# Stochastic Methods for Fine-Grained Image Segmentation and Uncertainty Estimation in Computer Vision

Philipe Ambrozio Dias
*Marquette University*

# STOCHASTIC METHODS FOR FINE-GRAINED IMAGE SEGMENTATION AND UNCERTAINTY ESTIMATION IN COMPUTER VISION

by

Philipe Ambrozio Dias, B.S., M.S.

A Dissertation submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy

Milwaukee, Wisconsin

December 2020

ABSTRACT
STOCHASTIC METHODS FOR FINE-GRAINED IMAGE SEGMENTATION AND UNCERTAINTY ESTIMATION IN COMPUTER VISION

Philipe Ambrozio Dias, B.S., M.S.

Marquette University, 2020

In this dissertation, we exploit concepts of probability theory, stochastic methods and machine learning to address three existing limitations of deep learning-based models for image understanding. First, although convolutional neural networks (CNN) have substantially improved the state of the art in image understanding, conventional CNNs provide segmentation masks that poorly adhere to object boundaries, a critical limitation for many potential applications. Second, training deep learning models requires large amounts of carefully selected and annotated data, but large-scale annotation of image segmentation datasets is often prohibitively expensive. And third, conventional deep learning models also lack the capability of uncertainty estimation, which compromises both decision making and model interpretability.

To address these limitations, we introduce the *Region Growing Refinement (RGR)* algorithm, an unsupervised post-processing algorithm that exploits Monte Carlo sampling and pixel similarities to propagate high-confidence labels into regions of low-confidence classification. The *probabilistic Region Growing Refinement (pRGR)* provides RGR with a rigorous mathematical foundation that exploits concepts of Bayesian estimation and variance reduction techniques. Experiments demonstrate both the effectiveness of (p)RGR for the refinement of segmentation predictions, as well as its suitability for uncertainty estimation, since its variance estimates obtained in the Monte Carlo iterations are highly correlated with segmentation accuracy.

We also introduce FreeLabel, an intuitive open-source web interface that exploits RGR to allow users to obtain high-quality segmentation masks with just a few freehand scribbles, in a matter of seconds. Designed to benefit the computer vision community, FreeLabel can be used for both crowdsourced or private annotation and has a modular structure that can be easily adapted for any image dataset.

The practical relevance of methods developed in this dissertation are illustrated through applications on agricultural and healthcare-related domains. We have combined RGR and modern CNNs for fine segmentation of fruit flowers, motivated by the importance of automated bloom intensity estimation for optimization of fruit orchard management and, possibly, automatizing procedures such as flower thinning and pollination. We also exploited an early version of FreeLabel to annotate novel datasets for segmentation of fruit flowers, which are currently publicly available. Finally, this dissertation also describes works on fine segmentation and gaze estimation for images collected from assisted living environments, with the ultimate goal of assisting geriatricians in evaluating health status of patients in such facilities.

## ACKNOWLEDGEMENTS

Throughout these four years of doctoral studies, I had the privilege of exchanging experiences with many remarkable people, in multiple parts of the world.

First, I would like to thank my advisor Dr. Henry Medeiros, who provided me with all the support needed for a successful period of studies. In contrast to many unfortunate stories of absent and/or abusive advisor-advisee relationships, I would like to thank Henry for his respect, trust, kindness during these years of partnership.

I would like to express my gratitude to Dr. Amy Tabb, Dr. Francesca Odone, Dr. Dong Hye Ye, and Dr. Richard Povinelli for being part of my dissertation committee and providing greatly beneficial feedback, despite the time constraints in such a complicated period. My special thanks to Amy, for her mentorship as well as many valuable career/life advices, as well as Francesca for the mentorship and kind reception during my time at the University of Genoa (Italy).

Within the Marquette community, I would like to thank Tony for the conversations that brightened many stressful days during these years, as well as Ms. Katie Tarara, for her work and her kind, compassionate approach towards students of the EECE department. I am also very grateful to all my colleagues during these four years at the COVISS lab. Thank you Andres, Yev, Sam, Reza, Siddique, Jamir, Scott, Zhou, Paris, German, and Miguel.

Many people outside the university were crucial for my academic success and quality of life during this process. I would like to thank all my friends from soccer in Milwaukee, with special thanks to Jameson, Cesar, and Dan who helped me in many occasions. My gratitude also towards all my italian colleagues for receiving me so well in Genova, in special Enrico, Francesco, Vanessa and Veronica for making me feel at home.

Moreover, I would like to thank Xie, for all her support and affection over the last year, as well as Sydney and Alexandre Martins for making my life in Milwaukee

much better. A special thanks to Alexandre, for his support and friendship that were crucial for my success and continually make me a better human being.

My eternal love and gratitude towards all my family and friends back in Brazil and Germany, who have been at my side during all my journey despite the difficulties related to the geographical distance.

Finally, I would specially like to thank my parents Elisabeth and Ademir for their love and for supporting me in all my decisions in such an international journey, with immeasurable efforts that I will be always indebted for.

**Dedication**

To Mother Nature, Pachamama, Gaia, Universe or any of the possible nicknames for the beautifully complex and unified system of which we are part of.

May you continue to inspire me. May I have the resilience and ability to apply my capacities for the benefit of society and of nature. May the difficult times we are currently facing, and the ones that are yet to come, teach us to live according to you, and to do our best to show you some gratitude for the privileges we experience as living rational beings within such a complex chain of events.

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1
# INTRODUCTION

Computer vision is an interdisciplinary field that focuses on extracting meaningful information from images and videos. Information obtained through image understanding can be used for several applications, ranging from activity recognition [11] to object tracking [12], autonomous navigation [13], security and surveillance [14], among many others.

One of the most important aspects of image understanding is the identification of the objects present in an image, which can be carried out at different levels of granularity. In this context, there are four well-known subproblems of image understanding: image classification, object detection, semantic segmentation, and instance segmentation. Figure 1.1 illustrates each of these tasks.

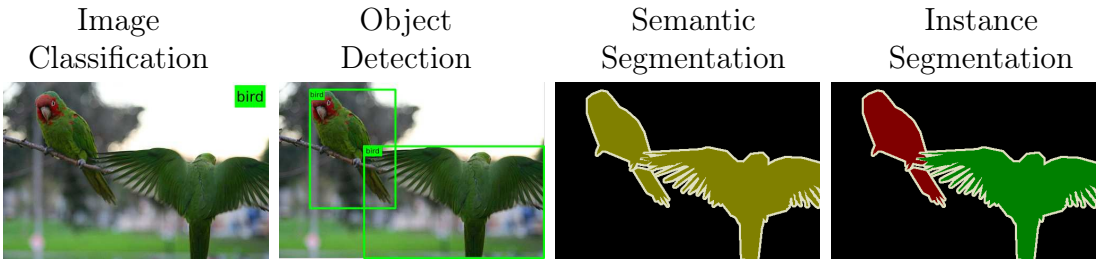



Figure 1.1: Examples of annotations for each image understanding task, for an image composing the PASCAL VOC dataset [3].

The objective of image classification is to label the scene at the image level, or, in simpler terms, identify *what* is present in an image. Meanwhile, object detection attempts to fit bounding boxes around specific objects, which in terms of granularity can be seen as the simplest answer to the question of *where* each meaningful entity

is located in the scene. Segmentation tasks aim at pixel-wise classification of known objects or of different instances of these objects. While semantic segmentation focuses on labeling each pixel according to the *class* that the corresponding object belongs to, instance segmentation requires unique labels for each instance of the given objects present in the scene.

In recent years, systems based on deep learning have remarkably improved the state of the art in many computer vision tasks. The combination of deep Convolutional Neural Networks (CNN) and increasingly larger publicly available datasets has led to substantial improvements to image classification techniques [4]. Currently, CNNs are at the core of facial recognition algorithms used in social networks [15], drones capable of following a person [16], and self-driving cars [13] – all applications considered unfeasible a decade ago.

This dissertation aims to contribute to the advancement of computer vision methods based on deep convolutional neural networks. More specifically, it intends to address the three problems described below.

## 1.1 Problem statement #1

***For segmentation tasks, the performance of conventional CNN architectures is intrinsically limited, providing segmentation masks that poorly adhere to object boundaries***. CNNs exploit downsampling strategies to learn hierarchical features (from low-level to high-level features), an operation that compromises pixel-level details and ultimately leads to imprecise, coarse segmentation in scenarios that require pixel-wise predictions.

For many potential applications of computer vision, segmentations with high boundary adherence are crucial for correct scene interpretations. In action and activity recognition, relevant visual cues for human-human and human-object interactions include contact between agent and object, particular body silhouettes and orientation,

and locations of body parts such as the hands [11, 17, 18]. Moreover, many automation tasks often require manipulation of objects or instruments, where the quality of object pose and morphology estimation directly impact success rate [19,20]. The wide range of image segmentation applications also includes image editing, self-driving vehicles [13], virtual clothing try-on for online shopping [21], and medical imaging. As explained in detail in Objective 1 below, we intend to use local image information and stochastic techniques to improve boundary adherence of segmentations provided by modern CNNs.

## 1.2 Problem statement #2

***While abundant and reliable data has been crucial for the advances on image understanding tasks achieved by deep learning models, large-scale annotation of image segmentation datasets is often prohibitively expensive.*** Depending on the image understanding task, the required dataset annotations may range from tags at the image level (image classification), to bounding boxes (object detection) or pixel-level annotations (image segmentation). For all cases, varied and high-quality image annotations are crucial for both training and evaluation of models that are accurate and robust.

Currently, most Convolutional Neural Network (CNN) models successful at image understanding tasks [22–24] are pre-trained on the ImageNet [25] and COCO [5] datasets, due to their large variability. However, manually labeling large datasets is challenging and time-consuming. As an example, the compilation of the COCO dataset – one of the largest and most popular datasets for object detection and image segmentation – required $55k$ worker hours for annotation of instance segmentation labels. As explained in detail in Objective 2 below, we intend to exploit outcomes from our first objective to design an open-source image annotation interface that generates high-quality segmentation masks from simple and fast user-provided inputs.

## 1.3 Problem statement #3

***Conventional deep learning models do not provide measurements of uncertainties of their own estimations, an information that is tightly associated with model robustness and interpretability.*** Models based on representation learning are commonly portrayed as black-boxes: since they comprise a very large amount of parameters that are optimized based on the data, for the most part, they do not offer sufficient insights to understand how the given task is being solved. Such lack of interpretability further demonstrates the importance of uncertainty estimation to indicate the reliability of a model in different scenarios.

For many real-world applications, high-quality uncertainty estimation can therefore be as crucial as task accuracy for decision making. Self-driving vehicles are an example: to increase the overall system's robustness to challenging scenarios for detection of pedestrians and other obstacles, assorted types of sensors are exploited. In such cases, uncertainty estimates determines the reliability of the predictions provided by each component, and therefore guide critical decision making that can be fatal in case of failures. As explained in detail in Objective 3 below, we intend to apply modern concepts of Bayesian deep learning as well as techniques from stochastic processing to design image understanding frameworks that are capable of estimating uncertainties associated with their own predictions, with special emphasis on image segmentation tasks.

## 1.4 Objectives

This dissertation has three specific research objectives, each associated with one of the three problems stated above: image segmentation, image annotation and uncertainty estimation.

### 1.4.1 Specific objectives

**Objective 1: Devise and apply algorithms for semantic segmentation refinement**

Within the image segmentation domain, this work proposes to advance the field in two principal aspects. First, we envision a general unsupervised post-processing algorithm for segmentation refinement that, different from pre-existing techniques, does not require any task- or dataset-specific training. Second, we plan to combine this novel algorithm with modern semantic segmentation networks for novel real-world applications where segmentations with high-quality at pixel-level are crucial for task automation.

**Objective 2: Devise alternative methods that facilitate the annotation of image segmentation datasets**

In the context of image annotation, this dissertation proposes the development of an open-source image annotation tool that facilitates the compilation of high-quality segmentation masks. To benefit the computer vision community, we envision an interface that can be used for both crowdsourced or private annotation, with a modular structure that can be easily adapted for any image dataset.

**Objective 3: Devise mechanisms for uncertainty estimation from deep learning-based predictions**

Finally, to indicate the reliability of semantic segmentation predictions in its many possible application scenarios, we envision designing the post-processing algorithm proposed in Objective 1 in such a way that, in addition to refined segmentation labels, it also outputs its own confidence on the quality of the refined score for each pixel in the image. This information can then be exploited to devise uncertainty estimation metrics that shall ideally show a high correlation with actual segmentation

quality, indicating potential scenarios of failure and guiding further model improvements.

### 1.5 Dissertation timeline

So far, the outcomes from the research described in this dissertation have been published in the form of the six following peer-reviewed manuscripts, with an additional manuscript currently under review:

- [1] P. A. Dias, A. Tabb, and H. Medeiros, "Apple flower detection using deep convolutional networks," Computers in Industry, vol. 99, pp. 17-28, 2018.
- [2] P. Dias, H. Medeiros, and F. Odone, "Fine segmentation for Activity of Daily Living analysis in a wide-angle multi-camera set-up," in 5th Activity Monitoring by Multiple Distributed Sensing Workshop (AMMDS) in conjunction with British Machine Vision Conference, 2017.
- [3] P. A. Dias and H. Medeiros, "Semantic segmentation refinement by Monte Carlo region growing of high confidence detections," in Asian Conference on Computer Vision. Springer, 2018, pp. 131-146.
- [4] P. A. Dias, A. Tabb, and H. Medeiros, "Multispecies fruit flower detection using a refined semantic segmentation network," IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3003-3010, 2018.
- [5] P. A. Dias, Z. Shen, A. Tabb, and H. Medeiros, "FreeLabel: A Publicly Available Annotation Tool based on Freehand Traces," in Winter Conference on Applciations of Computer Vision (WACV), 2019.
- [6] P. A. Dias, D. Malafronte, H. Medeiros, and F. Odone, "Gaze Estimation for Assisted Living Environments," in Winter Conference on Applications of Computer Vision (WACV), 2020.
- [7] P. A. Dias and H. Medeiros, "Probabilistic semantic segmentation refinement by monte carlo region growing," arXiv preprint arXiv:2005.05856, 2020.

Figure 1.2 provides a timeline depicting the chronological order in which these works were developed and our plan for future research. Figure 1.3 illustrates how the concepts exploited and developed in this work are related to one another and to the objectives of this dissertation.

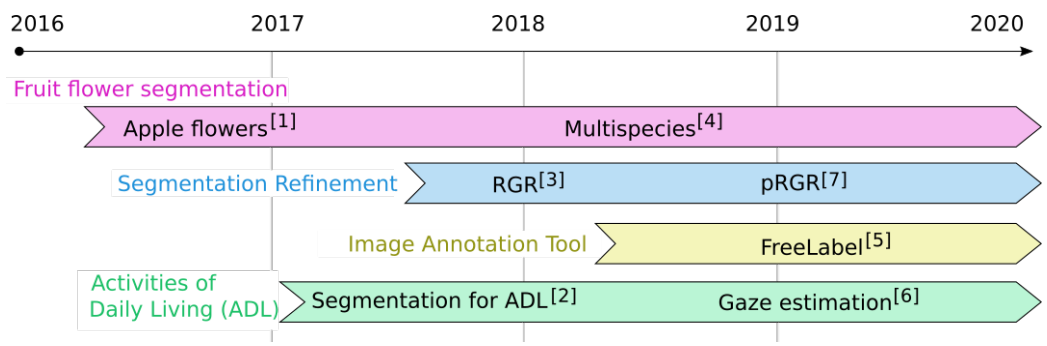



Figure 1.2: Timeline depicting the chronological order in which the works composing this dissertation were developed.

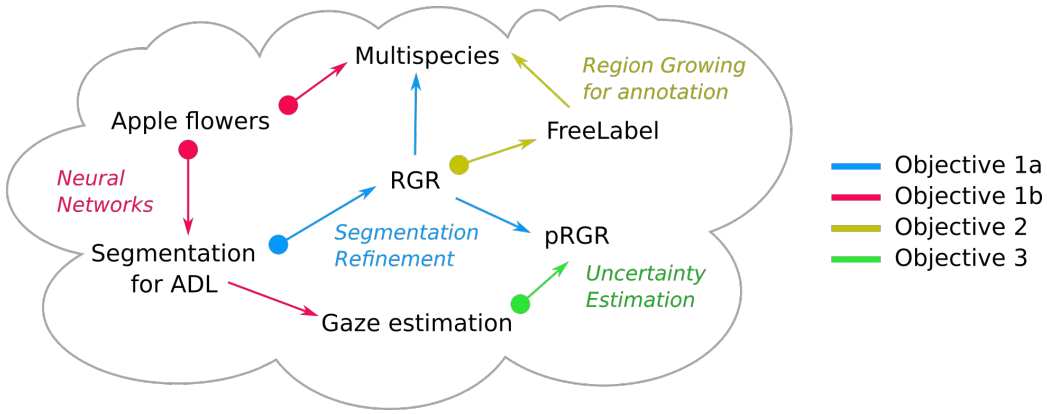



Figure 1.3: Illustration of how the concepts exploited and developed in this work are related to one another and to the objectives proposed for this dissertation. Solid circles indicate the first contact with the corresponding topics, while arrows indicate the connection and chronological order in which these concepts were exploited across the different works. Colors and legend on the right side of the image indicate the objectives related to each work/concept.

With financial and technical support from our collaborators at the United States Department of Agriculture (USDA), the segmentation of fruit flowers has been exploited as a case study for the application of the methods investigated and developed in this research. Critical crop management decisions in fruit production are guided by the number of flowers present in an orchard, but bloom intensity is still typically estimated by means of human visual inspection [26]. Previous automated computer vision systems for flower identification were based on hand-engineered techniques that work only under specific conditions and with limited performance. To the best of our knowledge, our work in [1] was the first to employ CNNs for flower detection, combining superpixel-based region proposals with a classification network to detect apple flowers.

The second application domain of this research is part of a collaboration with the University of Genoa, in which we have been investigating vision-based strategies to monitor automatically and unobtrusively the health status of patients in an assisted living environment equipped with cameras. Activities of Daily Living assessments can be done by observing how patients interact with one another as well as with surrounding objects, which requires addressing multiple tasks such as pose estimation, object segmentation and gaze estimation. After our first work on apple flower segmentation, we employed in [2] similar techniques for semantic segmentation in assisted living environments, combining a CNN and knowledge about camera configuration to extract segmentations of people and objects of interest.

This latter work provided insights that led to the development of the Region Growing Refinement (RGR) module [3], a general post-processing algorithm for segmentation refinement that can be considered a core element of this dissertation. From the segmentation scores provided by any CNN, RGR samples multiple sets of seeds from within regions classified with high confidence by the CNN, propagating their labels into regions where the classification is uncertain. This is done by region growing

based on pixel spatial and color similarity, a quick and effective process that allows increasing boundary adherence of originally coarse segmentation masks.

In [4], we exploited RGR and modern end-to-end residual CNNs to increase the segmentation quality of our original model for flower segmentation, in addition to extending it for segmentation of multiple species of flowers without requiring any preprocessing or dataset-specific training. Since the segmentation of fruit flowers is a rather new domain for application of computer vision and deep learning techniques, our research also included the collection and processing of multi-species fruit flower datasets [27].

We leveraged the knowledge gathered during the annotation of flower datasets to design FreeLabel [5], a web-based tool that allows user to trace lines or "freehand" scribbles of different thicknesses for the different categories present in an image. These scribbles are propagated to the remaining unlabeled pixels using the RGR algorithm, since RGR has the advantages of being fully unsupervised (thus category agnostic), simple to implement, with computational time and parameterization that allow quick and simple user interactions. Our tool allows users to obtain high-quality segmentation masks with just a few freehand scribbles, in a matter of seconds.

Described in [6], a more recent development for our second application domain consisted of a simple neural network regressor that estimates the gaze direction of individuals in a multi-camera assisted living scenario. In conjunction with object detection [2], gaze direction could define mutual relationships between objects and their users (e.g. the user is sitting on a chair with a book on his/her lap vs. sitting on a chair reading the book) and classify simple actions (e.g. mopping the floor, reading a book). We propose an approach that relies solely on the relative positions of facial keypoints to estimate gaze direction, with these features extracted using an off-the-shelf model. From the perspective of the overall framework for ADL analysis, leveraging the facial keypoints is beneficial because a single feature extractor module

can be used for two required tasks: pose estimation and gaze estimation. Moreover, this work represented a first contact and application of concepts of uncertainty estimation for neural networks. Gaze estimation is a task with levels of difficulty that vary according to the scenario of observation. Even for humans, it is much easier to tell where someone is looking if a full-view of the subject's face is available, while the task becomes significantly more challenging when the subject is facing backwards with respect to the observer's point of view. For this reason, we leverage concepts used by Bayesian neural networks to design a model that provides an estimation of its uncertainty for each prediction of gaze direction.

More recently, we developed the *probabilistic Region Growing Refinement* (pRGR) algorithm [7], an extension of RGR that provides it with a solid mathematical foundation that exploits a probabilistic framework to guide all the steps of the algorithm. Combining techniques from Bayesian estimation, many parameters that were previously determined in an ad-hoc manner are now initialized using Bayesian conjugate priors and updated as assignments of pixels to clusters occur. Moreover, variance reduction techniques are exploited to optimize the sampling steps within the Monte Carlo refinement iterations, and a novel parameterization allows for the emulation of varied receptive field sizes, such that pRGR further improves segmentation refinement performance by recovering finer boundary details and attenuating the effects of false-positive pixel labels. In [7], we also provide an important contribution in the context of uncertainty estimation, as we experimentally demonstrate that the variance of pRGR's Monte Carlo estimations can be exploited as an uncertainty estimation mechanism that is highly correlated with segmentation accuracy values.

In summary, this dissertation details a collection of methods that were motivated by challenges faced when applying the most modern image understanding techniques for real-world applications. In addition to filling gaps such as extending deep learning methods for novel scenarios that include fruit flower segmentation and

analysis of activities of daily living, this application-oriented research strategy provided us with insights that led to the design of novel methods that are rather general, advancing the state of the art on segmentation refinement, image annotation, and uncertainty estimation.

### 1.5.1 Dissertation organization

This dissertation consists of seven chapters. Chapter 1 contains a brief introduction to the field of computer vision and the three main problems addressed by methods proposed in this dissertation. In Chapter 2, we provide background information on basic concepts of computer vision, machine learning (including neural networks), probability theory, stochastic methods, as well as an overview of the main datasets and methods proposed in the literature for image semantic segmentation, and a summary of existing techniques for uncertainty estimation for computer vision tasks. Chapter 3 describes our proposed techniques for segmentation refinement, namely the RGR and the pRGR algorithms. In Chapter 4, we describe FreeLabel, our open-source, web-based interface for annotation of image segmentation datasets. Chapter 5 focuses on our first application scenario, describing our two works for fine segmentation of fruit flowers. Chapter 6 describes our proposed techniques for fine image segmentation and gaze estimation for assisted living environments, which is the second application scenario for techniques developed in this dissertation. Finally, Chapter 7 concludes this dissertation with an overview of its findings and a discussion of possible directions for future work.

# CHAPTER 2
# BACKGROUND

## 2.1 Basic concepts of computer vision

### 2.1.1 Image representation

Digital images are composed of *picture elements* known as *pixels*, typically arranged in the form of 2D spatial arrays as illustrated in Figure 2.1. Spatial coordinates are commonly referenced to as $x$ and $y$ coordinates, corresponding to column and row positions in this 2D array, respectively. The *resolution* or size of an image is most commonly expressed in the form $height \times width$, i.e., the number of rows times the number of columns composing the image array. Hence, a gray-scale image $I$ is represented in a $I \in \mathbb{R}^{w \times h}$ domain.

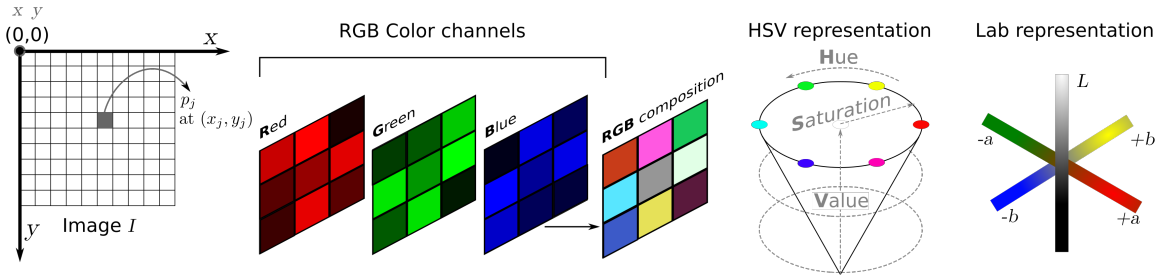



Figure 2.1: Visualization of the representation of images as arrays, as well as illustrations of the RGB, HSV and CIELab color representations.

#### Color spaces

Color can be represented using a variety of *color spaces*. The most well-known representation is the *RGB*, where colors are represented as combinations of

*Red, Green* and *Blue* components or *color channels*. As illustrated in Figure 2.1, representations of RGB images consist of three 2D arrays of the same size $H \times W$, each array corresponding to a color channel.

While the RGB representation is convenient for printing and digital visualization, alternative color representations are frequently used in computer vision tasks to better emulate the human perception of colors. The HSV color-space describes colors in terms of *Hue*, *Saturation* and *Value* components. It has the advantage of dissociating brightness (expressed as *value*) from chromaticity (*hue*) and saturation. Intuitively, chromaticity can be understood as the color tone, while saturation indicates the purity of the color [28]. Studies on human vision and color-based image retrieval have demonstrated that most of the color information is contained in the hue channel, with saturation playing a significant role in applications where identifying white (or black) objects is important [29, 30].

Among many other representations introduced by the *Commission Internationale de l'Éclairage (CIE)*, the CIELab color space, illustrated in Figure 2.1, expresses colors in terms of three channels: *lightness*, where 0 corresponds to black and 100 indicates diffuse white; channel $a$ indicating color position in a range between red/magenta and green (negative values indicate green while positive values indicate magenta); and channel $b$ indicating color position between yellow and blue (negative values indicate blue and positive values indicate yellow). Designed to approximate human vision, the CIELab representation is a relatively perceptually-uniform space [31], such that Euclidean distances between any two colors in this space are commonly considered good approximations of human perception differences.

### 2.1.2 Features

In computer vision and image processing, *feature* corresponds to information that is meaningful for describing an image and its regions of interest for further pro-

cessing. Feature extraction is therefore crucial in image analysis, since it represents the transition from pictorial (qualitative) to nonpictorial (quantitative) data representation [28]. Starting from representation and analysis at pixel-level, features are commonly described as *low-level* when they describe patterns at each individual pixel and/or only its immediate neighbors. This can range up to analysis and descriptions of regions that cover most (or the whole) of an image, which are then referred to as *high-level* features. The concept of low-, mid- up to high-level features is particularly relevant for understanding the success of modern methods that are capable of learning *hierarchical features*, where descriptors at a higher-level can be obtained through the successive combination of lower-level descriptors. We provide in Section 2.3 a more detailed explanation on hierarchical features, in the context of modern convolutional neural networks.

### 2.1.3 Basic concepts on image processing

Approaches based on mathematical morphology compose an important subset of traditional image processing techniques [28,29]. Operations are typically performed in local neighborhoods around pixels, which can be of variable sizes and shapes according to designed *structuring elements* or "windows". Basic operations such as erosion and dilation have the effect of "growing" or "shrinking" objects in a binary image, and can be combined into operations such as opening and closing to fill holes or open weakly-connected objects, or image enhancement techniques as top-hat and bottom-hat operations that combines opening and closing procedures to enhance contrast and details in presence of shading [29].

#### Kernels and convolutional filters

Other popular operations performed at local neighborhoods are filtering using *kernels* or *convolutional filters*. They can range from simpler strategies such as basic

Gaussian kernels for image smoothing and Laplacian kernels for edge detection, up to more complex hand-engineered *wavelets* for analysis of textures and other patterns of relevance [29]. Analogously to signal processing operations on 1D signals, convolutional filters are applied over the whole image in a *sliding window* fashion, a procedure exploited by modern approaches described in the next sections.

### Histograms transformations and thresholding

While the basic concepts behind these techniques are easier to understand using binary and gray-scale images as examples, most of them are naturally extended to analysis of color images. In this domain, image processing techniques using histogram representations are also very common. Exemplified in Figure 2.2, histogram equalization aims at spreading the histogram components to improve image contrast, while histogram matching consists in approximating its distribution to the characteristic form of a pre-existent reference distribution [28], which can be of particular relevance to aid computer vision algorithms with robustness to variation on image acquisition conditions.

Moreover, color *thresholding* is one of the most basic approaches for identification of objects or regions of interest, where pixels are labeled according to intensity values larger or lower than pre-defined values named threshold [28]. Figure 2.3 illustrates the output of a thresholding operation on the hue channel of the input image.

### Geometric transformations

In contrast to such operations that alter intensity values of pixels, another set of image processing techniques known as geometric operations focus instead on altering the spatial relationship between pixels. Studies and techniques on geometry for computer vision constitute an important and vast field of research, with the *"Multiple*

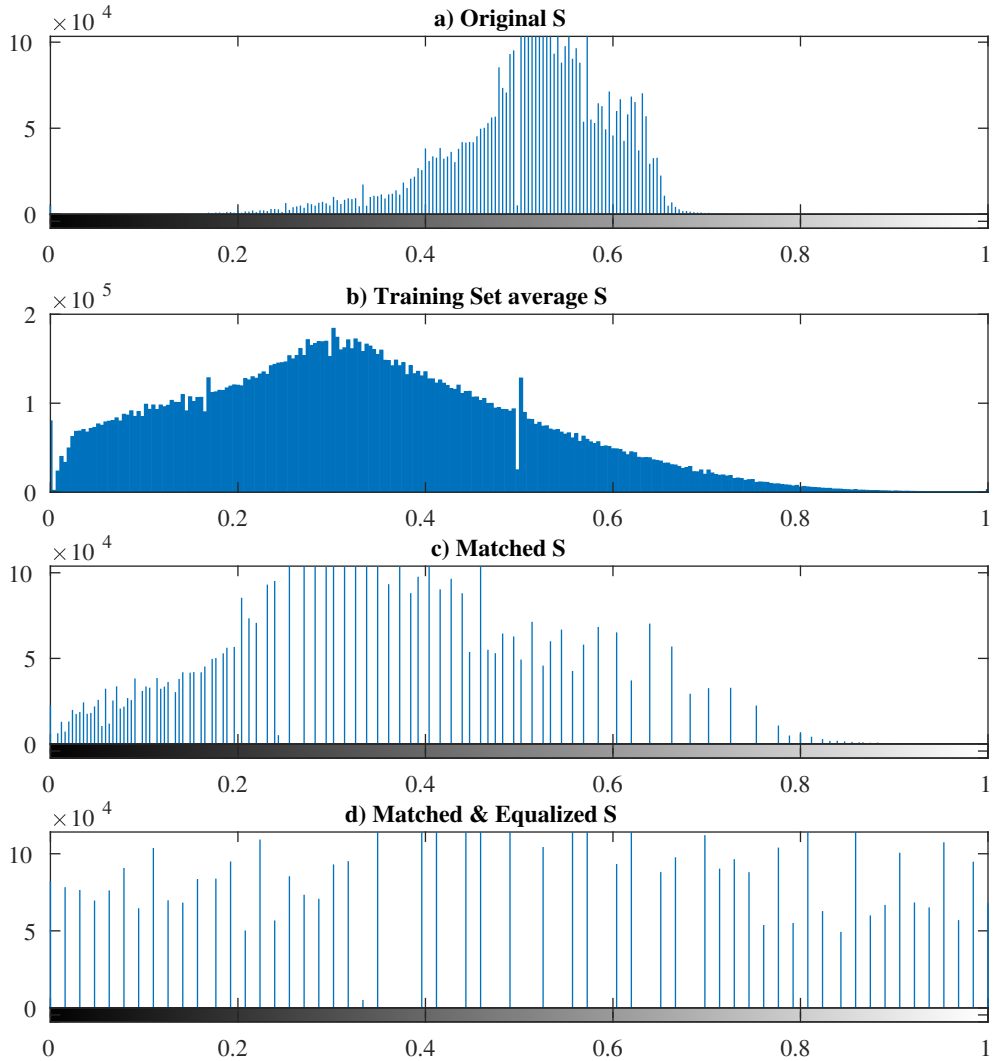



Figure 2.2: Example of histogram matching and equalization. Histogram c) is obtained by matching a) to b), while histogram d) is the result of equalizing histogram c).

*view geometry in computer vision"* book by Hartley & Zisserman [32] as a widely used reference discussing its major concepts. For this dissertation, the following concepts are of particular relevance to understand modern state-of-the-art techniques as well as novel approaches herein introduced.

Figure 2.3: Example of image thresholding. *Left:* input image; *middle:* hue channel after transforming the image to the HSV color space; *right:* binary image obtained by thresholding the hue channel.

As summarized in [29], geometric transformations consist of two main operations: i) a spatial transformation of coordinates, and ii) an interpolation of intensity values that define final values of transformed pixels. Spatial transformations known as *scaling, rotation, translation* and *shearing* form a set of coordinate transformations referred to as *affine* transformations, which can be formulated using affine or transformation matrix such as the one in Eq. 2.1.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{2.1}$$

where $x$, $y$, and $z$ are the original coordinates of the original image point in homogeneous form [32], $t_{ij}$ are the coefficients of the transformation matrix, and $x'$, $y'$, and $z'$ are the coordinates of the transformed point. In general terms, affine transformations preserve linear relationships between points, straight lines and planes, such that a given pair of parallel lines remains parallel after the transformation. Figure 2.4 illustrates each transformation, with the corresponding parameterization of transformation matrices for each case.

As described in following sections, the concept of *invariance* to affine transformations has been of great importance for the development of computer vision

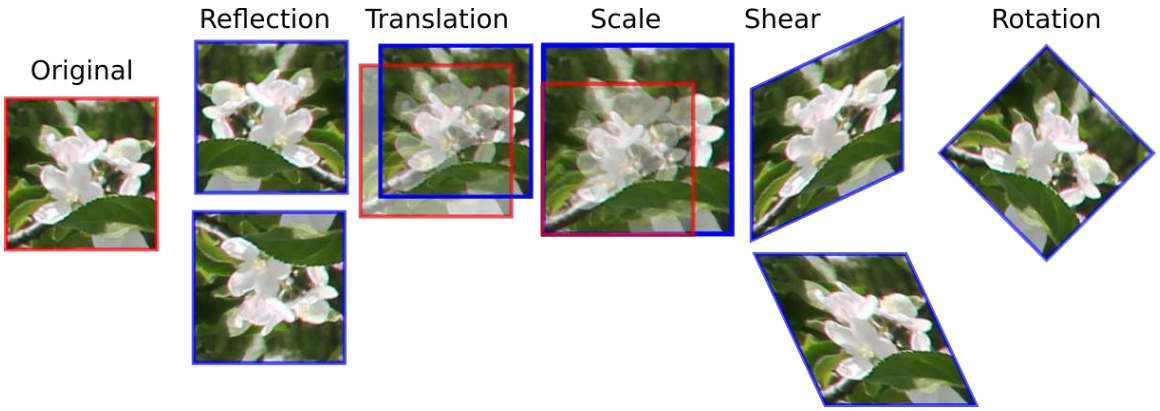



Figure 2.4: Illustration of different affine transformations.

algorithms that aim at robustness against different acquisition conditions. The intuition for such cases is that, ideally, a descriptor of an object or any entity of interest should provide the same output regardless if the entity is subjected to translation, rotation or other affine transformations.

## 2.2 Basic concepts of machine learning and pattern recognition

In this section, we provide definitions of techniques and concepts of machine learning and pattern recognition that are exploited in many modern computer vision systems.

### 2.2.1 Machine learning

As defined in [33], machine learning algorithms can be defined as *"computational methods using experience to improve performance or to make accurate predictions"*. In this context, experience refers to information already accessible to the learner, and according to the different types of experience, the learning scenarios can be divided in three main categories:

- *supervised* learning: scenario where models (or "learners") are trained using data examples for which labels or ideal output values are available, targeting good predictions on unseen data points;
- *unsupervised* learning: in contrast to the supervised case, in this scenario models are designed to learn meaningful information from input data without accessing output labels;
- *reinforcement* learning: models are designed to learn based on experience, by means of interactions with an environment that yields associated rewards. In this scenario, learners must be designed to properly balance between exploiting information already available versus exploring novel, unknown domains.

**Data subsets: training, validation, testing**

Since the main goal of machine learning algorithms is to provide accurate predictions for unseen inputs, datasets used for designing and evaluating efficient algorithms are commonly structured with *training, validation* and *testing* subsets. Compared to the training samples used to learn internal model parameters (such as weights), the validation samples are used to tune parameters commonly known as *hyperparameters*, which are free parameters that guide the learning process. Ideally, parameters adjusted using a validation set will allow learned models to generalize well to the fully unseen testing samples, which are ultimately used to evaluate the performance of models in the task of interest.

In cases where limited data is available, cross-validation strategies are of particular relevance. In such scenarios, a single subset is randomly divided into $N$ folds (i.e., partitions) of equal sizes, and $N$ iterations are performed such that each fold is used exactly once as validation data. At the end, the final performance measure corresponds to the average of the values obtained in each iteration, with the goal

of better assessing how well predictions will generalize to an independent data set. Figure 2.5 illustrates a cross-validation process using $N = 10$ folds.

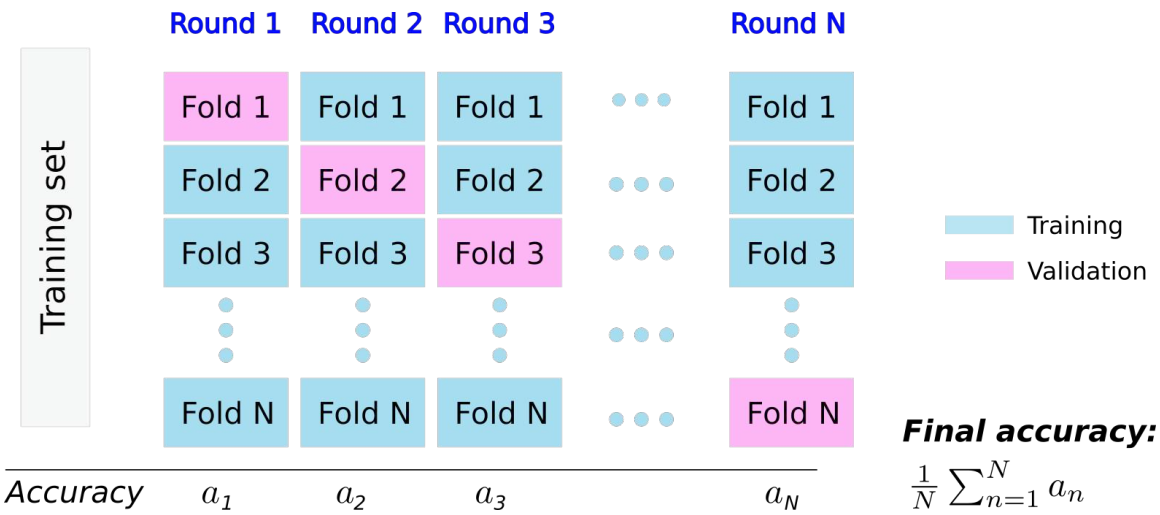



Figure 2.5: Illustration of a cross-validation process for N folds. Drawing inspired on [1]

### Overfitting, underfitting and model complexity

Proper selection of training, validation and testing subsets critically impacts model behaviors known as *overfitting* or *underfitting* of data. These problems typically occur when the data subset used for model training has a different distribution from the unseen data composing the target test set. Overfitting refers to the scenario in which the model learns parameter values that fit very well the underlying training distribution but fails to generalize well to the unseen testing subset. The opposite case is underfitting, the scenario in which the learner fails to properly capture the underlying distribution of the training data.

[1] https://scikitlearn.org/stable/modules/cross_validation.html

The concept of model *complexity* is crucial for the design of machine learning models that avoid under/overfitting. Informally, it can be understood as a function of the number of learnable parameters a model has, which in turn defines the size of the hypothesis space a model can exploit for decisions. Learners with more parameters than available training samples tend to overfit data, while models with too few parameters might fail to capture complex structures of data.

**Loss functions and optimization**

Learning is framed as an optimization problem where *loss (or cost) functions* play a critical role. In simple terms, loss functions measure the distance or difference between the outputs predicted by a model and the true outputs or labels [33]. Moreover, loss functions often incorporate *regularization* parameters, whose goal is to reduce the chance of overfitting without increasing the amount of training data or reducing the learner's capacity in terms of learnable parameters.

The optimization process then consists of adjusting the models' parameters such that the corresponding loss function is minimized (or maximized, if formulated in terms of similarity). As such, many optimization approaches are based on methods that resort to derivatives or gradients to reach the optimal cost function value. The most widely known family of optimizers are *gradient descent* methods, which minimize costs by adjusting parameters (or weights) according to the cost (or error) gradients with respect to them. In this context, an important challenge in machine learning processes is the localization of the *global minimum* without settling in *local minima*, which are non-optimal solutions for corresponding problems [34].

Classification problems are usually formulated based on maximum log-likelihood to guide optimization. We detail the favorable aspects of maximum likelihood estimators in Section 2.5. As for the logarithm, some of the main reasons for its widespread usage include the fact that multiplications can be handled in the

form of summations, and numerical problems can be avoided (e.g., very small values that could lead to divisions by zero). Moreover, as the log function monotonically increases, both likelihood and its log-likelihood counterpart share the same maxima and minima.

Building upon the concept of Shannon's Entropy in information theory (we refer the reader to [35] for details), the *cross entropy* function is widely used in loss functions for classification models. Defined according to Eq. 2.2, in a multi-class classification task the cross-entropy $H(\cdot)$ is computed with $y_c$ corresponding to the one-hot label for each class $c$, while $p_c$ is the model's prediction for the corresponding class. As detailed in Section 2.3, cross-entropy functions are a particularly good match for outputs using activation functions of the exponential family (e.g. sigmoid and softmax).

$$H(y, p) = -\sum_{c} y_c log(p_c). \tag{2.2}$$

#### 2.2.1.1 Traditional machine learning approaches

Machine learning algorithms are exploited for multiple application scenarios, with corresponding learning goals commonly categorized into: classification, ranking, regression, dimensionality reduction and clustering [33]. In the following paragraphs, we provide basic definitions of each category, as well as brief introductions to widely used techniques that are of particular relevance for this dissertation.

**Classification**

Classification tasks consist of categorizing each data entry into a predefined class. Linear classifiers are a simple example of classification models, where data points are classified based on linear combinations of available features. That is, for data inputs $\mathbf{x}$ and a set of learnable parameters (or weights) $\mathbf{w}$, a linear model

generates output values $\tilde{y}$ according to Eq. 2.3. Defined as in Eq. 2.4, logistic classifiers yield instead non-linear lines separating classes, by exploiting the *logistic sigmoid* function, represented as $\sigma(\cdot)$ and defined according to Eq. 2.5.

$$\tilde{y} = \mathbf{w}^{\mathbf{T}}\mathbf{x}, \tag{2.3}$$

$$\tilde{y} = \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{x}), \tag{2.4}$$

$$\sigma(a) = \frac{1}{1 + e^{-a}}. \tag{2.5}$$

Perceptrons and *Neural Networks (NNs)* are additional examples of classifier/regressor models closely related to the concepts of linear and logistic regression. Since NNs and their modern derivations are of central importance for this dissertation, we discuss these models in detail in Section 2.3.

First introduced in [36], *Support Vector Machines (SVMs)* are supervised learning models that search for a hyperplane that maximizes the margin distance to each class. As illustrated in Figure 2.6, the optimization process of an SVM model for a classification task does not only focus on finding a "line" (hyperplane) that assigns each training data point to the correct class, but instead a hyperplane whose distances to the nearest points of each class are as large as possible. This characteristic allows SVM models to generalize better than classifiers such as those based on logistic regression.

However, for most applications the data is not linearly separable, requiring SVM formulations using *soft margin*. In such cases, additional variables named *slack variables* (represented with $\xi_i$) are added to measure and allow adjusting the influence of data outliers. Moreover, kernel functions such as the popular radial basis function (RBF, or *Gaussian*) are used within SVMs for handling non-linearly separable data. Basically, the difference between a linear kernel and a RBF one consists in the way distances between samples are computed in the feature space. For two samples $\mathbf{x}$ and

[2]https://gdcoder.com/support-vector-machine-vs-logistic-regression

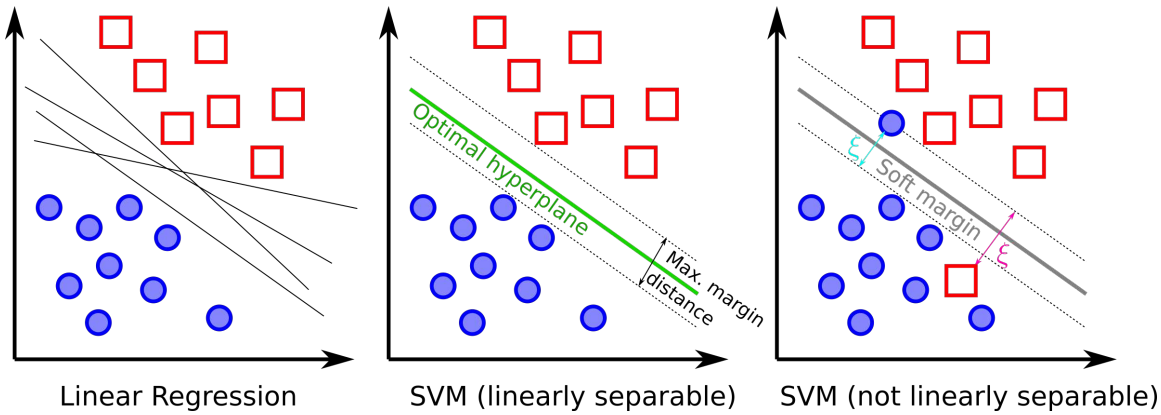



Figure 2.6: Basic representation of SVM approaches for classification (drawing inspired on [2]). Simple regressors are optimized based only on classification error, which is maximum for any of the separation line in the left plot. The SVM selects instead the hyperplane that yields the largest separation margin to all classes. In cases where the data is not linearly separable, soft margins based on the concept of slack variables $\xi$ are learned.

$\mathbf{x}'$, the kernel function is given by $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \cdot \phi(\mathbf{x}')$. A linear kernel computes the distance between them according to Equation 2.6, while a RBF kernel computes it according to Equation 2.7.

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x^T} \cdot \mathbf{x}' + 1), \tag{2.6}$$

$$k(\mathbf{x}, \mathbf{x}') = e^{(-\gamma ||\mathbf{x^T} \cdot \mathbf{x}'||^2)}. \tag{2.7}$$

The weights of an SVM classifier are adjusted according to the following cost function:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i, \qquad \text{subject to: } y_i(\mathbf{w^T} \cdot \phi(\mathbf{x_i}) + b) \geq 1 - \xi_i, \xi_i \geq 0,$$

where $\mathbf{w}$ corresponds again to the vector of weights, $b$ is a bias, $\mathbf{x_i}$ is an input vector (i.e., a sample in the feature space), $\mathbf{y_i}$ are the corresponding labels and $\xi_\mathbf{i}$ can be understood as the error margin that defines whether an example is within the margin or is misclassified.

The regularization cost $C$ and the width $\gamma$ of the Gaussian kernel are the two main parameters controlling the performance of SVMs with a Gaussian kernel function. By regulating the penalty applied to misclassifications, the parameter $C$ controls the trade-off between maximizing the margin with which two classes are separated and the complexity of the separating hyperplane. The parameter $\gamma$ regulates the flexibility of the classifier's hyperplane. For both parameters, excessively large values can lead to overfitting.

The optimization of $C$ and $\gamma$ is a problem without straightforward numerical solution. Therefore, it is typically solved using grid search strategies [37,38] in which multiple parameter combinations are evaluated according to a performance metric. We refer to [37, 38] for further details on the formulation of SVMs.

### Regression

Regression tasks consist of associating a value in the real, continuous domain to each data input. Similarly to classification domain, formulations using linear and logistic functions are common examples of regression algorithms, where the learned functions are used for interpolation of regression values instead of as separation between classes [33, 35].

### Clustering

Clustering tasks focus on grouping data entries according to some similarity criteria. *K-means* is a popular approach for clustering, where entries composing a dataset are grouped into a predefined $K$ number of clusters. Let $X = \{x_1, ...x_N\}$ represent a dataset with $N$ samples, and $S$ represent the set of clusters $\{s_1, .., s_K\}$. Each cluster $s_j \in S$ is initialized at a unique centroid position, and an iterative process is conducted with the goal of assigning each entry $x_i \in X$ to a cluster in $S$, in such a way that an optimization criterion is met. As assignments occur, the

centroid $\mu_j$ of each cluster $s_j$ is updated as the mean of the data points assigned to $s_j$. One common criterion is to minimize the Euclidean distance between data points and cluster centroids, such that the optimization process aims to find the set of clusters that satisfies Eq. 2.8.

$$\arg\min_{S} \sum_{n=1}^{N} \sum_{j=1}^{K} \|x_n - \mu_j\|^2 . \tag{2.8}$$

Hence, the algorithm consists of a two-stage optimization process where data points are iteratively re-assigned to clusters, and clusters centroids (or means) are re-computed as assignments are updated until a convergence criterion is met.

Another well-known optimization strategy used for clustering is the *Expectation-Maximization* (EM) algorithm, which is based instead on a probabilistic framework. As detailed in [35], it is associated with the concept of Gaussian Mixture Models (or GMMs), where a superposition of Gaussian densities is used to capture more complex distributions. For clustering, EM is employed as optimization strategy to learn parameters associated to GMMs describing multiple clusters, such that while K-means employs a hard assignment of data points to a single cluster, GMMs allows soft assignments through its probabilistic formulation.

**Dimensionality reduction**

Principal Component Analysis (PCA) is one of the most widespread techniques for dimensionality reduction. It consists of projecting $N$-dimensional input data onto a $K$-dimensional subspace in such a way that this projection minimizes the reconstruction error (i.e., the $L_2$ norm between the original and the projected data) [33]. The problem is therefore expressed as

$$\min_{\mathbf{P} \in \mathcal{P}_K} \|\mathbf{PX} - \mathbf{X}\|_2^2, \tag{2.9}$$

where $\mathbf{X}$ corresponds to the mean-centered data matrix and $\mathcal{P}_k$ is the set of orthogonal projection matrices. PCA can be performed by computing the eigenvectors and

eigenvalues of the covariance matrix and ranking principal components according to the obtained eigenvalues [39].

## 2.3 Neural networks

Neural Networks were originally designed with the basic working process of the human brain as inspiration. Similar to their biological analogue, they are collections of interconnected activation elements called neurons. As represented in Figure 2.7, these elements consist of a set of learnable parameters named *weights* and *biases*, followed by a pre-defined *activation function* that emulate, in simpler intuition terms, the synapses generated by neurons in our brains. Mathematically, the output or

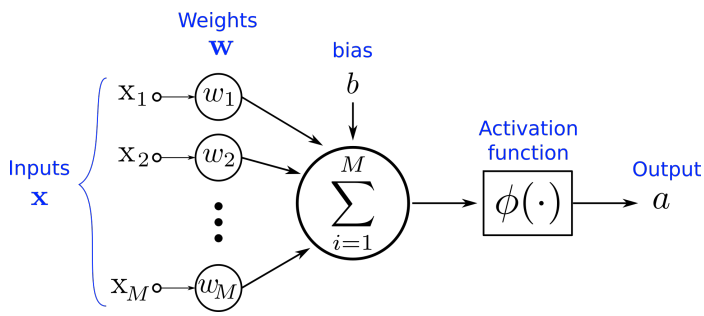



Figure 2.7: Basic structure of an unit or neuron composing a neural network.

activation $a$ of a given neuron can be expressed as in Eq. 2.10, where $\phi$ represents a chosen activation function, while $\mathbf{w}$ and $b$ denote the set of weights and the associated bias parameter, respectively.

$$a = \sum_{i=1}^{M} \phi(\mathbf{w}^{\mathbf{T}}\mathbf{x}) + b. \tag{2.10}$$

Stacked into *layers*, neurons composing artificial NNs are thus interconnected such that, apart from the ones composing input layers, their input signals are "synapses" or activations from each element connected to it. As illustrated in Figure

2.8, the *architecture* of artificial NNs consists of a sequence of layers, most commonly structured in a *feedforward* manner. That is, the information propagates in only one direction, across three main types of layers: an *input layer* where neurons activate according to a given data input (e.g. pixels composing an image); *hidden layers* that compose the middle of the architecture and can be stacked in multiple numbers to increase the network's representation capacity; and finally an *output layer*, whose neurons provide the activations used as final outputs of the model.

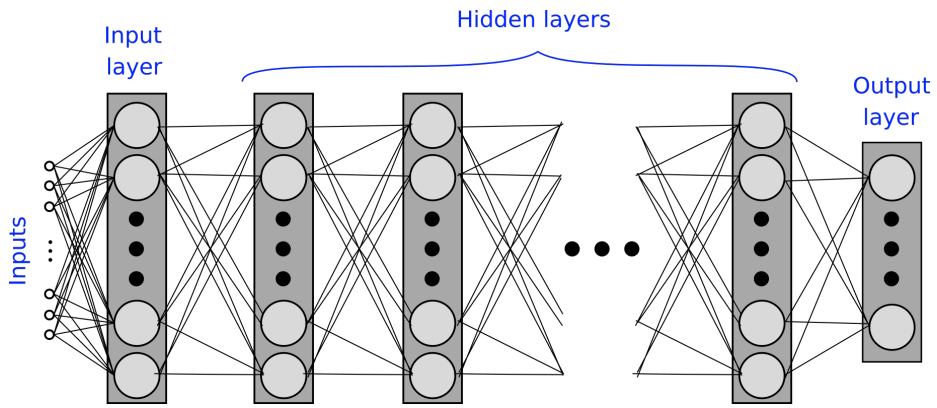



Figure 2.8: Diagram illustrating the basic architecture of a fully connected deep neural network.

### Activation functions

Activation functions are chosen according to the application and position in the NN architecture. A large of set activation functions can be found in modern NNs. We refer the reader to [34] for more details, and describe below the functions employed by models relevant to this dissertation.

The *sigmoids* are a family of S-shaped curves that includes the logistic function defined as Eq. 2.5, with both names frequently used interchangeably. Sigmoids

are commonly used as activation functions for output neurons in binary classification tasks, as it is a non-linear function whose values range are in the interval $[0, 1]$. Similarly, the *softmax* or normalized exponential function is commonly used for scenarios of multi-class categorization. Defined as in Eq. 2.11, it ensures that all outputs of a model add to 1, such that its output can be loosely interpreted as the probabilities that a given input belongs to the corresponding classes [34].

$$\text{softmax}(z) = \frac{e^{-z_i}}{\sum_j e^{-z_j}} \qquad \log \text{softmax}(z) = z_i - log\left(\sum_j e^{-z_j}\right) \tag{2.11}$$

Both sigmoid- and softmax-based output units are usually employed in conjunction with loss functions defined in terms of maximum log-likelihood. As the second part of Eq. 2.11 indicates, the log of this exponential-based function has a linear component corresponding to the input $z_i$, such that its gradient is particularly suitable for learning procedures based on gradient descent: even when the sigmoid or softmax saturates, this linear component ensures a non-zero gradient, and the learning process can thus proceed.

While suitable for output units when paired with log-based losses, sigmoids have a zero derivative for larger absolute input values, such that their usage in hidden layers makes NNs highly susceptible to learning problems related to vanishing gradients. In these cases, *rectifiers* are used to compose the so called *Rectified Linear Units (ReLUs)*. For a given input $z$, the output of a rectifier corresponds simply to $max(0, z_i)$. Hence, its derivative is 1 for values larger than 0, and 0 for negative values (undefined for input equal 0), making ReLUs particularly suitable for hidden layers.

### Deep neural networks

Neural networks with architectures comprising multiple hidden layers are called *deep neural networks*. As commonly described in textbooks [35], neural net-

works are said to be *universal approximators*, with observations such as the ones made in [40] that networks with a single hidden layer have the capacity to approximate any bounded continuous function, while networks containing at least two layers can approximate any function to an arbitrary accuracy.

While the concept was introduced decades ago, deep learning models became feasible relatively recently, after the introduction of large publicly available datasets such as ImageNet [25], of graphics processing units (GPUs), and of training algorithms that exploit GPUs to efficiently handle large amounts of data [34, 41].

### 2.3.1 Convolutional neural networks (CNNs)

Convolutional Neural Networks (CNNs) constitute the majority of current deep learning architectures, especially for computer vision purposes. In contrast to units composing *fully connected layers*, CNNs employ *convolutional filters* to create sparse connections among layers. As an example, let us consider the model introduced in [4] and commonly named "AlexNet", whose architecture is illustrated in Figure 2.9: designed for image processing, its first convolutional input layer employs *kernels* (illustrated in green) that assess only a $5 \times 5$ region of the input image at a time. Each position in the kernel corresponds to a learnable weight that multiplies input values and, as in any other neural unit, is then processed using an activation function (typically a ReLU). Thus, the convolution operation will output a single activation value for each evaluated input region. Similar to the signal processing counterpart operation, the convolutional operation in a CNN evaluates all regions in the input by sliding the kernel over all input data points, with step sizes corresponding to a predefined *stride* value.

Importantly, the kernel weights are shared at every input location, i.e., the same weight values are used for evaluation over the whole input. This strategy provides CNNs with the ability of learning representations that are invariant to trans-

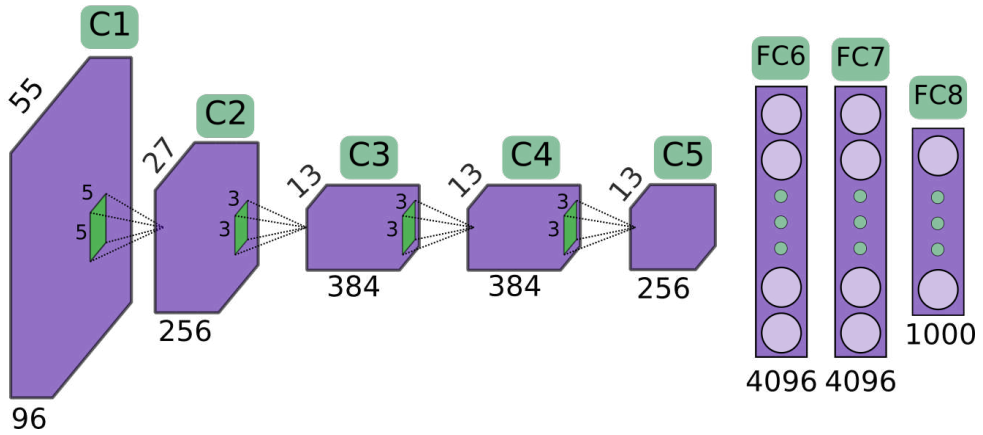



Figure 2.9: Diagram illustrating the architecture of the model introduced in [4], now widely known as AlexNet.

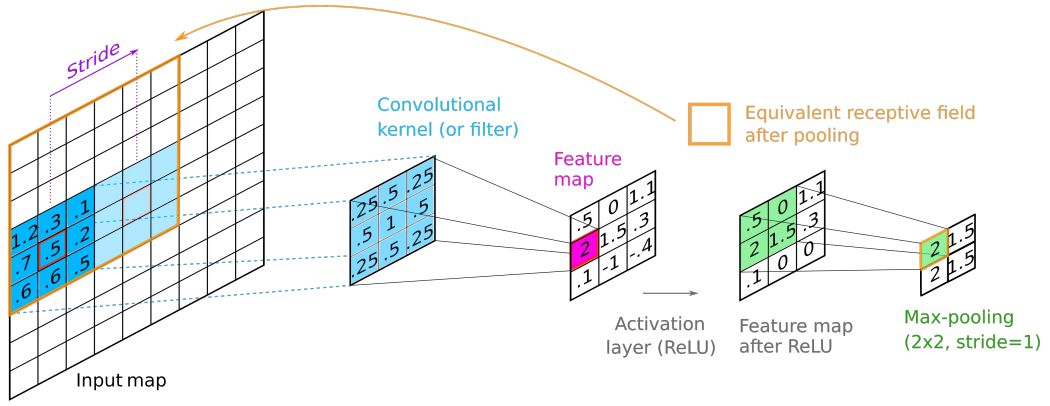



Figure 2.10: Diagram illustrating the typical composition of a convolutional layer, combining a convolutional kernel, followed by an activation function (ReLU) and a max-pooling layer.

lation, a powerful tool to describe sets of complex descriptors as described in detail in the next paragraphs. As Figure 2.10 illustrates, the collection of kernel outputs at the different input locations generates new representations commonly referred to as *feature maps*.

### Pooling

After each convolutional layer, most CNNs employ a *pooling layer*. Pooling is another type of kernel-based operation which, instead of using learnable weights, transform multiple input values into a single output by means of fixed operations like averaging or *max*, with the latter being the most commonly used in CNNs. Figure 2.10 illustrates the typical composition of a convolutional layer, where a learnable kernel is followed by a ReLU layer and also a max-pooling operator.

These operations are employed to increase the model's *receptive field*, i.e., to evaluate increasingly larger regions of the inputs. Intuitively, relying solely on kernels of small sizes (e.g., $5 \times 5$) pixels would lead to models only capable of learning local feature descriptors, insufficient to characterize complex, larger structures. However, increasing kernel sizes would imply more weight parameters to be learned, which in turn tends to make training unfeasible as training data requirements increase. Thus, pooling operations serve as a mechanism to *downsample* input representations and allow the evaluation of increasingly larger receptive fields.

### Hierarchical features

The combination of multiple convolutional layers and downsampling techniques yields deep CNNs with their extraordinary ability to learn *hierarchical features*. They are a key factor for the success of these models in comparison to previous hand-engineered methods described in Section 2.4.3 [41]. As described in [42], the convolutional layers C1-C2 in Figure 2.9 learn to identify low-level features such as corners and other edge/color combinations. The following layers C3-C5 combine this low-level information into more complex structures, such as motifs, object parts and finally entire objects.

**Data augmentation, fine-tuning, transfer learning**

Traditional deep CNNs are composed of millions of parameters: as an example, the early AlexNet [4] contained 60 million parameters. Thus, although many large publicly available datasets have been introduced, gathering domain specific training data to train such deep models is a daunting task. One alternative to reduce the required amount of labeled data is *data augmentation*, a technique used to benefit the training of multiple machine learning models. Data augmentation is typically performed by applying transformations such as translation, rotation and color space shifts to pre-labeled data, as illustrated in Figure 2.11 for an image composing the PASCAL dataset [43].

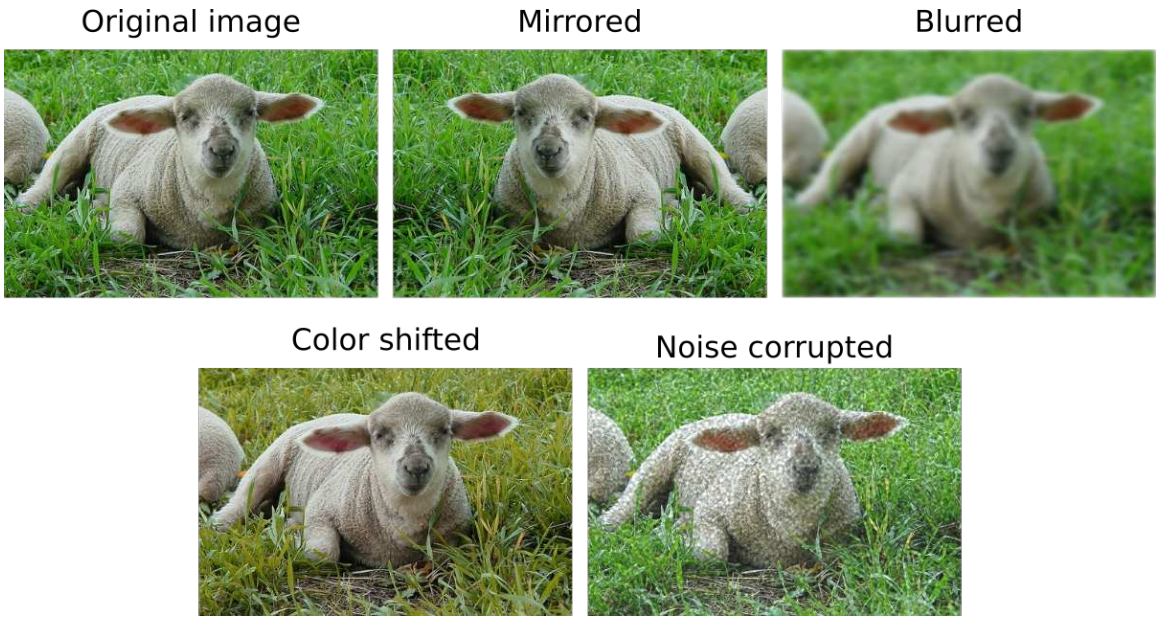



Figure 2.11: Examples of common data augmentation strategies in computer vision, using an image composing the PASCAL dataset [43] for illustration.

In addition, various *transfer learning* approaches such as fine-tuning have been investigated [44, 45]. Earlier layers of a deep neural network tend to contain more generic information (low-level features), which is then combined by the later layers into task specific objects of interest. Thus, a network that can recognize different objects present in a large dataset must contain a set of low-level descriptors robust enough to characterize a wide range of patterns. Under this premise, fine-tuning procedures typically aim at adjusting the higher-level part of a network pre-trained on a large generic dataset, rather than training the full network from scratch. This greatly reduces the need for task-specific data, since only a smaller set of parameters has to be refined for the particular application [45].

**Skip-connections**

Increasing the number of layers is the most natural way of increasing the capacity of deep neural networks. However, over time it has been observed that a limit can be encountered where the performances of models start to decrease as their number of layers becomes too large. In [46], He et al. introduces the concept of *skip-connections* and *residual learning* to address this problem. As Figure 2.12 illustrates, skip connections provide a direct pathway between the input and the output of the corresponding layer. This pathway is equivalent to an identity layer, such that weights composing the layer under consideration have only to learn a residual mapping with respect to the identity function. Hence, if a shallower option is the optimal solution for a certain part of the network, it is easier for the learning process to converge to such a solution where weights would be set to approximately 0, while a pathway for gradient backpropagation still exists. By exploiting skip-connections, the authors introduced the ResNet model, one of the most popular network architectures used as a *backbone* in many state-the-of-art models for various image understanding tasks.

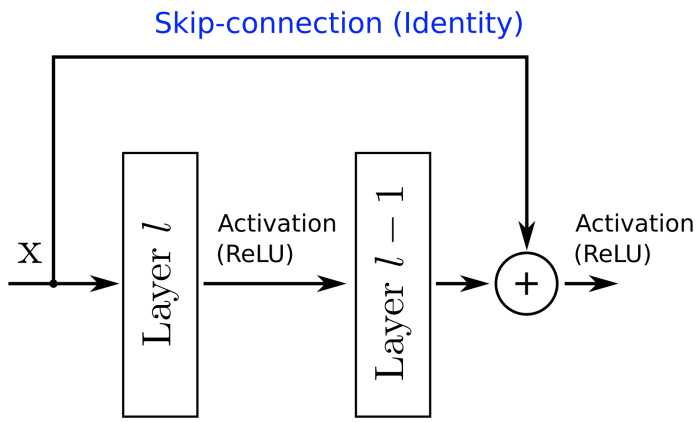



Figure 2.12: Representation of the concept of skip-connections for the design of residual networks.

## 2.4 Image semantic segmentation

In this section, we first provide an overview of the main datasets and evaluation metrics used for the design and assessment of image segmentation methods. Then, the most relevant approaches introduced for such tasks are reviewed, including techniques based on hand-engineered features as well as modern deep learning strategies.

### 2.4.1 Datasets

#### ImageNet

Introduced in [25], the ImageNet dataset contains over 14 million images labeled for $> 21k$ categories defined according to the hierarchical structure of WordNet [47]. In addition to the dataset, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) for object detection and image classification was held until 2017, and was crucial for the development of deep CNNs that revolutionized image classification tasks. The AlexNet model [4] was introduced at the ILSVRC 2012, winning the competition by a large margin and setting an important turning point where

CNN-based models started to be widely and successfully exploited for many tasks. Figure 2.13 shows some examples of images composing the ImageNet dataset.


Figure 2.13: Examples of images composing the ImageNet dataset. From left to right, these images contain annotations for the classes *person*, *bird* and *banjo*, respectively.

### PASCAL VOC

Introduced in its first version in 2005, the *PASCAL Visual Object Classes (VOC)* is still arguably the most widely used benchmark for semantic segmentation, with the leaderboard of its 2012 version[3] being constantly updated with the performances of novel state-of-the-art approaches [43]. In all its versions, the PASCAL VOC dataset consists of images collected from Flickr[4] and annotated according to the presence of the 20 different semantic categories listed below:

- *Vehicles:* airplane, bicycle, boat, bus, car, motorcycle, train;
- *Household:* bottle, chair, dining table, potted plant, sofa, tv/monitor
- *Animals:* bird, cat, cow, dog, horse, sheep
- *Other:* people

Figure 2.14 shows some examples of the annotations in the PASCAL VOC dataset.

[3]http://host.robots.ox.ac.uk:8080/leaderboard/
[4]http://www.flickr.com/

Figure 2.14: Examples of images and annotations composing the PASCAL dataset.

### MS COCO and COCO-Stuff

Inspired by the success of the ImageNet dataset for image classification, the *Microsoft Common Objects In COntext (COCO)* dataset [5] was introduced in 2015 to foster advances in object recognition, localization, and segmentation. For segmentation tasks, the version of the COCO dataset released in 2014 is split into 82$k$ *train*, 40$k$ *val* and 40$k$ *test* images, annotated for 80 different classes and with approximately 270$k$ segmented people and 886$k$ segmented object instances in the train+val images alone [5].

(a) COCO (b) COCO-Stuff

Figure 2.15: Examples of images and annotations composing the a) COCO [5] and b) COCO-Stuff [6] datasets.

The COCO dataset was recently augmented by Caesar et al. [6] into the COCO-Stuff dataset. While the original COCO annotations covered only foreground or "things" classes (e.g. animals, people, vehicles), this dataset includes pixel-level

annotations for background regions that are amorphous and/or considered as of lower relevance, such as *grass, sky, tree* and *flowers*. Compared to "things", identification of "stuff" classes is of particular importance to identify acquisition conditions, physical/material types and contextual relationships between multiple "things" and "stuff". This is particularly relevant to extend image segmentation methods to novel applications where, in comparison to traditional datasets, rather unconventional classes and scenarios are under investigation, such as the agricultural domains investigated in this dissertation. Figure 2.15 illustrates examples from the COCO and COCO-Stuff datasets.

**DAVIS**

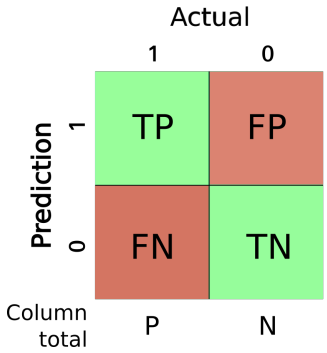
Motivated by the successes of datasets such as PASCAL and COCO on accelerating research in semantic segmentation, the Densely Annotated VIdeo Segmentation (DAVIS) dataset was introduced in [7] to foment advances on Video Object Segmentation. As illustrated in Figure 2.16, the DAVIS dataset comprises high quality video sequences and, compared to the PASCAL and COCO datasets, contains ground truth annotations with significantly higher quality at pixel-level for each frame.

Figure 2.16: Examples of images and annotations composing the DAVIS [7] dataset.

The first version DAVIS 2016 contained 50 video sequences with a total $3,455$ frames, which was extended in the DAVIS 2017 version [48] to a total to $10,474$ annotated frames composing 150 video sequences. Moreover, since 2017 the DAVIS Challenge on Video Object Segmentation has been held. In addition to the original semi-supervised tracking task where only the first frame of a video sequence is provided, the DAVIS 2018 [49] introduced an interactive track where annotations in form of scribbles are provided for certain frames, while the DAVIS 2019 [50] introduced an unsupervised task where object proposals must be generated without any input at test time.

### 2.4.2 Evaluation metrics

Originally introduced for communication tasks [51], metrics defined in terms of *true positives (TP), true negatives (TN), false positives (FP),* and *false negatives (FN)* have become very popular in machine learning problems. Illustrated in Figure 2.17 by means of a binary *Confusion Matrix*, these definitions are used to compute metrics such as *Recall*, *Precision*, *Accuracy*, and $F_1$ *score* (or *F-measure*), as Eqs. 2.12-2.15 indicate.




$$\text{recall} = \frac{TP}{P} \tag{2.12}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{2.13}$$

$$\text{accuracy} = \frac{TP + TN}{P + N} \tag{2.14}$$

$$\text{F-measure} = \frac{2}{{}^{1}/_{\text{precision}} + {}^{1}/_{\text{recall}}} \tag{2.15}$$

Figure 2.17: Confusion matrix and common performance metrics calculated from it.

In simpler terms, precision expresses how many of the returned results for a certain task are relevant, while recall indicates how many of the existing relevant instances of information were returned. To compare the overall performance of different methods in classification tasks, recall and precision metrics can be combined into the F-measure ($F_1$) or *Precision-Recall (PR)* curves (example in Figure 2.18), whose overall shape can be summarized in terms of the *Area Under the Curve (AUC)*. One of the most widely used metrics for object detection tasks, *Average Precision (AP)* values summarizes the shape of PR curves by approximating its area under curve as the mean precision at a set of equally spaced recall levels in the interval $[0, 1]$.

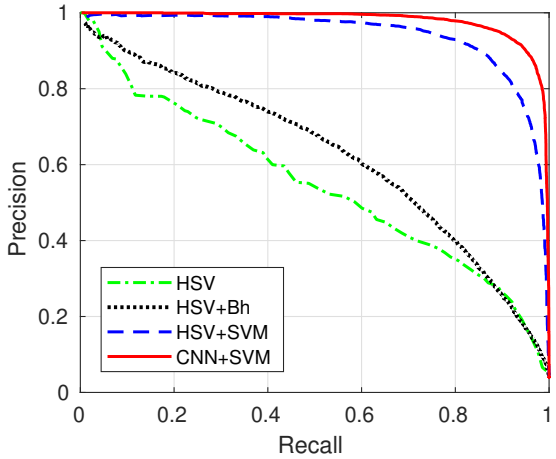



Figure 2.18: Example of usage of Precision-Recall (PR) curves for comparison of different methods on a segmentation tasks. Extracted from [9], one of the works that are part of this dissertation.

Analysis in terms of precision-recall curves (PR) and the corresponding $F_1$ score are of particular relevance in scenarios of imbalanced datasets. In such scenarios, evaluations of performance using only accuracy measurements may be misleading, since they are insensitive to changes in the rate of class distribution. In contrast, by

definition, the computation of precision gives to false positive detections the same relative weight as true positives, such that PR metrics are more robust to dataset imbalance.

Many evaluation metrics have been proposed for assessing performance of image segmentation techniques. The simplest are global accuracy metrics, which compute the overall percentage of pixels labeled correctly in a dataset. To better handle potential class imbalance, in multi-class scenarios the average of accuracies for each class is considered a better metric. Defined according to Eq. 2.16, the *mean Intersection over Union (mIoU)* or Jaccard index is arguably the most widely used metric for semantic segmentation tasks, as exemplified by the leaderboards in the PASCAL VOC 2012 [43] and the DAVIS [7] datasets. It consists of dividing, for each class in a set $\mathcal{C} = \{1, ..., n_c\}$, the number of pixels of the intersection between a predicted mask $M$ and the corresponding class ground truth $G$ by their union, and finally averaging the results over all classes.

$$mIoU = \frac{1}{n_c} \sum_{\mathcal{C}} \frac{|M \cap G|}{|M \cup G|} = \frac{1}{n_c} \sum_{\mathcal{C}} \frac{TP}{TP + FP + FN}. \tag{2.16}$$

**Contour-oriented metrics**

For scenarios where boundary adherence is of particular interest, evaluation strategies that focus on contour accuracy have been introduced. In addition to the conventional Jaccard $\mathcal{J}$ index, the DAVIS dataset [7] proposes a *Contour Accuracy* metric $\mathcal{F}$ defined according to

$$\mathcal{F} = \frac{2P_c R_c}{P_c + R_c}, \tag{2.17}$$

where $P_c$ and $R_c$ correspond to precision and recall metrics as in the conventional $F_1$ score computation, respectively, with the difference that both values are computed

only at pixels composing the contour $c(M)$ of a given prediction mask $M$ with respect to the contour $c(G)$ of a ground-truth mask $G$.

Moreover, assessments of boundary adherence quality can be also performed by computing segmentation accuracy only on narrow regions closer to the ground-truth boundaries. Ghiasi et al. describes in [52] such an approach for evaluations using $mIoU$ for the PASCAL dataset, where a performance curve is constructed by computing segmentation $mIoU$ for varied thicknesses around the boundaries of the ground-truth masks.

### 2.4.3 Approaches based on hand-engineered feature descriptors

Initial methods proposed for image segmentation were based on bottom-up approaches, where pixels were labeled based on low-level features such as color, texture or other morphological characteristics [28]. While different levels of quantization have been exploited (pixels, segments, superpixels), these methods achieve only limited success on tasks requiring the identification of complex structures/objects, since higher-level context information is required to characterize elements such as people and animals, among many others.

Introduced in [53] and still popular in image matching tasks, the Scale Invariant Feature Transform (SIFT) generates a large dictionary of local feature vectors called SIFT keys for each image. Evaluating images in a scale-pyramid manner, key locations are identified as regions where extreme values of a difference-of-Gaussian function are present. For each keypoint, image gradients and orientations are extracted such that features invariant to scaling, translation and rotation are obtained. Sets of SIFT keypoints can then be used as descriptors of structures of interest, such that object detection can be performed by comparing the similarity between the keypoint in these sets.

As reported in [43], methods based on the concept of aggregating features into bag-of-visual-words (BOW) [54] achieved at the time the best performances on the PASCAL VOC 2007 classification tasks. Each BOW corresponds to a histogram of the frequency of pre-defined patterns in patches of an image, using, say, SIFT as descriptors of selected affine regions. In [55], Dalal and Triggs introduced the concept of locally normalized Histogram of Oriented Gradients (HOG) computed instead on a dense grid of uniformly spaced cells as an object descriptor for human detection. As in described in [54], detection is performed by combining descriptors with a Support Vector Machine (SVM) classifier, with results significantly outperforming detectors based on previous descriptors such as the Haar wavelet [56].

In [57], mixtures of multiscale deformable part models are combined with SVMs formulated using latent variables to overcome difficulties of training part-based models while using labels in the form of bounding boxes. Applied to images using sliding windows, this method combines a coarse scale HOG representation with a set of finer-scale HOG part filters. Outperforming previous methods, it became the standard approach exploited by state-of-the-art models for object detection in the PASCAL dataset between 2008 and 2012 [43]. To reduce computational costs associated with exhaustive search using sliding windows over the whole image, methods such as *selective search* [58] attempt to optimize the selection of regions to be evaluated through a data-driven approach. Such strategies represent a return to bottom-up approaches, which merge initial regions using complimentary grouping criteria, multiple color spaces, and different invariance properties. In this way, a diversified set of region proposals is obtained for further classification.

While for segmentation tasks such bottom-up approaches were originally most popular, to exploit object modeling at higher (global) levels many works pursued top-down strategies where object detection was first performed, followed by the segmentation of structures inside the returned bounding boxes. Following earlier attempts

that focused on learning average masks for each object category [3], works such as [59] proposed coupling global and local representations to refine average class masks using an optimization approach that includes as final constraints the concept of superpixels, explained in the following Section 2.4.4. As reviewed in [43], until 2012 combinations and extensions of techniques based on multiple bottom-up segmentations, hierarchical random field models and part-based feature descriptors were the core components of state-of-the-art entries on the PASCAL VOC leaderboard.

### 2.4.4 Weak and unsupervised segmentation

#### Clustering techniques

*Superpixels* are perceptually meaningful clusters of pixels generated by grouping together pixels that share similar appearance and spatial distribution. The *simple linear iterative clustering* (SLIC) superpixel algorithm is one of the most widely-used algorithms for superpixel segmentation. It adapts $k$-means clustering to group pixels according to a weighted distance measure that considers both color and spatial proximity [60]. Figure 2.19 illustrates the segmentation of an image using the SLIC algorithm.

Figure 2.19: Example of superpixel segmentation of an image using the SLIC algorithm introduced in [60].

In [61], Stutz et al. provide a review of existing superpixel approaches, with a comprehensive evaluation that ranked 28 state-of-the-art algorithms according to several metrics such as average recall, average undersegmentation error, boundary recall and also realtime capability. In addition to clustering-based algorithms, alternative formulations for segmentation into superpixels include, for example, energy based optimization. The Extended Topology Preserving Segmentation (ETPS) method [62] is one example, standing out in their evaluation with high performance in terms of boundary adherence, recall, stability and runtime.

Some limitations of conventional superpixel algorithms include lack of adaptiveness in terms of adjusting to local features and generating clusters of flexible sizes, as well as poor robustness to mistakes in the initialization of parameters.

Models exploiting Bayesian estimation have been introduced to overcome these limitations of superpixel algorithms, with strategies that range from pixel-related Gaussian Mixture Models (GMMs) [63, 64] to non-parametric mixture models [65]. In such approaches, previously fixed normalization hyperparameters are replaced by Bayesian priors, which are updated in conjunction with other cluster statistics in the form of covariances as assignments of pixels to clusters take place.

### Graph cuts

Energy minimization approaches using the graph cuts paradigm are suited to interactive segmentation where hard constraints are specified via squiggles for background and foreground classes [66–68]. The popular GrabCut algorithm [69] improved over interactive tools such as Intelligent Scissors (Magnetic Lasso) [70], relaxing some of the labeling burden on the user. The user selects a bounding box of background pixels and can further edit the generated segmentation by drawing firm background/foreground traces. Gaussian Mixture Models (GMMs) are used for color modeling and a Gibbs energy is iteratively minimized using minimum cut.

### Level sets

The level set approach has been used in segmentation since the 1990s, and can also be formulated as an energy minimization problem. Given an initialization, a boundary is evolved in the direction of a local minimum found via front propagation by solving partial differential equations [71]. An issue with level set implementations in the 2000s was runtime, and interactive approaches focused on reducing runtime using GPU implementation [72, 73]. One approach [72] allowed user input to adjust model parameters, while in [73] the authors reformulated energy functionals to incorporate user input. In [74], bounding-box initialization and the level set formalism were used for interactive segmentation. The TouchCut [75] interface exploits level-sets to grow segmentation masks from single points, which is effective when foreground and background colors are significantly different.

### Propagation by pixel-affinity

In a similar fashion that superpixel algorithms segment input images into clusters [61], several matting and segmentation algorithms use low-level information such as texture, color affinity and spatial proximity to classify unlabeled regions based on sparse annotations [76–78].

### Joint propagation and CNN training

Recent approaches aiming at interactive or weakly-supervised semantic segmentation focus on architectures in which the propagation of sparse annotations and the optimization of network parameters are performed jointly. Different works combine Fully Convolutional Networks (FCNs, described in Section 2.4.5 below) with: GrabCut [79]; superpixels and graphical modeling [80, 81]; novel loss functions and training strategies for weakly-supervised and interactive learning [80, 82, 83]. In [84],

the idea of Laplacian matting matrices is combined with superpixels and a DeepLab-ResNet ( described in Section 2.4.5 below) [22] to identify layers (soft segments) that are semantically meaningful. For annotation of video sequences, in [85] an FCN is used to map input pixels onto an embedded space where pixels belonging to the same instance are close together, followed by a nearest-neighbor approach that classifies pixels based on reference masks provided at the first frame and on sparse user inputs.

### 2.4.5 Deep learning-based approaches

Following the success of CNNs on image classification tasks, the work of Girschick et al. [86] introduced the concept of region-based CNNs (R-CNN), outperforming by a large margin previous hand-engineered methods for object detection. In that work, a CNN pre-trained on a large auxiliary dataset (ImageNet) and then fine-tuned using a smaller but more specific dataset (PASCAL dataset for object detection) is used to classify region proposals generated using the selective search algorithm from [58]. The Faster R-CNN proposed in [87] improved this model by replacing selective search with the concept of Region Proposal Networks (RPNs), which share convolutional layers with the classification network. Both modules compose a single, unified network for object detection.

Yet, tasks requiring image labeling at pixel-level are particularly challenging for CNN-based systems. The combination of max-pooling and striding operations constitute a standard strategy for learning hierarchical features, which are a determining factor in the success of deep learning models. They explore the transition invariance favored by image-level labeling tasks, i.e., the fact that only the presence of an object matters, not its location. Such premise, however, does not hold for pixel-dense classification tasks, which require precise object localization. As pointed out by Chen et al. in [88], such spatial insensitivity and image downsampling inherent to

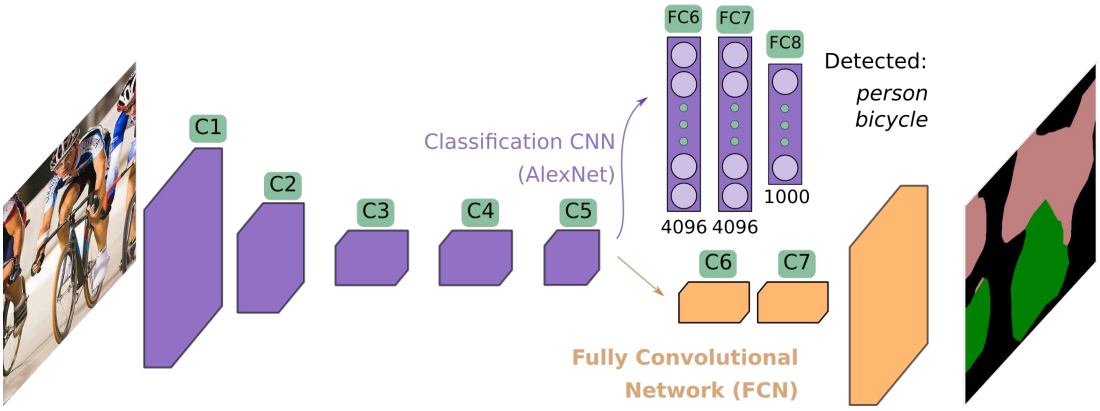



Figure 2.20: Representation of a fully convolutional network (FCN). In comparison to an original classification network such as AlexNet [4], fully convolutional networks replace fully connected classification layers by further convolutional layers and an upsampling step to return segmentation masks.

conventional CNNs are two major hindering factors for their performance in image segmentation.

This is clearly exemplified by segmentation predictions generated by models such as the ones introduced by Eigen & Fergus [89] and the earlier Fully Convolutional Networks (FCNs) [90]. As represented in Figure 2.20, FCN architectures essentially consist of image classification CNN models with their fully connected layers replaced by further convolutions, followed by a bilinear upsample of the feature maps to generate a pixel-dense prediction. These models generate coarse segmentation masks with limited boundary adherence, an open problem that has driven many advances in the field.

Different strategies have been proposed to alleviate the effects of downsampling [91]. Many approaches focus on developing better upsampling strategies to improve segmentation accuracy. In this context, the concept of *fractional convolutions* is exploited to define learnable *deconvolution layers*. Represented in Figure 2.21, fractional convolutions differ from conventional ones as they use fractional strides $< 1$

in order to expand the receptive field. Since in CNNs they are used to achieve upsample while traditional convolutions provide downsampling, fractional convolutions are also often called transposed convolutions.

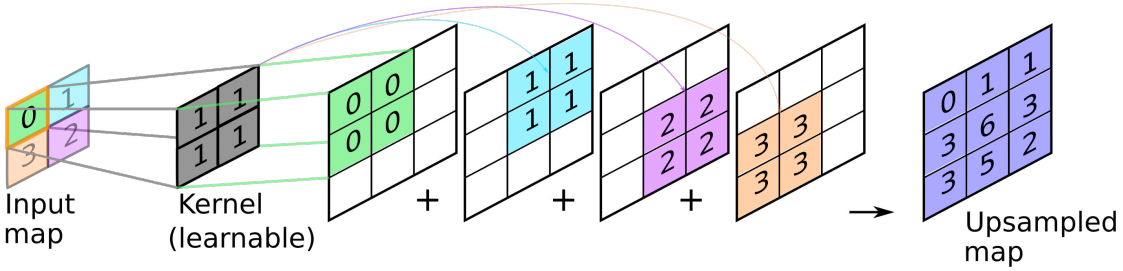



Figure 2.21: Representation of fractional convolutions, commonly used for upsampling (inspired on drawing available at [5]).

Architectures designed with encoder-decoder schemes are at the core of many recent models proposed for image segmentation. With a basic representation of their architectures provided in Figure 2.22, the U-Net [92] and the SegNet [93] models are examples of improved strategies that focused on encoder-decoder architectures where the decoder path includes skip-connections to convey information from encoder layers to better guide upsampling. While SegNet guides upsampling according to the indices associated with positions that yielded the pooling outputs, U-Net copies and concatenates resized versions of feature maps from earlier encoding layers to provide the decoding layers with low-level details available before downsampling.

Another direction concentrates on reducing the amount of details lost through downsampling. The DeepLab model introduced in [22] is one of such models, and currently one of the most successful approaches for semantic image segmentation using deep learning. It combines the ResNet-101 [46] model with the concept of *atrous* or

[5] `http://d2l.ai/chapter_computer-vision/transposed-conv.html`

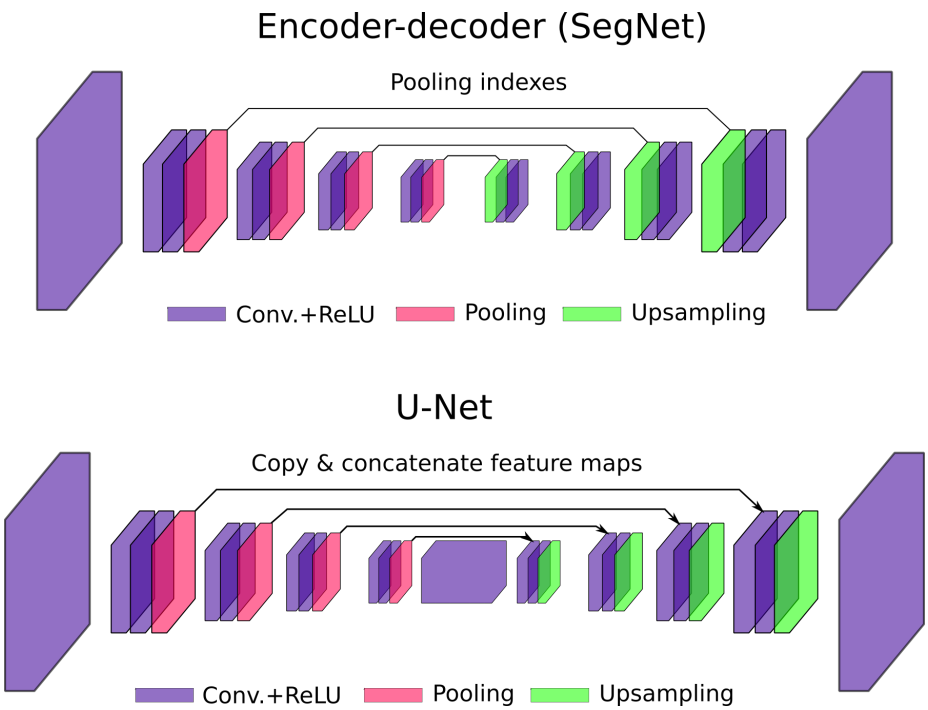



Figure 2.22: Basic representations of encoder-decoder (SegNet [93]) and U-Net architectures [92].

dilated convolutions. Illustrated in Figure 2.23, this concept was first exploited in [94] for the design of convolutional filters in CNNs.

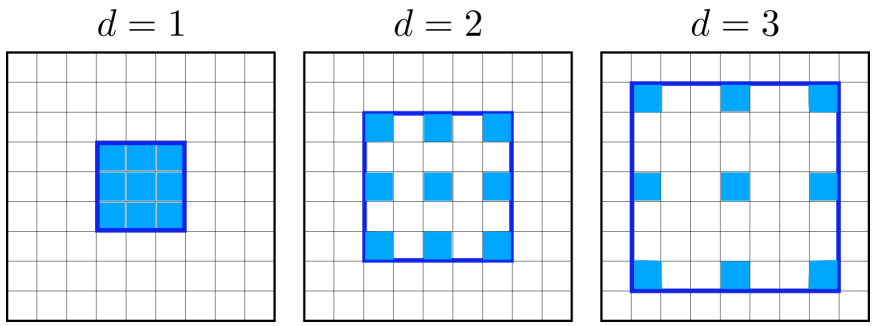



Figure 2.23: Representation of the concept of dilated convolutions, for different dilation rates $d$. Illustration inspired on figure present in [22].

Figure. 2.24 provides a simplified illustration of the DeepLab architecture, as well as a representation of the second version DeepLabV2. Similar to works such as PSPNet [95], DeepLabV2 also exploits the concept of spatial pyramid pooling (SPP) for detection of objects at multiple scales. Introduced in [96], SPP adapts for the design of CNNs the concept of scale pyramid introduced in [97] for image evaluation. More specifically, Chen et al. introduces in [22] the concept of *atrous spatial pyramid pooling (ASPP)*, where representations at multiple scales are obtained using dilated (or atrous) convolutions with varied dilation rates.

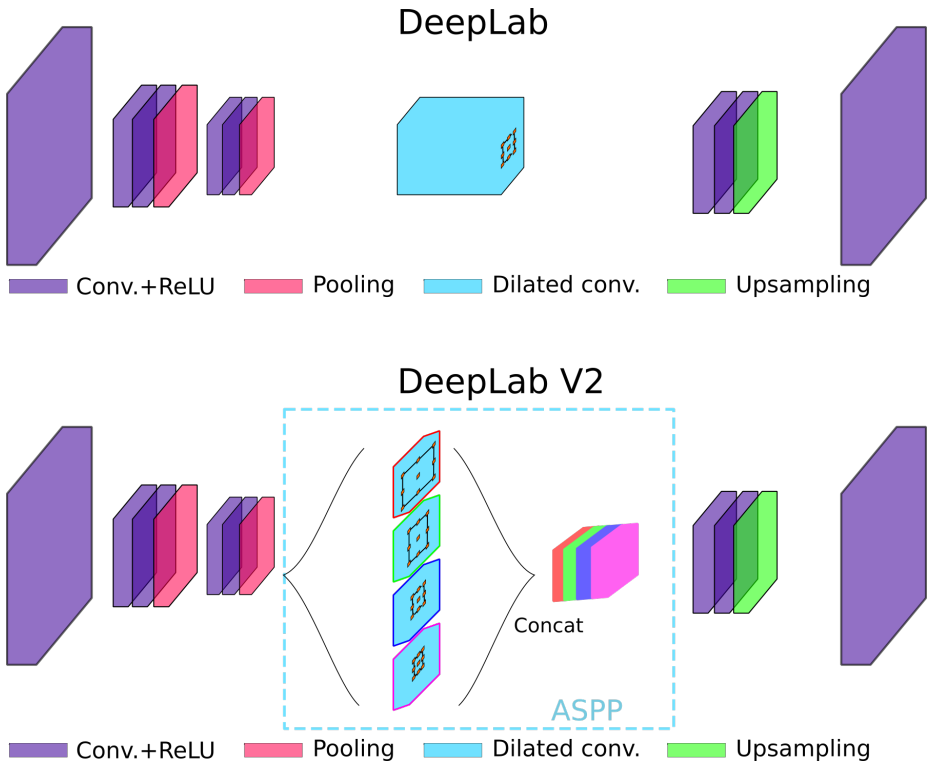



Figure 2.24: Representations of DeepLab [88] and DeepLabV2 [22] architectures, including the concept of atrous spatial pyramid pooling (ASPP), where multiple dilated convolutions are performed with different dilation rates.

By combining these strategies, the DeepLab models significantly reduce the downsampling rate and achieve state-of-the-art performance in challenging semantic segmentation datasets such as the PASCAL VOC [43] and COCO [5]. More recently, the current state-of-the-art DeepLabV3+ model [98] was introduced, combining ASPP strategies (adjusted to exploit image-level features) with a decoder module to refine segmentation along boundaries.

The RefineNet model [99] adopts similar strategies. As most of the best performing recent approaches, it leverages the ability of residual networks [46] to learn deeper and more complex representations of images. RefineNet [99] employs a multi-path refinement structure such that long-range residual connections are formed. Each block has as input two of the feature maps collected from the residual network at resolutions of 1/4, 1/8, 1/16 and 1/32 of that of the original image. The inputs are combined through a sequence of adaptive convolutions for task-specific fine tuning, upsampling for multi-resolution fusion and residual pooling to capture background context.

A variation of the semantic segmentation problem is instance segmentation, which requires detecting and segmenting individual object instances. Coarse segmentations significantly hamper this type of task, since neighboring instances of objects of the same class are frequently merged into a single segmentation. Dai et al. in [100] introduced the concept of instance-sensitive FCNs, in which an FCN is designed to compute score maps that determine the likelihood that a pixel belongs to a relative position (e.g. right-upper corner) of an object instance. FCIS [24] is an extension of that approach, which achieved the state-of-the-art performance and won the COCO 2016 segmentation competition.

### 2.4.6 Post-processing techniques

In addition to adjustments in CNN architectures, some studies focus on investigating techniques that employ low-level image features to aid CNN-based models in image segmentation tasks. Examples include exploiting superpixels [61] as a preprocessing step, where pixels are grouped based on low-level properties (e.g. color similarity) and each group is evaluated using hand-engineered hierarchical features [101] or CNNs [102, 103]. Another possible approach consists of adapting the GrabCut model [69], which is a well-known method for interactive segmentation based on the minimization of an energy function that models the background and foreground as Gaussian mixture models (GMM).

Likewise, techniques such as superpixels and conditional random fields (CRFs) have also been employed for the post-processing of segmentations generated by deep CNN models. In [104], Krähenbühl and Koltun introduced the DenseCRF, which is an efficient algorithm for fully connected CRFs containing pairwise potentials that associate all pairs of pixels in an image. In contrast to conventional fully connected CRFs implementations, DenseCRF improves computational efficiency thanks to an approximate inference algorithm in which pairwise potentials are modelled as combinations of Gaussian kernels. More specifically, an appearance kernel captures the aspect that pixels similar in terms of color and spatial location are likely to share the same labels, while a smoothness kernel is designed to remove spurious regions. Crucially, such kernels are designed with hyperparameters that require supervised optimization.

As previously mentioned, the DeepLab paper [88] proposes to integrate its novel architecture with the DenseCRF model from [104] to refine segmentation masks especially along boundaries. Chen et al. introduced the DT-EdgeNet model in [105], which is a computationally cheaper alternative that replaces the DenseCRF with a

domain-transform approach that refines segmentation using a modern edge-preserving filtering method. Similarly to CRF, however, DT-EdgeNet also comprises parameters that have to be optimized in a supervised manner when applied to different datasets.

## 2.5 Basic concepts of probability theory and stochastic methods

This section covers concepts from probability theory that are directly applicable to the methods proposed in this dissertation. We assume the reader is familiar with basic concepts such as *expected values*, *variances*, and *covariances*, as well as the basic normal (or Gaussian) distribution. For detailed explanation of these concepts, we refer to [106].

### Bayes' rule

As mentioned in [34], machine learning systems can benefit from concepts of probability theory to properly reason and make decisions in the presence of uncertainties. Expressed in Eq. 2.18, Bayes' rule is of particular relevance, as it provides a systematic approach to update *prior* knowledge about the probability distribution of a state or, in machine learning configurations, a set of model parameters $\theta$, according to new *evidences* or observations $y$. The Bayes' rule allows this computation as a function of the *likelihood* $P(\theta|y)$ that an evidence $y$ is observed, given a model according to $\theta$

$$\underbrace{P(\theta|y)}_{\text{Posterior}} = \frac{\overbrace{P(y|\theta)}^{\text{Likelihood}}\ \overbrace{P(\theta)}^{\text{Prior}}}{\underbrace{P(y)}_{\text{Evidence}}}. \tag{2.18}$$

The component $P(y)$ corresponds to the probability of the observation for all possible values of $\theta$, i.e., $P(y) = \int P(\theta)P(y|\theta)d\theta$ in the case of continuous $\theta$. Therefore, it is a constant value that is often omitted, such that the unnormalized posterior density

given in Eq. 2.19 below is commonly used [107]

$$P(\theta|y) \propto P(y|\theta)P(\theta). \tag{2.19}$$

In the context of machine learning, Bayes' rule is commonly used by framing data measurements or predictions as observations. In other words, it allows estimating the probability that a hypothesis or configuration given by $\theta$ is correct, based on data inputs (observations) and the likelihood of such observations given that set of parameters $\theta$.

**Estimators**

A popular strategy for designing predictive models is to frame the model's output as the likelihood, such that model optimization focuses on finding the set of parameters that yield the *maximum likelihood* [35]. For models where no reliable prior knowledge is available, the maximum likelihood estimator (MLE) has also the advantage of relying solely on measurements to find a set of parameters that is most likely to explain the data distributions being used for training.

Least square estimators are another family of estimators widely used in machine learning frameworks. In particular, the mean squared error (MSE) or squared Euclidean distance is a popular choice, as it has important characteristics such as linearity, positive-definiteness and is differentiable for all input values, which makes it suitable for optimization approaches based on gradient computation.

### 2.5.1 Monte Carlo estimation and variance reduction techniques

As described in [35], in many scenarios it is unfeasible to obtain exact estimations of desired properties such as expected values. Possible root causes include high dimensionality of certain distributions, or the fact that they have highly complex forms for which expectations do not have an analytical solution.

In such cases, approximate inference strategies are exploited in many machine learning systems to provide estimations within a certain range of uncertainty. Deterministic approximation schemes include the concept of *variational inference* [34, 35], where an approximate distribution is used for estimation of a distribution of interest, with their differences minimized through strategies commonly based on Kullback-Leibler (KL) divergence.

In this dissertation, we focus instead on stochastic approximation schemes based on numerical sampling, which provide approximate estimations with a reduced cost. More specifically, we exploit the concept of *Monte Carlo sampling* for approximate inference in the algorithms described in this work. Let $f(\mathbf{x})$ represent some function of interest for which the expected value $\mathbb{E}[f(\mathbf{x})]$ cannot be efficiently computed analytically. A numerical computation according to Eq. 2.20 requires evaluating all possible values of $\mathbf{x}$, which is practically unfeasible in most cases. Hence, a possible approximation scheme consists in sampling values of $\mathbf{x}$ instead of evaluating it exhaustively.

$$\mathbb{E}[f(\mathbf{x})] = \int f(\mathbf{x})P(\mathbf{x})d\mathbf{x}. \tag{2.20}$$

Following Eq. 2.21, approximation through Monte Carlo sampling consists in randomly drawing $N$ samples $\mathbf{x_1}, ..., \mathbf{x_N}$ according to the distribution $P(\mathbf{x})$ and computing the associated empirical average $\hat{\mu}$ as approximation for the integral defining $\mathbb{E}[f(\mathbf{x})]$.

$$\mathbb{E}[f(\mathbf{x})] \approx \hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x_i}) \tag{2.21}$$

This approximation is justified through the law of large numbers. As explained in [108], for samples independent and identically distributed (i.i.d.) with $P(\mathbf{x})$ the weak law of large numbers states that $\lim_{N\to\infty} P(|\hat{\mu} - \mu| \leq \epsilon) = 1$, for any $\epsilon > 0$. Assuming the mean $\mu$ exists, the strong law of large numbers state that

$P(\lim_{N\to\infty} |\hat{\mu} - \mu| = 0) = 1$, which indicate that *"Monte Carlo will eventually produce an error as small as we like"* [108].

Moreover, the estimator $\hat{\mu}$ obtained using a Monte Carlo framework has mean and variance described according to Eqs. 2.22 and 2.23, respectively, with two consequences: i) from Eq. 2.22, it follows that the estimator is unbiased; and ii) from Eq. 2.23, it follows that if the variance of each sample is lower than $\infty$, the estimator's variance decreases according to the number of samples and converges to zero [34,108].

$$\mathbb{E}[\hat{\mu}] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[f(\mathbf{x_i})] = \frac{1}{N}\sum_{i=1}^{N}\mu = \mu, \tag{2.22}$$

$$\sigma^2[\hat{\mu}] = \frac{1}{N^2}\sum_{i=1}^{N}\sigma^2[f(\mathbf{x})] = \frac{\sigma^2[f(\mathbf{x})]}{N}. \tag{2.23}$$

Hence, from the central limit theorem it follows that an approximation using $N$ samples converges to a normal distribution $\mathcal{N} \sim (\mu, {}^{\sigma^2[f(\mathbf{x})]}/_{N})$, which allows estimating confidence intervals around the approximate estimation obtained for a given number of samples [34,108].

#### 2.5.1.1 Variance reduction techniques

In addition to the simplest strategy of randomly drawing samples according to a uniform distribution, more sofisticated sampling strategies exist. Since the variance of the Monte Carlo estimator is given according to ${}^{\sigma^2}/_{N}$, two directions can be exploited to improve the quality of estimation: i) increase the number of samples, which however increases the computational cost of the overall process; and ii) reduce the variance associated to each estimation. The second direction is the focus of a family of strategies called *variance reduction techniques*. In the paragraphs below we briefly describe two popular variance reduction strategies that are exploited in methods covered in this disseration, with [108] as a reference for further information.

### Stratification

Stratified sampling consists in guiding the sampling process by dividing the domain of interest $\mathcal{D}$ into separate regions, and sampling from each region according to a pre-defined criteria. The most straightforward approach consists in sampling equal numbers of samples from each region, which can be particularly helpful in scenarios of data imbalance. Another strategy is *systematic sampling*, which consists in first randomly selecting a sample, and then selecting every $n-th$ element positioned with respect to the first sample. The step size $n$ is defined according to the pre-defined number of samples and the population size, which can be done over the whole domain or within pre-defined strata composing a stratified approach.

### Antithetics

Antithetical sampling consists in selecting pairs of correlated samples that are symmetric, opposite to each other according to some criteria. For example, if the domain of sampling is within a range $[0, 1]$, antithetical consists in first randomly drawing a sample $x$, and then selecting an extra sample at the position $1 - x$. For zero-centered, symmetric distributions such as a normal distribution with zero-mean, antithetic sampling corresponds to selecting a second sample that has the opposite sign of the randomly selected one.

## 2.6 Uncertainty estimation techniques for computer vision

In statistical fields, uncertainty is commonly categorized into *aleatoric* and *epistemic* uncertainties. The first refers to uncertainties whose root cause is observation noise, which might be intrinsic and remain constant for a given task (*homoscedastic* uncertainty) or vary for different inputs (*heteroscedastic*). In contrast, *epistemic*

uncertainty refers to uncertainties in the model that attempts to explain the data, which in machine learning corresponds to uncertainty on the parameterization [109].

Well-calibrated uncertainty estimations are crucial to indicate potential model failures and provide insights towards model interpretability. However, modern neural networks fail to capture the uncertainty of their own predictions. As reviewed in [110], prediction scores provided by modern deep models are not calibrated with respect to the expected error, in contrast to earlier architectures such as the well-known LeNet-5 [111]. In other words, modern neural networks output confidence scores that do not represent true probabilities, although many recent works refer to output values normalized using a *softmax* function as prediction probabilities.

Therefore, recent works have been exploring multiple strategies to provide deep learning models with the ability to predict their own uncertainties, with Bayesian deep models in the spotlight. Kendall & Gal provide in [109] a discussion on types of uncertainties that need to be modeled for computer vision, based on the distinction between aleatoric and epistemic uncertainties.

While epistemic uncertainties can be explained away with more training data, aleatoric uncertainties cannot, and as such are rather most effectively modeled to guide decision making such as relying on an alternative method or consulting an external oracle. In this context, concepts of Bayesian deep learning have become increasingly popular to provide modern neural networks with the ability of uncertainty estimation. In [109], outputs are modeled as corrupted with Gaussian random noise, such that heteroscedastic aleatoric uncertainty can be estimated by designing models with an extra output parameter trained using a customized loss function that learns to regress the variances of this noise.

Eq. 2.24 is an example of such a loss function, where the mean squared error is combined with components associated to the models' own predictive uncertainty provided as the extra output $\hat{\sigma}$. Here, $\theta$ corresponds to the set of models' parameters,

$X$ is the set of data points, while $y$ and $\tilde{y}$ indicate ground-truth and predicted labels, respectively. Intuitively, the division by $\hat{\sigma_i}^{-2}$ in the first component on the right-hand-side of the equation guides the optimization such that the model learns to output higher uncertainties when the MSE component is large, while the $log\hat{\sigma}_i$component acts as a regularization factor that prevents the model from outputting high uncertainty values for all predictions.

$$\mathcal{L}(\theta) = \frac{1}{|X|} \sum_i \frac{1}{2\hat{\sigma_i}^{-2}} ||y_i - \hat{y}_i||^2 + \frac{log\ \hat{\sigma}_i}{2}. \tag{2.24}$$

To identify epistemic uncertainties, approaches using ensembles [112] and variants of dropout at test time [113] have been exploited to estimate the uncertainties associated with model parameters. We refer to the corresponding articles for more details.

# CHAPTER 3
# SEMANTIC SEGMENTATION REFINEMENT

For many applications of computer vision, image segmentation with high accuracy at pixel-level is a key requirement. The agricultural field, where image segmentation has been exploited as part of perception modules targeting pollination, orchard management, and harvesting in horticultural scenarios is an example [2, 114, 115].

Deep learning models based on convolutional neural networks (CNN) have substantially improved the state of the art in image understanding. However, conventional CNN-based segmentation models are limited by the typical downsampling employed to learn hierarchical features. Pixel-level details are lost in this process, resulting in segmentation masks that poorly adhere to object boundaries.

To mitigate these limitations, modern image segmentation models employ strategies such as *atrous* convolutions [88], encoder-decoder architectures with skip-connections [90,92,99], pyramid scaling [22], among others. Large improvements have been achieved through these strategies in comparison to conventional CNN architectures, but the segmentation they produce still tends not to be finely aligned with the boundaries of objects. Post-processing approaches such as conditional random fields (CRFs) [22, 104] have been successful in segmentation refinement, but their performance depends on proper optimization of parameters for each specific dataset and predictor module being used.

In this chapter, we describe two versions of the *Region Growing Refinement (RGR)* algorithm, an unsupervised and easily generalizable post-processing module that operates on the principle of appearance-based region growing to refine the predictions generated by a CNN for semantic segmentation. Both versions of RGR share the same fundamental steps: in a Monte Carlo framework, initial pixels are sampled as high-quality seeds from regions labeled with high-confidence scores and grown into

clusters for segmentation refinement. In the second version, named *probabilistic Region Growing Refinement (pRGR)* and released in [116], the original RGR algorithm published in [1] is extended with a solid mathematical foundation where all steps are guided by a probabilistic framework, as well as design improvements that increase its segmentation performance in various benchmarks.

We provide next specific descriptions and discussions associated with RGR and pRGR in Sections 3.1 and 3.2, respectively. At the beginning of each section, we provide a list of the main contributions associated to each corresponding work.

## 3.1 Region Growing Refinement (RGR)

In this section, we describe the *Region Growing Refinement (RGR)* algorithm [1], which provides the following contributions to the state of the art on image semantic segmentation:

- *RGR provides high-quality segmentation refinement without requiring any dataset-specific parameter optimization, working in a fully unsupervised manner.* Based on the classification scores available from a detector (e.g. a segmentation CNN), our method first divides the image into three regions: high confidence background, high confidence object, and uncertainty region. The pixels within the uncertainty region, which are the ones that tend to be misclassified by CNN-based methods, are then labeled by means of region growing. We apply Monte Carlo sampling to select initial seeds from the high confidence regions, which are then grown according to a distance metric computed in the 5-D space of spatial and color coordinates.
- *we demonstrate the applicability of RGR with experiments using different CNNs, datasets and baselines.* We first employ the Fully Convolutional Instance-aware Semantic Segmentation (FCIS) algorithm [24] as a predictor module as well

as the baseline for our performance evaluation. Since the ground truth annotations composing the MS COCO dataset [5] contain non-negligible inaccuracies, we also assess RGR's efficacy on the PASCAL VOC 2012 [43] validation set and on selected video sequences from the DAVIS dataset [7]. As a result, in addition to relatively small increases in segmentation accuracy for the MS COCO (+1.5% in $AP$) and the PASCAL datasets (+0.5% in $mIoU\%$), we report significantly better quantitative results for the DAVIS sequences (+3.2% in $\mathcal{J}(IoU)\%$), which more realistically reflect the segmentation improvement observed through qualitative (visual) inspection.

- we also compare the RGR algorithm against the state-of-the-art but supervised methods DenseCRF [104] and DT-EdgeNet [105], for refinement of DeepLab [22] predictions. *RGR provides both running time and segmentation refinement performance comparable to the optimized versions of both supervised methods, but without requiring neither dataset- nor model-specific fine-tuning.*

### 3.1.1 Proposed approach

The method we propose for refinement of segmentation boundaries is a generic **unsupervised** post-processing module that can be coupled to the output of any CNN or similar model for semantic segmentation. Our RGR algorithm consists of four main steps: 1) identification of low and high confidence classification regions; 2) Monte Carlo sampling of initial seeds; 3) region growing; and 4) majority voting and final classification. The operations that comprise these steps are described in detail below. In our description, we make reference to Figure 3.1 and Algorithm 1, which list the operations performed by our method for each proposed detection in an image. If detections for multiple classes are present, the algorithm is executed on the score maps associated with each class, and the final classification is defined by computing the maximum likelihood across classes.

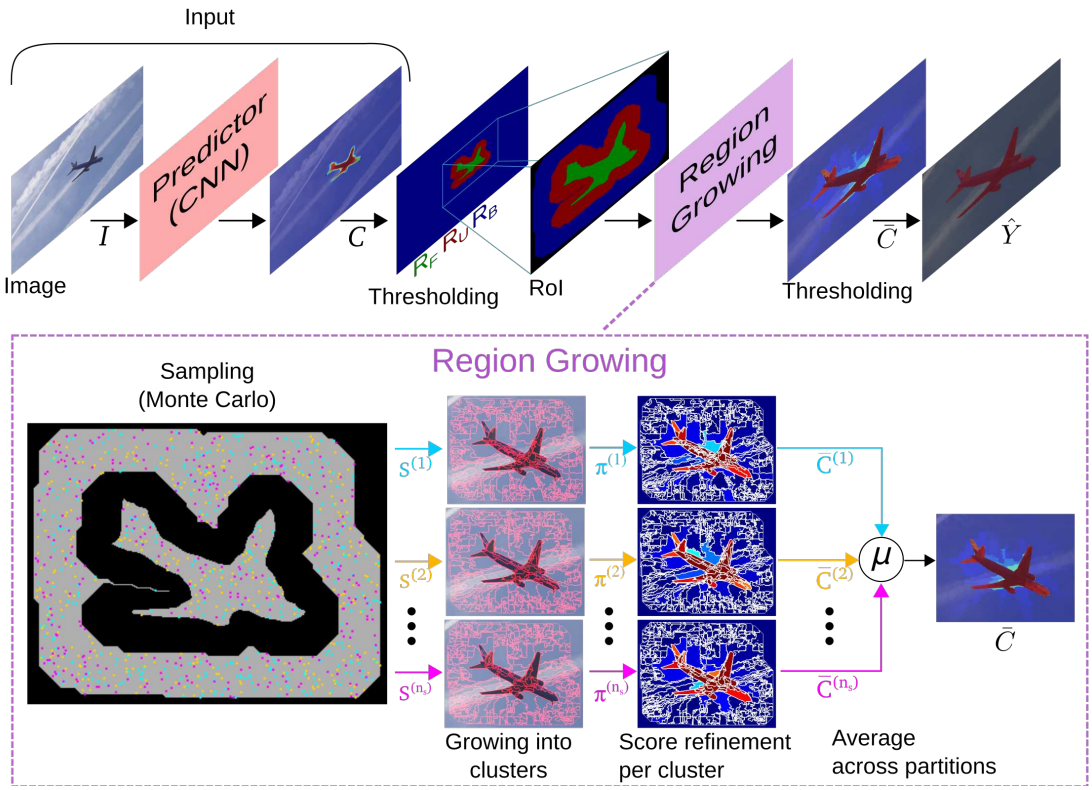



Figure 3.1: Diagram illustrating the sequence of tasks performed by the proposed RGR model for segmentation refinement. Each task and its corresponding output (shown below the arrows) are described in Algorithm 1.

*1) Step 1 - Thresholding the image into three regions:* Conventional models typically infer a final classification by pixel-wise thresholding the scores obtained for each class. Instead of relying on intermediate threshold values, our refinement module directly exploits the available classification scores to differentiate between three regions: a) high confidence foreground (or object) $R_F$; b) high confidence background $R_B$; and c) uncertainty region $R_U$. As defined in Eq. 3.1, these regions are identified

using two high confidence thresholds $t_f$ and $t_b$

$$\begin{aligned} R_F &= \{p_j | C(p_j) \geq t_f\}, \\ R_U &= \{p_j | t_b < C(p_j) < t_f\}, \\ R_B &= \{p_j | C(p_j) \leq t_b\}, \end{aligned} \tag{3.1}$$

where $p_j$ is the $j$-th pixel in the input image $I$, and $C(p_j)$ is its corresponding score in the detection confidence map $C$ computed by the original predictor (usually a CNN). High confidence regions correspond to areas where pixels present scores near the extremes of the likelihood range. For normalized score maps, values lower than a threshold $t_b \sim 0.0$ identify pixels in the high confidence background, while values larger than $t_f \sim 1.0$ indicate high confidence foreground elements. To recover possible false negatives, morphological shrinking is performed on the background $R_B$ boundaries. The fourth leftmost image in Figure 3.1 illustrates the division of the image into the three regions $R_F$, $R_U$, and $R_B$.

**Algorithm 1** RGR refinement algorithm

**Input:** Image $I$, confidence map $C$
**Output:** Refined semantic segmentation $\hat{Y}$ of image $I$

```
Threshold C into three regions: background R_B, foreground R_F and uncertain zone R_U
Define a Region of Interest (RoI) according to Eq. 3.2
for i = 1 to n_s do
    Form a set S^(m) of initial seeds by uniformly sampling from R_B ∪ R_F
    Generate a set of clusters π^(m) by performing region growing using the SNIC algorithm with S^(m) as input
    For each generated cluster ψ_k^(m) ∈ π^(m), compute confidence map C̄^(m) according to Eq. 3.4
end for
Compute the pixel-wise average C̄ of C̄^(m), m = 1, ..., n_s
Generate Ŷ by pixel-wise thresholding C̄
```

*2) Step 2 - Monte Carlo sampling of initial seeds:* Inspired by the notion of pixel affinity employed by many algorithms for superpixel segmentation, our method applies a region growing approach to classify pixels within the uncertainty zone. More specifically, we build on the simple non-iterative clustering (SNIC) algorithm [117].

SNIC selects seeds on a regular grid over the whole image. In contrast, our algorithm selects seeds by Monte Carlo sampling only high confidence regions. Such an adaptation provides three main advantages for segmentation refinement. First, our random selection of seeds allows clusters of flexible size. This is beneficial for: i) growing into larger regions that were missed by the predictor, and ii) forming smaller clusters, rather than *leaking* into nearby pixels. Second, it enforces the classification of unlabeled pixels to derive from high confidence information. And third, at each Monte Carlo iteration, clusters are grown from different sets of randomly selected seeds. Combined with majority voting per cluster and pixel-wise averaging across iterations (detailed in Step 3), this procedure acts as a filter against false positives detected with high confidence by the predictor.

Our Monte Carlo approach consists of $n_s$ iterations. At each iteration $m$, a set $S^{(m)}$ of initial seeds is defined by uniformly sampling the high-confidence area $R_H = R_B \cup R_F$. Let $p_h$ represent pixels within the region $R_H$, where the index $h = 1, \ldots, |R_H|$. Uniform sampling of seeds can thus be performed index-wise according to $h \sim U(1, |R_H|)$. We determine the number of seeds to be sampled $|S^{(m)}|$ based on the sampling domain area (i.e., $|R_H|$) and the desired average spacing between samples $\gamma$, i.e., $|S^{(m)}| = {}^{|R_H|}/_{\gamma}$. The spacing between seeds ensures the availability of paths through which all the initial centroids can propagate throughout the uncertainty region.

*3) Step 3 - Region Growing:* The dashed block at the bottom of Figure 3.1 illustrates the sequence of operations performed by RGR for region growing. To reduce the computation time, we restrict the region growing to a Region of Interest (RoI)

around the uncertain zone $R_U$. Based on the spatial distance to $R_U$, the background $R_B$ can be split into two regions: far background $R_{fB}$ and near background $R_{nB}$. Since the far background is unlikely to influence the clustering of the uncertain region, $R_{fB}$ can be ignored during region growing. Hence, we define the RoI for region growing as

$$RoI = R_{nB} \cup R_F \cup R_U. \tag{3.2}$$

Initial centroids are then grown according to an adaptation of the SNIC algorithm. As in SNIC, we measure the similarity between a pixel and a centroid as their distance in a five-dimensional space of color and spatial coordinates. Let the spatial position of a pixel $p_j$ be represented by a vector $\mathbf{x_j} = [x_j,\ y_j]^T$, while its color is expressed in the CIELAB color-space by $\mathbf{c_j} = [l_j,\ a_j,\ b_j]^T$. The distance $d\left(p_j, \psi_k\right)$ between the $j-th$ pixel and the $k-th$ centroid is given by Eq. 3.3, where $\theta_s$ and $\theta_m$ are normalizing factors for spatial and color distances, respectively

$$d\left(p_j, \psi_k\right) = \sqrt{\frac{\left\|\mathbf{x}_j - \mathbf{x}_k\right\|_2^2}{\theta_s} + \frac{\left\|\mathbf{c}_j - \mathbf{c}_k\right\|_2^2}{\theta_m}}. \tag{3.3}$$

Also as in SNIC, our region growing implementation relies on a priority queue, which is constantly populated with nodes that correspond to unlabeled pixels 4- or 8-connected to a region being grown. This queue is sorted according to the similarity between the candidate pixel and the average (centroid) of the growing region, given by Eq. 3.3. While the queue is not empty, each iteration consists of: 1) popping the first element of the queue, which corresponds to the unlabeled pixel that is most similar to a neighboring centroid $k$; 2) annexing this pixel to the respective cluster $\psi_k^{(m)}$; 3) updating the region centroid; and 4) populating the queue with neighbors of this pixel that are still unlabeled.

We add a constraint to the original SNIC algorithm in order to reduce the incidence of false detections. A node is only pushed into the queue if its distance to

the corresponding centroid is smaller than a certain value $d_{Max}$. This strategy ensures that an unlabeled pixel within $R_U$ will only inherit information from high confidence pixels that are sufficiently similar to it. This creates the possibility of "orphan" elements, i.e., pixels for which no node is ever created. Such pixels are therefore classified as background. For each set of initial seeds $S^{(m)}$, the region growing process generates a cluster map $\pi^{(m)}$ that associates each pixel to a respective cluster $\psi_k^{(m)}$.

*4) Step 4 - Majority voting and final classification:* Following the region growing process, RGR conducts a majority voting procedure to ultimately classify each generated cluster into foreground or background. As expressed in Eq. 3.4, a pixel $p_j$ contributes a positive vote for foreground classification if its original prediction score $C(p_j)$ is larger than a threshold $t_0$. We compute the ratio of positive votes across all pixels $p_j \in \psi_k^{(m)}$ to generate a refined likelihood map $\bar{C}^{(m)}$ for each set of clusters according to

$$Y_k^{(m)} = \left\{ p_j \in \psi_k^{(m)} | C(p_j) > t_0 \right\}, \tag{3.4}$$

$$\bar{C}^{(m)}(p_j) = \frac{|Y_k^{(m)}|}{|\psi_k^{(m)}|}. \tag{3.5}$$

The likelihood maps $\bar{C}^{(m)}$ obtained from the $n_s$ Monte Carlo samplings are averaged to generate a final pixel dense score map $\bar{C} = \frac{1}{m}\sum_{i=1}^{m} \bar{C}^{(m)}$. Finally, each pixel is classified into foreground if more than 50% of its average votes are positive. Otherwise, the region is labeled as background, that is,

$$\hat{Y}(p_j) = \mathbb{1}_{\bar{C}(p_j)>0.5}, \tag{3.6}$$

where $\hat{Y}(p_j)$ is the final binary classification map and $\mathbb{1}$ is an indicator variable that assumes the value 1 when the corresponding condition is satisfied.

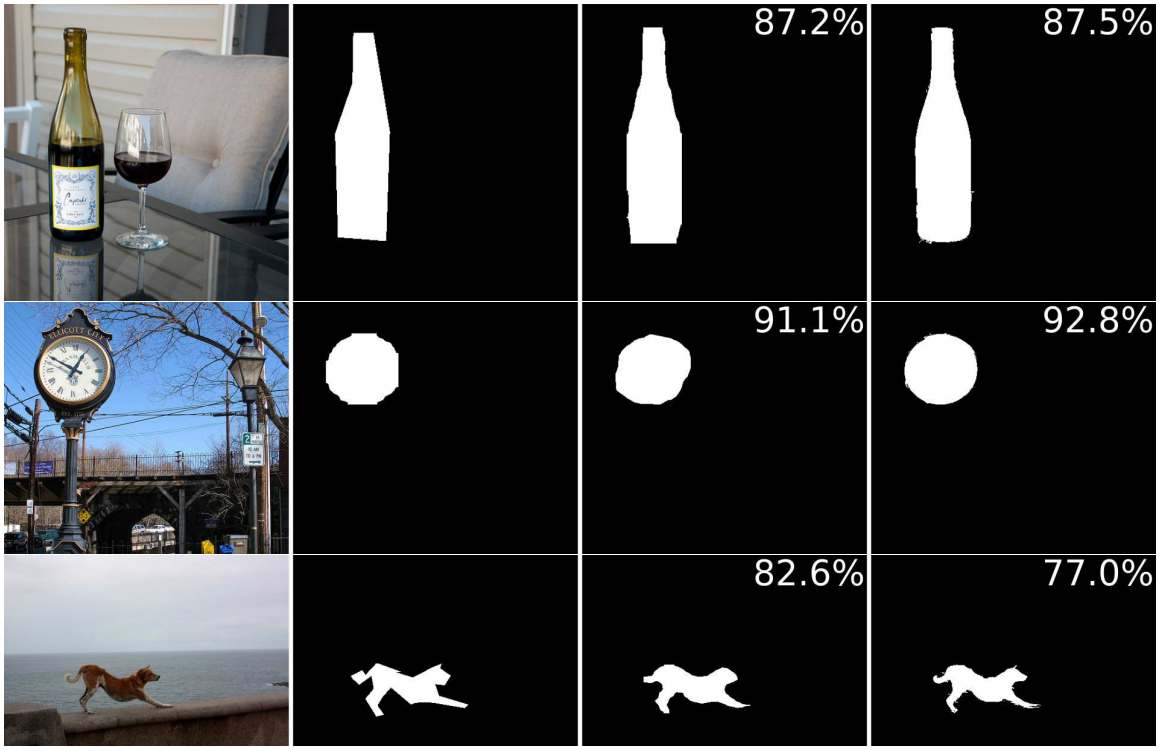



Figure 3.2: Additional examples of imperfect ground-truth annotations in the COCO 2016 dataset. From left to right: original image, ground truth, FCIS detection, FCIS+RGR. The obtained $IoU$ with respect to the ground-truth is displayed above each corresponding detection [1] (©2018 IEEE).

### 3.1.2 Experiments

To the best of our knowledge, the COCO dataset is the largest publicly available dataset for semantic/instance segmentation, with a total of 2.5 million labeled instances in $328,000$ images [5]. However, as discussed in Section 3.1.2.1 and further illustrated in Figure 3.2, COCO's ground truth annotations contain imprecisions intrinsic from its labeling procedure. To minimize the influence of dataset-specific errors, we assess the performance of our algorithm on: i) the COCO 2016 validation set; ii) the PASCAL VOC 2012 dataset; and iii) selected video sequences of the DAVIS dataset.

Since RGR is an unsupervised post-processing refinement module, it can be coupled to any semantic segmentation network. In Section 3.1.2.1, we evaluate its performance in comparison to SNIC superpixels for refinement of FCIS predictions. In Section 3.1.2.2, RGR, CRF, DT-EdgeNet and GrabCut are compared for refinement of DeepLab predictions.

#### 3.1.2.1 Comparison Against Superpixel Refinement

We denote the combination of the publicly available FCIS model and the refinement module RGR as FCIS+RGR. Since RGR works in a region growing manner that is inspired by the concept of superpixel segmentation, our performance analysis also includes FCIS+SNIC, a naive refinement method that consists of performing majority voting within superpixels generated with SNIC.

Following a grid-search executed on the PASCAL VOC dataset, for all our experiments FCIS+SNIC employs SNIC with $\theta_m = 0.1$ and superpixel size of $100px$. RGR thresholds were defined as follows. First, $t_0$ corresponds to the original detector optimal threshold. For FCIS this value is 0.4, as reported in [24]. To identify the high confidence foreground, we empirically selected a high confidence threshold $t_f$ corresponding to $1.5 \times t_0$, hence 0.6. As for the background detection, we set the high confidence lower threshold to $t_b = 0.0$.

**COCO 2016 Segmentation Dataset.** Based on the COCO official guidelines, we evaluate the performance obtained by FCIS, FCIS+SNIC, and FCIS+RGR on the validation set composed of $40k$ images. Table 3.1 summarizes the performance of the three methods according to the standard COCO metrics, including average precision ($AP$) averaged over $0.5 : 0.95$ intersection over union (IoU) thresholds and at different scales. While increasing the number of true positive detections (+0.3% in $AR$) for all

the scenarios, the naive FCIS+SNIC approach also decreases the $AP$ by introducing a larger number of false positives.

Table 3.1: Comparison between results obtained by FCIS, FCIS+SNIC and FCIS+RGR on the COCO 2016 (val), the PASCAL VOC 2012 (val), and the DAVIS datasets [1] (©2018 IEEE).

| | **COCO 2016** | | | | | | **VOC 2012** | **DAVIS** | |
|---|---|---|---|---|---|---|---|---|---|
| | $AP$ (%) | $AP_{50}$ (%) | $AP_{75}$ (%) | $AP_S$ (%) | $AP_M$ (%) | $AP_L$ (%) | $mIoU$ (%) | $\mathcal{J}(IoU)$ (%) | $\mathcal{F}$ (%) |
| FCIS | 35.1 | 60.1 | 36.5 | 9.8 | 38.4 | 59.8 | 70.6 | 71.2 | 69.9 |
| FCIS + SNIC | 34.9 | 59.0 | 36.4 | 9.3 | 38.2 | 59.6 | 70.6 | 72.8 | 70.4 |
| **FCIS + RGR** | **36.9** | **60.6** | **39.3** | **11.4** | **40.7** | **60.5** | **71.1** | **74.4** | **72.7** |

Our refinement model, RGR, on the other hand, increases the baseline FCIS overall performance by 1.8%. Compared to the improvement of 0.5% in $AP_{50}$, the increase of 2.8% in $AP_{75}$ demonstrates that RGR is particularly successful for cases where the detections obtained from the input CNN are accurately located.

Figure 3.3 presents the average precision obtained for each of the 80 COCO categories. Since its region growing is based on local affinity characteristics, it is natural that the RGR refinement is especially effective for objects with more uniform appearance (e.g. airplane, frisbee). Nevertheless, these results also demonstrate the robustness of the refinement provided by RGR, since no object category shows a noticeable decrease in average precision.

A closer visual inspection of the output labels also shows that the metrics obtained might be misleading. Despite the extensive efforts described in [24] to create such a massive dataset, for some instances the ground truth segmentation annotations lack accurate adherence to real object boundaries. As a consequence, improvements

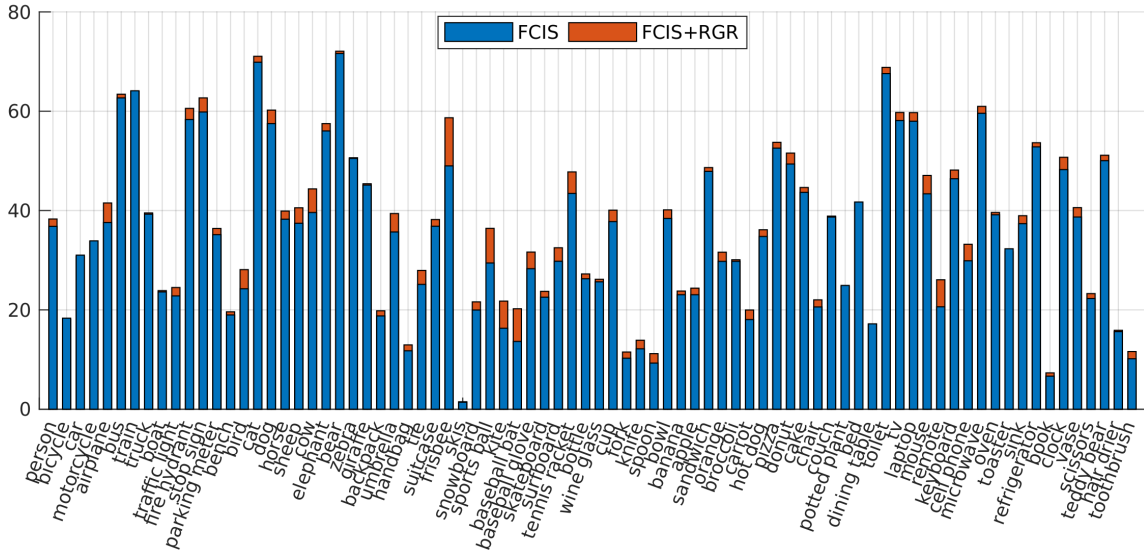



Figure 3.3: *AP* obtained by FCIS and FCIS+RGR for each of the 80 COCO categories [1] (©2018 IEEE).

obtained through RGR refinement are not reflected in the final metrics for a significant number of images. Figure 3.4 provides examples of the imprecisions in ground truth annotations, collected from different object classes. While for the *airplane* example the segmentation obtained using RGR is clearly better in qualitative terms, its overlap (IoU) with the ground truth is almost 7.0% lower than the one associated with the FCIS output.

**PASCAL VOC 2012 Segmentation Dataset.** Table 3.1 also contains a summary of the results obtained by FCIS, FCIS+SNIC and our method FCIS+RGR on the PASCAL VOC 2012 validation set. The segmentation refinement generated using RGR provides a final $mIoU$ slightly better (+0.5%) than both FCIS and the refined version using naive superpixel-wise majority voting.

However, such a small difference in performance does not properly reflect the higher boundary adherence provided by RGR refinement. First, boundaries constitute

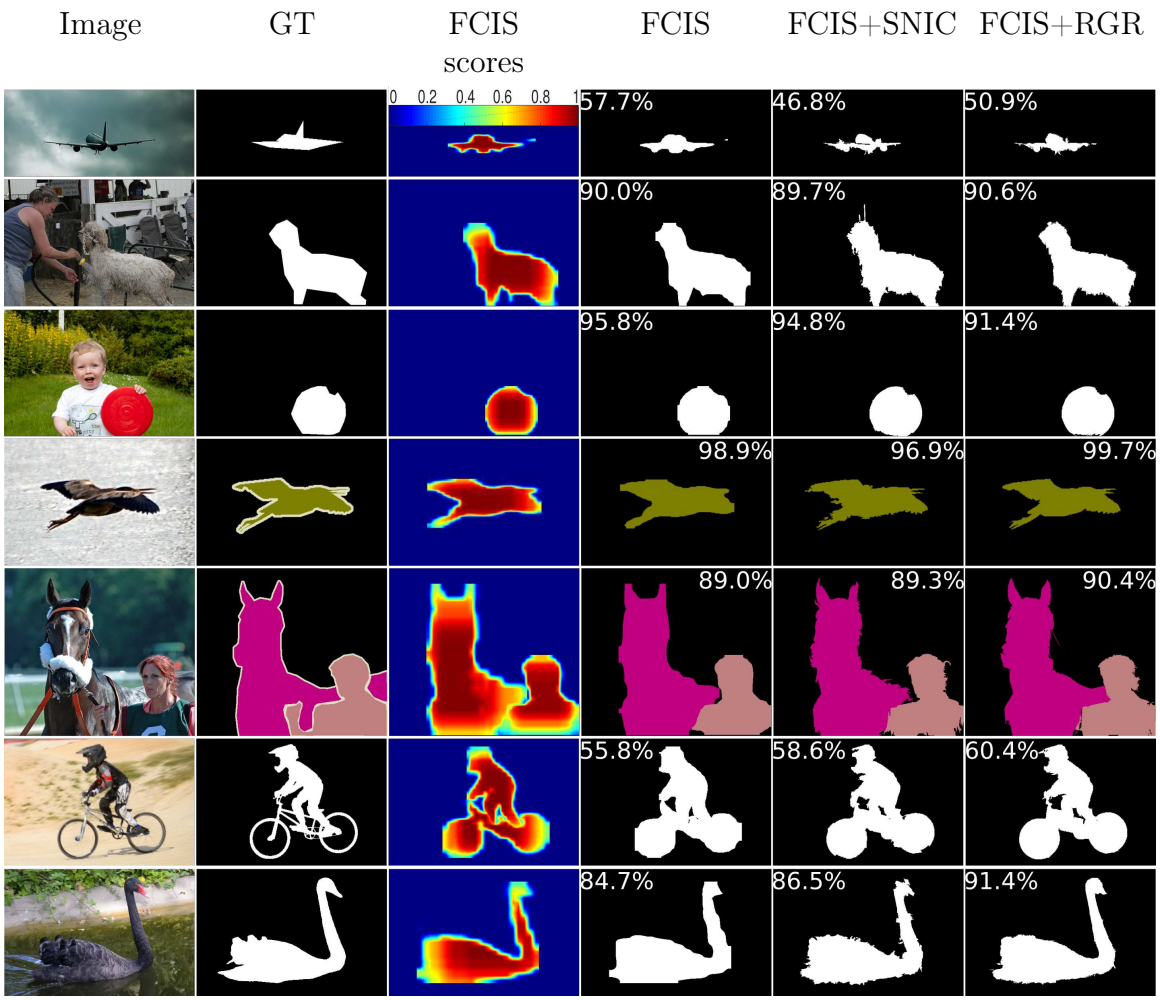



Figure 3.4: Examples of detections on the COCO (three top-most rows), PASCAL (fourth and fifth rows) and DAVIS (two bottom-most rows) datasets. From left to right: original image, ground truth, FCIS scores, FCIS detection, FCIS+SNIC, FCIS+RGR. The obtained $AP$ (COCO), $mIoU$ (PASCAL) and $IoU$ (DAVIS) are displayed above each corresponding detection. Ground truth annotations in the first and second rows are also examples of COCO annotations with poor boundary adherence [1] (©2018 IEEE). Appendix A includes additional qualitative examples.

only a small fraction of the total image pixels that are considered for performance computation. Moreover, ground-truth annotations composing the PASCAL dataset are structured such that the evaluation metrics disregard regions 5 pixels-wide around

the boundaries of each object. As a consequence, often clear improvements in terms of boundary adherence are not reflected in overall mean intersection over union ($mIoU$).

Thus, we follow the strategy presented in [52] and also evaluate segmentation accuracy on a narrow region closer to the boundaries to better quantify the improvements obtained by RGR. Figure 3.5 shows the mean IoU obtained according to the width of the evaluated region around boundaries. RGR outperforms FCIS+SNIC by +2.0% in terms of mean $IoU$ in regions 10-pixels near the boundaries, which better reflects the more detailed segmentation provided by RGR that is clear in the qualitative examples such as in Figure 3.4.

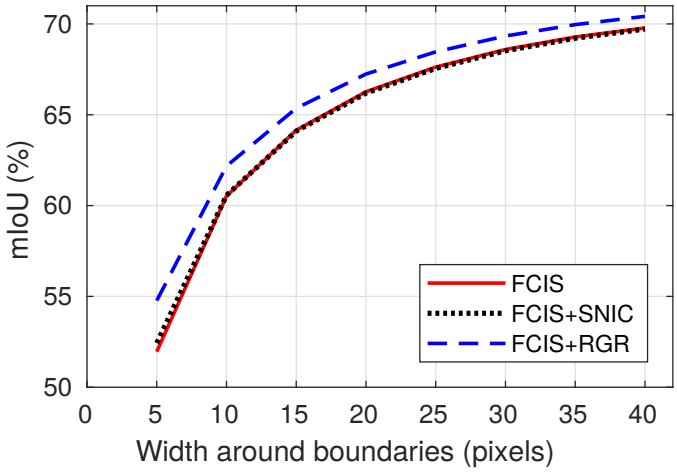



Figure 3.5: Mean IoU of FCIS, FCIS+SNIC and FCIS+RGR on PASCAL on regions near object boundaries [1] (©2018 IEEE).

**DAVIS 2016 Selected Video Sequences.** Given the aforementioned inaccuracies in the annotations available for the COCO and PASCAL datasets, we additionally report quantitative results on selected video sequences of the DAVIS dataset. The DAVIS dataset for Video Object Segmentation is composed of high quality video sequences [7], with pixel-accurate ground truth segmentation for each frame. From its

50 original video sequences, we selected a total of 24 sequences that contain target objects whose corresponding classes are contained within the 80 objects categories composing the COCO dataset. These sequences encompass 13 different COCO classes, including classes for which FCIS+RGR did not provide significant performance improvements on the COCO evaluation (e.g. *person, bicycle, boat* and *bear*).

As summarized in Table 3.1, segmentations obtained using FCIS+RGR have an average $\mathcal{J}(IoU)$ 3.2% higher than those generated by FCIS, and 1.6% better than the results obtained using a naive superpixel-wise majority voting for refinement. As described in Section 2.4.2, the official metrics for performance evaluation on DAVIS also include the contour accuracy metric $\mathcal{F}$. Segmentations refined using RGR yielded an increase of 2.8% in $\mathcal{F}$, confirming its ability to improve boundary adherence. Figure 3.6 presents the results obtained for each video sequence, with FCIS+RGR providing improvements in the range between 1.1% and 6.9% depending on the sequence. The fact that larger quantitative improvements are obtained for the DAVIS sequences corroborates our argument that the annotations available for the COCO and PASCAL datasets provide limited information regarding boundary accuracy/adherence of segmentation methods.

#### 3.1.2.2 Comparison Against CRF, DT-EdgeNet and GrabCut

We also compare the performance of our method against the state-of-the-art DenseCRF [104], which has been successfully exploited by DeepLab models for semantic segmentation refinement, as well as its alternative DT-EdgeNet [105] and the GrabCut model [69]. We adopt the publicly available model of DeepLab-LargeFOV as the base model for our comparison. This model is pre-trained on MS COCO and fine-tuned on the augmented *trainval* PASCAL VOC 2012 dataset.

We perform our evaluation for the refinement of the segmentations produced by the DeepLab-LargeFOV model on the PASCAL validation set. In this case, we

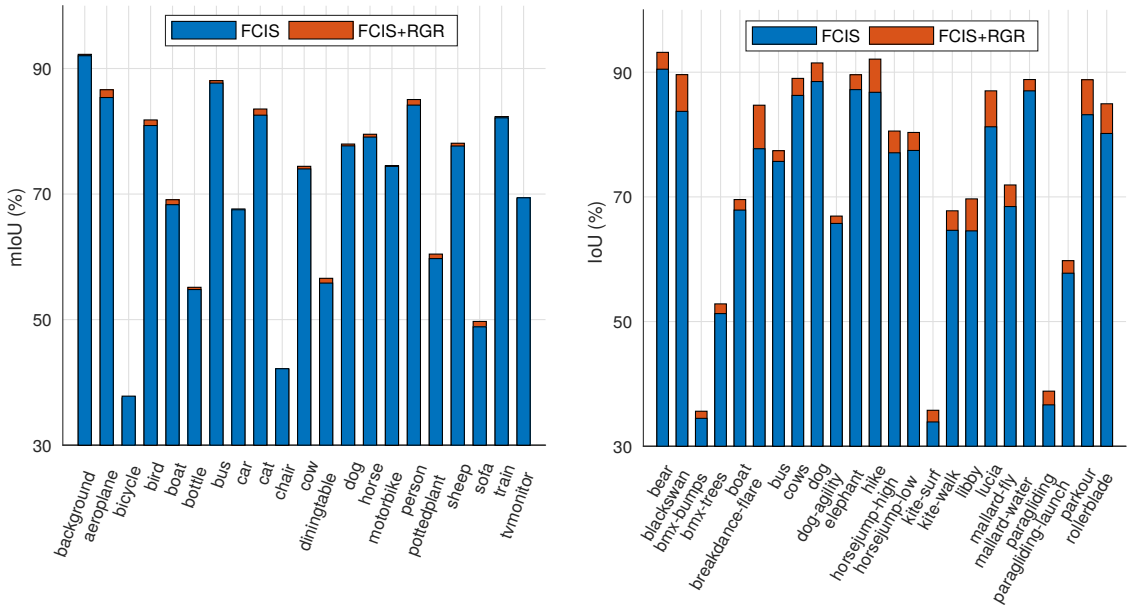



Figure 3.6: Results obtained obtained by FCIS and FCIS+RGR on transfer learning experiments. *Left:* IoU for each of the PASCAL VOC 2012 classes (val). *Right:* IoU obtained for each of the selected DAVIS video sequences [1] (©2018 IEEE).

configure RGR to use a different $t_f$ sampled from the distribution $U(0.5, 0.9)$ in each MC region growing iteration. The other thresholds remain fixed at $t_b = 0.0$ and $t_0 = 0.5$.

Figure 3.7 provides a visual comparison between segmentation refinements obtained using GrabCut, CRF, DT-EdgeNet and our RGR algorithm. In the first two images, we observe that RGR refinement recovers fine details of both the train's and the boat's structures. Similarly, the person's fingers on the third image, the bird's beak and feathers in the fourth one and the airplane's gears in the fifth image highlight the level of segmentation details recovered by RGR from the coarse segmentations shown in the third column. We additionally point out the last two examples, where the limbs of the person and the sheep are recovered from a very coarse segmentation.

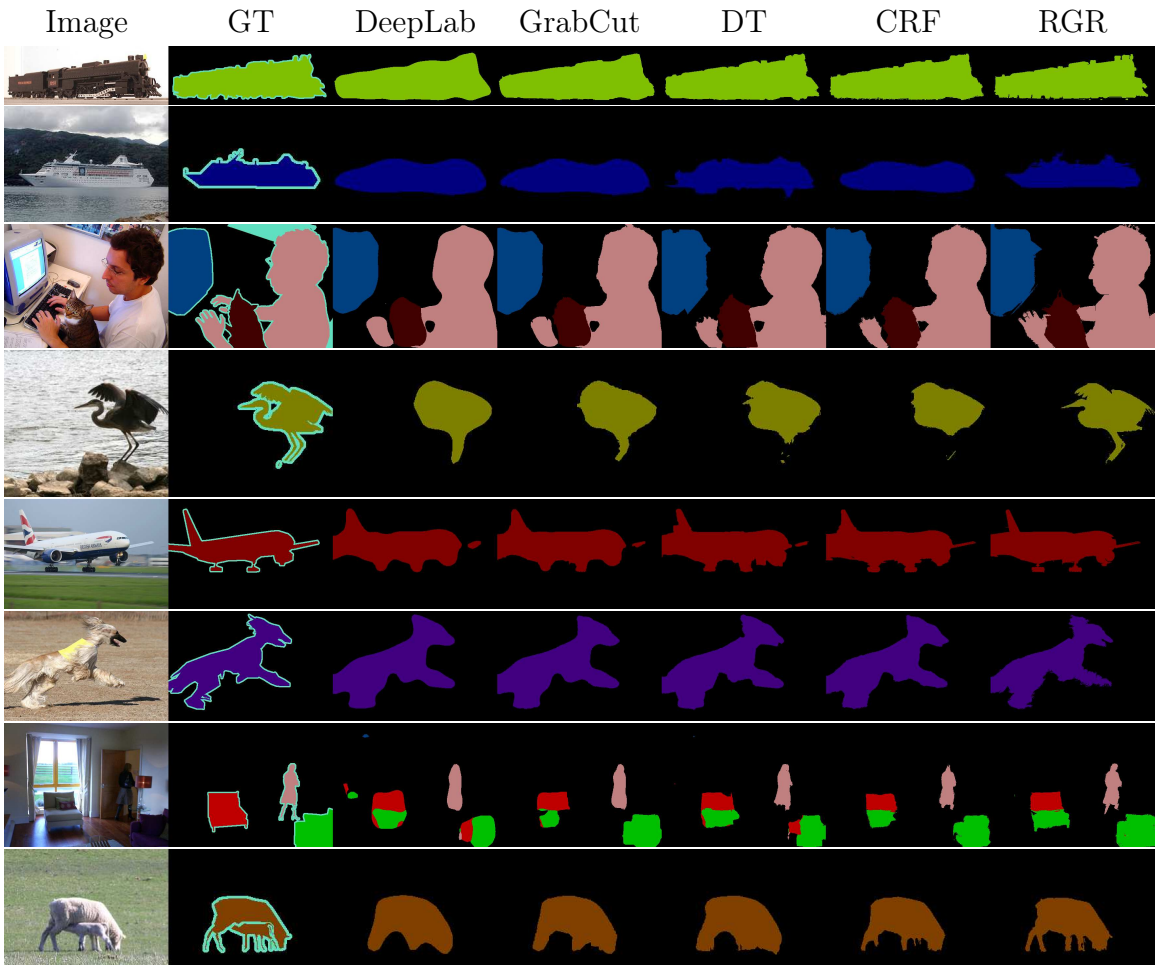



Figure 3.7: Examples of detections on the PASCAL VOC 2012 dataset. From left to right: original image, ground truth, Deeplab detection, GrabCut refinement, DT-EdgeNet refinement, CRF refinement, RGR refinement [1](©2018 IEEE).

Quantitative results are summarized in Table 3.2. GrabCut demonstrated limited ability to refine the CNN predictions, which is expected since: i) it assumes that all pixels initialized as background/foreground are correct, and ii) the formulation using GMMs shows limited performance when the background/foreground appearance varies significantly. In contrast, by growing from a much higher number of seeds, RGR forms multiple clusters that significantly differ in terms of appearance but can share the same semantic labels.

The combination of an optimized CRF with DeepLab-LargeFOV culminates in a $mIoU$ of 80.1%, while the alternative using optimal DT-EdgeNet provides 78.9%. Our fully unsupervised RGR algorithm yields 79.2% $mIoU$ on this same scenario, being hence competitive with both optimized supervised models but with the advantage of not requiring any dataset-specific fine-tuning.

Table 3.2: Comparison between different refinement methods for different networks [1] (©2018 IEEE).

| | **VOC 2012** | | **DAVIS** |
|---|---|---|---|
| | *mIoU (%)* | | *IoU (%)* |
| DeepLab | 76.1 | FCIS | 71.2 |
| DeepLab+GrabCut | 77.9 (+1.8) | FCIS+GrabCut | 71.2 (+0.0) |
| DeepLab+DT | 78.9 (+2.8) | FCIS+DT | NA* |
| DeepLab+CRF | 80.1 (+3.9) | FCIS+CRF | 73.9 (+2.7) |
| DeepLab+RGR | 79.2 (+3.1) | FCIS+RGR | 74.4 (+3.2) |

*implementation public available only for the DeepLab model.

The higher generalization capability of our method is highlighted by transfer learning experiments. We compared RGR and CRF for the refinement of FCIS detections in the scenario described in Section 3.1.2.1, i.e., on the same selected sequences of the DAVIS dataset but without performing any dataset-specific fine-tuning for any method. While RGR again improves segmentation quality by 3.2%, in this scenario CRF provides only 2.7%. This demonstrates the advantage of our unsupervised approach, which can be employed on different datasets without the need for fine-tuning. Albeit providing significant improvements in segmentation performance,

the DenseCRF is strongly dependent on supervised optimization of hyper-parameters for specific datasets and predictor models.

#### 3.1.2.3 Inference Time

Finally, we evaluate the inference time of RGR in comparison with dense CRF. As explained above, our algorithm consists of four main steps: 1) thresholding, 2) Monte Carlo sampling of seeds, 3) region growing, and 4) majority voting. Steps 1, 2 and 4 are currently implemented in MATLAB® (R2017a), while the region growing step is implemented in C++ based on the SNIC algorithm. Runtime assessment was performed on an Intel Xeon™CPU E5-2620 v3 @ 2.40GHz (62GB). The average runtime per image in the PASCAL VOC dataset is $\sim 0.5$ seconds with 3 Monte Carlo iterations. This is lower than the 0.8 seconds average inference time of CRF's publicly available implementation, as reported in [105].

A breakdown of the runtime analysis shows that the region growing step requires only $\sim 0.2$ seconds per Monte Carlo (MC) iteration, which is comparable to the 0.18 seconds required by the CPU-based implementation of the domain transform method described in [105]. We highlight that the multiple MC iterations are easily parallelizable. As summarized in Figure 3.8, using MATLAB's Parallel Pool the runtime obtained for 10 MC iterations is less than 0.1 seconds higher than the 0.5 sec/image obtained for 3 MC iterations, which corresponds to a segmentation performance improvement of 0.1% in the scenario described in Section 3.1.2.1. Appendix A includes further analysis of performance sensitivity to parameters.

#### 3.1.2.4 Failure Cases

As demonstrated by the experimental results, the quality of the refinement obtained using RGR depends on the accuracy of the detector model used as input, especially in terms of localization. Although Monte Carlo sampling improves robust-

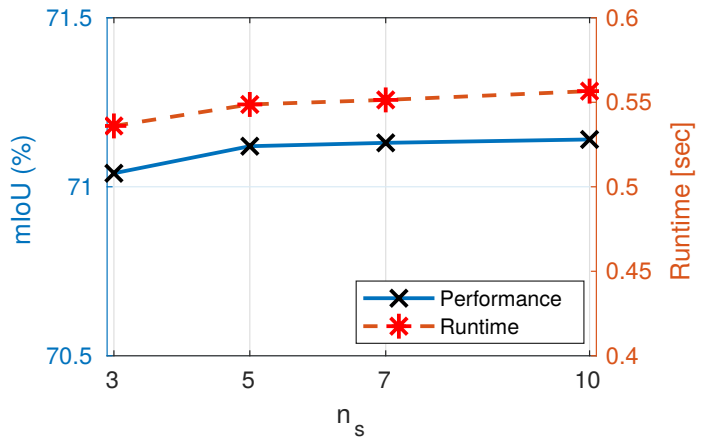



Figure 3.8: Runtime (orange) and performance (blue) of RGR according to the number of MC iterations ($n_s$) in the scenario described in Section 3.1.2.1 [1] (©2018 IEEE).

ness against high confidence false positives, errors might be propagated if the score maps generated by the detector module (CNN) contain regions with high concentrations of false positives. An example of such a case is found in Figure 3.4. Given the high confidence of false positives generated by the FCIS model for internal parts of the bicycle wheels, RGR is unable to correct these mistakes and hence these regions remain incorrectly segmented.

## 3.2 probabilistic Region Growing Refinement (pRGR)

In this section, we present the *probabilistic Region Growing Refinement (pRGR)* algorithm [116], an extension of RGR that provides the following contributions:

- *a solid mathematical foundation that exploits a probabilistic framework to guide all the steps of the algorithm;*
- combining techniques from Bayesian estimation, *many parameters that were previously determined in an ad-hoc manner are now initialized using Bayesian conjugate priors and updated as assignments of pixels to clusters occur.* More-

over, variance reduction techniques are exploited to optimize the sampling steps within the Monte Carlo refinement iterations;

- with *a novel parameterization that allows for the emulation of varied receptive field sizes, pRGR further improves segmentation refinement performance* by recovering finer boundary details and attenuating the effects of false-positive pixel labels;
- we experimentally demonstrate the applicability of pRGR in a variety of scenarios that include state-of-the-art models such as DeepLabV3+ [98]. Such *experiments also suggest the combination of DenseCRF [104] and pRGR as a powerful strategy for segmentation refinement;*
- we observe that *the variance of pRGR's Monte Carlo estimations can be exploited as an uncertainty estimation mechanism*, with experiments demonstrating its high correlation with final segmentation accuracy values;
- upon publication, code will be made available at `coviss.org/code`.

We report experiments using different CNNs, datasets and baselines. For easy comparison against DenseCRF and RGR baselines, we first report experiments on refinement of segmentation predictions provided by DeepLab [88] and DeepLabV2 [22] for the PASCAL VOC 2012 [43] validation set. We then report experiments conducted with the state-of-the-art DeepLabV3+ [98] segmentation model on the PASCAL *val* set and also on selected sequences from the DAVIS dataset [7].

### 3.2.1 Proposed approach

In this section, we describe the sequence of steps and corresponding mathematical formulation that comprise our probabilistic Region Growing Refinement (pRGR) method. Similar to RGR, *the proposed pRGR algorithm is a generic unsupervised post-processing module* for refinement of segmentation boundaries that can be coupled with the output of any CNN or similar model for semantic segmentation.

While sharing similar concepts, pRGR advances RGR by employing a probabilistic formulation in which all the steps of the algorithm are derived using a mathematically coherent framework. In addition, concepts of variance reduction and Bayesian estimation are used for the initialization and update of parameters in a principled manner.

The main operations composing pRGR are summarized in Figure 3.9. At a high level, the steps performed by both RGR and pRGR can be summarized as:

1. identification of high confidence classification regions;
2. Monte Carlo seed sampling from high-confidence regions;
3. region growing of seeds into clusters;
4. pixel-score averaging within clusters;
5. averaging across multiple Monte Carlo iterations.

In the case of multi-class segmentation, both RGR and pRGR perform these steps on the scoremaps associated with each class, and the final classification is defined by computing the maximum likelihood across classes. In the remainder of this section, we justify these operations and derive the set of equations guiding the steps composing our method.

#### 3.2.1.1 Probabilistic seed sampling from high-confidence regions

Let the inputs for our refinement algorithm be represented as an observed image $I \in \mathbb{R}^{w \times h}$ and corresponding confidence maps $C \in \mathbb{R}^{w \times h \times \mathcal{C}}$. Here, $w \times h$ are the dimensions of the input image $I$, and $C$ are the scoremaps for each class in the set $\mathcal{C}$, generated by any modern segmentation CNN. For simplicity, we first introduce the method for the binary case where $|\mathcal{C}| = 1$, as all steps are performed on each class scoremap independently in the multiclass scenario.

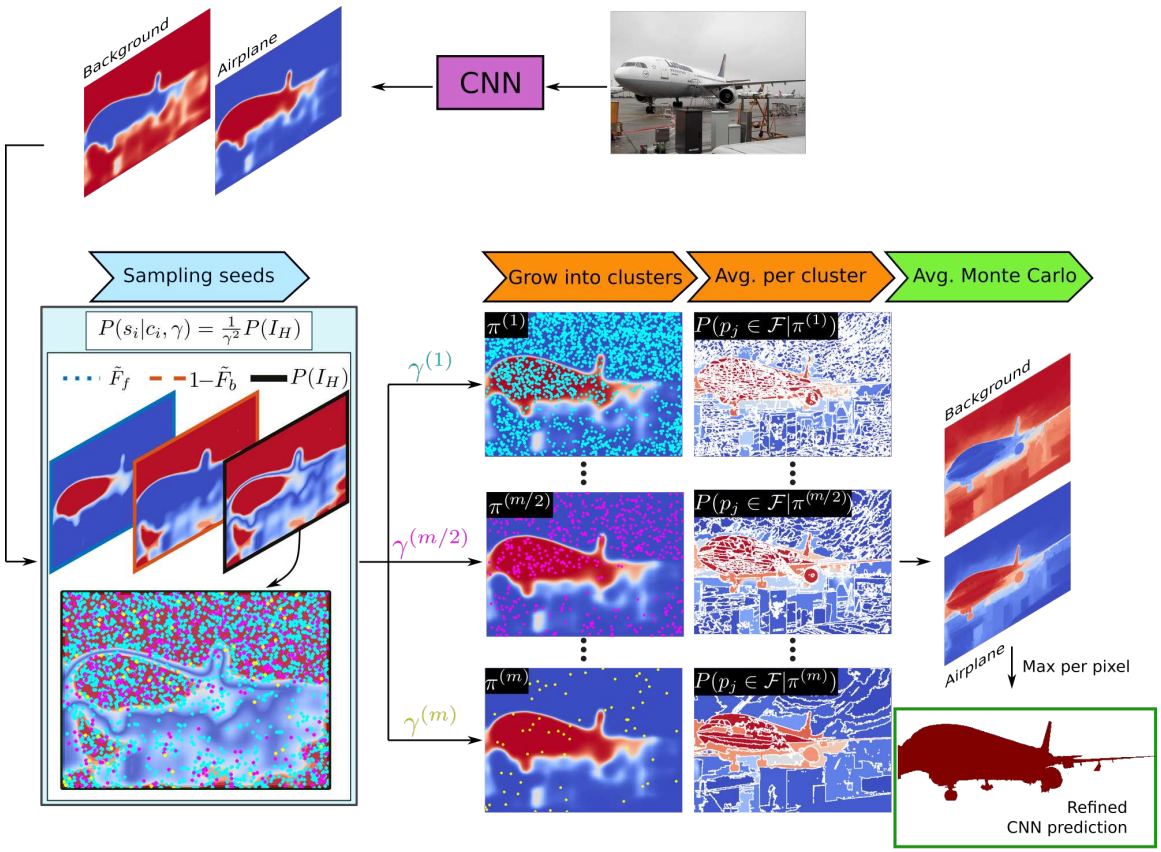



Figure 3.9: Diagram illustrating the sequence of steps performed by the proposed pRGR model for segmentation refinement. Each step and their corresponding equations are discussed in detail in Section 3.2.1.

Let $\pi$ represent the partition of $I$ into a set of clusters $\pi = \{\psi_1, \psi_2, \ldots, \psi_{|S|}\}$, which are grown from the set of seeds $S = \{s_1, s_2, \ldots, s_{|S|}\}$. To estimate the probability that a pixel $p_i$ should be sampled as a high-confidence seed $s_i$, let the thresholds defining high-confidence background and high-confidence foreground be denoted by $t_b$ and $t_f$, respectively. From that, we define $I_H = \{c_i < t_b \text{ or } c_i \geq t_f\}$ as the event that a pixel with confidence score $c_i$ belongs to a high-confidence background or foreground

region. The probability $P(I_H)$ is thus given by

$$\begin{aligned}
P(I_H) =& P(c_i < t_b \text{ or } c_i \geq t_f) \\
=& P(c_i < t_b) + P(c_i \geq t_f) - P(c_i < t_b)P(c_i \geq t_f) \\
=& 1 - P(c_i \geq t_b) + P(c_i \geq t_f) - [-1 - P(c_i \geq t_b)]P(c_i \geq t_f) \\
=& 1 - P(c_i \geq t_b) + P(c_i \geq t_b)P(c_i \geq t_f) \\
=& 1 - F_b\left(c_i\right) + F_b\left(c_i\right) F_f\left(c_i\right), \qquad (3.7)
\end{aligned}$$

where $F_b(\cdot)$ and $F_f(\cdot)$ are the cumulative density functions (CDFs) corresponding to the distributions of $t_b$ and $t_f$, respectively.

As discussed in [1], sampling with a spacing $\gamma$ between seeds ensures the availability of paths for them to grow throughout the uncertainty region. That is, seeds are uniformly sampled among $\gamma \times \gamma$ points within the high-confidence region, such that, given the thresholds $t_f, t_b$ and the inter-seed spacing $\gamma$, the probability of sampling a seed $s_i$ at a pixel with confidence score $c_i$ is

$$P(s_i | c_i < t_b \text{ or } c_i \geq t_f, \gamma) = \frac{1}{\gamma^2}. \qquad (3.8)$$

While in RGR the seed spacing $\gamma$ is fixed for all sample-grow iterations, for pRGR we adopt a strategy where $\gamma$ is itself sampled in a stratified manner from a uniform distribution $\gamma \sim \mathcal{U}(\gamma_l, \gamma_h)$, where $\gamma_l$ and $\gamma_h$ are the minimum and the maximum spacing values. As indicated by Eq. 3.8, the parameter $\gamma$ directly impacts the number of seeds to be sampled, which is inversely proportional to the expected sizes of the clusters to be formed through seed growing. Hence, *sampling $\gamma$ using a stratified approach allows for the emulation of the refinement process at multiple receptive field sizes*, a common practice exploited in many modern segmentation architectures [95, 98].

Since $t_l$ and $t_h$ are independent of $\gamma$, we have

$$\begin{aligned} P(s_i, I_H|\gamma) =& P(s_i|I_H, \gamma)P(I_H) \\ =& \frac{1}{\gamma^2}\left[1 - F_b\left(c_i\right) + F_b\left(c_i\right)F_f\left(c_i\right)\right]. \end{aligned} \tag{3.9}$$

Marginalizing over the event $I_H$,

$$\begin{aligned} P(s_i|\gamma) &= P(s_i, I_H|\gamma) + P(s_i, \bar{I}_H|\gamma) \\ &= P(s_i, I_H|\gamma), \end{aligned} \tag{3.10}$$

where the second equation is based on the fact that seeds are sampled only from the high-confidence region, i.e., $P(s_i|\bar{I}_H, \gamma) = 0$.

Let $m = 1, ..., n_s$ represent the index of a Monte Carlo growing iteration, such that $s_i^{(m)}$ represents the $i$-th seed in iteration $m$, and let $\gamma^{(m)}$ be the corresponding inter-seed spacing. Based on Eqs. 3.9 and 3.10, the seed samples are distributed according to

$$s_i^{(m)} \sim \frac{1}{\left(\gamma^{(m)}\right)^2}\left[1 - F_b\left(c_i\right) + F_b\left(c_i\right)F_f\left(c_i\right)\right]. \tag{3.11}$$

**Thresholds distribution**

Semantic segmentation methods based on deep-learning models typically comprise three main steps. First, a CNN computes unbounded scoremaps with the activations of each pixel for each class. By applying a softmax function across all the classes for each pixel, these scoremaps are then normalized into the range $[0, 1]$. Finally, class labels are assigned to each pixel through an $\arg\max$ operation across the normalized scoremaps.

Therefore, no single fixed threshold is applied to the class scoremaps for classification. Hence, to estimate the CDFs $F_b, F_f$ required in Eq. 3.9, we approximate them using two non-parametric distributions $\tilde{F}_b$ and $\tilde{F}_f$. As depicted in Figure 3.10,

from the output of the arg max step we identify the pixels $p_f \in \mathcal{F}$ labeled as foreground and the pixels $p_b \in \mathcal{B}$ labeled as background. For a scenario of multiple classes such as the one illustrated in Figure 3.10, foreground corresponds to pixels labeled as part of the category under evaluation (e.g. *person*), while background corresponds to the union of all the remaining categories (i.e., *non-person*). Then, we estimate the CDFs $\tilde{F}_f \approx F(c_f)$ and $\tilde{F}_b \approx F(c_b)$ of the scores $c_f$ and $c_b$ computed by the CNN for the pixels predicted within foreground $\mathcal{F}$ and background $\mathcal{B}$, respectively. To that end, we use a normal kernel function that is evaluated at equally-spaced points over the range $[0, 1]$ of normalized scores predicted for each region.


Figure 3.10: Example of non-parametric estimation of the probability $P(I_H)$ that a pixel is part of the region labeled with high-confidence. *Left:* original segmentation collected from a CNN. *Center:* pixel scores predicted by the CNN for the *person* category. *Right:* $\tilde{F}_b, \tilde{F}_f$ are the cumulative density functions of scores for non-person (i.e., "background") and person (foreground) pixels, respectively.

#### 3.2.1.2 Similarity measurement

Once in possession of high-confidence seeds, pRGR proceeds to grow these initial pixels into clusters based on spatial and color similarity. Again, let each pixel $p_j$ be described by a 5D feature vector $\mathbf{z_j} = [\mathbf{x_j}, \mathbf{c_j}]^T$, where $\mathbf{x_j} = [x_j, y_j]^T$ are its 2D spatial features and $\mathbf{c_j} = [l_j, a_j, b_j]^T$ its 3D color (CIELab) features. Similarly, let $\mathbf{x_k}, \mathbf{c_k}$ represent the features of the centroid of a cluster $\psi_k$. Then, following

the formulation in [1] (which is based on the SLIC superpixel algorithm [60]), the similarity between $p_j$ and a cluster $\psi_k$ is given by

$$d\left(p_j, \psi_k\right) = \frac{\|\mathbf{x_j} - \mathbf{x_k}\|_2^2}{\sigma_s} + \frac{\|\mathbf{c_j} - \mathbf{c_k}\|_2^2}{\sigma_m}, \tag{3.12}$$

where we replace the $\theta_s, \theta_m$ parameters in the original notation by $\sigma_s, \sigma_m$, respectively, to adjust to our following notation based on variances.

Equation 3.12 can be generalized to

$$\begin{aligned} d\left(p_j, \psi_k\right) &= \frac{1}{\sigma_s}(\mathbf{x_j} - \mathbf{x_k})^T(\mathbf{x_j} - \mathbf{x_k}) + \frac{1}{\sigma_m}(\mathbf{c_j} - \mathbf{c_k})^T(\mathbf{c_j} - \mathbf{c_k}) \\ &= (\mathbf{x_j} - \mathbf{x_k})^T \mathbf{\Sigma_x}^{-1}(\mathbf{x_j} - \mathbf{x_k}) + (\mathbf{c_j} - \mathbf{c_k})^T \mathbf{\Sigma_c}^{-1}(\mathbf{c_j} - \mathbf{c_k}), \end{aligned} \tag{3.13}$$

where $\mathbf{\Sigma_x} = \sigma_s \mathbf{I_2}$, $\mathbf{\Sigma_c} = \sigma_m \mathbf{I_3}$, and $\mathbf{I_k}$ is an identity matrix of size $k$. Furthermore, let $\mathbf{z_k} = [\mathbf{x_k}, \mathbf{c_k}]^T$ and

$$\mathbf{\Sigma_k} = \begin{bmatrix} \mathbf{\Sigma_x} & \mathbf{0_{2\times 3}} \\ \mathbf{0_{3\times 2}} & \mathbf{\Sigma_c} \end{bmatrix}, \tag{3.14}$$

where $\mathbf{0_{u\times v}}$ is an $u \times v$ zero matrix. Then, Eq. 3.13 becomes

$$d\left(p_j, \psi_k\right) = (\mathbf{z_j} - \mathbf{z_k})^T \mathbf{\Sigma_k}^{-1}(\mathbf{z_j} - \mathbf{z_k}). \tag{3.15}$$

We assume that for each partition $\pi$, *each pixel $p_j$ with features $\mathbf{z_j}$ is best described by one and only one cluster $\psi_k$ which is normally distributed with a mean (centroid) $\mathbf{z_k}$ and covariance $\mathbf{\Sigma_k}$*. The distribution of $\mathbf{z_j}$ is therefore given by

$$P(\mathbf{z_j}|\mathbf{z_k}, \mathbf{\Sigma_k}) = \frac{1}{2\pi^{5/2}|\mathbf{\Sigma_k}|^{1/2}} e^{\frac{-1}{2}(\mathbf{z_j}-\mathbf{z_k})^T \mathbf{\Sigma_k}^{-1}(\mathbf{z_j}-\mathbf{z_k})}. \tag{3.16}$$

The corresponding log-likelihood $l(\mathbf{z_j}|\mathbf{z_k})$ is then given by

$$\begin{aligned} l(\mathbf{z_j}|\mathbf{z_k}) &= -\frac{1}{2}(\mathbf{z_j} - \mathbf{z_k})^T \mathbf{\Sigma_k}^{-1}(\mathbf{z_j} - \mathbf{z_k}) - \ln\left(2\pi^{5/2}|\mathbf{\Sigma_k}|^{1/2}\right) \\ &= -\frac{1}{2}d(\mathbf{z_j}, \mathbf{z_k}) - \alpha, \end{aligned} \tag{3.17}$$

where $d(\mathbf{z_j}, \mathbf{z_k}) = (\mathbf{z_j} - \mathbf{z_k})^T \mathbf{\Sigma_k}^{-1}(\mathbf{z_j} - \mathbf{z_k})$ and $\alpha = \ln\left(2\pi^{5/2}|\mathbf{\Sigma_k}|^{1/2}\right)$. Hence, with $\mathbf{z_j} \sim \mathcal{N}(\mathbf{z_k}, \mathbf{\Sigma_k}^{-1})$, the distance in Eq. 3.15 is equivalent to the log-likelihood of

the point $\mathbf{z_j}$ (without the constant offset corresponding to the normalization factor). Therefore, *minimizing the distance* $d\left(p_j, \psi_k\right)$ *is equivalent to maximizing* $l(\mathbf{z_j}|\mathbf{z_k})$.

#### 3.2.1.3 Cluster assignment probability for growing

The probability that a pixel $p_j$ is assigned to a cluster $\psi_i$ is then given by

$$P(p_j \in \psi_i|S) = P\left(d(\mathbf{z_j}, \bar{\mathbf{z}}_\mathbf{i}) = \min_{\psi_k \in \pi} d(\mathbf{z_j}, \bar{\mathbf{z}}_\mathbf{k})\right), \tag{3.18}$$

where $\bar{\mathbf{z}}_\mathbf{k} = \mathbb{E}\left[\mathbf{z}|\psi_k\right]$ is the expected value of $\mathbf{z}$ within a cluster $\psi_k$. That is, the probability that a pixel $p_j$ is assigned to cluster $\psi_i$ is given by the probability that the distance between $\mathbf{z_j}$ and the centroid $\bar{\mathbf{z}}_\mathbf{i}$ is the minimum distance among all the clusters centroids $\bar{\mathbf{z}}_\mathbf{k}$. Since $d(\mathbf{z_j}, \bar{\mathbf{z}}_\mathbf{i})$ follows a chi-squared distribution with $n$ degrees of freedom, where $n$ is the dimensionality of $\mathbf{z}$, the cluster assignment probability is the probability that the sample $d(\mathbf{z_j}, \bar{\mathbf{z}}_\mathbf{i}) \sim \chi^2_n$ is the minimum among the i.i.d. samples $d(\mathbf{z_j}, \bar{\mathbf{z}}_\mathbf{k}) \sim \chi^2_n, \forall \psi_k \in \pi$.

The distribution of the minimum over $\eta$ samples of a distribution with CDF $F(\cdot)$ is given by

$$F_{(1)}(x) = 1 - (1 - F(x))^\eta. \tag{3.19}$$

For $x \sim \chi^2_n$,

$$F(x) = \frac{\gamma(n/2, x/2)}{\Gamma(n/2)}, \tag{3.20}$$

where $\Gamma(\cdot)$ is the gamma function and $\gamma(\cdot, \cdot)$ is the lower incomplete gamma function. Equation 3.19 then becomes

$$F_{(1)}(x) = 1 - \left(1 - \frac{\gamma(n/2, x/2)}{\Gamma(n/2)}\right)^\eta. \tag{3.21}$$

With $x = d(\mathbf{z_j}, \bar{\mathbf{z}}_\mathbf{i})$ and $n = 5$, for our scenario $F_{(1)}(x)$ thus corresponds to the probability that another cluster is closer than $\psi_i$ to the pixel $p_j$. Hence, it follows

that

$$P(p_j \in \psi_i|S) = 1 - F_{(1)}(d(\mathbf{z_j}, \bar{\mathbf{z}}_\mathbf{i}))$$
$$P(p_j \in \psi_i|S) = \left(1 - \frac{\gamma(2.5, d(\mathbf{z_j}, \bar{\mathbf{z}}_\mathbf{i})/2)}{1.33}\right)^\eta, \tag{3.22}$$

which is thus the equation that guides pixel-cluster assignments for the region growing process.

#### 3.2.1.4 Pixel probability estimation

Given the set of clusters $\pi^{(m)} = \left\{\psi_1^{(m)}, \psi_2^{(m)}, \ldots, \psi_{|S|}^{(m)}\right\}$ generated at the $m$-th iteration of the algorithm, the expected class likelihood $\bar{c}_i^{(m)}$ value within each cluster $\psi_i^{(m)}$ is estimated as the average of the scores $c_j$ associated to its pixels $p_j \in \psi_i^{(m)}$, weighted according to the probability $P(p_j \in \psi_i^{(m)}|S^{(m)})$ of pixel-cluster assignment. That is,

$$\begin{aligned}\bar{c}_i^{(m)} &= P\left(\psi_i^{(m)} \in \mathcal{F}|S^{(m)}\right) \\ &= \frac{\sum_{p_j \in \psi_i^{(m)}} c_j P(p_j \in \psi_i^{(m)}|S^{(m)})}{\sum_{p_j \in \psi_i^{(m)}} P(p_j \in \psi_i|S)^{(m)}}.\end{aligned} \tag{3.23}$$

Then, $\bar{c}_i{}^{(m)}$ is the refined class probability for all pixels $p_j \in \psi_i^{(m)}$, i.e.,

$$\bar{c}_j^{(m)} = P(p_j \in \mathcal{F}|\pi^{(m)}) = \bar{c}_i^{(m)}. \tag{3.24}$$

In cases where no seed is sufficiently similar to a given pixel, the probabilities of assigning this pixel to any cluster will be low and the growing process will end without any assignment for this pixel. We refer to these elements as *orphan pixels*. In iterations where a pixel $p_o$ remains orphan, i.e., $p_o \notin \psi_i^{(m)}, \forall \psi_i \in \pi^{(m)}$, we keep its originally predicted score $c_o$ as $\bar{c}_o^{(m)} = P(p_o \in \mathcal{F}|\pi^{(m)})$.

Let $\Pi = \{\pi^{(1)}, ..., \pi^{(n_s)}\}$ represent the set of all partitions generated by the multiple Monte Carlo iterations. With enough iterations, we can approximate the

distribution

$$P(p_j \in \mathcal{F}|\pi) \approx \sum_{\pi^{(m)} \in \Pi} P(p_j \in \mathcal{F}|\pi^{(m)})\delta_\pi(\Pi), \tag{3.25}$$

where $\delta_\pi(\Pi)$ is the Dirac delta function, which is equal to one if $\pi \in \Pi$ and zero otherwise. Marginalizing over the set of partitions $\Pi$, we have

$$\begin{aligned} P(p_j \in \mathcal{F}) &= \int_\Pi P(p_j \in \mathcal{F}|\pi)P(\pi) \\ &\approx \frac{1}{n_s}\sum_m P(p_j \in \mathcal{F}|\pi^{(m)}) = \frac{1}{n_s}\sum_m \bar{c_j}^{(m)}, \end{aligned} \tag{3.26}$$

such that the final refined class probability for each pixel $p_j$ is given by $\tilde{c_j} = P(p_j \in \mathcal{F})$.

**Variance estimation**

In addition to the average computed in Eq. 3.26, it is also possible to compute for each pixel the variance of the estimations provided by the multiple Monte Carlo iterations. Analogously to the computation of the average $\tilde{c_j}$, the variance $\tilde{\sigma}_j^2$ across partitions can be computed as

$$\tilde{\sigma}_j^2 = Var\left[P\left(p_j \in \mathcal{F}|\Pi\right)\right] = \frac{1}{n_s}\sum_m \left(\bar{c_j}^{(m)} - \tilde{c_j}\right)^2. \tag{3.27}$$

As demonstrated in Section 3.2.3, the variance can be exploited as a measure of uncertainty that is highly correlated with segmentation accuracy. In practice, we observe that for significantly coarse predictions, it is advantageous to run the overall pRGR algorithm more than once to further improve the quality of segmentation. Let $r$ denote the ordinal index for each complete run in a set of runs $R = \{1, ..., |R|\}$. Then, including the index $r$ in Eq. 3.26, each run provides an estimate $\tilde{c_j}^{(r)} = P(p_j \in \mathcal{F}|\Pi^{(r)})$ for a pixel $p_j$. To obtain a final estimation $P(p_j \in \mathcal{F})$, we exploit inverse variance weighting to combine the estimations provided by each run. That is,

$$P(p_j \in \mathcal{F}) = \frac{\sum_{r \in R} \tilde{c_j}^{(r)} / \tilde{\sigma}_j^{2(r)}}{\sum_{r \in R} 1 / \tilde{\sigma}_j^{2(r)}}. \tag{3.28}$$

#### 3.2.1.5 Initialization and update of cluster statistics

As mentioned above, we assume clusters are normally distributed according to $\mathcal{N}(\mathbf{z_k}, \mathbf{\Sigma_k})$, which implies a normally distributed likelihood function. Moreover, to allow for flexible clusters that adapt to local image and prediction characteristics, similar to [64, 65], we update the terms in the spatial and color covariances in Eq. 3.14 separately, i.e.,

$$\mathbf{\Sigma_x} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}, \quad \mathbf{\Sigma_c} = \begin{bmatrix} \sigma_l^2 & 0 & 0 \\ 0 & \sigma_a^2 & 0 \\ 0 & 0 & \sigma_b^2 \end{bmatrix}, \tag{3.29}$$

where $\sigma_x^2, \sigma_y^2$ are the variances along the horizontal and vertical coordinates, $\sigma_l^2$ is the variance of the $L$ color channel and $\sigma_a^2, \sigma_b^2$ are the variances for the $a$ and $b$ channels, respectively.

**Initialization**

To ensure normally distributed posteriors and facilitate the update process, we initialize the mean $\mathbf{z_k}$ and covariances $\mathbf{\Sigma_k}$ of each cluster using conjugate prior distributions [107, 118]. Since the spatial and color variances are assumed to be independent, we can define Normal-inverse-chi-squared (NI$\chi^2$) prior distributions of the form

$$\begin{aligned} \mu|\sigma^2 &\sim \mathcal{N}\left(\mu_0, {}^{\sigma^2}/_{\kappa_0}\right) \\ \sigma^2 &\sim Inv - \chi^2\left(v_0, \sigma_0^2\right), \end{aligned} \tag{3.30}$$

where $\mu$ and $\sigma^2$ are the means and variances for each of the five dimensions of $(\mathbf{z_k}, \mathbf{\Sigma_k})$, with the subscripts dropped for simplicity. The means $\mu_0$ of the normal distributions are initialized according to the locations and colors of the corresponding seeds, while $\kappa_0$ is fixed as 1 as a seed is worth one observation of variance $\sigma^2$.

*Spatial variances.* Initializing the inverse-chi-squared parameters $(v_o, \sigma_0^2)$ associated to the variances is more complex. Under the assumption of normally distributed clusters, the expected size of a cluster is directly proportional to the expected values of its spatial variances. Since the inter-seed spacing is known in the form of the sampled parameter $\gamma$, we expect the average cluster sizes to be proportional to $\gamma \times \gamma$. Thus, the spatial variances can be initialized as

$$\sigma_{0,x}^2 = \sigma_{0,y}^2 = \lambda(\gamma \times \gamma), \tag{3.31}$$

where $\lambda$ is an empirically defined proportionality constant. To allow clusters to grow larger and reach lower confidence areas without nearby seeds, based on a grid search performed on a subset of 350 randomly sampled images from the PASCAL dataset, we use the fixed value of $\lambda = 27$ in all our experiments, regardless of the CNN model used to generate the segmentation masks or the dataset under consideration.

As described in [118], the $v_0$ parameters give a sense of how many observations the corresponding prior knowledge is worth. Based on this intuition, we exploit again the fact that average expected cluster sizes are proportional to $\gamma \times \gamma$, such that $v_0 \propto \gamma^2$. Moreover, we note that the reliability of sample variance estimations is directly proportional to the quality of the corresponding initial seed, since it defines the initial mean values. Hence, it follows that in the case of lower quality seeds more weight must be given to the prior with respect to subsequent sample variance estimates. Combining both characteristics,

$$v_{0,x} = v_{0,y} \propto \left[\frac{\gamma}{P(s_k \in I_H)}\right]^2, \tag{3.32}$$

where $P(s_k \in I_H)$ corresponds to the probability that a seed is within the high-confidence region, obtained from Eq. 3.10.

*Color variances.* Determining an expected cluster color variance is not as straightforward. Hence, we first examined the color statistics of clusters formed using a conventional superpixel algorithm (SLIC [60]) on the same subset of 350 images from the PASCAL dataset. Multiple runs with a varying number of superpixels and compactness values indicated variances of approximately $\sigma_l^2 = 850$ and $\sigma_a^2 = \sigma_b^2 = 260$ to cover 99% of the samples within the superpixels. Based on these observed values, we then conducted a grid-search that led to the optimal initialization values of $\sigma_{0,l}^2 = 1000$ and $\sigma_{0,a}^2 = \sigma_{0,b}^2 = 300$, which are used in all our experiments.

Since the distribution of color similarities can change from image to image, we employ an antithetic sampling variance reduction strategy [108] in which initial color variance values are multiplied by a value $1 \pm \rho$. A value of $\rho = 0.6$ was defined for all experiments after a grid search over the interval $[0.1, 0.9]$ with a resolution of 0.1, using the same PASCAL subset described above. That is, we initialize $\sigma_{0,l}^2 = 1000 \times [1 \pm \rho]$ and $\sigma_{0,a}^2 = \sigma_{0,b}^2 = 300 \times [1 \pm \rho]$. The equivalent sample size $v_{0,\{lab\}}$ for the color variances is computed using the same approach as the one used for the spatial variances, which is given by Eq. 3.32.

Finally, as explained in Section 3.2.2, in the region growing process all clusters grow from the center outwards, as the first pixels assigned are the corresponding seed neighbors, with subsequent tentative assignments of pixels neighboring the ones just assigned. In terms of sample statistics, this means initial spatial sample variances are heavily biased towards smaller values, as the first pixels assigned are the ones nearest to the corresponding cluster's centroid. To compensate for this bias, we increase the $v_0$ weight of prior variance knowledge by multiplying it with a constant, i.e., for all the experiments we set $v_0 = \alpha \left[\gamma / P(s_k \in I_H)\right]^2$. We use $\alpha = 5$ for the spatial variances and, since this bias is much lower for the color statistics, we empirically set $\alpha = 0.1$ for the color variances.

**Updates**

As detailed in [118,119], from the combination of a NI$\chi^2$ prior with the corresponding normal likelihood, the parameters of the corresponding posteriors are then given by

$$v_n = v_0 + n; \qquad \kappa_n = \kappa_0 + n; \qquad \mu_n = \frac{\kappa_0 \mu_0 + n\bar{x}}{\kappa_n}; \tag{3.33}$$

$$\sigma_n^2 = \frac{1}{v_n}\left[v_0\sigma_0^2 + \sum_i (x_i - \bar{x})^2 + \frac{n\kappa_0}{\kappa_0 + n}(\mu_0 - \bar{x})^2\right], \tag{3.34}$$

where $\bar{x}$ denotes the sample mean and $n$ is the total number of samples, which corresponds to the cluster size, i.e., $n = |\psi_k|$. If sample sizes are not large enough, eventual biases in the estimation of the sample variance may arise, leading to clusters with incorrect sizes. We thus apply an update strategy in which sample variance estimations are computed only after the expected cluster sizes are reached, i.e., $|\psi_k| \geq [\gamma/_{P(s_k \in I_H)}]^2$.

**Posterior**

To compute the distances and the corresponding probabilities of assigning pixels to clusters, the posterior predictive distribution is given by a Student's $t$ distribution with $v_n$ degrees freedom. Since for the vast majority of iterations $v_0 \geq 30$, this posterior can be approximated as normally distributed according to $\mathcal{N}(\mu_n, \sigma_n)$.

### 3.2.2 Algorithm implementation

We implement pRGR by means of a main function that invokes Algorithm 2, which summarizes the proposed region growing process that assigns pixels to clusters. Figure 3.11 provides an illustration of the steps performed by the overall refinement algorithm. First, the main script performs the non-parametric estimation of thresholds distributions and subsequent computation of seed sampling probabilities. This

script then samples an initial set of seeds $S$ and invokes Algorithm 2 for region growing.

From the image features $\mathcal{Z}$ and the corresponding set of seeds $S$ as inputs, Algorithm 2 returns an array $L$ where each pixel is mapped to its corresponding cluster by means of an index.

**Algorithm 2** Proposed cluster assignment algorithm.

**Input:** $\mathcal{Z} = \{z_1, ..., z_{w \times h}\}$: set of 5D pixels;
$\mathcal{S}$ : set of seeds
**Define:** $L$: cluster label assigned to each pixel
$T$: no. of times a pixel has been sampled

```
for all j = {1, ..., w × h} do
    Initialize T[j] = 0 and L[j] = ∅
for all s_k ∈ S do
    Get pixel p_j at s_k position (x_k, y_k)
    Push element e_j = [j, k, 1] into priority queue Q1
while (Q1 ≠ ∅) or (Q2 ≠ ∅) do
    Pop e_j = [j, k, P_jk] from Q1
    if (T[j] < κ) and (L[j] = ∅) then
        Draw u ~ U[0, 1]
        Increment counter T[j] ← T[j] + 1
        if (u < P_jk) then
            Assign p_j to cluster ψ_k: L[j] = k
            Update cluster statistics with Eqs. 3.33 and 3.34
            for all p_n 8-connected to p_j do
                if L[n] = ∅ then
                    Compute P_nk = P(p_n ∈ ψ_k|S) Eq. 3.22
                    Push e_n = [n, k, P_nk] into Q1
        else if T[j] < η then
            Push e_j into recycling queue Q2
    if Q1 = ∅ then
        while Q2 ≠ ∅ do
            Pop element e_r = [r, k, P_rk] from Q2
            Recompute P̂_rk = P(p_r ∈ ψ_k|S) Eq. 3.22
            Push element ê_r = [r, k, P̂_rk] into Q1
return L
```

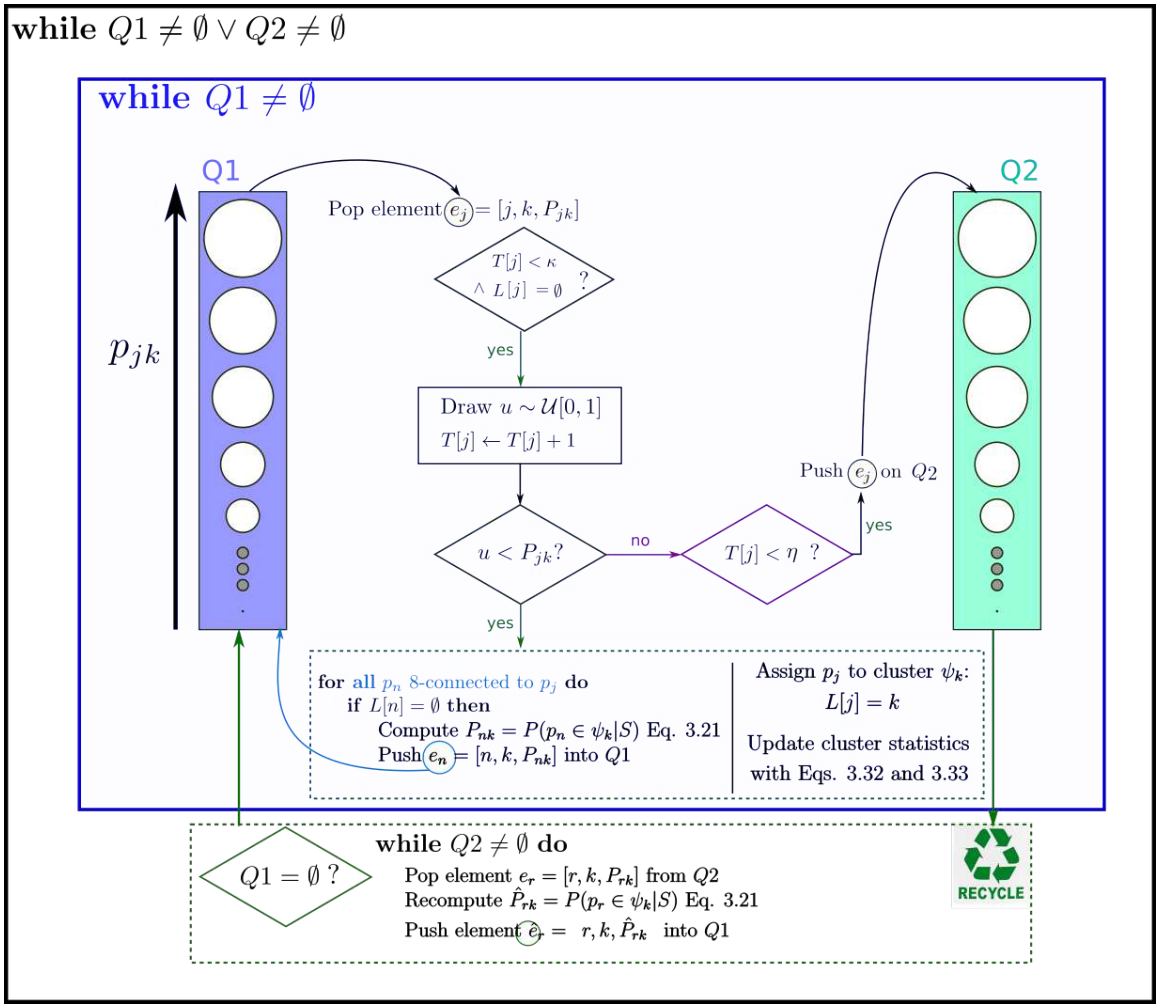



Figure 3.11: Diagram illustrating the steps performed by Algorithm 2.

Let an element $e_j = [j, k, P_{jk}]$ represent a tentative assignment of a pixel $p_j$ to a cluster $\psi_k$, with the corresponding probability $P_{jk} = P(p_j \in \psi_k | S)$ given by Eq. 3.22. For pixels sampled as seeds, elements are created with $P_{jk}$ set to 1.0. Inspired by the SNIC [117] implementation, such tentative assignment elements are pushed into a priority queue $Q1$ that is sorted in descending order according to the assignment probabilities $P_{jk}$. Assignments occur by popping elements from $Q1$ and sampling according to the corresponding probability. Starting from the corresponding seeds, when a pixel $p_j$ is effectively assigned to a cluster $\psi_k$, all its $p_n$ 8-connected

neighbors are evaluated: if they have not been clustered yet, elements $e_n = [n, k, P_{nk}]$ are pushed into $Q1$ as tentative assignments of these pixels to their now neighboring cluster $\psi_k$.

With such 8-connectivity enforcement during growing, we ensure that a pixel is visited (sampled) a maximum of 8 times. However, since that is only an upper-bound, we opt for an implementation that ensures that each pixel will be visited at least 8 times before being considered orphan. This is achieved through a recycling process using a recycle queue $Q2$. When an element is popped from $Q1$ but assignment does not occur, this element is pushed into a recycling queue $Q2$ if the corresponding pixel has been sampled less than 8 times. Whenever $Q1$ is emptied, all elements in $Q2$ are updated according to the latest clusters' statistics and re-pushed into $Q1$ for processing. Using this strategy, we ensure a fixed $\eta = 8$ to be used in Eq. 3.22.

Therefore, the algorithm converges once all pixels have either been assigned to a cluster or visited a maximum of 8 times. Once in possession of the corresponding mappings of pixels to clusters returned by Algorithm 2, the main function proceeds to compute pixel probability estimations according to Eqs. 3.23-3.28.

### Gaussian filtering

Since we must approximate the posterior distribution using a finite number of Monte Carlo iterations, pixels with high uncertainty might require additional refinement steps to produce accurate results. To avoid performing a large number of iterations that would impact a relatively small number of pixels, we smooth out spuriuous pixel activations using a $3 \times 3$ convolution with a Gaussian kernel on top of the refined scoremaps obtained using Eq. 3.26.

### 3.2.3 Experiments

We evaluate the performance of pRGR on: i) the 1449 images composing the *val* set of the PASCAL VOC 2012 dataset [43]; and ii) selected video sequences of the DAVIS dataset [7, 48]. While the PASCAL dataset is arguably the most widely used benchmark for semantic segmentation, its evaluation metrics disregard regions 5 pixels-wide around the boundaries of each object. As a consequence, often clear improvements in terms of boundary adherence are not reflected in overall mean intersection over union ($mIoU$). For that reason, we also include results using the DAVIS dataset [7], which is composed of high-quality video sequences with pixel-accurate ground truth segmentation for each frame.

#### Baselines

We compare pRGR with its predecessor RGR and also against DenseCRF [104], one of the most widely used post-processing module for semantic segmentation. For conciseness, we refer to the DenseCRF model as CRF in the paragraphs that follow. We also evaluate the combination of CRF+pRGR, in which our refinement algorithm is run on top of the predictions refined using CRF.

#### Networks

To assess our method for input predictions of varied quality, four different pre-trained, publicly available semantic segmentation models are considered. First, the DeepLab-COCO-LargeFOV (here DeepLab-LargeFOV for conciseness) model [88], a DeepLab model using large Field-Of-View that was also used for the evaluation of RGR in previous sections. We also evaluate the refinement of predictions generated by two DeepLabV2 models [22], one using a VGG [120] backbone and another using a

ResNet backbone [46]. Finally, we assess a DeepLabV3+ model [98] using an Xception backbone [121].[1]

As summarized in Section 2.4.5, these models represent different stages of recent developments in the state of the art of semantic segmentation. From their architectures, finer segmentations are expected as one moves from DeepLab to DeepLabV2 and finally DeepLabV3, both in terms of overall accuracy as well as boundary adherence. The datasets with which these models were trained also play an important role in their performance. We note that in terms of pre-training, the DeepLab-LargeFOV model exploits annotations from the MS-COCO dataset, the *trainaug* subset of PASCAL VOC 2012 and, unlike the others, the *val* set of PASCAL VOC 2012 in which our evaluations are performed. In contrast, both the DeepLabV2 and DeepLabV3+ models we use in our evaluation are trained only on the *trainaug* subset of VOC. Of the four models, only the DeepLabV2 (VGG) is not pre-trained on COCO.

**Parameterizations**

Since CRF relies on a grid-search of its hyperparameters for optimal performance, as detailed below we opted for publicly available models for which the optimal CRF configurations are also provided. Regarding RGR, for all experiments, parameterization is done as reported in [1] and Section 3.2.3, where a different high-confidence foreground threshold value $t_f$ is sampled from the distribution $U(0.5, 0.9)$ in each region growing iteration.

For all the cases described above, pRGR is configured to perform 20 Monte Carlo iterations for each class scoremap. A total of 10 different values of the seed-spacing parameter $\gamma$ are sampled from the range $[2, \gamma_h]$, using systematic stratified sampling. For each $\gamma$, two iterations with antithetic color configurations are run

[1]The first three models are available at `http://liangchiehchen.com/projects/DeepLab_Models.html`. The DeepLabV3+ model can be found at `https://github.com/tensorflow/models/tree/master/research/deeplab`

with $\rho = 0.6$ as explained in Section 3.2.1.5. According to their output strides, the different networks under consideration require distinct levels of refinement in terms of receptive field sizes. For pRGR, this corresponds to varying the upper-limit $\gamma_h$, as it defines the maximum expected cluster sizes. Hence, $\gamma_h$ is the only parameter of pRGR that is empirically adjusted on a case-by-case basis. The values selected for our experiments are listed in Table 3.3. For all the experiments with CRF+pRGR, $\gamma_h$ is set to 16.

Table 3.3: Summary of pRGR configurations for each network

| | **DeepLab version** | | | |
|---|---|---|---|---|
| | **LargeFOV** | **V2 (VGG)** | **V2 (ResNet)** | **V3+** |
| *Double refinement* | ✓ | ✓ | | |
| $\gamma_h$ | 48 | 32 | 24 | 16 |

As illustrated in Figure 3.12, the segmentations provided by DeepLab-LargeFOV and DeepLabV2 (VGG) are fairly coarse, such that for these cases we perform two pRGR refinement steps using inverse variance weighing to combine the estimation results of each step, as explained in Eq. 3.28.

#### 3.2.3.1 Comparison with baselines on PASCAL

Table 3.4 summarizes the quantitative results provided by each combination of refinement methods with the corresponding four variations of semantic segmentation networks. Since boundaries constitute a small fraction of the total image pixels, to better quantify boundary adherence, we follow the strategy presented in [52] and also evaluate segmentation accuracy on narrower regions closer to the boundaries. Figure 3.12 presents qualitative examples of segmentation masks provided by each combina-

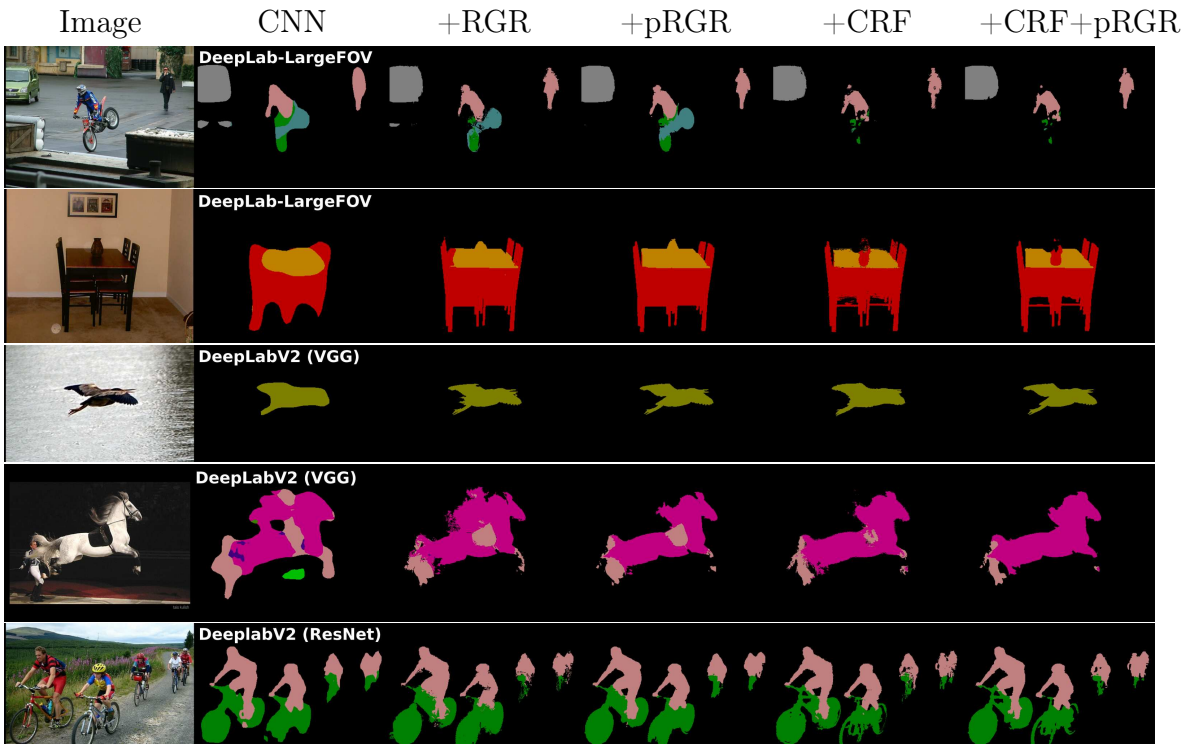



Figure 3.12: Qualitative results on PASCAL *val* images. In the second column, overlaid names correspond to the CNNs used for each prediction.

tion of methods, while Figure 3.13 shows the $mIoU$ values obtained by each method as a function of the object boundary width considered in the evaluation. Finally, Figure 3.14 details the performances of each method according to each category of the PASCAL dataset.

Table 3.4: Comparison and combination of pRGR and baselines on PASCAL dataset.

| | **VOC 2012 - mIoU(%)** | | | | |
|---|---|---|---|---|---|
| **CNN** | *No ref.* | *+CRF* | *+RGR* | *+pRGR* | *+CRF+pRGR* |
| DeepLab-LargeFOV | 76.05 | 80.23 | 79.21 | 80.11 | 80.58 |
| DeepLabV2 (VGG) | 68.96 | 71.57 | 70.97 | 71.22 | 71.94 |
| DeepLabV2 (ResNet) | 76.46 | 77.65 | 77.39 | 77.54 | 77.86 |

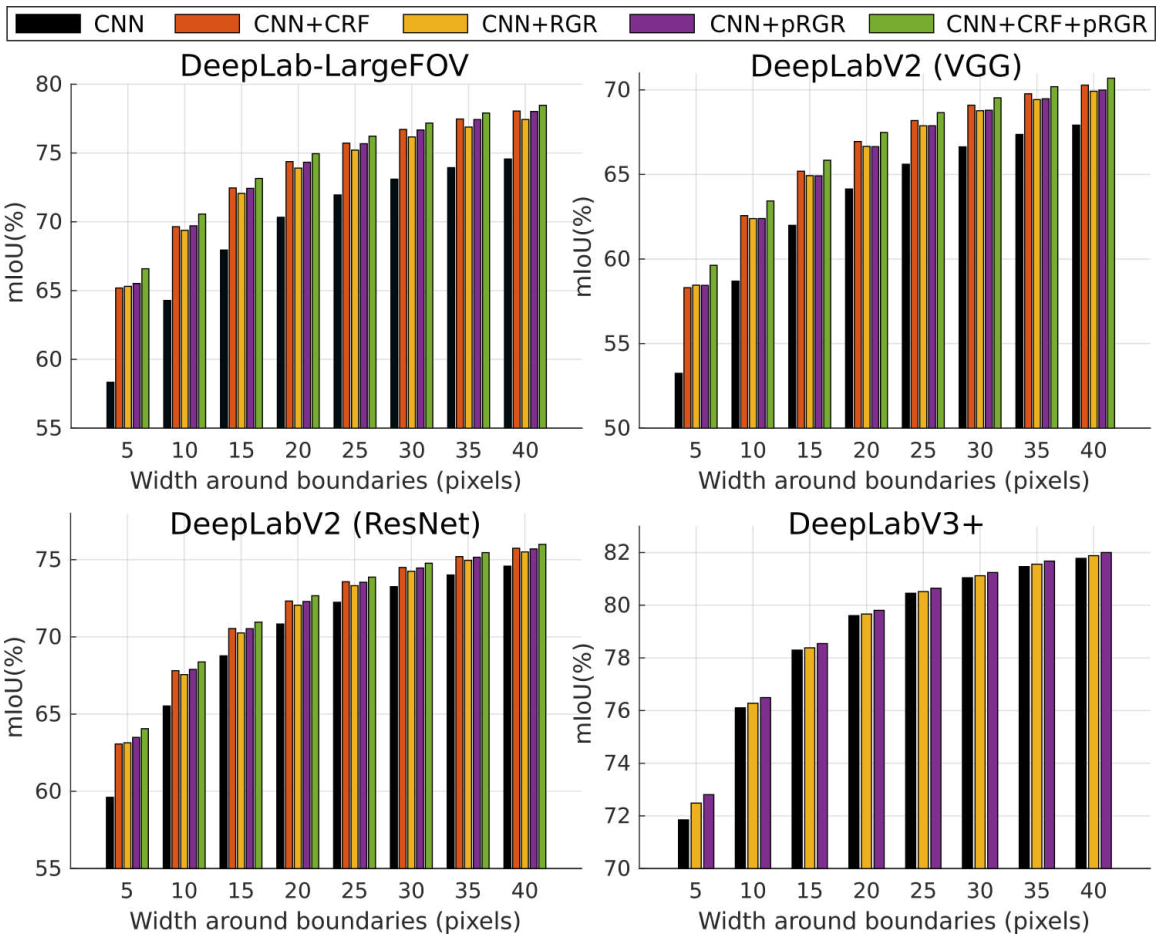



Figure 3.13: Summary of *mIoU* on PASCAL for regions of varying width near the object boundaries. Each color corresponds to a combination of the corresponding CNN with a refinement method, according to the legend above the figures.

### Boundary adherence

The results in Figure 3.13 highlight how all the methods under consideration improve segmentation accuracy especially in regions near boundaries. In comparison with the results shown in Table 3.4, even for scenarios such as DeepLabV2 (ResNet), where overall *mIoU* improvements are slightly above +1.0%, the segmentation ac-

curacy in regions $\leq 5px$ near the boundaries is improved by approximately +3.5% using pRGR.

**RGR vs pRGR**

Overall, our results demonstrate that pRGR consistently outperforms RGR in all the scenarios under consideration. In comparison with its precursor RGR, the probabilistic formulation of pRGR combined with refinement iterations at different receptive field sizes reduces the occurrence of noisy predictions and minimizes the impact of false positives. This is illustrated in Figure 3.12 near the bird's wings and beak, and also near the horse's crest.

**CRF vs pRGR**

In terms of overall accuracy, pRGR provides $mIoU$ values slightly lower than the ones obtained with CRF. However, the results summarized in Figure 3.13 indicate that predictions refined using pRGR are slightly better (FOV: +0.33%, VGG: +0.14%, ResNet: +0.43%) than the ones using CRF for regions $\leq 5px$ near the boundaries. This is also exemplified near the bird's wings in Figure 3.12. On the other hand, results detailed in Figure 3.14 for categories such as bicycles and chairs suggest that the main failure case of pRGR corresponds to enclosed regions with high amounts of false-positives, such as the internal areas of bicycles' wheels and chairs' spindles. Qualitatively, this is illustrated in the last example of Figure 3.12. As the region growing procedure is based on $8-$connectivity, it cannot correct such enclosed regions containing high amounts of false positives. In contrast, CRF is able to recover from such mistakes, which is reflected in the overall higher $mIoU$ values. However, it is important to note again that pRGR is entirely unsupervised, whereas CRF must be fine-tuned to the dataset and segmentation network under consideration.

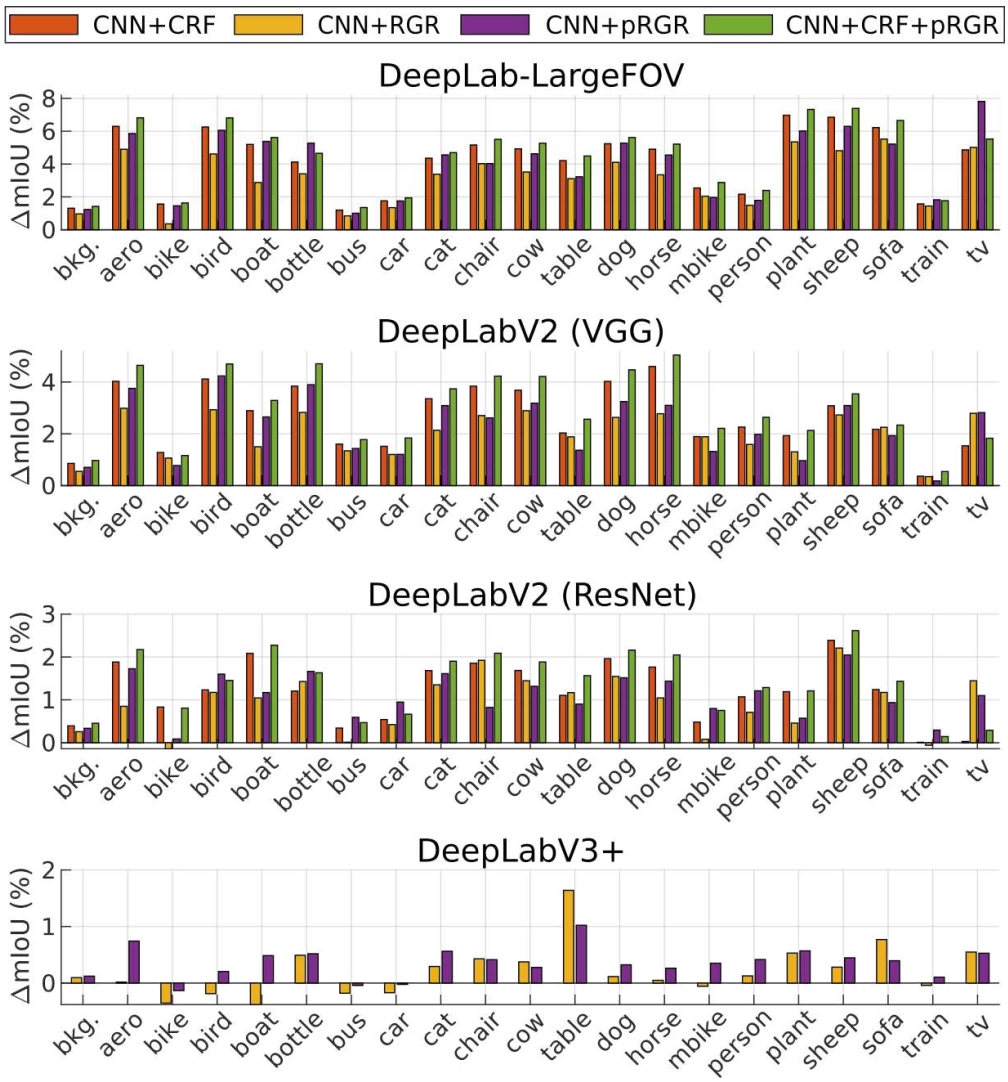



Figure 3.14: Improvements on segmentation accuracy ($\Delta mIoU(\%)$) provided by each refinement method according to specific categories on PASCAL dataset.

### CRF+pRGR

Our analysis suggests that, while CRF and pRGR provide similar overall performances, they have different success/failure cases. As such, combining CRF and pRGR is a potential strategy for further refining segmentation masks, which is corroborated by the results reported as $CRF + pRGR$ in Table 3.4 and Figures 3.12 and 3.13. In all the evaluated scenarios, this combination significantly outperforms

CRF alone, especially in regions near boundaries as shown quantitatively in Figure 3.13 and can be noticed in the chairs' and bird's details in Figure 3.12. Moreover, the fourth example in Figure 3.12 illustrates how pRGR can also mitigate some false positives partially attenuated by CRF, such as misdetections near the saddle and the horse's knee. Finally, results combining $CRF + pRGR$ also demonstrate that, if the amount of false-positives is reduced and enough high-quality seeds are available, pRGR can also improve segmentations in the failure case scenarios.

#### 3.2.3.2 Refinement of DeepLabV3+ predictions

Table 3.5 summarizes the performances of DeepLabV3+ before and after refinement using RGR and pRGR, for experiments on the PASCAL and DAVIS datasets. Unlike the previous experiments, here the CRF baseline is not considered, since no CRF implementation optimized for DeepLabV3+ is currently available.

Table 3.5: Effect of RGR and pRGR for refinement of DeepLabV3+ predictions.

| **CNN** | **Dataset** | | *No ref.* | *+RGR* | *+pRGR* |
|---|---|---|---|---|---|
| DeepLabV3+ | **VOC 2012** | *mIoU(%)* | 82.20 | 82.41 | 82.56 |
| | **DAVIS 2017** | $\mathcal{J}$ *mean* | 76.29 | 80.19 | 80.47 |
| | | $\mathcal{F}$ *mean* | 76.41 | 80.10 | 80.30 |

From Table 3.5 and the results in the lower right corner of Figure 3.13, experiments on the PASCAL dataset using DeepLabV3+ once again indicate that, although the gains in overall $mIoU$ are relatively small ($\approx 0.36\%$), both RGR and pRGR provide non-negligible improvements in terms of boundary adherence even for state-of-the-art semantic segmentation networks ($\approx 1.0\%$ for regions $\leq 5px$ near boundaries).

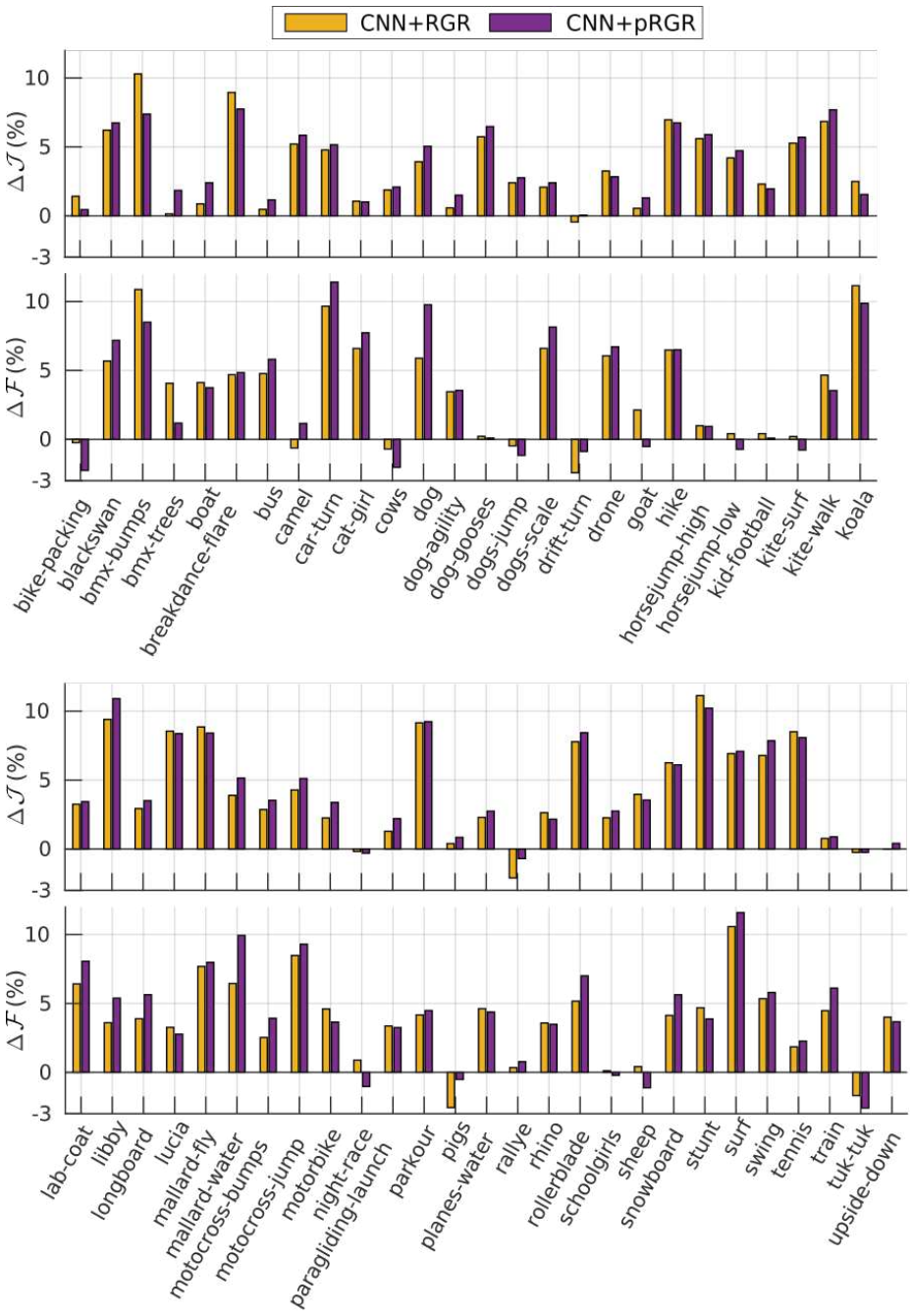



Figure 3.15: Improvements on segmentation accuracy provided by each refinement method according to specific sequences of the DAVIS dataset. *Top:* variations in $\mathcal{J}mean$. *Bottom:* variations in $\mathcal{F}mean$

To further validate this observation, we selected 53 video sequences listed in Figure 3.15 from the DAVIS 2016 [7] and 2017 [48] datasets for further experimen-

tation with the same DeepLabV3+ model. Since this model is trained for the 21 PASCAL categories, we selected only sequences where the target objects are within this set of categories.

As previously mentioned, the DAVIS evaluation metrics include both overall intersection over union (or Jaccard-index) $\mathcal{J}$ and also a contour accuracy metric $\mathcal{F}$ that assesses specifically the accuracy near object boundaries. Table 3.5 contains the results obtained using both metrics for predictions before and after RGR and pRGR refinement. Since the DAVIS annotations take into consideration all the pixels composing object boundaries, in this dataset, improvements in terms of boundary adherence have higher impact in final performance metrics than the ones observed for experiments on the PASCAL dataset. The results reveal improvements in the order of $\approx 4.0\%$ by both refinement methods, with pRGR marginally yet consistently outperforming its predecessor in both metrics.

Results for the $\mathcal{F}$ metric demonstrate that pRGR provides large improvements in terms of boundary adherence, with a 3.9% increase in mean $\mathcal{F}$. Figure 3.16 shows qualitative examples of such improvements. In all the examples, we observe how the refined segmentation masks include fewer pixels comprising the surrounding background. In the first two images, details such as the people's hair and feet are recovered. In the last image, the refined segmentation properly adheres to the dogs' fur, and correctly separates the person from the dog.

From the results for the individual DAVIS sequence detailed in Figure 3.15, lower performances are observed for some sequences containing vehicles and animals as targets. For the first case, failures mostly arise from propagating false-positive detections of shadows under vehicles. In the case of animals, limb extremities can be lost when such elongated structures are detected with low confidence, are far from the animal's body, and share a similar color with the surrounding background. Yet, we emphasize that significant improvements are observed for most scenarios evaluated.

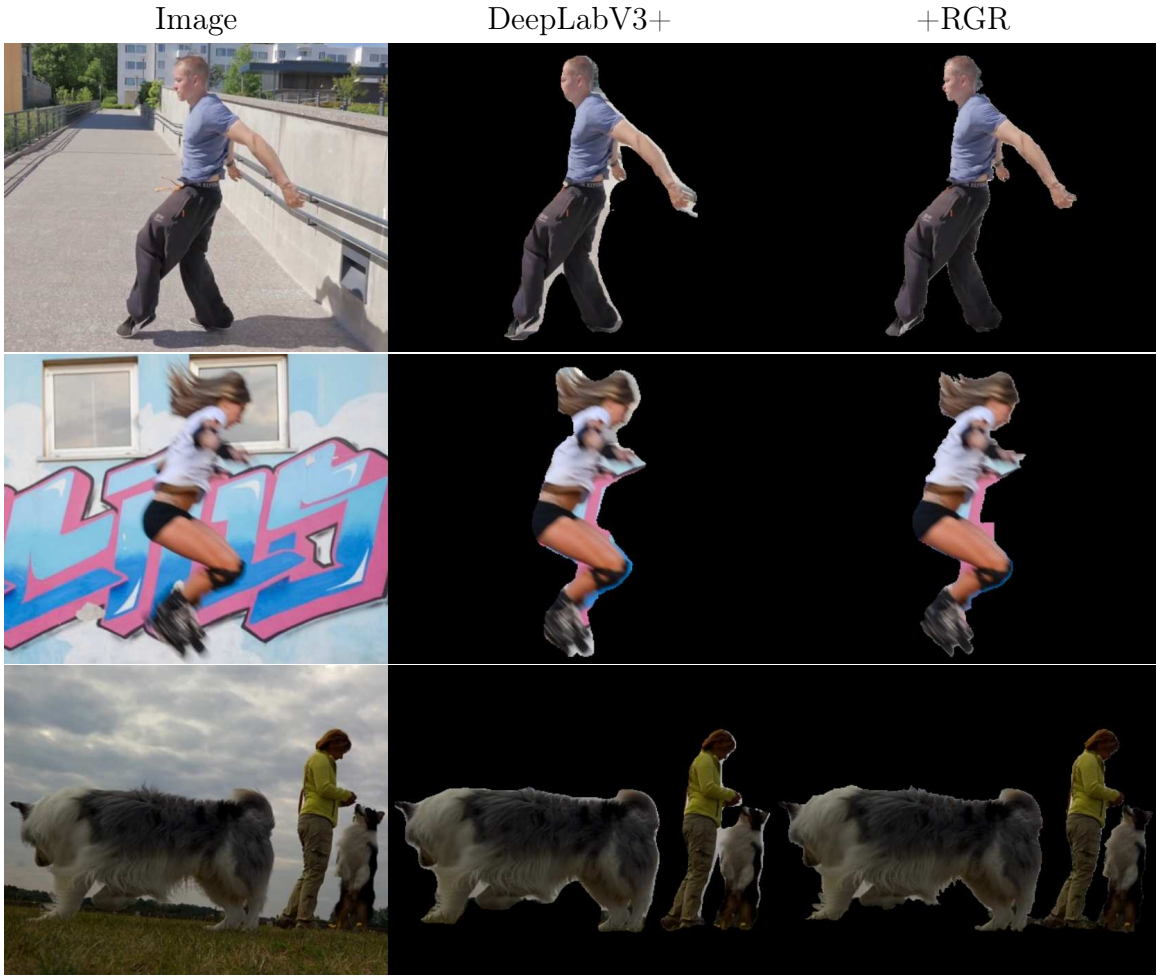



Figure 3.16: Examples of details recovered through pRGR refinement of DeepLabV3+ predictions for images in the DAVIS dataset.

#### 3.2.3.3 Uncertainty estimation

As noted by Kendall & Gal [109], the normalized scores provided by CNNs do not necessarily reflect the uncertainties of these classification models. In [122], Bayesian Deep Learning is exploited using Monte Carlo dropout [123] and Concrete dropout [113] to capture uncertainties of a DeepLabV3+ model for semantic segmentation. In our pRGR framework, the variance of estimations across multiple Monte Carlo refinement iterations (computed using Eq. 3.27) can be exploited as a measure of classification uncertainty. To validate this claim, we evaluated the $mIoU$ values on

the PASCAL dataset for increasingly high thresholds of variance values. Similarly, we establish a baseline for comparison by computing the accuracy of the original network's predictions for increasingly high thresholds of predicted class scores.

Figure 3.17 presents the results collected for experiments using DeepLab-LargeFOV predictions. For both cases, the curves on the top row suggest a significant correlation between prediction scores (for CNN predictions) and estimated variances (from pRGR outputs) with the actual segmentation accuracy.

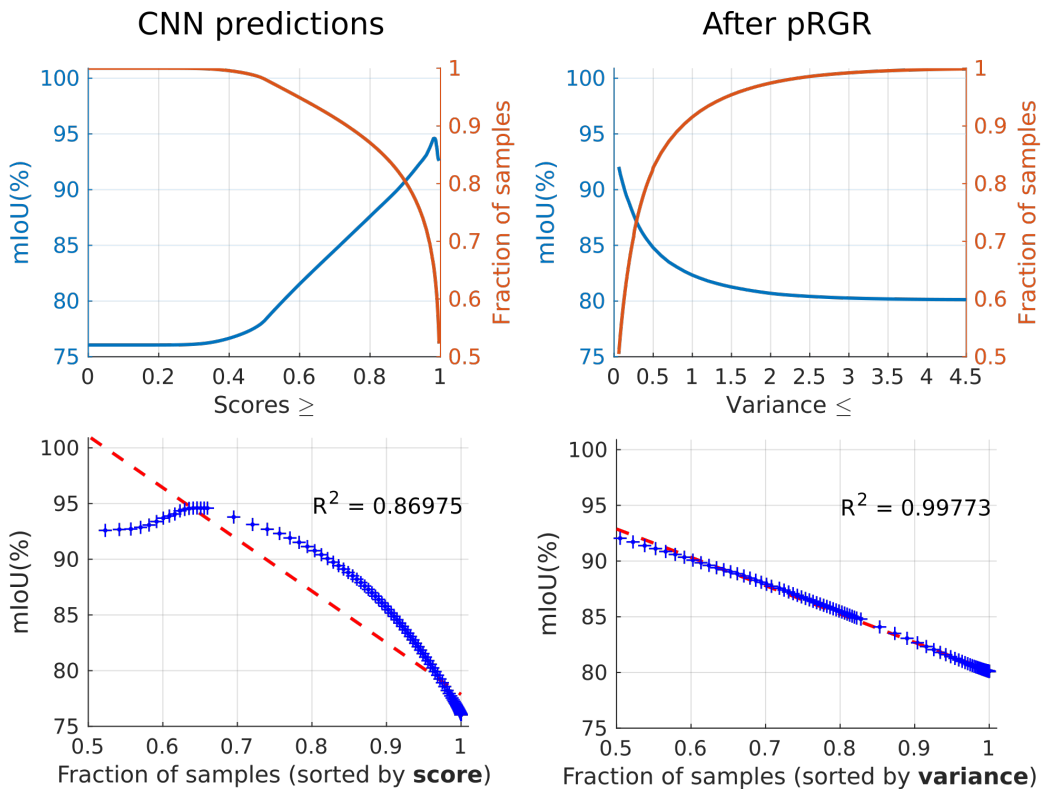
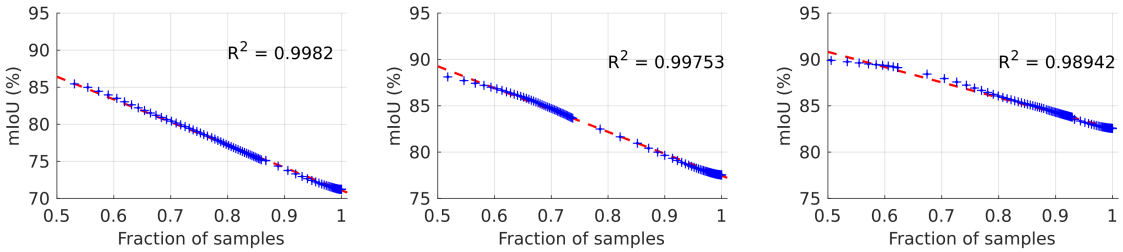



Figure 3.17: Correlation between segmentation accuracy and *left)* original CNN prediction scores; *right)* variance across pRGR Monte Carlo refinement iterations.

However, for the CNN predictions, sharper slope variations are observed both at the beginning and the end of the $mIoU$'s curve. Since for both cases the fraction of

samples covered varies non-linearly as the threshold values increase, we also analyze the accuracy vs. the fraction of samples to assess the correlation between segmentation quality and uncertainty estimations. More specifically, the graphs on the bottom row of Figure 3.17 are obtained by plotting the left y-axis vs. the right y-axis for each corresponding plot from the top row. This analysis corresponds to assessing how segmentation accuracy decays as larger fractions of samples with increasingly high uncertainty are considered.

Ideally, the confidence scores produced by the network should relate linearly with the precision of the predictions. However, as the plot on the bottom left side of the figure shows, the correlation coefficient between the output scores (blue crosses) and the desired linear relationship (dashed red line) is relatively low ($R^2 \approx 0.87$). As the graph on the bottom right side of the figure indicates, the variances of the estimates produced by pRGR correlate almost perfectly with the ideal linear relationship ($R^2 \approx 1.0$). Figure 3.18 provides analogous plots for results obtained using predictions from the DeepLabV2 (VGG), DeepLabV2 (ResNet) and DeepLabV3+ network configurations, with coefficients $R^2 \geq 0.989$ observed for all cases.




(a) DeepLabV2(VGG) (b) DeepLabV2(ResNet) (c) DeepLabV3+

Figure 3.18: Correlation between rankings according to accuracy and pRGR's uncertainty estimate, for refinement of predictions collected from different DeepLab models.

#### 3.2.3.4 Runtime analysis

All experiments reported in this section were performed with an implementation of the algorithm described in Section 3.2.2 that combines a main function written in MATLAB® with a C++ implementation of Algorithm 2, which is responsible for region growing. Figure 3.19 summarizes average measurements of runtime observed for refinement of predictions provided by *a*) a DeepLabV2 (VGG) model and *b*) a DeepLabV2 (ResNet) model for the 1449 images composing the PASCAL validation set. As described in Table 3.3, predictions provided by the VGG-based model are refined using two refinement steps, in contrast to the single refinement loop performed on predictions provided by the ResNet-based model.

Overall, these results indicate that an average of 0.95 second is required for each refinement loop on each class-specific scoremap, with the overall runtime per image scaling near linearly according to the amount of detected classes. A breakdown of the average runtime according to the steps performed by the algorithm reveals that data loading (image and scoremaps) requires $\sim 0.15$ second, while the growing process of pixel-cluster assignments requires nearly all the remaining 0.80 second. Since each refinement iteration can be performed in parallel, in the near future, we intend to accelerate the algorithm through a GPU-based implementation.

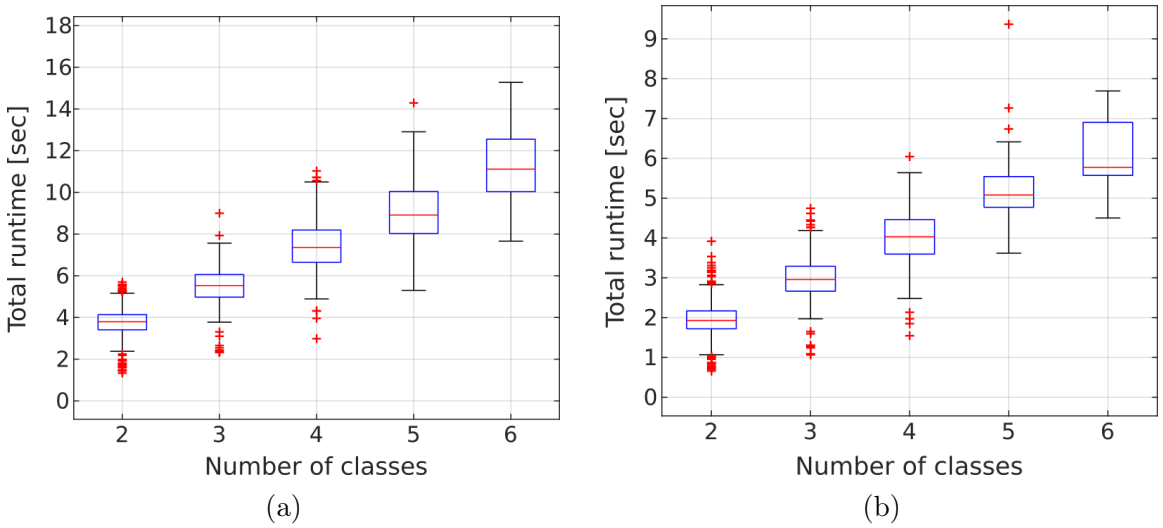



Figure 3.19: Runtime analysis of pRGR's current implementation. For predictions collected for the PASCAL VOC *val* set, the plots summarize runtime according to number of detected classes for: *a)* a double refinement procedure, and *b)* a single refinement procedure.

# CHAPTER 4
# SEMI-AUTOMATED ANNOTATION OF IMAGE SEGMENTATION DATASETS

The rapid rise in popularity of deep learning models in computer vision has brought a corresponding demand for labeled data. Depending on the image understanding task, the required annotations may range from tags at the image level (image classification), to bounding boxes (object detection) or pixel-level annotations (image segmentation).

For all cases, varied and high-quality image annotations are crucial for both training and evaluation of models that are accurate and robust. Currently, most CNN models successful at image understanding tasks [22–24] are pre-trained on the ImageNet [25] and COCO [5] datasets, due to their large variability.

However, manual labeling of large datasets is challenging and time-consuming. The costs reported for the COCO dataset in [5] illustrate these difficulties. Containing over 2.5 million object instances, its labeling using Amazon's Mechanical Turk (AMT) required: $\approx 20k$ worker hours for category labeling at image-level; $\approx 10k$ hours for instance spotting; and staggering $\approx 55k$ hours for instance segmentation.

To meet the need for large, labeled datasets, several approaches have been proposed. Different types of crowdsourcing strategies have been used to generate labeled data quickly, from commercially available solutions such as the AMT, annotation parties [3], volunteer/citizen science initiatives [124], and custom-built pipelines [125].

In the domain of image segmentation, rather than selecting individual pixels, a popular strategy consists of approximating segmentations as polygons, which can be problematic for objects with complex boundary structures. Other strategies focus on labeling pre-segmented regions, such as superpixels [6,126]. Although these strategies accelerate the annotation process, the segmentation quality is at risk in scenarios where the pre-computed regions fail to properly attach to boundaries.

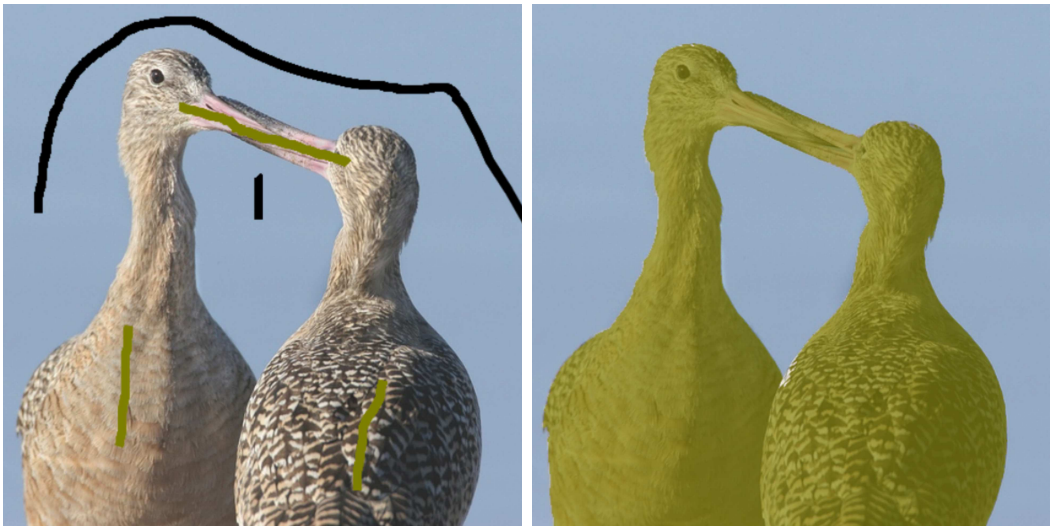


Figure 4.1: Our annotation tool generates high-quality segmentation masks using simple freehand traces as input. From the few user traces illustrated in the left image, FreeLabel outputs the object segmentations indicated by the yellow overlay in the right image [8] (©2019 IEEE).

To minimize the need for finely-annotated training data, the development of weakly-supervised training methods is also an active field of research. Strategies for the propagation of sparse annotations include graph cuts [69], level sets [75] and graphical models [81]. As the leaderboard of the PASCAL VOC 2012 dataset[1] shows, the performance of models trained in this way is still noticeably worse than models trained with fully annotated masks.

We combine ideas from both the existing annotation tools and the field of semi-supervised learning to facilitate and minimize the amount of user interactions for annotating image segmentation masks, ultimately reducing labeling costs. Our contribution consists of a web-based tool, named FreeLabel [8], which allows the user to trace lines or “freehand" scribbles of different thicknesses for the different categories present in an image (Figure 4.1). These scribbles are propagated to the remaining

[1] http://host.robots.ox.ac.uk:8080/leaderboard/

unlabeled pixels using the Region Growing Refinement (RGR) algorithm for semantic segmentation refinement, which was published in [1] and described in Section 3.1.

We assess the applicability of our tool in two contexts: the first is general object segmentation, exemplified by the PASCAL VOC dataset that has pixel-accurate labels for multiple different categories; and the second is the annotation of images of fruit tree flowers, which has applications in precision agriculture [2, 9]. In the first context, we analyze how long it takes for users to become familiar with our tool, and also the average annotation time and the segmentation quality they obtain in comparison with the official PASCAL ground-truth.

The first context serves as training for the second, where images of flowers of multiple fruit tree species are annotated [9]. In this scenario, we evaluate how well users can annotate images for which no ground-truth is available, and thus no intermediate feedback is provided.

In summary, our contributions to the state of the art are:

- *FreeLabel, an open-source, web-based tool for interactive annotation that is shown to be intuitive and effective*, allowing users to generate high-quality segmentations in an average time of 60 seconds per object for the PASCAL dataset;
- *FreeLabel can be easily configured for any object category or dataset*, an advantage inherited from the underlying unsupervised region growing algorithm and the modular implementation of the tool;
- *public release of the tool at* `coviss.org/freelabel`
- *the web-based structure of FreeLabel allows crowdsourcing and, when data privacy is of concern, private annotation* using a local deployment.

## 4.1 Related work

As reported in [3] and discussed in Section 2.4.1, the process of annotating images composing the PASCAL dataset with pixel-level accuracy was extremely time-

consuming, even though a 5-pixel wide tolerance margin was allowed around each object.

Comprising 2.5 million objects instances in $328k$ images, the COCO dataset was labeled by AMT workers using an adapted version of the OpenSurfaces interface [127]. The OpenSurfaces interface resembles the LabelMe web-based annotation tool [128], which was introduced in 2008 and is still widely used for segmentation annotation. Users provide object segmentations by tracing polygons along its boundary and typing the object name after completing the polygon. However, as mentioned in [5, 128], quality control is an important concern with this scheme. High-quality segmentations of objects with complex boundary structures require large numbers of vertices, leading to a trade-off between quality versus time spent to label each object. For annotation of the COCO dataset, its authors opted to minimize costs by collecting only one annotation for each instance, which required on average 79 seconds per object. Yet, despite efforts such as quality verification steps, the dataset still contains some segmentation masks that poorly attach to the object boundaries [1].

The Cityscapes dataset for semantic urban scene understanding [129] was also annotated using layered polygons. To ensure that rich and high-quality pixel-level segmentation masks were obtained, its corresponding $5k$ images were annotated in-house. Over 1.5 hours were required on average for annotation and quality control of each image with a restricted pool of high-quality annotators.

Alternative labeling strategies exploit superpixels to facilitate the annotation process. The interface used for labeling the COCO-Stuff dataset [6] combines SLICO superpixels [60] with a size-adjustable paintbrush tool that enables labeling of large regions at once. As mentioned by Tangseng et al. in [126], superpixel errors can lead to significant annotation errors with this kind of interface. To minimize these artifacts, the authors described in [126] a interface that performs morphology-based boundary smoothing and allows the annotator to select the desired superpixel size to

improve boundary adherence. However, this increases the complexity of the task, as the user has to try different configurations and label each superpixel individually.

Recently, an alternative approach for interactive segmentation was introduced in [130], where a CNN is trained to generate segmentation masks from extreme points specified by the user. The tool is shown to provide annotations of good quality in a timely manner, but requires supervised training and more computational resources.

### 4.1.1 Good practices for design of annotation tools

Vondrick et al. in [131] provide a set of best practices for crowdsourced video annotation, based on a large-scale, three-year on image annotation approaches. A critical observation is that annotating platforms must aim at minimizing the cognitive load of the user. As backed by psychology studies [132], minimizing interruptions and choices helps reduce user anxiety and increase efficiency. Moreover, they observed that providing motivational feedback increases the workers' confidence that their work will not be rejected, which encourages them to continue annotating.

Games With A Purpose (GWAP) exploit the idea that adding game-like elements to interfaces additionally motivates users to perform tasks of interest. The ESP Game [133] for image labeling is a widely known example: an image is shown to two players (users) and, without external communication, both enter possible words until a word is agreed upon. The common word becomes a label for the image. Other examples are the Peekaboom game for object localization [134], Verbosity to collect commonsense facts about words [135], and Phylo for multiple sequence analysis [136].

Users play for the desire of being entertained, rather than for money or altruism [137]. Timed response, score keeping, and randomness are important features for designing challenging and hence enjoyable games [137], as players are motivated to play to increase their skill level or to score higher than other players. Compared to subjective and verbal instructions, scores are a more intuitive form of feedback to

the user as they combine multiple aspects that are relevant to the task into a single performance metric.

## 4.2 FreeLabel annotation tool

Our objective is to develop a web-based labeling interface that: i) is intuitive to use, ii) allows users to quickly provide high-quality annotations, and iii) can be easily adapted for different datasets and categories. As observed in Section 4.1.1, a good user interface should minimize the cognitive load on the user. Thus, instead of using propagation techniques that require supervised training or manual tuning of different sets of parameters, our tool exploits the RGR algorithm for unsupervised region growing. Based on related works, limitations of current tools, and our own previous experiences with image annotation, we opted for designing a tool in which the user input consists of simply drawing scribbles (freehand traces) or straight line segments on the images.

By keeping all the parameterizations of the RGR algorithm fixed, we avoid any non-intuitive burden on the users. The quality of the segmentation provided by RGR is proportional to the amount and quality of initial seeds available. Hence, the user interaction to guide the growing process is quite intuitive, with simple guidelines:

- traces are grown based on color similarity and must be provided within the boundaries of the corresponding objects;
- thicker traces act as enforcement for the growing algorithm, since more seeds are available than for thin traces;
- if any region is incorrectly labeled by RGR, the user can easily correct it by adding a new trace of the correct category.

In addition to its simple formulation, we found the RGR implementation to be notably suitable for multi-class segmentation annotation. Its growing process is class agnostic, propagating initial seeds into clusters regardless of seed label. This

is advantageous in terms of running time, as the growing process has the same computational complexity regardless of the number of classes present in the image (the average runtime is lower than 1 second for PASCAL images [1]). After clusters are formed for each set of seeds, they are classified into semantic categories by means of simple majority voting. Figure 4.2 shows an example of this process, where each cluster is assigned to the class for which it contains the most labeled pixels.

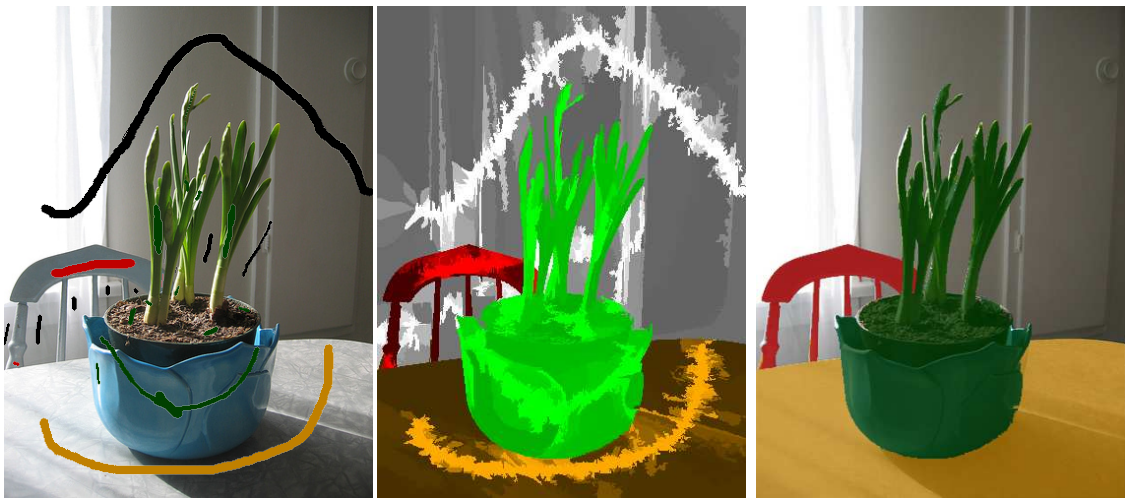


Figure 4.2: Illustration of how traces are propagated to neighboring pixels. *Left:* input traces drawn by the user. *Center:* the brightness (intensity) of the color in each pixel is proportional to the score computed for its most likely category. For better visualization, *background* traces are shown in black, while the *background* likelihood is in grayscale from black (lowest) to white (highest). *Right:* final segmentation obtained using maximum category likelihood per-pixel, with transparent *background* [8] (©2019 IEEE).

### 4.2.1 FreeLabel functionality

Figure 4.3 shows a screenshot illustrating the functionality of our interface, together with an example of high-quality segmentation masks obtained from only a few user interactions.

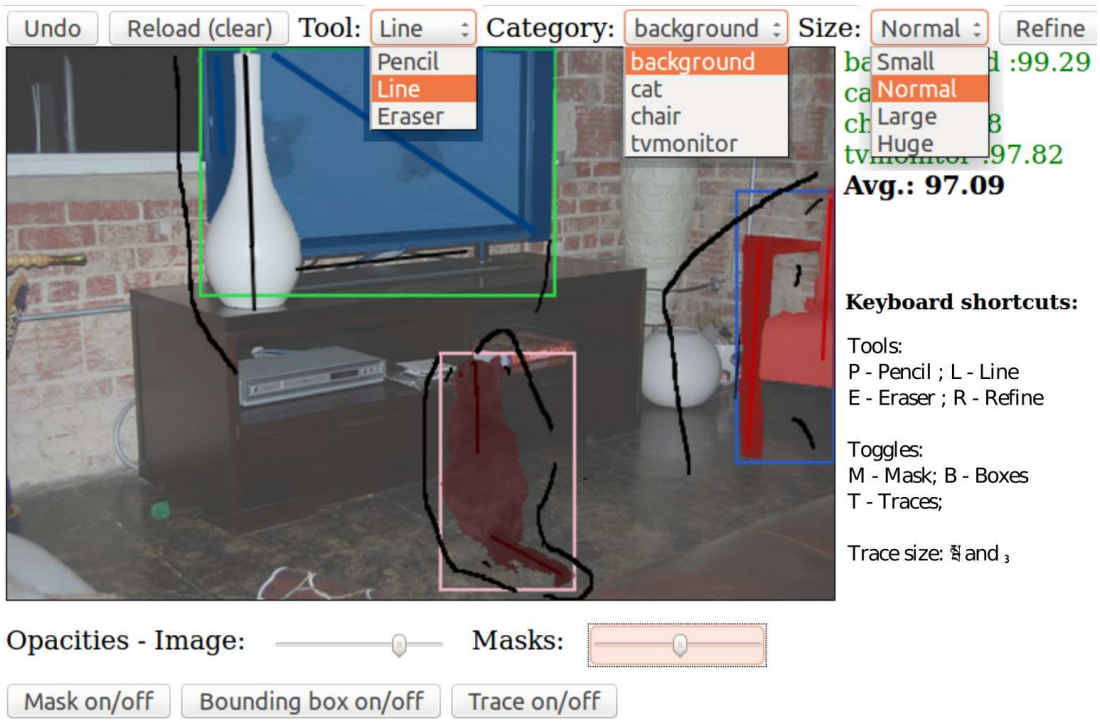



Figure 4.3: FreeLabel's graphical user interface.. Users can draw with a freehand pencil or line segments. An eraser allows undoing small errors. Dialog boxes allow the user to select the object categories associated with the current trace, as well as adjust tool sizes. To help with visibility, other options such as opacity and masks are available via slider bars [8] (©2019 IEEE).

Three tools are available for drawing and adjusting traces using the mouse:

- *Pencil* : used for quickly tracing freehand scribbles. Once the user holds down the mouse's left-button, traces corresponding to the mouse trajectory are drawn. It is especially useful for regions that do not require high precision;
- *Line* : traces straight lines connecting the point where the user clicked the mouse button to the point where it was released. It is especially helpful for straight and thin structures, such as chairs' legs and animals' limbs.
- *Eraser* : used to correct imprecisions in provided scribbles, such as small portions protruding outside the corresponding object's boundary.

Each tool can be configured with four different thicknesses: *small* ($1px$ thick), *normal* ($2px$), *large* ($4px$) or *huge* ($8px$). After tracing scribbles over the image, the user can invoke the RGR algorithm by simply clicking the *Refine* button, which automatically grows segmentation masks from the provided traces. To annotate smaller objects, the user can zoom in/out using the mouse scroll, as in any modern web-browser. Finally, keyboard shortcuts are available for all the commands to facilitate the annotation process.

In addition to intuitive commands, visualization is another key factor that impacts the labeling experience and annotation quality. Similar to the PASCAL, COCO, and other datasets, a specific color is associated to the traces and masks of each category. For the background, traces are shown in black and the masks are invisible. To handle scenarios where the image is too dark or contains colors with poor contrast to traces and/or masks, our interface allows the user to control the brightness (opacity) of both the image and the segmentation masks using the sliders under the canvas. Moreover, masks and traces can be hidden/shown with the click of the corresponding toggle buttons.

### 4.2.2 Implementation

Our FreeLabel tool for segmentation annotation relies on three main building blocks: a graphical user interface (GUI), the Django framework, and the RGR algorithm. Figure 4.4 summarizes the relationships among these elements.

An important criterion for our design choices concerns how easily the user's inputs and the RGR algorithm can be combined for the computation of segmentation masks. With an open-source web interface as ultimate our goal, we adapted RGR's original MATLAB® implementation to Python and opted for the Django [138] platform as the web framework.

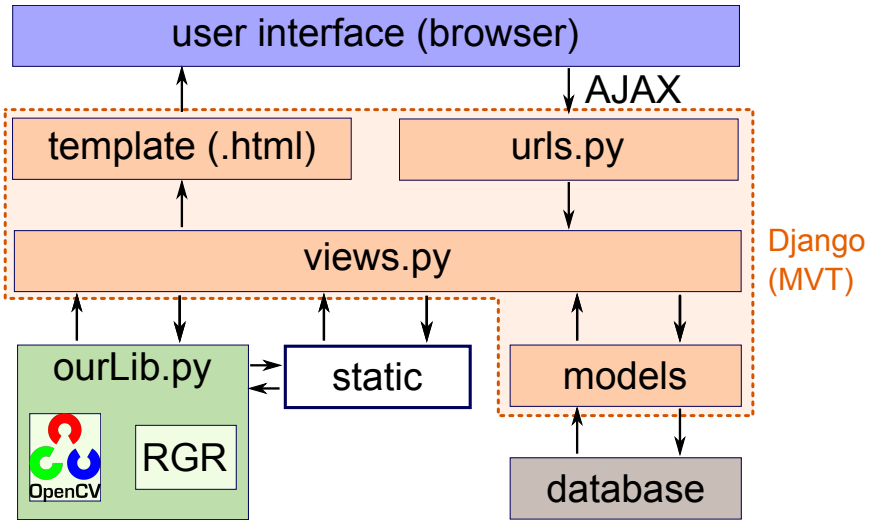



Figure 4.4: Diagram summarizing how the different modules of FreeLabel interact with one another [8] (©2019 IEEE).

Django is a free, open source Python framework that follows the Model-View-Template architectural pattern. The *Model* layer allows access to database information without requiring any knowledge of the intricacies of database rules. The *View* logic layer of Django handles the communication between the *Model* and the *Templates*, which correspond to the exhibition layers that define what is shown to users through the browser.

Using Figure 4.4 as guidance, a top-down walk-through of our tool's implementation starts with the graphical interface displayed by the web browser to the user. The design and functionality described in Section 4.2.1 and exemplified in Figure 4.3 are implemented as customized Django templates, using HTML/Javascript. For actions requiring the execution of Python commands, the template (.html) file triggers an AJAX call that is mapped to a corresponding function in *views* (.py). This layer mediates the access to the database (through the *Model* layer), static files, or any customized Python function.

Aiming at a modular implementation that can be easily tailored for different datasets or configurations, we package the implementation of RGR and other custom functions into a separate Python library (*ourLib.py*). This includes functions using the OpenCV [139] library, which are responsible for image loading and converting the outputs of RGR from mathematical arrays to images for visualization.

RGR is used as the core component of FreeLabel, and adapted in two minor aspects to compose the annotation tool. The original algorithm described in Section 3.1 focuses on the refinement of a CNN's semantic segmentation predictions, a scenario with coarse segmentation masks as input. While for that case sampling fewer seeds is beneficial to filter out false-positives, in our scenario we aim at minimizing the required number of user interactions. Since the user inputs tend to be sparse but highly-accurate, we increase the percentage of seeds sampled in each Monte Carlo iteration to 75% of the annotated pixels, with 8 iterations per run. Moreover, we remove RGR's constraint that automatically classifies as background any pixel significantly distant from labeled neighbors in terms of appearance and spatial position. By removing this constraint, RGR will assign to each unlabeled pixel the category provided for its nearest neighbor, regardless of how far they might be. If the propagated label is incorrect, the user can easily improve the segmentation by tracing an additional scribble in the corresponding region.

## 4.3 Experiments and results

We evaluate our tool in terms of: i) quality of the obtained segmentation masks, and ii) time required by users to annotate images using FreeLabel. To that end, we defined first a task in which users were asked to annotate images from the PASCAL VOC 2012 dataset. We opted for this dataset as it contains good quality segmentations of multiple object categories and is widely used by the computer vision

community, such that it represents a good reference standard for anyone searching for a suitable annotation tool.

Inspired by the idea of GWAP, we designed a game-like version of FreeLabel for the annotation of PASCAL images. Ideally, users must provide high-quality segmentation but also be as quick as possible, which represents a trade-off for which it is difficult to provide the annotators with clear guidelines. We therefore employ a game with a simple unified score metric that combines annotation time and mean intersection over union ($mIoU$) between the obtained masks and corresponding ground-truth annotations, which is computed according to the official PASCAL metrics. The quality of the segmentation must be the main priority, while the time spent on each image is a secondary concern. Thus, as summarized in Figure 4.5, we use accuracy ($mIoU$) as the base factor for score computation, with a "bonus" multiplying factor that is proportional to the time spent on each image.

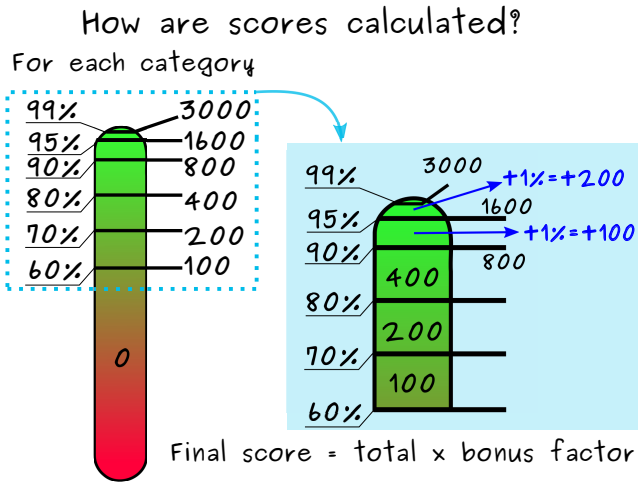



Figure 4.5: Score chart presented as reference for the game where users are asked to label PASCAL images in an accurate and timely manner [8] (©2019 IEEE).

The main goal of this metric is to provide feedback to the users regarding how well they are performing the task, such that we do not focus on a more rigorous formulation for score computation. Instead, we aim at motivating the user to obtain the highest accuracy possible by increasing the base score progressively as the $mIoU$ approaches 100%.

Let $N$ denote the number of objects in an image. Based on the performance of expert labelers, we roughly estimated an expected time of 60 seconds for an image with $N = 1$, plus an extra 30 seconds per object when $N \geq 2$. To motivate users to be quick, we thus multiply the base score with a bonus factor according to Eq. 4.1: $2\times$ if the user annotates the image in no more than the expected time $T$, linearly decaying to $1\times$ if the annotation time $t$ takes longer than $2T$.

$$bonus = \max\left(2 + \frac{T - t}{T}, 1\right) \tag{4.1}$$

$$T = 60 + 30 \times (N - 1) \quad \text{[sec]}.$$

After showing the participants a training video, we asked seven different users to label an average of 25 images each, in a task expected to take approximately 1 hour. We followed the official PASCAL annotation guidelines [3], indicating with bounding boxes the objects to be annotated by the users.

Figure 4.6 summarizes the average quality ($mIoU$) and average time needed to annotate the different objects in the images. Overall, users provided segmentations with 92.8% overlap with the ground-truth masks, at a mean pace of 61.3 seconds per object. As a reference, this is significantly faster than the average 79 sec/object required for annotating the COCO dataset using the OpenSurfaces tool [5].

We also observed which strategies were adopted by the most successful users. The two rightmost plots in Figure 4.6 summarize the frequency of usage of the *Refine* button by each user and the average image area covered by their scribbles, respectively. User #2 exemplifies the usefulness of interactivity using RGR: by frequently

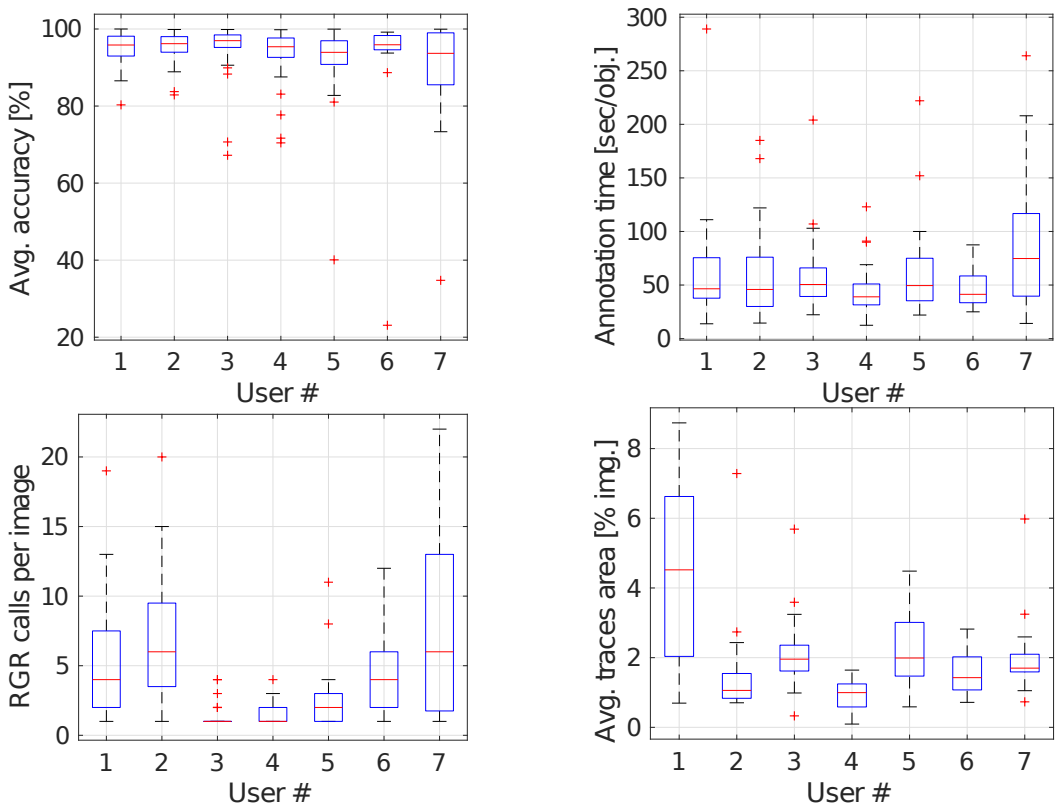



Figure 4.6: Distribution of the accuracies, annotation times, number of *Refine* calls and average image area covered by user traces for annotating images from the PASCAL dataset [8] (©2019 IEEE).

using the *Refine* option, this user obtained one of the highest accuracy averages, with fewer low-quality outliers. This user also drew fewer traces and thus finished the task faster than others who provided annotations of similar quality.

Figure 4.7 allows an analysis per object category that further highlights the benefits of FreeLabel. As the median values of 95.5% overall accuracy and 50.1 seconds per object suggest, the presence of outliers is confirmed by inspecting results for categories such as *bicycle, chair* and *potted plant*. These are notably harder to label than instances from classes like *airplane, cows* and *trains*, which present fewer enclosed regions or thin structures. However, despite requiring longer annotation times, high-quality segmentations can still be obtained for such harder categories.

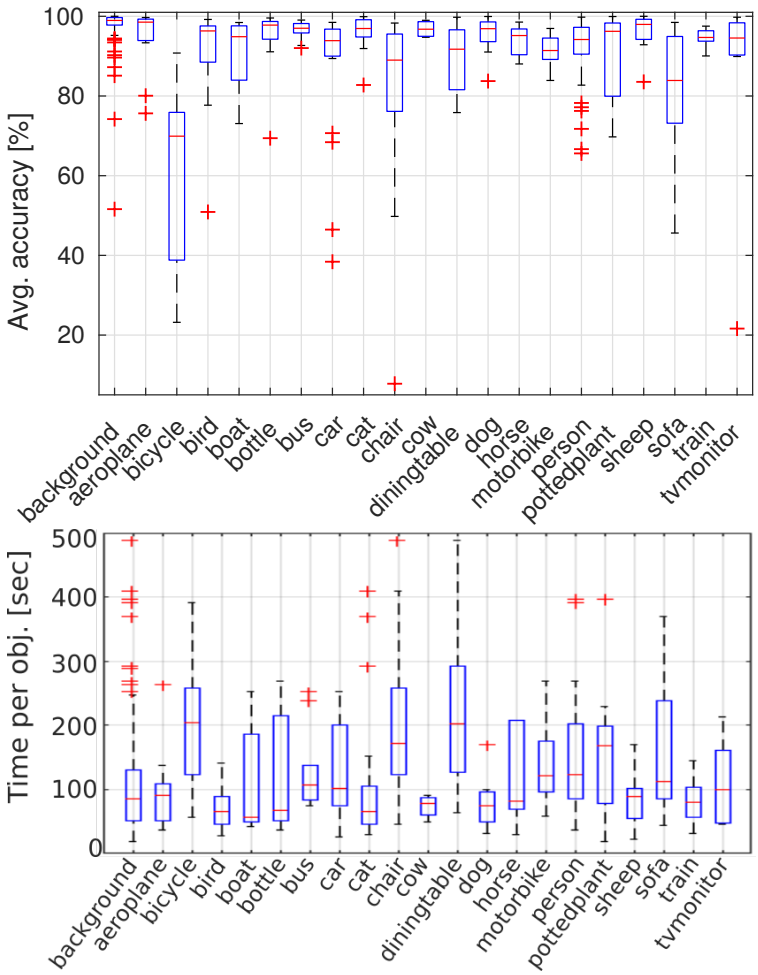



Figure 4.7: Distribution of average accuracy (top) and annotation time (bottom) for objects of different categories in the PASCAL dataset [8] (©2019 IEEE).

Figure 4.8 is a compilation of annotation examples provided by the users, with the *bicycle* example illustrating the quality of segmentation that can be obtained even for complex objects.

### 4.3.1 Annotation of unlabeled images

To demonstrate the suitability of FreeLabel for the realistic scenario of annotating unlabeled datasets, we performed experiments in which eight users were asked to annotate images of a significantly different dataset. We chose the dataset made

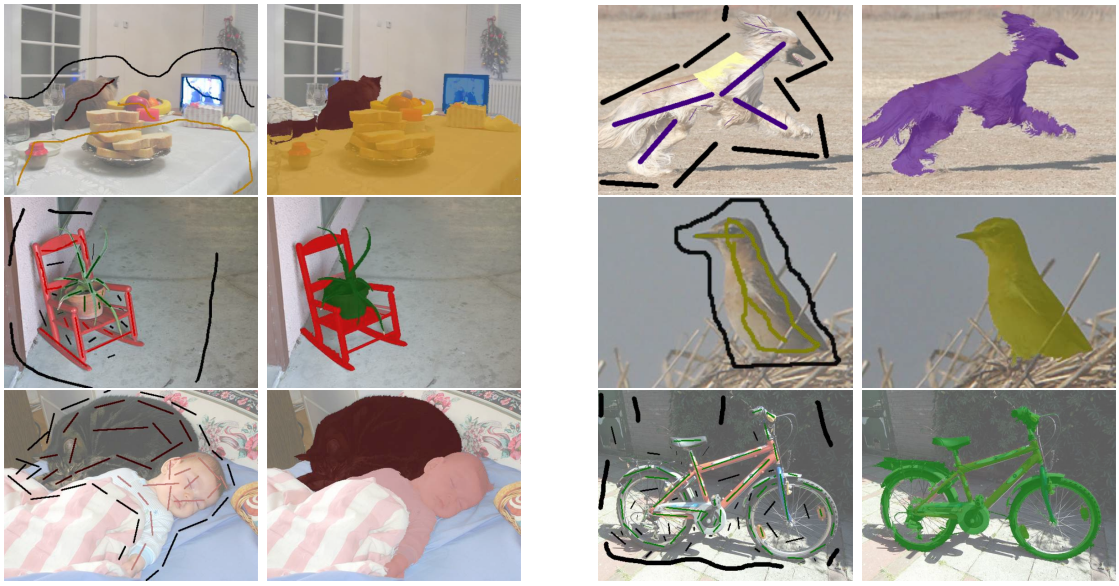


Figure 4.8: Examples of annotations provided by users for the PASCAL dataset using FreeLabel. For both columns, images in the *left* illustrate user annotations. In the *right*, final grown mask generated by FreeLabel from the corresponding inputs [8] (©2019 IEEE).

publicly available in [2, 27], which contains images of multiple species of fruit-flowers that were acquired under varied conditions. Since these are high-resolution images ($2704 \times 1520px$) containing dozens of small flowers, we split each image into 16 blocks of $676 \times 380$ pixels to facilitate the annotation process.

With the lessons learned from the PASCAL experiments, we designed a new training sequence (video available together with the tool) that emphasizes good strategies for efficient labeling with FreeLabel. Before annotating the flowers, all users were required to annotate 10 PASCAL images with a minimum accuracy of 90% per category. Our rationale is that annotating the PASCAL images in a game-format works as a training session in which the users become familiar with the interface and grasp the main guidelines for annotating any type of image segmentation dataset.

Preliminary experiments indicated that the lack of performance feedback harms the motivation of the users, and as consequence, the quality of the correspond-

ing segmentations. Hence, we structured the annotation sessions such that each user was required to label 9 blocks of different flower images, in batches of 3 blocks each. Each batch contained 2 non-annotated blocks and 1 block for which ground-truth was available. We used the ground-truth image blocks as checkpoints: if the segmentation provided by the user did not meet a certain accuracy threshold, the user would have to redo the entire batch of 3 images. The ground-truth annotations are never shown to the users, such that while only every third image is actually used to compute the average accuracy, we "deceive" the users to believe that all images are verified and must thus be accurately labeled. Moreover, we used a rather lower accuracy threshold of 70%, as the main intent is just to avoid very poor annotations.

Figure 4.9: Examples of flower annotations provided by users using FreeLabel. From left to right, images contain flowers of apple, peach and pear flowers, respectively. The colormap boundaries illustrate how many users labeled the enclosed regions as flower. Colors proportionally range from dark blue (one user) to dark red (all users labeled it as flower) [8] (©2019 IEEE).

The examples in Figure 4.9 demonstrate the effectiveness of this strategy for the annotation of unlabeled images, illustrating for each enclosed region how many users labeled it as flower. This representation qualitatively demonstrates how the annotations provided by the different users for the three different datasets converge to ideal segmentation masks. Such convergence suggests that majority voting can be

used to approximate the ideal masks, which we then use to statistically evaluate the variability of the annotations provided for images without ground-truth.

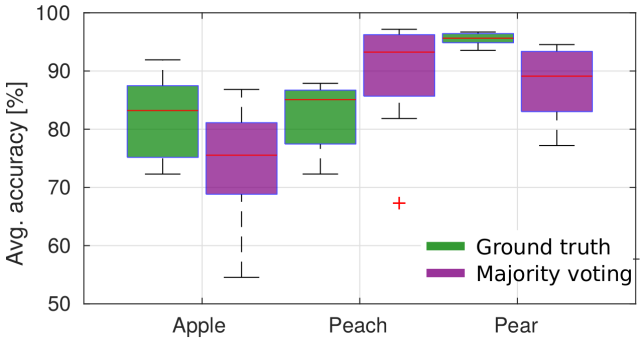



Figure 4.10: Distribution of the average accuracy obtained by the users for annotation of flower datasets [8] (©2019 IEEE).

Figure 4.10 summarizes the average accuracy and deviations observed for the images with and without ground-truth available (in green and purple, respectively). The average overlap between the segmentations provided by the users and the available ground-truth masks were higher than 80% for the three different datasets, reaching 95.5% for the *Pear* image. The higher deviations for the *Apple* and *Peach* datasets are mostly associated with the annotation of small flower buds and mistakes related to bright leaves on the apple images. Such mistakes are visible as well in the examples in Figure 4.9. Finally, the deviations observed for ground-truth images are similar to the ones observed for the images without ground-truth, which indicates a relatively consistent user performance for both groups of images.

**Acknowledgement.** This work was supported by USDA ARS agreement #584080-5-020. Mention of trade names or commercial products in this publication is solely for the purpose of providing specific information and does not imply recommendation or endorsement by the U.S. Department of Agriculture. USDA is an equal oppor-

tunity provider and employer. We thank our colleagues Abubakar Siddique, Enrico Prampolini, Reza Mozhdehi, Jamir Jyoti, Brian Stumph, Scott Wolford, and Larry Crim who collaborated with data collection. We thank NVIDIA for providing the GPU used in this work.

## 4.4 Active learning and semi-supervised annotation

In addition to the design of tools such as FreeLabel to simplify the annotation process, another research direction towards minimization of annotation costs focus on designing better sample selection strategies. Training deep neural networks using samples selected at random from a large annotated dataset is inefficient. While many data entries may contain redundant information from the perspective of the features that the network must learn to perform its job, other features of interest may be underrepresented due to the lack of sufficiently diverse training samples.

Optimal selection of training data is relevant for reducing the computational power required for data processing, as well as minimizing the human efforts required for dataset labeling. In this context, the concept of Active Learning (AL) is of particular relevance. Instead of generating a training dataset by annotating all available data at once, the AL process consists in the following sequence of steps that are performed in a loop, until either all available data has been used or a satisfactory model performance is obtained:

1. from a pool of unlabeled data, label a subset of data;
2. train the model using the available labeled data;
3. deploy the trained model generated by step 2) for inference on the remaining unlabeled data entries;
4. based on the model's inferences and a predefined sampling criteria, select an extra subset of unlabeled data to be annotated by human experts.

In such AL frameworks, the sampling criteria thus play a key role on defining which unlabeled data points shall be labeled by the human experts for further model training. As summarized in [140], *diversity sampling* and *uncertainty sampling* are two of the main strategies used to that end.

Diversity sampling focuses on covering a diverse set of data in order to expand the model's exposure to different characteristics of the data distribution. This is of particular relevance for scenarios where knowledge about the model's capabilities is rather limited or unavailable, or, as summarized in [140], to handle "unknown unknowns". Meanwhile, another scenario is where uncertainty estimations provide insights on the model's "known unknowns", i.e., data entries for which inferences provided by the model itself indicate the need for further training.

In this context, we thus investigated strategies that use uncertainty estimation techniques based on the probabilistic Semantic Segmentation Refinement (pRGR) algorithm [116] to select the most informative samples for training a semantic segmentation network, aiming at sample selection strategies that will lead to more balanced training sets.

In Section 3.2.3.3, we demonstrated using the PASCAL dataset that pRGR provides pixel-wise uncertainty estimations which highly correlate with the accuracy of semantic segmentation predictions. Thus, our efforts towards an active learning system focus on efficiently exploiting pRGR's uncertainty estimations as a proxy to select the most informative images for training the network. Specifically, we need to identify principled strategies for: i) aggregating the pixel-wise uncertainty estimates shown in Figure 3.17 into image-level estimates, and ii) identifying strategies to effectively sample according to these estimates. The remainder of this section discusses some of our efforts in that direction.

**From pixel-level to image-level estimates.**

We have considered two approaches to combine uncertainty estimates at pixel-level into image-level descriptors. While the first strategy simply consists on computing the average pixel uncertainty values across the whole image, the second approach exploits histograms of pRGR's variance estimations across the images. More specifically, we compute the frequency of estimated pixel-confidence values in a range divided into 100 bins, and then compute the entropy of the obtained histogram.

To compare these different strategies for ranking images according to uncertainty, we followed the approach described in [141] and generated sparsification plots to quantify how well an uncertainty estimation rank correlates with an oracle rank sorted by error (i.e., accuracy). As illustrated in Figure 4.11, sparsification curves are generated with the intuition that when gradually removing the samples with the highest uncertainty, the accuracy should monotonically increase.

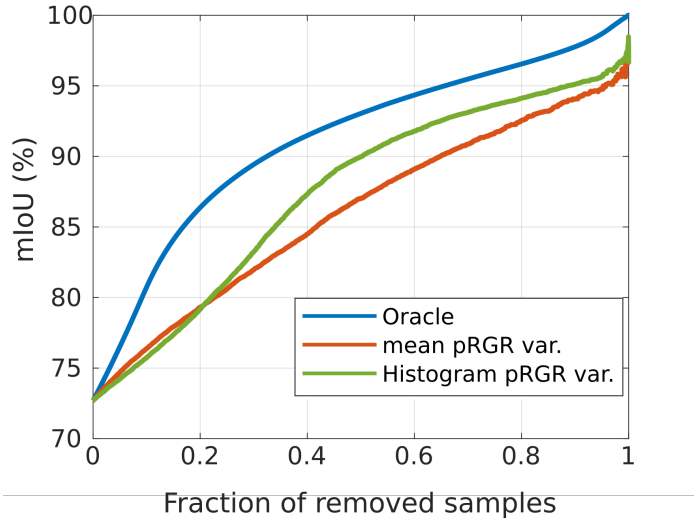



Figure 4.11: Sparsification curves of the different evaluated techniques for ranking images of the PASCAL *trainaug* dataset according to uncertainty.

More specifically, Figure 4.11 depicts rankings provided by mean and histogram-based approaches after collecting pRGR's variances generated from refinement of predictions provided by a DeeplabV2 (ResNet) model trained on the PASCAL dataset. These results show that while the mean of the pixel-wise variances performs relatively poorly (i.e., it shows the lowest similarity with the oracle), the entropy of the histograms of the variances shows improved performance for uncertainties higher than approximately 0.2.

**Sampling according to uncertainty.**

Given a reliable ranking of images according to estimated uncertainties, the next step towards active learning consists on identifying an effective sampling strategy according to uncertainty. While the most straightforward approach consists of selecting samples of highest uncertainty for training, some problems can arise from this strategy. At early learning stages, when more general descriptors still have to be learned, it can be difficult for the model to learn from the most challenging samples. Moreover, a problem known as *cold start* can also occur [142], where a poor starting training set can lead to inaccurately biased uncertainty estimations. Therefore, we perform experiments considering three sampling strategies for uncertainty-based ranking of images: i) sample the most uncertain images; ii) sample the least uncertain images; iii) diversify the sampling over the whole rank of images, selecting samples over equally spaced bins across the uncertainty-based ranking. We refer to the latest strategy as *stair-wise* sampling, and provide in Figure 4.12 an illustration of the three sampling strategies considered.

Figure 4.13(a) summarizes the results obtained for experiments combining a ranking based on the histogram of pRGR uncertainties with each of these three sampling strategies, with the goal of training a DeepLab-v3+ model on the PASCAL dataset. In this experiment, we try to identify informative training images composing

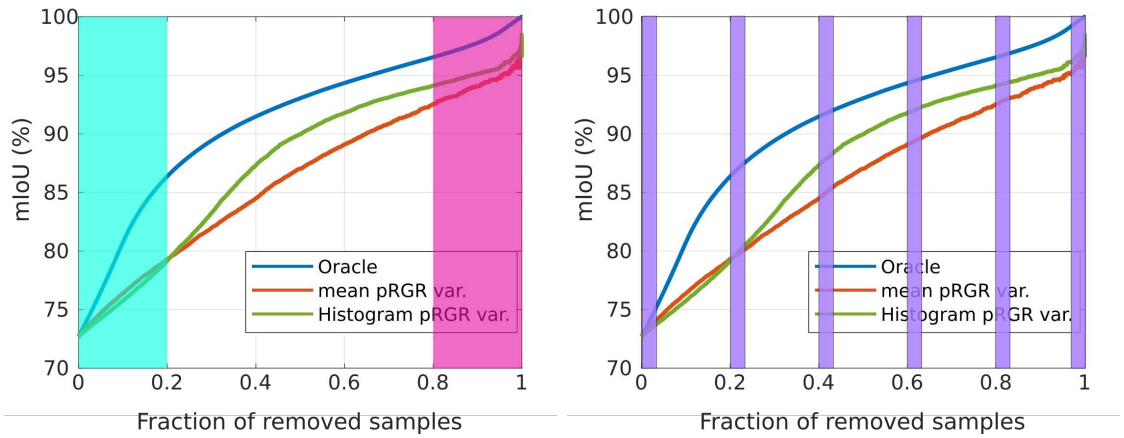



Figure 4.12: Illustration of the three sampling strategies considered for uncertainty-based sample selection. *Cyan:* select samples with lowest uncertainty; *magenta:* select samples with highest uncertainty; *purple:* select samples in a stair-wise fashion over the uncertainty-based ranks.

the PASCAL *trainaug* subset, by ranking them according to uncertainty estimations provided by pRGR after refining predictions of a DeepLab-v2 model already trained on the PASCAL dataset. In an approach inspired by transfer learning techniques, our assumption is that a new model could then be more effectively trained using only a subset of the most informative images.

As a baseline, we randomly sample three subsets of images, train the model on them, and estimate their mean performances in terms of mIoU in the validation set. In Figure 4.13, the plot on the left side demonstrates that training using a subset defined according to stair-wise sampling leads to performances significantly better than the ones observed by sampling the least or most uncertain images. Moreover, the mIoU curves indicate that subsets sampled using this strategy allow on average slightly more effective learning than randomly sampling images.

Our final experiment on the PASCAL dataset aimed at emulating the first training loop of an active learning system. First, a DeepLab-v3+ model is trained on a randomly selected subset containing 20% of the available training images. Then,

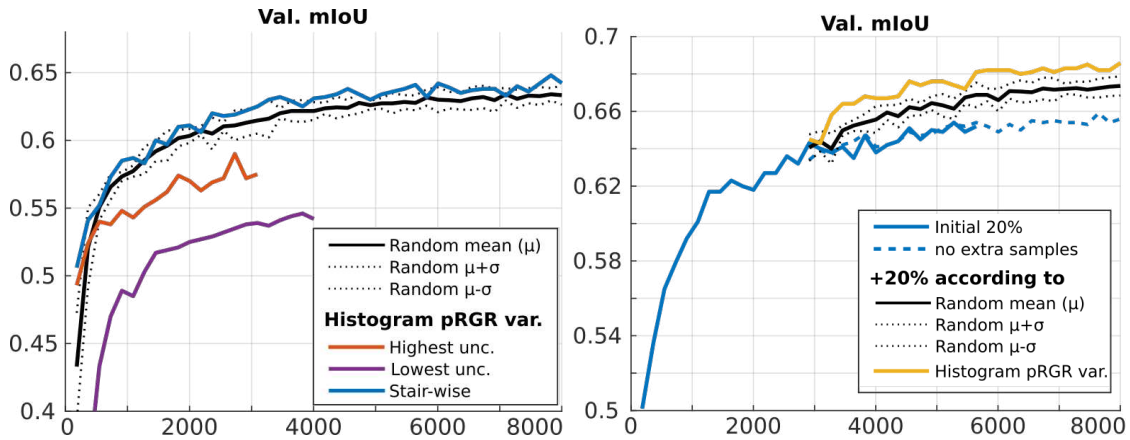



Figure 4.13: Curves of segmentation quality (in terms of validation mIoU) for models trained on subsets of the PASCAL *trainaug* dataset. *Left:* evaluation of different sampling strategies for training a DeepLab-v3+ model, based on uncertainty estimates provided by a DeepLab-v2 model. *Right:* segmentation performance on validation set after one active learning loop for a DeepLab-v3+ model.

our uncertainty-based sampling strategy is evaluated for the selection of an additional subset containing 20% new training images. As baselines, we compare performance with results obtained by training with five different randomly selected subsets of images. Stair-wise sampling is used for the method based on uncertainty estimations, as well as diversification to balance the number of samples per class. Results are summarized in the plot shown at the right-side of Figure 4.13, indicating that an histogram-based active learning strategy performs on average better than randomly selecting samples, with the AL approach yielding a curve of validation mIoU clearly above the upper-range provided by the multiple random runs.

**Experiments with satellite imagery.**

Finally, we performed AL experiments using satellite imagery, as part of collaboration with Dr. Dalton Lunga at the Oak Ridge National Laboratory (ORNL). The identification and quantification of structures such as building footprints and road maps from satellite images is of great importance for tasks that include disaster

response planning, geographical analysis of human occupation and mobility patterns, and many other applications related to socio-economics studies [143]. In addition to its potential applications, the increasing amounts of available high-quality remote sensing imagery also contributes to making it a topic of great interest. In contrast to conventional image datasets, satellite imagery is characterized by extra challenges related to the volume of data that has to be stored, annotated, and processed. Hence, it constitutes a particularly relevant study-case for AL approaches that intend to optimize the processes of data preparation and processing.

Part of the SpaceNet datasets and challenges [144], the SpaceNetV2 dataset comprises images and annotations collected from five different areas of interest (AOI) across the globe, which cover areas of the cities of Las Vegas, Rio de Janeiro, Shanghai, Paris, and Khartoum. With the challenge focusing on building footprint detection, SpaceNetV2 comprises over $685,000$ building footprints in total [145]. In the next paragraphs, we describe AL experiments performed using the SpaceNetV2 dataset as a simplified feasibility study for the domain of satellite imagery.

From an application perspective, one of the main potential benefits of an active learning framework is to optimize transfer learning across different scenarios. For a building segmentation model already trained on one AOI, high segmentation quality for another AOI should be achievable at a cost of only a few extra scenario-specific training samples. In this context, we performed experiments where a model pre-trained on 600 images from the Vegas AOI is fine-tuned using selected training samples from: a) the Shanghai AOI; and b) the Khartoum and Paris AOIs.

For the experiment using the Shanghai AOI, 1000 images were divided into an available training pool of 600 images, and a subset of 400 images for validation. We compare an uncertainty-based sampling strategy to random sampling to perform two active learning loops using 200 images each. More specifically, our active learning model uses the histogram of pRGR's variances as uncertainty estimations, with stair-

wise sampling of the uncertainty-ranked images as well as stair-wise diversification strategy based on the total pixel area detected as buildings by the pre-trained model for each image. The baseline model consists of the average performance over three models trained on randomly generated training sets.

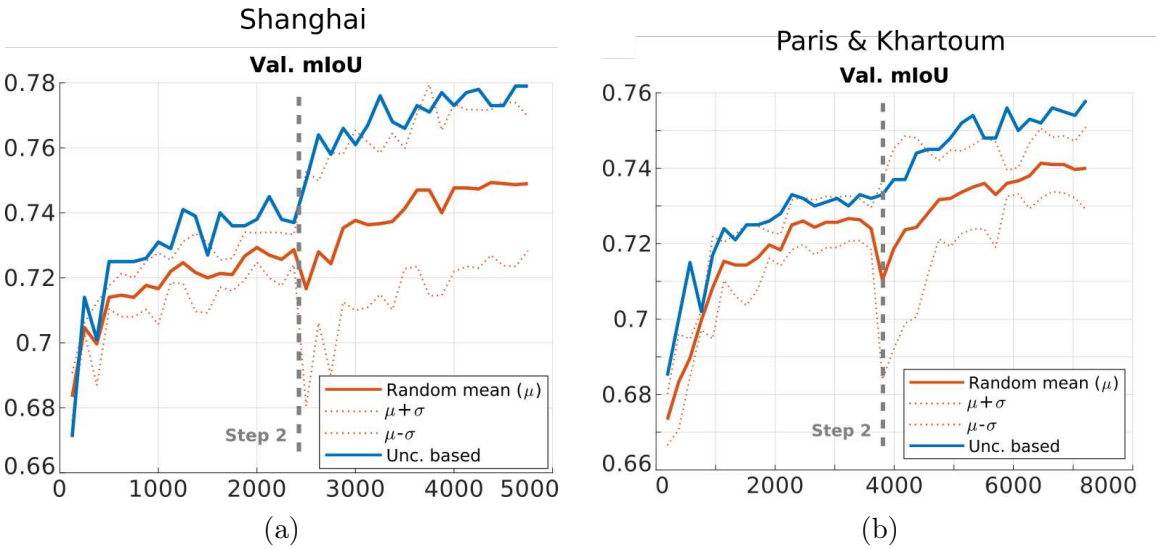



Figure 4.14: Mean intersection over union on the validation sets of a) the Shanghai AOI and b) the Paris + Khartoum AOIs. The orange lines correspond to models trained using randomly selected samples. The blue lines show the results using uncertainty-based sample selection.

Figure 4.14(a) demonstrates the benefits of our proposed active learning strategy, which achieves a peak performance of 77.9% after two learning loops (i.e., 40% of the training data). This corresponds to a mIoU 3.0% higher than the average observed for the random baseline. Moreover, the first learning loop using only 200 images reaches an mIoU only 0.4% lower than the one obtained using 400 randomly selected images. These results corroborate the potential of the active learning system to reduce the number of training samples needed and their associated labeling costs.

For the same baseline and proposed active learning strategy, our second experiment uses a total of 1500 from the Khartoum and Paris AOIs, split into 900 images as the training pool and 600 images for validation. Similarly to the previous experiment, we compare the strategies to perform two learning loops using 180 images (20% of available training pool) each.

As illustrated in Figure 4.14(b), results once again indicate performance improvements by selecting training samples using the proposed strategy, with a peak performance nearly 2.0% higher than the ones in average provided by randomly sampling. Another noteworthy point are the levels of improvement after each step. For both experiments, improvements after two steps are nearly two times higher than the ones observed after a single step, suggesting that potential further improvements could be obtained by either splitting the process into smaller steps and/or sampling additional training samples. Figure 4.15 qualitatively illustrates the benefits provided by active learning strategies for the different AOIs, which can be summarized as:

- **Improvements with extra steps (left column):** false positives such as the court on upper left image are removed as more steps are performed, while previously missed structures are detected in the bottom left example;
- **Fewer samples needed to reach acceptable performances (middle column):** as both examples in the middle column show, segmentations of nearly equal quality can be obtained after a single active learning step using an uncertainty-based sampling, which corresponds to half the samples used by the random sampling approach;
- **Active learning provides higher segmentation quality with the same amount of training samples (right column):** for the upper right example, training through active learning (uncertainty based) leads to fewer false negatives, better segmenting structures such as the buildings near upper left and

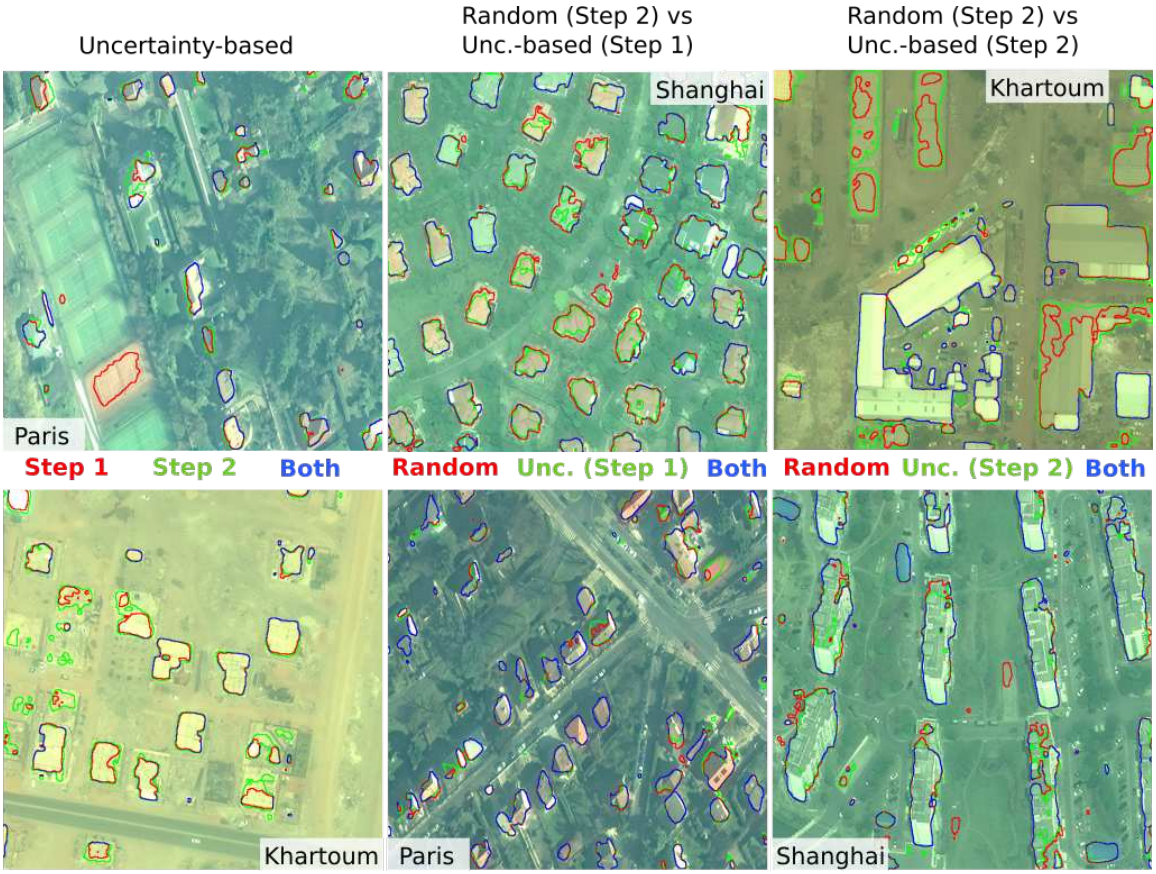



Figure 4.15: Cross-model active learning performance illustration. *Left)* Comparison of two uncertainty-based active learning steps on the Kharoum and Paris AOIs. *Middle)* Comparison of two steps of model training using random sampling, and one step model training using uncertainty-based sample selection on the Shanghai and Paris AOIs. *Right)* Comparison of both approaches after two active learning iterations on the Khartoum and Shanghai AOIs.

lower right image corners. In scenarios such as the bottom right example, false positive such as detecting parts of the road as buildings become less frequent.

# CHAPTER 5
# FRUIT FLOWER SEGMENTATION

Bloom intensity corresponds to the number of flowers present in orchards during the early growing season. Various studies have established the relationships between bloom intensity, fruit load, and fruit quality [146, 147]. Together with factors such as climate, bloom intensity is especially important to guide thinning, which consists of removing some flowers and fruitlets in the early growing season. Proper thinning directly impacts fruit market value, since it affects fruit size, coloration, taste and firmness. Moreover, accurate estimates of bloom intensity can also benefit packing houses, since early crop-load estimation greatly contributes to optimizing postharvest handling and storage processes.

Despite its importance, there has been relatively limited progress so far in automating bloom intensity estimation. Currently, this activity is typically carried out manually with the assistance of rudimentary tools. More specifically, it is generally done by visually inspecting a random sample of trees within the orchard and then extrapolating the estimates obtained from individual trees to the remainder of the orchard [148]. As the example in Figure 5.1 illustrates, obstacles that hamper this process are: 1) manual tree inspection is time-consuming and labor-intensive, which contributes to making labor responsible for more than 50% of apple production costs [149]; 2) estimation by visual inspection is characterized by large uncertainties and is prone to errors; 3) extrapolation of the results from the level of the inspected trees to the row or parcel level relies heavily on the grower's experience; and 4) inspection of a small number of trees does not provide information about the spatial variability in the orchard, although the benefits of precision agriculture practices are known to optimize fruit quality and yield. [150]

Figure 5.1: Example of image from a flower detection dataset used in this work [9] (©2018 IEEE).

These limitations, added to the short-term nature of flower appearance until petal fall, make an automated method highly desirable. Multiple computer vision systems have been proposed to solve this problem, but most of these methods rely on hand-engineered features [151], making their overall performance acceptable only under relatively controlled environments (e.g., at night with artificial illumination). Their applicability is in most cases species-specific and highly vulnerable to variations in lightning conditions, occlusions by leaves, stems, or other flowers [26].

In the last decade, deep learning approaches based on convolutional neural networks (CNNs) led to substantial improvements in the state of the art of many computer vision tasks [152]. Recent works have adapted CNN architectures to agricultural applications such as fruit quantification [125], classification of crops [153], and plant identification from leaf vein patterns [154]. Yet, as reviewed in the following Section 5.1, existing methods for fruit flower segmentation still rely on hand-engineered approaches that focus mostly on color analysis. In this context, to the best of our knowledge, our work in described in Section 5.2 and published in [9] was the first to employ CNNs for flower detection.

## 5.1 Related Work

While previous techniques for flower detection were based only on color information, methods designed for fruit quantification have exploited more modern computer vision techniques. For this reason, this section is divided into two parts: first, the most relevant works on automated flower detection are reviewed; then, a discussion of the relevant literature on fruit quantification is provided.

**Flower quantification.** Aggelopoulou and colleagues presented in [155] one of the first works using computer vision techniques to detect flowers. That method is based on color thresholding and requires image acquisition at specific daylight times, with the presence of a black cloth screen behind the trees. Thus, although its reported error in predicted yield is relatively low (18%), this approach is only applicable for such controlled scenarios.

Similar to the work of Thorp and Dierig [156] for identification of *Lesquerella* flowers, the technique described by Hočevar et al. in [157] does not require a background screen, but it is still not robust to changes in the environment. The image analysis procedure is based on hard thresholding according to color (in the HSL color space) and size features, such that parameters have to be adjusted whenever changes in illumination (daylight/night), flowering density (high/low concentration), or camera position (far/near trees) occur.

Horton and his team described in [158] a system for peach bloom intensity estimation that uses a different imaging approach. Based on the premise that the photosynthetic activity of this species increases during bloom period, the system relies on multispectral aerial images of the orchard, yielding an average detection rate of 84.3% for 20 test images. Similarly to the aforementioned methods, the applicability of this approach also has the intrinsic limitation of considering only color/spectral

information (thresholding near-infrared and blue bands). Hence, its performance is also sensitive to changes in illumination conditions.

**Fruit quantification.** While early attempts for autonomous fruit detection also relied on hand-engineered features (e.g. color, texture, shape) [26], recent works have been exploring more advanced computer vision techniques. A multi-class image segmentation system is proposed by Hung et al. in [159], classifying image pixels into leaves, almonds, trunk, ground, and sky. Their method combines sparse autoencoders [152] for feature extraction, logistic regression for label associations, and conditional random fields to model correlations between pixels.

Other methods are based on support vector machine (SVM) classifiers that use information obtained from different shape descriptors and color spaces as input [160, 161]. Compared to previous techniques for flower detection, these methods are more robust since morphological characteristics are taken into account. However, as many other shape-based and spectral-based approaches [162–165], these techniques are still limited by background clutter and variable lighting conditions in orchards [148].

Metadata information has also been recently exploited for fruit quantification. Bargoti and colleagues in [166] built on [159] to propose an approach that considers pixel positions, orchard row numbers, and the position of the sun relative to the camera. Similarly, Cheng et al. [167] propose the use of information such as fruit number, fruit area, area of apple clusters, and foliage area to improve accuracy of early yield prediction, especially in scenarios with significant occlusion. However, the inclusion of metadata is highly prone to overfitting, particularly when limited training data is available and the variability of the training set is hence low [166].

Following advances on the field of object detection, recent works adapt the Faster R-CNN model [87] for fruit detection. Bargoti and Underwood in [168] present

a Faster R-CNN trained for detection of mangoes, almonds, and apples fruits on trees. Stein et al. in [169] extend this model for tracking and localization of mangoes, combining it with a monocular multi-view tracking module that relies on a GPS system. Sa et al. in [170] applies the Faster R-CNN to RGB and near-infrared multi-modal images, where each modality is fine-tuned independently and optimal results are obtained using a late fusion approach. The method introduced in [125] for counting apples and oranges employs a fully convolutional network (FCN) to perform fruit segmentation and a convolutional network to estimate fruit count. Still in the context of agricultural applications, CNNs have been also successfully used for plant identification based on leaf vein patterns [154].

In summary, existing methods for flower identification are based on hand-engineered image processing techniques that work only under specific conditions. Color and size thresholding parameters composing these algorithms have to be readjusted in case of variations of lightning conditions, camera position with respect to the orchard (distance and angle), or expected bloom intensity. Recent techniques employed for fruit quantification exploit additional features and machine learning strategies, providing insights to further develop strategies for flower detection.

With the goal of devising a technique for flower segmentation that is robust to clutter, changes in illumination and applicable for different flower species, the present dissertation describes two novel approaches developed using CNNs for flower segmentation. The first approach, published in [9], combines superpixel-based region proposals with a classification CNN that is fine-tuned to become particularly sensitive to apple flowers. The second approach, published in [2], combines instead an end-to-end CNN with the RGR algorithm described in 3.1 to improve segmentation quality. Trained only on a dataset of apple flowers, this approach provides high-quality segmentations even in scenarios involving different acquisition conditions and flower species.

## 5.2 Apple flower detection using deep convolutional networks

Inspired by successful works using convolutional neural networks (CNNs) in multiple computer vision tasks, we propose a novel method for apple flower detection based on features extracted using a CNN.

The main contributions of this work are:

- *to the best of our knowledge, our method [9] is the first to employ CNNs for flower segmentation.* In our approach, an existing CNN trained for saliency detection is fine-tuned to become particularly sensitive to flowers. This network is then used to extract features from portraits generated by means of superpixel segmentation. After dimensionality reduction, these features are fed into a pre-trained classifier that ultimately determines whether each image region contains flowers or not.
- *our CNN-based method significantly outperforms state-of-the-art approaches*, which are based on color-based approaches;
- we provide an *extensive evaluation on a challenging dataset acquired under realistic and uncontrolled conditions, as well as an analysis of the generalization capability of the proposed approach on additional datasets previously unseen by the model.*

### 5.2.1 Proposed approach

This section first describes the prediction steps performed by the proposed method, i.e., the sequence of operations applied to an image in order to detect the presence of flowers. Subsequently, we describe the fine-tuning procedure carried out to obtain the core component of our model: a CNN highly sensitive to flowers. We conclude the section with a discussion of alternative flower detection approaches against which we evaluate our proposed method and a brief discussion of relevant details regarding the implementation of our method.

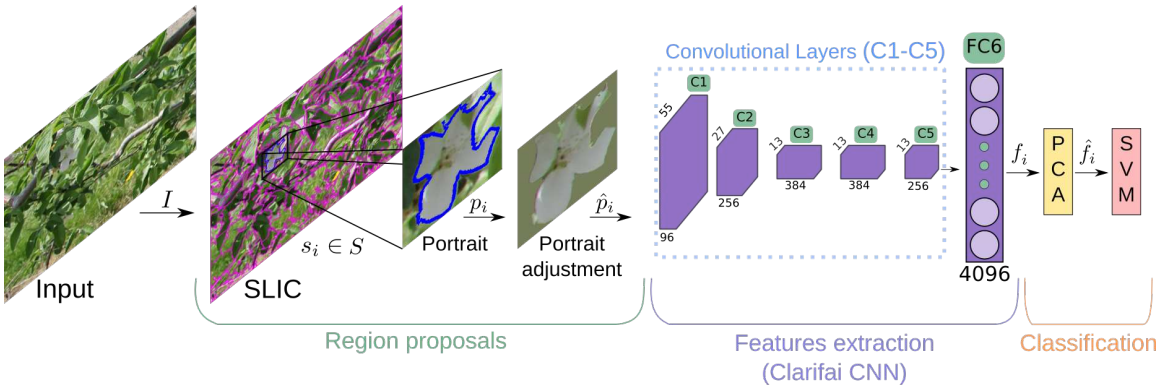



Figure 5.2: Diagram illustrating the sequence of image analysis tasks performed by the proposed model for flower identification. Layers FC7-FC8, present in the original architecture shown in Figure 2.9, are used only during fine-tuning (training). For final prediction, features are collected from the output of layer FC6. Each task and its corresponding output (shown above the arrows) are described in Algorithm 3.

In the discussion that follows, we refer to our proposed approach for flower detection as the CNN+SVM method. As illustrated in Figure 5.2, our method consists of three main steps: i) computation of region proposals; ii) feature extraction using our fine-tuned CNN, which follows the Clarifai architecture [42]; and iii) final classification of each region according to the presence of flowers. The operations that comprise these steps are described in detail below. In our description, we make reference to Algorithm 3, which lists the operations performed by our method on each input image. The sensitivity of the method to specific design choices is analyzed in Section 5.2.3.1.

*1) Step 1 - Region proposals:* The first step in the proposed method consists of generating region proposals by grouping similar nearby pixels into superpixels, which are perceptually meaningful clusters of variable size and shape (Line 1 of Algorithm 3). To this end, we use the *simple linear iterative clustering* (SLIC) superpixel algorithm [60], described in detail in Section 2.4.4. The second leftmost image in Figure 5.2

illustrates the superpixels $s_i \in S$ generated by the SLIC algorithm when applied to a typical image obtained in an orchard.

Although other approaches such as Faster R-CNN [87] provide a unified architecture in which both region proposal and classification modules can be fine-tuned for a specific task, they have more parameters that need to be learned in a supervised manner. Since in most cases flowers are salient with respect to their surrounding background, an unsupervised, local context-based approach such as superpixel segmentation should be sufficient to obtain region proposals suitable for flower detection.

**Algorithm 3** Proposed approach for flower detection

**Input:** Image $I$.
**Output:** Regions in $I$ containing flowers.
1: Partition $I$ into a set of superpixels $S$ using SLIC.
2: **for** each superpixel $s_i \in S$ **do**
3: Crop smallest square portrait $p_i$ enclosing $s_i$.
4: Generate $\hat{p}_i$ by mean-padding the background surrounding $s_i$ in $p_i$.
5: Extract features $f_i$ from the mean-centered $\hat{p}_i$ using the fine-tuned CNN.
6: Obtain $\hat{f}_i$ by performing PCA analysis on $f_i$.
7: Classify $s_i$ by applying a pre-trained SVM on $\hat{f}_i$.

Once the image is segmented into superpixels, as Algorithm 3 indicates, we iterate over each superpixel in the image. Since the input size required by the Clarifai CNN model is $227 \times 227$, we first extract the smallest square portrait enclosing the superpixel under analysis (Line 3), which we denote $p_i$. The output of this step is illustrated in the third leftmost image of Figure 5.2 for one superpixel. The background surrounding the superpixel of interest within a portrait is then padded with the training set mean, i.e., the average RGB color of all images composing the dataset (greenish color). Finally, the portrait is resized to $227 \times 227px$ (Line 4). The resulting region proposal, $\hat{p}_i$, is illustrated in the fourth image of Figure 5.2.

*2) Step 2 - Feature extraction:* In the feature extraction step (Line 5), each of the portraits generated above is mean-centered and then evaluated individually by our CNN. The mean-centering step consists of subtracting from the portrait the same average training set RGB mean used for padding its background. This procedure is commonly employed to facilitate training convergence of deep learning models, since it ensures similarly ranged features within the network. For each input portrait, we collect as features the output of the rectified linear unit (ReLU) associated with the first fully connected layer of the network (FC6). With a dimensionality of $N = 4,096$, the feature vector $f_i \in \mathbb{R}^N$ collected at this stage of the network encapsulates the hierarchical features extracted by layers $C1 - C5$, which contain the information required for accurate classification.

*3) Step 3 - Classification:* To classify each proposed region as containing a flower or not, we first perform principal component analysis (PCA) to reduce the feature dimensionality to a value $k < N$ such that the new feature vector $\hat{f}_i \in \mathbb{R}^k$ (Line 6). As demonstrated in our experimental evaluation in Section 5.2.3.1 a value of $k = 69$, which corresponds to approximately 94% of the original variance of the data, provides performance levels virtually identical to those of the original features. Finally, based on these features, a pre-trained SVM model classifies superpixels according to the presence of flowers (Line 7). Details on SVM training are provided in the next section.

#### 5.2.1.1 Network fine-tuning and SVM training

Based on the techniques introduced by Girshick et al. in [86] and Zhao et al. in [171] for object and saliency detection, in our model an existing CNN architecture is made particularly sensitive to flowers by means of fine-tuning. In the work of Zhao et al. [171], the Clarifai model [42] was adopted as the starting point and fine-tuned

for saliency detection. We further tuned Zhao et al.'s model for flower identification using labeled portraits from our training set, which we describe below.

The generation of training samples for network tuning takes place in a manner similar to that used for prediction. For each labeled image composing the training set, we compute region proposals according to Step 1 described above. Using these training examples, $10,000$ backpropagation training iterations are performed in order to minimize the network classification error. After fine-tuning, we compute the CNN features of the training examples, reduce their dimensionality to $k = 69$, and use them to train the SVM classifier.

**Image dataset**

Images of apple trees were collected using a camera model Canon EOS 60D under natural daylight illumination (i.e., uncontrolled environment). This dataset, which we refer to as *AppleA*, is composed of a total of 147 images with resolution of $5184 \times 3456$ pixels acquired under multiple angles and distances of capture. Figure 5.3 shows some images that comprise this dataset. For performance evaluation and learning purposes, the entire dataset was labeled using a MATLAB GUI in which the user selected only superpixels that contain parts of flowers in, at least, approximately half of its total area. As summarized in Table 5.1, the labeled images were randomly split into training and validation sets composed of 100 and 47 images, respectively. This corresponds to a total of $91,488$ training portraits (i.e. superpixels) and $42,430$ validation ones. The training examples were used to fine-tune the network and train the SVM, while the validation examples were used in the performance evaluation discussed in Sections 5.2.3.1 and 5.2.3.2.

True Positives False Negatives False Positives

Figure 5.3: Examples of images composing the *AppleA* dataset, with the corresponding detections provided by the proposed algorithm.

Table 5.1: Statistics of the training and validation dataset (*AppleA*).

| | | **Portraits (i.e., superpixels)** | | |
|---|---|---|---|---|
| | **Images** | Positives | Negatives | Total |
| Training | 100 | 3,691 (4%) | 87,797 (96%) | 91,488 |
| Validation | 47 | 1,719 (4%) | 40,711 (96%) | 42,430 |
| *Total* | 147 | 5,410 (4%) | 128,508 (96%) | 133,918 |

### Data augmentation

According to our labeling, only 4% of the samples contain flowers (positives). Imbalanced datasets represent a problem for supervised machine learning approaches, since overall accuracy measures become biased towards recognizing mostly the majority class [172]. In our case, that means the learner would present a bias towards classifying the portraits as negatives. To overcome this situation and increase the amount of training data, we quadrupled the number of positive samples using data augmentation. As illustrated in Figure 5.4, this was accomplished by mirroring each positive sample with respect to: (i) the vertical axis, (ii) the horizontal axis, and (iii) both axes.

(a) (b) (c) (d)

Figure 5.4: Example of data augmentation. a) Original portrait. b) Portrait mirrored with respect to the vertical axis, c) the horizontal axis, d) and both axes.

#### Parameters' optimization

As reviewed in Section 2.2.1.1, support vector machines (SVMs) are supervised learning models that search for a hyperplane that maximizes the margin distance to each class. To optimize the regularization cost $C$ and the width of the Gaussian kernel $\gamma$, we use the grid search strategies described in [37, 38] and Section 2.2.1.1.

#### Implementation Details

Most image analysis tasks were performed using MATLAB® R2016b. Additionally, we used the open source Caffe framework [173] for fine-tuning and extracting features from the CNN. To reduce computation times by exploiting GPUs, we used the cuSVM software package for SVM training and prediction [174].

### 5.2.2 Comparison approaches

As has been noted in Section 5.1, current algorithms for automated identification of flowers are mostly based on binarization by thresholding information from different color-spaces (typically RGB or HSV) [155, 156], occasionally combined with size filtering [157]. We implement three alternative baseline approaches which reflect the state of the art in fruit/flower detection. The first approach, which we call the

HSV method, replicates the algorithm described by Hočevar and his team in [157]. Images are binarized at pixel-level based on HSV color information, followed by filtering according to minimum and maximum cluster sizes.

We refer to the second baseline implementation as HSV+Bh. Similar to our proposed approach, the starting point for this method is the generation of superpixels using the SLIC algorithm. We then compute a 100-bin histogram of each superpixel in the HSV color space, detailed in Section 2.1. In our experiments, we construct a single 1-D histogram consisting of 100 bins, which corresponds to the concatenation of a 50-bin hue channel histogram, a 40-bin saturation histogram and a 10-bin value histogram. Afterwards, we use the Bhattacharyya distance [175] to compare each superpixel histogram against the average histogram of all positive samples composing the training set. We compute the Bhattacharyya distance using a Gaussian kernel function, as formulated in [176, 177]. The average Bhattacharyya distance is taken as the likelihood that the superpixel includes a flower, and superpixels with distance lower than a threshold are classified as flowers.

Since the technique described above is based on the average Bhattacharyya distance in the HSV color space, it makes no distinction between poorly and highly informative training sample features. Its ability to make accurate classification decisions is therefore limited in such complex feature spaces. Inspired by works on fruit quantification [160, 161], we extend this method by combining the same HSV histograms with an SVM classifier for apple flower detection. That is, rather than determining whether a superpixel contains a flower based on the Bhattacharyya distance, we train an SVM classifier that uses the HSV histograms as inputs. We call this approach the HSV+SVM method.

### 5.2.3 Experiments and results

Experiments were performed with three main goals. Our optimal CNN+SVM model extracts features from the CNN's first fully connected layer ($FC6$), reduces feature dimensionality to 69, and performs final classification using SVM. Thus, we first evaluated the impact of these specific design choices on the final performance of our method. We then compared it against the three baseline methods (HSV, HSV+Bh and HSV+SVM). Finally, we evaluated the performance of the proposed approach on previously unseen datasets to determine its generalization capability.

As described in Section 5.2.1.1, our datasets are severely imbalanced. We therefore perform our analysis in terms of precision-recall curves (PR) and the corresponding $F_1$ score [51], since as discussed in Section 2.4.2 such metrics are more robust to imbalance than simple accuracy computation. While the maximum $F_1$ score indicates the optimal performance of a classifier, the area under the respective PR curve (AUC-PR) corresponds to its expected performance across a range of decision thresholds, such that a model with higher AUC-PR is more likely to generalize better.

#### 5.2.3.1 Analysis of design choices

In order to validate our design choices, we performed experiments to evaluate how the final performance of the classifier is affected by: (i) the dimensionality of the feature space, (ii) the point where features are collected from the CNN, and (iii) the type of input portrait.

**Dimensionality analysis**

As reviewed in Section 2.2.1.1, PCA is one of the most widespread techniques for dimensionality reduction, projecting $N$-dimensional input data onto a $k$-dimensional subspace in such a way that this projection minimizes the reconstruction

error [33]. In this application, the original dimensionality corresponds to the number of elements in the CNN vectors extracted from a fully connected layer, i.e., $N = 4,096$ as represented for the last layer in Figure 5.2. The first two columns of Table 5.2 show the reduced dimensionality $k$ of the feature vector and the corresponding ratio of the total variance of the $N$-dimensional dataset that is retained at that dimensionality for layer $FC6$. As the table indicates, the first most significant dimension alone already covers almost half of the total variance, and 23 dimensions are sufficient to cover nearly 90% of it.

Table 5.2: Classification performance according to the number of principal components (dimensions) selected after applying PCA to the extracted features. Best results in terms of $F_1$ and AUC-PR are shown in boldface.

| **No. of dimensions** | **Variance ratio** | $\mathbf{F_1}$ | **Recall** | **Precision** | **AUC-PR** |
|---|---|---|---|---|---|
| 1 | 48.3% | 90.4% | 92.2% | 88.6% | 96.5% |
| 2 | 63.7% | 91.4% | 92.7% | 90.2% | 94.0% |
| 5 | 79.9% | 91.9% | 92.3% | 91.5% | 96.9% |
| 15 | 87.4% | 91.5% | 92.6% | 90.4% | 94.3% |
| 23 | 89.6% | **92.1**% | 92.9% | 91.2% | 95.2% |
| 69 | 93.8% | 91.9% | 92.6% | 91.2% | **97.2**% |
| 150 | 95.8% | 91.3% | 92.7% | 90.0% | 97.1% |
| 300 | 97.2% | 91.6% | 91.6% | 91.7% | 97.2% |
| 500 | 98.0% | 91.8% | 91.8% | 91.8% | 95.0% |
| 1080 | 99.0% | 91.7% | 91.5% | 91.8% | 94.9% |

We investigated then how samples are mapped into the lower dimensional feature space. Figure 5.5 shows the projections in dimensions 1 and 2 as well as dimensions 1 and 3. These plots illustrate how the convolutional network maps the samples into a space where it is possible to differentiate between multiple clusters.

With *dim.* as an abbreviation for *dimension*, let ↓ denote low dimensionality values and ↑ high values, respectively. The following clusters can be identified: flowers (↓ dim.1, ↑ dim.2); grass/floor (↑ dim.1, ↑ dim.2); branches/leaves (↓ dim.2); sky (↑ dim.3). This indicates that positive and negative samples are almost linearly separable even for $2D$ projections of the original feature space.

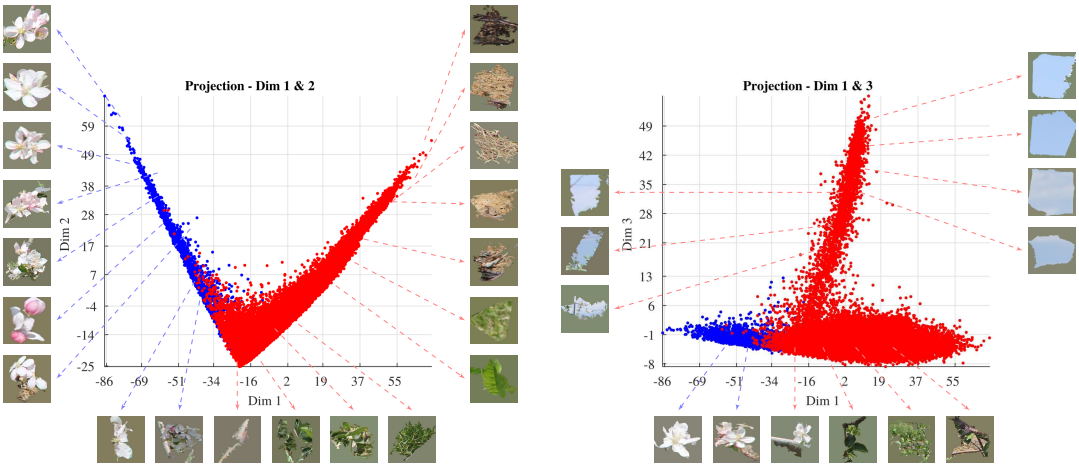



Figure 5.5: Projections of samples on 2D feature spaces, with positive samples in blue and negatives ones in red. *Left:* sample distribution on the plane corresponding to dimensions 1 and 2 according to PCA. *Right:* sample distribution on the plane corresponding to dimensions 1 and 3.

To quantitatively assess how the classification performance is affected by the dimensionality of the feature space, we trained SVM classifiers for different numbers of dimensions. For each dimensionality, Table 5.2 presents the optimal performance metrics and the corresponding AUC-PR. As expected, these results demonstrate that the impact of dimensionality on the optimal performance of our method is rather low. A very good performance is already obtained using a 2D feature space, with both $F_1$ score and AUC-PR only around 0.7% and 3.2% lower than the highest obtained values, respectively. In terms of optimal recall and precision, this is equivalent to missing

four positive samples out of $1,719$, while including 19 additional false-positives out of $40,711$. Moreover, the table shows that a dimensionality of 69 is nearly optimal: the performance in terms of optimal $F_1$ score is only 0.2% lower than the highest value obtained (23 dimensions) and it is optimal in terms of AUC-PR.

Although in the discussion above we present results obtained using SVMs, such a high separability even for low dimensionalities indicates that the final prediction accuracy of our model is almost independent of the type of classifier employed. This conjecture is validated in the next subsection, where we demonstrate that the performance of our system does not change significantly by either including an additional fully connected layer to our CNN or by carrying out classification using the using network's softmax layer directly.

**Feature analysis**

As explained in Section 5.2.1, after fine-tuning the model, we use it to extract features that allow the classification of superpixels according to the presence of flowers within them. Three combinations of features and classification mechanisms were investigated: (A) predict using solely the neural network, by means of its softmax output layer, (B) train an SVM classifier on features collected after the last fully connected layer (FC7), (C) train an SVM classifier on features collected after the first fully connected layer (FC6). Figure 5.6 shows the points where features are collected and how classification scores are computed using these features. Following the notation used in Figure 5.2, C1-C5 correspond to the convolutional layers of the fine-tuned Clarifai network, FC6-FC7 are the fully connected layers, and FC8 is the softmax layer.

For approaches B and C, features are collected from the output of the rectified linear units (ReLUs) located right after the respective fully connected layers. The same sequence of operations is performed for both methods B and C, i.e., the frame-

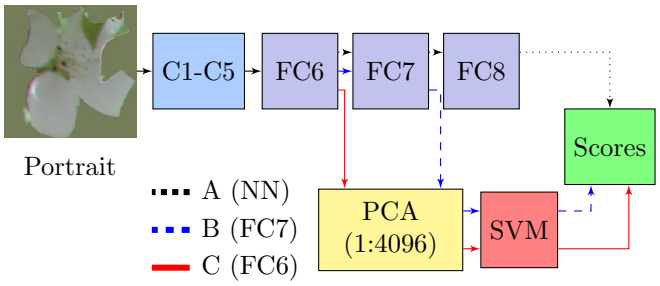



Figure 5.6: Diagram illustrating how classification scores are computed using the extracted features.

work is the same regardless of whether the features are collected from the last (FC7) or first fully connected layer (FC6). Based on the results obtained in the previous section, for both cases 69 dimensions are kept after PCA analysis.

Results obtained for classification on the validation set are summarized in Table 5.3 and Figure 5.7. As Figure 5.7 indicates, all three approaches show very similar performance. A closer inspection of Table 5.3 reveals that the SVM-based approaches slightly outperform the direct use of the neural network softmax layer both in terms of optimal $F_1$ score and AUC-PR. The performances obtained with methods B and C are very similar for both metrics. We therefore opted for method C, which uses features extracted from the earlier layer FC6 and provides slight increases in both optimal $F_1$ score and AUC-PR.

Table 5.3: Classification performance according to the CNN layer at which features are collected - Methods A, B, C.

| | **AUC-PR** | **$F_1$** | **Recall** | **Precision** |
|---|---|---|---|---|
| A (NN) | 96.9% | 90.6% | 91.7% | 89.6% |
| B (FC7) | 97.2% | 91.6% | 91.8% | 91.4% |
| C (FC6) | 97.3% | 91.9% | 92.6% | 91.2% |

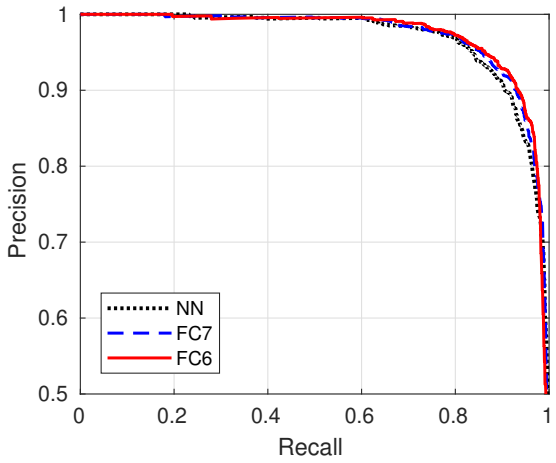



Figure 5.7: PR curves illustrating the performance on the validation set according to the CNN layer at which features are collected. *NN* stands for prediction using solely the network softmax output layer, while *FC6* and *FC7* correspond to SVM classifiers trained on features collected at the first and second fully connected layers, respectively.

### Different types of portraits

Using superpixels for region proposal computation and subsequent generation of portraits implies that our goal is to evaluate whether the superpixel itself is composed of flowers or not. In order to assess the influence of the local context surrounding the superpixel on the classification results, in addition to the approach based on replacing the region around the superpixel with the mean RGB value, two alternative approaches for portrait generation were considered. The first consists of retaining the unmodified image area surrounding the superpixel, whereas the second corresponds to blurring the background surrounding the superpixel with a low-pass filter. For all three cases, the portrait is mean-centered before being fed into the neural network. The three types of evaluated portraits are illustrated in Figure 5.8.

(a) (b) (c)

Figure 5.8: Example of the three types of portrait evaluated. a) *Original*; b) Blurred background (*Blur*); c) Mean padded background.

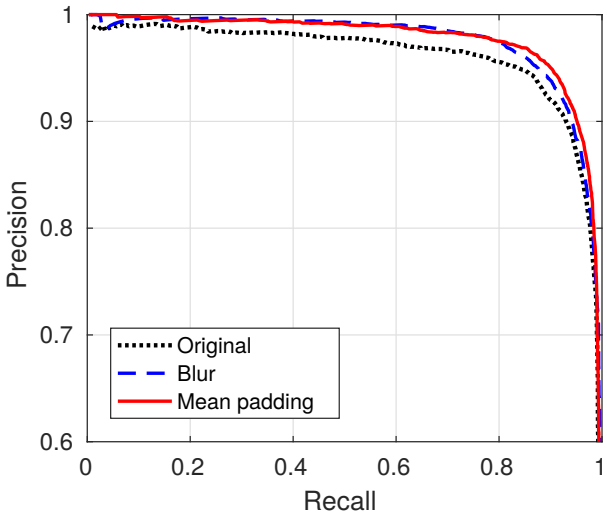



Figure 5.9: Classification performance according to the portrait adjustment strategy. *Original:* portraits evaluated without any further adjustment; *Blur:* portraits where the background is blurred; *Mean padding:* strategy of padding the background with the training set mean.

Figure 5.9 shows the PR curves obtained for each portrait type. The best performance is obtained with mean-padded portraits, a behavior explained by the existence of cases such as the ones illustrated in Figure 5.10. The superpixels highlighted in the images on the top row do not contain flowers in more than 50% of their area and should therefore not be classified as flowers. However, these superpixels are surrounded by flowers, as depicted in the corresponding figures in the bottom row,

and hence the approach of simply cropping a square region around the superpixel leads to cases in which the portrait contains a well-defined flower. As a consequence, features extracted from the CNN for the entire portrait will indicate the presence of flowers and therefore lead to high confidence false positives, which explain the non-maximal precision values in the upper-left part of the respective PR curve. This problem is eliminated by mean-padding the background.

Figure 5.10: Examples of superpixels incorrectly classified for *Original* and *Blur* portraits. The superpixels are shown in the top row and the bottom row shows the entire portraits enclosing the superpixels.

#### 5.2.3.2 Comparison against baseline methods

The analysis in Section 5.2.3.1 above validates the design choices of our optimal CNN+SVM model described in Section 5.2.1. That is, our optimal model uses mean-padded portraits and 69-dimensional features obtained from the FC6 layer of the CNN. In this section, we compare this optimal CNN+SVM model against the three baseline methods described in Section 5.2.2.

The parameters of all four methods were optimized using a grid search, as described in Section 5.2.1.1. Optimization of the SVM hyperparameters based on $F_1$

score resulted in the following values for regularization factor ($C$) and RBF kernel bandwidth ($\gamma$): HSV+SVM ($C = 180; \gamma = 10$); CNN+SVM ($C = 30$; $\gamma = 10^{-4}$). For the HSV+BH method, we performed an analogous grid search to optimize the standard deviation associated with the Gaussian kernel function, obtaining an optimal parameter of $\sigma = 5$. For the HSV method, we performed an extensive grid search on our training dataset to select an optimal set of threshold values. This procedure indicated that pixels composing flowers are distributed over the entire H range of $[0, 255]$, with optimal ranges of S within $[0, 32]$, V within $[139, 255]$, minimum size of $1,200$ pixels and maximum size of $45,000$ pixels.

Once the optimal parameters for all the classification models were determined, we evaluated the overall performance of each method using 10-fold cross-validation. All the $133,918$ samples composing the full *AppleA* dataset were combined and divided into 10 folds containing $13,391$ samples each. A total of 10 iterations was performed, in which each subsample was used exactly once as validation data.

The PR curves associated with each method are shown in Figure 5.11. Table 5.4 provides the AUC-PR for each method along with the metrics obtained for the optimal models as determined by the $F_1$ score.

Table 5.4: Summary of results obtained for our approach (CNN+SVM) and the three baseline methods (HSV, HSV+BH and HSV+SVM). Best results are shown in boldface.

| | **AUC-PR** | **$F_1$** | **Recall** | **Precision** |
|---|---|---|---|---|
| HSV | 54.9% | 54.1% | 58.3% | 50.4% |
| HSV+BH | 61.6% | 64.6% | 56.9% | 60.5% |
| HSV+SVM | 92.9% | 87.1% | 88.4% | 87.8% |
| CNN+SVM | **97.7%** | **93.4%** | **92.0%** | **92.7%** |

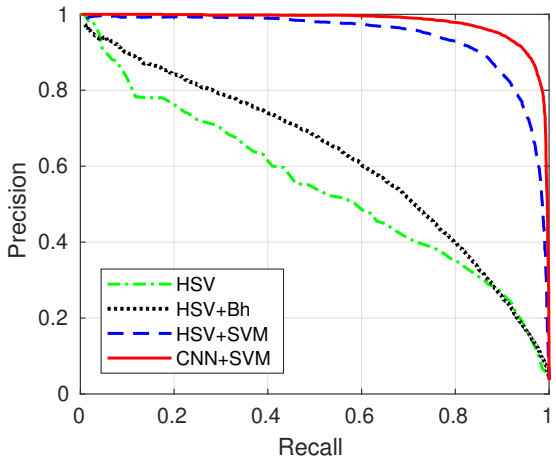



Figure 5.11: Precision-recall (PR) curve illustrating the performance of our proposed approach (CNN+SVM) in comparison with the three baseline methods (HSV, HSV+Bh, and HSV+SVM).

The HSV method, which closely replicates existing approaches for flower detection, performs poorly in terms of both recall and precision. Such low performance is expected for methods that rely solely on color information. Since these techniques do not consider morphology or higher-level context to characterize flowers, they are very sensitive to changes in illumination and to clutter. Small performance improvements are obtained using the HSV+Bh method, which replaces pixel-wise hard thresholding by HSV histogram analysis at the superpixel level, thereby incorporating a limited amount of context information into its classification decisions.

As illustrated by the results obtained with the HSV+SVM method, the use of an SVM classifier on the same HSV color features leads to dramatic improvements in both $F_1$ and AUC-PR ratios (around 20% and 30%, respectively). Rather than giving the same importance to all histogram regions, the SVM classifier is capable of distinguishing between poorly and highly informative features. However, as depicted in Figure 5.12, the precision of this method is still compromised by gross errors such

as classifying parts of tree branches as flowers, since it does not take into account any morphological information.

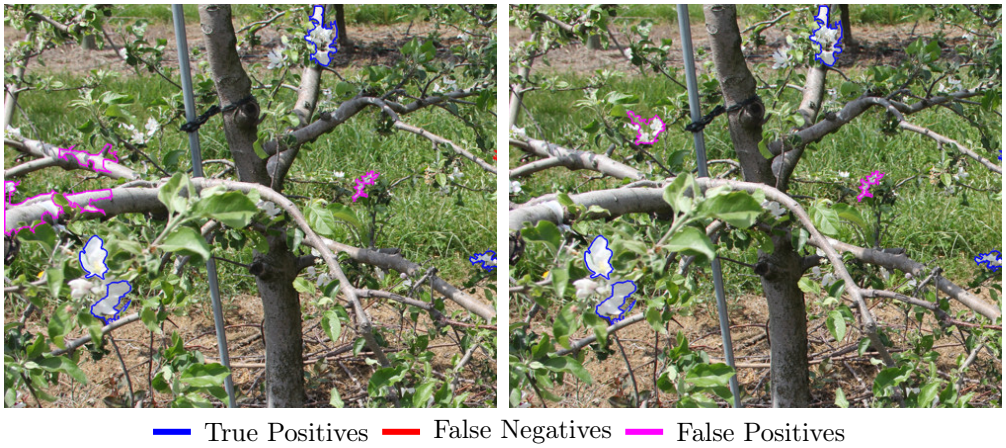



Figure 5.12: Example of classification results obtained using (left) the baseline HSV+SVM method and (right) our proposed CNN+SVM method. Some examples of false positives generated by the HSV+SVM method that our approach correctly classifies can be seen on the branches near the left border of the image.

The proposed approach (CNN+SVM) outperforms both baseline methods by extracting features using a convolutional neural network. Differently from the previous methods, the hierarchical features evaluated within the CNN take into account not only color but also morphological/spatial characteristics from each superpixel. Our results demonstrate the effectiveness of this approach, with significant improvements in both recall and precision which culminate in an optimal $F_1$ score higher than 92% and AUC-PR above 97% for the evaluated dataset. Figure 5.3 shows examples of the final classification yielded by this method.

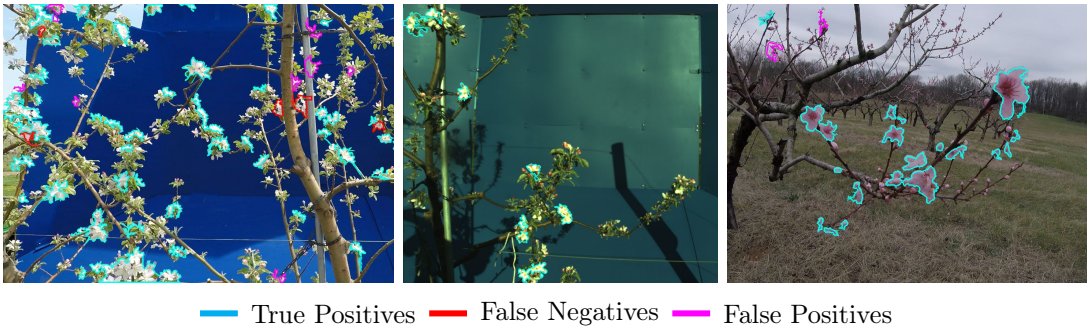



Figure 5.13: Examples of images composing the additional datasets *AppleB* (left), *AppleC* (middle) and *Peach* (right), overlaid with the corresponding detections obtained by our method.

#### 5.2.3.3 Performance on additional datasets

To evaluate the generalization capability of our method, we assessed its performance on three additional datasets, composed of 20 images each and illustrated in Figure 5.13. We compare the results of our method with the performance of the best performing baseline approach (HSV+SVM). No dataset-specific adjustment of parameters is performed for our method nor for the baseline, i.e., both methods are assessed with the same optimal configuration obtained for the *AppleA* dataset.

Two of the additional datasets also correspond to apple trees, but with a blue background panel positioned behind the trees to visually separate them from other rows of the orchard, a common practice in agricultural vision systems. We denote the first dataset *AppleB*, which is composed of images with resolution $2704 \times 1520px$ acquired using a camera model GoPro HERO5. In this dataset there is a substantial number of occlusions between branches, leaves and flowers.

The second dataset, which we call *AppleC*, is composed of images with resolution $2456 \times 2058px$ acquired with a camera model JAI BB-500GE. In this dataset occlusions are less frequent but the saturation color component of the images is concentrated in a much narrower range of the spectrum than in the original *AppleA*

dataset. The contrast between objects such as flowers and leaves is therefore significantly lower.

The third additional dataset contains images of peach flowers (we therefore call it *Peach*) with resolution $2704 \times 1520px$ acquired using a camera model GoPro HERO5. Peach blossoms show a noticeable pink hue in comparison to the mostly white apple flowers composing the training dataset. Additionally, images were acquired during an overcast day, such that in comparison to the training set (*AppleA*) the illumination is lower and the sky composing the background is gray instead of blue. Although the main scope of this work is on apple flower detection, we ultimately aim at a highly generalizable system that can be applied by fruit growers of different crops without the need for species-specific adjustments. In fruit orchards, each species of tree is typically constrained to specific areas. Hence, rather than differentiating between flower species, it is preferable to have a system that can distinguish between flowers and non-flower elements (e.g. leaves, branches, sky) regardless of species. Thus, this dataset represents a good evaluation of detection robustness.

**Transfer learning steps**

For all three additional datasets, both feature extraction and final classification were performed using the same parameters obtained by training with the *AppleA* dataset, without any dataset specific fine-tuning. Our transfer learning strategy relies solely on generic pre-processing operations that approximate the characteristics of the previously unseen images to those of the training samples.

Our first pre-processing step consists of removing the different backgrounds of the additional datasets. Whether the background is composed of a blue panel (*AppleB* and *AppleC*) or a gray sky (*Peach*), background identification for subsequent subtraction can be performed by means of texture analysis. For each image, we compute the corresponding local entropy, which is then binarized using Otsu's threshold [178]

to identify low texture clusters. We then apply morphological size filtering to the binarized image and model the background as a multimodal distribution, following a similar procedure described in [179].

To model the background, we compute the mean of the R, G, and B channels of the $m$ largest (in terms of number of pixels) low texture clusters to build a $m$-modal reference set. The likelihood that remaining low texture clusters belong to the background is estimated as the Euclidean distance between their means and the nearest reference in the RGB space. This metric allows differentiating between low texture components composing the background from the ones composing flowers, without any dataset specific color thresholding. For the *AppleB* and *Peach* datasets, we adopted a bimodal distribution, where the modes correspond to the blue panel/gray sky and trunk/branches. Since the blue panel in the background of images composing the *AppleC* dataset is reflective, shadows are visible and therefore we included a third mode to automatically filter these undesired elements out. Automatically determining the number of background components is part of our future work.

Afterwards, histogram equalization and histogram matching are performed on the saturation channel of each image, as exemplified in Figure 5.14. Finally, to mitigate the effects of illumination discrepancies, we subtract the difference between the mean of the value channel components in the input image and in the training set.

Figure 5.15 shows the PR curves summarizing the performance on these datasets of our method (CNN+SVM) in comparison with the best performing baseline approach (HSV+SVM). The proposed method provides AUC-PR above 85% for all datasets, significantly outperforming the baseline method. Since the HSV+SVM method relies solely on color information, its results are acceptable only for the *AppleB* dataset, the one that most closely resembles the training dataset. Its performance is notably poor for the *Peach* set, as this species differs to a great extent from apple

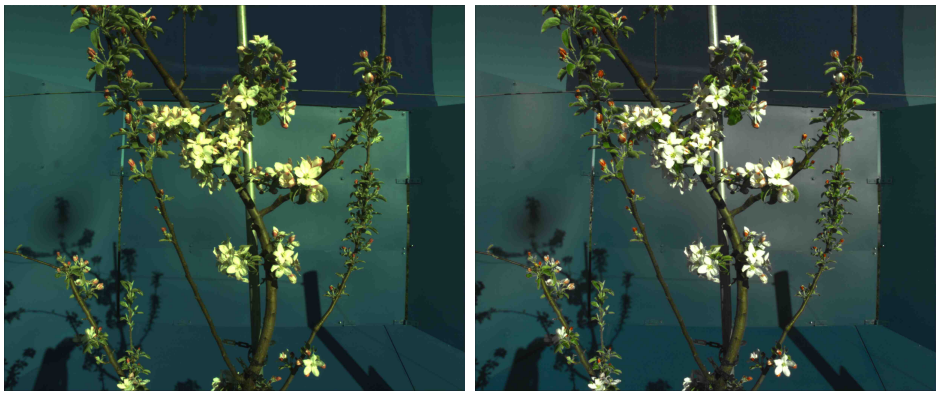


Figure 5.14: Example of image before (left) and after (right) histogram adjustment through matching and equalization.

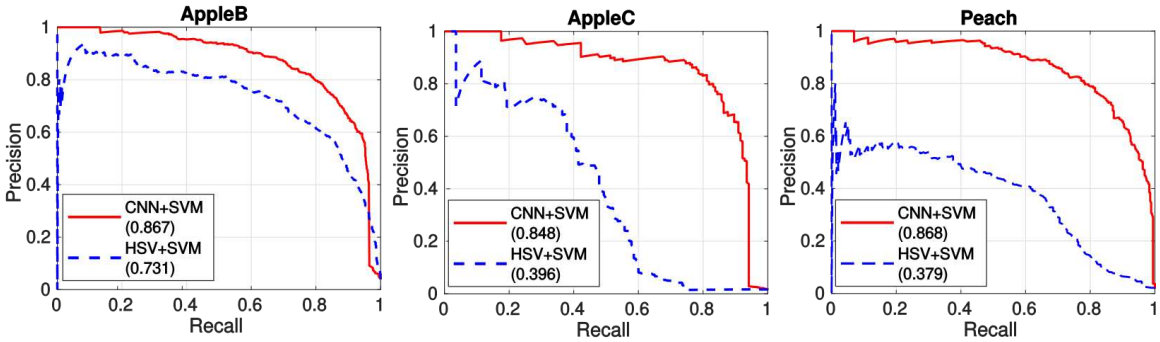



Figure 5.15: PR curves expressing the performance of our method (CNN+SVM) and the optimal baseline (HSV+SVM) approach on the three additional datasets. The AUC-PR values associated with each curve are presented within parentheses.

flowers in terms of color. A large performance difference is also evident for the *AppleC* dataset, in which flowers and leaves share more similar color components than in the training set. Table 5.5 shows that the proposed approach also outperforms the baseline by a large margin in terms of optimal $F_1$ score and the corresponding precision and recall values.

Additionally, it is noteworthy that a large number of superpixels classified as false positives by our proposed approach (CNN+SVM) correspond to regions where flowers are indeed present, but compose less than 50% of the corresponding superpixel

Table 5.5: Summary of results obtained for our approach (CNN+SVM) and the best baseline method (HSV+SVM) for the three additional datasets. Best results in terms of $F_1$ are shown in boldface.

| | | $\mathbf{F_1}$ | **Recall** | **Precision** |
|---|---|---|---|---|
| *AppleB* | HSV+SVM | 70.7% | 69.8% | 71.6% |
| | CNN+SVM | **80.2**% | 81.9% | 78.5% |
| *AppleC* | HSV+SVM | 48.6% | 37.9% | 68.0% |
| | CNN+SVM | **82.2**% | 81.2% | 83.3% |
| *Peach* | HSV+SVM | 49.0% | 61.3% | 40.8% |
| | CNN+SVM | **79.9**% | 81.5% | 78.3% |

total area. This is illustrated in Figure 5.16, which contains examples for the three additional datasets. In other words, the sensitivity of the feature extractor to the presence of flowers is very high and the final performance would be improved if the region proposals were more accurate.

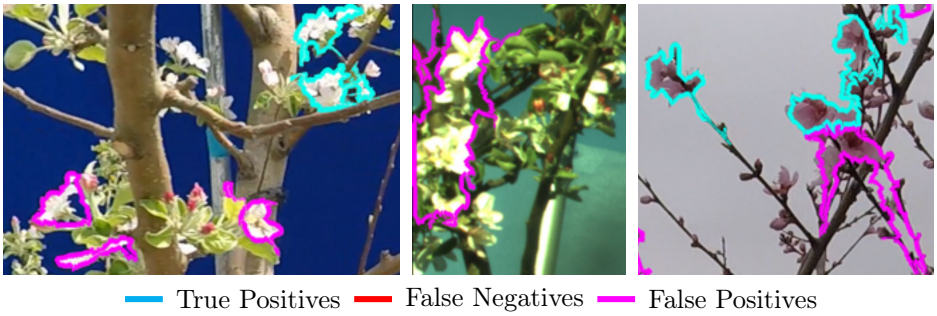



Figure 5.16: Example of incorrect detections caused by poor superpixel segmentation.

## 5.3 Multispecies fruit flower detection using a refined semantic segmentation network

The method described in the previous Section 5.2 and published in [9] combines superpixel-based region proposals with a classification network to detect apple flowers. Although that method significantly outperforms color-based approaches, existing superpixel algorithms rely solely on local context information, representing the main source of mistakes in scenarios where flowers and the surrounding background present similar colors.

As described in Section 2.4.5, end-to-end fully convolutional networks [90] have been replacing traditional fully connected architectures for image segmentation tasks [91]. The work described in this section exploits these approaches to design an improved method for automated flower segmentation, with contributions that can be summarized as:

- *A novel technique for flower identification that is i) automated, ii) robust to clutter and changes in illumination, and iii) generalizable to multiple species.* Using as starting point a fully convolutional network (FCN) [22] pre-trained on a large multi-class dataset, we describe an effective fine-tuning procedure that adapts this model for flower segmentation. To increase the segmentation quality in terms of adherence to actual flowers' boundaries, we exploit the RGR algorithm introduced in Section 3.1 for segmentation refinement. Our final method evaluates in less than 50 seconds high-resolution images each of which covers a full tree. Although the task comparison is not one-to-one, human workers may need on average up to 50 minutes to count the number of flowers per tree.
- *A feasible procedure for evaluating high-resolution images with deep FCNs on commercial GPUs.* Fully convolutional computations require GPU memory space that increases exponentially according to image resolution. We employ

an image partitioning mechanism with partially overlapping windows, which reduces artifacts introduced by artificial boundaries when evaluating disjoint image regions.

- *Release of an annotated dataset with pixel-accurate labels for flower segmentation on high resolution images [27].* We believe this can greatly benefit the community, since annotating images at a pixel level is a very time consuming yet critical task for both training and evaluation of segmentation models.

### 5.3.1 Proposed approach

In this section, we first describe the pre-training and fine-tuning procedures carried out to obtain a CNN highly sensitive to flowers. Subsequently, we describe the sequence of operations that our pipeline performs to segment flowers in an image.

#### 5.3.1.1 Network training

Reviewed in Section 2.4.1, the COCO-Stuff dataset [6] includes pixel-level annotations of classes such as *grass, leaves, tree* and *flowers*, which are relevant for our application. In the same work, the authors also discuss the performance of modern semantic segmentation methods on COCO-Stuff, with a DeepLab-based model outperforming the standard FCN. Thus, we opted for the publicly available DeepLabV2(ResNet) [22] model pre-trained on the COCO-Stuff dataset as the starting point for our pipeline. Rather than fine-tuning the DenseCRF model used in the original DeepLab work, we use our generic RGR algorithm as a post-processing module to obtain fine-grained segmentations.

The base model was originally designed for segmentation within the 172 COCO-Stuff classes. To adapt its architecture for our binary flower segmentation task, we perform procedures known as *network surgery* and *fine-tuning* [86]. The surgery procedure is analogous to the pruning of undesired branches in trees: out of

the original 172 classification branches, we preserve only the weights and connections responsible for the segmentation of classes of interest.

We considered first an architecture preserving only the *flower* classification branch, followed by a sigmoid classification unit. However, without the normalization induced by the model's original softmax layer, the scores generated by the transferred *flower* branch are unbounded and the final sigmoid easily saturates. To alleviate the learning difficulties caused by such a poor initialization, we opted for tuning a model with two-branches, under the hypothesis that a second branch would allow the network to learn a background representation that properly normalizes the predictions generated by the foreground (*flower*) branch.

We have observed experimentally that nearby leaves represent one of the main sources of misclassification for flower segmentation. Moreover, predictions for the class *leaf* presented the highest activations when applying the pre-trained model to our training dataset. For these reasons, we chose for this branch together with the one associated with *flowers* to initialize our two-branch flower segmentation network.

The adapted architecture was then fine-tuned using the training set described in Section 5.3.2, which contains 100 images of apple trees. For our experiments, the procedure was carried out for $10,000$ iterations using the Caffe framework [173], with an initial learning rate of $10^{-4}$ that polynomially decays according to $10^{-4} \times (1 - i/10000)^{0.9}$, where $i$ is the iteration number. To achieve robustness to scale variations, our fine-tuning procedure employs the same strategy used for model pre-training, where each training portrait is evaluated at (0.50, 0.75, 1.00, 1.25, 1.50) times its original resolution.

While the validation set has pixel-accurate annotations obtained using the procedure described in Section 5.3.2, the training set was annotated using the less precise but quicker superpixel-based procedure described in our previous work [9]. Moreover, we employed the same data augmentation procedure described in Section

5.2.1. Following the original network parameterization, we split the 100 training images into portraits of $321 \times 321$ pixels, corresponding to a total of $52,644$ training portraits after augmentation.

#### 5.3.1.2 Segmentation pipeline

The method we propose for fruit flower segmentation consists of three main operations: 1) divide a high resolution image into smaller patches, in a sliding window manner; 2) evaluate each patch using our fine-tuned CNN; 3) apply the refinement algorithm on the obtained scoremaps to compute the final segmentation mask. These steps are described in detail below. In our description, we make reference to Algorithm 4 and Figure 5.17.

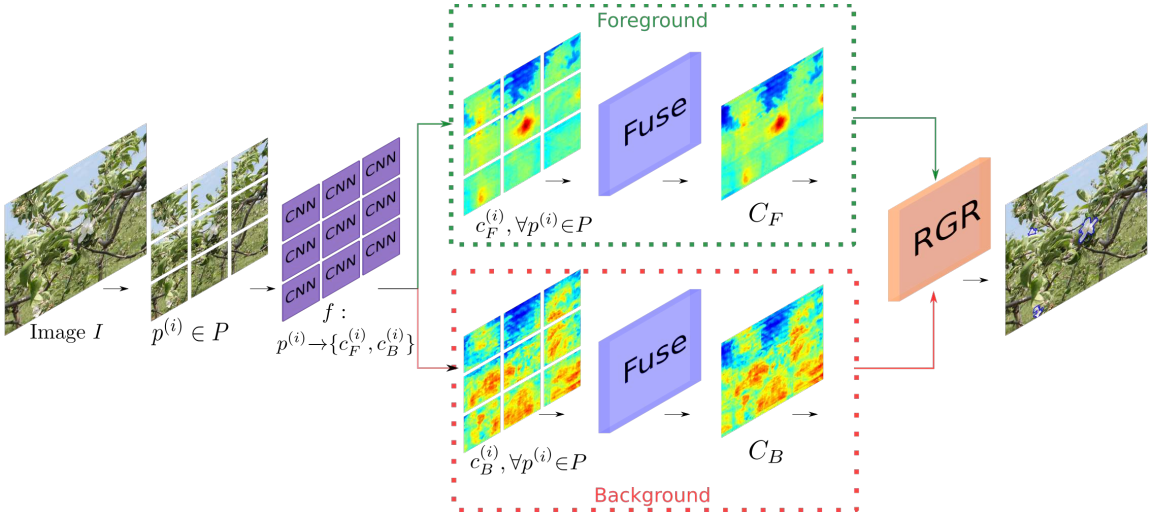



Figure 5.17: Diagram illustrating the sequence of tasks performed by the proposed method for flower detection. Each task and its corresponding output (shown below the arrows) are described in Algorithm 4. In the heatmaps, blue is associated with lower scores, while higher scores are illustrated with red [2] (©2018 IEEE).

**Algorithm 4** Proposed approach for flower detection

**Input:** Image $I$.
**Output:** Estimated flower segmentation map $\hat{Y}$ of image $I$.
1: Sliding window: divide $I$ into a set of $n$ portraits $P$.
2: **for** each portrait $p^{(i)} \in P$ **do**
3: Compute scoremaps $c_B^{(i)}$ and $c_F^{(i)}$ using the fine-tuned CNN
4: Obtain $C_B$ and $C_F$ by fusing $c_B^{(i)}$ and $c_F^{(i)}$ $(i = 1, \ldots, n)$, respectively according to Eq. 5.2.
5: Normalize $C_B$ and $C_F$ into $\tilde{C}_B$ and $\tilde{C}_F$, respectively according to Eq. 5.3.
6: Generate $\hat{Y}$ by applying RGR to $\tilde{C}_B$ and $\tilde{C}_F$.

*1) Step 1 - Sliding window:* As mentioned above, the adopted CNN architecture either crops or resizes input images to $321 \times 321$ portraits. Since our datasets are composed of images with resolution ranging from $2704 \times 1520$ to $5184 \times 3456$ pixels (see Section 5.3.2), we emulate a sliding window approach to avoid resampling artifacts. More specifically, we split each input image $I$ into a set $P$ of $n$ portraits $p^{(i)} \in P$. Each portrait is $321 \times 321$ pixels large, i.e., $p^{(i)} \in \mathbb{R}^{r \times r}$ with $r = 321$. Cropping non-overlapping portraits from the original image introduces artificial boundaries that compromise the detection quality. For this reason, in our approach each portrait overlaps a percentage $\rho$ of the area of each of its immediate neighbors. For our experiments, we adopted $\rho = 10\%$. When the scoremaps are fused, the results corresponding to the overlapping pixels are discarded. Figure 5.18 illustrates this process for a pair of subsequent portraits. The scores obtained for each portrait are depicted as a heatmap, where blue is associated with lower scores and higher scores are illustrated with red.

*2) Step 2 - CNN prediction:* We evaluate in parallel each portrait $p^{(i)}$ with our fine-tuned network for flower identification. The CNN is equivalent to a function $f$

$$f : p^{(i)} \rightarrow \{c_F^{(i)}, c_B^{(i)}\}, \tag{5.1}$$

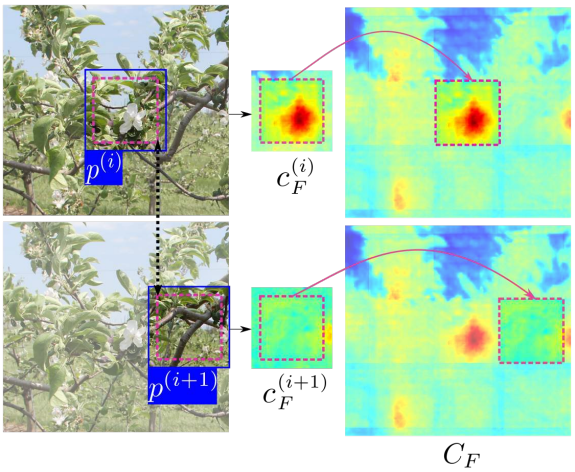
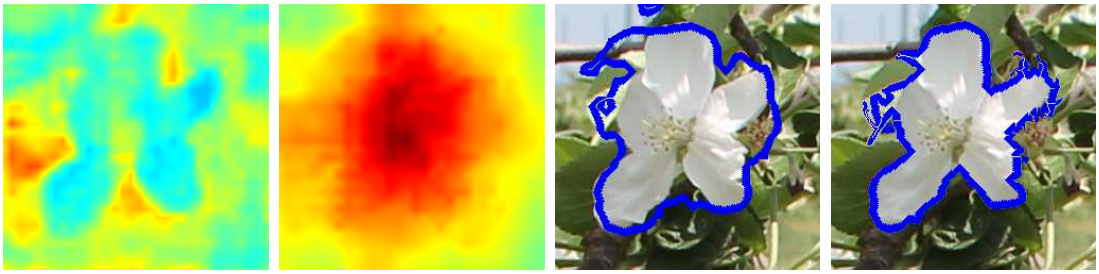



Figure 5.18: Illustration of the sliding window and subsequent fusion process that comprise our segmentation pipeline. Each portrait overlaps a certain area of its neighbors, which is discarded during fusion to avoid artifacts caused by artificial boundaries [2] (©2018 IEEE).

which maps each input $p^{(i)}$ into two pixel-dense scoremaps: $c_F^{(i)} \in \mathbb{R}^{r \times r}$ represents the pixel-wise likelihood that pixels in $p^{(i)}$ belong to the foreground (i.e., flower), while $c_B^{(i)} \in \mathbb{R}^{r \times r}$ corresponds to the pixel-wise background likelihood. The heatmaps in Figures 5.19(a) and (b) are examples of scoremaps computed for a given portrait.



(a) $c_B^{(i)}$ (b) $c_F^{(i)}$ (c) Coarse segmentation (d) Refined segmentation

Figure 5.19: Example of segmentation refinement for a given pair of scoremaps. a) Background scoremap $c_B^{(i)}$. b) Foreground scoremap $c_F^{(i)}$. c) Coarse segmentation by direct thresholding of the scoremaps. d) Refined segmentation using RGR [2] (©2018 IEEE).

*3) Step 3 - Fusion and refinement:* After evaluating each portrait, we generate two global scoremaps $C_B$ and $C_F$ by combining the predictions obtained for all $p^{(i)} \in P$. Let $s^{(i)}$ represent the pixel-coordinates of $p^{(i)}$ in $I$ after discarding the padding pixels. The fusion procedure is defined as

$$\forall p^{(i)} \in P, \quad C_{F,B}(s^{(i)}) = c^{(i)}_{F,B}, \tag{5.2}$$

such that both scoremaps $C_B$ and $C_F$ have the same resolution as $I$. As illustrated in Figure 5.18, the padded areas of $c^{(i)}_{F,B}$ (outside the red box) are discarded during fusion. For every pixel in the image, a single prediction score is obtained from exactly one portrait, such that artifacts introduced by artificial boundaries are avoided.

After fusion, the scoremaps $C_B$ and $C_F$ are normalized into scoremaps $\tilde{C}_B$ and $\tilde{C}_F$ using a softmax function

$$\tilde{C}_{F,B}(q_j) = \frac{\exp(C_{F,B}(q_j))}{\exp(C_B(q_j)) + \exp(C_F(q_j))}, \tag{5.3}$$

where $q_j$ is the $j$-th pixel in the input image $I$. With this formulation, for each pixel $q_j$ the scores $\tilde{C}_B(q_j)$ and $\tilde{C}_F(q_j)$ add to one, i.e., they correspond to the probability that $q_j$ belongs to the corresponding class.

As Figure 5.19(c) shows, the predictions obtained directly from the CNN are coarse in terms of adherence to actual flower boundaries. Therefore, rather than directly thresholding $\tilde{C}_F$, this scoremap and the image $I$ are fed to the RGR refinement module described in Chapter 3. For our application, the refinement algorithm relies on two high-confidence classification regions $R_F$ and $R_B$ defined according to

$$R_{F,B} = \left\{ q_j | \tilde{C}_{F,B}(q_j) > t_{f,b} \right\}, \tag{5.4}$$

where $t_b$ and $t_f$ are the high-confidence background and foreground thresholds. As detailed in Section 5.3.3, the threshold $t_0$ guiding the majority voting within each cluster can be empirically tuned according to the dataset under consideration. Based

on a grid-search optimization on our training dataset, we selected $t_0 = 0.3$ for all our experiments and fixed $t_b = 0.1$ and $t_f = 1.25 \times t_0$.

### 5.3.2 Datasets

We evaluate our method on four datasets that we created and made publicly available: *AppleA*, *AppleB*, *Peach*, *Pear* [27]. We note that the datasets here described are extended versions of the datasets described in Section 5.2. As summarized in Table 5.6, images from different fruit flower species were collected in diverse uncontrolled environments and under different angles of capture.

Table 5.6: Datasets specifications.

| **Dataset** | **No. images** | **Weather** | **Background panel** | **Camera model** | **Resolution** |
|---|---|---|---|---|---|
| *AppleA* | 100 (train) + 30 (val) | Sunny | No | Canon EOS 60D | $5184 \times 3456$ |
| *AppleB* | 18 | Sunny | Yes | GoPro HERO5 | $2704 \times 1520$ |
| *Peach* | 24 | Overcast | No | GoPro HERO5 | $2704 \times 1520$ |
| *Pear* | 18 | Overcast | No | GoPro HERO5 | $2704 \times 1520$ |

Both datasets *AppleA* and *AppleB* are composed of images of apple trees, which were collected in a USDA orchard on a sunny day. In both datasets, the trees are supported with trellises and planted in rows.

*AppleA* is a collection of 147 images acquired using a hand-held camera. From this total, we randomly selected 100 images to build the training set used to train the CNN. Out of the remaining 47 images, 30 were randomly selected to compose the testing set for which we report results in Section 5.3.3. This dataset contains flowers that greatly vary in terms of size, cluttering, occlusion by leaves and branches.

Flowers composing its images have an average area of $10,730$ pixels, but with a standard deviation of $17,150$ pixels. On average, flowers compose only 2.5% of the total image area within this dataset, which is otherwise vastly occupied by leaves.

For the *AppleB* dataset, differently from *AppleA*, a utility vehicle equipped with a background unit was used for imaging, such that trees in other rows are not visible in the images. Figure 5.20 illustrates the utility vehicle used for image acquisition, and Figures 5.21 and 5.22 illustrate the differences between datasets *AppleA* and *AppleB*.


Figure 5.20: Utility vehicle used for imaging. For the *AppleB* dataset, this vehicle was used in conjunction with a background panel [2] (©2018 IEEE).

The *Peach* and *Pear* datasets differ both in terms of species and acquisition conditions, therefore representing adequate scenarios for evaluating the generalization capabilities of the proposed method. Both datasets contain images acquired on an overcast day and without a background unit. Compared to the *AppleA* dataset, images composing these datasets present significantly lower saturation and value means. Tables 5.7 and 5.8 summarize the differences among datasets in terms of the statis-

True Positives — False Negatives — False Positives

Figure 5.21: Examples of flower detection in one image composing the *AppleA* dataset.

True Positives — False Negatives — False Positives

Figure 5.22: Examples of flower detection in one image composing the *AppleB* dataset [2] (©2018 IEEE).

tics of the HSV color components, where $\mu$ stands for mean values and $IQR$ for interquartile ranges.

Regarding the flower characteristics, apple blossoms are typically white, with hue components spread in the whole spectrum (high $IQR_H$) and low saturation mean. Flowers composing the *AppleB* dataset present higher brightness ($\mu_V$), while peach flowers show a pink hue centered on $\mu_H = 325°$, with higher saturation and lower

Table 5.7: HSV statistics of images composing each dataset [2] (©2018 IEEE).

| | **H [0 – 360°]** | | **S [%]** | | **V [%]** | |
|---|---|---|---|---|---|---|
| Dataset | $\mu_H$ | $IQR_H$ | $\mu_S$ | $IQR_S$ | $\mu_V$ | $IQR_V$ |
| *AppleA* | 74.6 | 49.3 | 32.9 | 24.3 | 53.7 | 30.2 |
| *AppleB* | 219.6 | 21.1 | 88.6 | 44.3 | 47.1 | 16.9 |
| *Peach* | 223.8 | 199.9 | 11.8 | 20.7 | 42.3 | 46.6 |
| *Pear* | 85.9 | 178.8 | 16.4 | 23.4 | 42.4 | 20.8 |

Table 5.8: HSV statistics of flowers composing each dataset.

| | **H [0 – 360°]** | | **S [%]** | | **V [%]** | |
|---|---|---|---|---|---|---|
| Dataset | $\mu_H$ | $IQR_H$ | $\mu_S$ | $IQR_S$ | $\mu_V$ | $IQR_V$ |
| *AppleA* | 136.6 | 205.5 | 6.3 | 9.8 | 77.3 | 24.3 |
| *AppleB* | 56.3 | 80.2 | 7.5 | 9.8 | 86.7 | 23.1 |
| *Peach* | 325.2 | 26.7 | 21.2 | 13.3 | 50.2 | 13.7 |
| *Pear* | 215.4 | 173.2 | 5.9 | 5.9 | 84.7 | 22.4 |

value means. Moreover, pear flowers are slightly different in terms of color (greener) and morphology, as illustrated in Figure 5.25.

#### 5.3.2.1 Labeling

Image annotation for segmentation tasks is a laborious and time-consuming activity. Labels must be accurate at pixel-level, otherwise both supervised training and the evaluation of segmentation techniques are compromised. As mentioned in the previous chapter, most existing annotation tools rely on approximating segmentations as polygons, which provide ground truth images that frequently lack accurate adherence to real object boundaries [1]. This problem is particularly noticeable for objects with complex boundaries, such as flowers. Hence, we used our Freelabel annotation tool, which is described in Chapter 4, to annotate the images in the dataset.

Figure 5.23 shows one example of the annotations provided by the user and the corresponding segmentations generated by RGR.

Figure 5.23: Example of ground truth obtained from freehand annotations. *Left:* positive examples are annotated in blue, while hard negatives are indicated in red. *Right:* segmentation obtained after RGR refinement [2] (©2018 IEEE).

### 5.3.3 Experiments and results

We aim at a method capable of accurate multi-species flower detection, regardless of image acquisition conditions and without the need for dataset-specific training or pre-processing. To verify that our method satisfies all these requirements, we performed experiments on the four different datasets described in Section 5.3.2 while only using the *AppleA* dataset for training.

We adopt as the main baseline our previous model described in Section 5.2, which highly outperformed existing methods by employing the Clarifai CNN architecture to classify individual superpixels. We therefore refer to that model as SPPX+CLARIFAI and to our new method as DEEPLAB+RGR. We also compare our

results against a HSV-BASED method [157] that segments images based only on HSV color information and size filtering according to threshold values optimized using grid-search.

All three methods were tuned using the *AppleA* training dataset, with differences in the pipeline for transfer learning. For the three unseen datasets, the SPPX+CLARIFAI relies on a pre-processing step that enhances contrast and removes the different backgrounds present in the images. Our new method DEEPLAB+RGR does not require any pre-processing. Instead, it employs the same pipeline regardless of the dataset, requiring only adjustments in portrait size. As summarized in Table 5.6, images composing the *AppleA* dataset have resolution 4.3× larger than images in the other three datasets. Thus, we split images in these datasets into portraits of $155 \times 155$ pixels, rather than the $321 \times 321$ pixels portraits used for *AppleA*.

The quantitative analysis of segmentation accuracy relies on precision, recall, $F_1$ and intersection-over-union (IoU) metrics [43] computed at pixel-level, instead of the superpixel-wise metrics used in our previous work. Table 5.9 summarizes the results obtained by each method on the different datasets.

Our new model outperforms the baseline methods for all datasets evaluated, especially in terms of generalization to unseen datasets. By combining a deeper CNN architecture and the RGR refinement module, DEEPLAB+RGR improves both prediction and recall rates in the validation *AppleA* set by more than 15%. Figure 5.21 provides a qualitative example of flower detection accuracy in this dataset.

As Figure 5.22 illustrates, images composing the *AppleB* dataset present a higher number of flower buds and illumination changes, especially in terms of sunlight reflection by leaves. Despite the larger variance in comparison with the previous dataset, the performance obtained by DEEPLAB+RGR surpasses 77% in terms of $F_1$.

Table 5.9: Summary of results obtained for each method [2] (©2018 IEEE).

| | | IoU | $F_1$ | Recall | Precision |
|---|---|---|---|---|---|
| *AppleA* | HSV-BASED | 28.0% | 43.7% | 56.5% | 35.7% |
| | SPPX+CLARIFAI | 51.3% | 67.8% | 73.2% | 63.1% |
| | DEEPLAB+RGR | **71.4%** | **83.3%** | **87.7%** | **79.4%** |
| *AppleB* | HSV-BASED | 49.3% | 66.0% | 58.9% | 75.1% |
| | SPPX+CLARIFAI | 50.6% | 67.2% | 68.4% | 66.1% |
| | DEEPLAB+RGR | **63.0%** | **77.3%** | **91.2%** | **67.1%** |
| *Peach* | HSV-BASED | 0.1% | 1.4% | 1.4% | 1.6% |
| | SPPX+CLARIFAI | 49.1% | 67.2% | 71.3% | 61.2% |
| | DEEPLAB+RGR | **59.0%** | **74.2%** | **64.8%** | **86.8%** |
| *Pear* | HSV-BASED | 39.7% | 56.8% | 65.6% | 50.1% |
| | SPPX+CLARIFAI | 40.5% | 57.6% | 49.6% | 68.7% |
| | DEEPLAB+RGR | **75.4%** | **86.0%** | **79.2%** | **94.1%** |

True Positives False Negatives False Positives

Figure 5.24: Examples of flower detection in one image composing the *Peach* dataset. *Left:* detections provided by the SPPX+CLARIFAI method. *Right:* detections obtained with our new DEEPLAB+RGR method [2] (©2018 IEEE).

Results obtained for the *Peach* dataset demonstrate the limitation of color-based methods and two important generalization characteristics of our model. The HSV-BASED method is incapable of detecting peach flowers, since their pink color is very different from the white apple blossoms used for training. On the other hand, our method presents $F_1$ near 75%, indicating that it can properly detect even

flowers that differ to a great extent from apple flowers in terms of color. Moreover, images composing this dataset are characterized by a cloudy sky and hence poorer illumination. Most cases of false negatives correspond to flower buds, due to the lack of such examples in the training dataset. As illustrated in Figure 5.24, poor superpixel segmentation leads the SPPX+CLARIFAI approach to incorrectly classify parts of the sky as flowers. This problem is overcome by our new model, which greatly increases precision rates to above 80%.

Furthermore, the high recall rate provided by DEEPLAB+RGR in the *Pear* dataset demonstrates its robustness to slight variations in both flower morphology and color. As shown in Figure 5.25, similar to the *Peach* dataset, these images also present a cloudy background. In addition to that, their background is characterized by a high level of clutter caused by the presence of a large number of branches. These high texture components compromise the background removal model used by SPPX+CLARIFAI. Still, the DEEPLAB+RGR method detects pear flowers with more than 90% precision.

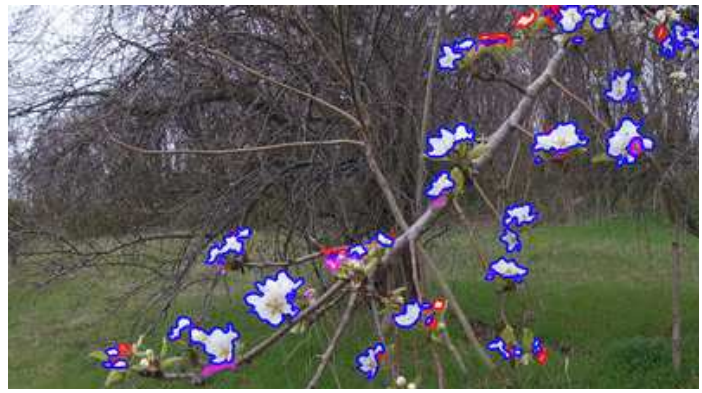



Figure 5.25: Examples of flower detection in one image composing the *Pear* dataset [2] (©2018 IEEE).

The results obtained by our method for the *AppleB*, *Peach* and *Pear* datasets can be further improved by adjusting the parameter $\tau_0$ used for final classification and refinement. As summarized in Figure 5.26, increasing $t_0$ from 0.3 to 0.5 increases in 3% the $F_1$ performance on *AppleB*, reaching both recall and precision levels around 80%. For the *Peach* dataset, decreasing $t_0$ to 0.2 increases the recall rate to above 70%. Such adjustment can be carried out quickly through a simple interactive procedure, where $t_0$ is chosen according to its visual impact on the segmentation of a single image.

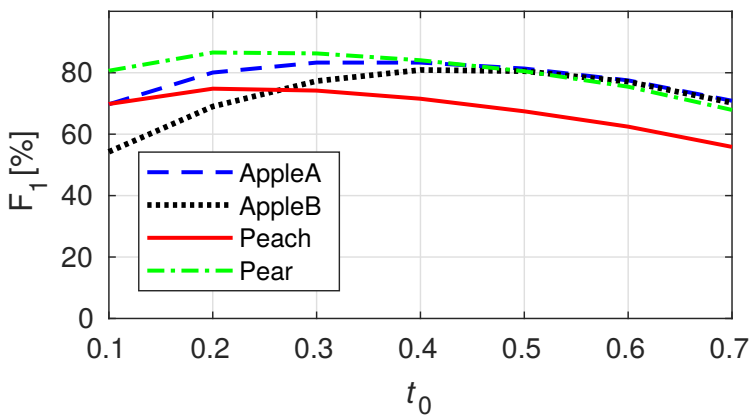



Figure 5.26: Segmentation performance in terms of $F_1$ measure on each dataset according to the parameter $t_0$ [2] (©2018 IEEE).

### Computation time

In terms of inference time, the current implementation of our algorithm on an Intel Xeon$^{TM}$CPU E5-2620 v3 @ 2.40GHz (62GB) with a Quadro P6000 GPU requires on average 50 seconds to evaluate each high-resolution image composing our datasets. Around 5 seconds are required to save portraits as individual files and load

their corresponding prediction scores, a process that can be simplified by generating portraits directly within the neural network framework.

**Acknowledgement** We acknowledge the support of USDA ARS agreement #584080-5-020, and of NVIDIA Corporation with the donation of the GPU used for this research.

# CHAPTER 6
# VISION-BASED ANALYSIS OF ACTIVITY OF DAILY LIVING

The 2017 United Nations report on population ageing estimates the number of people aged 60 years or older to nearly double by 2050 [180]. In this context, the future viability of medical care systems depends upon the adoption of new strategies to minimize the need for costly medical interventions, such as the development of technologies that maximize health status and quality of life in aging populations.

Therefore, there is increasing interest in estimating the health status and frailty of the elderly. *Frailty* is a condition of increased risk of negative health outcomes, including institutionalization, hospitalization and death, due to multi-systemic impairments in multiple domains such as physical, cognitive, and social [181]. Currently, clinicians use evaluation scales that incorporate mobility and Instrumented Activities of Daily Living (IADL) assessments (i.e., a person's ability to use a tool such as a telephone without assistance) [182] to determine the health status of elderly patients and to recommend habit changes. These assessments are episodic and highly subjective, generally taking place at a doctor's office and based on questionnaires or self-reported outcomes.

Despite the potential of recent advances in many areas of computer vision, no current technology allows automatic and unobtrusive assessment of mobility and IADL over extended periods of time in long-term care facilities or patients' homes. Patient activity analysis to date has been limited to simplistic scenarios [183], which do not cover a wide range of relatively unconstrained and unpredictable situations.

Vision-based analysis of mobility and characterization of Activities of Daily Living (ADLs) is challenging. As the examples in Figures 6.6 and 6.7 illustrate, images acquired from assisted living environments cover a wide scene where multiple people can be performing different activities in a varied range of scenarios. Moreover, it

encompasses multiple underlying complex tasks including: detection of subjects and objects of interest, identification of body joints for pose estimation, and estimation of the gaze of the subjects in the scene.

This chapter describes two novel approaches towards the long-term goal of creating video analytics tools to robustly and accurately measure relevant mobility parameters of elderly patients in a relatively uncontrolled environment, such that monitoring can take place unobtrusively over long periods of time. More specifically, it addresses two fundamental building blocks of this long-term research. The first approach, published in [184] and detailed in Section 6.1, addresses the precise segmentation and tracking of individuals in video-streams acquired from assisted living enviroments. The second approach, published in [10] and detailed in Section 6.2, proposes a novel strategy for gaze estimation, which is a critical element to determine how humans interact with the surrounding environment.

Detailed in [185,186], the assisted living environment where our research takes place has been used for studies on automatic assessment of mobility information and frailty [187]. More specifically, the environment is an assisted living facility situated in the Galliera Hospital (Genova, Italy), in which patients, after being discharged from the hospital, are hosted for a few days. The facility is a fully-equipped apartment where patients may be monitored by various sensors, including localization systems and RGB-D and conventional video cameras, which are arranged as shown in Figure 6.1.

## 6.1 Fine segmentation for Activity of Daily Living analysis in a wide-angle multi-camera set-up

This section addresses a fundamental building block of this long-term research: the precise segmentation and tracking of individuals in video-streams acquired by a multi-camera system [184]. The main contributions of this work are summarized as:


Figure 6.1: Images and layout of the instrumented assisted living facility; in color, the fields of view of the video cameras.

- *We describe a framework for fine-grained segmentation of human subjects and objects of interest for the characterization of Activities of Daily Living (ADL).* In addition to the segmentation of human targets in the scene, we also identify objects belonging to a set of pre-defined classes of interest that comprises *sofa, chair,* and *dining table.* Our precise segmentation will allow us to develop robust models of person-person and object-person interaction to be used for ADL analysis, to access functional abilities (that is, the ability of using tools), independence, and social awareness.
- The main challenges of our work are the complexity of the scenario and the fact that the objects of interest may appear in different parts of the image, at different scales, poses and deformations — because of significant distortions due to the need of adopting wide angle optics (see Figure 6.2). To address these challenges, *we propose a multi-stream network in which different patches of a video-frame are fed to separate copies of the network. View-specific distortions are taken into consideration by applying simple ad-hoc geometric transformations to the image patches.*
- *The outputs of the different branches of the network are combined using a fusion mechanism and refined using superpixel segmentation and a probabilistic*

*temporal consistency model.* We assess the performance of our method on the benchmark DAVIS dataset [7, 48] as well as on video streams acquired within our protected discharge facility. The results show that our approach outperforms state-of-the-art video segmentation methods on selected sequences of the standardized datasets and that it generates very accurate semantic object segmentation in real-world videos.


Figure 6.2: Example of frame containing perspective distortion. By applying view-specific transformations (e.g. 45° rotation) to the distorted regions, the segmentation can be substantially improved.

We note that the work described in this section was performed before the development of the RGR algorithm described in Chapter 3. In fact, insights gathered from this work played an important role in the development of RGR.

### 6.1.1 Related work

In order to make inferences about activities of daily living, accurate knowledge about the spatio-temporal relationships among people and objects is needed. Therefore semantic image segmentation is an important element of smart environments.

For applications on real-world scenarios, a limitation of existing methods described in Section 2.3 for semantic segmentation arises from the fact that most benchmark datasets (e.g., the PASCAL [43] and the COCO [5] datasets) contain images that focus on the segmentation of a few (frequently only one or two) salient and large foreground objects. Addressing this limitation is one of the objectives of the proposed work.

When such segmentation must be carried out in a temporally consistent manner across video frames, as is the case in such scenarios, this task is known as semantic video segmentation. Similarly to image semantic segmentation, semantic video segmentation has also benefited from the introduction of publicly available datasets such as DAVIS [7, 48] and SegTrack [188]. Most recent works on video segmentation, however, focus on the simpler task of object segmentation, which consists of accurately segmenting an object in every frame of a video sequence given a mask representing this object in the first video frame.

Although some approaches for video segmentation not based on CNNs have shown good performance [189], methods that employ CNN features to model the appearance of the object tend to perform better [190]. Again, approaches based on FCNs also dominate this field [191–193]. One-shot video object segmentation (OSVOS) [191], for example, uses the FCN of [90] to carry out object segmentation in a frame-by-frame basis without imposing temporal constraints. MSK (MaskTrack+Flow+CRF) [192] performs object segmentation using the DeepLab network of [22] with the object segmentation masks of the previous frame provided as a fourth input channel to the network in order to take into consideration the temporal information. In [193], Jampani et al. propose the Video Propagation Network (VPN), which is one of the few recent approaches to perform both semantic and object video segmentation. Their method uses a bilateral filtering network [194] to carry out tem-

poral propagation and a CNN for spatial segmentation. It has also been integrated with other FCNs such as DeepLab [22].

In contrast to image segmentation methods, most of the above methods solve the problem of object segmentation in videos, disregarding semantic information. Moreover, they focus on segmenting a few large and prominent foreground objects from the background, and hence cannot be directly applied to monitoring the scenarios under consideration.

### 6.1.2 Proposed approach

The method we propose for semantic segmentation of video frames uses a core RefineNet model [99] to compute the likelihood that each pixel belongs to a certain category of interest. In particular for this preliminary study, we opted for the ResNet-101 model that provides state-of-the-art performance on the PASCAL VOC 2012 dataset, which includes all the objects classes we currently consider for ADL analysis: *person, chair, sofa, dining table.*

Since our video sequences for ADL analysis consist of frames acquired with static wide-angle cameras, we also incorporate in our approach strategies to compensate for distortions and to detect non-centered objects. Specifically, instead of evaluating the entire input frame, we devise a semantic video segmentation architecture based on a spatial multi-stream arrangement, as illustrated in Figure 6.3. In each stream, a region is cropped from the input frame and fed into a RefineNet module, which outputs 20 pixel-dense feature maps corresponding to the likelihood that a pixel belongs to a certain PASCAL class. For each class, the computed scores are then combined by a late fusion layer. In addition, in order to compensate for perspective distortions from our wide-angle cameras, each portrait is evaluated both with and without a 45° counter-clockwise rotation. After adding the responses ob-

tained from all the portraits, a pixelwise maximum likelihood evaluation indicates to which class a pixel most likely corresponds.

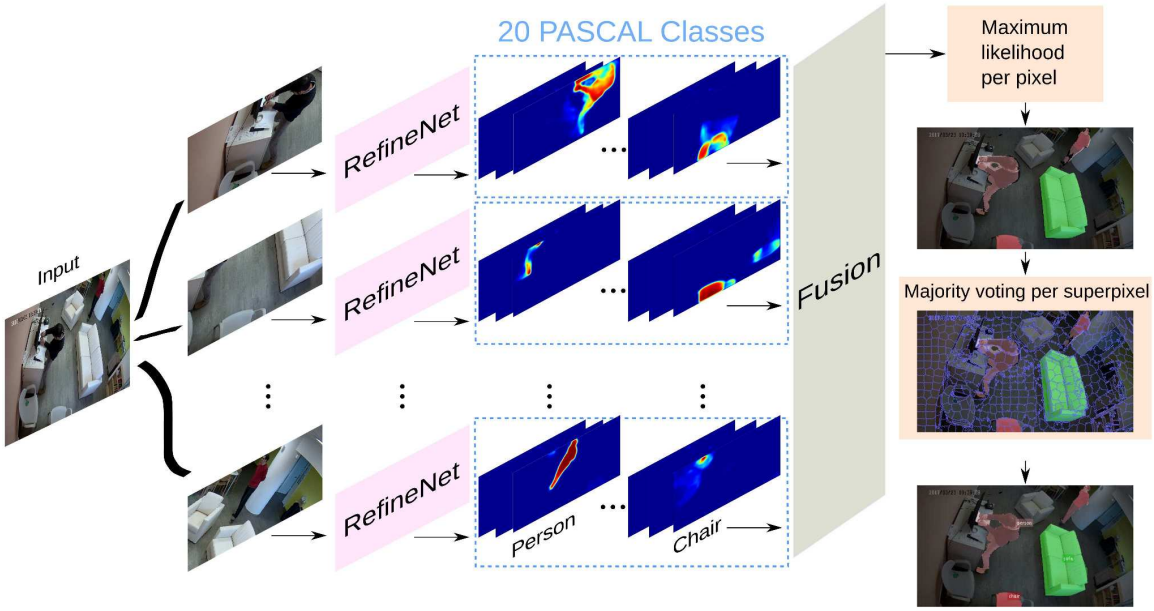



Figure 6.3: Diagram illustrating the sequence of image analysis performed by the proposed model for semantic segmentation of objects of interest.

Conceptually, this type of architecture allows an adaptive feature extraction arrangement in which each CNN module composing a stream as well as the late fusion layer can be fine-tuned through task-specific training. In our current proof-of-concept implementation no supervised fine-tuning is performed, such that all the RefineNet modules share the same pre-trained weights, view-specific segmentation is approximated by the 45° rotations, and late fusion is performed by a simple summation.

Another important requirement in our reference application is the accuracy of the segmentation so that interactions between different objects and agents can be reliably estimated. Although the segmentations obtained with RefineNet are finer than the ones obtained with deconvolutional models such as FCN, a close inspection

reveals that the results can be improved especially in terms of boundary adherence. Unsupervised techniques such as superpixel segmentation are capable of better exploring local information to estimate boundaries of objects composing an input image. Therefore, in our approach, we additionally segment the images using superpixels and each superpixel is then classified according to a majority voting scheme based on the scores obtained from the RefineNet method. That is, if more than 50% of the pixels composing a superpixel present a score over a certain threshold for a given class, the superpixel is considered a positive detection of the object represented by that class. We selected the Extended Topology Preserving Segmentation (ETPS) [62] superpixel algorithm to compose our image segmentation model, described in the Section 2.4.4.

For images acquired from our discharge facility, in addition to framewise analysis, temporal information can be used based on prior information estimated from the environment. Since the images are acquired by static cameras, the likelihood of sharp transition of labels across two subsequent frames (at a 25 fps rate) is rather low. Therefore, for these video sequences we also include a simple probabilistic model that, in addition to the likelihood scores obtained using RefineNet, takes into account temporal information by attributing a higher probability of detection to pixels detected in the previous frame, while the probability of transition between labels (e.g. background to foreground or vice-versa) is lower.

Let $p_j^{(t)}$ represent a pixel $j$ at frame $t$, while $\mathcal{F}$ denotes any foreground class. With this representation, let $c_j^{(t)}$ denote the class likelihood estimated by the CNN for $p_j$ at time instant $t$. As per Eq. 6.1 , we define an adjusted estimation $\tilde{c}_j^{(t)}$ of the probability that pixel $p_j$ belongs to foreground at instant $t$, which takes into consideration both $c_j^{(t)}$ as well as the label $y_j^{(t-1)}$ assigned to $p_j$ at the previous time instant.

Two events are considered: i) the case where the class predicted in frame $t-1$ is the same as the one predicted for frame $t$; and ii) the complimentary case where

different classes are predicted at instants $t-1$ and $t$. As summarized in Eq. 6.2, we multiply the predicted scores by an empirically defined constant $\alpha$ as a temporal prior: for the case where $y_j^{(t)} = y_j^{(t-1)}$, $\alpha = 0.9$; in the complimentary case where different classes are predicted at instants $t-1$ and $t$, the prediction score is multiplied by $1-\alpha = 0.1$.

$$\tilde{c}_j^{(t)} = P\left(p_j^{(t)} \in \mathcal{F} | c_j^{(t)}, y_j^{(t-1)}\right) \tag{6.1}$$

$$\tilde{c}_j^{(t)} = c_j^{(t)} \times \left[\alpha \mathbb{1}_{y_j^{(t)} = y_j^{(t-1)}} + (1-\alpha)\mathbb{1}_{y_j^{(t)} \neq y_j^{(t-1)}}\right] \tag{6.2}$$

### 6.1.3 Assessment on benchmark data

Quantitative evaluation of video segmentation methods requires pixel-accurate and per-frame ground truth annotation, a notoriously labor-intensive and time-consuming task. For that reason, we quantitatively assess the performance of our method on video sequences composing the DAVIS 2016 dataset, which reflects many of the properties of our reference application. It comprises scenarios such as target occlusion, motion-blur, scenes with depth and appearance/pose changes, all of which are likely to occur in our application-specific video sequences. For an evaluation that resembles the environment of our application, where the classes of objects to be detected are known a priori, we selected only video sequences where targets correspond to objects contained on the PASCAL VOC 2012, disregarding sequences containing unknown objects.

To verify the efficacy of the proposed per-superpixel majority voting scheme, we compare two approaches against the baseline methods: one composed only by RefineNet (which we refer to as RN) and one combining RefineNet and superpixel analysis (which we refer to as RS). Although we provide a comparison against multiple video segmentation techniques proposed for the DAVIS challenge, it is important to

Table 6.1: Jaccard index (J) / Contour accuracy (F) Per-Sequence

| ***Sequence*** | **RS** | **RN** | **MSK [192]** | **VPN [193]** | **OFL [190]** | **NLC [195]** |
|---|---|---|---|---|---|---|
| *bmx-bumps* | 45.8/60.5 | 42.6/63.0 | 57.1/67.8 | 41.8/59.2 | 47.5/52.9 | **63.5**/**73.4** |
| *bmx-trees* | 44.9/64.6 | 44.5/64.2 | **57.5**/**73.6** | 33.5/46.2 | 14.9/16.4 | 21.2/33.0 |
| *breakdance -flare* | **86.4**/91.2 | 83.5/**92.2** | 77.6/78.4 | 82.7/90.8 | 75.6/78.3 | 80.4/80.8 |
| *hike* | 90.6/94.2 | 85.5/94.8 | 93.1/96.0 | 88.0/95.4 | **93.4**/**96.6** | 91.8/94.3 |
| *hockey* | 83.2/81.4 | 80.8/83.1 | 83.4/79.1 | 78.5/80.3 | **84.9**/ **88.9** | 81.0/80.8 |
| *horsejump -high* | 82.1/84.4 | 80.2/86.0 | 81.7/85.1 | 81.8/86.3 | **86.3**/**90.4** | 83.4/88.1 |
| *horsejump -low* | 82.6/86.8 | **82.7**/**89.6** | 80.6/81.2 | 74.4/71.3 | 82.2/85.9 | 65.1/65.9 |
| *kite-surf* | 64.7/44.8 | 60.7/42.5 | 60.0/43.8 | 62.3/**53.5** | **70.3**/49.7 | 45.3/44.8 |
| *lucia* | 89.9/**92.4** | 86.8/**92.4** | **91.1**/89.5 | 86.4/90.2 | 89.7/89.4 | 87.6/87.2 |
| *motocross -bumps* | **89.2**/81.9 | 88.0/**82.8** | 59.9/55.4 | 87.2/82.4 | 47.4/48.0 | 61.4/56.0 |
| *motorbike* | 79.1/75.6 | 79.9/75.3 | 56.6/59.7 | **80.8**/**81.4** | 47.6/50.4 | 71.4/57.1 |
| *paragliding -launch* | 61.4/20.6 | 59.2/19.4 | 62.1/22.9 | 61.4/23.1 | **63.7**/**25.3** | 62.8/24.3 |
| *parkour* | **89.3**/90.3 | 85.9/**92.1** | 88.2/87.4 | 87.3/91.7 | 85.9/87.0 | 90.1/91.6 |
| *rollerblade* | 84.5/86.1 | 81.7/89.5 | 78.7/85.0 | 81.4/87.9 | **89.2**/**94.0** | 81.4/86.8 |
| *scooter -gray* | 73.5/66.8 | 72.6/**68.1** | **82.9**/65.9 | 76.8/68.7 | 25.8/20.8 | 58.6/46.7 |
| *swing* | 77.0/66.2 | 75.1/66.3 | 81.9/74.5 | 82.5/**78.7** | 56.2/59.2 | **85.1**/77.8 |
| *tennis* | 85.1/90.5 | 82.5/92.1 | 86.1/91.1 | 79.0/89.4 | 81.7/87.2 | **87.1**/**92.7** |
| **Mean** | **77.0**/75.2 | 74.8/**76.1** | 75.2 / 72.7 | 74.4 / 75.1 | 67.2 / 65.9 | 71.6 / 69.5 |

note that our goal is not a task-restricted model that aims to achieve top ranking performance on this specific dataset. For this reason, unlike most reference methods, we do not perform any type of fine-tuning using the training sequences provided by the DAVIS dataset. The only additional information used as prior knowledge are the classes composing the foreground/target of each sequence.

Following the official guidelines for the DAVIS Challenge, we compare our method against the baseline ones in terms of *Jaccard index ($\mathcal{J}$)* and *contour accuracy*

*($\mathcal{F}$)*, as defined in Section 2.4.2. Table 6.1 summarize the results obtained for each video sequence, with the best results highlighted in bold[1].

For the evaluated sequences, both RN and RS approaches provide results that are competitive to the state-of-the-art methods, with average performance slightly superior for both metrics. This is particularly relevant considering that the baseline methods mostly have the advantage of being fine-tuned for this dataset. In addition, several of the sequences in the table were used as training sequences for some methods, and are hence not an indicative of their performance on data previously unseen by the trackers. For four video sequences, the performance in terms of segmentation similarity ($\mathcal{J}$) obtained using RefineNet based methods are superior to the ones provided by existing approaches. Similarly, for five sequences RN and RS achieve better contour accuracy ($\mathcal{F}$) than the baseline methods.

In addition, the performance of our method is consistent over time. Results obtained in terms of $\mathcal{J}$ decay ($\mathcal{D}_{\mathcal{J}}$) and $\mathcal{F}$ decay ($\mathcal{D}_{\mathcal{F}}$) evidence this characteristic as indicated in Figure 5.11 (left), which shows that our method outperforms all the other approaches in these metrics. Moreover, a closer inspection based on pixelwise precision and recall (PR) metrics reveals that for most cases the detections provided by the proposed method are very precise. Figure 6.4 (right) shows the PR curves summarizing the average performance of both RN and RS methods for the selected sequences. As the figure indicates, both approaches can simultaneously obtain precision and recall of approximately 85%, with the RS approach providing slightly higher precision at higher recall rates.

Figure 6.5 illustrates the segmentation accuracy for six scenarios that particularly resemble some challenges likely to occur in image analysis for ADL, e.g. poses variation, occlusion, and scenes with depth and appearance changes. In these images,

[1]Note that results for OSVOS are not included in the table because the authors of [194] do not report their results on the official training set, which contains most of our selected sequences.

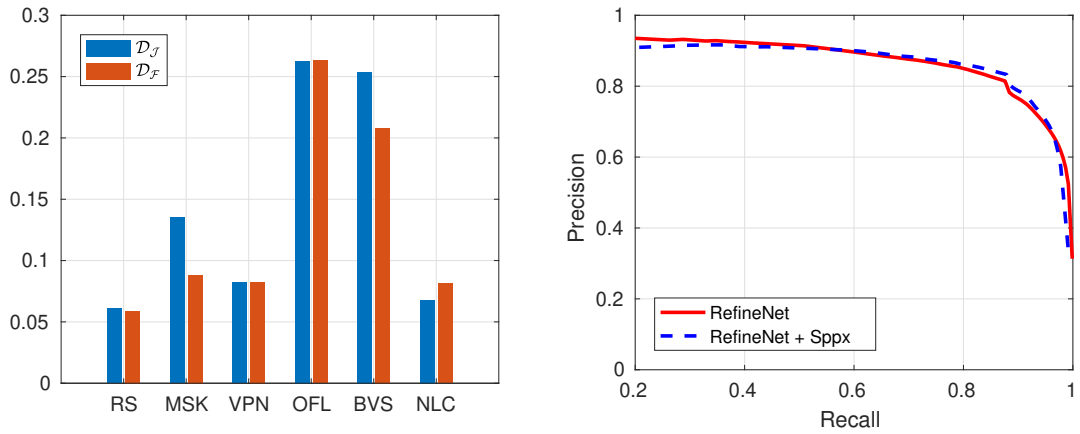



Figure 6.4: Performances on video sequences selected from the DAVIS 2016 dataset. *Left:* Mean $\mathcal{J}$ decay ($\mathcal{D}_\mathcal{J}$) and $\mathcal{F}$ decay ($\mathcal{D}_\mathcal{F}$) for each method. *Right:* Average precision recall curve of RefineNet based methods.

pixels correctly detected are marked in green. The red color indicates false positives, while false negatives are shown in blue. These results are a good evidence of the model robustness against such challenges. As illustrated by the frames extracted from the sequences *hockey* and *paragliding-launch*, false negatives mostly correspond to regions corresponding to objects unknown to the RefineNet model (i.e., not present in the PASCAL dataset).

### 6.1.4 Application to ADL

While the evaluation performed on benchmark data gives indications of the method performance in terms of the detection of objects of interest, the determination of its suitability for ADL analysis requires a task-oriented assessment using videos acquired within the protected discharge facility.

To quantitatively estimate the detection accuracy, we manually counted the number of correctly identified objects and false positives within two sequences of 50 frames, each acquired with one of the two cameras (named *view1* and *view2*) installed in our discharge facility. An object is considered correctly detected when at least 70%


True Positives False Positives False Negatives

Figure 6.5: Examples of segmentation accuracy for scenarios including unusual poses, occlusion, depth and appearance changes.

of its total area has been properly segmented, while a false positive corresponds to incorrect isolated detections of any size. To reduce labeling bias and in order to keep an approximately constant tolerance, each pair of detections was evaluated by the same human subject.

Two different approaches were evaluated. The first one (which we refer to as *RN*) consists of directly evaluating each frame using solely the RefineNet model, while the second (*MRST*) corresponds to the proposed method summarized in Figure 6.3, which employs multiple streams, superpixel enhancement, and the aforementioned temporal probabilistic model in conjunction with the pre-trained CNN.

Figure 6.6 shows qualitative results demonstrating that the proposed approach can detect and segment most of the relevant objects present within a scene, such as *person, chairs* and *tables*. Tables 6.2 and 6.3 summarize the quantitative results obtained for the video sequences acquired with cameras *view1* and *view2*. Both tables present the total number of correct detections for each object category under consideration as well as the average number of correct detections per frame. The table also

shows the total and average number of false positives. Both tables show significantly higher number of total correct detections for results obtained using MRST, for all object classes in both views. Although MRST generated 3 additional false positives in *view1*, all three occurred in the first three frames evaluated, before the temporal model stabilized.


Figure 6.6: Examples of segmentation obtained for images acquired with cameras *view1* (top row) and *view2* (bottom row). *Left column:* results obtained using solely RefineNet; *Right column:* results obtained combining multiple streams, RefineNet, superpixel enhancement and temporal probabilistic model.

Given the position of the second camera in the discharge facility (upper corner of the room), images acquired with this camera are particularly relevant since perspective distortions are present in every frame. The higher number of people and sofas detected in these frames using MRST demonstrate the effectiveness of the multiple streams and image rotations to obtain segmentations somewhat robust against the existing distortions.

Table 6.2: Analysis of 50 frames - View 1

| | RN | | MRST | |
|---|---|---|---|---|
| ***Class*** | Total | Avg. | Total | Avg. |
| *People* | 136 | 2.72 | 141 | 2.82 |
| *Chairs* | 126 | 2.52 | 188 | 3.76 |
| *Tables* | 65 | 1.30 | 74 | 1.48 |
| *TV* | 19 | 0.38 | 38 | 0.76 |
| *FP* | 4 | 0.08 | 7 | 0.14 |

Table 6.3: Analysis of 50 frames - View 2

| | RN | | MRST | |
|---|---|---|---|---|
| ***Class*** | Total | Avg. | Total | Avg. |
| *People* | 31 | 0.62 | 56 | 1.12 |
| *Chairs* | 48 | 0.96 | 50 | 1.00 |
| *Sofas* | 0 | 0.00 | 48 | 0.96 |
| *FP* | 9 | 0.18 | 0 | 0.00 |

### 6.1.5 Preliminary experiments using FreeLabel and RGR

After the development of the RGR and FreeLabel algorithms described in Chapters 3 and 4, we performed preliminary experiments to verify the potential of exploiting these tools for annotation and segmentation of video sequences acquired from the described assisted living environment.

In Figure 6.7, the left-most column exemplifies the quality of segmentation masks that can be generated by annotating frames using FreeLabel, while the middle and right-most columns contain segmentation masks obtained after refinement of DeepLab-LargeFOV [22] predictions (model trained on PASCAL) using superpixels and RGR, respectively. These results reveal a promising direction for future work, as fine-grained ground-truth masks can be obtained using FreeLabel, despite the limited resolution of the frames ($270 \times 480px$) and small size of the objects in terms of covered image area. Moreover, segmentation refinement using RGR results on significantly better predictions in terms of adherence to actual objects' boundaries, in comparison to the results obtained using superpixel-based refinement.

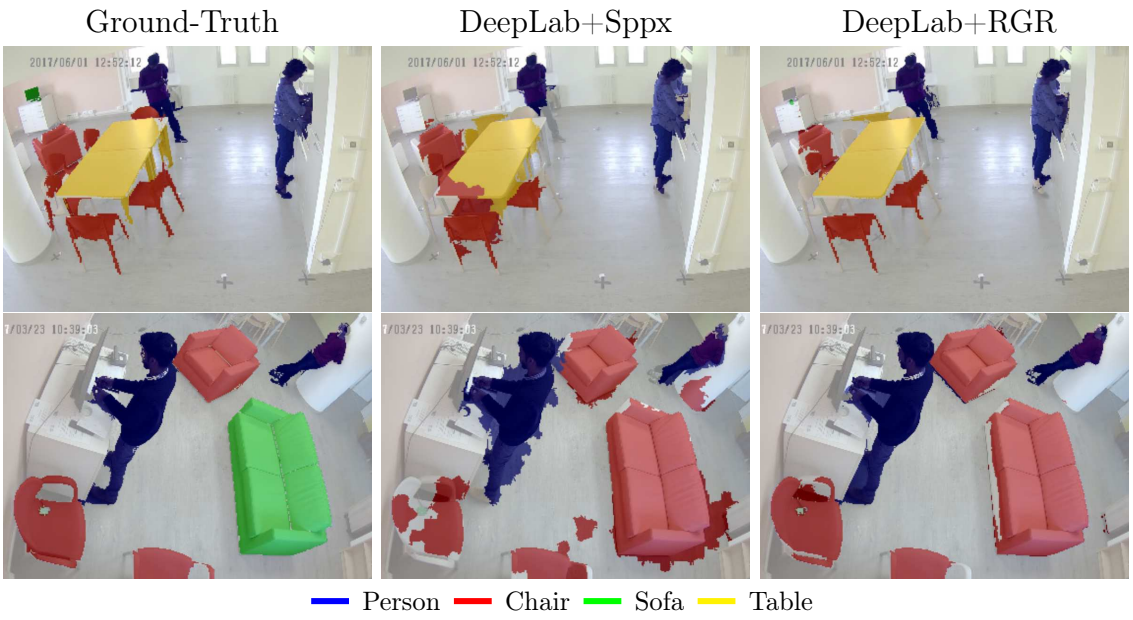



Figure 6.7: Preliminary usage of FreeLabel and RGR on images from the discharge facility described in this chapter. *Left:* annotations generated using FreeLabel; *middle:* segmentation obtained by combining DeepLab and superpixels-based refinement; *right:* segmentation obtained by combining DeepLab and RGR-based refinement.

## 6.2 Gaze estimation for assisted living environments

In this section we focus on gaze estimation [10], which is a critical element to determine how humans interact with the surrounding environment. It has been applied to design human-computer interaction methods [196] and to analyze social interactions among multiple individuals [197]. For our application, in conjunction with object detection [184], gaze direction could define mutual relationships between objects and their users (e.g., the user is sitting on a chair with a book on his/her lap vs. sitting on a chair reading the book) and classify simple actions (e.g., mopping the floor, getting dressed, cooking food, eating/drinking).

The contributions of the work described in this section can be summarized in three main points:

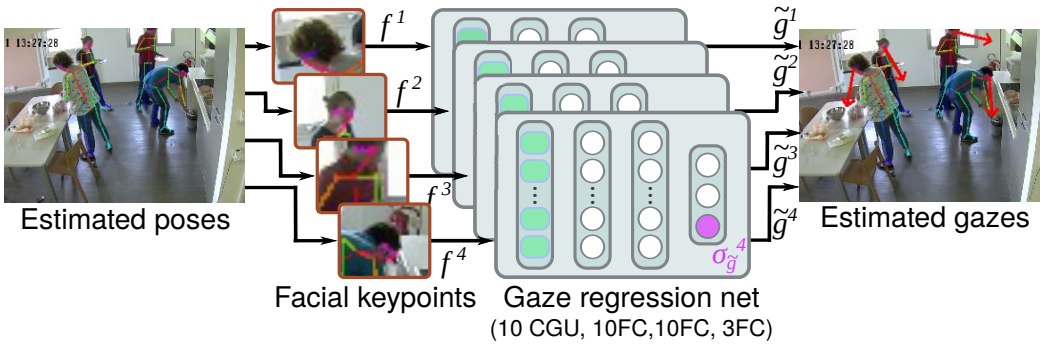



Figure 6.8: Overview of our apparent gaze estimation approach. The anatomical keypoints of all the persons present in the scene are detected using a pose estimation model [198]. The facial keypoints of each person are then provided as inputs to a neural network regressor that outputs estimations of their apparent gaze and its confidence on each prediction [10] (©2020 IEEE).

- *we propose an approach that relies solely on the relative positions of facial keypoints to estimate gaze direction.* As shown in Figure 6.8, we extract these features using the off-the-shelf OpenPose model [198]. From the coordinates and confidence levels of the detected facial keypoints, our regression network estimates the apparent gaze of the corresponding subjects. From the perspective of the overall framework for ADL analysis, leveraging the facial keypoints is beneficial because a single feature extractor module can be used for two required tasks: pose estimation and gaze estimation. Code is available at `coviss.org/codes`
- the complexity of gaze estimation varies according to the scenario, such that the quality of predictions provided by a gaze regressor is expected to vary case-by-case. For this reason, *our model is designed and trained to provide an estimation of its uncertainty for each prediction of gaze direction.* To that end, we leverage concepts used by Bayesian neural networks for estimation of aleatoric uncertainty.
- in cases such as self-occlusion, one or more facial keypoints might not be detected, and OpenPose assigns a confidence of zero to the corresponding feature.

To handle the absence of detections, *we introduce the concept of Confidence Gated Units (CGU)* to induce our model to disregard detections for which a low confidence level is provided.

### 6.2.1 Related work

Estimating the relative pose of subjects is crucial to perform high level tasks such as whole body action recognition and understanding the relationship between a person and the environment. Appearance-based pose estimation systems attempt to infer the positions of the body joints of the subjects present in a scene. Traditional methods relied on models fit to each of the individual subjects found in a given image frame [199,200]. More recent approaches employ convolutional architectures [198,201] to extract features from the entire scene, therefore making the whole process relatively independent of the number of subjects in the scene.

At a finer level, the analysis of human facial features may provide additional information [202] about well-being. For example, facial expression recognition [203,204] can be used in sentiment analysis [205]. Facial analysis can also provide information on gaze direction, which is useful to better understand the interaction between a person and his/her surrounding environment [197]. Recent contributions in this area attempt to infer the orientation of a person's head by fitting a 3D face model to estimate both 2D [206] and 3D gaze information [207]. Other contemporary methods resort to different types of information, which include head detection, head orientation estimation, or contextual information about the surrounding environment [208]. In the context of human-computer interaction, the work in [209] employs an end-to-end architecture to track the eyes of a user in real-time using hand-held devices.

However, most works and datasets on inference of head orientation and gaze focus on specific scenarios, such as images containing close-up views of the subjects' heads [206, 210], with restricted background size and complexity. More similar to

our scenario of interest, the GazeFollow dataset introduced in [211] contains more than 120k images of one or more individuals performing a variety of actions in relatively unconstrained scenarios. Together with the dataset, the authors introduce a two-pathway architecture that combines contextual cues with information about the position and appearance of the head of a subject to infer his/her gaze direction. A similar model is introduced in [212], with applicability extended to scenarios where the subject's gaze is directed somewhere outside the image.

Gaze estimation is a task with multiple possible levels of difficulty, which vary according to the scenario of observation. Even for humans, it is much easier to tell where someone is looking if a full-view of the subject's face is available, while the task becomes much harder when the subject is facing backwards with respect to the observer's point of view. In modeling terms, this corresponds to heteroscedastic uncertainty, i.e., uncertainty that depends on the inputs to the model, such that some inputs are associated to more noisy outputs than others.

As explained in Section 2.6, conventional deep learning models do not provide estimations of uncertainties for their outputs, with Bayesian deep learning approaches becoming increasingly more popular to understand and estimate uncertainty with deep learning models [213–215]. Under this paradigm, a customized loss function is sufficient for learning a regressor model that also predicts the variance of this noise as a function of the input [109], without need for uncertainty labels.

### 6.2.2 Proposed approach

Our method estimates a person's apparent gaze direction according to the relative locations of his/her facial keypoints. As Figure 6.8 indicates, we use OpenPose [198] to detect the anatomical keypoints of all the persons present in the scene. Of the detected keypoints, we consider only those located in the head (i.e., the nose, eyes, and ears) of each individual.

Let $p^j_{k,s} = [x^j_{k,s}, y^j_{k,s}, c^j_{k,s}]$ represent the horizontal and vertical coordinates of a keypoint $k$ and its corresponding detection confidence value, respectively. The subscript $k \in \{n, e, a\}$ represents the nose, eyes, and ears features, with the subscript $s \in \{l, r, \emptyset\}$ encoding the side of the feature points.

Aiming at a scale-invariant representation, for each person $j$ in the scene we centralize all detected keypoints with respect to the head-centroid $h^j = [x^j_h, y^j_h]$, which is computed as the mean coordinates of all head keypoints detected in the scene. Then, the obtained relative coordinates are normalized based on the distance of the farthest keypoint to the centroid. In this way, for each detected person we form a feature vector $f \in \mathbb{R}^{15}$ by concatenating the relative vectors $\hat{p}^j_{k,s} = [\hat{x}^j_{k,s}, \hat{y}^j_{k,s}, c^j_{k,s}]$

$$f^j = \left[\hat{p}^j_{n,\emptyset}, \hat{p}^j_{e,r}, \hat{p}^j_{e,l}, \hat{p}^j_{a,r}, \hat{p}^j_{a,l}\right]. \tag{6.3}$$

#### 6.2.2.1 Network architecture using gated units

Images acquired from assisted living environments can contain multiple people performing different activities, such that their apparent pose may vary significantly and self-occlusions frequently occur. For example, in lateral-views at least an ear is often occluded, while in back-views nose and eyes tend to be occluded. As consequence, an additional challenge intrinsic to this task is the representation of missing keypoints. In such cases, OpenPose outputs 0 for both the spatial coordinates $(x, y)^j_{k,s}$ and also the detection confidence value $c^j_{k,s}$. Since the spatial coordinates are centralized with respect to the head-centroid $h^j$ as the $(0, 0)$ reference of the input space, a confidence score $c^j_{k,s} = 0$ plays a crucial role in indicating both the reliability and also the absence of a keypoint.

Inspired by the Gated Recurrent Units (GRUs) employed in recurrent neural networks [216], we propose a Confidence Gated Unit (CGU) composed of two internal units: i) a ReLU unit acting on an input feature $q_i$, and ii) a sigmoid unit to emulate

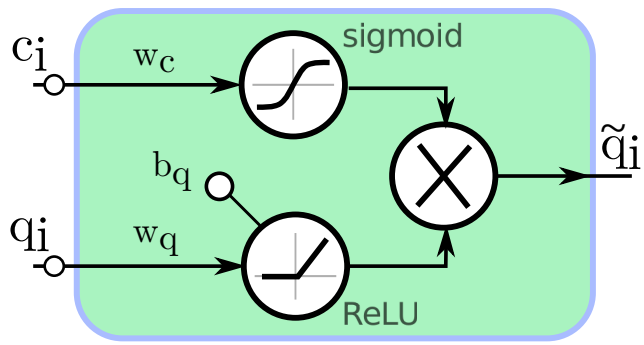



Figure 6.9: The proposed Confidence Gated Unit (CGU) [10] (©2020 IEEE).

the behavior of a gate according to a confidence value $c_i$. As depicted in Figure 6.9, we opt for a sigmoid unit without a bias parameter, to avoid potential biases towards models that disregard $c_i$ when trained with unbalanced datasets where the majority of samples are detected with high confidence. Finally, the outputs of both units are then multiplied into an adjusted CGU output $\tilde{q}_i$.

For our application, a CGU is applied to each pair coordinate-confidence $(\hat{x}^j_{k,s}, c^j_{k,s})$ and $(\hat{y}^j_{k,s}, c^j_{k,s})$. To properly exploit the full range of the sigmoid function and thus reach output values near 0 for $c^j_{k,s} = 0$, we centralize and standardize the input confidence scores according to the corresponding dataset statistics. In this way, our proposed network for gaze regression has a combination of 10 CGUs as input layer.

Moreover, the variety of view-points from which a subject might be visible in the scene, occlusions and unusual poses lead to a vast range of scenarios where the difficulty of the gaze estimation varies significantly. Hence, we design a model that incorporates an uncertainty estimation method, which indicates its level of confidence for each prediction of gaze direction. From an application perspective, this additional information would allow us to refine the predictions by choosing between different cameras, models, or time instants.

The gaze direction is approximated by the vector $\tilde{g}^j = [\tilde{g}_x, \tilde{g}_y]$, which consists of the projection onto the image plane of the unit vector centered at the centroid $h^j$. In terms of architecture design, this corresponds to an output layer with 3 units: two that regress the $(\tilde{g}_x, \tilde{g}_y)$ vector of gaze direction, and an additional unit that outputs the regression uncertainty $\sigma_{\tilde{g}}$.

Following ablative experiments and weight visualization to identify dead units, we selected an architecture where the CGU-based input layer is followed by 2 fully-connected (FC) hidden layers with 10 units each, and the output layer with 3 units. Thus, the architecture has a total of 283 learnable parameters and can be summarized as: (10 CGU, 10 FC, 10 FC, 3 FC).

#### 6.2.2.2 Training strategy

While all the weights composing the fully-connected layers are initialized as in [217], we empirically observed better results when initializing the parameters composing CGU units with *ones*. Since these compose only the input layer, initializing the weights as such does not represent a risk of gradient explosion as no further backpropagation has to be performed. Intuitively, our rationale is that the input coordinate features should not be strongly transformed in this first layer, as at this initial point no information from additional keypoints is accessible. Regarding regularization, we empirically observed better results without regularization in the input and output layers, while a L2 penalty of $10^{-4}$ is applied to parameters of both FC hidden layers.

Regardless of the dataset, we trained our network only on images where at least two facial keypoints are detected. Since we are interested on estimating direction of gaze to verify whether any object of interest is within a person's field of view, we opt for optimization and evaluations based on angular error. Thus, training was

performed using a cosine similarity loss function that is adjusted based on Eq. 2.24 [109] to allow uncertainty estimation.

Let $\mathcal{T}$ be the set of annotated orientation vectors $g$, while $\tilde{g}$ corresponds to the estimated orientation produced by the network and $\sigma_{\tilde{g}}$ represents the model's uncertainty prediction. Our cost function is then given by

$$\mathcal{L}_{cos}(g, \tilde{g}) = \frac{1}{|\mathcal{T}|} \sum_{g \in \mathcal{T}} \frac{\exp(-\sigma_{\tilde{g}})}{2} \frac{-g \cdot \tilde{g}}{||g|| \cdot ||\tilde{g}||} + \frac{log\, \sigma_{\tilde{g}}}{2}. \tag{6.4}$$

From a Bayesian perspective, the MSE component of the loss function described in [109] corresponds to a prior that follows a Gaussian distribution. As described in [218], the von Mises distribution can be interpreted as an analog of the Gaussian distribution for the scenario where an angular distribution over $(0, 2\pi)$ is to be modeled. In this context, by replacing the MSE component with the cosine similarity, our loss function is thus equivalent to the log-likelihood of a Von Mises distribution.

Moreover, with this loss function no additional label is needed for the model to learn to predict its own uncertainty. The $\exp(-\sigma_{\tilde{g}})$ component is a more numerically stable representation of $\frac{1}{\sigma_{\tilde{g}}}$, which encourages the model to output a higher $\sigma_{\tilde{g}}$ when the cosine error is higher. On the other hand, the regularizing component $\log(\sigma_{\tilde{g}})$ helps avoiding an exploding uncertainty prediction.

In terms of model optimization, all experiments were performed using the Adam [219] optimizer with early stopping based on angular error on the corresponding validation sets. Additional parameters such as batch size and learning rate varied according to the dataset. Hence, we describe them in detail in Section 6.2.3.

### 6.2.3 Experiments and results

We evaluate our approach on two different datasets. The first is the GazeFollow dataset [211], on which we compare our method against two different baselines.

The second dataset, which we refer to as the *MoDiPro* dataset, comprises images acquired from an actual discharge facility as detailed in Section 6.2.3.2.

#### 6.2.3.1 Evaluation on the GazeFollow dataset

**Dataset split and training details**

The publicly available GazeFollow dataset contains more than 120k images, with corresponding annotations of the eye locations and the focus of attention point of specific subjects in the scene. We use the direction vectors connecting these two points to train and evaluate our regressors. The left polar histogram in Figure 6.10 summarizes the angular distribution of gaze annotations composing the GazeFollow training set. About 53% of these samples correspond to subjects whose gaze direction lies within the quadrant $[-90°, 0°]$ with respect to the horizontal axis. On the other hand, in only 29% of the cases their gaze direction is within the $[-180°, -90°]$ quadrant.

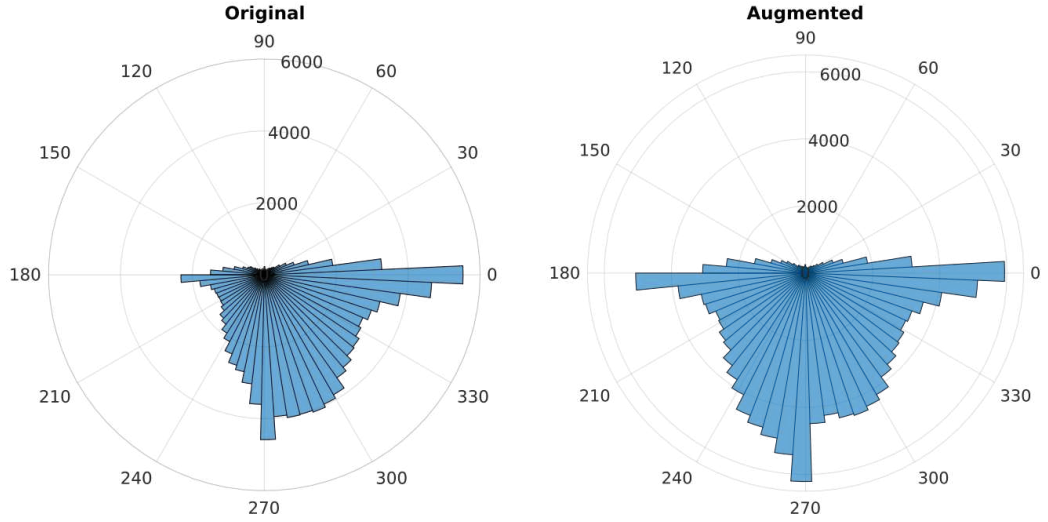



Figure 6.10: Angular distribution of gaze annotations composing the training set of the GazeFollow dataset. *Left:* before augmentation; *right:* after augmentation.

To compensate such bias, we augment the number of samples in the later quadrant by mirroring with respect to the vertical-axis a subset of randomly selected samples from the most frequent quadrant. As the plot in the right side of Figure 6.10 illustrates, this augmentation procedure yields a more symmetrical angular distribution.

Finally, to train our model, we split the training set into two subsets: 90% for *train*, and 10% for validation *val* subset. Training is performed using a learning rate $5 \times 10^{-3}$, batches of 1024 samples and early-stopping based on angular error on the *val* subset. The *test* set comprises 4782 images, with ten different annotations per image. For evaluation, we follow [211] and assess each model by computing the angular error between their predictions and the average annotation vector.

The GazeFollow dataset is structured such that for each image only the gaze from a specific subject must be assessed. For images containing multiple people, this requires identifying which detection provided by OpenPose corresponds to the subject of interest. To that end, we identify which detected subject has an estimated head-centroid that is the closest to the annotated eye-coordinates $E_{GT}$, which are provided as ground-truth. To avoid mismatches when the correct subject is not detected but detections for other subjects on the scene are available, we impose that gaze is estimated only if $E_{GT}$ falls within a radius of $1.5 \times \delta$ around the head-centroid, where $\delta$ corresponds to distance between the centroid and its farthest detected facial keypoint.

We compare our method against two baselines. The first, which we refer to as Geom, relies solely on linear geometry to estimate gaze from the relative facial keypoints positions. Comparison against this baseline aims at evaluating if training a network is needed to approximate the regression $f \rightarrow g$, instead of directly approximating it by a set of simple equations. The second baseline is the model introduced together with the GazeFollow dataset in [211], which consists of a deep

neural-network that combines a gaze pathway and a saliency pathway that are jointly trained for gaze estimation. We refer to this baseline as GF-MODEL.

**Comparison against geometry-based baseline**

The GEOM baseline is based on the model introduced in [220] for face orientation estimation. Although the referenced model makes minimal assumptions about the facial structure [220], it additionally requires mouth keypoints and pre-defined model ratios. Thus, GEOM consists of a simplification of this model as follows.

In short[2], let $\vec{s}$ represent the facial symmetry axis that is computed as the normal of the eye-axis. We estimate the facial normal $\vec{n}$ as a vector that is normal to $\vec{s}$ while intersecting $\vec{s}$ at the detected nose position. Then, the head pitch $\omega$ is estimated as the angle between the ear-centroid and the eye-centroid, i.e., the average coordinates of eyes and ears detections, respectively. Finally, gaze direction is estimated by rotating $\vec{n}$ with the estimated pitch $\omega$.

The GEOM baseline requires the detection of the nose and at least one eye. Out of the 4782 images composing the GazeFollow *test* set, GEOM is thus restricted to a subset $Set1$ of 4258 images. As summarized on Table 6.4, results obtained on subset $Set1$ demonstrate that our model NET provide gaze estimations on average 23° more accurate than the ones obtained with the simpler baseline. Such a large improvement in performance suggests our network learns a more complex (possibly non-linear) relationship between keypoints and gaze direction. Examples available on Figure 6.11 qualitatively illustrate how the predictions provided by our NET model (in green) are significantly better than the ones provided by the baseline GEOM (in red).

[2]we refer to Appendix B for a detailed formulation of GEOM.

| | *Set1* | *Set2* | *Full* |
|---|---|---|---|
| *No. of images* | 4258 | 4671 | 4782 |
| GEOM | 42.63° | - | - |
| NET0 | 19.52° | 25.70° | - |
| NET | 19.41° | 23.37° | - |
| GF-MODEL [211] | - | - | 24° |

Table 6.4: Comparison in terms of angular errors between our method and baselines on the GazeFollow test set.

GEOM GF-MODEL NET (ours) Avg. annotation

Figure 6.11: Examples of gaze direction estimations provided by the different models evaluated on GazeFollow.

### Comparison against GazeFollow model

Since our network is trained on images where at least two facial keypoints are detected, we apply the same constraint for evaluation. In the test set, OpenPose detects at least two keypoints for a subset *Set*2 containing 97.7% of the 4782 images composing the full set.

The results of our evaluation are summarized in Table 6.4, while qualitative examples are provided in Figure 6.11. As reported in [211], gaze predictions provided by the GF-MODEL present a mean angular error of 24° on the *test* set. Our NET model provides an mean angular error of 23.37° for 97.7% of these images, which

strongly indicates that its performance is on par with GF-MODEL network despite relying solely on the relative position of 5 facial keypoints to predict gaze.

### Impact of using Confidence Gated Units (CGU)

To verify the benefits of applying our proposed CGU blocks to handle absent keypoint detections, i.e., keypoints with 0 confidence score, we evaluated the performance of our model with and without feeding the confidence scores as inputs. We refer to the latter case as NET0, where the CGU blocks composing the input layer are replaced by simple ReLU units initialized in the same way as described in Section 6.2.2.2. Results summarized in Table 6.4 indicate an error decrease of 2.3° when providing confidence scores to an input layer composed of CGUs. In addition to experiments summarized in Table 6.4, we also evaluated a model where the CGU units are replaced by simple additional ReLU units to handle confidence scores. In particular, for the 1536 images where OpenPose detects less than 4 facial keypoints, a significant decrease on angular error is observed when using CGU units: 30.1° mean error, in comparison to 30.9° provided by the model with solely ReLU based input layer.

### Quality of uncertainty estimations

In addition to the overall mean angular error, we also evaluate how accurate are the uncertainty estimations provided by our NET model for its gaze direction predictions. As depicted in Figure 6.12, significantly lower angular errors are observed for gaze predictions accompanied by low uncertainty network predictions. Uncertainties lower than 0.1 are observed for 80% of the *test* set, a subset for which the gaze estimations provided by our NET model are on average off by only 16.5°.

Moreover, the high correlation between uncertainty predictions and angular error ($\rho = 0.56$) is clearly depicted by the plots provided in Figure 6.13. For each

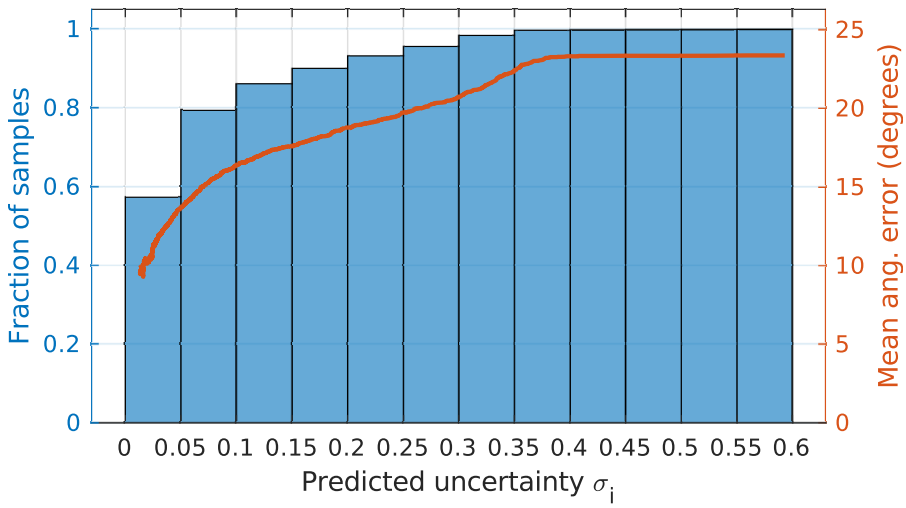



Figure 6.12: Cumulative mean angular error according to uncertainty predicted by our model for each sample [10] (©2020 IEEE).

sample in these plots, the radial distance corresponds to its predicted uncertainty $\sigma_i$, while the angle corresponds to predicted direction of gaze $\tilde{g}$, i.e $\alpha_i = tan^{-1}(-\tilde{g}_y/\tilde{g}_x)$. For both *train* and *test* sets, the associated colormap shows that lower errors (in dark blue) are observed for predictions with lower uncertainty, with increasingly higher errors (green to red) as the uncertainty increases (farther from the center).

### Performance according to keypoint occlusions

Furthermore, the central and the right-most scatter plots in Figure 6.13 also allow an analysis on how the performance of our model and its uncertainty predictions vary according to specific scenarios. For most cases, the number of detected keypoints ($k$) indicates specific scenarios: $k = 2$ is mostly related to back-views, where nose and two other keypoints (both eyes or a pair eye-ear) are missing; $k = 3$ and $k = 4$ are mostly lateral-views; $k = 5$ are frontal-views, where all keypoints are visible. Since images are $2D$ projections from the environment, back- and frontal-views are the ones

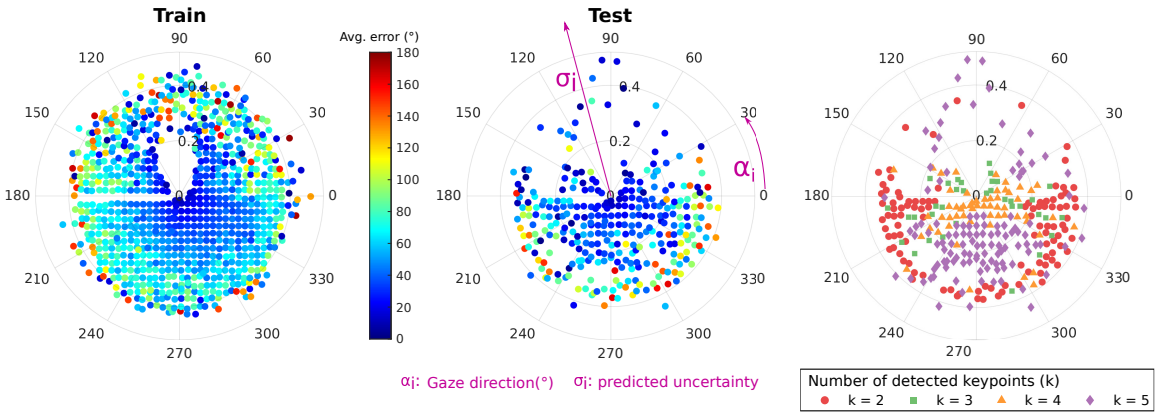



Figure 6.13: Distribution of gaze direction ($\alpha_i$) and uncertainty predictions ($\sigma_i$) provided by our proposed model. *Left/center*: colormap depicts angular error of predictions. *Right*: colors represent the amount of keypoints detected by OpenPose for the corresponding samples. For better visualization, the samples are grouped into equally spaced bins [10] (©2020 IEEE).

more affected by the information loss implicit in the image formation process, while for lateral-views estimation of gaze direction tends to be easier.

An analysis of the scatter plots demonstrates that the predictions provided by our model reflect these expected behaviors. For samples with $k = 2$ (back-view), both uncertainty predictions and angular error tend to be higher, while for most cases of $k = 3$ and $k = 4$ the predictions are associated with lower uncertainty and higher angular accuracy. Predictions for $k = 5$ are spread, indicating that the model's uncertainty predictions are not just defined by the amount of available keypoints but also reflect the intrinsic uncertainty of determining the head orientation from frontal views.

#### 6.2.3.2 Results on the assisted living dataset

**Dataset split and training details**

We compiled a dataset, which we call *MoDiPro*, consisting of 1,060 video frames collected from the two video cameras arraged as shown in Figure 6.1. For $CAM1$, 530 frames were sampled from 46 different video sequences; for $CAM2$, 530 frames were sampled from 27 different video sequences. To limit storage while discarding minimal temporal information, the resolution of the acquired frames was limited to $480 \times 270$ pixels, at 25 fps. In most frames multiple subjects are simultaneously visible, with a total of 22 subjects performing different activities.

As exemplified also in Figure 6.14, cameras $CAM1$ and $CAM2$ cover different parts of the environment. Images acquired with $CAM2$ present significant distortion, which increases the complexity of the task.

We randomly split the available sets of images into camera-specific training, validation and test subsets. Since frames composing the same video sequence can be highly correlated, we opt for a stratified strategy where video sequences are sampled. That is, all frames available from a certain video sequence are assigned to either *train*, *val* or *test* subsets. Aiming at an evaluation that covers a wide variety of scenes, the proportions chosen in terms of total number of frames are: 50% for training, 20% for validation, 30% for testing. Fine-tuning experiments are performed using learning rates $1 \times 10^{-5}$, while $1 \times 10^{-4}$ is adopted when training models only on *MoDiPro* images. Batches with 64 samples are used, with early-stopping based on angular error on the *val* subset. Moreover, all results reported on Table 6.5 and discussed below correspond to average values obtained after train/test on 3 different random splits.

To assess the cross-view performance of our method, we train our NET model with 7 different combinations of images from the *MoDiPro* and GazeFollow datasets.

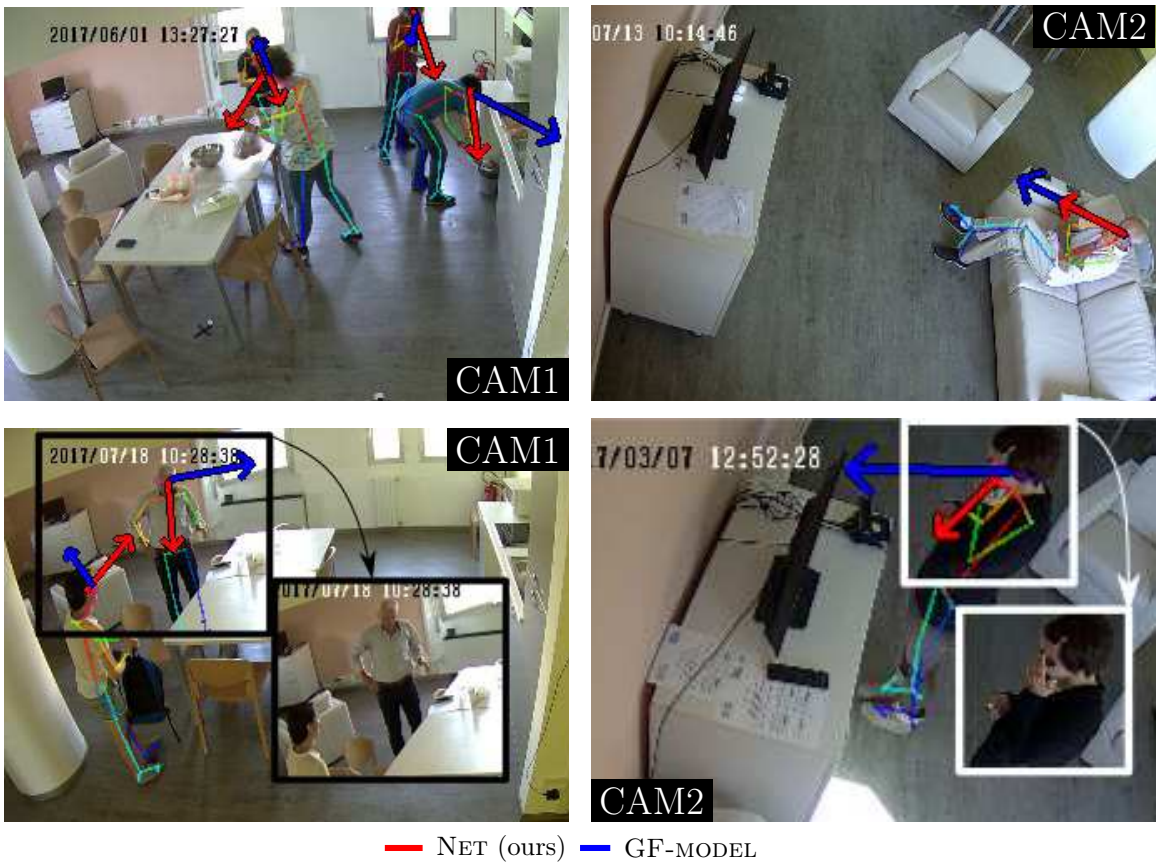



Figure 6.14: Examples of results for our gaze direction estimation approach in the *MoDiPro* dataset [10] (©2020 IEEE).

As summarized in Table 6.5, models NET#0-2 are trained in $CAM1$-only, $CAM2$-only, and both *MoDiPro* cameras. NET#3 corresponds to the model trained only on GazeFollow frames (GF), while NET#4-6 are obtained by fine-tuning the pre-trained NET#3 on each of the three possible sets of *MoDiPro* frames.

**Performance according to camera view**

Cross-view results obtained by NET#0 on $CAM2$ and NET#1 on $CAM1$ demonstrate how models trained only on a camera-specific set of images are less robust to image distortions, with significantly higher angular errors for images composing

| | Train | | | Test | | |
|---|---|---|---|---|---|---|
| Model | GF | *Cam1* | *Cam2* | *Cam1* | *Cam2* | *Mean* |
| Net#0 | | ✓ | | 16.16° | 39.12° | - |
| Net#1 | | | ✓ | 29.56° | 26.37° | - |
| Net#2 | | ✓ | ✓ | 18.52° | 23.02° | 20.94° |
| Net#3 | ✓ | | | 27.64° | 26.98° | 27.31° |
| Net#4 | ✓ | ✓ | | 16.17° | 27.36° | - |
| Net#5 | ✓ | | ✓ | 27.56° | 24.01° | - |
| Net#6 | ✓ | ✓ | ✓ | 17.82° | 20.15° | 19.05° |
| GF-model | ✓ | | | 43.49° | 60.82° | 52.15° |

Table 6.5: Performance of our method on the *MoDiPro* dataset for different combinations of training/testing sets.

unseen subsets. Trained on both $CAM1$ and $CAM2$, the model Net#2 demonstrates a more consistent performance across views. In comparison with the camera specific models, a 3° lower angular error on $CAM2$ is obtained at cost of only 1.4° error increase on $CAM1$.

In addition, error comparisons between models Net#0-2 and Net#4-6 demonstrate that pre-training the model on the GF dataset before fine-tuning on *MoDiPro* images leads to consistently lower mean angular errors, with an optimal performance of 17.82° for $CAM1$ and 20.15° for $CAM2$. This corresponds to an overall average error 1.9° lower than the model Net#2 not pre-trained on GF, while more than 7° better than the model Net#3 trained solely on GF. In terms of camera-specific performance, for $CAM1$ optimal performances with error below 17° are obtained when not training on $CAM2$. On the other hand, predictions for $CAM2$ are significantly better when training is performed using additional $CAM1$ and/or GazeFollow images. We hypothesize the distortions characteristic of $CAM2$ images easily lead to overfitting, thus the advantage of training on additional sets of images. As

a final remark we may notice that overall NET#6 provides the best an most stable result across the two views.

### Comparison against GF-MODEL

Finally, we compare the predictions provided by our NET models to the ones obtained by the publicly available version of GF-MODEL[3]. As summarized in Table 6.5, gaze predictions provided by GF-MODEL on the *MoDiPro* dataset are remarkably worse in terms of angular error than the ones predicted by any of our NET#0-6 models, including the NET#3 also trained only on GF images.

Closer inspection of GF-MODEL predictions suggests two disadvantages of this model with respect to ours when predicting gaze on images from real assisted living environments: i) sensitivity to scale; ii) bias towards salient objects. Images composing the GazeFollow dataset typically show a close-view of the subject of interest, such that only a small surrounding area is covered by the camera view. In contrast, images from assisted living facilities such as the ones in the *MoDiPro* dataset contain subjects covering a much smaller region of the scene, i.e., they are smaller in terms of pixel area. Our NET model benefits from the adopted representation of keypoints, with coordinates centered at the head-centroid and normalized based on the largest distance between centroid and detected keypoints. Moreover, visual inspection of GF-MODEL predictions reveals examples such as the two bottom ones in Figure 6.14: in the left, while our model correctly indicates that the subjects look at each other, GF-MODEL is misled by the saliency of the TV and possibly the window; in the right, the saliency of the TV again misguides the GF-MODEL, while our model properly indicates that the person is looking at the object she is holding.

[3]This version provides 25.8° mean angular error on the GazeFollow test set, in comparison to the 24° reported in [211]

#### 6.2.3.3 Runtime Analysis

Our network requires on average $0.85ms$ per call on a NVIDIA GeForce 970M, with one feedforward execution per person. The overall runtime is thus dominated by OpenPose, which requires $77ms$ on COCO images with a NVIDIA GeForce 1080 Ti (as reported in [198]).

**Acknowledgements** Part of this work has been carried out at the Machine Learning Genoa (MaLGa) center, Università di Genova (IT) thanks to the students mobility supported by Erasmus+ K107. We acknowledge NVIDIA Corporation for the donation of a GPU used for this research.

# CHAPTER 7
# CONCLUSION

In this final chapter, we summarize the main findings and contributions of this dissertation through a brief overview of the novel methods herein introduced. Outcomes are contextualized with respect to the Objectives proposed in Section 1.4, with additional discussion of possible directions for future work.

## 7.1 Objective 1a: segmentation refinement

We have first presented RGR, an effective unsupervised post-processing algorithm for segmentation refinement. Traditionally, the final classification step of existing methods for semantic segmentation consists of thresholding score maps obtained from CNNs. Based on the concepts of Monte Carlo sampling and region growing, our algorithm effectively achieves an adaptive thresholding strategy by exploiting high confidence detections and low-level image information. Experimental results demonstrate the efficacy of RGR refinement, showing increased precision on three different segmentation datasets. Our algorithm provides segmentation improvements competitive with existing state-of-the-art refinement methods, but with the advantage of not requiring any dataset- or model-specific optimization of parameters.

We then introduced pRGR, an updated version of our fully unsupervised RGR algorithm for semantic segmentation refinement. By combining concepts of probability theory, Bayesian estimation, and variance reduction, pRGR not only provides a solid mathematical foundation for RGR, but also further improves the quality of the segmentations obtained after refinement.

Through a Monte Carlo formulation where seed-spacing parameters are sampled in a stratified manner, pRGR evaluates varied receptive field sizes across its multiple region growing iterations of high-confidence seeds. Combined with a strat-

egy where cluster covariances are initialized using conjugate priors and updated as pixel-cluster assignments occur, these new features allow pRGR to refine segmentation masks to significantly higher pixel-accuracy levels. As demonstrated through experiments on the PASCAL and DAVIS datasets using four different configurations from the DeepLab family, segmentation predictions refined with pRGR are improved especially in terms of boundary adherence and removal of false-positive pixel labels.

Moreover, the practical relevance of the proposed algorithm also includes a possible combination with the DenseCRF model to further improve the segmentation quality provided by each of these methods alone, as demonstrated by our experimental results. As future work, we hypothesize that the performance of existing CNNs for semantic segmentation could be improved by training the network with pRGR as an additional step before computing losses. Such an arrangement could lead to a detector module that optimally interacts with pRGR, identifying the most informative points for the refinement process.

## 7.2 Objective 1b: applications of semantic segmentation

### Fruit-flower segmentation

In terms of applications, for the agricultural domain we first introduced a novel approach for fruit flower segmentation that, to the best of our knowledge, was the first to exploit deep learning techniques. In comparison with existing methods, which are mainly based solely on color analysis and have limited applicability in scenarios involving changes in illumination or occlusion levels, the hierarchical features extracted by our CNN effectively combine both color and morphological information, leading to significantly better performance for all the cases under consideration. Experiments performed on four different datasets demonstrated that the proposed CNN-based model allows accurate flower identification even in scenarios of different

flower species and illumination conditions, with optimal recall and precision rates near 80% even for datasets significantly dissimilar from the training sequences.

We then presented a novel automated approach for multispecies flower segmentation that exploits state-of-the-art end-to-end deep learning techniques for semantic image segmentation. The applicability of our method was demonstrated by its high flower segmentation accuracy across datasets that vary in terms of illumination conditions, background composition, image resolution, flower density and flower species. Without any supervised fine-tuning or image pre-processing, our model trained using only images of apple flowers succeeded in generalizing for peach and pear flowers, which are noticeably different in terms of color and morphology.

In the future, we intend to further improve the generalization capabilities of our model by training and evaluating it on multi-species flower datasets. We ultimately aim to devise a completely autonomous system capable of online bloom intensity estimation. The current implementation of our model can evaluate high-resolution images of complete trees an order of magnitude faster than human workers. While in this work we do not create maps of flowers at the block level, this method will scale well for precision agricultural applications such as predicting thinning spray treatments and timing.

**Assisted living environments**

Effective assisted living environments must be able to perform inferences on how their occupants interact with one another as well as with surrounding objects. To accomplish this goal using a vision-based automated approach, multiple tasks such as pose estimation, object segmentation and gaze estimation must be addressed.

In Section 6.1, we proposed a fine semantic video segmentation method for ADL analysis in assisted living applications. Our method employs the RefineNet semantic image segmentation approach in a multi-stream framework that allows the ac-

curate segmentation of multiple objects in videos obtained using wide-angle cameras. Our approach further improves segmentation accuracy by incorporating a superpixel majority voting post-processing mechanism as well as a temporal probabilistic model, which later motivated the development of our RGR algorithm. Preliminary results show that our approach outperforms existing video segmentation methods in publicly available video sequences and performs accurate segmentation in a real-world assisted living facility.

Gaze direction provides some of the strongest indications of how a person interacts with the environment. In Section 6.2, we propose a simple neural network regressor that estimates the gaze direction of individuals in a multi-camera assisted living scenario, relying only on the relative positions of facial keypoints collected from a single pose estimation model. One benefit of exploring a single feature extraction backbone for both pose and gaze estimation is that it reduces the complexity of the overall model for automated analysis of activities of daily living.

To handle cases of keypoint occlusion, our model exploits a novel confidence gated unit in its input layer. Experimental results on a public benchmark demonstrate that our approach performs on par with a complex, dataset-specific baseline. Moreover, experiments on images from a real assisted living environment demonstrate that our model is able to handle such complex scenarios, while results obtained on the GazeFollow dataset demonstrate that our method estimates gaze with accuracy comparable to a complex task-specific baseline, without relying on any image features except the relative positions of facial keypoints.

In contrast to conventional regression methods, our proposed model also provides estimations of uncertainty of its own predictions, with results demonstrating a high correlation between predicted uncertainties and actual gaze angular errors. Analysis of performance according to the number of detected keypoints also indicates

that the proposed Confidence Gated Units improve the model's performance for cases of partial absence of features.

We envision a system that assists clinicians in the assessment of the health status of individuals in an assisted living environment, providing them with automatic reports of patients' mobility and IADL patterns. Thus, to identify human-human and human-object interactions, we plan to combine gaze estimations with our semantic segmentation models. Following the promising preliminary experiments described in Section 6.1.5, we plan to exploit FreeLabel for annotation of novel domain-specific datasets, as well as pRGR for segmentation refinement. We also intend to extend our Monte Carlo models to the temporal domain so that they can predict the expected poses of people and objects in order to improve temporal consistency.

## 7.3 Objective 2: image annotation

In Chapter 4, we introduced FreeLabel, an interactive interface for fast and high-quality annotation of image segmentation datasets. In contrast to annotation tools that require drawing polygons fully enclosing objects to be segmented, FreeLabel simplifies the user interactions to freehand scribbles and straight lines. By means of the proposed RGR algorithm, such inputs are grown into segmentation masks that tightly adhere to actual object boundaries.

We designed FreeLabel with a modular structure that relies solely on open-source libraries, such that it is currently released as a publicly available tool that can be easily adapted for annotation of a wide range of datasets. Thus, in addition to providing better segmentation masks than the ones obtained with public polygon-based interfaces (e.g., the LabelMe interface [128]), it represents an alternative to paid services for annotation of high quality datasets. Its web-based interface can be deployed both locally or in external servers, allowing annotations through both private (confidential) or crowdsourced strategies.

Our experiments demonstrate that segmentations with high overlap to ground-truth annotations of the PASCAL dataset can be obtained in a matter of seconds. Through short tutorial videos and a game-like version of FreeLabel, users quickly learned how to use the tool and were capable of properly annotating significantly different datasets.

As future work, we intend to devise a parallelized, real-time version of the pRGR algorithm and incorporate it into FreeLabel, turning it into an interactive tool that automatically grows user scribbles as they are created. Our preliminary analysis of effective user strategies for annotation indicate that a more responsive interface may reduce annotation time while increasing accuracy. Moreover, we consider integrating FreeLabel, pRGR, and modern semantic segmentation CNNs into a weakly-supervised active learning framework. Similar to the preliminary experiments reported in Section 4.4 for analysis of satellite imagery, we envision a system where predictions of a pre-trained network are used as initial annotations and shown to the users together with associated uncertainty estimations provided by pRGR, indicating which regions should be further annotated.

With minor adjustments, we believe FreeLabel could also be efficiently used with touch-screen devices. Studying annotation efficiency and accuracy as well as effective user strategies in such an interface might provide additional insights into best practices for the annotation of large-scale datasets. Finally, we plan to perform larger scale image annotation experiments using AMT. Feedback received from five AMT workers in a preliminary experiment included encouraging comments such as *"I was surprised how well the bounding tools worked. They seemed to accurately pick up my responses"*, and *"the interface was easy to understand for anyone mildly familiar with MS paint"*. However, it is still unclear which mechanisms would be effective to motivate a large number of workers to perform such a complex and detail-oriented

task, which requires, even with the assistance of semi-automated tools, significant attention to detail.

## 7.4 Objective 3: uncertainty estimation

Finally, this dissertation also described efficient mechanisms for equipping deep learning-based frameworks with the ability of uncertainty estimation. More specifically, it described two effective strategies for estimation of heteroscedastic (i.e., data-dependent) uncertainties.

First, in the context of image semantic segmentation we demonstrated how pRGR can be also exploited to generate accurate estimates of segmentation uncertainty at pixel-level. Experiments have shown that the pixel-wise variances estimated using the Monte Carlo framework underlying the design of pRGR show a strong inverse correlation with segmentation accuracy values. In other words, pRGR variance estimates can be exploited for uncertainty estimation of segmentation predictions, which expands its range of applications to scenarios such as active learning [221], human-in-the-loop systems for image labeling [222], and semi- or weakly-supervised methods for image segmentation [223, 224].

Second, by adapting for the scenario of angular regression the approach introduced in [109], we have demonstrated the effectiveness of exploiting a customized loss function to design a gaze regression neural network that is capable of estimating its own predictive uncertainties.

In the future, we plan to investigate strategies for uncertainty calibration [110], as well as techniques for estimation of epistemic uncertainties. In particular, our framework is based on similar fundamental principles that guide the design of Monte Carlo dropout [113, 213] and model ensemble techniques [112]. A comprehensive framework that encompasses all these strategies would allow for the estimation of different types of uncertainty in a unified and theoretically sound manner.

# BIBLIOGRAPHY


[1] P. A. Dias and H. Medeiros, "Semantic segmentation refinement by monte carlo region growing of high confidence detections," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 131–146.

[2] P. A. Dias, A. Tabb, and H. Medeiros, "Multispecies fruit flower detection using a refined semantic segmentation network," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3003–3010, 2018.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010. [Online]. Available: http://link.springer.com/10.1007/s11263-009-0275-4

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

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[6] H. Caesar, J. Uijlings, and V. Ferrari, "COCO-Stuff: Thing and Stuff Classes in Context," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1209–1218.

[7] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung, "A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 724–732.

[8] P. A. Dias, Z. Shen, A. Tabb, and H. Medeiros, "FreeLabel: A Publicly Available Annotation Tool based on Freehand Traces," in *Winter Conference on Applciations of Computer Vision (WACV)*, 2019.

[9] P. A. Dias, A. Tabb, and H. Medeiros, "Apple flower detection using deep convolutional networks," *Computers in Industry*, vol. 99, pp. 17–28, 2018.

[10] P. A. Dias, D. Malafronte, H. Medeiros, and F. Odone, "Gaze Estimation for Assisted Living Environments," in *Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[11] M. Ma, H. Fan, and K. M. Kitani, "Going deeper into first-person activity recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1894–1903.

[12] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[13] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art," *arXiv preprint arXiv:1704.05519*, vol. abs/1704.05519, 2017.

[14] A. Siddique and H. Medeiros, "Tracking passengers and baggage items using multi-camera systems at security checkpoints," *arXiv preprint arXiv:2007.07924*, 2020.

[15] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[16] A. F. Echeverri, H. Medeiros, R. Walsh, Y. Reznichenko, and R. Povinelli, "Real-time hierarchical bayesian data fusion for vision-based target tracking with unmanned aerial platforms," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 1262–1270.

[17] L. Shao, L. Ji, Y. Liu, and J. Zhang, "Human action segmentation and recognition via motion and shape analysis," *Pattern Recognition Letters*, vol. 33, no. 4, pp. 438–445, 2012.

[18] A. Gupta, A. Kembhavi, and L. S. Davis, "Observing human-object interactions: Using spatial and functional compatibility for recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 10, pp. 1775–1789, 2009.

[19] Z. Wang, Z. Li, B. Wang, and H. Liu, "Robot grasp detection using multimodal deep convolutional neural networks," *Advances in Mechanical Engineering*, vol. 8, no. 9, p. 1687814016668077, 2016.

[20] A. A. Shvets, A. Rakhlin, A. A. Kalinin, and V. I. Iglovikov, "Automatic instrument segmentation in robot-assisted surgery using deep learning," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 624–628.

[21] X. Han, Z. Wu, Z. Wu, R. Yu, and L. S. Davis, "Viton: An image-based virtual try-on network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7543–7552.

[22] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, April 2018.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[24] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully Convolutional Instance-aware Semantic Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2359–2367.

[25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[26] A. Gongal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis, "Sensors and systems for fruit detection and localization: A review," pp. 8–19, 2015.

[27] P. A. Dias, A. Tabb, and H. Medeiros, "Data from: Multi-species fruit flower detection using a refined semantic segmentation network," 2018. [Online]. Available: https://data.nal.usda.gov/dataset/data-multi-species-fruit-flower-detection-using-refined-semantic-segmentation-network

[28] O. Marques, *Practical image and video processing using MATLAB*. John Wiley & Sons, 2011.

[29] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.

[30] M. A. Stricker and M. Orengo, "Similarity of color images," in *Storage and Retrieval for Image and Video Databases III*, vol. 2420. International Society for Optics and Photonics, 1995, pp. 381–393.

[31] I. C. Consortium *et al.*, "Image technology colour management-architecture, profile format, and data structure," *Specification ICC. 1: 2004-10 (Profile version 4.2. 0.0)*, 2004.

[32] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[33] M. Mohri, "Foundations of Machine Learning - Book," *The MIT Press*, 2012.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[35] C. M. Bishop, *Pattern recognition and machine learning.* springer, 2006.

[36] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[37] A. Ben-Hur and J. Weston, "A user's guide to support vector machines." *Methods in molecular biology (Clifton, N.J.)*, vol. 609, pp. 223–239, 2010.

[38] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A Practical Guide to Support Vector Classification," *BJU international*, vol. 101, no. 1, pp. 1396–400, 2008.

[39] L. I. Smith, "A tutorial on Principal Components Analysis Introduction," *Statistics*, vol. 51, p. 52, 2002.

[40] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[41] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[42] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014.

[43] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015. [Online]. Available: http://link.springer.com/10.1007/s11263-014-0733-5

[44] G. Csurka, "Domain adaptation for visual applications: A comprehensive survey," *arXiv preprint arXiv:1702.05374*, 2017.

[45] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[47] G. A. Miller, *WordNet: An electronic lexical database.* MIT press, 1998.

[48] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 davis challenge on video object segmentation," *arXiv:1704.00675*, 2017.

[49] S. Caelles, A. Montes, K.-K. Maninis, Y. Chen, L. Van Gool, F. Perazzi, and J. Pont-Tuset, "The 2018 davis challenge on video object segmentation," *arXiv:1803.00557*, 2018.

[50] S. Caelles, J. Pont-Tuset, F. Perazzi, A. Montes, K.-K. Maninis, and L. Van Gool, "The 2019 davis challenge on vos: Unsupervised multi-object segmentation," *arXiv:1905.00737*, 2019.

[51] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[52] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *European Conference on Computer Vision (ECCV).* Springer, 2016, pp. 519–534.

[53] D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[54] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22. Prague, 2004, pp. 1–2.

[55] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, jun 2005, pp. 886–893 vol. 1.

[56] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1. IEEE, 2001, pp. I–I.

[57] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[58] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[59] W. Xia, Z. Song, J. Feng, L.-F. Cheong, and S. Yan, "Segmentation over detection by coupled global and local sparse representations," in *European Conference on Computer Vision*. Springer, 2012, pp. 662–675.

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

[61] D. Stutz, A. Hermans, and B. Leibe, "Superpixels: an evaluation of the state-of-the-art," *Computer Vision and Image Understanding*, vol. 166, pp. 1–27, 2018.

[62] J. Yao, M. Boben, S. Fidler, and R. Urtasun, "Real-time coarse-to-fine topologically preserving segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2947–2955.

[63] O. Freifeld, Y. Li, and J. W. Fisher, "A fast method for inferring high-quality simply-connected superpixels," in *2015 IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 2184–2188.

[64] Z. Ban, J. Liu, and L. Cao, "Superpixel segmentation using gaussian mixture model," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4105–4117, 2018.

[65] R. Uziel, M. Ronen, and O. Freifeld, "Bayesian adaptive superpixel segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8470–8479.

[66] Y. Boykov and G. Funka-Lea, "Graph Cuts and Efficient N-D Image Segmentation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109–131, nov 2006. [Online]. Available: http://link.springer.com/10.1007/s11263-006-7934-5

[67] Y. Y. Boykov and M. P. Jolly, "Interactive graph cuts for optimal boundary amp; region segmentation of objects in N-D images," in *IEEE International Conference on Computer Vision*, vol. 1, 2001, pp. 105—-112 vol.1.

[68] D. Freedman and T. Zhang, “Interactive graph cut based segmentation with shape priors,” in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, jun 2005, pp. 755—-762 vol. 1.

[69] C. Rother, V. Kolmogorov, and A. Blake, “" grabcut" interactive foreground extraction using iterated graph cuts,” *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.

[70] E. N. Mortensen and W. A. Barrett, “Intelligent scissors for image composition,” in *Computer Graphics and Interactive Techniques*. ACM, 1995, pp. 191–198.

[71] D. Cremers, M. Rousson, and R. Deriche, “A Review of Statistical Approaches to Level Set Segmentation: Integrating Color, Texture, Motion and Shape,” *International Journal of Computer Vision*, vol. 72, no. 2, pp. 195–215, apr 2007. [Online]. Available: https://link.springer.com/article/10.1007/s11263-006-8711-1

[72] J. Cates, A. Lefohn, and R. Whitaker, “GIST: an interactive, GPU-based level set segmentation tool for 3D medical images,” *Medical Image Analysis*, vol. 8, no. 3, pp. 217–231, sep 2004. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1361841504000246

[73] D. Cremers, O. Fluck, M. Rousson, and S. Aharon, “A probabilistic level set formulation for interactive organ segmentation,” in *Medical Imaging 2007: Image Processing*, vol. 6512. International Society for Optics and Photonics, mar 2007, p. 65120V.

[74] Y. Liu and Y. Yu, “Interactive Image Segmentation Based on Level Sets of Probabilities,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 2, pp. 202–213, feb 2012.

[75] T. Wang, B. Han, and J. Collomosse, “Touchcut: Fast image and video segmentation using single-touch interaction,” *Computer Vision and Image Understanding*, vol. 120, pp. 14–30, 2014.

[76] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski, “A Bayesian Approach to Digital Matting,” in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2001, pp. 264–271.

[77] Q. Chen, D. Li, and C.-K. Tang, “KNN matting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2175–2188, 2013.

[78] Q. Zhu, L. Shao, X. Li, and L. Wang, "Targeting accurate object extraction from an image: A comprehensive study of natural image matting," *IEEE Transactions on Neural networks and Learning Systems*, vol. 26, no. 2, pp. 185–207, 2015.

[79] M. Rajchl, M. C. H. Lee, O. Oktay, K. Kamnitsas, J. Passerat-Palmbach, W. Bai, M. Damodaram, M. A. Rutherford, J. V. Hajnal, B. Kainz, and D. Rueckert, "DeepCut: Object Segmentation From Bounding Box Annotations Using Convolutional Neural Networks," *IEEE Transactions on Medical Imaging*, vol. 36, no. 2, pp. 674–683, feb 2017.

[80] A. Kolesnikov and C. H. Lampert, "Seed, expand and constrain: Three principles for weakly-supervised image segmentation," in *European Conference on Computer Vision*. Springer, 2016, pp. 695–711.

[81] D. Lin, J. Dai, J. Jia, K. He, and J. Sun, "Scribblesup: Scribble-supervised convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3159–3167.

[82] M. Tang, A. Djelouah, F. Perazzi, Y. Boykov, and C. Schroers, "Normalized Cut Loss for Weakly-Supervised CNN Segmentation," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. IEEE Confe, p. 10, 2018.

[83] S. Mahadevan, P. Voigtlaender, and B. Leibe, "Iteratively Trained Interactive Segmentation," in *British Machine Vision Conference*, 2018.

[84] Y. Aksoy, T.-H. Oh, S. Paris, M. Pollefeys, and W. Matusik, "Semantic soft segmentation," *ACM Transactions on Graphics*, vol. 37, no. 4, p. 72, 2018.

[85] Y. Chen, J. Pont-Tuset, A. Montes, and L. Van Gool, "Blazingly Fast Video Object Segmentation with Pixel-Wise Metric Learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1189–1198.

[86] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, 2014.

[87] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 91–99.

[88] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *International Conference on Learning Representations (ICLR)*, 2015.

[89] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *IEEE international conference on computer vision*, 2015, pp. 2650–2658.

[90] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, 2015, pp. 3431–3440.

[91] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied Soft Computing*, vol. 70, pp. 41–65, 2018.

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

[93] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for scene segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[94] G. Papandreou, I. Kokkinos, and P.-A. Savalle, "Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 390–399.

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

[96] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[97] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2. IEEE, 2005, pp. 1458–1465.

[98] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," *arXiv preprint arXiv: 1706.05587*, 2017.

[99] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[100] J. Dai, K. He, Y. Li, S. Ren, and J. Sun, "Instance-sensitive fully convolutional networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 534–549.

[101] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller, "Multi-class segmentation with relative location prior," *International Journal of Computer Vision*, vol. 80, no. 3, pp. 300–316, 2008.

[102] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2012.

[103] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, "Feedforward semantic segmentation with zoom-out features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3376–3385.

[104] P. Krähenbühl and V. Koltun, "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials," in *Advances in neural information processing systems*, 2011, pp. 109–117.

[105] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. L. Yuille, "Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4545–4554.

[106] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[107] A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin, *Bayesian Data Analysis, Third Edition*, ser. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013.

[108] A. B. Owen, *Monte Carlo theory, methods and examples*, 2013.

[109] A. Kendall and Y. Gal, "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5574–5584.

[110] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, 2017, pp. 1321–1330.

[111] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[112] W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler, "The Power of Ensembles for Active Learning in Image Classification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9368–9377.

[113] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in neural information processing systems*, 2017, pp. 3581–3590.

[114] H. Williams, M. Nejati, S. Hussein, N. Penhall, J. Y. Lim, M. H. Jones, J. Bell, H. S. Ahn, S. Bradley, P. Schaare, P. Martinsen, M. Alomar, P. Patel, M. Seabright, M. Duke, A. Scarfe, and B. MacDonald, "Autonomous pollination of individual kiwifruit flowers: Toward a robotic kiwifruit pollinator," *Journal of Field Robotics*, vol. 37, no. 2, pp. 246–262, 2020.

[115] Q. Zhang, M. Karkee, and A. Tabb, "The use of agricultural robots in orchard management," in *Robotics and automation for improving agriculture*. Burleigh Dodds Science Publishing, 2019, pp. 187–214.

[116] P. A. Dias and H. Medeiros, "Probabilistic semantic segmentation refinement by monte carlo region growing," *arXiv preprint arXiv:2005.05856*, 2020.

[117] R. Achanta and S. Sabine, "Superpixels and Polygons using Simple Non-Iterative Clustering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4651–4660.

[118] P. M. Lee, *Bayesian Statistics: An Introduction*, 4th ed. Wiley Publishing, 2012.

[119] K. P. Murphy, "Conjugate bayesian analysis of the gaussian distribution," Tech. Rep., 2007. [Online]. Available: https://www.cs.ubc.ca/~murphyk/Papers/bayesGauss.pdf

[120] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR (Presented at International Conference on Learning Representations, 2015)*, vol. abs/1409.1556, 2014.

[121] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[122] J. Mukhoti and Y. Gal, "Evaluating bayesian deep learning methods for semantic segmentation," *arXiv preprint arXiv:1811.12709*, 2018.

[123] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[124] M. V. Giuffrida, F. Chen, H. Scharr, and S. A. Tsaftaris, "Citizen crowds and experts: observer variability in image-based plant phenotyping," *Plant Methods*, vol. 14, p. 12, feb 2018. [Online]. Available: https://doi.org/10.1186/s13007-018-0278-7

[125] S. W. Chen, S. S. Shivakumar, S. Dcunha, J. Das, E. Okon, C. Qu, C. J. Taylor, and V. Kumar, "Counting apples and oranges with deep learning: a data-driven approach," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 781–788, 2017.

[126] P. Tangseng, Z. Wu, and K. Yamaguchi, "Looking at outfit to parse clothing," *arXiv preprint arXiv:1703.01386*, 2017.

[127] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "OpenSurfaces: A richly annotated catalog of surface appearance," *ACM Transactions on Graphics*, vol. 32, no. 4, p. 111, 2013.

[128] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[129] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes dataset for semantic urban scene understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

[130] K. K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool, "Deep Extreme Cut: From Extreme Points to Object Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[131] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation: A set of best practices for high quality, economical

video labeling," *International Journal of Computer Vision*, vol. 101, no. 1, pp. 184–204, 2013.

[132] B. Schwartz, *The Paradox of Choice: Why More Is Less*, ser. Harper Perennial. HarperCollins, 2003. [Online]. Available: https://books.google.com/books?id=zutxr7rGc{_}QC

[133] L. Von Ahn and L. Dabbish, "Labeling images with a computer game," in *SIGCHI conference on Human factors in computing systems*. ACM, 2004, pp. 319–326.

[134] L. Von Ahn, R. Liu, and M. Blum, "Peekaboom: a game for locating objects in images," in *SIGCHI conference on Human Factors in computing systems*. ACM, 2006, pp. 55–64.

[135] L. Von Ahn, M. Kedia, and M. Blum, "Verbosity: a game for collecting commonsense facts," in *SIGCHI conference on Human Factors in computing systems*. ACM, 2006, pp. 75–78.

[136] A. Kawrykow, G. Roumanis, A. Kam, D. Kwak, C. Leung, C. Wu, E. Zarour, L. Sarmenta, M. Blanchette, J. Waldispühl, and Others, "Phylo: a citizen science approach for improving multiple sequence alignment," *PloS one*, vol. 7, no. 3, p. e31362, 2012.

[137] L. von Ahn and L. Dabbish, "Designing games with a purpose," *Communications of the ACM*, vol. 51, no. 8, p. 57, 2008.

[138] "Django (version 2.0)," https://djangoproject.com/, accessed: 2018-09-13.

[139] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[140] R. Munro, "Human-in-the-loop machine learning," 2019.

[141] E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox, "Uncertainty estimates and multi-hypotheses networks for optical flow," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 652–667.

[142] F. Ozdemir, Z. Peng, C. Tanner, P. Fuernstahl, and O. Goksel, "Active learning for segmentation by optimizing content information for maximal entropy," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer, 2018, pp. 183–191.

[143] H. L. Yang, J. Yuan, D. Lunga, M. Laverdiere, A. Rose, and B. Bhaduri, "Building extraction at scale using convolutional neural network: Mapping of the united states," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 8, pp. 2600–2614, 2018.

[144] A. Van Etten, D. Lindenbaum, and T. M. Bacastow, "Spacenet: A remote sensing dataset and challenge series," *arXiv preprint arXiv:1807.01232*, 2018.

[145] "Spacenet on amazon web services (aws). last modified october 1st, 2018." https://spacenet.ai/datasets/, accessed: 2020-10-13.

[146] C. G. Forshey, *Chemical fruit thinning of apples.* Geneva, N.Y.: New York State Agricultural Experiment Station, 1986.

[147] H. Link, "Significance of flower and fruit thinning on fruit quality," *Plant growth regulation*, vol. 31, no. 1, pp. 17–26, 2000.

[148] A. Gongal, A. Silwal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis, "Apple crop-load estimation with over-the-row machine vision system," *Computers and Electronics in Agriculture*, vol. 120, pp. 26–35, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.compag.2015.10.022

[149] S. Singh, M. Bergerman, J. Cannons, B. Grocholsky, B. Hamner, G. Holguin, L. Hull, V. Jones, G. Kantor, H. Koselka, G. Li, J. Owen, J. Park, W. Shi, and J. Teza, "Comprehensive Automation for Specialty Crops: Year 1 results and lessons learned," *Intelligent Service Robotics*, vol. 3, no. 4, pp. 245–262, 2010.

[150] N. Zhang, M. Wang, and N. Wang, "Precision agriculture – a worldwide overview," *Computers and Electronics in Agriculture*, vol. 36, no. 2-3, pp. 113–132, 2002.

[151] K. Kapach, E. Barnea, R. Mairon, Y. Edan, and O. B. Shahar, "Computer vision for fruit harvesting robots state of the art and challenges ahead," *International Journal of Computational Vision and Robotics*, vol. 3, no. 1/2, p. 4, 2012.

[152] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[153] M. Dyrmann, A. K. Mortensen, H. S. Midtiby, and R. N. Jørgensen, "Pixel-wise classification of weeds and crops in images by using a fully convolutional neural network," in *Proceedings of the International Conference on Agricultural Engineering, Aarhus, Denmark*, 2016, pp. 26–29.

[154] G. L. Grinblat, L. C. Uzal, M. G. Larese, and P. M. Granitto, "Deep learning for plant identification using vein morphological patterns," *Computers and Electronics in Agriculture*, vol. 127, pp. 418–424, 2016.

[155] A. D. Aggelopoulou, D. Bochtis, S. Fountas, K. C. Swain, T. A. Gemtos, and G. D. Nanos, "Yield prediction in apple orchards based on image processing," *Precision Agriculture*, vol. 12, no. 3, pp. 448–456, 2011.

[156] K. R. Thorp and D. A. Dierig, "Color image segmentation approach to monitor flowering in lesquerella," *Industrial Crops and Products*, vol. 34, no. 1, pp. 1150–1159, 2011.

[157] M. Hočevar, B. Širok, T. Godeša, and M. Stopar, "Flowering estimation in apple orchards by image analysis," *Precision Agriculture*, vol. 15, no. 4, pp. 466–478, 2014.

[158] R. Horton, E. Cano, D. Bulanon, and E. Fallahi, "Peach Flower Monitoring Using Aerial Multispectral Imaging," *2016 ASABE International Meeting*, 2016.

[159] C. Hung, J. Nieto, Z. Taylor, J. Underwood, and S. Sukkarieh, "Orchard fruit segmentation using multi-spectral feature learning," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5314–5320, 2013.

[160] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, and V. Kumar, "Devices, systems, and methods for automated monitoring enabling precision agriculture," in *IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2015, pp. 462–469.

[161] W. Ji, D. Zhao, F. Cheng, B. Xu, Y. Zhang, and J. Wang, "Automatic recognition vision system guided for apple harvesting robot," *Computers and Electrical Engineering*, vol. 38, no. 5, pp. 1186–1195, 2012.

[162] R. Linker, O. Cohen, and A. Naor, "Determination of the number of green apples in RGB images recorded in orchards," *Computers and Electronics in Agriculture*, vol. 81, pp. 45–57, 2012.

[163] J. P. Wachs, H. I. Stern, T. Burks, and V. Alchanatis, "Low and high-level visual feature-based apple detection from multi-modal images," *Precision Agriculture*, vol. 11, no. 6, pp. 717–735, 2010.

[164] Q. Wang, S. Nuske, M. Bergerman, and S. Singh, "Automated crop yield estimation for apple orchards," in *Experimental robotics*. Springer, 2013, pp. 745–758.

[165] U.-O. Dorj, M. Lee, and S.-s. Yun, "An yield estimation in citrus orchards via fruit detection and counting using image processing," *Computers and Electronics in Agriculture*, vol. 140, pp. 103–112, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169916312455

[166] S. Bargoti and J. Underwood, "Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards," *Journal of Field Robotics*, vol. 00, no. 0, pp. 1–22, 2016.

[167] H. Cheng, L. Damerow, Y. Sun, and M. Blanke, "Early Yield Prediction Using Image Analysis of Apple Fruit and Tree Canopy Features with Neural Networks," *Journal of Imaging*, vol. 3, no. 1, p. 6, 2017.

[168] S. Bargoti and J. Underwood, "Deep Fruit Detection in Orchards," *Australian Centre for Field Robotics*, pp. 1–8, 2016.

[169] M. Stein, S. Bargoti, and J. Underwood, "Image Based Mango Fruit Detection, Localisation and Yield Estimation Using Multiple View Geometry," *Sensors*, vol. 16, no. 11, p. 1915, 2016. [Online]. Available: http://www.mdpi.com/1424-8220/16/11/1915

[170] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, "DeepFruits: A Fruit Detection System Using Deep Neural Networks," *Sensors*, vol. 16, no. 8, p. 1222, 2016.

[171] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 1265–1274, 2015.

[172] S. Visa, "Issues in mining imbalanced data sets - a review paper," *in Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference, 2005*, pp. 67–73, 2005.

[173] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[174] A. Carpenter, "Cusvm: a cuda implementation of support vector classification and regression," *Update*, vol. 15, no. 10, pp. 1–9, 2009.

[175] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bulletin of the Calcutta Mathematical Society*, vol. 35, pp. 99–109, 1943.

[176] A. Hoak, H. Medeiros, and R. J. Povinelli, "Image-based multi-target tracking through multi-bernoulli filtering with interactive likelihoods," *Sensors*, vol. 17, no. 3, p. 501, 2017.

[177] R. Hoseinnezhad, B. N. Vo, B. T. Vo, and D. Suter, "Visual tracking of numerous targets via multi-Bernoulli filtering of image data," *Pattern Recognition*, vol. 45, no. 10, pp. 3625–3635, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2012.04.004

[178] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan 1979.

[179] A. Tabb and H. Medeiros, "Automatic segmentation of trees in dynamic outdoor environments," *Computers in Industry*, vol. 98, pp. 90–99, 2018.

[180] T. U. Nations, "World population ageing," http://www.un.org/en/development/desa/population/publications/pdf/ageing/WPA2017_Report.pdf, 2017, accessed: 2018-09-03.

[181] L. P. Fried, L. Ferrucci, J. Darer, J. D. Williamson, and G. Anderson, "Untangling the concepts of disability, frailty, and comorbidity: implications for improved targeting and care," *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences*, vol. 59, no. 3, pp. M255–M263, 2004.

[182] A. Pilotto, L. Ferrucci, M. Franceschi, L. P. D'Ambrosio, C. Scarcelli, L. Cascavilla, F. Paris, G. Placentino, D. Seripa, B. Dallapiccola *et al.*, "Development and validation of a multidimensional prognostic index for one-year mortality from comprehensive geriatric assessment in hospitalized older patients," *Rejuvenation research*, vol. 11, no. 1, pp. 151–161, 2008.

[183] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, and A. Bauer, "Monitoring activities of daily living in smart homes: Understanding human behavior," *IEEE Signal Processing Magazine*, vol. 33, no. 2, pp. 81–94, 2016.

[184] P. Dias, H. Medeiros, and F. Odone, "Fine segmentation for Activity of Daily Living analysis in a wide-angle multi-camera set-up," in *5th Activity Monitoring by Multiple Distributed Sensing Workshop (AMMDS) in conjunction with British Machine Vision Conference*, 2017.

[185] C. Martini, N. Noceti, M. Chessa, A. Barla, A. Cella, G. A. Rollandi, A. Pilotto, A. Verri, and F. Odone, "La visual computing approach for estimating the motility index in the frail elder," *13th International Joint Conference on*

*Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2018.

[186] M. Chessa, N. Noceti, C. Martini, F. Solari, and F. Odone, "Design of assistive tools for the market," in *Assistive Computer Vision*, M. Leo and G. Farinella, Eds. Elsevier, 2017.

[187] C. Martini, A. Barla, F. Odone, A. Verri, G. A. Rollandi, and A. Pilotto, "Data-driven continuous assessment of frailty in older people," *Frontiers in Digital Humanities*, vol. 5, p. 6, 2018.

[188] F. Li, T. Kim, A. Humayun, D. Tsai, and J. M. Rehg, "Video segmentation by tracking many figure-ground segments," in *ICCV*, 2013.

[189] N. Maerki, F. Perazzi, O. Wang, and A. Sorkine-Hornung, "Bilateral Space Video Segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[190] Y.-H. Tsai, M.-H. Yang, and M. J. Black, "Video Segmentation via Object Flow," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[191] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, "One-Shot Video Object Segmentation," in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[192] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A.Sorkine-Hornung, "Learning video object segmentation from static images," in *Computer Vision and Pattern Recognition*, 2017.

[193] V. Jampani, R. Gadde, and P. V. Gehler, "Video Propagation Networks," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, july 2017.

[194] V. Jampani, M. Kiefel, and P. V. Gehler, "Learning sparse high dimensional filters: Image Filtering, Dense CRFs and Bilateral Neural Networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[195] A. Faktor and M. Irani, "Video segmentation by non-local consensus voting," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[196] P. Majaranta and A. Bulling, "Eye tracking and eye-based human–computer interaction," in *Advances in physiological computing*. Springer, 2014, pp. 39–65.

[197] J. Varadarajan, R. Subramanian, S. R. Bulò, N. Ahuja, O. Lanz, and E. Ricci, "Joint estimation of human pose and conversational groups from social scenes," *International Journal of Computer Vision*, vol. 126, no. 2, pp. 410–429, Apr 2018.

[198] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[199] X. Zhang, C. Li, X. Tong, W. Hu, S. Maybank, and Y. Zhang, "Efficient human pose estimation via parsing a tree structure based human model," in *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[200] M. Brubaker, D. Fleet, and A. Hertzmann, "Physics-based human pose tracking," in *NIPS Workshop on Evaluation of Articulated Human Motion and Pose Estimation*, 2006.

[201] S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[202] T. Baltrušaitis, P. Robinson, and L. Morency, "Openface: an open source facial behavior analysis toolkit," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–10.

[203] A. T. Lopes, E. de Aguiar, A. F. D. Souza, and T. Oliveira-Santos, "Facial expression recognition with convolutional neural networks: Coping with few data and the training sample order," *Pattern Recognition*, vol. 61, pp. 610 – 628, 2017.

[204] K. Zhang, Y. Huang, Y. Du, and L. Wang, "Facial expression recognition based on deep evolutional spatial-temporal networks," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4193–4203, Sept 2017.

[205] J. Jayalekshmi and T. Mathew, "Facial expression recognition and emotion classification system for sentiment analysis," in *2017 International Conference on Networks Advances in Computational Technologies (NetACT)*, 2017, pp. 1–8.

[206] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[207] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "It's written all over your face: Full-face appearance-based gaze estimation," in *IEEE Conference on Computer*

*Vision and Pattern Recognition Workshops (CVPRW).* IEEE, 2017, pp. 2299–2308.

[208] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 607–626, 2009.

[209] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[210] K. A. Funes Mora, F. Monay, and J.-M. Odobez, "EYEDIAP: A Database for the Development and Evaluation of Gaze Estimation Algorithms from RGB and RGB-D Cameras," in *ACM Symposium on Eye Tracking Research and Applications.* ACM, Mar. 2014.

[211] A. Recasens, A. Khosla, C. Vondrick, and A. Torralba, "Where are they looking?" in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[212] E. Chong, N. Ruiz, Y. Wang, Y. Zhang, A. Rozga, and J. M. Rehg, "Connecting gaze, scene, and attention: Generalized attention estimation via joint modeling of gaze and scene saliency," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 383–398.

[213] Y. Gal, "Uncertainty in deep learning," Ph.D. dissertation.

[214] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *arXiv preprint arXiv:1511.02680*, 2015.

[215] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[216] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[217] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE International Conference on Computer Vision (ICCV).* IEEE Computer Society, 2015, pp. 1026–1034.

[218] "Lecture notes on "the exponential family: Basics"," https://people.eecs.berkeley.edu/~jordan/courses/260-spring10/other-readings/chapter8.pdf, accessed: 2020-10-13.

[219] D. Kinga and J. B. Adam, "A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[220] A. Gee and R. Cipolla, "Determining the gaze of faces in images," *Image and Vision Computing*, vol. 12, no. 10, pp. 639–647, 1994.

[221] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *34th International Conference on Machine Learning-Volume 70*, 2017, pp. 1183–1192.

[222] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool, "Deep extreme cut: From extreme points to object segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 616–625.

[223] A. Kolesnikov and C. H. Lampert, "Seed, expand and constrain: Three principles for weakly-supervised image segmentation," in *European Conference on Computer Vision*. Springer, 2016, pp. 695–711.

[224] G. Song, H. Myeong, and K. Mu Lee, "Seednet: Automatic seed generation with deep reinforcement learning for robust interactive segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1760–1768.

# APPENDIX A
# REGION GROWING REFINEMENT - SUPPLEMENTARY MATERIAL

## A.1 Qualitative examples - MS COCO 2016 validation set

In addition to the examples provided in our main document, we provide in Figure A.1 further illustrations of refined detections on the MS COCO 2016 validation set.

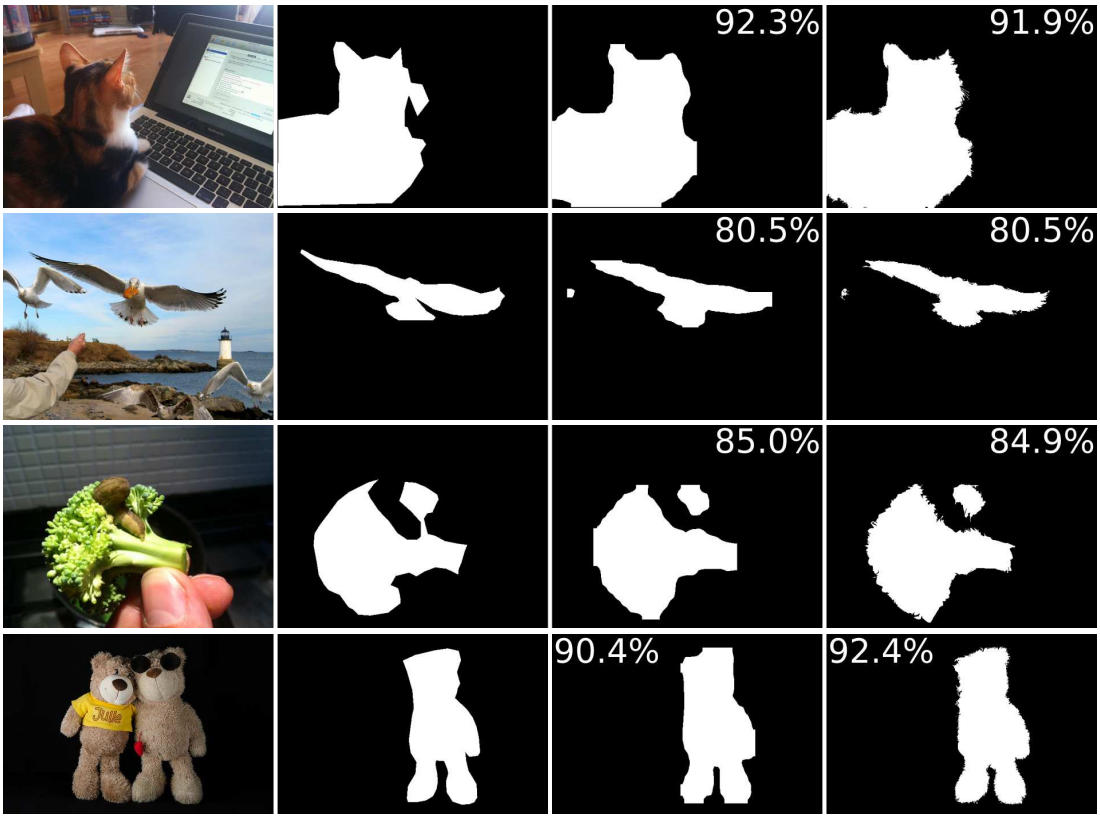



Figure A.1: Additional examples of detections on the COCO 2016 dataset. From left to right: original image, ground truth, FCIS detection, FCIS+RGR. The obtained $IoU$ with respect to the ground-truth is displayed above each corresponding detection.

Our method RGR (Region Growing Refinement) significantly increases the segmentation adherence to object boundaries: in the first example, the coarser segmentation provided by FCIS is refined to such an extent that even the cat whiskers are properly segmented. This behavior is also noticeable for the bird's feathers, the buds composing the broccoli crown, and the teddy bear's fur.

Figure A.2 illustrates how RGR refinement can also recover object details from small and coarse input segmentations. While the original FCIS segmentation consists of three disconnected blobs, RGR reconstructs the toothbrush shape with some finer details on the object's extremities.

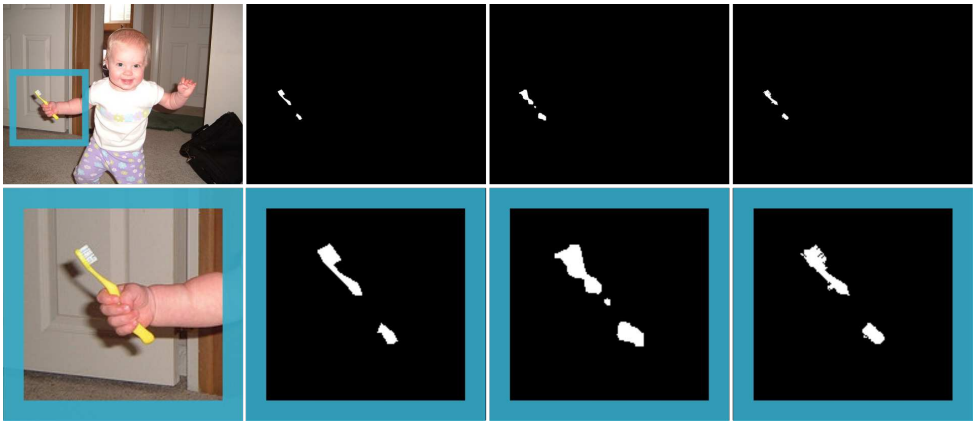


Figure A.2: Noteworthy example of segmentation refinement on the COCO 2016 dataset. From left to right: original image, ground truth, FCIS detection, FCIS+RGR. Our method is capable of recovering object details from small and coarse input segmentations.

## A.2 Qualitative examples - DAVIS 2016 dataset

Figure A.3 contains additional examples of refined detections on selected sequences from the DAVIS dataset. In the first image, our refinement model recovers details such as the bear's ears and fur. As for the frames containing people, it is noteworthy how finer details such as fingers, feet shape and hair are properly segmented after refinement. Moreover, the mallard example illustrates how the model is robust to scenarios where object and background present very similar appearance in terms of color.

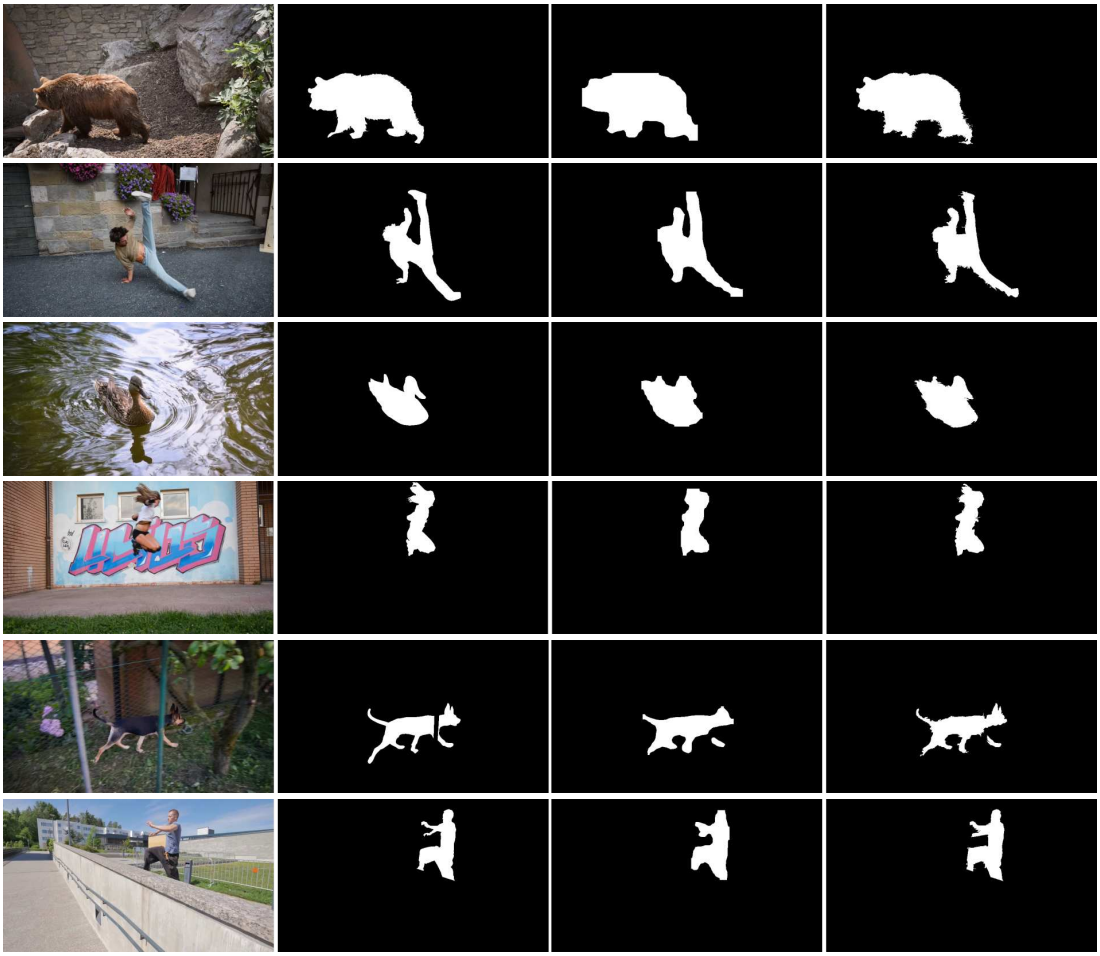


Figure A.3: Examples of detections on the DAVIS dataset. From left to right: original image, ground truth, FCIS detection, FCIS+RGR.

## A.3 Analysis of sensitivity to parameters

As described in Section 3 (*Proposed Approach*) of our main document, our proposed method contains four pre-defined parameters:

- $n_s$: number of Monte Carlo iterations performed for region growing;
- $\theta_m$: normalizing factor for color distance in Eq. (2). With its counterpart $\theta_s$ fixed, the parameter $\theta_m$ regulates the weight between color and spatial distances when measuring the similarity between pixels;
- $\gamma$: average spacing between initial seeds. Together with the cardinality of the high-confidence region $R_H$, this parameter defines the number of seeds to be sampled in each Monte Carlo iteration;
- $d_{Max}$: the maximum distance allowed between a pixel and a candidate cluster. If the distance between a pixel and the corresponding centroid is larger than $d_{Max}$, the node associating these two elements is not pushed into the queue for region growing.

We provide in the main paper an analysis of performance sensitivity of RGR according to $n_s$. In addition, we provide in the present section an analysis of sensitivity according to $\theta_m, \gamma$ and $d_{Max}$. To that end, different configurations on the PASCAL VOC 2012 validation set are evaluated, in the same scenario described in Section 4.2 of the main document. As summarized in Figure A.4, selecting $\theta_m$ values lower than 1 ensures that no significant changes in performance are observed.

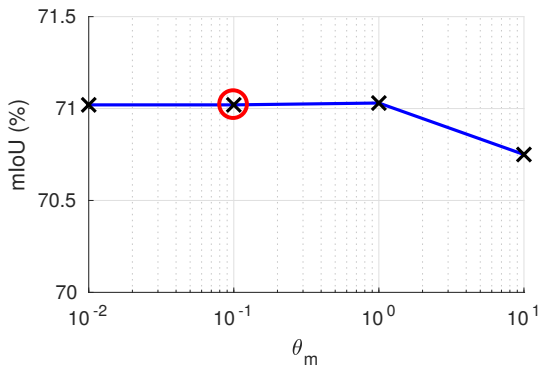



Figure A.4: Mean intersection over union on the PASCAL VOC 2012 dataset according to the distance normalizing factor $\theta_m$. The value selected for the final model is highlighted in red.

As for the number of samples per iteration, the analysis summarized in Figure A.5 confirms that no significant variations in performance occur if a sufficient number of initial seeds is sampled.

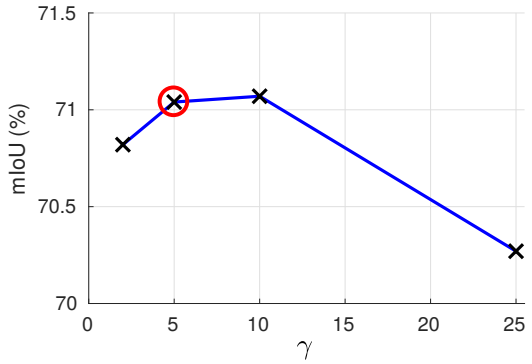



Figure A.5: Mean intersection over unionn on the PASCAL VOC 2012 dataset according to the average spatial distance between samples $\gamma$. The value selected for the final model is highlighted in red.

Finally, results provided in Figure A.6 illustrate that the parameter $d_{Max}$ has very low impact in the final performance, such that a value of $d_{Max} = 500$ is sufficient to ensure optimal performance.

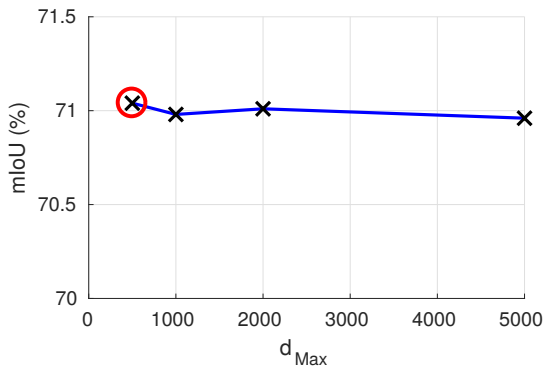



Figure A.6: Mean intersection over union on the PASCAL VOC 2012 dataset according to the maximum distance allowed between pixels composing a cluster $d_{Max}$. The value selected for the final model is highlighted in red.

## A.4 Repeatability despite randomness

As summarized in Fig. 7 (right) of our main manuscript, running RGR with 5+ MC iterations provides only minor improvements (about 0.1%) over refinements obtained with 3 MC iterations. Such results were obtained from independent RGR runs, indicating the repeatability of the method. This repeatability is confirmed by the statistics observed in 10 RGR runs with the same configuration described in Section 4.1 (3 MC) for refinement of FCIS predictions on the PASCAL dataset. As shown below in Fig. A.7 (left), the $mIoU$ was 71.03%, with a standard deviation of 0.02%. Fig. A.7 (right) summarizes this variation across the different PASCAL categories. As the figure shows, the maximum deviation over 10 independent runs is 0.14% (for the *horse* category).

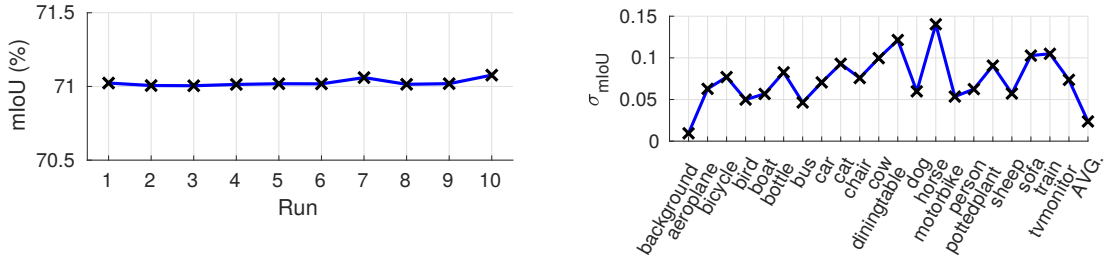



Figure A.7: Repeatability of RGR runs in terms of $mIoU$ variation. *Left*: average over all categories for each run. *Right:* standard deviation of $mIoU$ per category.

# APPENDIX B
# GAZE ESTIMATION - BASELINE GEOM

For the following formulation, let $N_x$ represent the X-coordinate of the nose, $E_x^L$ represent the X-coordinate of the left eye, and $A_y^R$ represent the Y-coordinate of the right ear. Additional variables are analogous, with superscripts $\{R, L\}$ indicating right/left side, and subscripts $\{x, y\}$ corresponding to the coordinates x and y, respectively.

## B.1 Estimating facial normal

Inspired by the method described in the reference [13] of the main document, the first step of *Geom* consists of estimating the facial normal based on the available keypoints. Yet, differently from [13] that also requires mouth keypoints, in our case only eyes and nose coordinates are used.

Let $E$ represent the eye-centroid computed as average of the right and left eyes $E_x^R$ and $E_x^L$, respectively. The facial symmetry axis $\vec{s}$ is approximated as a vector that is perpendicular to the eye-axis $\overrightarrow{E^R E^L}$, i.e. $\vec{s} \cdot \overrightarrow{E^R E^L} = 0$.

Then, the facial normal $\vec{n}$ is estimated as a vector that is normal to $\vec{s}$ and contains the nose $N$. To identify the coordinates of the point $P$ where $\vec{s}$ and $\vec{n}$ intersect, a set of two dot product equations is used. As schematized in the following drawing, $P$ is associated to $E$ as summarized by Eq. B.1:

$$
\begin{aligned}
&dx = \frac{E_x}{2}; \quad dy = \frac{E_y}{2}; \\
&tan\theta = \frac{dy}{dx} = \frac{\beta}{\alpha} \rightarrow \beta = \alpha \frac{dy}{dx} \\
&P = (P_x, P_y) = (E_x, E_y) + (\beta, \alpha) \\
&P = (E_x, E_y) + \left(\alpha \frac{dy}{dx}, \alpha\right)
\end{aligned} \tag{B.1}
$$

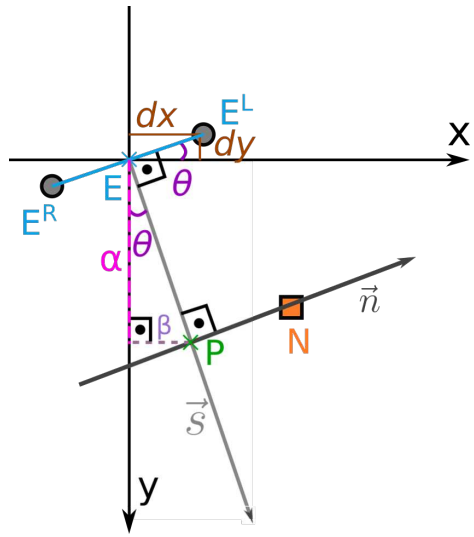



Figure B.1: Computing facial facial symmetry axis $\vec{s}$ and facial normal $\vec{n}$.

Hence, to find $P$ we can write the directions of $\vec{s}$ and $\vec{n}$ as $\overrightarrow{EP}$ and $\overrightarrow{PN}$, such that:

$$
\begin{aligned}
\vec{s} &= \overrightarrow{EP} = (P_x, P_y) - (E_x, E_y) \\
\overrightarrow{EP} &= (E_x, E_y) + \left(\alpha \frac{dy}{dx}, \alpha\right) - (E_x, E_y) \\
\overrightarrow{EP} &= \left(\alpha \frac{dy}{dx}, \alpha\right) \\
\vec{n} &= \overrightarrow{PN} = (N_x, N_y) - (P_x, P_y) \\
\overrightarrow{PN} &= (N_x, N_y) - (E_x, E_y) - \left(\alpha \frac{dy}{dx}, \alpha\right) \\
\overrightarrow{PN} &= \left(N_x - E_x - \alpha \frac{dy}{dx}, N_y - E_y - \alpha\right)
\end{aligned}
\tag{B.2}
$$

Since $\vec{n}$ is normal to $\vec{s}$, it follows that:

$$\vec{s} \cdot \vec{n} = 0 \rightarrow \overrightarrow{EP} \cdot \overrightarrow{PN} = 0$$

$$\overrightarrow{EP} \cdot \overrightarrow{PN} = \begin{bmatrix} \alpha \frac{dy}{dx} & \alpha \end{bmatrix} \cdot \begin{bmatrix} N_x - E_x - \alpha \frac{dy}{dx} \\ N_y - E_y - \alpha \end{bmatrix} = 0$$

$$= N_x \alpha \frac{dy}{dx} - E_x \alpha \frac{dy}{dx} - \alpha^2 \left( \frac{dy}{dx} \right)^2$$

$$+ N_y \alpha - E_y \alpha - \alpha^2 = 0$$

$$= \alpha^2 \left[ \left( \frac{dy}{dx} \right)^2 + 1 \right]$$

$$+ \alpha \left[ E_x \frac{dy}{dx} - N_x \frac{dy}{dx} - N_y + E_y \right] = 0$$

$$\alpha \left[ \left( \frac{dy}{dx} \right)^2 + 1 \right] = - \left[ E_x \frac{dy}{dx} - N_x \frac{dy}{dx} - N_y + E_y \right]$$

$$\alpha = \frac{\frac{dy}{dx}(N_x - E_x) + N_y - E_y}{\left(\frac{dy}{dx}\right)^2 + 1}$$

Once $\alpha$ is obtained, $P$ can be estimated using Eq.B.1 and thus the direction of the facial normal $\vec{n}$ is approximated as $\overrightarrow{PN}$.

## B.2 Estimating head pitch

The angle between ears and eyes allows an estimation of head pitch rotation (looking up or down). From the previous prediction using roll (eyes positions), the second step is to shift this prediction up/down according to this pitch estimation.

Let $A$ represent the ear-centroid computed as average of the right and left ears $A^R$ and $A^L$, respectively, while $\omega$ represents the angle between $A$ and the eye-centroid $E$ that is used as pitch estimation. To apply the rotation $R(\omega)$ matrix defined in Eq. B.3, we can obtain $sin\omega$ and $cos\omega$ as

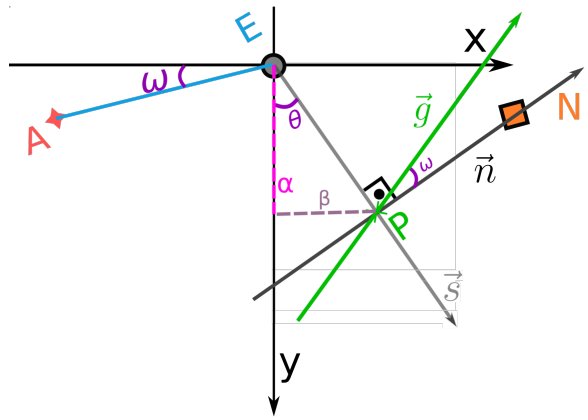



Figure B.2: Illustration of how the pitch angle $\omega$ is computed according to eyes and ears coordinates, and then applied to $\vec{n}$ to estimate $\vec{g}$.

$$R(\omega) = \begin{bmatrix} cos\omega & -sin\omega \\ sin\omega & cos\omega \end{bmatrix} \tag{B.3}$$

$$cos\omega = \frac{E_x - A_x}{\left\|\overrightarrow{AE}\right\|}; \quad sin\omega = \frac{E_y - A_y}{\left\|\overrightarrow{AE}\right\|}, \tag{B.4}$$

such that the rotation matrix becomes

$$\begin{aligned} R(\omega) &= \begin{bmatrix} cos\omega & -sin\omega \\ sin\omega & cos\omega \end{bmatrix} \\ &= \frac{1}{\left\|\overrightarrow{AE}\right\|} \begin{bmatrix} E_x - A_x & -E_y + A_y \\ E_y - A_y & E_x - A_x \end{bmatrix}. \end{aligned}$$

Thus, gaze direction $\vec{g}$ is estimated as

$$\tilde{\vec{g}} = R(\omega)\vec{n} = \frac{\overrightarrow{PN}}{\left\|\overrightarrow{AE}\right\|} \begin{bmatrix} E_x - A_x & -E_y - A_y \\ E_y - A_y & E_x - A_x \end{bmatrix}. \tag{B.5}$$

## B.3 Cases of missing keypoints

The presented *Geom* method requires the detection of at least the nose and one eye for gaze estimation. Cases of missing keypoint detections are handled as:

- *one ear missing*- such cases mostly correspond to images where a lateral view of the subject's face is available. Predictions are done using the available complete pair eye-ear (either left or right) as replacement for the corresponding eye-centroid and ear-centroid;
- *one eye missing*- instead of using the eye-axis, the ear-axis is used to estimate the facial normal. Then, pitch is estimated using the available complete pair eye-ear (either left or right) as replacement for the corresponding eye-centroid and ear-centroid;
- *both ears not detected*- pitch is not estimated, so gaze is estimated using just the facial normal, i.e. $\tilde{\vec{g}} = \vec{n}$;

# APPENDIX C
# **COPYRIGHT**