Designing a Labeling Application for Image Object Detection

Marco Tebaldi

Department of Information Engineering

Università degli Studi di Padova

A thesis submitted for the degree of

Master of Science in Computer Science and Engineering
Laurea Specialistica in Ingegneria Informatica

October 2010

Abstract

We seek to build a large collection of images with ground truth labels to be used for training object detection and recognition algorithms. Such data is useful for supervised learning and quantitative evaluation. To achieve this, we developed a user interface tool that allows easy image annotation. The tool provides functionalities such as drawing boxes, querying images, and browsing the database. Using this annotation tool, we can collect a large dataset that spans many object categories, often containing multiple instances over a wide variety of images. We quantify the contents of an existing dataset and compare against other state of the art datasets used for object recognition and detection. Also, we show how to extend our dataset to automatically enhance object labels with WordNet, discover object parts, and increase the number of labels using minimal user supervision.

To my grandparents,

the coolest ones I could have had.

Acknowledgements

I would like to gratefully acknowledge the enthusiastic supervision of my advisor Serge Belongie; this work would not have been possible without his support, guidance, and the resources he provided. I would like to thank Fan Chung Graham, Hovav Shacham, and Lawrence Saul from the University of California San Diego, not only for their advice on my thesis and future plans, but, also for their encouragement. Special thanks to Emanuele Menegatti for participating in the thesis project and helping me especially in the final process. From my Padova, Aberdeen and San Diego days, I am indebted to all my professors for providing some of my most enjoyable courses, as well as piquing my interest in graduate school and academia. I also thank Nadav and Ori Ben-Haim for the resources they provided me at Orpix-inc and the UCSD Lab.

I cannot end without thanking my friends and family – my friends for moving me away from school when it was getting late and the library was already closed, giving me the chance to recharge with all their fun; and my family for being always present when I was abroad. To my parents, I thank you for not telling me what I could and could not do, or, what I should and should not do. Your support has been a pillar of my life.

Contents

List of Figures

LIST OF FIGURES

List of Tables

LIST OF TABLES

1

Overview of the project

Suppose a company that manages a mass surveillance system has to implement a technology to recognize vehicles ownership to realize an automatic system as a software solution of electronic toll collection on pay-per-use roads and cataloging the movements of traffic or individuals. They can use existing closed-circuit television or road-rule enforcement cameras, or ones specifically designed for the task, but they would need a software to detect such vehicles and a set of information from those images. This technology is called Automatic Number Plate Recognition (ANPR), which uses optical character recognition on images to read the license plates on vehicles. ANPR can be used to store the images captured by the cameras as well as the text from the license plate, with some configurable to store a photograph of the driver.

Orpix is a computer vision company which furnishes such technology to those kind of informative systems to detect images. In order to detect objects in images through the use of complex algorithms, those systems need a software which is combining the use of image processing techniques and trained databases of examples of such objects.

For instance, indeed, that ANPR system uses a detector specialized in cars and vehicles (see Figure 1.1). Therefore it needs a software which is recognizing cars and number plates from the images captured by its camera system. Such software will work on the base of a dataset containing "examples", on which those objects are cars and number plates. The detecting technology will then use its database's records to

query and match the results. It is therefore very important to have a good and reliable *database of labeled images*, which plays a central role, to improve the performances of such software systems.

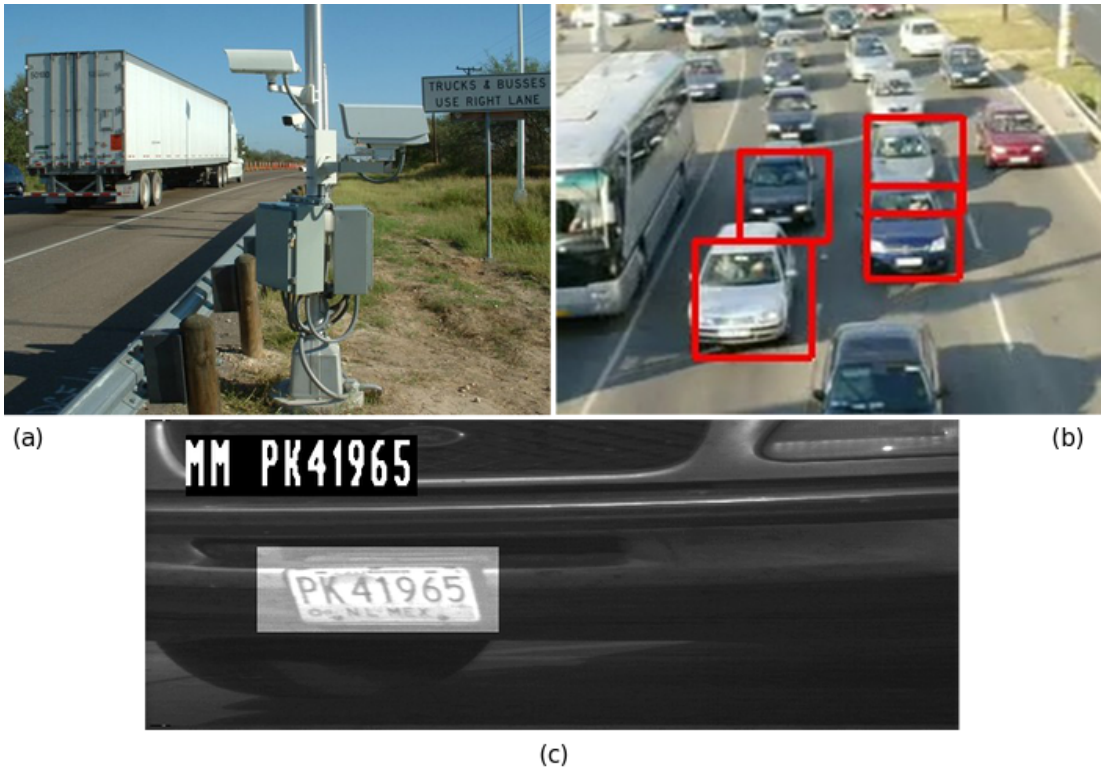

Figure 1.1: Automatic Number Plate Recognition - Used by various police forces and as a method of electronic toll collection on pay-per-use roads and cataloging the movements of traffic or individuals. (a) Closed-circuit cameras. (b) Vehicles are detected on a real-time scene. (c) The number plate is recognized by the automatic system. Source: Perceptics Corporation.

1.1 The Orpix Labeling project

Orpix has its own technology and algorithms to deal with detectors of objects from images and video frames.

They are developing object and pattern recognition software that supports a variety of recognition techniques in a common and consistent image analysis framework. The framework consists of Detectors, each of which is targeted at recognizing a specific family of Objects and Patterns. They provide user-friendly tools to define Detectors based on a simple training procedure. The various detectors can be used in domain specific solutions (e.g., Bio Imaging, Surveillance, Media, Manufacturing, etc.) using their Image Analysis Framework. The Image Analysis Framework can process many detectors on a single image, the results are then further analyzed together for a specific domain using contextual information.

I've been contacted to create a dataset of objects covering a wide range of environments and several object categories, to interface it with their existing informative systems and state of the art image and video analysis framework that supports various application domains. To achieve this scope I took part in the practical generation of such amount of labeled data by designing an application on which, given images in input, an user can label and store examples of objects for a desired content. The easiest way of doing that was by allowing users to draw a box around each object in an image, and giving a label to the annotation to describe it. We called this project "Orpix Labeling Application".

1.1.1 Goals of the project

The goal of the project is to create a Labeling Application that generates human labeled data which is then used by the Orpix Object Detection Engine (ODE) to train various detectors. A detector defines the classifiers and additional parameters necessary to detect the desired content in an image. In this project we will provide User-Interface tools to train one of the following detector's types:

- Object detection – train a binary classifier that represent objects of interest based on examples of positive regions in a given training set.
- Texture detection – train a binary classifier of repeatable texture in an image.

- Edge detection – train a binary classifier of small patches around edges with similar characteristics.

The tool will also provide functionalities such as drawing boxes, querying images, and browsing the database of the training set.

We will then focus on the realization of a dataset which is the result of the labeling process and training of a database of images through the Orpix labeling application. The application design will therefore take into account considerations about the dataset creation and additive information which can improve its performance.

1.2 Design considerations

There are many aspects to consider in the design of a piece of software. The importance of each should reflect the goals the software is trying to achieve. One first aspect is the compatibility. The software that we are going to develop has to be able to operate with other products that are designed for interoperability with another product. For example, since the Orpix Software technologies are mainly developed in a MS Windows ambient, we want to assure full compatibility with their existing systems building a database which will then used by their internal software engine. Also, our software may be backward-compatible with an older version of itself, and since they develop code using their experience in C#, which is an object-oriented visual design description language, we decided to adapt our solution to their ambient. This decision is made also to limit future expenses with the maintenance of our code.

Another reason of choosing that platform is that new capabilities can be added to the software without major changes to the underlying architecture. We will also propose possible extensions which can be made in the near future without changing the entire structure of the system, but just adding a simple plug-in.

One important aspect is also the modularity of the system. The resulting software comprises well defined, independent components. That leads again to better maintainability. The components will be then implemented and tested in isolation before being integrated to form the desired software system. This allows division of work in our software development project.

One last aspect we are taking in consideration is that the software user interface must be usable for its target user/audience. We will make some considerations for

example on what to label and how each user will see the same objects with different eyes. Furthermore, default values for the parameters must be chosen so that they are a good choice for the majority of the use cases.

1.3 Design patterns

It is known in software design that a software architect may identify a design problem which has been solved by others before, i.e., a template or pattern describing a solution to a common problem. The reuse of such patterns can speed up the software development process, having been tested and proved in the past. We will then compare the results and performance of existing datasets for object detection, and make statistics to determine which one is a better model, or various aspects to follow and perhaps to improve.

1.4 Document Outline

This document follows a precise scheme. To make sure that the reader is familiar with the concepts concerning Image Object Detection, we propose in Chapter 2 a short introduction to a handful of general notions on the topics related to the meaning of having a quite large set of labeled examples at disposal to train a detector. The scope of this introduction is not to explain how object detection's algorithms work, but it is useful to know that such technologies are at close contact to our aims.

In Chapter 3 we go over the requirements analysis and we make important design considerations. We want to build a large collection of examples spanning many object categories, often containing multiple instances over a wide variety of images. This to be used for training object detection and recognition algorithms. Such data is useful for supervised learning and quantitative evaluation. We will take into account in this section a quantitative analysis of an existing dataset which made possible the comparison of learning algorithms and experiments on new dataset enhancements.

The implementation of the labeling application is showed in Chapter 4, where we discuss the design considerations which brought us to the realization of the software. Some techniques are discussed here in order to achieve better performances of the dataset. We discuss the functional description of the Application Layout as well as our

dataset extension. This section shows the method used to enhance the object labels through the analysis of synonyms for the same object category, and how we can take advantage of an heuristic to discover semantically meaningful object-part relationships, to provide a list of part labels indicating how they occur with a given object label.

Chapter 5 is dedicated to the user workflow. In this section we summarize the most common operations during the labeling process as a step-by-step workflow. And eventually we give some useful tips and labeling guidelines.

Finally, we make our conclusions in Chapter 6.

2

A brief introduction to Image Object Detection

The scope of our project is to create a dataset of labeled objects, through a process of supervised training data. The application that we want to design and develop will use such task to then train a detector for a certain class of objects or category. We will then make use of scientific disciplines and concepts as Machine Learning and Classification as well as Pattern Recognition, which are concerned with computer technologies related to computer vision.

The purpose of this short introduction is to provide the reader with background sufficient to follow the discussions concerning Image Object Detection.

2.1 Digital Image Processing

Digital image processing is the use of computer algorithms to perform image processing on digital images. Image processing and image analysis tend to focus on 2D images, how to transform one image to another, e.g., by pixel-wise operations such as contrast enhancement, local operations such as edge extraction or noise removal, or geometrical transformations such as rotating the image. This characterisation implies that image processing/analysis neither require assumptions nor produce interpretations about the

image content.

Digital image processing has a wide range of applications in intelligent transportation systems, such as Automatic Number Plate Recognition and Traffic Sign Recognition, as well as in digital camera images.

Digital cameras, indeed, generally include dedicated digital image processing chips to convert the raw data from the image sensor into a color-corrected image in a standard image file format. Images from digital cameras often receive further processing to improve their quality, a distinct advantage that digital cameras have over film cameras. The digital image processing typically is executed by special software programs that can manipulate the images in many ways. Many digital cameras also enable viewing of histograms of images, as an aid for the photographer to understand the rendered brightness range of each shot more readily.

An image histogram is a type of histogram which acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. By looking at the histogram for a specific image a viewer will be able to judge the entire tonal distribution at a glance, see Figure 2.1.

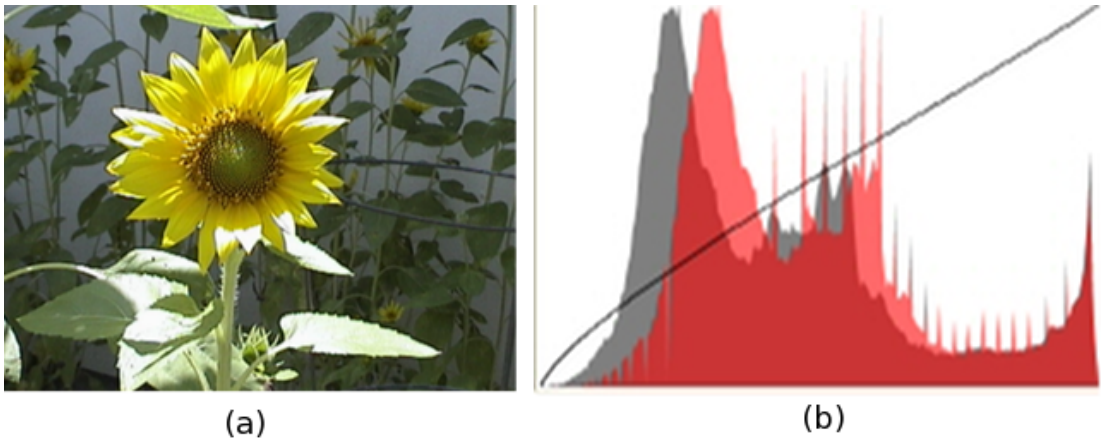

Figure 2.1: Histogram analysis of an image's tonal distribution - (a) A sunflower image. (b) Histogram of the sunflower image.

Digital image processing allows the use of much more complex algorithms for image processing, and hence, can offer both more sophisticated performance at simple tasks, and the implementation of methods which would be impossible by analog means. In

particular, digital image processing is the only practical technology for Classification, Feature extraction and Pattern/Object recognition, just to cite some.

2.2 Object Detection

Object Detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. Given an image, object detection is to determine whether or not the specified object is present, and, if present, determine the locations and sizes of each objects.

The research for object detection and recognition is focusing on:

1. *Representation*: how to represent an object;
2. *Learning*: Machine Learning algorithms to learn the common property of a class of objects;
3. *Recognition*: identify the object in an image using models learned from 2).

Different varieties of the recognition problem are described in the literature:

- *Object recognition*: one or several pre-specified or learned objects or object classes can be recognized, usually together with their 2D positions in the image or 3D poses in the scene.
- *Identification*: An individual instance of an object is recognized. For example the identification of a specific person's face or fingerprint, or identification of a specific vehicle.
- *Detection*: the image data is scanned for a specific condition. For example the detection of possible abnormal cells or tissues in medical images or detection of a vehicle in an automatic road toll system.

Detection based on relatively simple and fast computations is sometimes used for finding smaller regions of interesting image data which can be further analysed by more computationally demanding techniques to produce a correct interpretation.

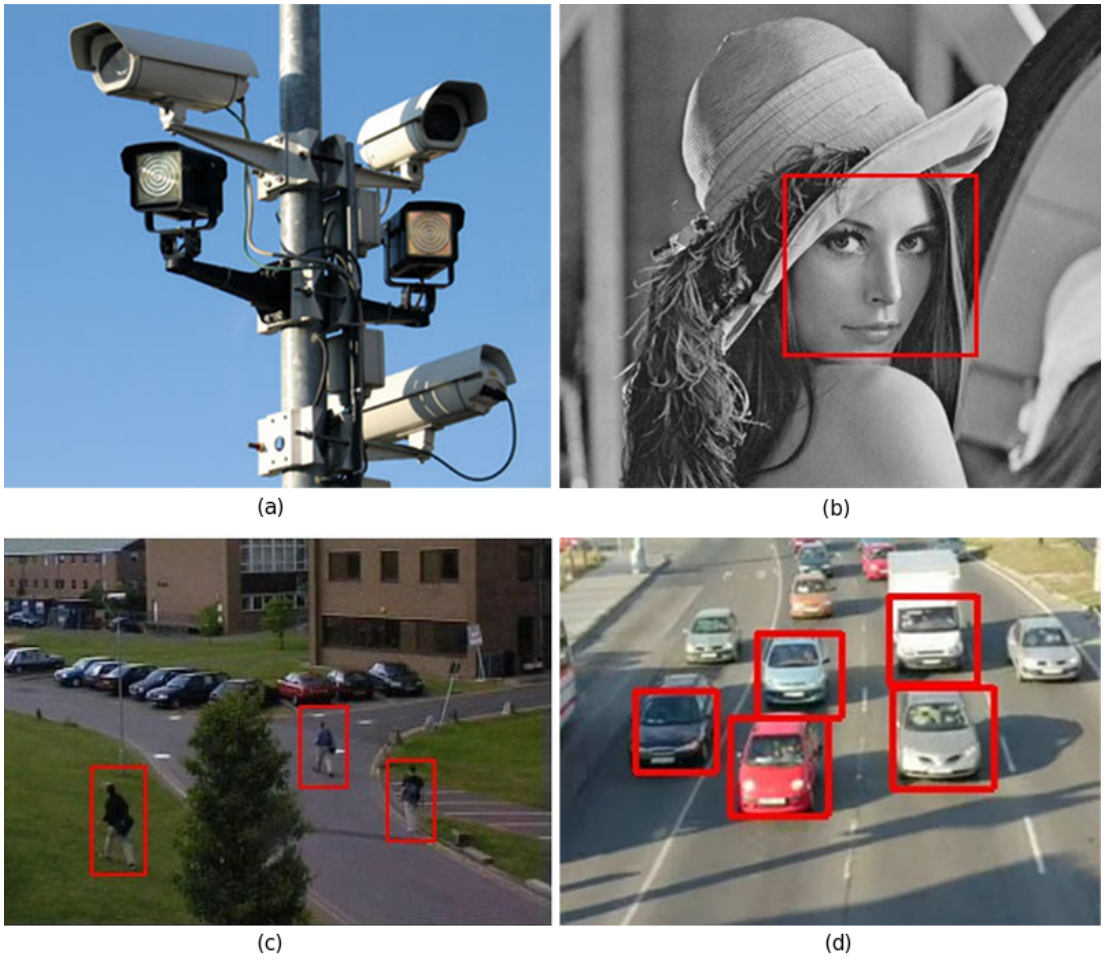

Figure 2.2: The use of object detection technologies in video surveillance. - They can use existing closed-circuit television or road-rule enforcement cameras, or ones specifically designed for the task. (a) A set of cameras adopted by systems which commonly use also infrared lighting to allow the camera to take the picture at any time of the day. (b) An example of a face detector. (c) How they are used by various police forces cataloging the movements of traffic or individuals, and (d) as a method of electronic toll collection on pay-per-use roads.

Well-researched domains of object detection include smaller regions of interest such as face detection, pedestrian detection, car detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

This technology uses optical character recognition on images to read the license plates on vehicles, to detect human movements, or to recognize faces. They can use existing closed-circuit television or road-rule enforcement cameras, or ones specifically designed for the task. Figure 2.2(a) shows a set of cameras adopted by systems which commonly use also infrared lighting to allow the camera to take the picture at any time of the day. Figure 2.2(b) shows an example of a face detector. This technology tends to be region-specific. In Figure 2.2 is shown another example of how they are used by various police forces cataloging the movements of traffic or individuals (c), and as a method of electronic toll collection on pay-per-use roads (d).

2.3 Machine Learning

Object detection systems have recently achieved high detection rates and real-time performance. However, these methods usually rely on a huge training database (around 5,000 positive examples for good performance [15]). While such huge databases may be feasible for building a system that detects a single object, it is obviously problematic for scenarios where multiple objects (or multiple views of a single object) need to be detected. Indeed, even for multi-view face detection the performance of existing systems is far from satisfactory. Performance depends crucially on the features that are used to represent the objects.

Learning systems usually consist of two elements, the learning algorithm and the features. The type of feature has great impact on the results. For instance, using local edge orientation histograms (EOH) as features in the learning algorithm greatly improves the learning of objects from a small database and enables improving the state-of-the art real-time systems for learning objects from different points of view [15].

These histogram features are not limited to simple object as faces or vehicles and can significantly improve results on different types of objects such as buildings, pedestrians, trees, etc.

Machine Learning, however, is a scientific discipline that is concerned with the design and development of algorithms that allow computers to evolve behaviors based

on empirical data, such as from sensor data or databases. A learner can take advantage of examples (data) to capture characteristics of interest of their unknown underlying probability distribution. Data can be seen as examples that illustrate relations between observed variables. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data). Hence the learner must generalize from the given examples, so as to be able to produce a useful output in new cases. We will discuss the training dataset in Section 2.4 of this chapter.

Artificial intelligence is a closely related field to machine learning, as are probability theory and statistics, data mining, pattern recognition, adaptive control, computational neuroscience and theoretical computer science.

2.3.1 Pattern Recognition

In machine learning, pattern recognition is the assignment of some sort of output value (or label) to a given input value (or instance), according to some specific algorithm. An example of pattern recognition is classification, which attempts to assign each input value to one of a given set of classes. However, pattern recognition is a more general problem that encompasses other types of output as well. Other examples are regression, which assigns a real-valued output to each input; sequence labeling, which assigns a class to each member of a sequence of values (for example, part of speech tagging, which assigns a part of speech to each word in an input sentence); and parsing, which assigns a parse tree to an input sentence, describing the syntactic structure of the sentence. Pattern recognition algorithms generally aim to provide a reasonable answer for all possible inputs, and to do "fuzzy" matching of inputs.

Pattern recognition is generally categorized according to the type of learning procedure used to generate the output value [3]. Supervised learning assumes that a set of training data (the training set) has been provided, consisting of a set of instances that have been properly labeled by hand with the correct output. A learning procedure then generates a model that attempts to meet two sometimes conflicting objectives: perform as well as possible on the training data, and generalize as well as possible to new data (usually, this means being as simple as possible).

2.3.2 Classification

In machine learning and pattern recognition, classification refers to an algorithmic procedure for assigning a given piece of input data into one of a given number of categories. An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.). An algorithm that implements classification, especially in a concrete implementation, is known as a *classifier*. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.

Classification normally refers to a supervised procedure, i.e. a procedure that learns to classify new instances based on learning from a training set of instances that have been properly labeled by hand with the correct classes (supervised learning, discussed later in Section 2.4.1).

2.4 Training a detector: the Dataset

A training set is a set of data used in various areas of information science to discover potentially predictive relationships. Training sets are used in artificial intelligence, machine learning, genetic programming, intelligent systems, and statistics. In all these fields, a training set has much the same role and is often used in conjunction with a test set.

In artificial intelligence or machine learning, a training set consists of an input vector and an answer vector, and is used together with a supervised learning method to train a knowledge database (e.g. a neural net, a naive bayes classifier, or, in our case, an object detector) used by an AI machine.

Thousands of objects occupy the visual world in which we live. Biederman [1] estimates that humans can recognize about 30000 entry-level object categories. Recent work in computer vision has shown impressive results for the detection and recognition of a few different object categories [12, 14, 25]. However, the size and contents of existing datasets, among other factors, limit current methods from scaling to thousands of object categories. Research in object detection and recognition would benefit from large image and video collections with ground truth labels spanning many different object categories in cluttered scenes. For each object present in an image, the labels

should provide information about the object's identity, shape, location, and possibly other attributes such as pose.

By analogy with the speech and language communities, history has shown that performance increases dramatically when more labeled training data is made available. One can argue that this is a limitation of current learning techniques, resulting in the recent interest in Bayesian approaches to learning [7, 19] and multi-task learning [21]. Nevertheless, even if we can learn each class from just a small number of examples, there are still many classes to learn.

Large image datasets with ground truth labels are useful for supervised learning of object categories. Many algorithms have been developed for image datasets where all training examples have the object of interest well-aligned with the other examples [12, 22, 25]. Algorithms that exploit context for object recognition [13, 20] would benefit from datasets with many labeled object classes embedded in complex scenes. Such datasets should contain a wide variety of environments with annotated objects that co-occur in the same images.

2.4.1 Supervised Learning

Supervised learning is the machine learning task of inferring a function from supervised training data. As seen before, the training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier (if the output is discrete) or a regression function (if the output is continuous) [23].

The inferred function should predict the correct output value for any valid input object. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Other factors to consider when choosing and applying a learning algorithm include the following:

- Heterogeneity of the data.

  If the feature vectors include features of many different kinds (discrete, discrete ordered, counts, continuous values), some algorithms are easier to apply than others. Many algorithms, including Support Vector Machines, linear regression, logistic regression, neural networks, and nearest neighbor methods, require that the input features be numerical and scaled to similar ranges (e.g., to the $[-1, 1]$ interval). Methods that employ a distance function, such as nearest neighbor methods and support vector machines with Gaussian kernels, are particularly sensitive to this. An advantage of decision trees is that they easily handle heterogeneous data.

- Redundancy in the data.

  If the input features contain redundant information (e.g., highly correlated features), some learning algorithms (e.g., linear regression, logistic regression, and distance based methods) will perform poorly because of numerical instabilities. These problems can often be solved by imposing some form of regularization.

- Presence of interactions and non-linearities.

  If each of the features makes an independent contribution to the output, then algorithms based on linear functions (e.g., linear regression, logistic regression, Support Vector Machines, naive Bayes) and distance functions (e.g., nearest neighbor methods, support vector machines with Gaussian kernels) generally perform well. However, if there are complex interactions among features, then algorithms such as decision trees and neural networks work better, because they are specifically designed to discover these interactions. Linear methods can also be applied, but the engineer must manually specify the interactions when using them.

In general, when considering a new application, the engineer can compare multiple learning algorithms and experimentally determine which one works best on the problem at hand. Tuning the performance of a learning algorithm can be very time-consuming. Given fixed resources, it is often better to spend more time collecting additional training

data and more informative features than it is to spend extra time tuning the learning algorithms.

For the purposes of our project, we will discuss in the next chapter a quantitative analysis of existing state-of-the art dataset, which will lead to some consideration taken during the design process of the Labeling Application to develop.

3

Requirements Analysis and Design Considerations

This chapter provides an overview of the design of the Labeling software system. During conceptual design it provides a "broad-brush" perspective of the design with detail added during subsequent design phases.

The focus during conceptual design is on describing enough of the design to allow an examination of the design's suitability in meeting the system requirements. In this fashion, the document presents many conceptual design concepts as design requirements.

While system interfaces are identified, they are not detailed until the preliminary design process; examples used here should be taken as illustrative only at this point in the design process. Where appropriate, baseline choices are indicated. These should be viewed more as discussion aids at this point.

The presentation of the design itself follows the structure presented in early sections: the design of the system as it relates to the functional architecture is presented first, followed by those design considerations drawn from the technical architecture. Because the functional architecture closely follows similar architectures found in other modern dataset tools, the initial concentration is on the technical design, which is more interesting during conceptual review.

3.1 Orpix Image Analysis Framework

The dataset we are going to design and then populate is going to interact with base technologies existing at Orpix to train various detectors. The Training module provides functionality for training object/pattern detectors from examples of labeled images coming from the dataset. The application provides labeling tools and an easy training procedure. The detectors can then be used with their Object Detection Engine (ODE) to process an image or a video.

The Image Analysis Framework uses the various detectors to analyze a query image or a video. It addresses the following basic queries:

- Category Matching – Find regions in an image/video frame containing an object or pattern from a defined Category.
- Instance Matching – Query a known image database and quickly retrieve all matches from the database.
- Classification – Given set of image class definitions, classify a query image based on its content.

Orpix Classification is a unique way to classify images based on their content. The image analysis framework uses the Detectors and the Class Definitions to determine a calss of an image. A class Definition is based on the various detectors.

Figure 3.1 shows the concept of category matching: examples of labeled data are coming from the dataset; the training module interacts with the detector which eventually queries an image looking for an object match.

Figure 3.2 shows the method used to search for instance matches: given an object, e.g. a vehicle, the system queries the dataset looking for matches of the specific vehicle, and it retrieves all such matches from the database.

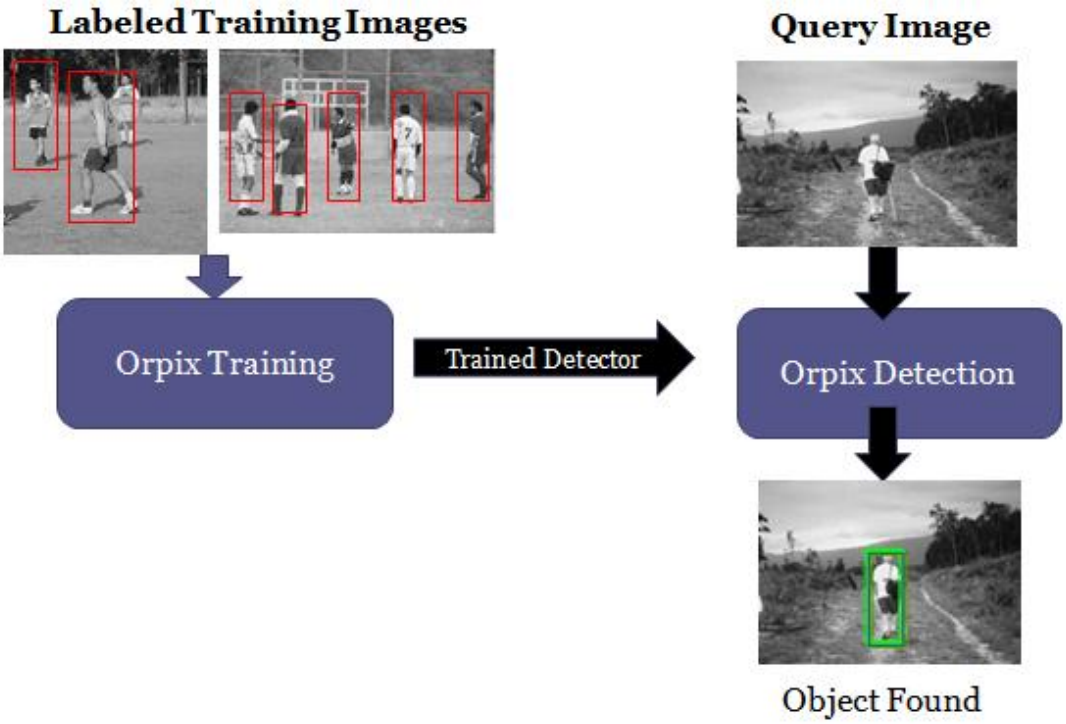

Figure 3.1: Category Matching diagram - Orpix Detection allows to use the detectors that were created using the Training Application to find various objects/pattern in a query image. Source: Orpix-inc.

Figure 3.2: Instance Matching diagram - This technology queries a known image database and quickly retrieves all matches from the database. Source: Orpix-inc.

3.2 Goals and Requirements of the model

The model presented in this document is intended to address the following goals and requirements:

- Flexibility in a laboratory style operating environment
- Common control behavior to simplify operations
- Reduced development costs through reuse and separation of functional and technical architectures
- Reduced integration costs
- Reduced maintenance costs

Since the Labeling Application will run independently on separate terminals, there are not specific constraints. The only requirements are about the running ambient and operative systems, which Orpix will provide locally to their users.

3.3 User Interfaces

The main key task of the project is to provide user interfaces (UI) for human interaction with the labeling software during normal operations. Traditionally, UI development is one whose effort is consistently underestimated, in part because the expectations of users change rapidly as the system design progresses. This results in many rewrites and redesigns of UI. Consequently design of UI support is based on the precept that flexibility is fundamental to UI development. For similar reasons, the Graphical User Interface (GUI) is developed using the Model/View/Controller (MVC) model. See diagram in Figure 3.3.

User interfaces are separated from control operations in the labeling application. Control components provide a common control surface that is used by the Graphical User Interface, scripts, and programmatic access. These control surfaces are available through the communications bus. This allows UI applications to be independent and it increases the modularity of the project. This means that simple UI may be produced quickly to aid in development and safely replaced with more sophisticated UI later.

Because of the flexibility of this approach, little is presented here on UI specifics.

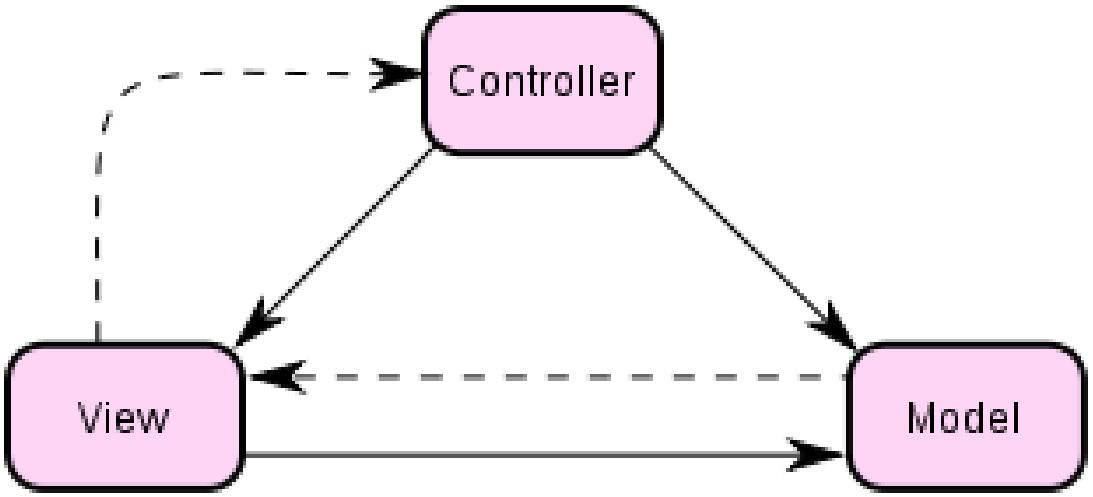

Figure 3.3: A simple diagram depicting the relationship between the Model, View, and Controller. - The solid lines indicate a direct association, and the dashed line indicate an indirect association (e.g., observer pattern).

User interfaces are composable. User interfaces are implemented as classes and can be combined to form UI applications. This allows sophisticated UI to be built quickly from simpler base UI components.

While there are no plans to provide web-based UI, the separation of UI and control makes this a relatively straightforward task if should web-based control be required in the future.

3.4 Existing state-or-the-art datasets

We take now into account in this section a comparison of existing state of the art dataset used for object detection paying attention on their different aspects. We will then make a quantitative analysis of an existing dataset which made possible new dataset enhancements about the labeling itself.

3.4.1 Comparison of datasets used for object detection and recognition

We compare five annotated datasets currently used for object detection and recognition: Caltech-101 [6], LabelMe [16], MSRC [28], CBCL-Streetscenes [2], and PASCAL2006 [5]. Table 3.1 summarizes these datasets. The Caltech-101 and CBCL-streetscenes provide location information for each object via polygonal boundaries. PASCAL2006 provides bounding boxes and MSRC provides segmentation masks.

Dataset	# Categories	# Images	# Annotations	Annotation type
Caltech-101	101	8765	8765	Polygons
LabelMe	183	30396	111490	Polygons
MSRC	23	591	1751	Region masks
CBCL-Streetscenes	9	3547	27666	Polygons
PASCAL2006	10	5304	5455	Bounding boxes

Table 3.1: Summary of datasets used for object detection and recognition research - For the LabelMe dataset, we provide the number of object classes with at least 30 annotated examples. All the other numbers provide the total counts.

Figure 3.4(a), at page 24, shows, for each dataset, the number of object categories and, on average, how many objects appear in an image. Observe that the CBCL-Streetscenes and LabelMe datasets often have multiple annotations per image, indicating that the images correspond to scenes and contain multiple objects. This is in contrast with the other datasets, which prominently feature a small number of objects per image.

Figure 3.4(b) is a scatter plot where each point corresponds to an object category and shows the number of instances of each category and the average size, relative to the image.

Figure 3.4(c) shows the number of labeled instances per object category for the five datasets, sorted in decreasing order by the number of labeled instances.

We also wish to quantify the quality of the polygonal annotations in order to make a technical decision on the annotation mode in our application.

Figure 3.4(d) shows the number of polygonal annotations as a function of the number of control points. The LabelMe dataset has a wide range of control points and the number of annotations with many control points is large, indicating the quality of the dataset. The PASCAL2006 and MSRC datasets are not included in this analysis since their annotations consist of bounding boxes and region masks, respectively.

3.4.2 LabelMe's Dataset: a quantitative analysis

When comparing different algorithms for object detection and recognition, labeled data is necessary to quantitatively measure their performance. Even algorithms requiring no supervision need this quantitative framework.

We summarize the content of the LabelMe database as of December 21, 2006 [17]. The database consists of 111490 polygons, with 44059 polygons annotated using the online tool and 67431 polygons annotated offline. There are 11845 static pictures and 18524 sequence frames with at least one object labeled.

Figure 3.5(a) shows a sorted histogram of the number of instances of each object description for all 111490 polygons . Notice that there are many object descriptions with a large number of instances.

While there is much agreement among the entered descriptions, object categories are nonetheless fragmented due to plurals, synonyms, and description resolution (e.g. "car", "car occluded", and "car side" all refer to the same category). This is a very interesting point that we will discuss in section 4.2.1, where we will address the issue of unifying the terminology to properly index the dataset according to real object categories.

Figure 3.5(b) shows a histogram of the number of annotated images as a function of the percentage of pixels labeled per image. The graph shows that 11571 pictures have less than 10% of the pixels labeled and around 2690 pictures have more than 90% of labeled pixels. There are 4258 images with at least 50% of the pixels labeled.

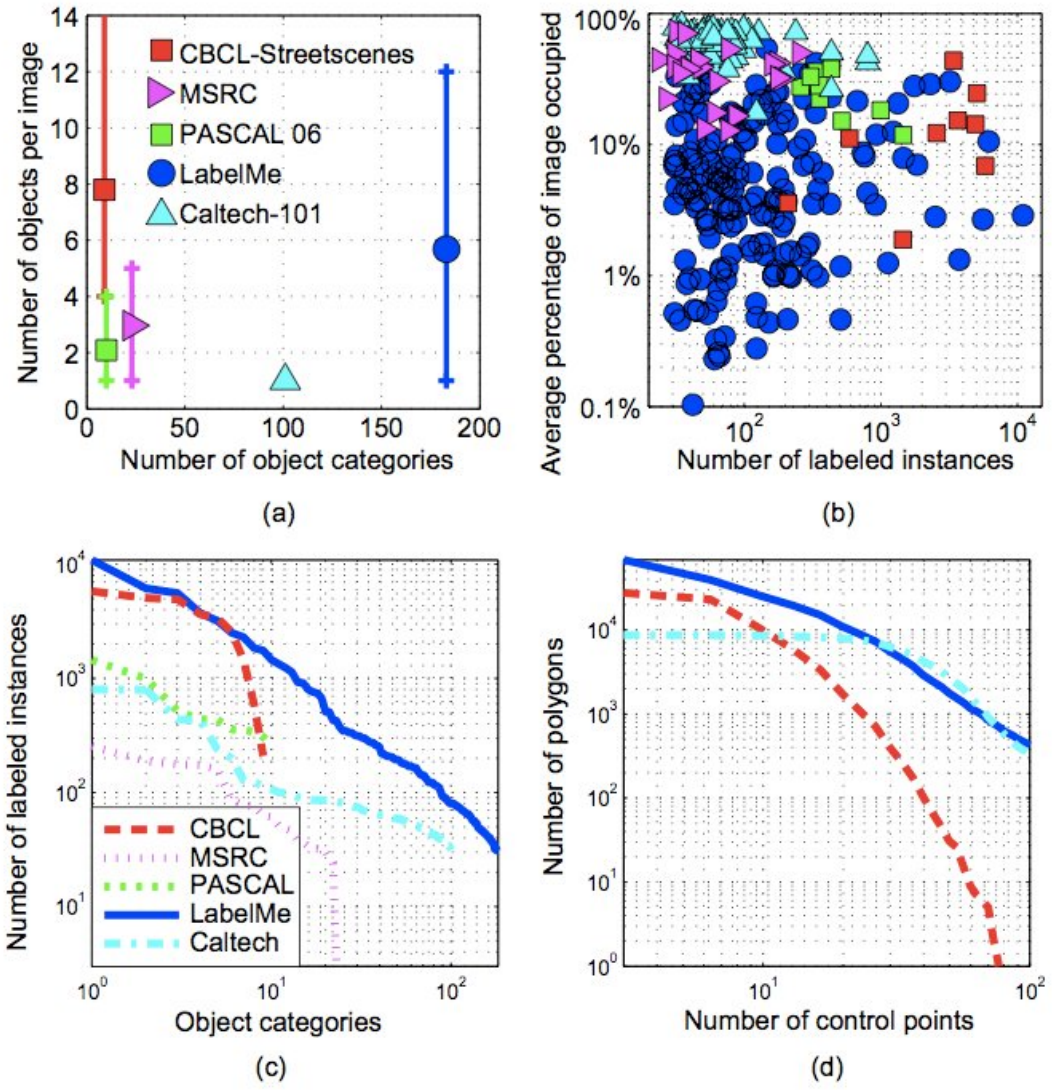

Figure 3.4: Comparison of five datasets used for object detection and recognition: Caltech101, MSRC, CBCL-Streetscenes, PASCAL2006, and LabelMe - (a) Number of object categories versus number of annotated objects per image. (b) Scatter plot of number of object category instances versus average annotation size relative to the image size, with each point corresponding to an object category. (c) Number of labeled instances per object category, sorted in decreasing order based on the number of labeled instances. (d) Depiction of annotation quality, where the number of polygonal annotations are plotted as a function of the number of control points (we do not show the PASCAL2006 and MSRC datasets since their annotations correspond to bounding boxes and region masks, respectively).

Figure 3.5(c) shows a histogram of the number of images as a function of the number of objects in the image. There are, on average, 3.3 annotated objects per image over the entire dataset. There are 6876 images with at least 5 objects annotated.

Figure 3.6 shows images depicting a range of scene categories, with the labeled objects colored to match the extent of the recorded polygon. For many images, a large number of objects are labeled, often spanning the entire image. This results in a very extended work on the annotations.

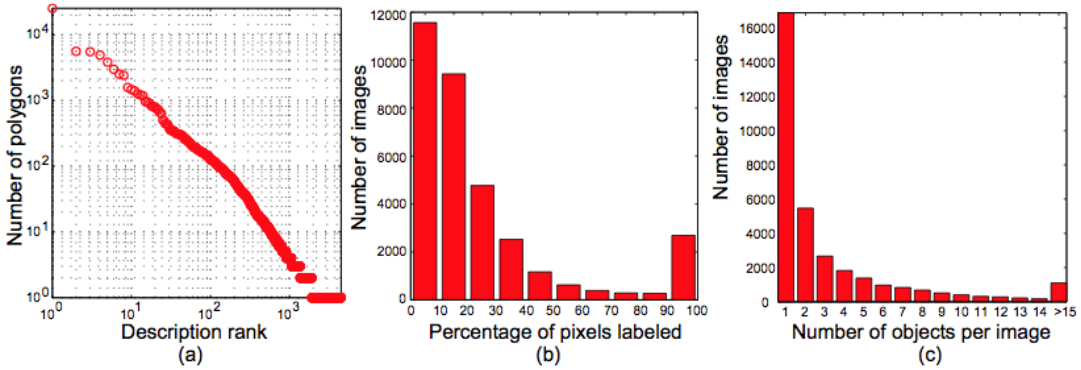

Figure 3.5: Summary of the LabelMe database content - (*a*) Sorted histogram of the number of instances of each object description. Notice that there is a large degree of consensus with respect to the entered descriptions. (*b*) Histogram of the number of annotated images as a function of the area labeled. The first bin shows that 11571 images have less than 10% of the pixels labeled. The last bin shows that there are 2690 pictures with more than 90% of pixel labeled. (*c*) Histogram of the number of labeled objects per image.

3.5 Labeling of the objects

The goal of the annotation tool is to provide a drawing interface that is easy to use, and allows instant sharing of the collected data. However, there are many concerns with this annotation collection scheme. One important concern is quality control. A critical issue is what to label. For example, should one label the entire body, just the head, or just the face of a pedestrian? What if it is a crowd of people? Should all of the people be labeled?

Finally, there is the text label itself. For example, should the object be labeled as a “person”, “pedestrian”, or “face”?

Figure 3.6: Examples of annotated scenes - These images have more than 80% of their pixels labeled and span multiple scene categories. Notice that many different object classes are labeled per image.

We leave these decisions up to each user, which can manage the relative categories. Our solution will provide a drop-down menu of standard object category names and let users use their own descriptions to create new categories. We will then consider some further information about the labeling in Section 4.2.1.

3.6 Resource management

When implementing the application we have to consider that Orpix has a finite set of resources that could be shared when operating in a multiple training mode. Some resources may be physical objects, such as cameras, removal memories for the transport of the data, computers and CPUs. Others are less obvious, such as data stream bandwidth of the terminals. Some resources are sharable, often up to a possibly dynamic limit (remote folders, for example, which may be shared by multiple users) and under some conditions. Resource allocation is determined by availability, priority, and policy, under the responsibility of the laboratory manager where the users are operating.

However, the final application will be used locally by one end-user, or a limited number of separate users at a time, to generate human labeled data in order to train a

certain detector which either belongs to a category or is part of a multi-selection work of parallel data training.

3.6.1 Image Quality

Image quality is quantified by the users and it is up to them the decision whether to label an object or not as discussed in Section 3.5.

3.6.2 Data Store

As of the conceptual design review, there is no requirement that Orpix provide a long-term archive of its data. Nevertheless, some temporary store is needed to collect data and as a source of images to label. The choices for removable media are both to be determined and likely to change over time. The size of temporary storage depends on the requirements for each user's terminal.

3.7 General Issues

The existing systems at Orpix have been developed and implemented using the C#/C++ .NET framework. This provides a starting point for us. Languages will then primarily be C# for high-end software and C++ for software with tight performance or space considerations. In terms of databases, currently a RDBMS is expected to be sufficient. Multiple scripting languages are supported, with MySQL the preferred choice for interfacing with the relational database. While the functional design focuses on providing a design that satisfies the software requirements, the technical design concentrates on implementation issues. We will then see in next chapter the decisions we took during the implementation phase of our labeling application.

4

Implementation

In this chapter we discuss the design considerations which brought us to the realization of the Labeling Application. Some techniques are discussed here in order to achieve better performances of the dataset. In Section 4.1 we discuss the functional description of the Application Layout. Section 4.2 is dedicated to our dataset extension, where we see how to enhance the object labels through the analysis of synonyms for the same object category, and how we can take advantage of an heuristic to discover semantically meaningful object-part relationships, to provide a list of part labels indicating how they occur with a given object label. In Section 4.3 we take into account the issue of quality of the labeling in contrast with the ease and speed of use of the application. Finally, we propose an overview of the Application Interface in Section 4.5.

4.1 Functional Description of the Application

To recognize an object class, one needs multiple images of different instances of the same class, as well as different viewing conditions. Many databases just provide captions, which specify that the object is present somewhere in the image. However, more detailed information, such as bounding boxes, polygons or segmentation masks, is tremendously helpful. We therefore desire a middle way between high quality labeling and ease of use of the user interface.

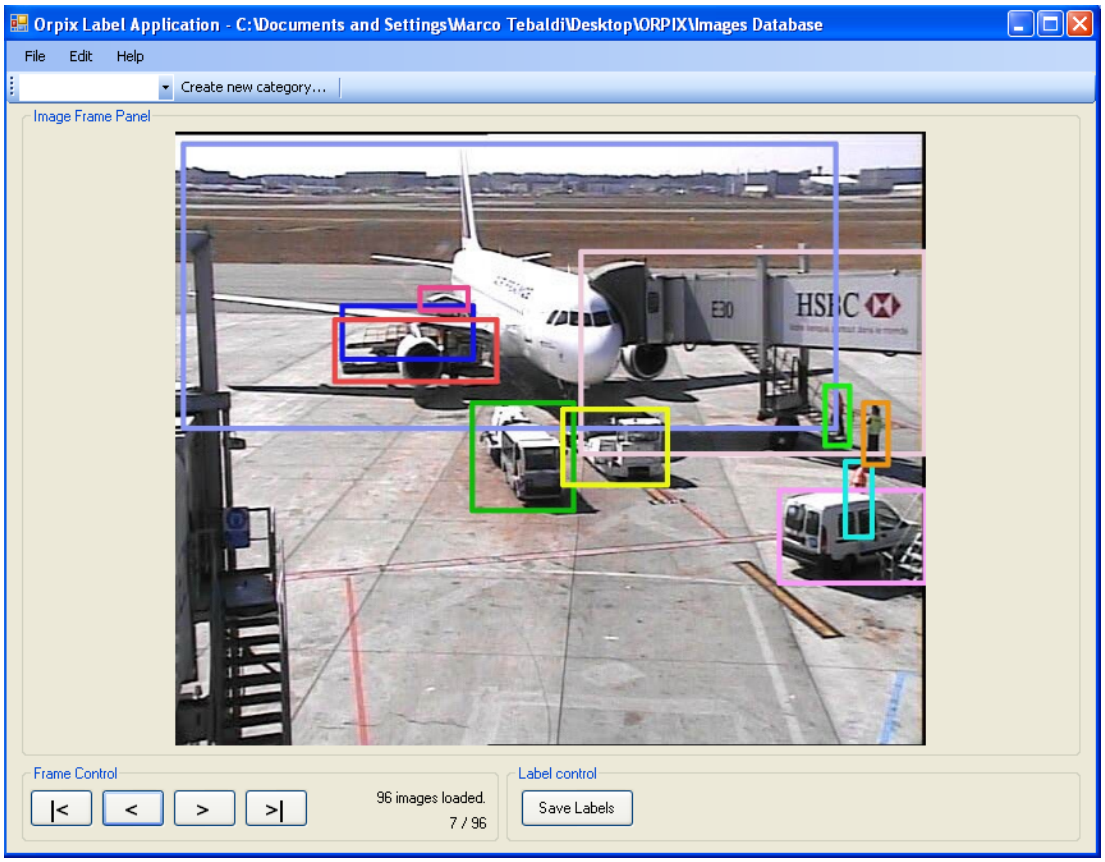

Figure 4.1: A screenshot of the labeling tool in use. - The user is shown an image along with possibly one or more existing annotations, which are drawn on the image. The user has the option of annotating a new object by clicking along the boundary of the desired object, and indicating its identity. The user may annotate as many objects in the image as he wishes.

To achieve this, we designed a C# drawing tool, as shown by the screenshot in Figure 4.1. The user is shown an image along with possibly one or more existing

annotations, which are drawn on the image. The user has the option of annotating a new object by clicking along the boundary of the desired object, and indicating its identity. The user may annotate as many objects in the image as he wishes.

The object's identity is defined by its category, that the user selects during the annotation process of labeling objects which appear in the images displayed by the application interface. The goal is to obtain a large number of examples in the dataset, for each object category.

In Figure 4.2 it is shown a screenshot taken during the development phase of our application. The program language used here is C#.

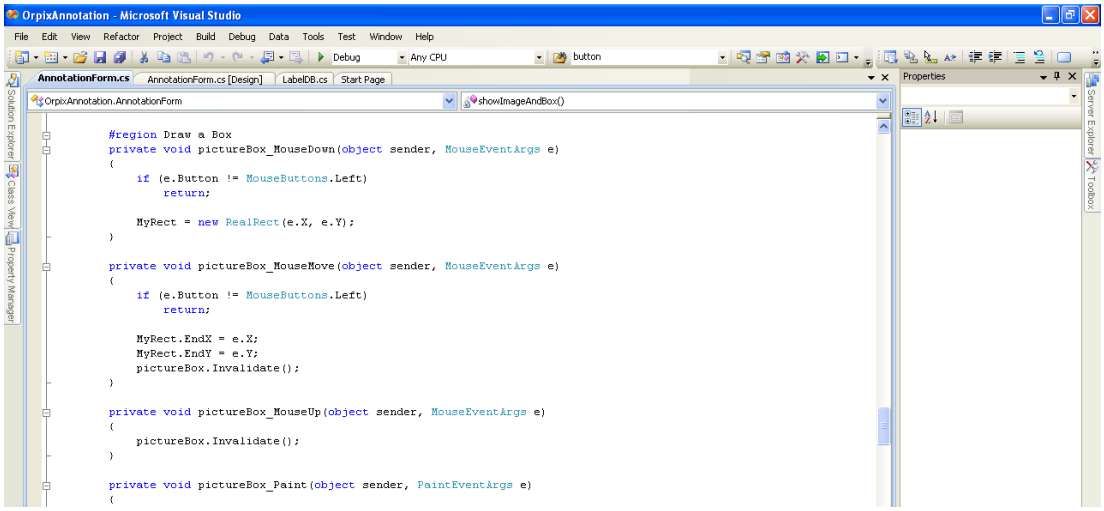

Figure 4.2: Writing the code in C# - A screenshot taken during the development phase of our application. The program language used here is C#.

The Application Layout will allow users to work on two Orpix technologies: Category Matching and Instance Matching.

- Category Matching – allows for matching a region in a query image to a general Category (i.e. Pedestrian, Face, Vehicles, License Plate, etc.);
- Instance Matching – allows for matching a region in a query image to one or more of a particular image in a large database of images.

The Labeling Application takes a directory of images and/or videos and allows the user to toggle through the images and video frames, and annotate a given frame.

Each annotation is assigned to a specific object, mode, and detector. We refer to this triple as a Category. When the user is done labeling, the application saves all the labels for a particular detector into a separate database entry. So for instance, if the user is labeling a set of images for two different detectors, Face Detector, and Pedestrian Detector, then two separate entries will be generated when the user saves the labels.

4.2 Extending the dataset

As seen in Section 2.4, training a detector is tremendously important to achieve quality and performances, and therefore the dataset plays a central role for our aims. We designed the Labeling Application so that our dataset will be populated with smart objects and enhanced object labels through the help of an electronic dictionary. Furthermore, it takes into account object-part hierarchies which are semantically meaningful. These features dramatically improve the dataset and its queries to match many more examples when the detector of a particular category is seeking for objects for its recognition. In Section 4.2.1 we see how to enhance the object labels through the analysis of synonyms for the same object category. In Section 4.2.2 we see instead how we can take advantage of an heuristic to discover semantically meaningful object-part relationships, to provide a list of part labels indicating how they occur with a given object label.

4.2.1 Enhancing Object Labels

Since the annotation tool does not restrict the text labels for describing an object or region, there can be a large variance of terms that describe the same object category. For example, an user may type any of the following to indicate the "car" object category: "car", "cars", "red car", "car frontal", "automobile", "suv", "taxi", etc. This makes analysis and retrieval of the labeled object categories more difficult since we have to know about synonyms and distinguish between object identity and its attributes. A second related problem is the level of description provided by the users. Users tend to provide basic-level labels for objects (e.g. "car", "person", "tree", "pizza"). While basic-level labels are useful, we would also like to extend the annotations to incorporate superordinate categories, such as "animal", "vehicle", and "furniture".

Person (27719 polygons)		Car (10137 polygons)	
Label	**Polygon count**	**Label**	**Polygon count**
person walking	25330	car	6548
person	942	car occluded	804
person standing	267	car rear	584
person occluded	207	car side	514
person sitting	120	car crop	442
pedestrian	121	car frontal	169
man	117	taxi	8
woman	75	suv	4
child	11	cab	3
girl	9	automobile	2

Table 4.1: Examples of label descriptions returned when querying the objects "person" and "car" after extending the labels with WordNet (not all the descriptions are shown) - For each description, the counts represents the number of returned objects that have the corresponding description. Note that some of the descriptions do not contain the query words. (Source: LabelMe Technical Report [16])

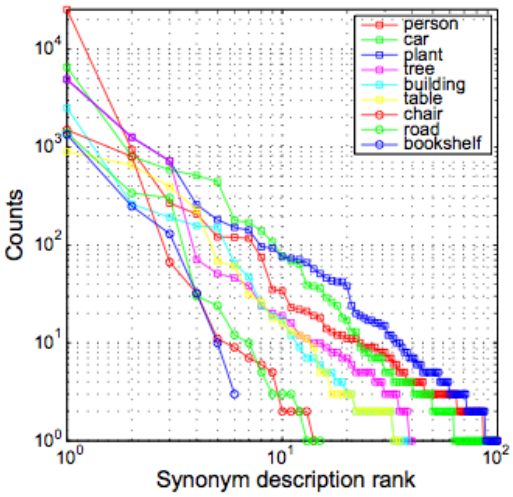

Figure 4.3: How the polygons returned by one query (in the WordNet-enhanced framework) are distributed across different descriptions - The distributions seem to follow a similar law: a linear decay in a log–log plot with the number of polygons for each different description on the vertical axis and the descriptions (sorted by number of polygons) on the horizontal axis. Table 4.1 shows the actual descriptions for the queries "person" and "car".

WordNet [9], an electronic dictionary, can be used to extend the label descriptions. WordNet organizes semantic categories into a tree such that nodes appearing along a branch are ordered, with superordinate and subordinate categories appearing near the root and leaf nodes, respectively.

We show what the benefit of adding WordNet to our Labeling Application would be to unify the descriptions provided by the different users. Table 4.1 shows examples of label descriptions that were returned when querying for "person" and "car" in the WordNet-enhanced framework. Notice that many of the original descriptions did not contain the queried word. Figure 4.3 shows how the number of polygons returned by one query (after extending the annotations with WordNet) are distributed across different label descriptions. It is interesting to observe that, considering the online LabelMe dataset [17], all of the queries seem to follow a similar law (linear in a log-log plot).

Category	**Original Description**	**WordNet Description**
person	27019	27719
car	10087	10137
tree	5997	7355
chair	1572	2480
building	2723	3573
road	1687	2156
bookshelf	1588	1763
animal	44	887
plant	339	8892
food	11	277
tool	0	90
furniture	7	6957

Table 4.2: Number of returned labels when querying the original descriptions entered into the labeling tool and the WordNet-enhanced descriptions - In general, the number of returned labels increases after applying WordNet. For entry-level object categories this increase is relatively small, indicating the consistency with which the corresponding description was used. In contrast, the increase is quite large for superordinate object categories. These descriptions are representative of the most frequently occurring descriptions in the dataset. (Source: LabelMe Technical Report [16])

Table 4.2 shows the number of returned labels for several object queries before

and after applying WordNet. In general, the number of returned labels increases after applying WordNet. For many specific object categories this increase is small, indicating the consistency with which that label is used. For superordinate categories, the number of returned matches increases dramatically. The object labels shown in Table 4.2 are representative of the most frequently occurring labels in the dataset.

One important benefit of including the WordNet hierarchy is that we can now query for objects at various levels of the WordNet tree. Figure 4.4 shows examples of queries for superordinate object categories. Very few of these examples were labeled with a description that matches the superordinate category, but nonetheless we can find them.

This is a great advantage we can bring to our implementation to increase the number of matching examples for each instance's category in the dataset.

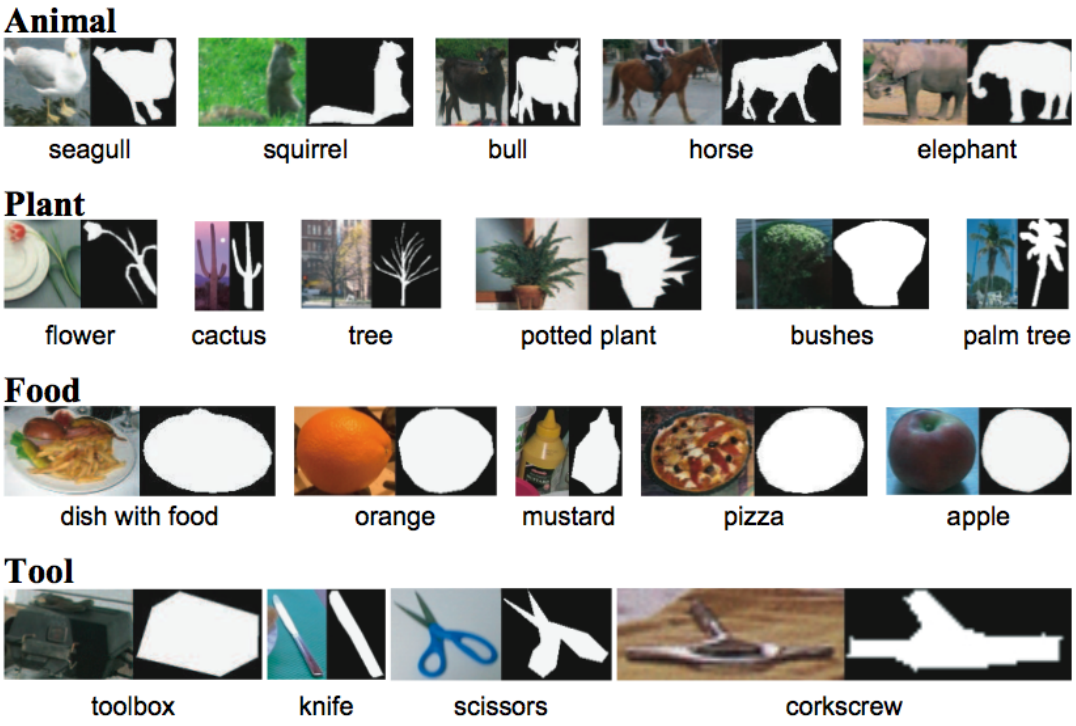

Figure 4.4: Queries for superordinate object categories after incorporating WordNet - Very few of these examples were labeled with a description that matches the superordinate category (the original descriptions are shown below each image). Nonetheless, we are able to retrieve these examples.

4.2.2 Object-parts Hierarchies

When two annotations have a high degree of overlap, this provides evidence of either (i) an object- part hierarchy or (ii) an occlusion.

Russell et al. [17] propose the following heuristic to discover semantically meaningful object-part relationships. Let I_O denote the set of images containing a query object (e.g. car) and $I_P \subseteq I_O$ denote the set of images containing part P (e.g. wheel). Intuitively, for a label to be considered as a part, the label's polygons must consistently have a high degree of overlap with the polygons corresponding to the object of interest when they appear together in the same image. Let the overlap score between an object and part polygons be the ratio of the intersection area to the area of the part polygon. Ratios exceeding a threshold of 0.5 get classified as having high overlap. Let $I_{O,P} \subseteq I_P$ denote the images where object and part polygons have high overlap. The object-part score for a candidate label is $N_{O,P}$ / $(N_P + \alpha)$ where $N_{O,P}$ and N_P are the number of images in $I_{O,P}$ and I_P respectively and α is a concentration parameter, set to 5. We can think of α as providing pseudocounts and allowing us to be robust to small sample sizes.

The above heuristic provides a list of candidate part labels and scores indicating how well they co-occur with a given object label. In general, the scores give good candidate parts and can easily be manually pruned for errors. Figure 4.5 shows examples of objects and proposed parts using the above heuristic. We can also take into account viewpoint information and find parts, as demonstrated for the car object category. Notice that the object-parts are semantically meaningful.

Once we have discovered candidate parts for a set of objects, we can assign specific part instances to their corresponding object. We do this using the intersection overlap heuristic, as above, and assign parts to objects where the intersection ratio exceeds the 0.5 threshold. For some robustness to occlusion, we compute a depth ordering of the polygons in the image and assign the part to the polygon with smallest depth that exceeds the intersection ratio threshold. Figure 4.6 gives some quantitative results on the number of parts per object and the probability with which a particular object-part is labeled.

Figure 4.5: Objects and their parts - Using annotation information alone, we automatically discover object-part relationships. We show example parts for the building, person, mountain, sky, and car object classes, arranged as constellations, with the object appearing in the center of its parts. For the car object class, we also show parts when viewpoint is considered.

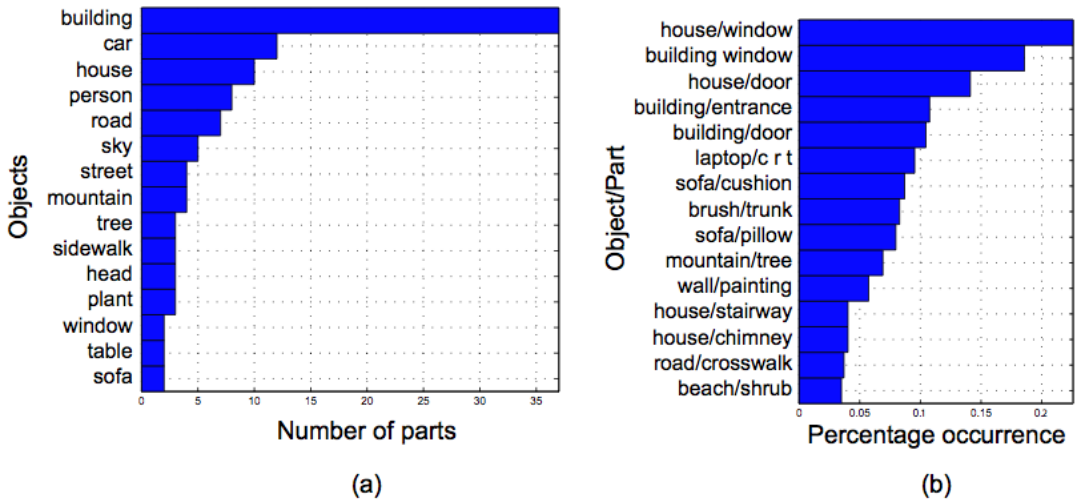

Figure 4.6: Quantitative results - Showing (a) how many parts an object has and (b) the likelihood that a particular part is labeled when an object is labeled. We are able to discover a number of objects having parts in the dataset. Also, a part will often be labeled when an object is labeled.

4.3 The Labeling Process

We have already discussed in Section 3.5 about the quality of the labels in terms of what to label and what not to label. Let's now see when we can consider the label itself as good or not, and make some considerations on the quality of the boundaries based on the examples of an existing dataset of labels, LabelMe [17].

4.3.1 Quality of the Annotation Boundaries

Figure 4.7 illustrates the range of variability in the quality of the polygons provided by different users for a few object categories. For the analysis in this section, we use the 44059 polygons provided by the online tool LabelMe. For each object category, we sort the polygons according to the number of control points. Figure 4.7 shows polygons corresponding to the 25th, 50th, and 75th percentile with respect to the range of control points clicked for each category.

Many objects can already be recognized from their silhouette using a small number of control points. Note that objects can vary with respect to the number of control points to indicate its boundary. For instance, a computer monitor can be perfectly

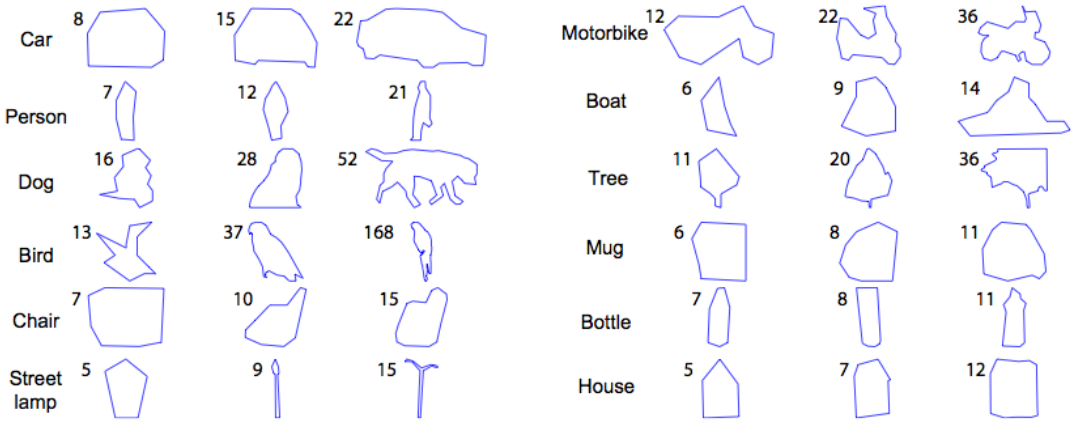

Figure 4.7: Illustration of the quality of the annotations in the dataset - For each object we show three polygons depicting annotations corresponding to the 25th, 50th, and 75th percentile of the number of control points recorded for the object category. Therefore, the middle polygon corresponds to the average complexity of a segmented object class. The number of points recorded for a particular polygon appears near the top-left corner of each polygon. Notice that, in many cases, the objects identity can be deduced from its silhouette, often using a small number of control points.

Figure 4.8: Image crops of labeled objects and their corresponding silhouette, as given by the recorded polygonal annotation - Notice that, in many cases, the polygons closely follow the object boundary. Also, many diverse object categories are contained in the dataset.

described, in most cases, with just four control points.

However, a detailed segmentation of a pedestrian might require 20 control points. Figure 4.8 shows some examples of cropped images containing a labeled object and the corresponding recorded polygon.

Figure 4.9: Annotation detail - Detail of the annotation used by our application using bounding boxes: the box surrounds the object avoiding to occlude other neighbor objects.

4.3.2 Bounding Boxes versus Polygonal Annotation

Even though the implementation in LabelMe of polygonal annotations implies a qualitative improvement of the labeling itself, we can't forget that we must keep the annotation process as simple as possible. From the end-user point of view, indeed, drawing polygons around each object may dramatically slow down the process. Ease of use and speed are key points of our labeling application. Therefore we will prefer the use of bounding boxes, which are also easier to store: only 4 XY-coordinates are required to keep the information of a box (top-left and bottom-right corners). End-users will then annotate each object simply by clicking on two corners to draw a box around it. The resulting box is similar to the one shown in the detail of Figure 4.9, where the box is as close as possible to the object's boundary.

When the user has an image to label in the "Image Frame Panel", and he previously selected a category from the menu, he only has to surround a box around the object to label, and to browse for other images. Also, the labeled data will be denoted by a bounding box of the visible area of the object, not the estimated total extent, in case of side poses or truncated objects[1]. When the user needs to annotate a different category, it's easy to browse through the existing categories from the drop-down menu, or to create a new category from the proper button on the toolbar.

4.4 Multiple annotations per image

The approach described in CBCL-Streetscenes dataset [2], discussed in Section 3.4.1, shows that the use of multiple annotations per image indicates that the images correspond to scenes and contain multiple object. We also follow the strategy used in LabelMe to populate the dataset with a number of labeled instances per object category.

Our annotation tool design choices emphasizes simplicity and ease of use.

Figure 4.10 shows how easy and clean the application interface appears. During this labeling process the user is annotating vehicles by drawing boxes around the objects. Note that in that case there are two cars annotated in the image, while an user might also label different category objects on the same image (e.g. the palmtrees).

With the added implementation of a hierarchy of the object-parts, also associated with synonyms and similar words in the description, we can dramatically increase the number of matches of a queried image.

[1] An object is generally "truncated" if more than $15 - 20\%$ of it lies outside the bounding box.

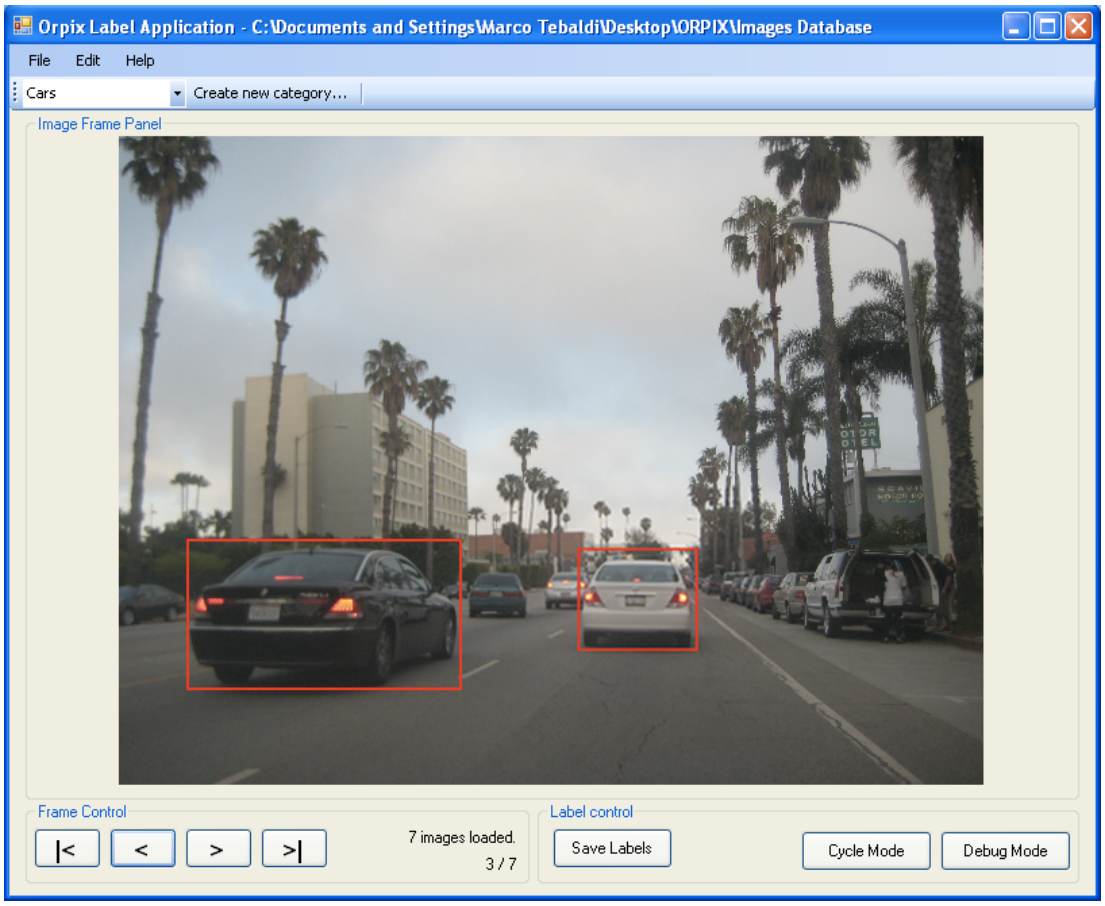

Figure 4.10: Drawing boxes with our Labeling Application - In Figure it is illustrated the Orpix Labeling Application. During this labeling process the user is annotating vehicles by drawing boxes around the objects.

4.5 Application Interface

When the user starts the Labeling Application, the interface displays a clean form with a few options. There may either be an image to start working on, if the user ran the application on the same machine before, or the option of selecting a particular directory as input of source images. The menu will always be active with the following options:

- File:
  - *New* - Label project: a reference file on which one will save the options, for example, for a specific set of categories to label.
  - *New* - Category: open a form to fill out when the user needs to create a new category during the labeling process.
  - *Open* - Label project: restores some customizations made for a specific set of labeled categories.
  - *Input Directory* - Opens a form to browse the local folders and select the desired directory.
  - *Save* - Saves the current Label project. Auto savings are set up and optional.
  - *Exit* - Saves the current Label project and closes the application.
- Edit:
  - *Category* - Accesses the database of categories and it's possible to modify each entry through an easy window form.
  - *Annotation* - Edits one or more annotations labeled in a selected category.
- Help:
  - *Guide* - Opens a quick user manual and FAQs.
  - *About* - Opens a window with credentials about the application, the company and the developers.

When the user is working on a set of images, the application shows its labeling options. The image is displayed in the Image Frame Panel, and it comes from a large image database covering a wide range of environments and several object categories.

The user may browse and select the desired category or define a new one by clicking on “Create New Category” on the tool bar, and filling out the form with name and description as well as other options. The application will save the category into a database entry. The user may also select a particular directory as input of source images.

The user can browse through images to label using the Frame Control, which offers proper buttons. The user may label a new object by selecting a mode (Box, Edge or Texture) and clicking control points along the object’s boundary to surround a box, or, if the selected mode is Edge or Texture, he finishes by clicking on the starting control point. Upon completion, the user either labels another object on the image, objects of the same category, or insert the object name in the Annotation Textbox. The user freely types in the object name and presses enter. This type of annotation makes use of shortcuts, which can be set by the user for faster frequent and repeated labellings.

This label is then recorded on the Label Database and is linked to its Category. The user is free to label as many objects depicted in the image as he chooses.

When he is satisfied with the number of objects labeled in an image, he may proceed to label another image from a desired set or press the “Next Image” or “Frame” button to see the next image (see Figure 4.11). The resulting labels are stored in the database, which can be exported and makes the annotations portable and easy to extend.

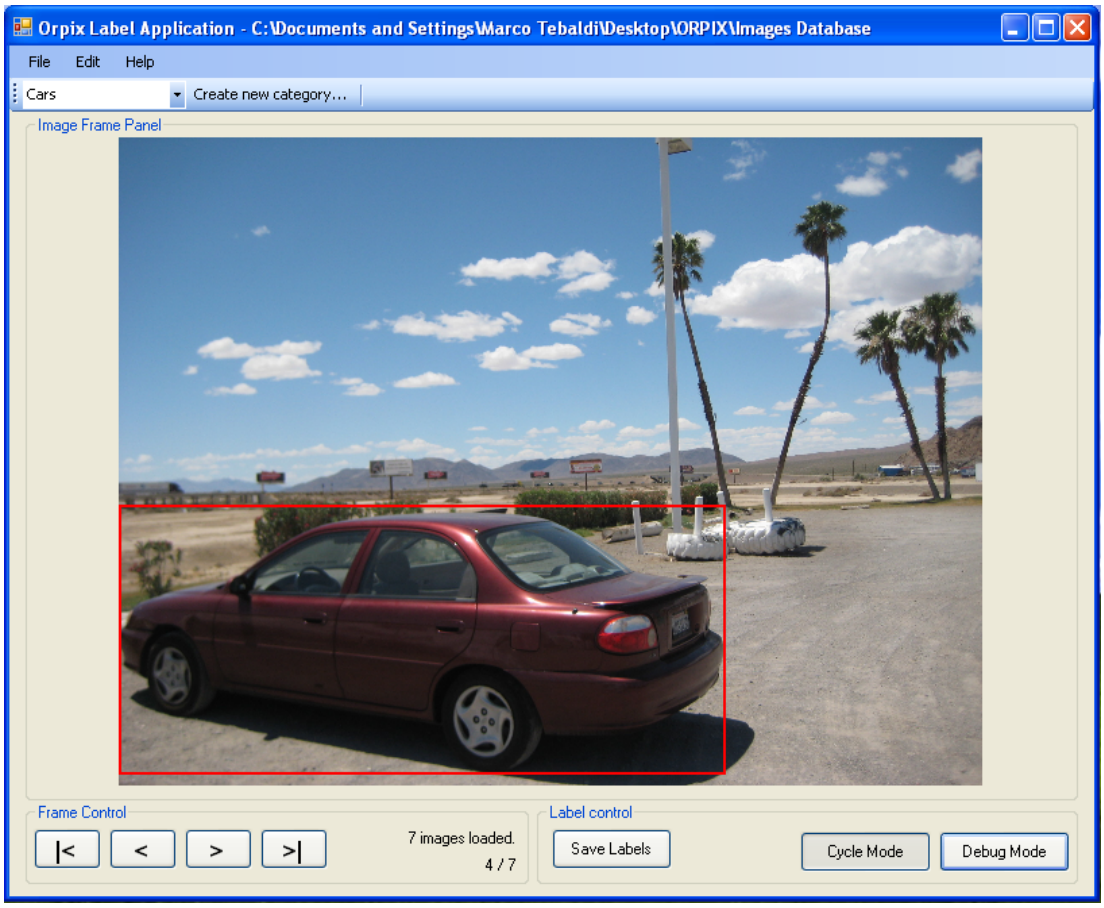

Figure 4.11: Annotating the objects in the Orpix Labeling Application - When the user is satisfied with the number of objects labeled in an image, he may proceed to label another image from a desired set or press the "Next Image" or "Frame" button to see the next image.

5

User Workflow

When an user opens the tool, an image from the database will be randomly selected and shown in Figure 5.1. He can then start by annotating as many objects as he wishes. Users mark objects belonging to a selected class (category) with bounding boxes. Images which are too complex / difficult to annotate with any certainty, or non photo-realistic, modified, or wrongly oriented images, are usually not to label. Users should not discard images solely because of e.g. challenging lighting.

Note that previously labeled objects may appear on the image. Users should not label previously labeled objects. Once an user has completed the image, he can view a new image by pressing the "Next Image" button from the Frame Control Panel.

The images and annotations should be possibly organized into folders, with the folder names providing information about the image contents of the depicted scenes/objects. The folders are grouped into two main categories: static pictures and sequences extracted from video.

Note that the frames from the video sequences are treated as independent static pictures and that ensuring temporally consistent labeling of video sequences is beyond the scope of this project.

In this chapter we summarize the most common operations during the labeling process as a step-by-step work flow. And eventually we give some useful tips and labeling guidelines.

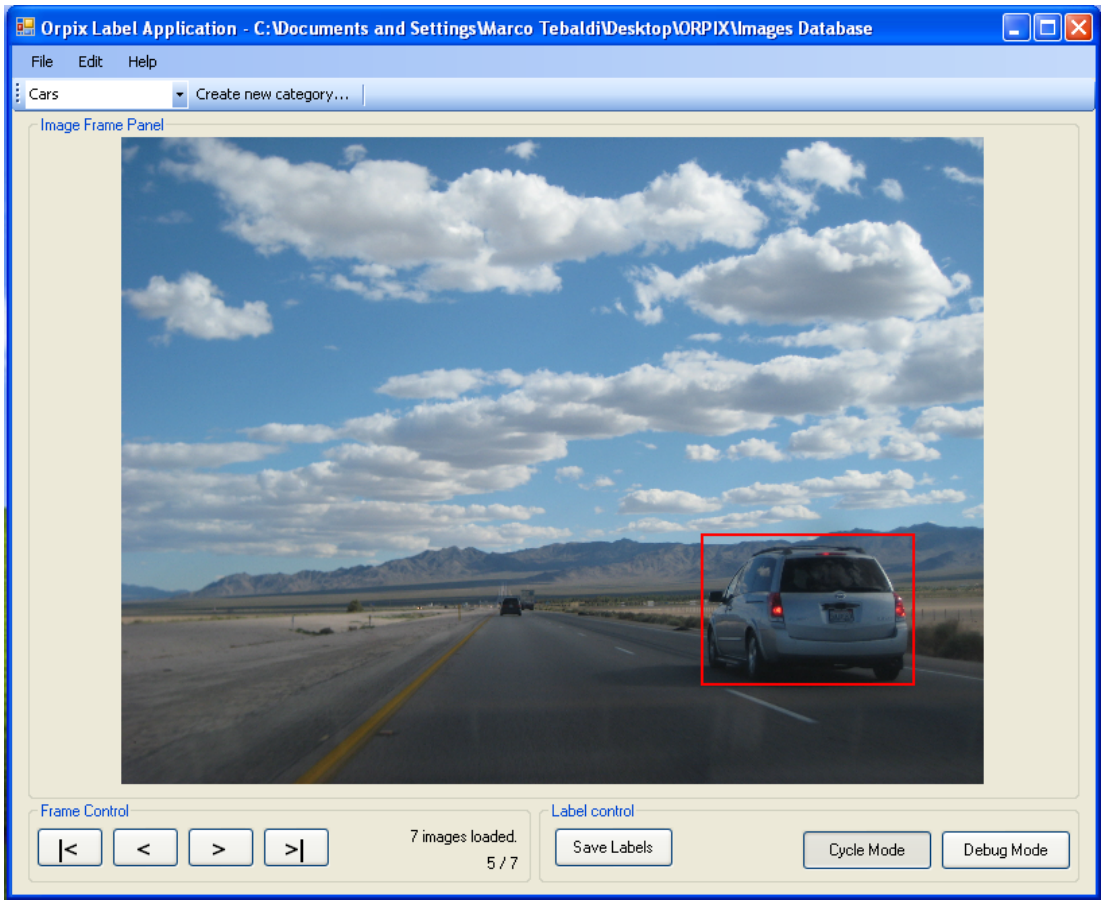

Figure 5.1: Opening the application: the tool is ready - The user is shown the interface. If the source has been selected, and image will be ready to be labeled. Note that previously labeled objects may appear on the image.

5.1 Input Images

To get started, we need to feed the application with some pictures in input, and then the user can start the labeling process to train the selected detector.

1. Click on File in the Menu.
2. Click Input Directory...
3. Browse the local folders and select the desired directory, then press OK.

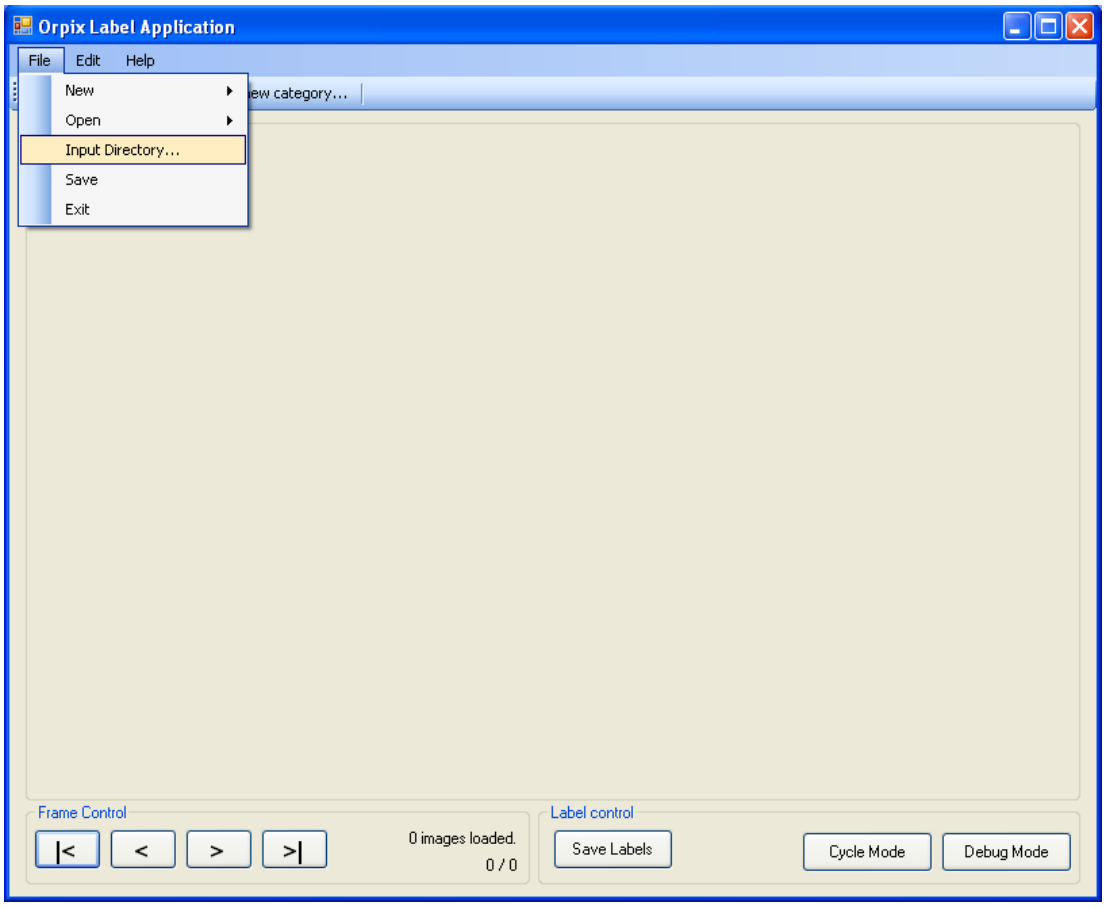

Figure 5.2: How to set the input folder of images to label - The user is shown the interface. Simply click on File, Input Directory and browse the desired folder of images or video.

5.2 Create a new Category

If the object to label belongs to a category which doesn't appear in the drop-down menu, then the user needs to create a new category. The steps are the following:

1. Click on "Create New Category...", on the toolbar, to open a new dialog box.
2. Fill out the form by selecting a name and description for the new category, as well as its color and mode (Box, Texture, Edge). Then press Save to start the labeling process.

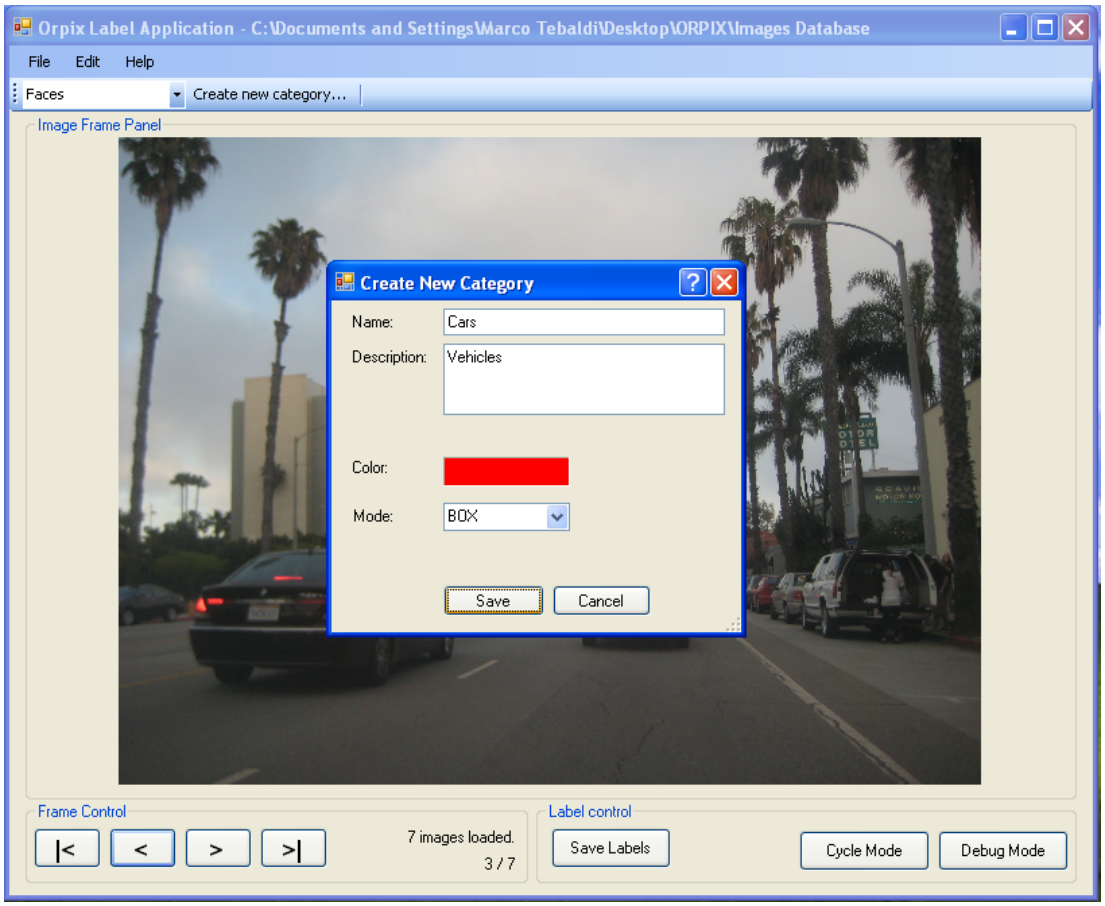

Figure 5.3: How to create a new category - Simply click on "Create New Category" and fill out the form by selecting a name and description for the new category, as well as its color and mode (Box, Texture, Edge).

5.3 Label an object

The following steps describe how to label an object displayed in the Image Frame Panel. If there are no images displayed, learn how to select an input directory in Section 5.1.

1. Select a category from the toolbar by clicking on the drop-down combo menu.
2. Cycle through the images by clicking on "Next" or "Previous" Image, on the Frame control panel. If the image is already labeled the application will show the labels for that category; the user can update such labels. Otherwise label the desired object by creating a boundary around the object, by clicking on the control points along the object's boundary to surround a box:
   (a) Start by pressing the left mouse button at some point along the boundary of the object (usually at the corners).
   (b) Continue clicking along the boundary of the object to create a bow which is surrounding the object.
3. An entry will be created assigning the annotated object to its category.
4. Save the Labels file by clicking on "Save Label".

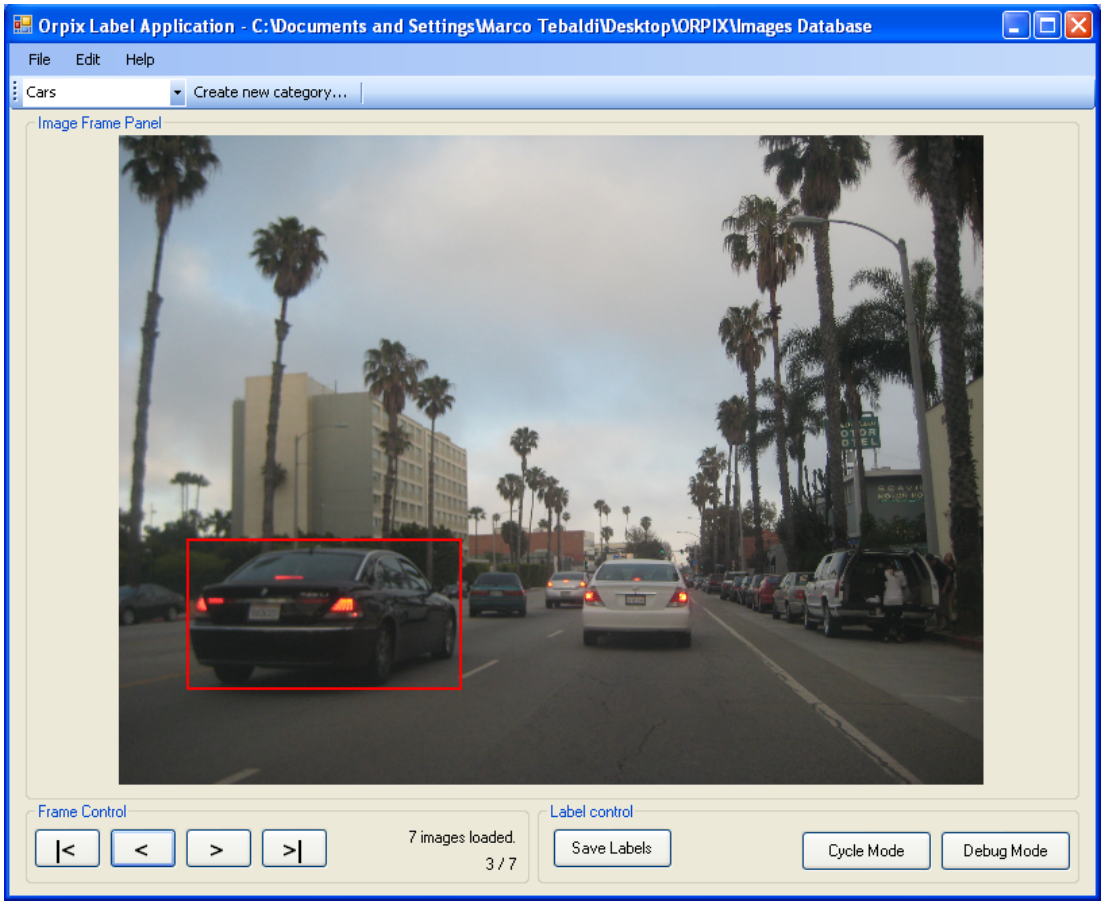

Figure 5.4: How to label an object - If the image is already labeled the application will show the labels for that category; the user can update such labels. Otherwise label the desired object by creating a boundary around the object, by clicking control points along the object's boundary to surround a box. Finally, click on Save the Labels file by clicking on "Save Label".

5.4 Cycle through categories or images

The following steps describe how to cycle through images belonging to a selected category or simply how to cycle through images coming from an input source.

1. Select or deselect "Cycle Mode" to cycle through the labeled instances or the input images, respectively.

2. Click on the drop-down menu in the toolbar, to select the desired Category to automatically:

   - Load all the labeled instances of the selected category, if Cycle Mode is selected;
   - or to cycle through input images which are loaded and can have either been previously labeled or not, if Cycle Mode is deselected.

3. Click on "Next" or "Previous" from the Frame Control panel to cycle through the loaded images.

4. If cycling through the labeled instances, the user can switch category anytime by selecting the desired category from the Toolbar drop-down category list.

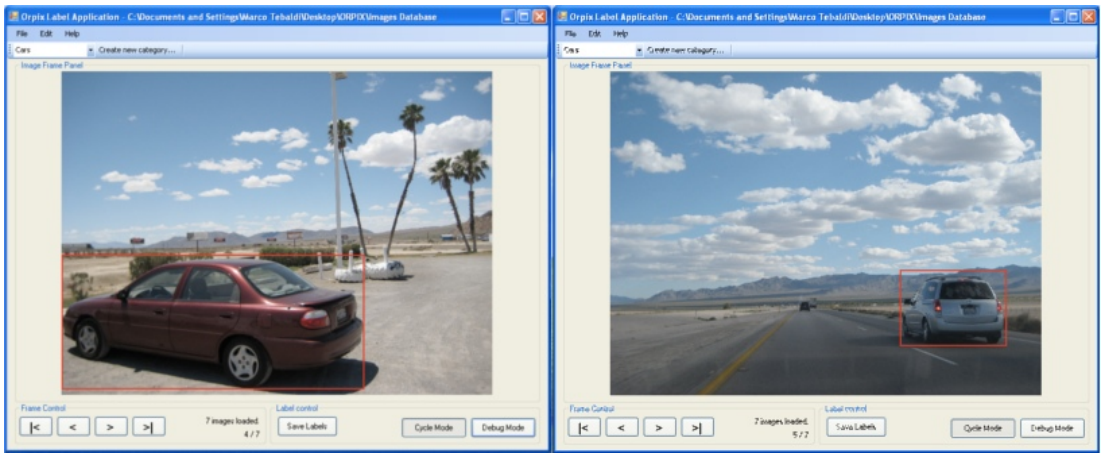

Figure 5.5: How to cycle through the database - Select or deselect "Cycle Mode" to cycle through the labeled instances or the input images, respectively. Then simply click on "Next" or "Previous" from the Frame Control panel to cycle through the loaded images.

5.5 Labeling guidelines

The following are some recomendations for how to provide annotations that are as useful as possible.

- Follow the object outline:

  Click on top of the object boundary so that the box accurately surrounds the object outline. The ideal outline should be as close as possible to the object's boundary.

- Labeling occluded objects:

  Sometimes objects will be only partially visible. This is especially true for regions like roads, buildings, etc., which become difficult to label when there are many occlusions. We recommend in these cases not to label the object if the occluded part is more than 20 %.

- Object naming:

  Use common English names for objects. Use a name that you think other people are likely to use to describe the same object. Users can use several words to describe an object. Example object names: sky, tree, building, road, sidewalk, person, car, chair.

6

Conclusions

Building a large dataset of annotated images with many objects is a costly and lengthy enterprise. Traditionally, datasets are built by a single research group and are tailored to solve a specific problem. Therefore, many currently available datasets only contain a small number of classes, such as faces, pedestrians, and cars.

Notable exceptions are the Caltech 101 dataset [8], with 101 object classes (this was recently extended to 256 object classes [11]), the PASCAL collection [4], and the CBCL-streetscenes database [2].

Web-based annotation tools provide a way of building large annotated datasets by relying on the collaborative effort of a large population of users [10, 16, 18, 26]. Recently, such efforts have had much success. The Open Mind Initiative [18] aims to collect large datasets from web users so that intelligent algorithms can be developed. More specifically, common sense facts are recorded (e.g. red is a primary color), with over 700K facts recorded to date. This project is seeking to extend their dataset with speech and handwriting data. Flickr [10] is a commercial effort to provide an online image storage and organization service. Users often provide textual tags to provide a caption of depicted objects in an image. Another way lots of data has been collected is through an online game that is played by many users. The ESP game [26] pairs two random online users who view the same target image. The goal is for them to try to read each others mind and agree on an appropriate name for the target image as quickly

as possible. This effort has collected over 10 million image captions since 2003, with the images randomly drawn from the web. While the amount of data collected is impressive, only caption data is acquired. Another game, Peekaboom [27] has been created to provide location information of objects. While location information is provided for a large number of images, often only small discriminant regions are labeled and not entire object outlines.

However, our goal is to provide a dynamic dataset that will lead to the training of detectors in the areas of object recognition and computer vision.

We presented quantitative results of dataset contents and other existing state of the art datasets used for object detection and recognition, showing and comparing the quality, breadth, and depth of each dataset. We showed how to enhance and improve the quality of our dataset through the application of WordNet, and heuristics to recover object parts to increase the dataset size and take into account viewpoint information and find parts. We then completed the design of our application and finally presented an easy User-Workflow.

An interesting future extension of our project would be for Semi-automatic labeling, which makes use of techniques of object recognition.

Once there will be enough annotations of a particular object class, one could train an algorithm to assist with the labeling and implement it to the labeling application. The algorithm would detect and segment additional instances in new images. At that point, the user task would be to validate the detection [24].

We could use the Orpix Labeling Application to train a detector and then run the detector on the retrieved unlabeled images. The user task will be to select the correct detections in order to further expand the amount of labeled data.

References

[1] I. Biederman. **Recognition by components: a theory of human image interpretation**. *Pyschological review*, **94**:115–147, 1987. 13

[2] S. Bileschi. **CBCL streetscenes**. Technical report, MIT CBCL (2006). The CBCL-Streetscenes dataset can be downloaded at `http://cbcl.mit.edu/software-datasets`. 22, 41, 55

[3] C. Bishop. **Pattern Recognition and Machine Learning**. *Berlin Springer*, 2006. 12

[4] M. Everingham, A. Zisserman, C. Williams, L. Van Gool, M. Allan, C. Bishop, O. Chapelle, N. Dalal, T. Deselaers, G. Dorko, S. Duffner, J. Eichhorn, J. Farquhar, M. Fritz, C. Garcia, T. Griffiths, F. Jurie, D. Keysers, M. Koskela, J. Laaksonen, D. Larlus, B. Leibe, H. Meng, H. Ney, B. Schiele, C. Schmid, E. Seemann, J. Shawe-Taylor, A. Storkey, S. Szedmak, B. Triggs, I. Ulusoy, V. Viitaniemi, and J. Zhang. **The 2005 Pascal Visual Object Classes Challenge**. *First PASCAL Challenges Workshop*, **3944, Springer**:117–176, 2005. 55

[5] M. Everingham, A. Zisserman, C.K.I. Williams, and L. Van Gool. **The 2006 Pascal Visual Object Classes Challenge (voc 2006) results**. Technical report, September 2006. The PASCAL2006 dataset can be downloaded at `http://www.pascal-network.org/challenges/VOC/voc2006/`. 22

[6] L. Fei-Fei, R. Fergus, and P. Perona. **One-shot learning of object categories**. IEEE Trans. Pattern Recognition and Machine Intelligence, In press. The Caltech 101 dataset can be downloaded at `http://www.vision.caltech.edu/ImageDatasets/Caltech101/Caltech101.html`. 22

[7] L. Fei-Fei, R. Fergus, and P. Perona. **A bayesian approach to unsupervised one-shot learning of object categories**. *IEEE Intl. Conf. on Computer Vision*, 2003. 14

[8] L. Fei-Fei, R. Fergus, and P. Perona. **Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories**. *IEEE CVPR 2004 Workshop on Generative-Model Based Vision*, 2004. 55

REFERENCES

[9] C. Fellbaum. **Wordnet: An Electronic Lexical Database**. Bradford Books, 1998. 34

[10] Flickr. **Photo Sharing Service**. `http://www.flickr.com`. 55

[11] G. Griffin, A.D. Holub, and P. Perona. **The Caltech-256**. Technical report, California Institute of Technology, 2006. 55

[12] B. Heisele, T. Serre, S. Mukherjee, and T. Poggio. **Feature reduction and hierarchy of classifiers for fast object detection in video images**. *CVPR 2001*, 2001. 13, 14

[13] D. Hoiem, A. Efros, and M. Hebert. **Putting objects in perspective**. *CVPR 2006*, 2006. 14

[14] B. Leibe and B. Schiele. **Analyzing appearance and contour based methods for object categorization**. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, June 2003. 13

[15] K. Levi and Y. Weiss. **Learning object detection from a small number of examples: the importance of good features**. *CVPR 2004*, **2(2)**:53–60, July 2004. 11

[16] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. **LabelMe: a database and web-based tool for image annotation**. *Technical Report AIM-2005-025. MIT AI Lab Memo*, Sep 2005. 22, 33, 34, 55

[17] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. **LabelMe: a database and web-based tool for image annotation**. *Intl. J. of Computer Vision*, **77(1-3)**:157–173, May 2008. 23, 34, 36, 38

[18] D.G. Stork. **The open mind initiative**. *IEEE Intelligent Systems and Their Applications*, **14(3)**:19–20, 1999. 55

[19] E. Sudderth, A. Torralba, W. T. Freeman, and W. Willsky. **Learning hierarchical models of scenes, objects, and parts**. *IEEE Intl. Conf. on Computer Vision*, 2005. 14

[20] A. Torralba. **Contextual priming for object detection**. *Intl. J. Computer Vision*, **53(2)**:153–167, 2003. 14

[21] A. Torralba, K. Murphy, and W. Freeman. **Sharing features: efficient boosting procedures for multiclass object detection**. *CVPR 2004*, 2004. 14

[22] M. Turk and A. Pentland. **Eigenfaces for recognition**. *J. of Cognitive Neuroscience*, **3(1)**:71–86, 1991. 14

[23] V.N. Vapnik. **The Nature of Statistical Learning Theory (2nd Ed.)**. *Springer Verlag*, 2000. 14

[24] T. Vetter, M. Jones, and T. Poggio. **A bootstrapping algorithm for learning linear models of object classes**. *CVPR 1997*, July 1997. 56

[25] P. Viola and M. Jones. **Rapid object detection using a boosted cascade of simple classifiers**. *CVPR 2001*, 2001. 13, 14

[26] L. von Ahn and L. Dabbish. **Labeling images with a computer game**. *Proc. SIGCHI conference on Human factors in computing systems*, 2004. 55

[27] L. von Ahn, R. Liu, and M. Blum. **Peekaboom: A game for locating objects in images**. *ACM CHI*, 2006. 56

[28] J. Winn, A. Criminisi, and T. Minka. **Object categorization by learned universal visual dictionary**. IEEE Intl. Conf. on Computer Vision (2005). The MSRC dataset can be downloaded at `http://research.microsoft.com/vision/cambridge/recognition/default.htm`. 22

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

The thesis work was conducted from January to October 2010 under the supervision of Ori Ben-Haim at Orpix-inc San Diego, Serge J. Belongie, University of California San Diego, and Emanuele Menegatti, University of Padova.

San Diego, California, October 2010.