 50% des détections considérées comme étant des vrais positifs. On peut noter l'importance que la qualité de l'étape de labélisation sémantique en comparant les résultats du *FixedTemplate* et du *DarwinDetector*. Pour une erreur d'estimation de la position de l'extrémité de l'outil de 40 pixels, un taux de rappel de 10% entre les deux approches est notable.

La figure D.11 illustre les résultats obtenus avec la méthode ShapeDetector pour un modèle de forceps.

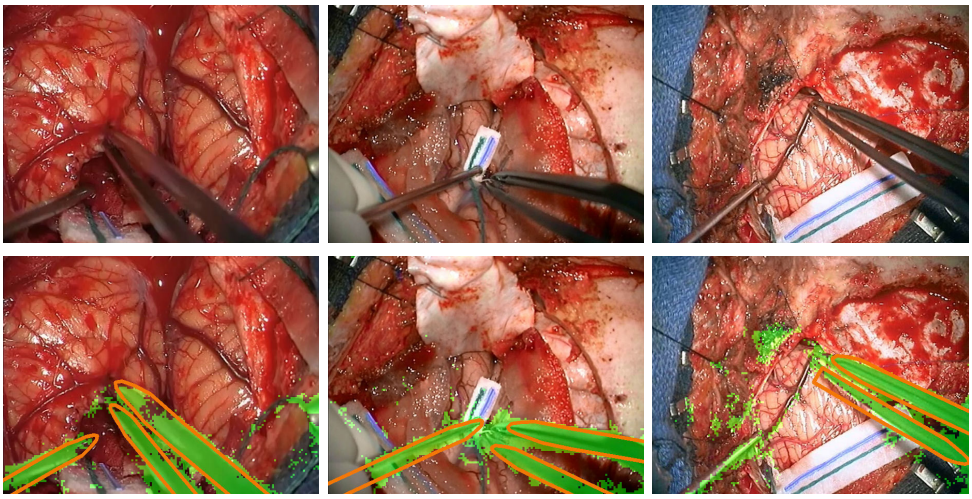


Figure D.11: Detections d'un forceps bipolaire obtenues avec le *ShapeDetector*. Les pixels classifiés comme appartenant à la classe outil sont marqués en vert.

D.6.2 Vitesse des détecteurs

Les résultats en vitesse ont été obtenus avec les spécifications d'ordinateur suivantes: DELL Precision T8600, Intel Xeon E5-2620 v2 @2.10GHz, NVIDIA GeForce Titan Black, et sont reportés dans la table D.1. Pour un temps de calcul initial d'environ $1.3\,s$, les optimisations CPU et GPU successives ont permis d'atteindre un temps de calcul de seulement $180\,ms$. Les optimisations ont été effectuées pour le *ShapeDetector*, et les temps sont reportés pour des images de taille 612×460 pixels.

Table D.1: Pour une image de 612×460 pixels, les temps de calcul du *ShapeDetector* selon différentes implémentations CPU and GPU.

	Niveau 1 (ms)	Niveau 2 (ms)	*ShapeDetector* (ms)
CPU non-opti.	/	1200	1290
CPU opti.	/	600	690
CPU SIMD	/	200	290
GPU	90	90	180

D.6.3 Discussion

D.6.3.1 Rotations dans le plan

A l'inverse de bien des méthodes de vision par ordinateur symbolisant une détection par une fenêtre rectangulaire, nous avons proposé l'utilisation d'une géométrie plus fine: des polygones. De par les multiples changements d'orientation dans le plan subis par les instruments chirurgicaux au cours d'une procédure, de larges portions de l'arrière-plan chirurgical seraient aussi présentes dans la détection. Ce phénomène est illustré dans l'image D.12, où des fenêtres rectangulaires symbolisant les détections sont représentées en rose, et les polygones proposés sont représentés en vert.

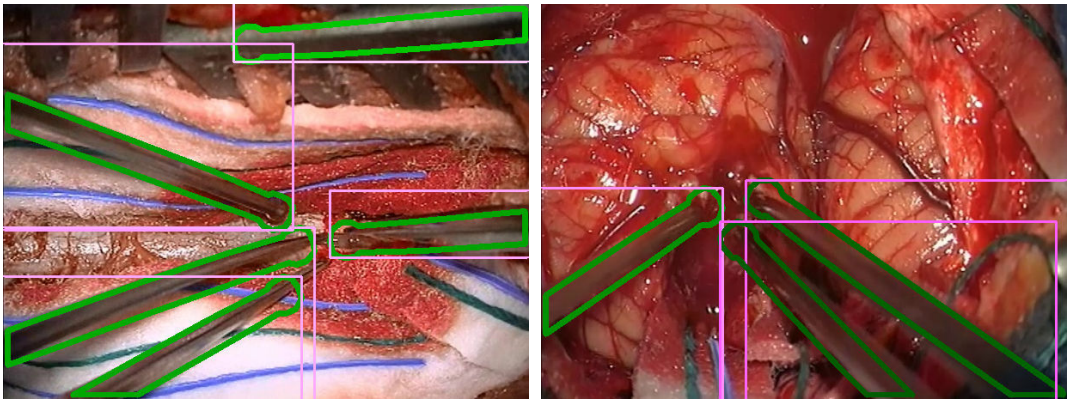


Figure D.12: Représentation des détections avec des boîtes englobantes (rose) et des polygones englobants (vert).

D.6.3.2 Labélisation sémantique

Les résultats obtenus par l'étape de classification au niveau pixel ne sont pas parfaits, comme en témoigne la présence de bruit principalement au niveau des contours des outils. Une amélioration de la qualité globale est nécessaire afin de pouvoir obtenir de manière plus précise la position de l'extrémité des outils. Avoir choisi de modéliser simplement deux classes pour cette étape apparait comme étant une bonne stratégie. La création d'une classe par catégorie d'outil chirurgical ne ferait qu'ajouter de la confusion entre classifieurs. En prenant le tube d'aspiration et le forceps bipolaire, les différences au niveau local sont presque inexistantes, les deux outils étant non texturés, métalliques et de couleur grise. De nouvelles classes pourraient être ajoutées dans le cas d'outils présentant des variations fortes de couleur, comme l'instrument IOL utilisé pour les chirurgies de la cataracte.

D.6.3.3 Détection et classification

Les approches proposées sont basées sur un apprentissage dépendant des données d'entrées et n'utilisent aucune connaissance a priori concernant la taille, forme, position, ou nombre des outils dans les images pour effectuer les détections. Cependant, plusieurs cas de détections erronées ou manquées ont été identifiés et la précision de la couche de labélisation sémantique apparait comme jouant un rôle prépondérant dans la qualité des détections.
Dans l'état actuel du *ShapeDetector*, il est possible d'utiliser plusieurs modèles d'outils SVM en même temps sur une image, même si ne permettant plus d'obtenir des détections en temps-réel. Cependant, le problème de classification entre outils demeure car les modèles SVM ont été conçus pour différentier un outil de l'arrière-plan chirurgical et non des autres outils. En conséquence, il est possible que le score d'une détection obtenue avec un modèle de tube d'aspiration soit plus élevé que le score d'une détection obtenue avec un modèle de forceps, sur une zone d'image laissant apparaitre un forceps (cf. image D.13).

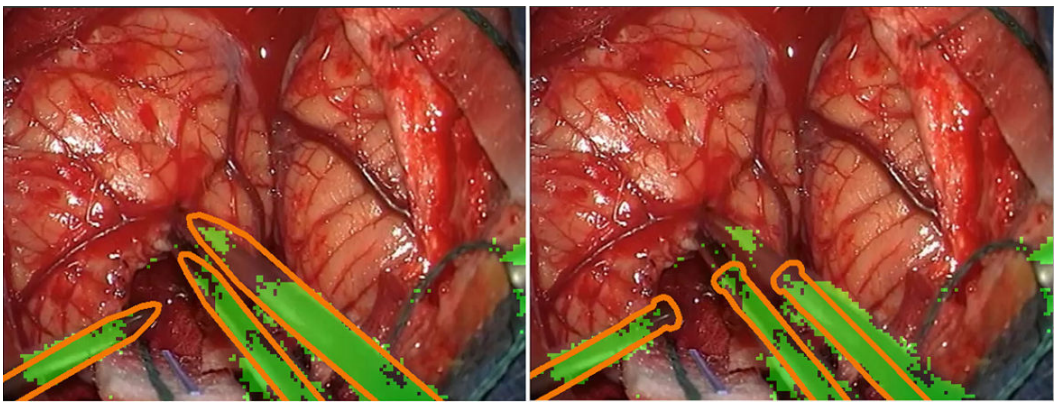


Figure D.13: Problème de classification du *ShapeDetector* avec un modèle de forceps (gauche) et de tube (droite).

D.7 Procédés de robustification

Afin d'intégrer une méthode de détection d'outils chirurgicaux dans un système opérationnel au sein du bloc opératoire avec un objectif médical précis, de meilleurs résultats de détection peuvent être requis. Nous avons exploré deux pistes permettant de robustifier nos méthodes de détection.

D.7.1 Utilisation de marqueurs externes

Dans certains cas, la classification entre deux types d'instruments chirurgicaux n'est pas possible en effectuant une simple analyse d'image. A ce titre, il est nécessaire de placer des marqueurs externes sur les outils afin de faciliter les tâches de détection et de classification.

Nous avons choisi d'utiliser une représentation similaire à un code barre, comme illustré sur l'image D.14. Pour effectuer l'identification de ce marqueur externe, nous considérons une ligne partant de l'extrémité de l'outil jusqu'au bord de l'image comme un signal 1D. Nous utilisons alors des techniques de traitement du signal comme l'application d'un filtre passe-bas ou de calcul de dérivée du signal.

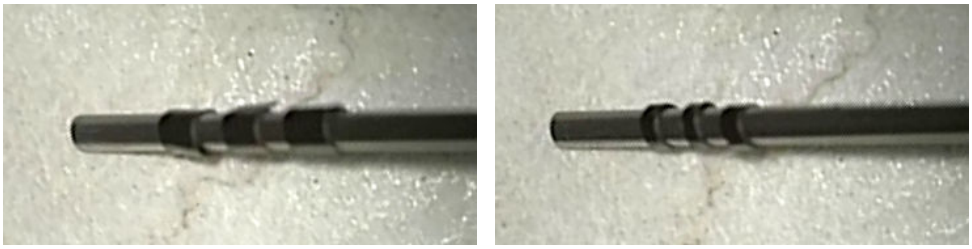


Figure D.14: Marqueur externe similaire à un code barre et placé sur des outils chirurgicaux.

D.7.2 Utilisation d'une méthode de suivi temporel

Comme établi par Benenson et al. [Benenson 2014], en partant d'une méthode de détection spatiale robuste et performante, l'utilisation de méthodes entre autre de suivi temporel permet d'améliorer encore un peu plus les résultats. A ce titre, nous nous sommes intéressés à l'utilisation de filtres de Kalman afin d'améliorer les résultats de détection de notre *ShapeDetector*.
Pour chaque 'track', c'est à dire chaque outil détecté et suivi temporellement, de lui associer deux filtres de Kalman. Le premier filtre permet d'estimer la position de l'extrémité de l'outil grâce à la position précédente et aux paramètres de vitesse de déplacement de l'outil. Le second permet l'estimation de l'orientation de l'outil à partir des orientations antérieures. La figure D.15 illustre les résultats obtenus, avec chaque 'track' représentée dans les images par une courbe noire.

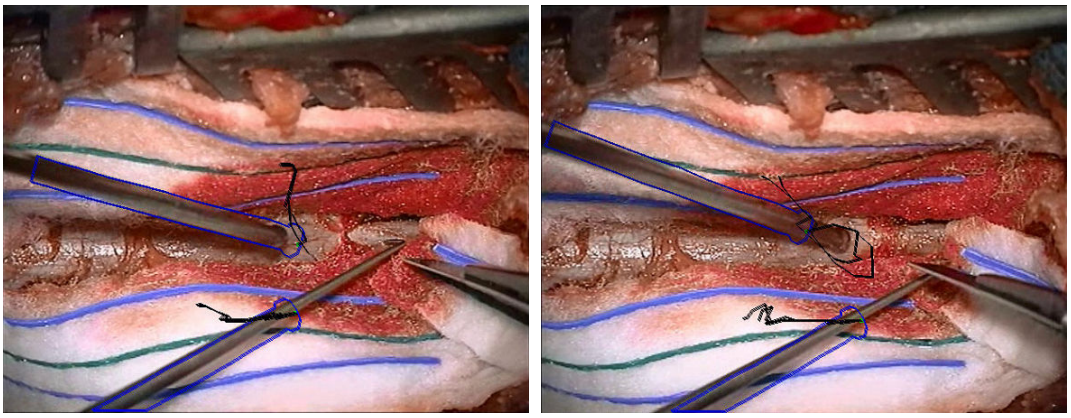


Figure D.15: Résultats du suivi temporel sur des séquences in-vivo.

D.8 Conclusion

Bien que devenant un environnement de plus en plus riche technologiquement, la salle opératoire reste un endroit où la sécurité des patients n'est pas assurée à 100% comme le montre le nombre toujours conséquent d'erreurs chirurgicales. La nécessité de développer des systèmes intelligents au bloc opératoire apparait comme croissante. Un des éléments clés pour ce type de système est la reconnaissance du processus chirurgical, passant par une identification précise des outils chirurgicaux utilisés.

Dans ces travaux de thèse, un nouveau jeu de données, deux méthodes de détection en temps-réel, ainsi qu'une méthodologie de validation des performances ont été proposés afin de répondre au problème d'identification des outils chirurgicaux dans des vidéos 2D. L'utilisation des vidéos provenant du microscope chirurgical est particulièrement intéressante car ne dérangeant par le chirurgien pendant l'intervention. De plus, le point de vue du microscope permet d'obtenir toutes les informations nécessaires pour l'identification des outils. Enfin, grâce aux nouvelles générations de microscopes, il sera possible d'extraire des informations de profondeur grâce à l'enregistrement de vidéos stéréoscopiques.

Pour le futur, il est important pour la communauté de mettre en place des jeux de données de référence ainsi que les méthodologies de validation correspondantes. De plus, effectuer l'évaluation de ce type d'approches dans des systèmes de plus haut niveau, en per- ou post-opératoire, apparait comme primordial.

Bibliography

[Allan 2013] Max Allan, Sébastien Ourselin, Steve Thompson, David J Hawkes, John Kelly et Danail Stoyanov. *Toward detection and localization of instruments in minimally invasive surgery.* Biomedical Engineering, IEEE Transactions on, vol. 60, no. 4, pages 1050–1058, 2013. (Cited pages 24, 25, 27, 29, 30, 31, 32, 33, 34, 35 and 37.)

[Babenko 2007] Boris Babenko, Piotr Dollár et Serge Belongie. *Task specific local region matching.* In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pages 1–8. IEEE, 2007. (Cited page 18.)

[Bardram 2011] Jakob E Bardram, Afsaneh Doryab, Rune M Jensen, Poul M Lange, Kristian LG Nielsen et Soren T Petersen. *Phase recognition during surgical procedures using embedded and body-worn sensors.* In Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on, pages 45–53. IEEE, 2011. (Cited pages 133 and 143.)

[Benenson 2012] Rodrigo Benenson, Markus Mathias, Radu Timofte et Luc Van Gool. *Pedestrian detection at 100 frames per second.* In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2903–2910. IEEE, 2012. (Cited pages 19, 44, 81 and 190.)

[Benenson 2013] Rodrigo Benenson, Markus Mathias, Tinne Tuytelaars et Luc Van Gool. *Seeking the strongest rigid detector.* In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 3666–3673, 2013. (Cited pages 17, 18, 19, 64, 66 and 186.)

[Benenson 2014] Rodrigo Benenson, Mohamed Omran, Jan Hosang et Bernt Schiele. *Ten Years of Pedestrian Detection, What Have We Learned?* arXiv preprint arXiv:1411.4304, 2014. (Cited pages 13, 38, 41, 42 and 199.)

[Bouarfa 2011] Loubna Bouarfa, Pieter P Jonker et Jenny Dankelman. *Discovery of high-level tasks in the operating room.* Journal of biomedical informatics, vol. 44, no. 3, pages 455–462, 2011. (Cited page 160.)

[Bouarfa 2012] Loubna Bouarfa, Oytun Akman, Armin Schneider, Pieter P Jonker et Jenny Dankelman. *In-vivo real-time tracking of surgical instruments in endoscopic video.* Minimally Invasive Therapy & Allied Technologies, vol. 21, no. 3, pages 129–134, 2012. (Cited page 140.)

[Burges 1998] Christopher JC Burges. *A tutorial on support vector machines for pattern recognition.* Data mining and knowledge discovery, vol. 2, no. 2, pages 121–167, 1998. (Cited pages 77 and 188.)

[Burschka 2005] Darius Burschka, Jason J Corso, Maneesh Dewan, William Lau, Ming Li, Henry Lin, Panadda Marayong, Nicholas Ramey, Gregory D Hager, Brian Hoffman*et al.* *Navigating inner space: 3-d assistance for minimally invasive surgery.* Robotics and Autonomous Systems, vol. 52, no. 1, pages 5–26, 2005. (Cited pages 24, 25, 26, 27, 29, 30, 31, 32, 33, 34 and 35.)

[Casals 1996] Alicia Casals, Josep Amat et E Laporte. *Automatic guidance of an assistant robot in laparoscopic surgery.* In Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on, volume 1, pages 895–900. IEEE, 1996. (Cited page 139.)

[Christe 2008] Barbara Christe, Elaine Cooney, Gregg Maggioli, Dustin Doty, Robert Frye et Jason Short. *Testing potential interference with RFID usage in the patient care environment.* Biomedical Instrumentation & Technology, vol. 42, no. 6, pages 479–484, 2008. (Cited page 142.)

[Cleary 2004] Kevin Cleary, Ho Young Chung et Seong K Mun. *OR2020 workshop overview: operating room of the future.* In International Congress Series, volume 1268, pages 847–852. Elsevier, 2004. (Cited page 5.)

[Dalal 2005] N. Dalal et B. Triggs. *Histograms of Oriented Gradients for Human Detection.* In CVPR, 2005. (Cited pages 15 and 18.)

[Dalal 2006] Navneet Dalal. *Finding people in images and videos.* PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2006. (Cited page 19.)

[Dantone 2014] M. Dantone, J. Gall, C. Leistner et L. van Gool. *Body Parts Dependent Joint Regressors for Human Pose Estimation in Still Images.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 36, no. 11, pages 2131–2143, 2014. (Cited page 132.)

[Doignon 2004] Christophe Doignon, Florent Nageotte et Michel De Mathelin. *Detection of grey regions in color images: application to the segmentation of a surgical instrument in robotized laparoscopy.* In Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, volume 4, pages 3394–3399. IEEE, 2004. (Cited pages 27, 31, 32 and 183.)

[Doignon 2007] Christophe Doignon, Florent Nageotte et Michel de Mathelin. *Segmentation and guidance of multiple rigid objects for intra-operative endoscopic vision.* In Dynamical Vision, pages 314–327. Springer, 2007. (Cited pages 24, 26 and 27.)

[Dollar 2006] Piotr Dollar, Zhuowen Tu et Serge Belongie. *Supervised learning of edges and object boundaries.* In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, volume 2, pages 1964–1971. IEEE, 2006. (Cited page 18.)

[Dollár 2007] Piotr Dollár, Zhuowen Tu, Hai Tao et Serge Belongie. *Feature mining for image classification.* In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pages 1–8. IEEE, 2007. (Cited page 18.)

[Dollár 2009a] P. Dollár, Z. Tu, P. Perona et S. Belongie. *Integral Channel Features.* In BMVC, 2009. (Cited pages xi, 65, 86 and 161.)

[Dollar 2009b] P. Dollar, C. Wojek, B. Schiele et P. Perona. *Pedestrian Detection: A Benchmark.* In CVPR, 2009. (Cited pages 13 and 15.)

[Dollár 2009c] Piotr Dollár, Zhuowen Tu, Pietro Perona et Serge Belongie. *Integral Channel Features.* In Proceedings of the British Machine Vision Conference, pages 91.1–91.11. BMVA Press, 2009. (Cited pages 17 and 20.)

[Dollár 2010] P. Dollár, S. Belongie et P. Perona. *The Fastest Pedestrian Detector in the West.* In BMVC, 2010. (Cited page 17.)

[Dollár 2011] P. Dollár, C. Wojek, B. Schiele et P. Perona. *Pedestrian Detection: An Evaluation of the State of the Art.* TPAMI, 2011. (Cited pages 13, 16, 20, 21, 39, 44, 82, 86, 87, 105, 131, 165, 192 and 193.)

[Dollár 2012] P. Dollár, R. Appel et W. Kienzle. *Crosstalk Cascades for Frame-Rate Pedestrian Detection.* In ECCV, 2012. (Cited page 17.)

[Enzweiler 2009] M. Enzweiler et D. M. Gavrila. *Monocular Pedestrian Detection: Survey and Experiments.* PAMI, 2009. (Cited page 15.)

[Ess 2009a] A. Ess, B. Leibe, K. Schindler et L. Van Gool. *Robust Multi-Person Tracking from a Mobile Platform.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 10, pages 1831–1846, 2009. (Cited page 15.)

[Ess 2009b] A. Ess, T. Müller, H. Grabner et L. Van Gool. *Segmentation-based Urban Traffic Scene Understanding.* In Proceedings of the British Machine Vision Conference, pages 84.1–84.11. BMVA Press, 2009. (Cited page 81.)

[Everingham 2010] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn et Andrew Zisserman. *The pascal visual object classes (voc) challenge.* International journal of computer vision, vol. 88, no. 2, pages 303–338, 2010. (Cited pages 16, 20, 21, 82, 184 and 193.)

[Everingham 2014] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn et Andrew Zisserman. *The Pascal Visual Object Classes Challenge: a Retrospective.* International Journal of Computer Vision, pages 1–39, 2014. (Cited page 88.)

[Felzenszwalb 2008] Pedro Felzenszwalb, David McAllester et Deva Ramanan. *A discriminatively trained, multiscale, deformable part model.* In Computer

Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008. (Cited pages 19 and 67.)

[Felzenszwalb 2010] P. Felzenszwalb, R. Girshick, D. McAllester et D. Ramanan. *Object Detection with Discriminatively Trained Part-Based Models.* PAMI, 2010. (Cited page 18.)

[Forestier 2013] Germain Forestier, Florent Lalys, Laurent Riffaud, D Louis Collins, Jurgen Meixensberger, Shafik N Wassef, Thomas Neumuth, Benoit Goulet et Pierre Jannin. *Multi-site study of surgical practice in neurosurgery based on surgical process models.* Journal of biomedical informatics, vol. 46, no. 5, pages 822–829, 2013. (Cited page 8.)

[Galleguillos 2010] Carolina Galleguillos et Serge Belongie. *Context based object categorization: A critical survey.* Computer Vision and Image Understanding, vol. 114, no. 6, pages 712–722, 2010. (Cited page 18.)

[Gallo 2011] Orazio Gallo et Roberto Manduchi. *Reading 1d barcodes with mobile phones using deformable templates.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 33, no. 9, pages 1834–1843, 2011. (Cited pages 145 and 148.)

[Geiger 2012] Andreas Geiger, Philip Lenz et Raquel Urtasun. *Are we ready for autonomous driving? The KITTI vision benchmark suite.* In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 3354–3361. IEEE, 2012. (Cited pages 15 and 45.)

[Gibaud 2014] Bernard Gibaud, Cédric Penet et Pierre Jannin. *OntoSPM: a core ontology of surgical procedure models.* SURGETICA, 2014. (Cited pages 7 and 164.)

[Gould 2012] Stephen Gould. *DARWIN: A framework for machine learning and computer vision research and development.* The Journal of Machine Learning Research, vol. 13, no. 1, pages 3533–3537, 2012. (Cited pages 89 and 194.)

[Haase 2013] Sven Haase, Jakob Wasza, Thomas Kilgus et Joachim Hornegger. *Laparoscopic instrument localization using a 3-D Time-of-Flight/RGB endoscope.* In Applications of Computer Vision (WACV), 2013 IEEE Workshop on, pages 449–454. IEEE, 2013. (Cited pages 24, 26, 27, 28, 29, 30, 31, 32, 34, 35, 37, 38 and 183.)

[Heitz 2008] Geremy Heitz et Daphne Koller. *Learning spatial context: Using stuff to find things.* In Computer Vision–ECCV 2008, pages 30–43. Springer, 2008. (Cited page 19.)

[Hinterstoisser 2010] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua et Nassir Navab. *Dominant orientation templates for real-time detection*

of texture-less objects. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 2257–2264, 2010. (Cited page 89.)

[Hinterstoisser 2011] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab et Vincent Lepetit. *Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes.* In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 858–865. IEEE, 2011. (Not cited.)

[Houliston 2009] Bryan Houliston, David Parry, Craig S Webster et Alan F Merry. *Interference with the operation of medical devices resulting from the use of radio frequency identification technology.* NZ Med J, vol. 122, no. 1297, pages 9–16, 2009. (Cited page 142.)

[Jannin 2006] Pierre Jannin, Christophe Grova et Calvin R Maurer Jr. *Model for defining and reporting reference-based validation protocols in medical image processing.* International Journal of Computer Assisted Radiology and Surgery, vol. 1, no. 2, pages 63–73, 2006. (Cited pages 20, 27, 34 and 35.)

[Jannin 2008] Pierre Jannin et Werner Korb. *Assessment of image-guided interventions.* In Image-Guided Interventions, pages 531–549. Springer, 2008. (Cited page 25.)

[Katić 2014] Darko Katić, Anna-Laura Wekerle, Fabian Gärtner, Hannes Kenngott, Beat Peter Müller-Stich, Rüdiger Dillmann et Stefanie Speidel. *Knowledge-Driven Formalization of Laparoscopic Surgeries for Rule-Based Intraoperative Context-Aware Assistance.* In Information Processing in Computer-Assisted Interventions, pages 158–167. Springer, 2014. (Cited page 8.)

[Ko 2005] Seong-Young Ko, Jonathan Kim, Dong-Soo Kwon et Woo-Jung Lee. *Intelligent interaction between surgeon and laparoscopic assistant robot system.* In Robot and Human Interactive Communication, 2005. ROMAN 2005. IEEE International Workshop on, pages 60–65. IEEE, 2005. (Cited page 140.)

[Kohn 2000] Linda T Kohn, Janet M Corrigan, Molla S Donaldson*et al.* To err is human:: Building a safer health system, volume 627. National Academies Press, 2000. (Cited pages 5 and 182.)

[Kranzfelder 2014] Michael Kranzfelder, Armin Schneider, Adam Fiolka, Sebastian Koller, Silvano Reiser, Thomas Vogel, Dirk Wilhelm et Hubertus Feussner. *Reliability of sensor-based real-time workflow recognition in laparoscopic cholecystectomy.* International journal of computer assisted radiology and surgery, pages 1–8, 2014. (Cited page 144.)

[Krupa 2003] Alexandre Krupa, Jacques Gangloff, Christophe Doignon, Michel F de Mathelin, Guillaume Morel, Joël Leroy, Luc Soler et Jacques Marescaux.

Autonomous 3-D positioning of surgical instruments in robotized laparoscopic surgery using visual servoing. Robotics and Automation, IEEE Transactions on, vol. 19, no. 5, pages 842–853, 2003. (Cited page 28.)

[Kumar 2013a] S. Kumar, M. Narayanan, S. Misra, S. Garimella, P. Singhal, J. Corso et V. Krovi. *Video-based Framework for Safer and Smarter Computer Aided Surgery.* In Hamlyn Symposium on Medical Robotics, pages 107–108, 2013. (Cited page 131.)

[Kumar 2013b] Suren Kumar, Madusudanan Sathia Narayanan, Pankaj Singhal, Jason J Corso et Venkat Krovi. *Product of tracking experts for visual tracking of surgical tools.* In Automation Science and Engineering (CASE), 2013 IEEE International Conference on, pages 480–485. IEEE, 2013. (Cited pages 24, 25, 26, 27, 28, 29, 30, 31, 34, 35, 36, 37, 38, 39, 183 and 184.)

[Lalys 2012] Florent Lalys, Laurent Riffaud, David Bouget et Pierre Jannin. *A framework for the recognition of high-level surgical tasks from video images for cataract surgeries.* Biomedical Engineering, IEEE Transactions on, vol. 59, no. 4, pages 966–976, 2012. (Cited page 133.)

[Lalys 2013] Florent Lalys, David Bouget, Laurent Riffaud et Pierre Jannin. *Automatic knowledge-based recognition of low-level tasks in ophthalmological procedures.* International journal of computer assisted radiology and surgery, vol. 8, no. 1, pages 39–49, 2013. (Cited pages 8, 12 and 183.)

[Lalys 2014] Florent Lalys et Pierre Jannin. *Surgical process modelling: a review.* International journal of computer assisted radiology and surgery, vol. 9, no. 3, pages 495–511, 2014. (Cited pages 7, 8 and 182.)

[Laptev 2006] Ivan Laptev. *Improvements of Object Detection Using Boosted Histograms.* In BMVC, volume 6, pages 949–958, 2006. (Cited page 18.)

[Lea 2012] Colin S Lea, James C Fackler, Gregory D Hager et Russell H Taylor. *Towards automated activity recognition in an intensive care unit.* In MICCAI Workshop on Modeling and Monitoring of Computer Assisted Interventions, pages 19–28, 2012. (Cited page 133.)

[Lehmann 2011] Alain Lehmann, Bastian Leibe et Luc Van Gool. *Fast prism: Branch and bound hough transform for object class detection.* International journal of computer vision, vol. 94, no. 2, pages 175–197, 2011. (Cited pages 77 and 188.)

[Lin 2008] Zhe Lin et Larry S Davis. *A pose-invariant descriptor for human detection and segmentation.* In Computer Vision–ECCV 2008, pages 423–436. Springer, 2008. (Cited page 17.)

[Maier-Hein 2014] Lena Maier-Hein, Sven Mersmann, Daniel Kondermann, Christian Stock, Hannes Gotz Kenngott, Alexandro Sanchez, Martin Wagner,

Anas Preukschas, Anna-Laura Wekerle, Stefanie Helfert*et al.* *Crowdsourcing for reference correspondence generation in endoscopic images.* In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014, pages 349–356. Springer, 2014. (Cited page 60.)

[Maji 2008] Subhransu Maji, Alexander C Berg et Jitendra Malik. *Classification using intersection kernel support vector machines is efficient.* In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008. (Cited page 18.)

[Makhoul 1999] John Makhoul, Francis Kubala, Richard Schwartz, Ralph Weischedel*et al.* *Performance measures for information extraction.* In Proceedings of DARPA broadcast news workshop, pages 249–252, 1999. (Cited pages 21 and 37.)

[Mathias 2013] Markus Mathias, Rodrigo Benenson, Radu Timofte et Luc Van Gool. *Handling occlusions with franken-classifiers.* In Computer Vision (ICCV), 2013 IEEE International Conference on, pages 1505–1512. IEEE, 2013. (Cited pages 17 and 42.)

[McKenna 2005] SJ McKenna, H Nait Charif et T Frank. *Towards video understanding of laparoscopic surgery: Instrument tracking.* In Proc. of Image and Vision Computing, New Zealand, 2005. (Cited pages 24, 25, 27, 28, 29, 30, 31, 32, 34, 36, 37, 183 and 184.)

[Miyawaki 2009] Fujio Miyawaki, Takashi Tsunoi, Hiromi Namiki, Takashi Yaginuma, Kitaro Yoshimitsu, Daijo Hashimoto et Yasuhiro Fukui. *Development of automatic acquisition system of surgical-instrument informantion in endoscopic and laparoscopic surgey.* In Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on, pages 3058–3063. IEEE, 2009. (Cited pages 28 and 144.)

[Neumuth 2007] T Neumuth, C Trantakis, F Eckhardt, M Dengl, J Meixensberger et O Burgert. *Supporting the analysis of intervention courses with surgical process models on the example of 14 microsurgical lumbar discectomies.* International Journal of Computer Assisted Radiology and Surgery, vol. 2, pages S436–S438, 2007. (Cited page 7.)

[Neumuth 2009] Thomas Neumuth, Pierre Jannin, Gero Strauss, Juergen Meixensberger et Oliver Burgert. *Validation of knowledge acquisition for surgical process models.* Journal of the American Medical Informatics Association, vol. 16, no. 1, pages 72–80, 2009. (Cited page 133.)

[Ojala 2002] T. Ojala, M. Pietikäinen et T. Mäenpää. *Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns.* PAMI, vol. 24, no. 7, pages 971–987, 2002. (Cited page 17.)

[Ouyang 2013] Wanli Ouyang et Xiaogang Wang. *Joint deep learning for pedestrian detection*. In Computer Vision (ICCV), 2013 IEEE International Conference on, pages 2056–2063. IEEE, 2013. (Cited page 18.)

[Overett 2008] Gary Overett, Lars Petersson, Nathan Brewer, Lars Andersson et Niklas Pettersson. *A new pedestrian dataset for supervised learning*. In Intelligent Vehicles Symposium, 2008 IEEE, pages 373–378. IEEE, 2008. (Cited page 15.)

[Papageorgiou 2000] Constantine Papageorgiou et Tomaso Poggio. *A Trainable System for Object Detection*. Int. Journal of Computer Vision, vol. 38, no. 1, pages 15–33, 2000. (Cited page 15.)

[Park 2010] D. Park, D. Ramanan et C. Fowlkes. *Multiresolution models for object detection*. In ECCV, 2010. (Cited pages 18 and 19.)

[Park 2013] Dennis Park, C Lawrence Zitnick, Deva Ramanan et Piotr Dollár. *Exploring weak stabilization for motion feature extraction*. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 2882–2889. IEEE, 2013. (Cited page 17.)

[Parlak 2011] Siddika Parlak, Ivan Marsic et Randall S Burd. *Activity recognition for emergency care using RFID*. In Proceedings of the 6th International Conference on Body Area Networks, pages 40–46. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011. (Cited page 143.)

[Pezzementi 2009] Zachary Pezzementi, Sandrine Voros et Gregory D Hager. *Articulated object tracking by rendering consistent appearance parts*. In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, pages 3940–3947. IEEE, 2009. (Cited pages 24, 25, 26, 27, 29, 30, 32, 33, 34, 36, 37, 183 and 184.)

[Rathod 2012] Mr Nachiket A Rathod et Siddharth A Ladhake. *Detecting and Decoding Algorithm for 2D Barcode*. International Journal of Emerging Technology and Advanced Engineering, vol. 2, 2012. (Cited page 145.)

[Reiter 2010] Austin Reiter et Peter K Allen. *An online learning approach to in-vivo tracking using synergistic features*. In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pages 3441–3446. IEEE, 2010. (Cited pages 24, 25, 27, 28, 29, 30, 31, 33, 34, 35, 36 and 37.)

[Reiter 2012a] Austin Reiter, Peter K Allen et Tao Zhao. *Feature classification for tracking articulated surgical tools*. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2012, pages 592–600. Springer, 2012. (Cited pages 24, 25, 26, 27, 28, 29, 30, 31, 33, 34, 36, 37, 41 and 162.)

[Reiter 2012b] Austin Reiter, Peter K Allen et Tao Zhao. *Marker-less articulated surgical tool detection.* In Computer assisted radiology and surgery, 2012. (Cited pages 24, 27, 29, 30, 32, 33, 34, 37 and 42.)

[Richa 2011] Rogério Richa, Raphael Sznitman, Russell Taylor et Gregory Hager. *Visual tracking using the sum of conditional variance.* In Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference On, pages 2953–2958. IEEE, 2011. (Cited pages 24, 25, 26, 27, 29, 31, 34 and 35.)

[Russell 2007] B. Russell, A. Torralba, K. Murphy et W. T. Freeman. *LabelMe: a database and web-based tool for image annotation.* IJCV, 2007. (Cited page 58.)

[Sabzmeydani 2007] Payam Sabzmeydani et Greg Mori. *Detecting pedestrians by learning shapelet features.* In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, pages 1–8. IEEE, 2007. (Cited page 17.)

[Schwartz 2009] William Robson Schwartz, Aniruddha Kembhavi, David Harwood et Larry S Davis. *Human detection using partial least squares analysis.* In Computer vision, 2009 IEEE 12th international conference on, pages 24–31. IEEE, 2009. (Cited page 17.)

[Sermanet 2013] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala et Yann LeCun. *Pedestrian detection with unsupervised multi-stage feature learning.* In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 3626–3633. IEEE, 2013. (Cited page 18.)

[Shahbaz Khan 2012] F Shahbaz Khan, Rao Muhammad Anwer, Joost van de Weijer, Andrew D Bagdanov, Maria Vanrell et Antonio M Lopez. *Color attributes for object detection.* In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 3306–3313, 2012. (Cited page 74.)

[Shotton 2006] Jamie Shotton, John Winn, Carsten Rother et Antonio Criminisi. *Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation.* In Computer Vision–ECCV 2006, pages 1–15. Springer, 2006. (Cited pages 75 and 89.)

[Shotton 2013] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook et Richard Moore. *Real-time human pose recognition in parts from single depth images.* Communications of the ACM, vol. 56, no. 1, pages 116–124, 2013. (Cited page 42.)

[Speidel 2006] Stefanie Speidel, Michael Delles, Carsten Gutt et Rüdiger Dillmann. *Tracking of instruments in minimally invasive surgery for surgical skill analysis.* In Medical Imaging and Augmented Reality, pages 148–155. Springer, 2006. (Cited pages 24, 27, 29, 30, 31, 32, 33, 34, 35, 36, 41 and 183.)

[Speidel 2008] Stefanie Speidel, Gunther Sudra, Julien Senemaud, Maximilian Drentschew, Beat Peter Müller-Stich, Carsten Gutt et Rüdiger Dillmann. *Recognition of risk situations based on endoscopic instrument tracking and knowledge based situation modeling.* In Medical Imaging, pages 69180X–69180X. International Society for Optics and Photonics, 2008. (Cited pages 24, 25, 26, 27, 29, 30, 32, 34, 35, 36, 40 and 184.)

[Speidel 2014] S Speidel, E Kuhn, S Bodenstedt, S Röhl, H Kenngott, B Müller-Stich et R Dillmann. *Visual tracking of da Vinci instruments for laparoscopic surgery.* In SPIE Medical Imaging, pages 903608–903608. International Society for Optics and Photonics, 2014. (Cited pages 24, 26, 27, 28, 29, 30, 31, 34, 36 and 37.)

[Speidela 2013] S Speidela, S Krappeb, S Röhla, S Bodenstedta, B Müller-Stichc et R Dillmanna. *Robust feature tracking for endoscopic pose estimation and structure recovery.* In SPIE Medical Imaging, pages 867102–867102. International Society for Optics and Photonics, 2013. (Cited pages 24, 25, 27, 33 and 37.)

[Sudra 2007] Gunther Sudra, Stefanie Speidel, Dominik Fritz, Beat Peter Müller-Stich, Carsten Gutt et Rüdiger Dillmann. *MEDIASSIST: medical assistance for intraoperative skill transfer in minimally invasive surgery using augmented reality.* In Medical Imaging, pages 65091O–65091O. International Society for Optics and Photonics, 2007. (Cited page 8.)

[Sznitman 2012] Raphael Sznitman, Karim Ali, Rogério Richa, Russell H Taylor, Gregory D Hager et Pascal Fua. *Data-Driven visual tracking in retinal microsurgery.* In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2012, pages 568–575. Springer, 2012. (Cited pages 24, 25, 26, 27, 28, 29, 30, 31, 34, 35, 36, 37, 132, 183 and 184.)

[Sznitman 2013] Raphael Sznitman, Rogerio Richa, Russell H Taylor, Bruno Jedynak et Gregory D Hager. *Unified detection and tracking of instruments during retinal microsurgery.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 35, no. 5, pages 1263–1273, 2013. (Cited pages 24, 25, 26, 27, 28, 29, 31, 32, 34, 35, 36 and 37.)

[Sznitman 2014] Raphael Sznitman, Carlos Joaquin Becker et Pascal Fua. *Fast Part-Based Classification for Instrument Detection in Minimally Invasive Surgery.* In Medical Image Computing and Computer Assisted Intervention. Springer, 2014. (Cited pages 24, 25, 26, 27, 29, 31, 32, 33, 34, 35, 36, 37, 38, 42, 44 and 183.)

[Tonet 2007] Oliver Tonet, Ramesh U Thoranaghatte, Giuseppe Megali et Paolo Dario. *Tracking endoscopic instruments without a localizer: A shape-analysis-based approach.* Computer Aided Surgery, vol. 12, no. 1, pages 35–42, 2007. (Cited pages 28 and 140.)

[Tu 2005] Zhuowen Tu. *Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering.* In Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, volume 2, pages 1589–1596. IEEE, 2005. (Cited page 18.)

[Van Der Togt 2008] Remko Van Der Togt, Erik Jan van Lieshout, Reinout Hensbroek, E Beinat, JM Binnekade et PJM Bakker. *Electromagnetic interference from radio frequency identification inducing potentially hazardous incidents in critical care medical equipment.* Jama, vol. 299, no. 24, pages 2884–2890, 2008. (Cited page 142.)

[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 pages 65 and 186.)

[Viola 2004] P. Viola et M. Jones. *Robust Real-Time Face Detection.* In IJCV, 2004. (Cited pages 17 and 18.)

[Voros 2007] Sandrine Voros, Jean-Alexandre Long et Philippe Cinquin. *Automatic detection of instruments in laparoscopic images: A first step towards high-level command of robotic endoscopic holders.* The International Journal of Robotics Research, vol. 26, no. 11-12, pages 1173–1190, 2007. (Cited pages 24, 25, 26, 27, 29, 30, 31, 32, 33, 34, 35, 36 and 183.)

[Walk 2010] Stefan Walk, Nikodem Majer, Konrad Schindler et Bernt Schiele. *New features and insights for pedestrian detection.* In Computer vision and pattern recognition (CVPR), 2010 IEEE conference on, pages 1030–1037. IEEE, 2010. (Cited page 17.)

[Wei 1997] Guo-Qing Wei, Klaus Arbter et Gerd Hirzinger. *Real-time visual servoing for laparoscopic surgery. Controlling robot motion with color image segmentation.* Engineering in Medicine and Biology Magazine, IEEE, vol. 16, no. 1, pages 40–45, 1997. (Cited page 140.)

[Wojek 2009] C. Wojek, S. Walk et B. Schiele. *Multi-cue onboard pedestrian detection.* In CVPR, 2009. (Cited page 15.)

[Wolf 2006] Lior Wolf et Stanley Bileschi. *A critical view of context.* International Journal of Computer Vision, vol. 69, no. 2, pages 251–261, 2006. (Cited page 18.)

[Wolf 2011] Rémi Wolf, Josselin Duchateau, Philippe Cinquin et Sandrine Voros. *3D tracking of laparoscopic instruments using statistical and geometric modeling.* In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011, pages 203–210. Springer, 2011. (Cited pages 24, 25, 26, 27, 29, 31, 32, 34, 35, 36, 37 and 184.)

[Yan 2013] Junjie Yan, Xucong Zhang, Zhen Lei, Shengcai Liao et Stan Z Li. *Robust multi-resolution pedestrian detection in traffic scenes.* In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 3033–3040. IEEE, 2013. (Cited page 19.)

[Yan 2014] Junjie Yan, Zhen Lei, Longyin Wen et Stan Z Li. *The fastest deformable part model for object detection.* In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2497–2504. IEEE, 2014. (Cited page 18.)

[Yao 2010] Wen Yao, Chao-Hsien Chu et Zang Li. *The use of RFID in healthcare: Benefits and barriers.* In RFID-Technology and Applications (RFID-TA), 2010 IEEE International Conference on, pages 128–134. IEEE, 2010. (Cited page 142.)

[Zhang 2002] Xiaoli Zhang et Shahram Payandeh. *Application of visual tracking for robot-assisted laparoscopic surgery.* Journal of Robotic systems, vol. 19, no. 7, pages 315–328, 2002. (Cited page 139.)

[Zhang 2007] Cha Zhang et Paul Viola. *Multiple-instance pruning for learning efficient cascade detectors.* In NIPS, 2007. (Cited page 68.)

[Zhang 2014] Shanshan Zhang, Christian Bauckhage et Armin B Cremers. *Informed Haar-like features improve pedestrian detection.* In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 947–954. IEEE, 2014. (Cited page 18.)

[Zhao 1991] Dongming Zhao et David G Daut. *Morphological hit-or-miss transformation for shape recognition.* Journal of Visual Communication and Image Representation, vol. 2, no. 3, pages 230–243, 1991. (Cited page 89.)

Détection en temps-réel des outils chirurgicaux dans des vidéos 2D de neurochirurgie par modélisation de forme globale et d'apparence locale.

Résumé : Bien que devenant un environnement de plus en plus riche technologiquement, la salle opératoire reste un endroit où la sécurité des patients n'est pas assurée à 100% comme le montre le nombre toujours conséquent d'erreurs chirurgicales. La nécessité de développer des systèmes intelligents au bloc opératoire apparait comme croissante. Un des éléments clés pour ce type de système est la reconnaissance du processus chirurgical, passant par une identification précise des outils chirurgicaux utilisés. L'objectif de cette thèse a donc porté sur la détection en temps-réel des outils chirurgicaux dans des vidéos 2D provenant de microscopes. Devant l'absence de jeux de données de référence, qui plus est dans un contexte neurochirurgical, la première contribution de la thèse a donc été la création d'un nouvel ensemble d'images de chirurgies du cerveau et du rachis cervical, mis à disposition en ligne. Comme seconde contribution, deux approches différentes ont été proposées permettant de détecter des outils chirurgicaux via des techniques d'analyse d'image. Tout d'abord, le *SquaresChnFtrs adapté*, basé sur une des méthodes de détection de piétons les plus performantes de la littérature. Notre deuxième méthode, le *ShapeDetector*, est une approche à deux niveaux n'utilisant aucune contrainte ou hypothèse a priori sur le nombre, la position, ou la forme des outils dans l'image. Par rapport aux travaux précédents du domaine, nous avons choisi de représenter les détections potentielles par des polygones plutôt que par des rectangles, obtenant ainsi des détections plus précises. Pour intégration dans des systèmes médicaux, une optimisation de la vitesse de calcul a été effectuée via un usage optimal du CPU, du GPU, et de méthodes ad-hoc. Pour des vidéos de résolution 612×480 pixels, notre *ShapeDetector* est capable d'effectuer les détections à une vitesse maximale de 8 Hz. Pour la validation de nos méthodes, nous avons pris en compte trois paramètres: la position globale, la position de l'extrémité, et l'orientation des détections. Les méthodes ont été classées et comparées avec des méthodes de référence compétitives. Pour la détection des tubes d'aspiration, nous avons obtenu un taux de manqué de 15% pour un taux de faux positifs par image de 10^{-1}, comparé à un taux de manqué de 55% pour le *SquaresChnFtrs adapté*. L'orientation future du travail devra porter sur l'intégration des informations 3D, l'amélioration de la couche de labélisation sémantique, et la classification des outils chirurgicaux. Pour finir, un enrichissement du jeu de données et des annotations de plus haute précision seront nécessaires.

Mots clés : Détection d'objet, vision par ordinateur, outils chirurgicaux, images de microscope.

Real-time detection of surgical tools in 2D neurosurgical videos by modelling global shape and local appearance.

Abstract: Despite modern-life technological advances and tremendous progress made in surgical techniques including MIS, today's OR is facing many challenges remaining yet to be addressed. The development of CAS systems integrating the SPM methodology was born as a response from the medical community, with the long-term objective to create surgical cockpit systems. Being able to identify surgical tools in use is a key component for systems relying on the SPM methodology. Towards that end, this thesis work has focused on real-time surgical tool detection from microscope 2D images. From the review of the literature, no validation data-sets have been elected as benchmarks by the community. In addition, the neurosurgical context has been addressed only once. As such, the first contribution of this thesis work consisted in the creation of a new surgical tool data-set, made freely available online. Two methods have been proposed to tackle the surgical tool detection challenge. First, the *adapted SquaresChnFtrs*, evolution of one of the best computer vision state-of-the-art approach for pedestrian detection. Our second contribution, the *ShapeDetector*, is fully data-driven and performs detection without the use of prior knowledge regarding the number, shape, and position of tools in the image. Compared to previous works, we chose to represent candidate detections with bounding polygons instead of bounding boxes, hence providing more fitting results. For integration into medical systems, we performed different code optimization through CPU and GPU use. Speed gain and accuracy loss from the use of ad-hoc optimization strategies have been thoroughly quantified to find an optimal trade-off between speed and accuracy. Our *ShapeDetector* is running in-between 5 and 8Hz for 612×480 pixel video sequences. We validated our approaches using a detailed methodology covering the overall tool location, tip position, and orientation. Approaches have been compared and ranked conjointly with a set of competitive baselines. For suction tube detections, we achieved a 15% miss-rate at 10^{-1} FPPI, compared to a 55% miss-rate for the *adapted SquaresChnFtrs*. Future works should be directed toward the integration of 3D feature extraction to improve detection performance but also toward the refinement of the semantic labelling step. Coupling the tool detection task to the tool classification in one single framework should be further investigated. Finally, increasing the data-set in diversity, number of tool classes, and detail of annotations is of interest.

Keywords: Object detection, template matching, surgical tools, microscope images.