Efficient interfaces for accurate annotation of 3D point clouds

Aleksey S. Boyko

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Thomas A. Funkhouser

March 2015

Abstract

Collecting massive 3D scans of real world environments has become a common practice for many private companies and government agencies. This data represents the real world accurately and densely enough to provide impressive visualizations. However, these scans are merely points. To truly tap into the potential that such a precise digital depiction of the world offers, these scans need to be annotated in terms of objects these points represent. Manually annotating this data is very costly. Existing machine-aided methods report high accuracies for object localization and segmentation. However, the central task of annotation, proper label assignment, is still a challenging task for these approaches.

The goal of this work is to design an interface that streamlines the process of labeling objects in large 3D point clouds. Since automatic methods are inaccurate and manual annotation is tedious, this work assumes the necessity of every object's label being verified by the annotator, yet puts the effort required from the user to accomplish the task without loss of accuracy at the center stage.

Inspired by work done in related fields of image, video and text annotation, techniques used in machine learning, and perceptual psychology, this work offers and evaluates three interaction models and annotation interfaces for object labeling in 3D LiDAR scans. The first interface leaves the control over the annotation session in the user's hands and offers additional tools, such as online prediction updates, group selection and filtering, to increase the throughput of the information flow from the user to the machine. In the second interface, the non-essential yet time consuming tasks (e.g., scene navigation, selection decisions) are relayed onto a machine by employing an active learning approach to diminish user fatigue and distraction by these non-essential tasks. Finally, a third hybrid approach is proposed—a group active interface. It queries objects in groups that are easy to understand and label together thus aiming to achieve advantages offered by either of the first two interfaces. Em-

pirical evaluation of this approach indicates an improvement by a factor of 1.7 in annotation time compared to other methods discussed without loss in accuracy.

Acknowledgements

There are many people who helped and inspired my work, thought, and curiosity along the way, and helped with shaping this work and making it happen.

Foremost, I would like to thank my advisor Professor Thomas Funkhouser for being an example and a role model for me throughout the years of graduate school. Work ethics, patience, wisdom, teaching style, readiness to roll up his sleeves and get to work, while not forgetting to be a great human being are only a few of the things that I hope I managed to learn and will be able to practice anywhere near to the extent that I observed in this man daily.

I would like to thank my committee members Professors Adam Finkelstein, Brian Kernighan, Szymon Rusinkiewicz, and Jianxiong Xiao for their feedback and comments, and inspiration by example.

In addition, I would like to thank Neptec, John Gilmore, and Wright State University for the 3D LiDAR dataset. I am grateful to Aleksey Golovinskiy for paving the way for my work, and especially for the extensive code base that facilitated my path into the world of LiDAR scans and point clouds.

This work would not have been possible without the funding from NSF (IIS-1251217, CCF-0937139), Google, and Intel (ISTC-VC).

I am grateful to David Mimno, Sean Gerrish, Robert Shapire, and David Blei for having, often unexpected, conversations and providing valuable insights into machine learning. I would also like to thank Rebecca Fiebrink for offering her knowledge of the field of human-computer interaction, and Nicholas Turk-Browne for sharing his wisdom in the field of neuroscience of attention, perception and memory. I am very grateful to all of the members of the Princeton Graphics, Vision and Sound Labs and other attendees and organizers of the Tiggraph retreats that tirelessly provided feedback on numerous drafts that are constituting most parts of this work. And I am

extremely grateful to 75 patient and anonymous users that contributed their time to multiple evaluations of my ideas.

I would like to thank Dmitri Dolgov, Martin Stumpe, Xiaofeng Ren, and Jiajun Zhu for making my internship experiences in the industry exciting and fulfilling. These intermissions helped me a lot to gain better perspective, clear my head, refresh my motivation, and regroup during my years in academia.

I am thankful to Nick and Daya Johnson, Katy Ghantous, Dan Reynolds, and Tatiana Boyko for going with me through the highs and lows of the graduate school years. I am especially grateful to Nick Johnson for constantly offering feedback on my writing and Tatiana Boyko for teaching me how to use less commas and more periods.

Finally, I would like to thank Jeff Dwoskin '10 for the LaTeX template and document class that has been used in the creation of this work.

To my dad and his dad, who wanted me to make my own choices.

To my mom and her mom, who wanted me to aim higher.

To Tanya, who wanted me to be happy.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Context

Humans have been actively collecting digital representation of the world for a long time. Recently, there has been an explosion in the amount of 3D data collected in urban environments, as several companies (e.g., Google [133, 115], Microsoft [65] and Navteq [48]) and government agencies (e.g., NASA [107], USGS [114], and NOAA [20]) are continuously collecting LiDAR data using scanners mounted on cars and/or airplanes. For example, one such data set combining aerial and terrestrial scans of Ottawa, Canada is shown in Figure 1.1.

LiDAR data is dense and accurate (e.g., as one shown in Figure 1.2), however, without proper processing this data is just a cloud of points in 3D space. While it provides immediate opportunity for visualization, its true value is not realized until semantic objects in the data are segmented and labeled (e.g., as in Figure 1.3). If a semantically annotated 3D model of a city could be acquired, then applications such as urban planning, augmented reality maps, virtual tourism, and emergency response planning can be greatly enhanced. Identifying locations of stop signs, traffic lights, and street signs can augment electronic maps and help guide navigation of

Figure 1.1: Scan of $6km^2$ in downtown Ottawa.

self-driving cars. Labeling fire hydrants, electrical power boxes, and fire escapes can help emergency response planning and disaster simulation.

A number of researchers have begun to develop systems aimed at automatic segmentation and labeling of 3D LiDAR point clouds. Due to the large size of collected point clouds automatic algorithms are preferable to reduce the necessity of expensive and slow human processing [53]. In particular, great progress has been made over the last several years on recognition of large urban structures [80]. Building extraction methods are proposed for low resolution aerial scans [17, 35, 72, 86] and partial scans of buildings [24]. A facades reconstruction algorithm, capable of filling occluded geometry and texture from densely sampled clouds, is proposed in [41], while a faster but lower-quality approach is described in [19]. Numerous road extraction methods were proposed for both aerial scans [28, 54] and terrestrial scans [61, 142]. A number of foliage detection and reconstruction methods have been proposed as well, e.g. [131, 137].

Smaller objects, such as cars, signs, and fire hydrants (Figure 1.2), populate environments and carry useful information about how this environment functions and how it is used by its inhabitants. Unfortunately, recognition of small objects is more difficult as indicated by the accuracies of current labeling algorithms being relatively poor for them. For example, state-of-the-art labeling algorithms based on supervised machine learning [46, 75, 116] achieve only $42\% - 82\%$ accuracies depending on the number of categories and their generality. This is far lower than the $95 + \%$ that is required for consumer mapping, augmented reality, and urban simulation applications. Therefore, to achieve such accuracies using existing approaches a person has to check and fix the predicted label for *every* object before a data set can be deployed to users (a practice common even for large urban structures at most companies that annotate electronic maps). This effectively undermines the intention of using existing algorithms for acquiring small object annotations to save human effort.

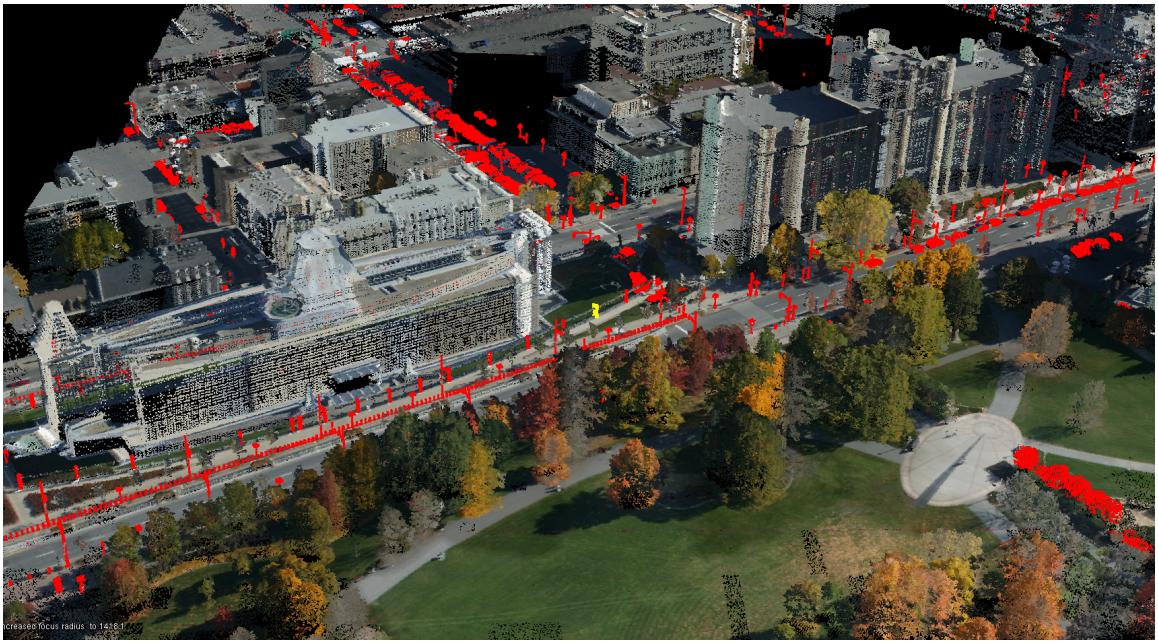


Figure 1.2: 3D point cloud of a section of the Ottawa scan. Small object are highlighted in red, other points are shown in the colors derived from camera.

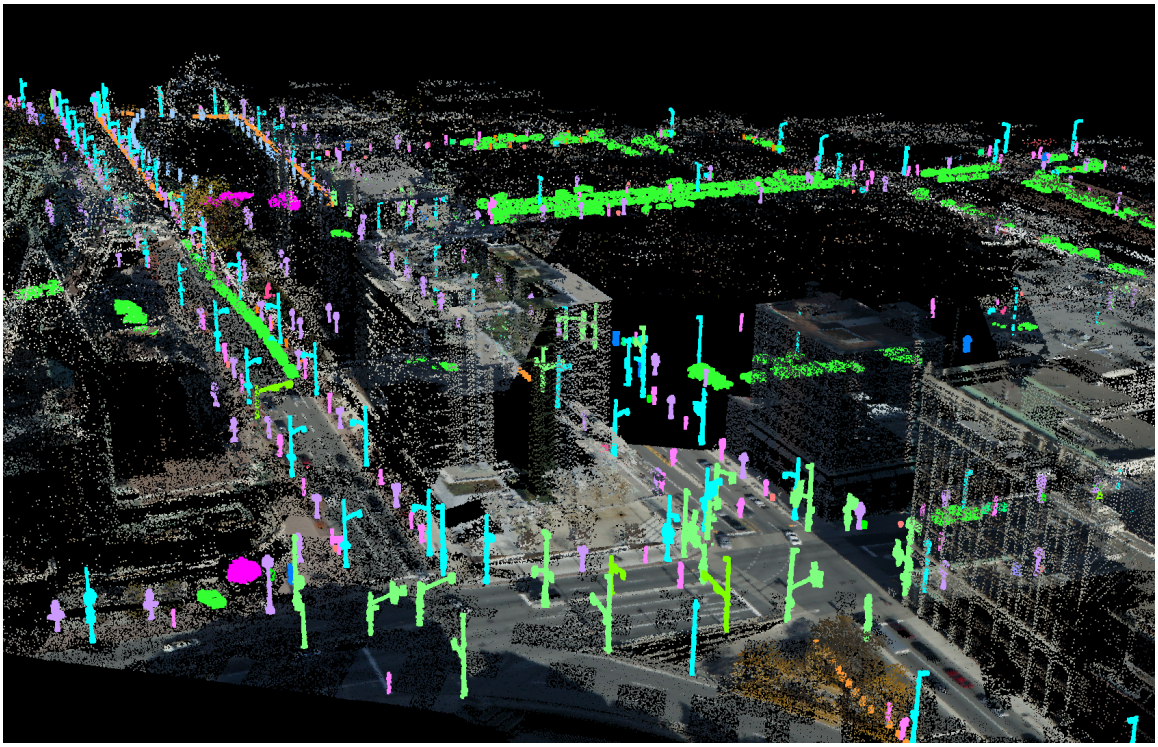


Figure 1.3: 3D point cloud with semantic labels of objects represented by color.

1.2 Overview

This work has the goal of efficiently acquiring accurate labeling for small objects in 3D point clouds. While sharing the goal with previous works, this work approaches the problem of object annotation as an interface problem [15]. If a machine cannot meet the required accuracy and needs continuous guidance and interaction with the user, as valuable as it is to continue improving the learning component of the system, it is important to make the existing algorithms efficiently interact with the currently inevitable human presence in the loop. Even more so since a human is the element that, depending on expertise and incentives, is expensive, hardest to engage and train, and prone to fatigue and frustration. Two major factors that contribute to the cost, motivation to participate, fatigue, and frustration in this case are the effort required by the interface to convey annotations and the time commitment expected of the user to annotate entire data sets. Thus to develop an interactive system for high-throughput and high-accuracy labeling it is important to understand what part of the annotation of 3D data has to be done by a human to enforce accuracy, and what can be safely delegated to the machine to improve speed and reduce the interaction complexity. A good interface should reflect this workload separation and implement efficient information exchange protocols between a user and the machine.

The approach taken in this work is different from ones in previous systems. First, the objective of the interfaces is to allow to confirm a label for *every* object in a specific data set with as much computer assistance as possible. This is different than previous work in computer vision focusing on crowd-sourced creation of object recognition benchmarks where computer assistance is specifically avoided to ensure unbiased ground truth labels (e.g., [32, 135]). Second, the goal is to minimize the total time required to provide and/or confirm a label for every object in a data set. This is different than most previous work on interactive machine learning where the goal is mainly to maximize the accuracy of a classifier with the fewest training

examples(e.g.,[101]). Finally, the goal is to label all objects residing in a single 3D environment, and thus it is possible for the system to show multiple objects to a user in a single view and ask him/her to label them with a single command (e.g., if they all require the same label). This is different than almost all previous interactive labeling and classification systems where objects must be displayed and/or labeled separately.

Following Chapter 2, which provides more thorough exploration of existing work, this dissertation explores the design space of interfaces for small object annotation in 3D in the following directions. Chapter 3 describes an interactive user-guided(UG) interface that extends a simple labeling interface by employing an online machine learning algorithm, keeping the annotator updated on the most current predictions. This change in turn justifies additional visualization and interface tools aimed at streamlining the annotation process. The main point of this interface is to provide users with more information and better tools while keeping them completely in control of the process and providing annotations according to their liking and preference. An interface described in Chapter 4 takes a step in the opposite direction by having the machine guide the annotation sequence. This interface employs an active learning(AL) approach that selects objects for an annotator to label based on a utility function derived from the classifier's uncertainty measure. This interface takes an extreme approach of reducing the involvement of a human user to a bare minimum. An interface presented in Chapter 5 takes a middle ground between the above two. It actively queries the user about objects to reduce the interactions required from the user. However, instead of using the classifier's evaluations for selection, it uses a human visual perception model to select groups that can be rapidly recognized and labeled together. Thus the third interface is named a group active(GA) interface. Finally, Chapter 6 summarizes the contributions and suggests avenues for future work.

Chapter 2

Related Work

Interface designs for annotation have been explored in related fields and thus provide both context and inspiration for this work. Large and growing size of available text, sound, images, video, and LiDAR scans data has prompted researchers to explore avenues for acquiring semantic annotation of entities contained within this data. Often, employed labels are expected to be useful to and understandable by humans. Thus many existing approaches rely on human annotators to provide such information, varying in the amount of effort required from the users and, respectively, how much of it is delegated to a machine.

This chapter describes the existing approaches to acquiring semantic annotation of LiDAR point clouds, as well as relevant methods and interfaces for acquisition of annotation in a broader range of data types (e.g., images, videos and text) ranging from automatic to fully manual and crowd-sourced interfaces. It also describes the works that model and evaluate the effort required from human operators when they perceive and process the unannotated data and convey annotations back to a machine.

2.1 Massive manual human annotation

The most straightforward way of acquiring a human-understandable annotation of data is directly requesting humans to provide it. For example, [95] describes a web tool called LabelMe that acquires image segmentations and labeling from the users of a web interface. An interface offers contour drawing tools and a menu of labels and, despite the annotation task with these tools being laborious, due to tapping into a large pool of online users, it allowed for a very large collection of high quality annotations. Acquired annotations were good enough to be useful for successful 3D scene reconstruction in [94]. In a follow-up work [112] the authors extended LabelMe web tool to allow for video annotation, and explored the influence of the database size on performance of vision algorithms. Also working with videos, [136] first reconstructs a scene from motion and then asks a user to interactively fix segmentations and label objects in the scene on separate video frames, with these corrections then propagated through the rest of the video.

Considerations of not only acquiring the annotation but doing it in a cost-efficient manner have been investigated as well. In [127, 125] the authors extensively explore the design space of a large scale video annotation tool. They evaluate strategies for incentivizing users and allocating monetary budget between paying human annotators and computational costs for inter-frame interpolation algorithms. For collecting a benchmark for video annotation in [84] non-expert users provided simpler parts of information by marking object tracking through frames while expert users then labeled events. In [109], to improve reliability of information acquired from online users, the authors proposed to dissect a complex task of annotating objects in images into three simple atomic tasks. Different users either provided localization of an object of requested category by drawing a box around it, verified a bounding box provided by someone else, or checked that all objects of a given category in the image had their

own box. All three tasks required much simpler decisions and interactions from any single user and allowed to improve both reliability and throughput of the interface.

Alternatively to motivating users monetarily for doing routine or complex tasks, games with purpose were introduced. The first work in this domain [124] proposed to acquire image annotation by making users try to agree on a label. Another example, the Bubble game proposed in [33], allows users to select a small circular area of a blurred image to see it sharper with the goal of identifying what exactly is shown in the image in smallest number of such interactions. Observations of human activity allowed an algorithm to learn what elements are relevant to humans in understanding and discerning fine-grain categories, such as bird species.

Methods referenced above focus on accurate and reliable annotation acquisition and aim to achieve scalability by dispersing the task to many users, either incentivizing them to be reliable or algorithmically accommodating for lack of reliability. The data sets acquired by these methods were shown to indeed be large and accurate since humans actually put most of the work into providing all of them. Because of this, however, these methods are expensive in man-hours. This is a serious practical concern. For example, if labels that were used in some annotation are not relevant for a small set of data consumers(e.g., too general for domain specialists), reacquisition of relevant annotation using the above approaches is prohibitively expensive.

2.2 Machine-aided interactive annotation

There is a large body of work directed at decreasing the necessary human effort in annotation by delegating parts of the labeling task to a machine. Some methods only require a user to provide a *batch* of training examples and expect the classifier to predict the labeling for the rest of the data. Others take a more interactive approach, where the annotator is continuously provided with additional information and feedback to help them better allocate their effort. In such collaborative efforts communication between the two parties is important and hence the human-computer interface design becomes as relevant as deciding on the work separation itself.

2.2.1 Batch learning classification and detection

A direct alternative for fully manual annotation is to acquire large portions of labeling "automatically", i.e. by employing an algorithm that is either explicitly designed to detect certain categories of objects (effort placed on the designer of the algorithm) or has been trained by a set of examples to do that(effort placed on the annotator). An extensive survey in [80] addresses a wide range of such methods employed for recognition and reconstruction of objects in 3D scans of urban environments. Explicitly encoding a model directly into an algorithm lacks the generality that can universally address labeling, given the number of categories and appearance varieties within them that the real world data offers.

More general methods employ some version of machine learning in the sense that discriminating ability is trained by a set of examples provided by a human and then employed to make predictions for a larger data. This approach is an instance of supervised learning. When training occurs on a batch of labeled instances simultaneously it is sometimes referred to as *batch learning*. The process in this case is usually separated into a sequence of, first, a batch of training examples being collected by

a human, then a model trained and predictions are produced. The variety in these methods usually concerns the kind of information about objects that the algorithms capture and how a model is built.

A number of batch learning methods proposed take a *bottom-up* approach to object recognition, where a label is assigned to each point based on local properties of the point cloud around it. In [73] points are labeled using a Bayesian classifier. Several works employ Markov Random Fields [5, 110, 134] and their extensions, such as Associative Markov Network [79, 78], where the label of a point not only depends on local shape descriptors, but also on the labels of its neighbors. These approaches successfully distinguish between such classes as wire, pole, ground, wall, scatter, which were shown to be useful for mobile robotics. Bottom-up approaches have also been successfully used for single category detection, or a two-class problem (e.g., distinguishing cars from non-cars). [91] investigates an application of recurrent neural network directly to LiDAR data that arrives in natural temporal order to derive signatures of objects that look like cars; evaluation reports 96% accuracy. Alternatively, [58] employs SVM with point feature histograms to a similar problem to report 96.7% accuracy. The last work is an instance of a *top-down* approach as feature histograms are collected not over points, but over entire objects that are created during a segmentation phase. A method described in [88] uses bottom-up hypotheses generation using local spin images [62] and a top-down constellation model verification approach to allow for a single category (cars) object detection, reporting 92% precision and 74% recall with a 17 objects training set. Although these methods report high accuracy of classification, unfortunately, they are only shown to work with a small number of classes, often having features and algorithms hand-tuned for given categories of objects.

More relevant to this dissertation are approaches that address annotation of small objects in point clouds, designed to support more categories. Work in [98] represents

point clouds in terms of primitives that fit the points, allowing to reformulate object detection as a graph search problem. Users then can request the system to detect given categories as long as they can encode that category using a primitives and a relationship graph. This approach is versatile, however it has a number of serious drawbacks. The vocabulary and scale of primitives that define the space of searchable shapes is provided before the most computationally expensive part of execution—construction of the graph. If the preexisting vocabulary of primitives is not expressive enough for some categories, refitting has to be done. Also, the necessity for the human to encode all categories manually with primitives is tiring and the authors suggest learning by example as a solution to this drawback. An approach described in [116] combines bottom-up hypothesis detection and further labeling by a category-depended voting for the object's centers. Results are reported for 2 categories of objects (cars and light poles) and the authors report 66-72% precision and 70-82% recall.

The approach described in [46] is the closest related to this work. It proposes a pipeline of separate localization, segmentation and classification stages and the qualities achieved by a number of algorithms are evaluated for each stage. Results of annotating objects in 17 categories are detailed in Table 2.1. Specifically, this work shows that automatic localization and segmentation algorithms achieve high (above 90%) accuracy in locating and segmenting small objects of various categories, however, proper label assignment only achieves 58% precision and 65% recall.

As the results presented in related work indicate, the accuracy decreases as number of categories increases and the differences between employed classes of interest become more intricate. This means that if such algorithms were employed to produce high quality annotations, in practice the annotator would be required to sift through the predictions to find and fix the mistakes. Together with providing a training set, this results in the user having to eventually go through the entire data set and verify

Class	# in Truth Area	Location # Found	Location (%)	Segmentation Pr	Segmentation Re	Recognition # in Test Area	Recognition # Predicted	Recognition Pr	Recognition Re
Short Post	338	328	(97)	92	99	116	131	79	91
Car	238	179	(75)	92	77	112	218	50	62
Lamp Post	146	146	(100)	89	98	98	132	70	86
Sign	96	96	(100)	83	100	60	71	58	65
Light Standard	58	57	(98)	91	92	37	51	45	62
Traffic Light	42	39	(93)	84	86	36	33	52	47
Newspaper Box	37	34	(92)	38	93	29	14	0	0
Tall Post	34	33	(97)	58	96	10	6	67	40
Fire Hydrant	20	17	(85)	88	100	14	10	30	21
Trash Can	19	18	(95)	60	100	15	14	57	40
Parking Meters	10	9	(90)	100	100	0	4	0	0
Traffic Control Box	7	7	(100)	80	100	5	0	0	0
Recycle Bins	7	7	(100)	92	100	3	1	0	0
Advertising Cylinder	6	6	(100)	96	100	3	0	0	0
Mailing Box	3	3	(100)	98	100	1	2	0	0
“A” - frame	2	2	(100)	86	100	0	0	0	0
All	1063	976	(92)	86	93	539	687	58	65

Table 2.1: Per-class results of the stages of the algorithm presented in [46]. Shown for each class are: the number of objects in the truth area, number and percent of truth objects found by the localization algorithm, precision and recall of segmentation, number of objects in the test area, the number of predictions made by the recognition algorithm, and the precision and recall rates of recognition.

annotation for each object manually. Furthermore, provision of the training set is a challenging task on its own for a user of such system. What objects and how many of them for each category need to be provided to ensure high prediction results? As Table 2.1 shows, several categories (Parking meters and A-frames) have so few instances in the data set, that providing them to the training set causes more problems than it helps. Firstly, an annotator needs to search through the data to find those few instances. Secondly, providing these instances only results in further confusion of the classifier and appearance of false positives among predictions in the test set. This steals from the accuracy of the categories that mispredicted objects actually belonged to. These observations suggest that, as valuable as it is to improve classification algorithms, until they cannot reliably provide desired accuracy, focusing on a user interface that allows to efficiently provide/fix annotations is mandated.

2.2.2 User-guided interfaces

A number of works that instead take an interactive labeling approach put the user in the driver's seat, allowing him or her to decide how to proceed at training the machine. In one of the early works in this direction a system for training a pixel classifier called Crayons was introduced [39]. The system uses an incremental version of a decision tree and allows users to sequentially correct the mistakes the most current state of the classifier makes. One of the observations made from the user study showed that the quality of the classifier in users' experience starts to play an important role only once the interactive speeds of updating the classifier are achieved. In [105] a similar approach was taken to allow the users to interactively correct errors made by a handwriting recognition system as they wrote. In [12, 55] the authors describe interactive systems where users iteratively train object classifiers in images by actively demonstrating objects to the system and providing more examples when the classifier appeared confused.

Interactive user-guided training is used in search applications employing what is known in data mining as *relevance feedback* [96]. For example, CueFlik described in [40] allows a user to interactively train an image search engine to rank images in terms of a user-defined concept. This work has been further extended by exploring different ways of presenting users with ranked results in [2] and the effects of the ability to see consequences of different modifications before committing to one [3].

Interactive methods have also been used for reconstruction and segmentation of 3D point clouds. For example, [56] describes a system that reconstructs 3D structure from video and then allows user to interactively agglomerate points into primitives, create repetitive patters and extend these patterns while fitting models to the data. A similar approach of fitting primitives to data while enforcing relationships between them is used for rapid reconstruction of architecture from LiDAR scans in [81]. In [104] an automatic segmentation is followed by an phase where a user can interactively

correct the automatically generated segmentation of objects directly in the RGBD data, with the system then populating the scene with matching shapes from a 3D repository.

In summary, in these systems the cost of human interaction is paid-off by higher accuracy of the execution of the task. A user can see where the machine makes a mistake and correct its model and thus improve its predictive power.

2.2.3 Active interfaces

Works in psychology [99, 10] suggest that minimizing interruptions and decreasing the number of options significantly decrease anxiety and increase efficiency in humans. Interruptions in annotation, among other things, consists of switching between labeling, next sample search, navigation, and selection tasks. To address this issue, many researchers explored active machine learning methods that delegate the task of sample selection onto a machine.

Active learning [101] is an approach in machine learning where the system uses some utility function [42] to select samples from a pool. This function should be such that it can guide the system to select objects in a manner that, upon user providing a label for it, would improve system's model and the quality of the predictions about unannotated data. This approach has been widely used for various annotation and retrieval tasks. For example, [126, 138] use active learning to select frames to be annotated in video, [37, 63, 66, 113, 120] - to select images to train a classifier, and [29, 70] to train the system to separate noise in keyword search results for an image dataset construction. In some cases, additional information, such as mutual location of categories, is relevant. Work in [106] proposes to not only to query the object's category but also its contextual relations with the surrounding objects in the image. When categories are hard to learn due to subtleness of the distinctions or reliance on very specialized knowledge, several works ([16, 36, 87]) employed active learning to

acquire the relevant attributes. These systems improve on the amount of questions a user needs to be asked to produce a good classifier, but disregard that not all annotations are equally expensive to acquire.

A number of active learning methods explicitly incorporate the notion of cost of annotation into their framework. [141] use a cost of annotating a frame in a video, which is a linear function of estimated false positives and false negatives in each frame, and ask the annotator to fix the most expensive one. The approach described in [49] optimizes medical diagnostic by making the machine actively decide which tests need to be performed, given a budget, to produce the most reliable diagnosis. To model cost in text annotation, [11] use the number of parsing discriminants in a sentence; and [52] derive a linear cost model as a function of number of tokens and number of words needing corrections by fitting it to the actual observations of human annotators. [100] examines annotation costs for four tasks and evaluates how they differ by annotators, whether they are predictable, and if incorporating predicted cost into an active learning utility function improves training the classifier. Several papers use a value of information (VOI) [69] framework that chooses the samples by balancing the risk of mislabeling a sample with the cost of acquiring the annotation. For example, [67] uses the length of a voicemail in time as the cost of annotating it. The cost of labeling a single object in an image in [122] is linear in the number of objects present in it. This work is further extended in [119] where machine learning is used to learn the actual cost model of annotations, in [121], where at each iteration the system selects a set of samples without exceeding a given budget, and in [123], where the authors summarize their findings into a single framework and provide more results showing robustness of their approach. Work in [64] replaces the usual interface where a label has to be found in a menu with an interface that at each iteration demonstrates a pair of images instead. The label of one of the images is known and the other is an unknown label, and the user is required to say if they belong to the same category. It

also uses a VOI framework evaluating the cost as a scaled expected number of binary questions that need to be asked until a match answer is received, showing that asking several binary recognition questions from a human is faster than making them sift through a large menu searching for a label.

2.3 Perception and interaction with groups

Human vision and their ability to efficiently comprehend what they see in the enormous stream of incoming visual information is what allows them to be good annotators. It is thus no surprise that models of human visual attention have been introduced and successfully used in computer vision, mobile robotics, and cognitive systems [14].

The human visual system is capable of rapidly grasping the *gist* of images after exposure of as little 100ms [97, 92]. Furthermore, humans can answer some questions about the contents of the images after very short exposure [111, 31], even when they are distracted with other tasks [74].

Other studies show that humans exhibit ability to rapidly [25] and robustly [26, 51] capture objects in sets of similar items[8], using a mechanism that captures a *summary statistic* of the observed set instead of independently processing each object. Such ensemble representation includes not only appearance but spatial patterns, helping with outlier detection, guiding focus of attention, and facilitating perception of crowded regions [1].

In 3D point clouds, especially, in man-made environments, like cities, where most small objects exist, patterns and recognizable groups exist in abundance (e.g., parking lots of cars or street lights along roads). Since every time a user is presented with a scene, their visual system already does work on recognizing and evaluating the gist

of the scene and the groups, an interface that allows a user to efficiently convey as much of the result of this processing as possible back to the machine is desirable.

One avenue that has been well studied in machine learning community is multiple-instance learning (*MIL*) [4, 144, 140]. The core difference from other machine learning approaches is that instead of receiving a set of labeled instances for training, the learner receives labels for "*bags*". Each bag contains one or more instances, and having a label assigned to it only means that some instances in that bag are of that category. First introduced in [34] in application to musk odor prediction, it found natural applications in image classification using both active [76, 103] and non-active machine learning approaches [18, 77, 118, 139, 143, 145]. Simply saying, for example, that an image contains cars is simpler than pointing to every car in it. The drawback, however, is that in bags that contain more than one object the exact labels of each object are predicted. This brings up same issues as with other existing methods: if predictions are of less than desired quality, how to efficiently locate and fix incorrect instances?

Another approach is to present users with groups of instances and ask them to manually select examples that can be labeled as positive or negative as done in [40, 2]; or manually verify that objects in the group indeed fit the label as in [32]. These methods produce confirmed annotations for objects just as we would like. However, interaction of manually verifying or removing instances one-by-one presents same shortcomings as other manual annotations. It requires a lot of per-instance manual interactions, which is costly.

Chapter 3

User-guided interactive annotation

Understanding real world environments is a natural function of human perception. Hence, an immediate intuition towards streamlining annotation is to allow the annotator to completely control the process. Existing work in annotation of 3D scans [46, 88, 116] focused on algorithmic parts of the system only mentioning which objects from an existing annotation are used for training, and which ones—for testing. From the perspective of an interface these works usually do not impose any constraints. A batch interface only requires tools to label objects and, once enough training annotations have been made, to request a classifier to be trained and used to predict labels for the unlabeled objects.

Practical application of such interface for purposes other than academic evaluations, however, contains a number of shortcomings. Given an interface like that and a new data set, an annotator should be able to estimate which ones and how many examples of each category the employed classifier requires to provide good predictions without any hints from the system. Additionally, in a practical setting, even once the necessary training set has been acquired and the predictions for the unlabeled objects have been made, the user still needs to sift through the predictions to find and fix the mistakes that the classifier made to correct them. Since it is the case that to ensure

accurate labeling of a data set the user needs to go through every object, then the separation of this process into two stages, with the first batch of labels provided with no relevant feedback, is practically artificial.

To address this problem, this chapter introduces an interactive user-guided(UG) interface. It updates the classifier and propagates predictions after every new label a user submits. This interface yields several advantages. Seeing the current state of the annotation enables users to choose where to focus their effort. When a prediction is correct, they can simply confirm it with a uniform keystroke rather than searching for a label in a menu every time. Constant label updates also allow additional label-based filtering, which in turn enables single motion selection of several objects. Since prediction updates happen constantly, these tools can be employed during the entire annotation process, not only after a single prediction request as a batch learning interface suggests.

3.1 Interface

A user-guided interface for object annotations in 3D point clouds is built to meet the following desired properties. Firstly, it needs to display the point cloud data in a way that objects that need annotation are visible and identifiable, as well as the information about label predictions is visible and useful. Secondly, users should be able to navigate around the data set and select objects that they desire to annotate. Thirdly, the interface should provide means for efficiently assigning proper labels and confirming correct predictions to the selected objects.

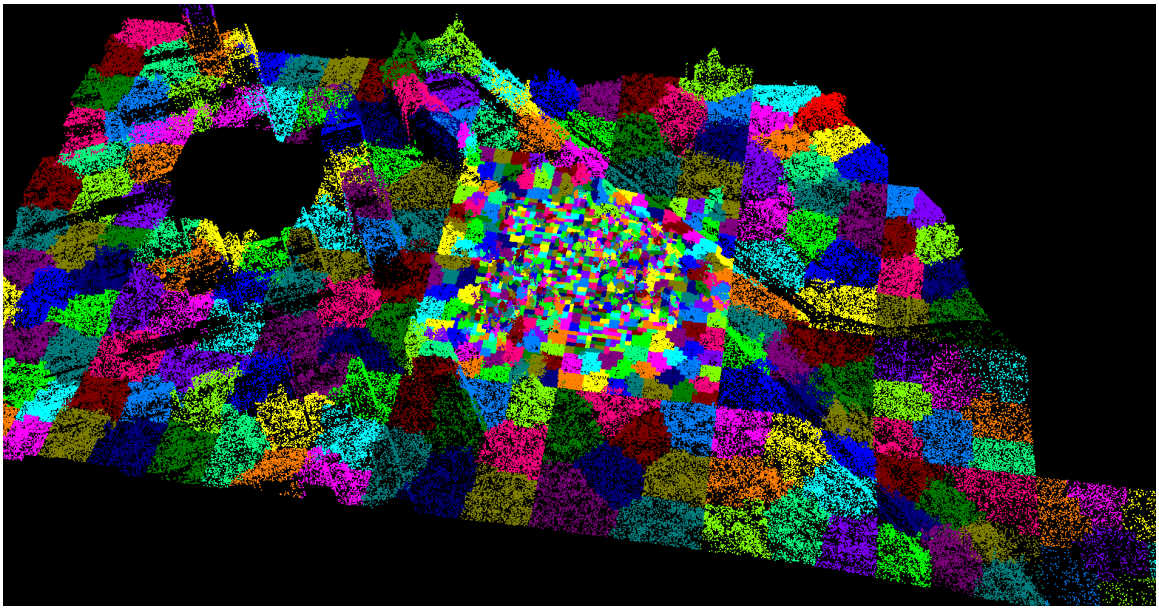


Figure 3.1: Level of detail representation of the displayed 3D point cloud. Each block is shown in a different color. Larger blocks store downsampled point clouds. Closer to the center blocks are smallest and the resolution of points is highest available.

Visualization of the point clouds is the first important step. It is known that context helps humans understand the objects in a given scene [85]. Hence showing as much of the context, i.e. points that surround the objects of interest, is important. Existing point clouds of natural environments that exhibit enough resolution to sample smaller objects well require a lot of memory. This poses a challenge of balancing the extent of the context with the machine's memory budget. To address this ques-

tion interfaces presented in this work take the approach of the level of details (LOD) visualization. Points are loaded at the highest density closer to the "center" (user's area of interest) of the scene. Resolution decreases with distance from the center. This is done by storing points in a multiresolution block hierarchy with decreasing sampling closer to the top of the hierarchy. The further a given block is from the center of the scene, the higher in the hierarchy the block is loaded. An illustration of this mechanism is shown in Figure 3.1.

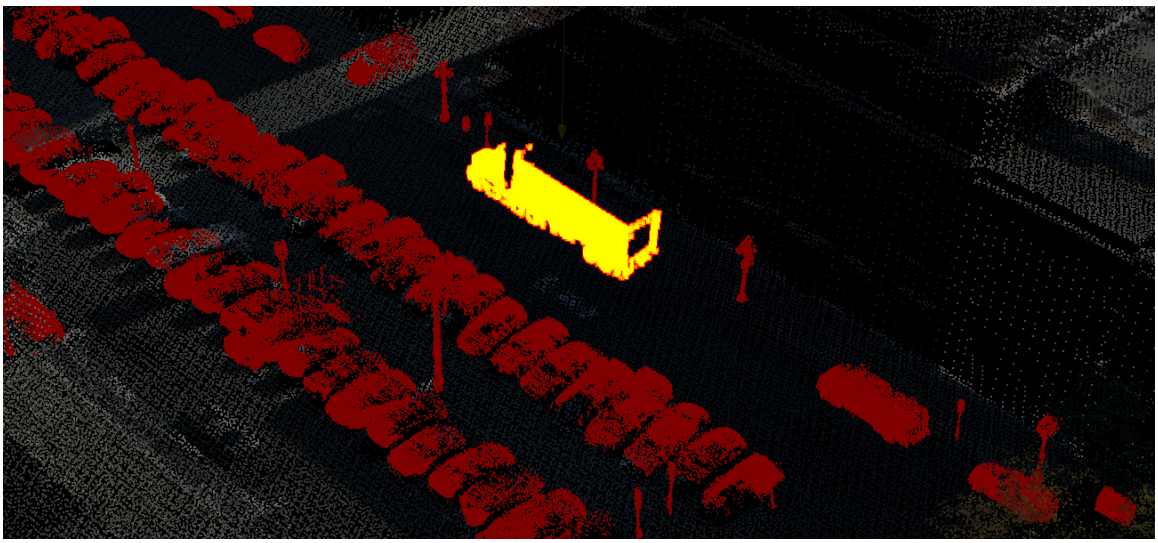


Figure 3.2: Selected object is highlighted in yellow. Other selectable objects are shown in their assigned label color - red stands for "Unknown", not yet assigned a label. The rest of the points are shown in original color and are shown for context and are not selectable.

A user can explore the scene by rotating, panning and zooming using a mouse. A user can select any object that is available for labeling with a mouse as well (Figure 3.2). To provide most relevant visualization to the user, the center of the scene (which is not necessarily the center of the screen) follows the user's selection. Since not all points belong to objects that are relevant for annotation, the interface visually separates objects from the rest of the cloud - background points. Selectable objects are shown in the color of the assigned labels and the user can optionally request the interface to show the assigned labels in plain text hovering over the objects as shown

in Figure 3.3. Background points are available to provide context and are presented in their original color. They can be concealed upon a user request to decrease the clutter when necessary.

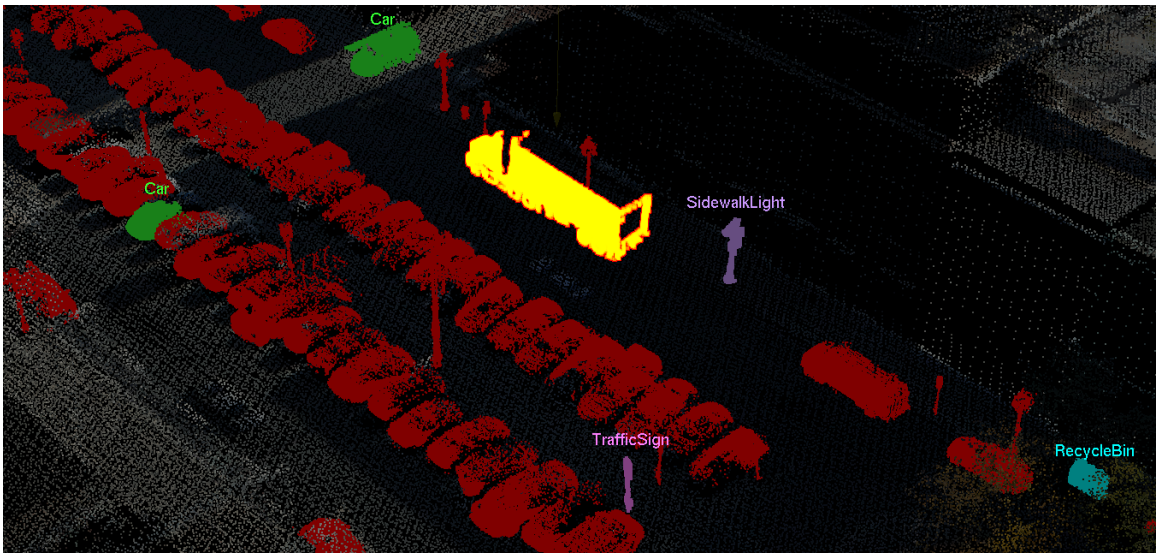


Figure 3.3: Selected object is highlighted in yellow. Other labeled objects are shown in the color of the label assigned to it. A user can request label superscription.

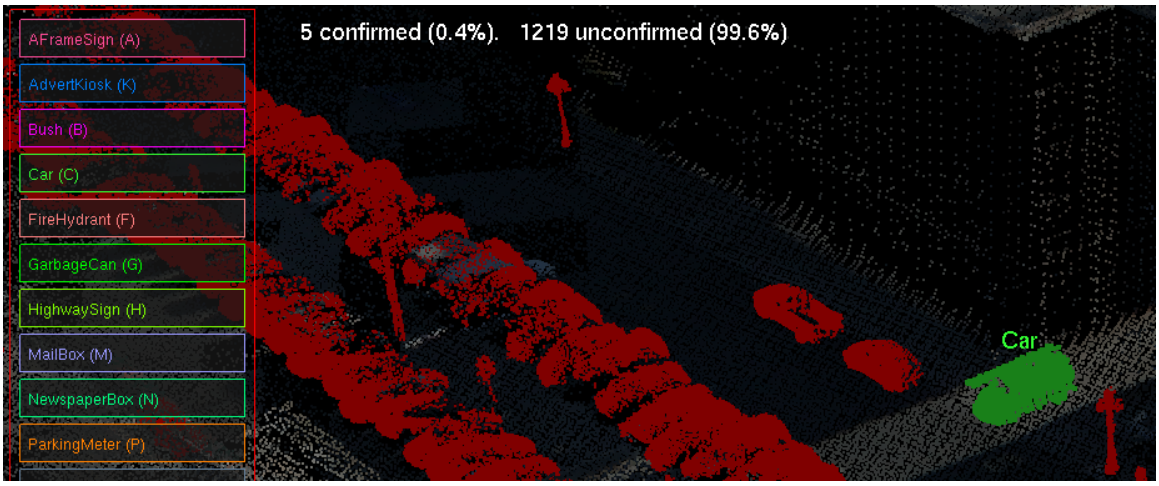


Figure 3.4: User-guided(UG) annotation interface. The labeling menu on the left consists of buttons with label names, keyboard shortcuts and a color reference. High level annotation progress numbers are also provided to the user top right of the menu.

Labels are offered in a translucent menu on the left (Figure 3.4). The menu constitutes a set of clickable buttons colored according to the label. This allows the

menu to also serve as a persistent legend for color coding reference. The buttons contain not only the name of the label but a keyboard shortcut. A label can be assigned in two ways. If any objects are selected, clicking on a button or using a respective shortcut assigns that label to the selected objects. If nothing is selected but a mouse cursor is pointing at an object and a shortcut key is used that maps to a label, that label is assigned to an object that the cursor is pointing at. If a label has been assigned to an object by a user this label is considered to be *confirmed* and is only used by the machine for training - its label is not changed by predictions.

Figure 3.5: Machine-predicted label for a selected unconfirmed object is hovering above the center of the screen and its button is expanded in the label menu.

To facilitate confirmations of correct predictions without requiring the user to search through the menu two enhancements are available. To visually highlight the prediction, the label is shown hovering over the object and its button is expanded in the menu (Figure 3.5) to facilitate the search. To further simplify interaction, a unique confirmation key is available to the user with the sole function of confirming that predicted label is correct for selected objects.

To avoid spending time on evaluating objects that they already labeled, users can request them to be hidden from view and cannot be selected. However, sometimes it may be valuable to see previous labeling decisions e.g., to maintain consistency in confusing cases. If the confirmed objects are visible, the user-provided and machine-

predicted objects and their respective superscribed labels are visually separated by showing the former in bolder style (Figure 3.6).

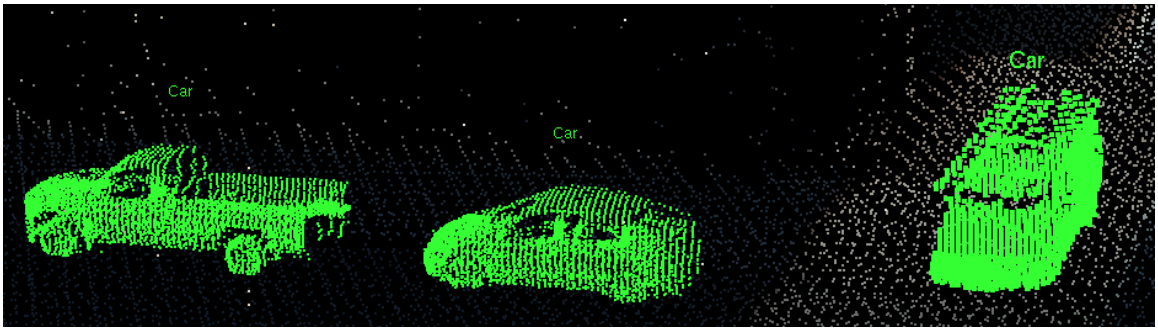


Figure 3.6: Objects with user-assigned labels and their label name superscripts are shown in bolder style than those with a machine-predicted label.

The machine incorporating every new annotation and immediately updating the predictions for the entire data from the very first labeling leads to both an opportunity and a challenge. The opportunity is that the user can see the current state of annotation and make an informed decision about what objects are still confusing to the classifier and fix them. Existence of larger categories of objects that are similar to each other in shape can thus lead to large leaps in accuracies only after a few instances of these categories are introduced. Figure 3.7 demonstrates how only after a few annotations several groups of objects are visually recognizable together. They are visibly separated into different categories by the classifier, even though examples of some classes have not even been introduced yet. The challenge is that every time another annotation is provided all unlabeled objects can potentially change their predictions. If an outlier influences a large subset of the data, this can result in losing accuracy.

Since humans rather efficiently perceive scenes as a whole and employ ensemble statistics representation, groups of similar objects are expected to be noticeable to the annotator (e.g., cars in Figure 3.7). An ability to easily select the entire group would allow labeling multiple objects with a single label assignments. This can considerably diminish the effort required to annotate the entire data set. It is hence desirable to

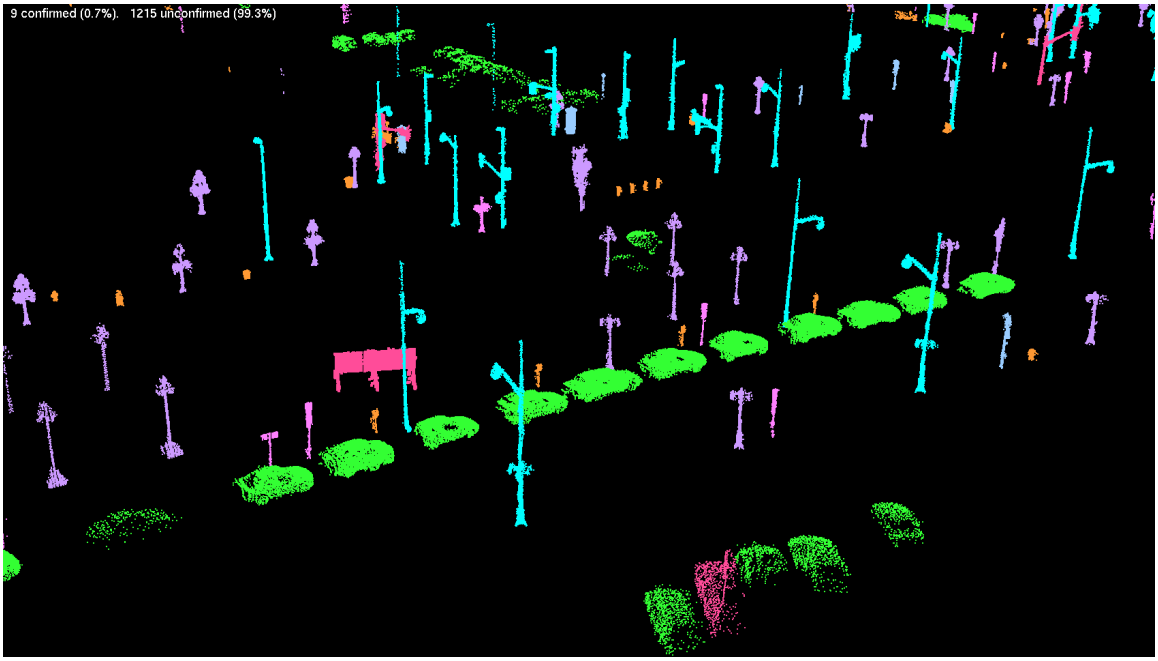


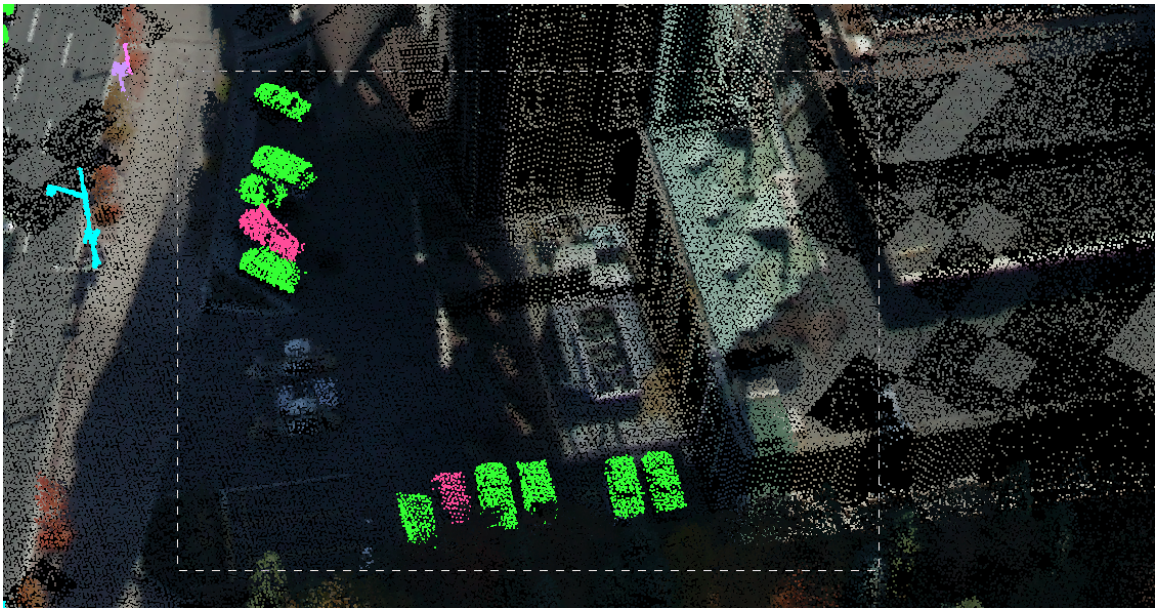
Figure 3.7: User-guided interface interactively updates predictions. After only 9 annotations groups of objects that share same label are visible, such as cars, street and sidewalk lights. Background points are concealed for image clarity.

introduce tools that facilitate selection of such groups in as little interactions and effort as possible. Such tools of group selection are also useful for confirming correct predictions. Confirmation of labels for multiple objects prevents future flipping of correct predictions for these objects in the future, since the machine is not allowed to change their labels.

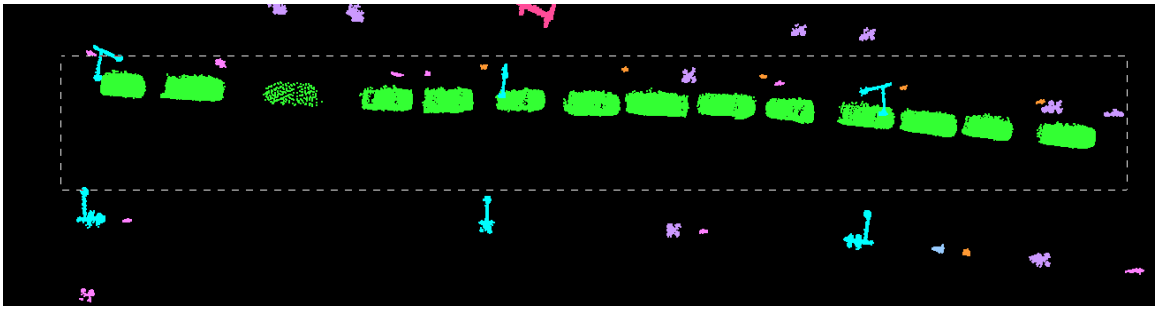
There are several options of selecting a group on a 2D screen. The most straightforward one that UG interface implements is to allow the user to click on each object in a desired group. Depending on the used keyboard modifier this will add, remove or replace contents of the current selection. This, however, requires a linear number of interactions in the size of the group and additional effort of carefully moving the cursor between objects and aiming at them.

Regional selection tools, such as lasso, are also available. However, the lasso tool requires even more cursor precision than simply clicking. Another option is a region selection by dragging a rectangular frame that envelopes all of the objects that fall

into the frame. Once again, depending on the keyboard modifiers it adds, removes or replaces the current selection (Figure 3.8a). This option has the advantage of simplicity - it requires only a single drag motion per group. The disadvantage is that it is often impossible to find a rectangle that envelopes only the desired objects, yet leaving the undesired outside (Figure 3.8b). It is an option, of course, to simply select the entire group in several selections of smaller subgroups. However, the smaller these subgroups are the less the advantage of frame selection over clicking in each instance.



(a) Selecting a group of cars to label them together.



(b) Selecting a group of cars with a rectangle will include outliers.

Figure 3.8: Rectangle region selection tool is represented by a dashed frame.

When the goal is to provide or confirm a label for multiple objects in a single action, an undesirable object in the selection is an object that is dissimilar from the

rest of the objects in a way that is relevant to the annotator. The machine learning employed in an annotation interface is expected to be able to capture such dissimilarities. Under this assumption, to avoid selecting undesirable objects is to prevent the interface from allowing the selection of objects that the classifier understands to be different, specifically, the ones that it predicts to be in different categories.

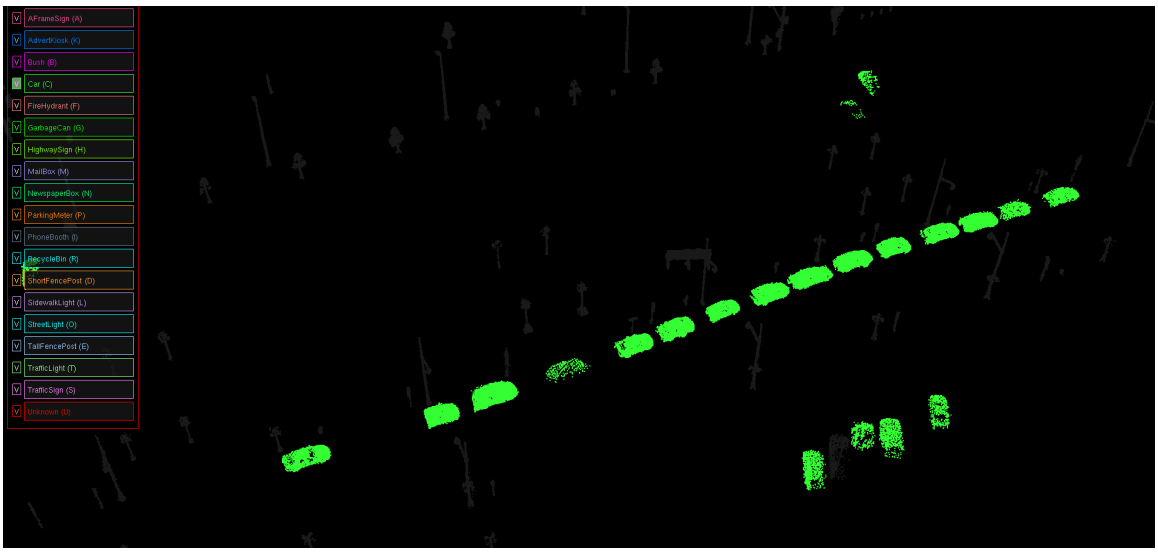
To take advantage of the simplicity of the rectangular region selection while keeping undesired objects from being selected, this interface introduces visibility filters. Every label is mapped to a user-controlled flag that specifies if objects that are assigned that label are shown and selectable. The control over the visibility flags is executed in several ways. Direct control is available via visibility toggles on each button in the label menu (Figure 3.9a).

(a) (b)

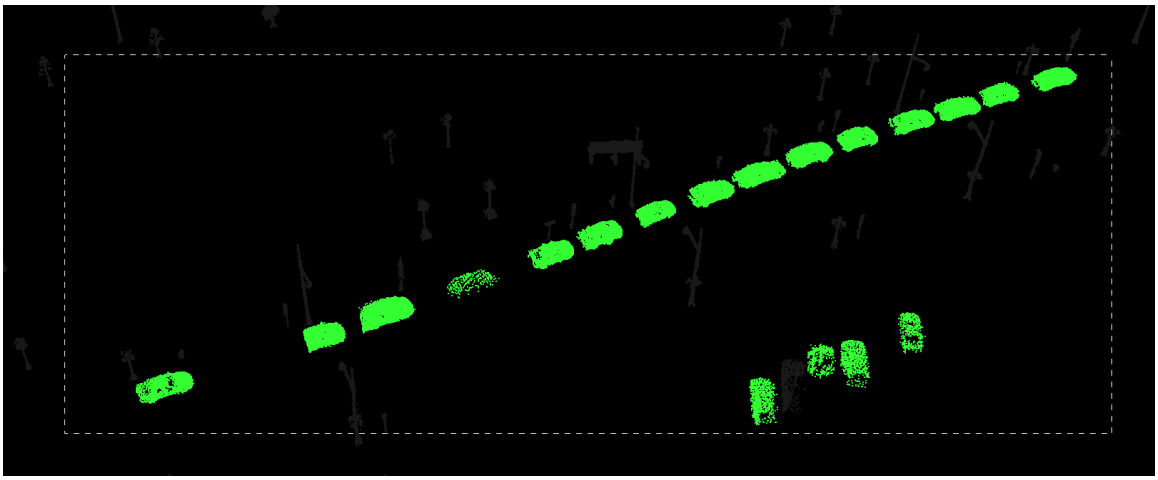
Figure 3.9: Visibility is controlled either by (a) V(-isibility) toggles in the label menu or (b) through exclusive visibility with label chosen through the user's selection of object.

When selecting a group for a single label it is desirable to have all labels besides the one in question be invisible. However, switching all of the labels manually to achieve that is an unnecessarily lengthy effort to request. Hence, an *exclusive visibility* mode is also available, which turns off all labels but the desired one (Figure 3.10a). Engaging exclusive visibility allows for a single motion group selection without catching outliers (Figure 3.10b). It is engaged using the same visibility toggles (with a keyboard modifier) on the respective button in the label menu. Additionally, if objects of a

single category are selected, a keyboard shortcut allows users to enable exclusive visibility for that category. This is useful when the selection is initiated by a single seed and a larger group is noticed in the vicinity (Figure 3.9b).



(a) Exclusive visibility enabled on the "car" label.



(b) Selecting a group of cars from Figure 3.8b with a rectangle will now only contain cars.

Figure 3.10: Rectangle region selection tool represented by a dragging motion frame.

It is expected that the classifier will not always be able to make reliable distinctions that will make visibility filtering universally applicable. The UG interface only provides the tools for the user to leverage it, when possible. Users are also expected to make mistakes and hence undo and redo commands are available.

3.2 Machine learning

Now that the functionality of the interface is described, it is important to identify the machine learning (*ML*) framework that should serve the purpose of repredictions at interactive rates. In machine learning, objects that require separating into different categories are represented as points (samples) in a feature space. Coordinates of each object are defined by its features - an n-dimensional vector of evaluations of descriptor functions that represent this object. Relationships between objects are defined by the relationships between their respective points. *Consistency* is often a desirable property of the feature space, meaning that features and the distances are chosen so that proximity of points in the feature space reflects the similarity of the respective objects according to a desired property e.g., label, shape or color. Machine learning algorithms are then used to extract relevant relationships between objects through evaluations in the feature space. For example, unsupervised learning algorithms search for hidden structure in the distribution of objects, allowing, for example, clustering of the data. Conversely, supervised learning algorithms take additional input, such as labels of a subset of samples, and use feature space relationships to predict labels for other objects.

Using different classifiers for different interfaces due to their better applicability for specific setting is possible. However, if this is the case, it may be hard to separate advantages granted by the interface and interaction models from those granted by a machine learning algorithm being better suited for the data. Thus, to provide a more accurate comparison of the interfaces in this work, a common machine learning framework (classification algorithm, descriptor features, and feature space distance metrics) is chosen.

To select a suitable ML algorithm, considerations of the requirements of both a batch mode annotation and the UG interfaces should be made. By virtue of many supervised algorithms initially designed and demonstrated to work in a batch mode,

Classifier	Precision # Predicted	Correct	(%)	Recall Correct	(%)
NN1	707	379	(54)	344	(64)
NN5	582	374	(64)	342	(63)
Random Forest	368	288	(78)	270	(50)
SVM	687	400	(58)	351	(65)

Table 3.1: Effects of classifiers on recognition rates from [46].

Batch interface can employ any of them. [46] evaluates several popular machine learning algorithms with application to labeling small objects in LiDAR scans on a data set similar to the one used in this work. Their results are summarized in Table 3.1. The authors use SVM for the final results, motivating it by displaying the highest recall, however, all presented algorithms are essentially interchangeable offering only marginal differences, trading off precision for recall.

The UG interface requires a classifier to incorporate new information incrementally. As important as accuracy of the classifier is for obtaining good annotation, [39] shows that it is more important to the users of an interactive system that the prediction updates come without delay. Moreover, according to the findings in [2] it is beneficial to provide users with an ability to preview a result of their actions before committing to them, or simply put—an ability to undo actions. This requires the algorithm to both support rapid updates after a new training sample is provided as well as when a sample is revoked. A transductive learning algorithm [22, 43], which doesn't require an additional step of creating generalization rules from training examples but rather makes predictions directly from these observations, is thus more suitable. The simplest algorithms among the ones in Table 3.1—K nearest neighbor(kNN) [30]—meets these requirements and NN1 is chosen as the classification algorithm for all interfaces presented in this work.

The same considerations of prioritizing simplicity are made in selecting of descriptor features. A number of shape descriptors have been used as features for identifying objects in point clouds, such as spin images [62], splash features [59], shape

Feature	Precision # Predicted	Correct	(%)	Recall Correct	(%)
Shape Features	568	313	(58)	298	(55)
+ Multiple Segs	591	336	(59)	314	(59)
+ Context	586	360	(64)	327	(61)

Table 3.2: Effect of features on recognition rates.

histograms [7], and a variety of coarser geometric descriptors, e.g. [58]. Sometimes some of them are used together to form an expanded feature vector or in tandem with higher level structures, e.g. constellation models [88]. Often, these works evaluate features that they offer together with a specific algorithm and evaluate using a specific, often unique to the given work, point cloud. A comparison of some of these features without variations of other factors is done in [45]. The authors incrementally add more shape descriptors to a feature vector and evaluate how much it improves the accuracy. Specifically, they start with simple geometric features, such as volume and point distribution along principal axes, then spin images are added. Further, the above features are computed for a set of clouds extracted using several segmentation algorithms and concatenated together. Finally, contextual features that describe relative locations of objects of given category with respect to other objects are added. Evaluation results are summarized in Table 3.2. The features are added in order of increasing complexity of their computation: point distributions for a single set of points is easier to compute than for multiple segmentations, and contextual features require iterating over predictions until convergence. However, this added complexity once again only yields marginal benefit. Since in this work machine learning is only an aid while the convenience of the annotation interface is the priority, this work opts for using a simple 6-dimensional feature vector. Specifically, the features used are: (1-3) the 5% trimmed spread, median, and median absolute deviation of Z-coordinates (Z is up), (4) the 5% trimmed maximum distance of a point from the centroid in

the XY plane, and (5-6) the variances of points distribution in the two principle axis directions in the XY-plane.

The last matter that needs definition is the structure of the feature space. A NN1 algorithm applied in a space with Euclidean distances, in essence, splits the space into Voronoi cells [9] centered at training instances. All unlabeled samples get assigned a label of the centroid of the cell they are in. This means that only the distribution of the labeled instances defines the classifiers predictions. Semi-supervised learning [21], which incorporates information about distributions of not only the labeled samples but the unlabeled data as well, has been shown to improve prediction accuracy [22]. Warping space to reflect structure intrinsic to the data allows us to directly apply supervised learning algorithms in a semi-supervised fashion [108]. Density-based clustering [71] algorithms, such as OPTICS[6] and DBSCAN[38], capture structure of the data based on the observation that samples found in the sets of same densities are more likely to be similar. To leverage the density information in a semi-supervised setting with a supervised learning algorithm the feature space in this work is defined by density-sensitive distances.

In this work, the system first computes density-sensitive distances between all samples in the data using the formulations similar to those in [23, 146]. The main intention of this computation is to squeeze together points in dense regions and distance them in low density areas. To do this a K nearest neighbors graph $G_{KNN} = (V, E)$ is defined over the data points in the feature space with edges weighted by the pairwise distances between respective points $d(v_i, v_j)$. The value of K in this work is the smallest value that allows G_{KNN} to be connected. Two points are considered to be similar if they are connected by a chain of points $P = p_1, ..., p_l$ along the edges of G_{KNN} that are in a dense region. If P_{ij} is the set of all paths in G_{KNN} that connects data points v_i and v_j then the density-sensitive distance between these points is defined as

$$D(v_i, v_j) = \frac{1}{\rho} \ln \left(1 + \min_{P \in P_{ij}} \sum_{k=1}^{l-1} \left(e^{\rho d(p_k, p_k+1)} - 1 \right) \right) \tag{3.1}$$

Intuitively, this expression results in increasing contributions of the longest edges on all the paths between two points, thus characterizing each of the paths by the sparsest region it traverses. Choosing the smallest of these characterizes the path that goes through the densest region available. Hence $D(v_i, v_j)$ between two points reflects whether these two points belong to a single cluster of similar density or if they are separated by sparser regions. Parameter ρ determines how soft this effect is: if $\rho \to 0$ $D(v_i, v_j)$ simply becomes the length of the shortest path and if $\rho \to \infty$ it becomes the length of the shortest among all paths longest edge.

Using this definition of distance in the feature space, a simple NN1 classifier becomes more respectful of the intrinsic data structure without sacrificing its simplicity.

3.3 User study

3.3.1 Batch interface

To evaluate the efficiency of the UG interface against the task of annotating objects in 3D point clouds, it needs to be compared to an interface used in previous works. The closest related work, however, does not address the design of the interface. Hence in this work a straw man *Batch* interface is developed in a manner that it meets the operational mode and necessary functionality implied from the related works. Specifically, it needs to display point clouds, and allow a user to select objects and label them. Once the training set is collected, the user needs to request the training of a model and reprediction of labels to the rest of the objects.

The difference between the UG interface and the straw man Batch interface consists of the following. Firstly, the classifier does not update predictions after each label provision. Instead, the user that assembled a sufficient training set should request a reprediction. At that point, all provided annotations are incorporated into the model and predictions for the unlabeled objects are produced. For the purpose of accurate simulation of a batch learning setting, this operation may only occur once and cannot be undone. Second, the frame selection and visibility filtering tools are disabled, since they are not an integral part of the batch learning setting in related works.

All other features and the machine learning algorithm are same for both the UG and Batch interfaces. This set of tools covers the requirements for a Batch learning annotation interface implicitly suggested in the related work. To allow for comparison of the entire session, it also provides additional tools to complete the annotation process after training set collection and label predictions.

3.3.2 Experimental set-up

Evaluation and comparison of the Batch and UG interfaces is done against the goal of this work - efficient and accurate annotation of small objects in point clouds. Both interfaces are designed to completely rely on a human user in driving the annotation process: deciding what objects and when to label or confirm. Hence, to compare fitness and advantages of either of them a human user study has been conducted.

For evaluation, a point cloud captured with 4 terrestrial (car-mounted) and 1 aerial (airplane-mounted) LiDAR scanners within a 6 km^2 section of downtown Ottawa, Canada [82] was used (Figure 1.1). The data set contains approximately 1 billion points. Each point is represented by a 3D position, RGB color, and scalar intensity. The colors and intensities of terrestrial points are very noisy and thus were not used. The data set was chosen because it is has been used for evaluation by previous systems for recognition of small objects in LiDAR scans of cities. In particular, it was first used in [46], which reported approximately 58% precision and 65% recall using fully automatic algorithms to recognize 1,063 objects in 17 semantic categories in a $0.3km^2$ *evaluation area*. The tests in this work are conducted on a similar set of 18 object categories (bush, fire hydrant, mailbox, newspaper box, parking meter, advertising kiosk, garbage can, recycle bin, phone booth, traffic sign, highway sign, A-frame sign, sidewalk light, street light, traffic light, short fence post, tall fence post, and car) in the same evaluation area of Ottawa, using a slightly expanded *ground-truth set* of 1,224 objects (Figure 3.11). Additionally, a *training area* has been allocated in a different part of the point cloud with 163 objects in same categories. It is used to train inexperienced users to work with the interface and point cloud data prior to an actual annotation session.

To acquire objects in this ground truth set the LiDAR points were first segmented into objects using the algorithms described in [45, 46]. Specifically, plane extraction algorithms were used to remove the points associated with major planar structures in

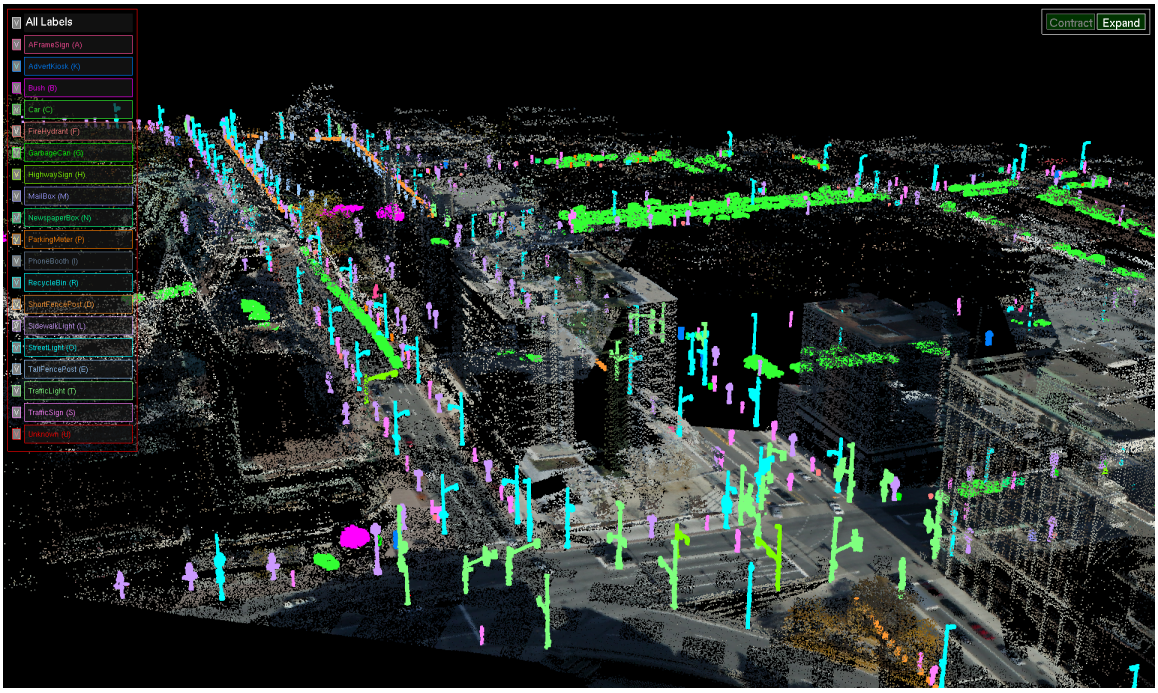


Figure 3.11: Test data set used in this work is a point cloud collected with LiDAR scanners in Ottawa. This image highlights the 1,224 small objects that users were asked to label in the experiments. Objects are colored according to their ground truth labels.

the environment (ground and buildings), and then hierarchical clustering and graph cut algorithms were used to detect the locations of potential objects and cluster points into objects. The results of these automatic segmentations were improved interactively with a simple tool that allows a user to specify points inside and/or outside any object and then resolves the object segmentation boundary with a graph cut. These improved segmentations were used in the tests – the user only had to provide a label for each object.

The choice of creating segmentations as a pre-processing step is motivated by several reasons. First, most small objects in a LiDAR scan of a city can be segmented automatically with high accuracy ($\sim 90\%$ in [46]), and so the segmentations used in the evaluations are quite representative of those created with state-of-the-art algorithms. Second, segmentations can often be refined automatically after objects have been labeled – e.g., using an algorithm that aligns objects of the same type to form

a consensus of the correct appearance to improve segmentations [68]. So, precise pre-segmentations may not be necessary. Of course, there are complicated interplays between the accuracy of the segmentations with the effectiveness of the shape-based classifiers, the robustness of the consensus algorithms, the ability of people to recognize objects, the quality of the data, and so on. These investigations, however, are outside the scope of this work, which focuses on improving a labeling interface. Finally, the interfaces presented in this work are not specific to labeling LiDAR data. They rather address general ideas for labeling any kind of data that can be shown in naturally occurring environments (e.g., cells in stained microscopic images or tumors in medical images). The segmentation challenges for each of those types of data is different. Hence, using pre-segmented objects avoids mixing evaluations of the annotation interface metaphors with the specific challenges of segmenting a particular data sets.

A group of 8 users has been recruited for the evaluation. Participants were randomly split into two sets of four users. One set of participants were assigned the Batch interface, and the other the UG one. The order of assigning participants to one of these two conditions was random, and no participant was aware of which condition was the control. Their task was to label all objects in the ground-truth set using their assigned interface. Each group ended up consisting of 3 males and 1 female between the ages of 20 and 40.

Each participant was given written instructions describing the goal of the experiment together with tools and commands relevant to their interface. They were further presented with definitions of the labels used in the experiment as well as photographic and point cloud appearance examples of the types of objects to be labeled. Both the instructions and the label references were available to the users throughout the experiments. After getting familiar with the descriptions, the users were asked to practice using their respective interface by executing a short interactive tutorial. It

consisted of annotating 163 objects in the training area with the interface they had been assigned with. However, in this stage the interfaces provided users with immediate feedback if any label was assigned incorrectly and only accepted the correct labels for every object.

Once the tutorial was completed, the participant proceeded to provide or confirm the label for all 1,224 objects in the evaluation area without any feedback or guidance other than instruction clarifying questions. Users of the Batch interface were requested to first produce annotations for a sufficient (according to their understanding) training set, then request reprediction, followed by confirming or correcting predicted labels for the rest of the objects. Once done labeling all objects, users completed an exit questionnaire.

User sessions have been recorded in two ways. First, all commands that were issued by users, such as selections, label assignments, confirmations and visibility changes, were recorded along with time records of those commands and information on changes in the selection sets and labels assignments. Secondly, periodic records of the current states of human and machine assignments as well as evaluations of precision, recall and F-measure against the ground truth have been recorded.

3.3.3 Results

The main question that needs to be answered is whether one interface displayed significant advantage in terms of annotation progress over the other. Figure 3.12 shows the dynamics of the average F-measures of respective interfaces as the function of the session times. It shows that in the earlier stages of the process, before the users of the Batch interface requested reprediction, the UG interface users display higher values due to the machine learning predictions. However, once the predictions are requested by all users of the Batch interface as well, the dynamics of both interfaces closely follow each other along a slowly growing plateau. The UG interface users

finished a little sooner but with a lower accuracy, however, both differences are not statistically significant.

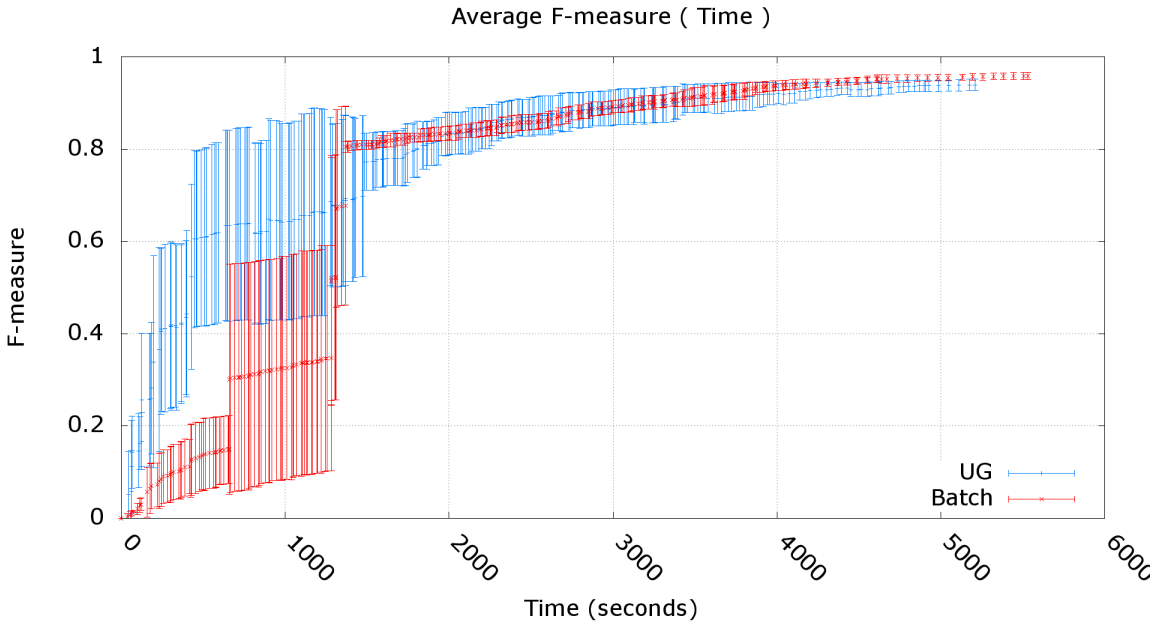


Figure 3.12: Average F-measure for users of the Batch and UG interfaces as a function of time.

Neither interface exhibits significant advantage in terms of the total session time. Differences in both the final F-measure and the finishing times are not statistically significant: Batch interface sessions lasted $4537(\sigma = 743)$ seconds yielding $96(\sigma = 1)\%$ F-measure and UG interface sessions lasted $4401(\sigma = 787)$ seconds with final F-measure values of $94(\sigma = 1)\%$. To understand why this is the case several parallel observations are made.

Accuracy dynamics as a function of user interactions instead of time in Figure 3.13 offers insight. First, it shows that UG interface, despite the users engaging in additional interactions of visibility filtering changes that are not available in Batch interface, require almost half as many interactions to complete the process. Second, it is interesting to observe that Batch interface users request predictions rather soon—between 200 and just over 500 interactions were enough for all 4 users to accumulate

what they believed to be a representative training set. In light of the entire process taking 2700 interactions on average this shows that actual users of such interface function differently in practice than modeled by previous works, in which half of the data or more were provided for the training set. Thirdly, this plot shows that although both interfaces are largely on par early in the process the tail of the sessions where corrections and confirmations of predictions take place in both interfaces, UG interface requires significantly fewer interactions, which suggests that many objects are indeed labeled or confirmed together.

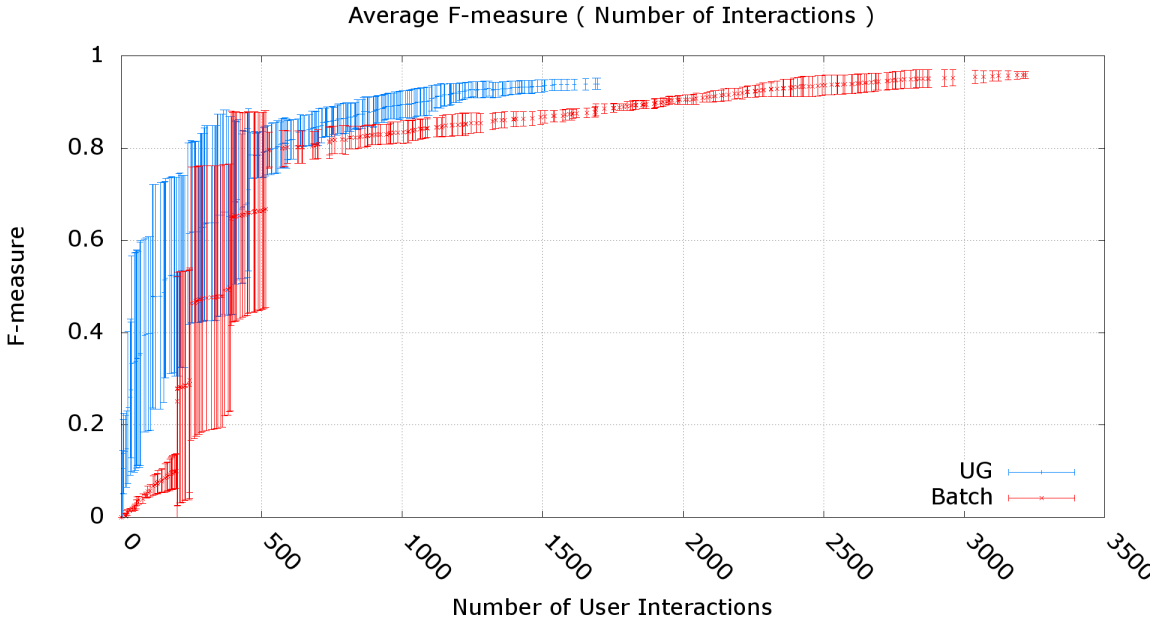


Figure 3.13: Average F-measure for users of the Batch and UG interfaces as a function of number of user interactions (selections, labeling, confirmation, visibility changes).

The observation that users were indeed capable of using the group selection tools and reliably conveying information about objects in them to the system is further confirmed in Figure 3.14, which shows distribution of correct and erroneous (as per ground truth) labeling by the size of the selection group when a label is confirmed. Only a little over 400 objects ($\sim 1/3$ of the data set) on average were labeled alone. This means that the vast majority of objects have been labeled by groups, some as

large as 268 objects. An important observation that can be made from this plot is that the size of the group does not correlate with the amount of incorrect labeling a user provides. As a matter of fact, vast majority of the mistakes are made objects 1-by-1, followed by small groups of up to 5 objects, while larger groups exhibited very few mistakes occurring.

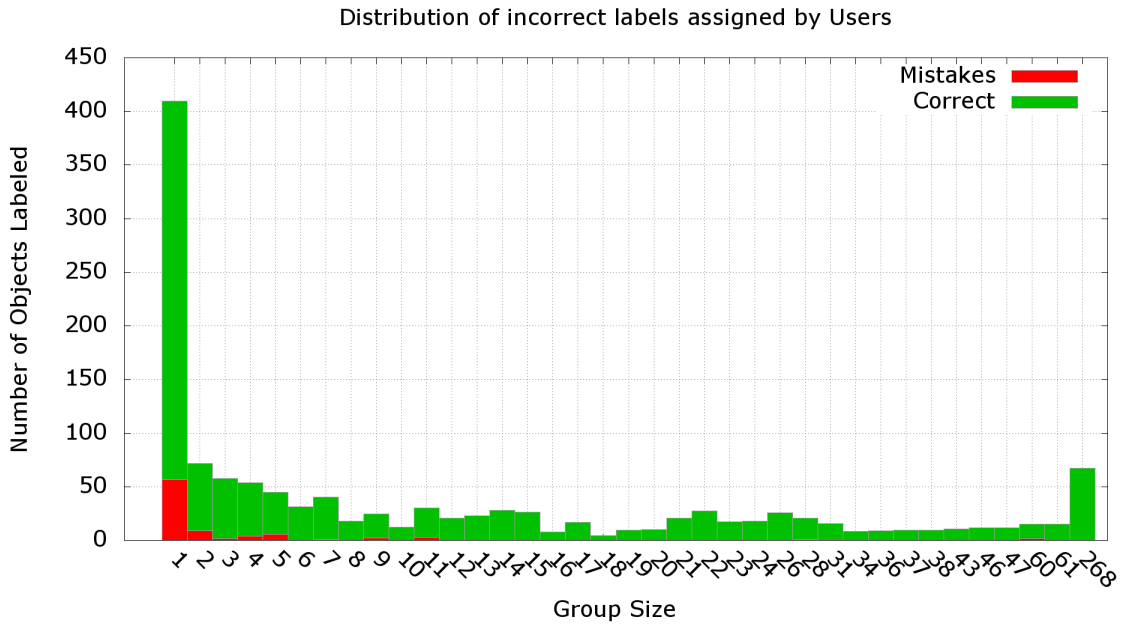
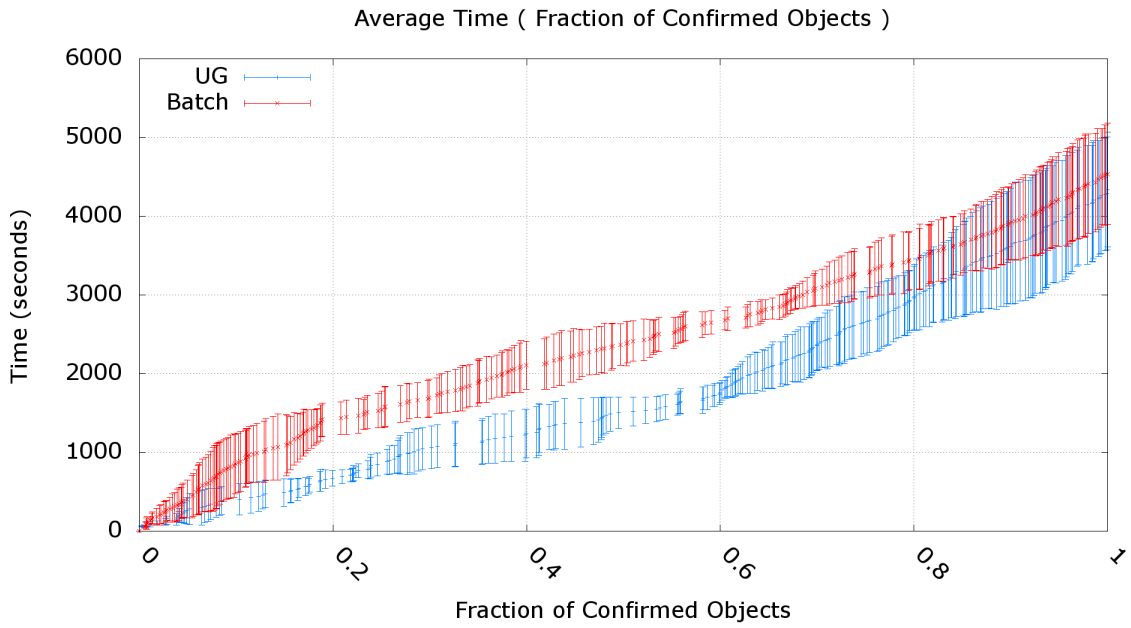
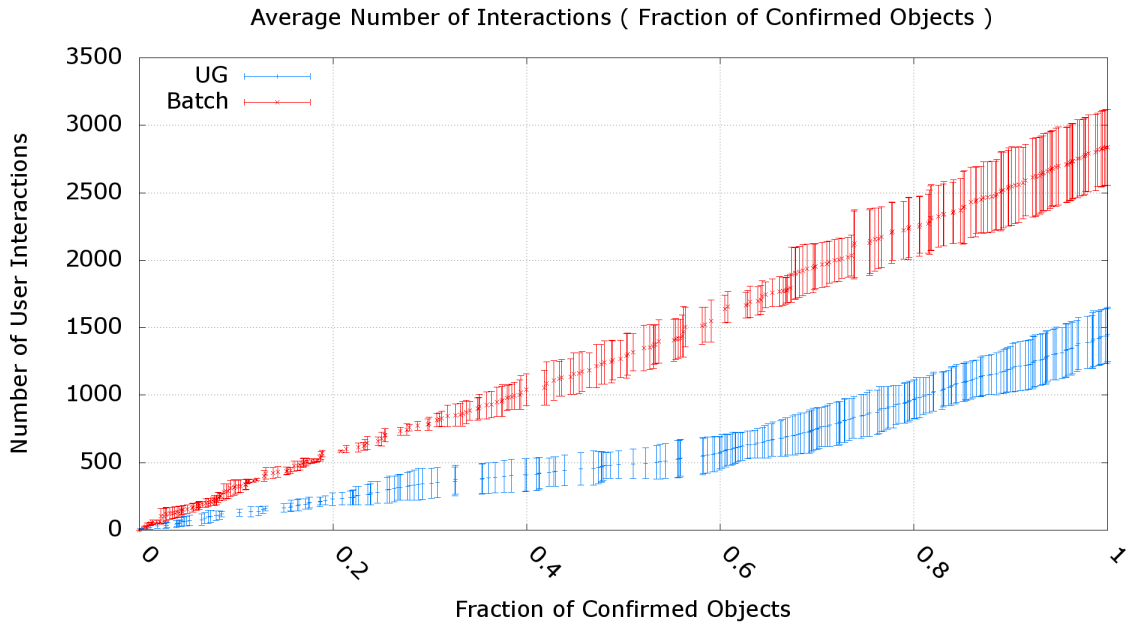


Figure 3.14: User mistakes distribution by group sizes for the UG interface. Average number of objects whose label was submitted in groups of given sizes is represented by the total height of respective bars. The height of the red fractions of the bars are average number of these objects that users mislabeled per ground truth.

F-measure plots demonstrate the joint evaluation of users' annotation efficiency and classifier's ability to correctly capture and propagate the differences between categories, however seeing the dynamic of users' sole contribution is relevant to understanding what causes the observed results of using the interfaces. Figure 3.15a shows average amount of time required to confirm labels for given fractions of the entire data set. It shows that although early in the process Batch interface users were slower to confirm objects, closer to the end of annotation the angle of the UG plot becomes worse indicating that users of the UG interface needed increasingly longer



(a) Average times needed to confirm given fractions of objects in the data set.



(b) Average number of user interactions needed to confirm given fractions of objects in the data set.

Figure 3.15: Dynamic of the label confirmations.

times to proceed. The same dynamic also appears in the plot in Figure 3.15b, which instead of time shows the average number of interactions necessary to confirm given fraction of objects. Here the increase of the angle of the UG plot persists. This tells

us that it wasn't simply taking longer closer to the end because users were tired, but that they actually increased the number of interactions between confirmations. This is not the case for the Batch interface. Although the earlier progress in terms of time for Batch interface users was slower, this was not the result of many interactions but rather indeed a slow start - users were searching for the objects to add to the data set. In terms of number of interactions, however, the angle of the line hardly changes throughout the process. This can be explained by the simplicity of the procedure for Batch interface - select, confirm or label, repeat.

These observations demonstrate that users could successfully take advantage of the group selection tools and visibility filters, which indeed diminished the number of necessary interactions while not hurting accuracy. However, having these tools available required more action planning by the users, which took additional time between providing labels. This resulted in the advantage in number of interactions not reflecting in the overall time lengths of the entire sessions, losing the potential benefit to the added interface complexity.

Chapter 4

Active learning annotation

As results of the Chapter 3 suggest, the actions that need to be made between providing labels—decisions about what objects to label next, camera navigation and object selection—take much time and effort from the annotator. Yet, label confirmation is what really requires users' knowledge and participation. Selecting objects and centering the virtual camera on the selection is something a machine can do. Also, since ultimately all objects need to be confirmed, there is no need to decide on what subset of objects needs to be shown and confirmed. Each one will end up being selected at some point in the session and hence only the order in which they are selected should be decided upon.

Active learning is a field of machine learning in which an algorithm interactively queries the annotator about samples that it deems most relevant to producing good prediction quality. The algorithms employs logic often guided by a *utility function* to choose such objects. In line with this philosophy of the machine making the selection decisions leaving the user to only provide correct annotation, the interface described in this chapter that takes advantage of this approach is called Active Learning (AL) interface.

4.1 Interface

Delegating the selection to the system offers several advantages that the interface should cater to. Firstly, it relieves the user from the necessity of constantly switching from the task of labeling to the task of analyzing multiple objects and making decision on what to select next before providing a single annotation. Secondly, it removes the loss of time dedicated to navigating between consequent selections since the virtual camera can be operated by the machine, which knows the destination object. Thirdly, this mechanism allows us to greatly simplify the interface by removing all of the accompanying tools. This further reduces the time it requires the user to provide labels by diminishing the amount of interaction-related decisions.

The simplification of the interface that immediately follows from the active learning strategy is the removal of the selection tools, both the single object and the frame group ones. Since the user does not need to take care of this, neither are the visibility selection tools necessary anymore.

This simplification, however, opens a question of how the machine should present the objects that it selects and handle the transition between selections. As exit questionnaire answers suggest, individual shapes of objects are as important for annotator's understanding of the label of the objects as their local context. With UG and Batch interfaces, the user controls the virtual camera while navigating between objects and making selections. This allows the user to maintain context awareness as well as acquire the necessary zoom for shape recognition at the level of their own comfort and desire. With the machine making selections, the way selected objects are shown needs to be chosen.

Possible approaches to addressing transition need to balance providing a zoom level that reveals the individual shape of the selected objects with exposing enough relevant context. Maintaining the entirety of the context available to the user is one extreme of this spectrum, requiring a zoom level that is often too small to understand

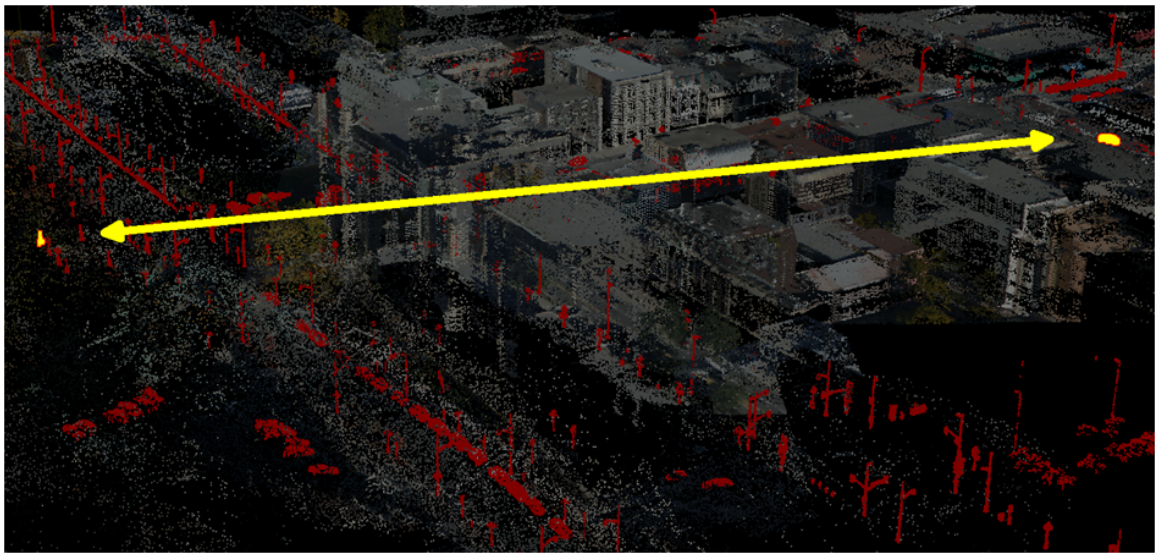


Figure 4.1: Two objects that the AL interface selected immediately one after the other are too far apart to reliably show their individual shapes in a common context.

individual shapes as can be seen in Figure 4.1. Showing only the selected objects runs into the risk of preventing the user from seeing contextual cues. This is especially relevant when recognizing point clouds in which appearance is confusing to the annotator due to occlusion, noise or low sampling as shown in Figure 4.2. Certainly, sequentially selected objects can be close to each other to make full context exposure feasible; and for many objects the shape is descriptive enough that the context will not matter, allowing the maximum close-up to work. However, neither of these extremes are universally applicable as the illustrations show and hence a more flexible approach is desirable.

One option is to create smooth transition animations with the virtual camera, which allow both continuous context observation as well as ultimate zoom on the target object. This, however, introduces a complex challenge of balancing the limitations of the temporal properties of human attention [117] with the additional time it takes to play the transition animations. Camera transition time investment is further questionable because only local context matters for understanding of an object as opposed to knowing what objects are located two city blocks away. Thus ani-

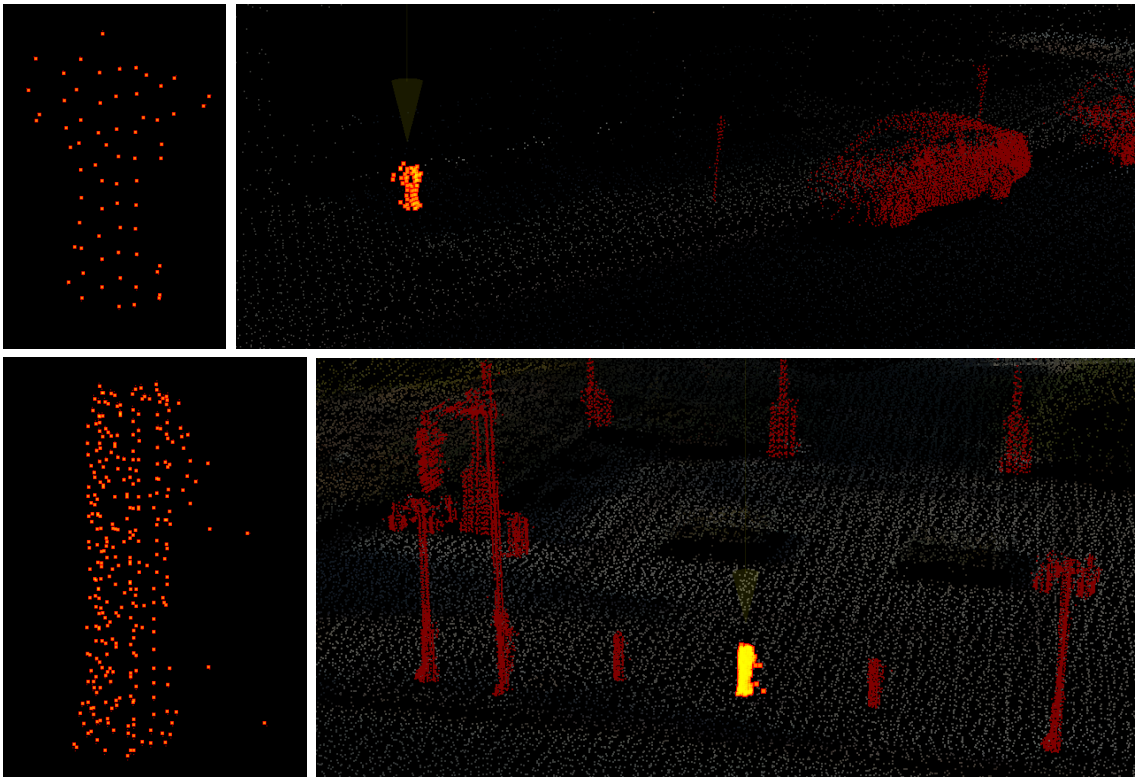


Figure 4.2: Context matters in point clouds annotation. Top row shows fire hydrant and bottom row shows a short post. Left column shows maximum zoom close-ups and right column shows the same objects within their local context.

mated transitions over larger distances will be mostly ignored either due to temporal attention limitation or due to irrelevancy, meanwhile consuming time.

Furthermore, however the transition is carried out, the view angle under which the objects are exposed are also important. If the selected point of view is not exposing relevant features, then the user will need to switch from labeling task into a navigation task to look for a better view. This is especially not favorable for *expert users* who use keyboard shortcuts for label assignments and thus switching to mouse to navigate and back to keyboard to label is an additional irritating distraction. Diminishing the need for these interactions is thus desirable as well. Virtual camera planning algorithms [27] that aim to expose the shape of 3D objects on 2D displays is an active research area. To expose the shape of objects various cues have been proposed in literature [83, 90], such as shading, texture and highlights. However, scans of natural scenes are not well

suited for many of these. In addition to often missing scans of parts of surfaces due occlusion, point clouds do not contain topology information and extracting surface from them to compute these properties is a challenging open problem of its own (e.g., [47]).

One cue, however, which is readily available without carrying out complex pre-computations, is motion and the AL interface takes advantage of it by employing the following selection display strategy. Once the system has decided on what object should be selected next, the interface centers on that object. Zoom level is chosen such that the sphere that bounds the axis-aligned bounding box of the object completely fits into the region of the screen not obscured by the control elements such as the label buttons. This approach zooms further for the larger objects whose bounding boxes are broader and zoom closer to smaller ones, which often require more detailed evaluation of intricate details. Once the virtual camera is positioned in this manner it automatically engages orbital rotation around the selection. The polar angle smoothly oscillates between 0 and $\frac{\pi}{2}$ with the angular velocity inversely proportional to the distance between the camera and the object.

This approach addresses the desired features listed above. The transition from one local context to another is immediate thus is not wasting time and user's attention on irrelevant information. Yet the trajectory of the camera exposes multiple projections of the selected object on the screen as well as varying size and direction of local context in a short period of time without the user having to perform any interactions (Figure 4.3). Decreasing the angular velocity at larger distances from the objects accounts for the camera capturing more local context, which takes longer to visually analyze. If for some objects this still misses relevant information or more context is required, the user can regain control by simply engaging in mouse manipulations or freeze camera motion with a keyboard shortcut. However, this is reserved for unusual or complicated cases.

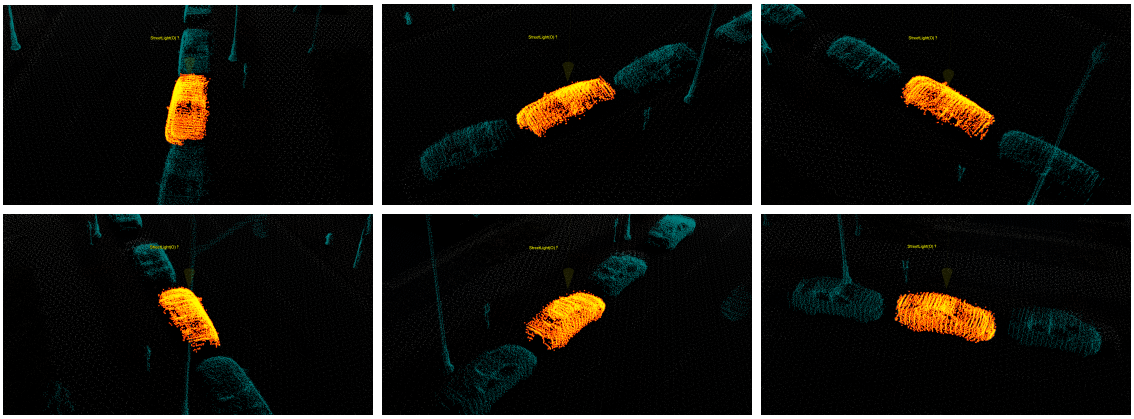


Figure 4.3: Some frames which an orbiting camera captures after an object is selected. Changing polar angle exhibits different projections of the object and local context.

In summary, the AL interface differs from the UG and the Batch interfaces in removing the unnecessary selection tools and adding centering of the camera and its orbiting around the newly selected object. All other tools including labeling, process(undo/redo operations), and navigation controls remain. Also, the textual hints of the currently assigned label name hovering over the current selection as well as the predicted label stressed in the label menu are all left intact.

4.2 Sampling strategies

The AL interface requires the machine to have a strategy to choose which object to select next. What strategy should it be? If all objects are to be selected at some point or the other is there a difference in showing them in any particular order? From the point of view of the total number of interactions - no. However, the better the classifier's predictions are and the sooner they become good, the less the annotator has to correct a label. Correcting a prediction means either sifting through the label menu and pointing at the right one with a mouse or finding the proper keyboard shortcut, while confirming it means simply pressing the confirmation key. Producing better predictions earlier also has the additional benefit of having higher accuracy if the user interrupts the session early—a property outside of the formulated goal of this work but nevertheless a good feature for a practical annotation system.

A variety of approaches have been suggested in machine learning literature for choosing samples that a user should annotate [42, 101]. These methods range from simplest ones of identifying the most beneficial objects using prediction uncertainty, to more costly methods that estimate improvement that labeling specific objects can yield to the measure of prediction accuracy, to maintaining a committee of differently parametrized predictors and letting them vote for the most desirable samples. All of these methods require a *utility function* that should be chosen such that values that it produces for each object could be used to decide on the sample that ends up being queried. The utility function is chosen depending on the desired goal, intended usage case and computational complexity constraints. As described in greater detail in Section 2.2.3, practically these functions are designed with the goal of improving the prediction of the classifier algorithm most with less samples being annotated. Since the goal of this work is annotation of the entire data set and the accuracy of the classifier is secondary, there is no incentive to employ the most computationally expensive methods.

This work considers the most common uncertainty sampling utility functions: choosing objects that exhibit the least prediction confidence, highest prediction entropy and the local density weighted versions of the above. Classifiers usually produce not only the prediction labels for a sample x but also assign each prediction label l a confidence value $P(l|x)$, which is often formulated and treated as a likelihood that x belongs to category l. In these terms it is now possible to describe each of the mentioned utility functions. The least confidence strategy selects a sample x^*_{LC} such that:

$$x^*_{LC} = \arg\max_x (1 - P(l^*|x))) \tag{4.1}$$

where $l^* = \arg\max_l P(l|x)$ is the label with the highest likelihood. This is the easiest function to compute, however, it only considers the most likely label. To include information about other labels into consideration, the object with the highest entropy x^*_E is chosen:

$$x^*_E = \arg\max_x \sum_i P(l_i|x) \log P(l_i|x)) \tag{4.2}$$

Uncertainty sampling chooses the most "confusing" samples, considering them most informative. Under an assumption that the objects proximate in feature space look alike and thus likely share a label, this means that it focuses on the exploration of the feature space by putting emphasis on disambiguating the least certain objects. Querying for objects in such a fashion results in *sampling bias* [13] with the algorithm focusing on samples along the class boundary, or simply querying outliers rather than representative samples, which is a problem for training a good classifier. This bias is often addressed by weighting the uncertainty values by a measure of representativeness. One such method, proposed in [102] and considered in this work, is density weighting. It postulates that most informative samples are not only the most uncertain ones, but also ones that are representative of the underlying distribution. Under this assumption, such sampling chooses sample $x^*_{DW,\phi}$ according to the following for-

mulation:

$$x^*_{DW,\phi} = \arg\max_x(\phi(x)(\frac{1}{U}\sum_{u=1}^{U} sim(x, x_u))^\beta). \tag{4.3}$$

Here, $\phi(x)$ is the uncertainty sampling function of choice (such as least confidence or entropy), U is the size of unlabeled samples, $sim(x, x_u)$ is a function that quantifies similarity between the current sample x and all other unlabeled samples x_u, and β is a parameter that controls the importance of the density weighting.

To make results indicative of the advantages of the interface and interaction model rather than comparing machine learning algorithms, the $NN1$ classifier described in Section 3.2 is used as well. However it needs to produce likelihoods $P(l|x)$ that are used to define the sampling functions described above. To produce these values, the algorithm not only assigns the label l of the closest labeled neighbor $x^l_{NN}(x)$ to a given sample x, but keeps track of the closest labeled instances of every known other label $x^{l_i}_{NN}(x)$. Specifically, it starts with $D(x, x^{l_i}_{NN}(x))$ as it is defined in Equation 3.1. To normalize for local densities, the distances are first shifted down so that the closest K nearest neighbors of every object are 0. This introduces asymmetry, which is restored for every pair of objects x_i, x_j by taking the largest shifted density-sensitive distance between $x_i \to x_j$ and $x_j \to x_i$. This locally-normalized distance is further denoted as $D_{LN}(x_i, x_j)$. The distances are converted into affinities between objects in the feature space

$$A_{FS}(o_i, o_j) = \frac{1}{1 + D_{LN}(x_i, x_j))} \tag{4.4}$$

so that closer neighbors result in higher values. And finally, the likelihoods are produced by normalizing the distribution of affinities for all known labels:

$$P(l_i|x) = \frac{A_{FS}(x, x^{l_i}_{NN}(x))}{\sum_{l_j} A_{FS}(x, x^{l_j}_{NN}(x))}. \tag{4.5}$$

4.3 Results

Evaluation of the AL interface is done against the same goal - improving the time that it requires a human annotator to label the entire data set. To be able to compare the observations of the improvement against the Batch and the UG interfaces presented in Chapter 3, evaluations are carried out on the same data set with the same experimental set-up as described in Section 3.3.2.

The AL interface takes care of the user navigation, however, a good sample selection strategy needs to be chosen. As [100] points out, applicability of active learning is task-dependent, and although it results in the reduction of necessary labeling to achieve a given level of accuracy in the majority of reported results, in some cases it has been shown to perform worse than random sampling (e.g., [44, 50]). Acquiring the necessary pool of human users to carry out evaluations is expensive and is not justified, since training the classifier is not the primary objective of the proposed interfaces. The mode of operation of the AL interface, however, actually requires nothing from the user but providing labels. Having ground truth labels available for the evaluation data allows us to replace human users with algorithmically simulated ones. Simulating users permits carrying out preliminary evaluations of the interface, such as selection of the sampling strategy that improves the classifier better on the given data. It is faster and cheaper than hiring actual users and it removes the variability in human annotators. Simulated user evaluations are presented in Section 4.3.1.

The answer to the main question on whether this interface improves the overall annotation time without deteriorating the quality still can only be answered by an evaluation with human users. It is not trivial for simulated users to give insight into how much visual interface and presentation factors impact the rapidness of labeling. Hence, a human user study with the interface operated by the chosen sampling strategy is conducted and results are presented in Section 4.3.2.

4.3.1 Simulated user evaluations

Simulating a user of the AL interface with available ground truth labels consists of substituting human input with a function that for each queried sample returns its ground truth label. It is faster and cheaper than hiring human users and allows for more annotation session trials, which are sometimes required since ties in utility are broken by choosing any of the candidates randomly. It also isolates the evaluation of the algorithm from including *human factors* such as errors, fatigue and variation in the level of engagement.

It is hard to fully simulate a human user with respect to actual time the entire annotation session lasts. Thus in these preliminary evaluations it is assumed that labeling all objects takes the same constant amount of time and thus number of interactions is taken as a proxy for the session time.

The desired winning sampling strategy should overcome the others in the following criteria to be useful for our goal. First, it should predominantly exceed or be on par with the level of accuracy demonstrated by other sampling strategies at the same stage of the session. This criterion is motivated by more labels being correctly predicted for the unconfirmed objects resulting in users more often confirming instead of searching for a correct label. Secondly, it is desirable that 100% accuracy is achieved sooner than by competitors (a special case of the first criterion).

Results from simulating the uncertainty sampling strategies described above are presented in Figure 4.4. In addition to the least confidence, highest entropy and density-weighted versions thereof, this plot contains two other control strategies. One is random sampling with no machine-generated predictions, in which case any order of labeling is equivalent to another as one interaction results in exactly one object acquiring a correct label. This is a baseline for the worst case scenario when all predictions are always wrong. The other one is random sampling with machine predictions enabled.

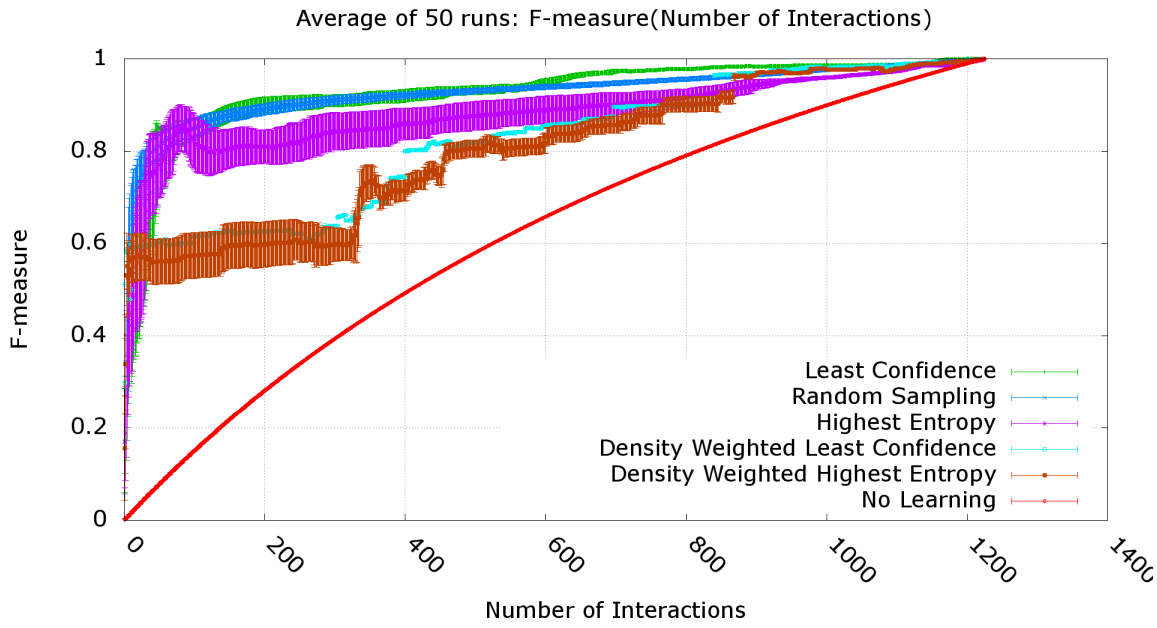


Figure 4.4: Average F-measure as function of number of interactions for simulated users of active learning driven by different sampling strategies.

From this plot it can be seen that the least confidence sampling consistently maintains the highest prediction accuracy. It is closely trailed by the random sampling throughout the first part of the process, however gains advantage later. The highest entropy strategy starts out on par with the leaders, however, very early loses accuracy and never regains it in the later stages. Density-weighted strategies are doing worst throughout most of the process. They also exhibit leap-plateau cycles, suggesting that these strategies are very reluctant to switch to samples from a different cluster until current clusters are exhausted. This explains that the magnitudes of the leaps get smaller further in the process: first the largest and the densest cluster is sampled until its density matches the second densest cluster and so on.

According to the first criteria of the desired sampling strategy, the results indicate that least confidence is the preferable sampling strategy. However, it is hard to evaluate how much they meet the second criterion not only due to all plots converging

and hard to tell apart in the upper right corner, but also because the length of all tails is defined by the longest session among all the runs in the group.

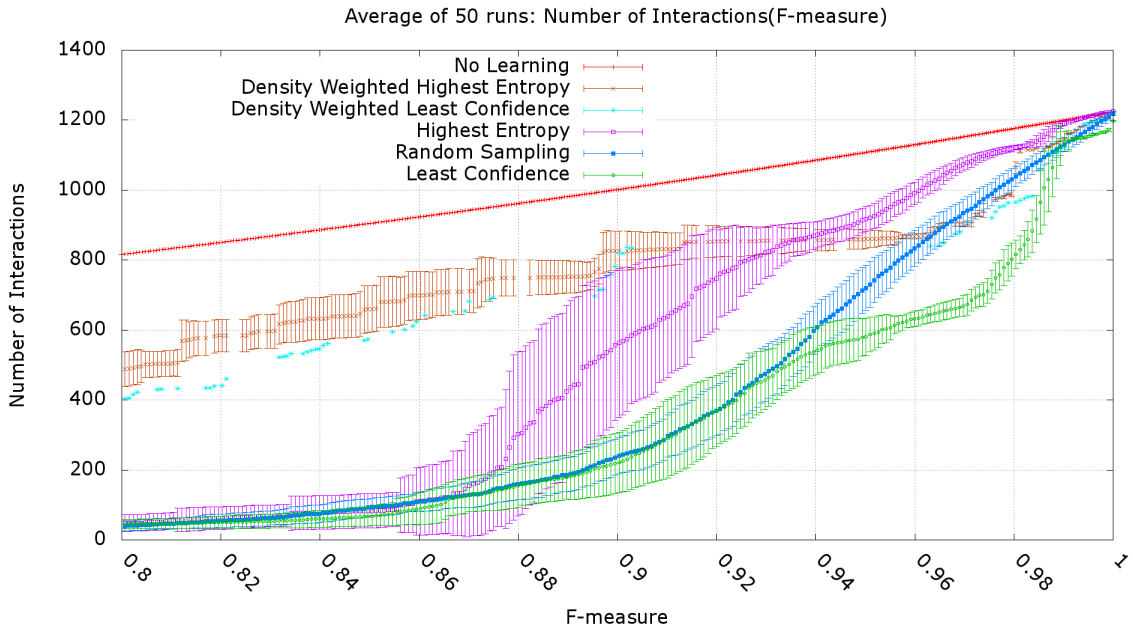


Figure 4.5: Average number of interactions to achieve certain level of annotation accuracy for simulated users of active learning driven by different sampling strategies.

To better explore performance with respect to earlier termination, Figure 4.5 shows the average number of interactions it took to achieve levels of accuracy between 80% and 100%. This plot shows that the least confidence strategy consistently achieves high levels of F-measure in less interactions than others. It reaches 100% accuracy after labeling on average 1197 ± 1 interactions on a 1224 objects data set while others - at 1218 and higher. These observations motivate the choice of the least confidence sampling strategy to be used in the human user studies.

4.3.2 Human user study

For evaluation of the AL interface with human users, 5 participants between ages of 20 and 40, 4 males and 1 female, were recruited. The experiment was set up as described in Section 3.3.2. Users were provided with instructions for the interface, examples and description of labels used, as well as requested to annotate all objects in a smaller data set in the user training mode of the AL interface.

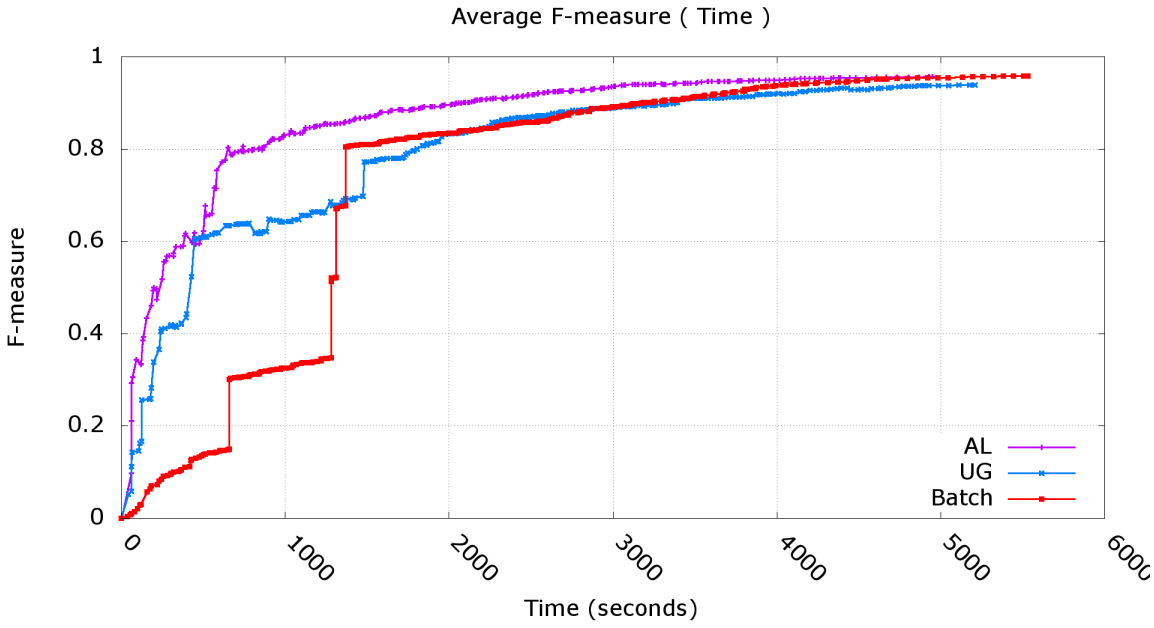


Figure 4.6: Average F-measure as a function of session time.

Figures 4.6 shows the dynamic of the average accuracies achieved by the users of the AL interface compared to those of the UG and Batch interfaces. Throughout most of the session, the average F-measure achieved by the users of the AL interface is consistently higher than that of users of other interfaces at any given time. To better illustrate the dynamic of the later fraction of the sessions Figure 4.7 depicts the average times required to achieve given levels of F-measure above 80%. The plot indicates that the users of AL interface were able to consistently achieve high levels of accuracy faster. It also shows that one of the users of the AL interface was able to achieve the highest accuracy still faster than any user of other interfaces. However,

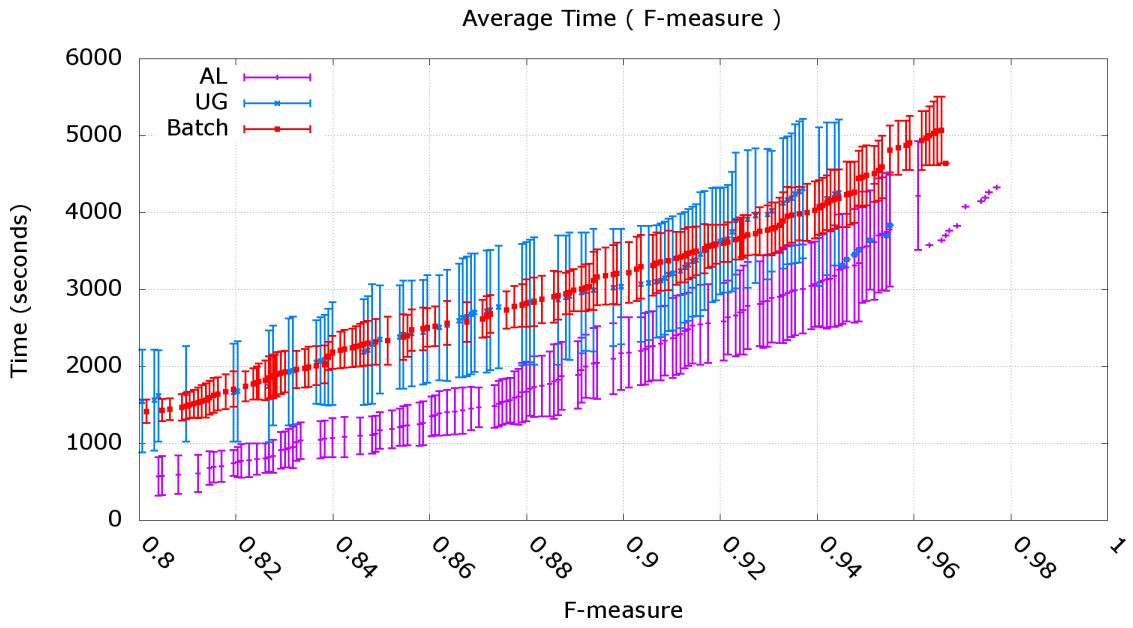


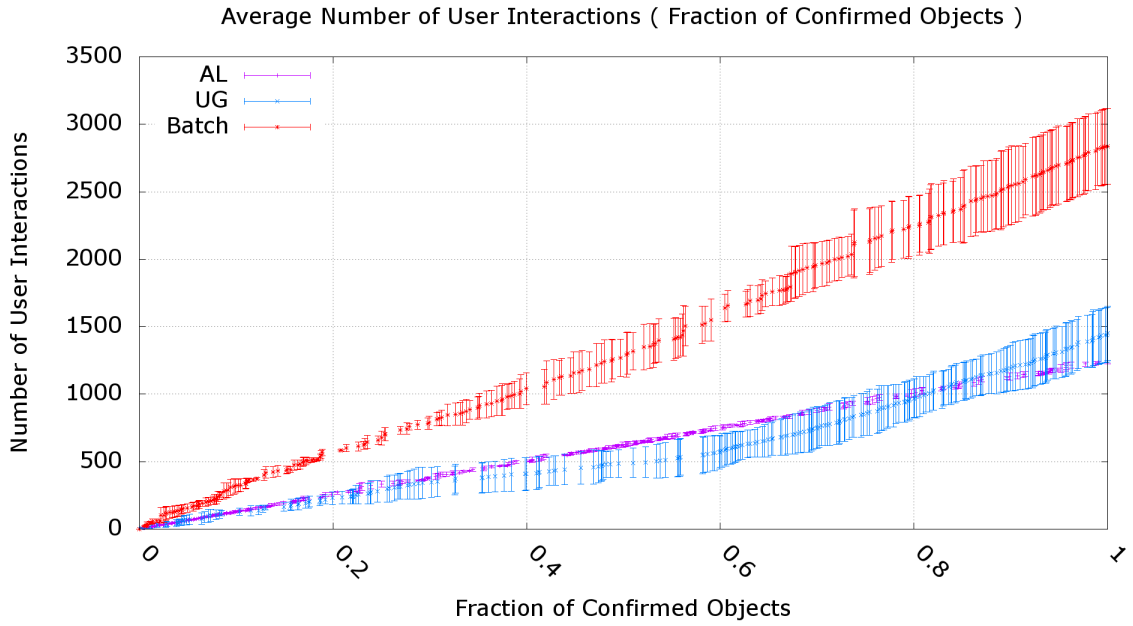
Figure 4.7: Average time required to achieve given levels of accuracy between 80% and 100%.

the average final accuracy of $96(\sigma = 1)\%$ was not significantly different from that of users of other interfaces.

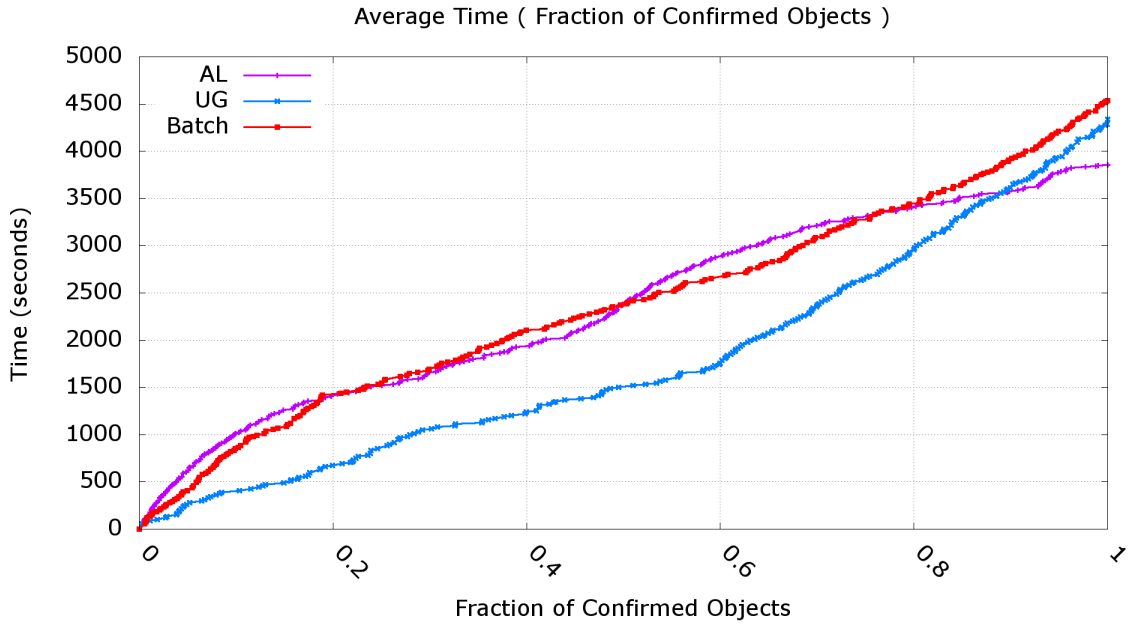
The average completions time of $3855(\sigma = 837)$ seconds for the users of the AL interface users is almost 10 minutes less than the next smallest time shown by the users of the UG interface. AL interface yields 15% and 13% improvement to the average completion times over the Batch and the UG interfaces, respectively. However, due to spreads in the completion times, this advantage is not significantly different either (p-value $\gg 0.05$).

Active learning significantly decreases the necessary number of user interactions as Figure 4.8a illustrates. Just as the case with Batch interface, the confirmation dynamic is linear, however, the number of interactions is expectedly halved, since Batch interface requires additional selection interactions. The plot also indicates that the AL interface requires less interactions than the UG interface. Although the UG

interface saves time by bulk selection and labeling, in the end the added contribution of other interactions (e.g., filtering) results in greater than one interaction per object.



(a) Average number of interactions required to confirm labels for given fraction of objects in data set.



(b) Average time required to confirm labels for given fraction of objects in data set.

Figure 4.8: Dynamic of the label confirmations.

The advantage in the number of interactions, however, does not fully transfer into a consistent advantage in the time scale as shown in Figure 4.8b. Most of the time the line representing the AL interface trails that of the Batch interface. This is explained by both of these interfaces providing the tools to only annotate one object at a time. When there are many unlabeled objects, finding a nearby object to label is not hard and does not require as much effort from the users of the Batch interface either. This dynamic changes towards the end of the session, where the AL becomes more time efficient than either of the other interfaces. At this point the data set is mostly labeled and it becomes more and more challenging for the users of the user-guided interfaces to find and navigate to the objects that are still not confirmed. Meanwhile for the AL interface, tail of the process is "business as usual": it simply takes the user directly to the next object instantly as it has throughout the session. This allows the users to maintain their pace and avoids the slowdown towards the end that is present in both the UG and the Batch interfaces.

Although the AL interface offers a very streamlined approach to annotation, it is limited to annotating one object at a time. Hence the process is inevitably linear in terms of minimal time required from a human to perform all cognitive tasks related to understanding an object and its context. This results in an existing, yet not drastically decisive advantage that the AL interface allows to achieve with respect to the overall annotation time.

Chapter 5

Group active learning annotation

As the literature summarized in Section 2.3 indicates, humans are very good at recognizing entire scenes. Human vision picks up on groups of similar objects, maintaining a summary statistic of what it sees instead of focusing on one object at a time. This allows humans to rapidly evaluate complex scenes in a glance.

For the purpose of an annotation system, however, simply being able to rapidly understand what is on the screen is not enough. Providing the means for rapidly conveying the information about the objects that are seen is as important. Moreover, since the human visual system is very good at recognizing objects, providing annotations in a manner that a machine can understand is actually the bottleneck, since this is the phase that involves mechanical manipulations with the input devices and the latencies that come with them. The better the effort analyzing every object on the screen is translated into providing information to the system, the higher the throughput of the interface is.

Earlier chapters shown that the two approaches - user-guided and machine-driven - to annotation offer their unique advantages to facilitate the process, yet both have disadvantages. The UG interface takes advantage of the human ability to analyze groups of objects and convey the label in kind. However, the added burden of making

selection and navigation decisions and performing respective interactions, as well as additional tools and elements of the interface that the user needs to use eat away most of that advantage. The AL interface addresses this problem by taking on all the selection tasks and the majority of the navigation ones, which allows in turn to simplify the interface as well. However, at the same time it loses the ability to fully leverage the perception capacities of users to acquire annotations in bulk where it is possible.

A hybrid *Group Active (GA)* interface is designed to reconcile these two approaches. It maintains the active approach to annotations, having the system make the selection decisions and perform most navigation for the user. However, instead of selecting objects in accordance to a classifier utility, it chooses groups that both are easy for the user to recognize and are likely sharing a label. Operating in this manner allows users to submit a label for entire groups of objects in a single action without having to spend much effort on manipulations to assemble these groups manually.

To build groups, the system uses a perceptual psychology motivated model. This model works in collaboration with the machine learning algorithm and leverages information received from previous user interactions. Annotations that the user provides, as in other interfaces, help the model to estimate the likelihood that objects share a label. Suggested groups can be labeled or rejected via contraction or expansion requests. Contractions and expansions allow the system to adjust its level of conservativeness in its future group suggestions. The following section describes these mechanisms in details as well as elements of the user interface that enable users to efficiently interact with this annotation system.

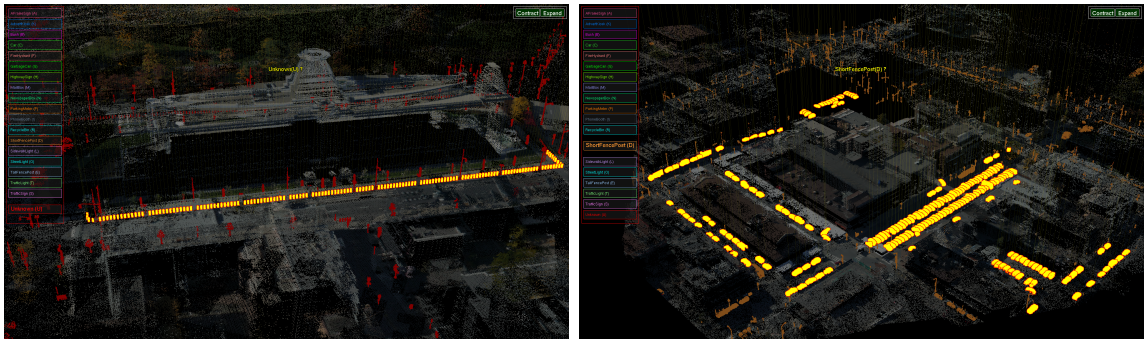
5.1 Interface

The GA interface operates in the following manner. It iteratively selects groups of objects, shows them to the user (drawn in bright yellow from a spinning centered view as shown in Figure 5.1), predicts a single label for all objects in the group (shown in text above the group), and then asks the user to execute one of three actions. The user can either confirm the predicted label (hit the space bar), specify a new label (select a label from a menu on the left of the screen or hit an alphanumeric character associated with the label), or ask the system to contract or expand the group (left-arrow or right-arrow key, respectively). If the user confirms or specifies a label for the group, then all objects within the group are assigned that label and subsequently displayed in the color associated with the label. Otherwise, a new group is shown to the user. The system iterates this group labeling interaction until labels have been confirmed or specified for all objects in the data set.

Since the GA interface, akin to AL, makes selections for the users, the necessary tool set and the mode of operation are similar. There is, however, one important distinction. The GA interface suggests groups automatically according to its internal understanding of the world at a given moment. It will inevitably make mistakes due to incorrect assumptions - either showing a group that doesn't share a label or a group that a user deems too large or too small for another reason. There needs to be a tool that allows the users to remedy such situations efficiently.

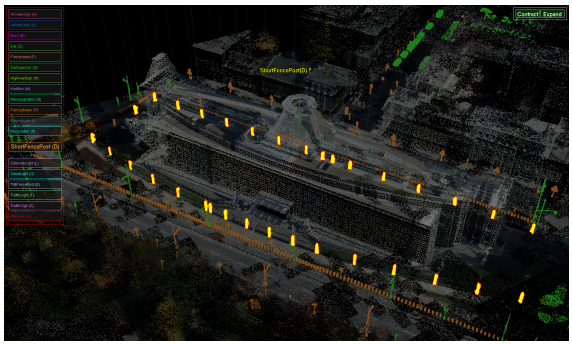
A straightforward approach to allow for these corrections is to let the user directly remove outliers or add missing objects with tools similar to those of the UG interface (visibility filters and region or single click selection). This approach, however, requires a user to analyze a group and then perform manual selection manipulations similar to those in the UG interface, which has been shown inefficient.

The GA interface instead takes an alternative approach. It is built naturally on the ability to suggest groups that fit the criterion of similarity and ease of perception.

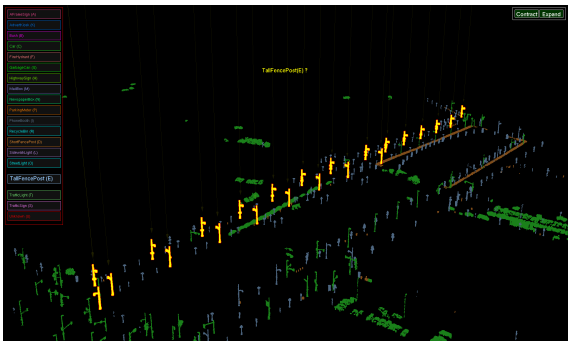


(a) 149 short posts

(b) 210 cars



(c) 33 tall fence posts



(d) 23 street lights

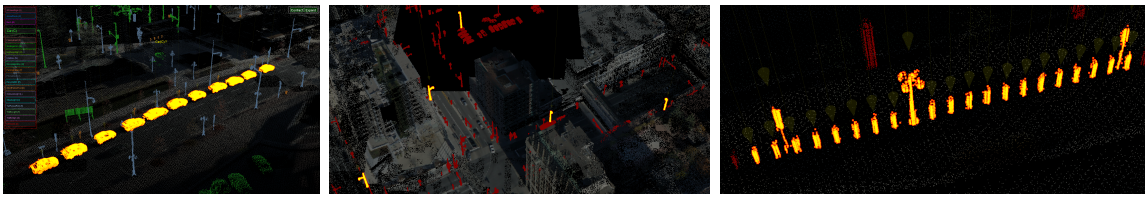
Figure 5.1: Screenshots of the groups suggested by the GA interface. Selected objects are highlighted in yellow, other objects are colored according to their labels.

If the system can select such groups from scratch in the data set, then selecting a supergroup or subgroup that best meets these criteria is a special case of the original function. This approach allows the user to alternate the groups in a simpler manner than performing tedious detailed manipulations. Respective commands—*contract* and *expand*—can be easily mapped to both keyboard shortcuts and visual interface buttons as shown in Figure 5.1.

Aside from additional group manipulation functionality, the visual interface is similar to the AL. It automatically centers on the selected objects and orbits the group. As the Figure 5.1, some groups are large and user may need additional time and even manipulations to carefully evaluate them. However, for larger groups the loss of time for mouse manipulation is compensated by the savings on the amount of labeling information gained when the entire group is labeled.

5.2 Attention model for group assembly

The main research challenge in implementing the GA interface is to develop effective methods for selecting groups to be annotated by a user at each step of an interactive labeling session.



(a) Pattern exists (b) No pattern (c) Outlier

Figure 5.2: Groups that illustrate the desired properties that the suggestions of the interface should meet. From left to right: (a) recognizable patterns in spatial organization of objects is helpful; (b) lack of such patterns is hindering; (c) presence of an outlier turns showing any group into a wasted effort since a single label cannot be provided to the group.

Ideally, selected groups should contain very large number of objects, all of which require the same label, most of which have not been previously labeled, and all of which are arranged in patterns quickly recognizable by a person – then, large groups of objects could be labeled quickly and easily. Figures 5.1 and 5.2 show several possible groups that illustrate what kind of groups the GA interface should and should not produce. Figure 5.2a shows a small group of cars neatly parked alongside a curb in a rather regular pattern. This group is fast to evaluate and the label confirmation in one action is possible, and is further facilitated by the correct prediction for the group—a simple space bar keystroke is necessary. It is different from a small group of streetlights shown in Figure 5.2b. They are very similar in shape, however, they appear dispersed throughout screen, not creating a visual group. This makes it harder to even identify which objects are selected, let alone efficiently compare their shape without much effort and manual camera manipulations. Requiring this amount of interaction from the annotator at the expectation of only 5 objects being labeled is not efficient. A much vaster group with many more streetlights is shown in Figure 5.1d.

This group is better for the purposes of annotation, since the objects form a naturally meaningful patter of streetlights along a single street and are thus easier to visually evaluate. An even larger and more dispersed group of cars is shown in Figure 5.1b and is also deemed beneficial. Although the group occupies a vaster area, it consists of regularly positioned cars that form groups that are easily recognized as cars parked along streets or in parking lots. Even though the user likely needs several zoom-ins and some additional navigation to verify that indeed all of the objects are cars, the advantage of acquiring confirmation of labels for 210 objects in a single action is worth the effort. Finally, a case that should be avoided at all cost is shown in Figure 5.2c. Regardless of how well a group is exhibiting an easy and recognizable pattern and how easy it is to visually understand what the objects in the group are, if there is an outlier present, the time spent analyzing yields no annotation benefit.

Building groups with these properties is non-trivial. Since the labels of objects are not known in advance, the system must predict them and model the probability that a group contains objects of the same type. Since different groups of objects require different amounts of time for a user to recognize their labels, the system must employ a model of human perception to estimate the cost of asking the user about a group. These considerations must be encoded into an objective function that models the benefit (time savings) of labeling objects in a group.

The objective function $B(G, L)$ that the GA interface employs estimates the expected benefit (time savings) of asking a user to select a label from a set L for a group of objects G (rather than annotating the objects one-by-one). Specifically, $B(G, L)$ is defined as:

$$B(G, L) = P_{Label}(G)T_{1\times1}(G, L) - T_{Group}(G, L) \tag{5.1}$$

where $T_{Group}(G, L)$ is the expected time it will take a user to provide a response for a group G; $P_{Label}(G)$ is the expected probability that the user will provide a label for

the group G (rather than contract or expand it); and $T_{1\times1}(G, L)$ is the expected time it would take a user to label all objects in G one-by-one (without group annotation).

This benefit formulation reflects the fact that the total time for the annotation session is reduced by $T_{1\times1}(G, L)$ time if the user provides a label for the group, which occurs with probability $P_{Label}(G)$. $T_{Group}(G, L)$ time is added to the session for the user to process the group, regardless of whether a label is provided or not. So, intuitively, $B(G, L)$ is higher if groups are larger ($T_{1\times1}(G, L)$ is higher), more likely to contain objects of the same label ($P_{Label}(G)$ is higher), and faster for a person to recognize the label ($T_{Group}(G, L)$ is lower).

While this formulation is nice theoretically, it requires estimating three terms ($P_{Label}(G)$, $T_{Group}(G, L)$, and $T_{1\times1}(G, L)$), all of which depend on unknowns (object labels, user behavior, etc.). The following subsections provide details for how the system estimates values for those terms.

5.2.1 Likelihood of group sharing a label

The first challenge is to estimate $P_{Label}(G)$, the probability that the user will provide a label for a given group G – i.e., confirm the predicted label or provide a new one explicitly. This can occur only if all objects in G belong to the same category (require the same label). Thus, $P_{Label}(G)$ is computed by estimating the joint probability that all objects $o_i \in G$ belong to the same category.

The challenge in estimating the joint probability is that the label that will be assigned by the user is unknown. In fact, it is possible that no instances of the correct label for a group have been previously entered by the user, or in fact that no labels have been entered at all. Therefore it is not always possible to simply train a classifier to estimate the probability of assigning any given label based on previous training data.

Instead, the system employs an estimator for the probability $P_{Label}(o_i, o_j)$ that any two objects o_i and o_j have the same label, and then combine those probabilities to estimate the joint probability of a single label for all objects in the group. To do this, we compute the product of $|G| - 1$ pairwise probabilities, where the pairs are chosen to be the highest probability ones that span all objects in G. That is, if MST_{Label} is the minimum spanning tree of a fully-connected graph, where nodes represent objects and edges represent $1 - P_{Label}(o_i, o_j)$, then:

$$P_{Label}(G) = \prod_{(o_i,o_j)\in MST_{Label}} P_{Label}(o_i, o_j) \tag{5.2}$$

This method for combining the pairwise probabilities is based on two assumptions. First, the pairwise probabilities, $P_{Label}(o_i, o_j)$, are more reliable for high probability pairs than for any other (i.e., long-range distances in the feature space should not be trusted), and thus combining probabilities for pairs of objects connected in the MST is more reliable than using other pairs at all. Second, the pairwise probabilities for pairs of objects in the MST are independent, and thus they can be multiplied to estimate the joint probability.

This formulation for $P_{Label}(G)$ requires estimation of the $P_{Label}(o_i, o_j)$, the probability that any two objects o_i and o_j have the same label. To do this, we follow the general density-based distance approach used in this work, but with several augmentations appropriate for the setting of this interface.

Equation 4.4 defines the density-sensitive locally normalized affinities $A_{FS}(o_i, o_j)$ between two objects in the feature space. This formulation alone, however, is static and does not account for user's interactions. Considering prior interactions of labeling and contraction is especially relevant for the GA interface to avoid repeatedly showing groups that do not share a label.

The probability that two objects will be assigned the same label is defined as:

$$P_{Label}(o_i, o_j) = A_{FS}(o_i, o_j)C(o_i, o_j) \tag{5.3}$$

where $C(o_i, o_j)$ is a penalty term to account for pairs of objects that have appeared in groups of objects contracted by the user in previous interactions. If two objects were part of a group that the user declined to label in the past, they probably should not be grouped again. Initially, $C(o_i, o_j)$ for every pair of objects is one. Then, every time a group is contracted by the user all $C(o_i, o_j)$ where $o_{i,j} \in G$ are multiplied by $1 - 1/|G|$. This factor provides a soft penalty for showing poor groups of objects repeatedly.

To account for objects that have already been assigned a label by the user (e.g., in a prior interaction within the same session), the distance $D_{LN}(o_i, o_j)$ (as defined in Section 4.2) is set to ∞ if the nearest labeled neighbors of o_i and o_j are assigned different labels. Otherwise, it is set to be the minimum of $D_{LN}(o_i, o_j)$ and $max(D_{LN}(o_i, o_j^{NN}), D_{LN}(o_j, o_i^{NN}))$, where o_i^{NN} and o_j^{NN} are the previously labeled objects (with the same label) that are closest in feature space to o_i and o_j, respectively. This adjustment creates zero-distance "wormholes" between separated clusters of objects that share the same label, a method akin to *must-link* constraints in previous work on constrained clustering (e.g. [130]). Using this adjusted $D_{LN}(o_i, o_j)$ in the computation of the $A_{FS}(o_i, o_j)$ (Equation 5.3) allows $P_{Label}(o_i, o_j)$ to incorporate the labeling activity provided by the user.

5.2.2 Estimating time for user to provide a response

The next problem that needs to be addressed is developing a model for $T_{Group}(G, L)$. It estimates how long it will take a user to provide an annotation response when shown a selected group G for a given set of candidate labels L.

Such a model is difficult to estimate accurately, since it depends on the spatial reasoning aptitudes of individual users and complex factors related to perception of groups. However, basic principles of perceptual psychology can be applied to form a simple, approximate model that is adequate for this system. The model used in this work is a sum of three terms:

$$T_{Group}(G, L) = T_{Id}(G, L) + T_{Verify}(G) + T_{Cmd}(L) \tag{5.4}$$

where $T_{Id}(G, L)$ is the time required to identify a label for the group, $T_{Verify}(G)$ is the time to verify that all objects in the group require the same label, and $T_{Cmd}(L)$ is the time to convey the response to the system. The second term is zero if there is only one object in G, and so

$$T_{1\times1}(G, L) = (T_{Id}(G, L) + T_{Cmd}(L))|G| \tag{5.5}$$

Predicting the Time to Recognize a Label for a Group

The human visual system is capable of rapidly grasping the gist of images representing scenes. As mentioned in Section 2.3, after exposure of as little 100ms, people can answer specific questions about what they saw. The process of recognizing sets of similar objects is also fast and robust. Since a group of objects can be represented cognitively with summary statistics [8], recognition times for salient groups can be extremely rapid. Since items perceived as a group can "pop-out" from clutter [60, 93], they can often be recognized as a whole more quickly and accurately than as individuals [1, 89].

Accordingly, the time $T_{Id}(G, L)$ for a person to identify the label for a group is modeled as a function that is independent of the size of the group, but dependent on the numbers of labels to choose from. Specifically, it is estimated as the choice

reaction time (CRT) of choosing among $|L|$ labels using Hick's law [57]:

$$T_{id}(G, L) = a_{Id} H_{eq}(|L|) \tag{5.6}$$

where $H_{eq}(n) = \log_2(n + 1)$ is the information-theoretic entropy of a decision among n equiprobable options, and a_{Id} is a processing speed constant factor.

Predicting the Time to Search for an Outlier

Although recognizing the label for a group of similar objects is extremely fast, the process of checking whether there is an outlier within a group can be much slower. In this work, the time $T_{Verify}(G)$ for a person to verify that the labels of all objects in the group are the same category is modeled using a sequence of binary decisions, where each binary decision determines whether two objects within G are in the same category or not. Since the better the Gestalt pattern of objects the more likely the visual attention will group them together, this work assumes that when a person decides whether an object is similar to others in a group he or she will compare it to the one it appears in the best pattern available in the group. The total time is estimated as a sum of binary decisions made for pairs of objects connected in a minimum spanning tree (MST_{Gest}):

$$T_{Verify}(G) = \Sigma_{(o_i,o_j)\in MST_{Gest}} T_{Verify}(o_i, o_j, G) \tag{5.7}$$

where $T_{Verify}(o_i, o_j, G)$ is the time it takes a person to decide whether two objects within G have the same label, and MST_{Gest} is a minimum spanning tree constructed based on a cognitive measure of affinities between objects, A_{Gest}, which is defined below.

The time it takes for a person to make a decision about whether two objects have the same label or not $T_{Verify}(o_i, o_j, G)$ is modeled as a choice reaction time (CRT) using Hick's law:

$$T_{Verify}(o_i, o_j, G) = a_{Verify}(o_i, o_j, G)H(P_{Label}(o_i, o_j)) \tag{5.8}$$

where $a_{Verify}(o_i, o_j, G)$ is a task complexity function that depends on geometric properties of the object pair within the group G and $H(p) = p\log_2(\frac{1}{p}+1)+(1-p)\log_2(\frac{1}{1-p}+1)$ is the information-theoretic entropy of a binary decision, where p is the probability of each outcome, in our case modeled as the affinity between the two object shapes ($p = P_{Label}(o_i, o_j)$).

When estimating the task complexity function, $a_{Verify}(o_i, o_j, G)$, people are expected to recognize the match between labels of two objects more quickly if they are closer to one another and/or are arranged in a regular pattern governed by Gestalt rules [128, 129]. These factors are combined as follows:

$$a_{Verify}(o_i, o_j, G) = T_{Verify}(A_{Gest}(o_i, o_j, G) + (1 - A_{Gest})d(o_i, o_j)) \tag{5.9}$$

where T_{Verify} is the fastest possible time required to recognize whether two objects have the same labels, $A_{Gest}(o_i, o_j, G)$ is a measure [0-1] of how much the pair of objects participates in a group with strong Gestalt principles, and $d(o_i, o_j)$ is the distance between the objects in relative units

$$d(o_i, o_j) = \frac{|o_i - o_j|_2}{max(|o_i|_\infty, |o_j|_\infty)} \tag{5.10}$$

Intuitively, the value of $a_{Verify}(o_i, o_j, G)$ is equal to T_{Verify} for objects connected by strong Gestalt cues (when $A_{Gest}(o_i, o_j, G) = 1$), and otherwise is larger than T_{Verify} when Gestalt cues are weaker and/or the distance between objects is larger.

Estimating the Effects of Gestalt Cues

Gestalt principles of visual perception [128, 129] attempt to describe how people tend to organize visual elements into groups. These principles include similarity (objects that look alike are grouped together), proximity (objects that are close to each other are grouped), continuity (groups form if they are aligned with each other), and "good Gestalt" (objects are grouped if they are parts of simple regular patterns). While similarity principle is enforced in the formalism used in the model through employing $P_{Label}(o_i, o_j)$ in Equation 5.8, the principles of proximity, regular pattern and continuity are enforced via $A_{Gest}(o_i, o_j, G)$:

$$A_{Gest}(o_i, o_j, G) = R(o_i, o_j, G)S(o_i, o_j) \tag{5.11}$$

where $R(o_i, o_j, G)$ is a value [0-1] representing how much o_i and o_j participate in a regular pattern within G, and $S(o_i, o_j)$ represents a measure of the proximity of two objects relative to their sizes.

When estimating $R(o_i, o_j, G)$, objects are considered to be forming a regular pattern if they are situated along a line and share similar spacing between them. Hence the smallest size of the group that can exhibit such traits is three, and $R(o_i, o_j) = R(o_i)$ is set to 1 for all pairs/objects in groups of less than three. To evaluate the regularity for any three objects $o_{i,j,k}$ the translation between every two is replicated in either direction and the distances to the third is computed. Then the smallest of these distances $d_r(o_i, o_j, o_k)$ is used to compute regularity score as

$$R(o_i, o_j, o_k) = \exp(-d_r(o_i, o_j, o_k)/min(|o_i|_\infty, |o_k|_\infty, |o_k|_\infty)) \tag{5.12}$$

Regularity available for two objects o_i and o_j within a group G is defined as:

$$R(o_i, o_j, G) = max_{o_k \in G}(R(o_i, o_j, o_k)) \tag{5.13}$$

$S(o_i, o_j)$ accounts for the effect that similarities of objects are harder to recognize when the objects are smaller on the screen. Since the virtual camera is positioned so that the entire group is within view, objects that are further apart relative to their sizes in world space appear smaller on the screen after perspective projection. So, the effects of object proximity also incorporate information about their relative size as:

$$S(o_i, o_j) = \frac{1}{1 + max(0, d(o_i, o_j) - c)} \tag{5.14}$$

Estimating Time for User to Enter a Label

Finally, the time $T_{Cmd}(L)$ for a person to provide a label to the system is modeled as another choice among $|L|$ labels using Hick's law [57]:

$$T_{Cmd}(L) = a_{Cmd} H_{eq}(|L|) \tag{5.15}$$

where a_{Cmd} is a processing speed constant factor.

In summary, this model predicts that groups can be labeled more quickly if they have objects with more similar shapes, stronger regular patterns, closer proximities in 3D, larger sizes on the screen, and fewer potential labels.

5.3 Group assembly

Given the objective function, the main algorithmic task is to select the group of objects with largest benefit to present to the user at each step of the interactive labeling process. The overall goal is to construct a sequence of groups that minimize the total time required by a user to label and/or confirm every object in the data set. Achieving this goal is NP-Hard. Choosing just one group is an instance of the optimal subset selection problem, which is NP-hard with simple objective functions (e.g., [132]), let alone optimally choosing the entire sequence using the described benefit function that non-linearly depends on the other objects in the group. Hence to make use of this formalism in an interactive system, the GA interface selects groups that approximately optimize this goal given current estimates of the object labels and annotation times.

To limit the search space, first a hierarchical clustering tree $\mathfrak{T}$ of all objects in the data set $\mathfrak{D}$ is constructed. Each node of $\mathfrak{T}$ represents a group of objects in the leaf nodes of its subtree. Then, the best group is sought for among its non-root nodes by evaluating $B(G, L)$ of these groups and picking one with the highest value. This limits the search space to a set of candidates that is linear in $\|\mathfrak{D}\|$.

Even with this reduced search space, building $\mathfrak{T}$ is non-trivial. In particular, hierarchical clustering based on $B(G, L)$ can lead to groups with poor regular patterns, since decisions made greedily in the early stages of the algorithm are based mainly on shape similarity and spatial proximity between objects (e.g., when just two objects are merged into a group) and therefore are likely to form small groups that cannot later be merged into large ones with good regular patterns. To avoid this, instead of joining groups according to $B(G, L)$, during the tree construction groups are joined in the best-first order of $A_{Gest}(o_i, o_j, \mathfrak{D})$ - a version of $A_{Gest}(o_i, o_j, G)$ from Equation 5.11 that represents the best possible Gestalt score of a pair o_i, o_j with any third object from the entire $\mathfrak{D}$, not only from G. Specifically, the algorithm traverses pairs of

objects $o_i \in G_i, o_j \in G_j, G_i \cap G_j = \emptyset$ in a descending order of $A_{Gest}(o_i, o_j, \mathfrak{D})$ until $B(G_{ij}, L) \geq \max(B(G_i, L), B(G_j, L))$. Once such pair $(o_j, o_j)_k$ is discovered groups G_i and G_j are merged into G_{ij}. This enforces the priority of joining pairs of objects that are globally in stronger patterns.

Constructing $\mathfrak{T}$ in this way still requires evaluating $B(G, L)$ for $O(\|\mathfrak{D}\|^2)$ candidate groups. It is thus desirable to avoid having to recompute MST_{Label} (in Equation 5.2) and MST_{Gest}(in Equation 5.7) in every evaluation. When a decision to join groups G_i and G_j into G_{ij} is made during the tree construction, it is mandated by a pair of objects (o_j, o_j) in respective groups that appear in a globally strong pattern according to $A_{Gest}(o_i, o_j, \mathfrak{D})$. A set of all such edges $\{(o_j, o_j)_k\}$ that contributed to the appearance of a group G on $\mathfrak{T}$ thus forms a spanning tree $ST^{\sim}(G)$ over all objects in G with edges, chosen with the objective of the largest $A_{Gest}(o_i, o_j, \mathfrak{D})$ and non-decreasing benefit. $ST^{\sim}(G)$ is not exactly MST_{Gest} due to $A_{Gest}(o_i, o_j, \mathfrak{D}) \neq A_{Gest}(o_i, o_j, G)$, however $ST^{\sim}(G)$ is readily available without additional computations and allows for a recursive formulation of $T_{Verify}(G_{ij})$ from Equation 5.7

$$T^{\sim}_{Verify}(G_{ij}) = T_{Verify}(o_i, o_j, \mathfrak{D}) + T^{\sim}_{Verify}(G_i) + T^{\sim}_{Verify}(G_j) \tag{5.16}$$

which reuses values previously computed for G_i and G_j. For the same reason of immediate availability $ST^{\sim}(G)$ is used instead of MST_{Label}. However, $ST^{\sim}(G)$ is built not directly considering $P_{Label}(o_i, o_j)$ at all. Hence, using a product on $ST^{\sim}(G)$ as in Equation 5.2 is likely to over-constrain construction of $\mathfrak{T}$ and build only very small groups due to many elements of the product being much smaller than what they would have been if the actual MST_{Label} was used. To adjust, we replace the product with the minimum, which only estimates the group's density in the feature space instead of performing a likelihood computation, in the following fashion:

$$P^{\sim}_{Label}(G_{ij}) = \min(P_{Label}(o_i, o_j), P^{\sim}_{Label}(G_i), P^{\sim}_{Label}(G_j)) \tag{5.17}$$

This is also a recursive formulation allowing for reusing of values computed for subgroups during the tree construction.

These two approximations were made in the interest of constructing candidate groups (nodes of $\mathfrak{T}$) with large-scale regular patterns at interactive rates. They do not guarantee optimality of the candidate groups according to $B(G, L)$. However, $B(G, L)$ is evaluated for each group in the tree $\mathfrak{T}$ and the group with the highest value is chosen to be shown to the user. Empirical observations show, that this approach produces groups with good $B(G, L)$ at interactive rates ($\sim$1 second for constructing $\mathfrak{T}$ and evaluating $B(G, L)$ for all nodes).

Additionally, the advantage of constructing $\mathfrak{T}$ allows for a natural implementation of contraction and expansion operations. If the user rejects the group that has been selected by the algorithm, then in case of expansion request the most beneficial node above the previous one is selected. If the contraction is requested, then the most beneficial node in the subtree of the current node is chosen. To avoid slow contractions that remove one object at a time, an exponential falloff is used, i.e. subgroups are only considered if their size is less than half of the size of the group being contracted.

5.4 Results

Ultimately, the improvement that the GA interface offers to human annotators can only be evaluated via human user study, which is described later in this section. There are, however, a number of parameters that govern the time cost estimation in the benefit function of the attention model. These parameters are used in both constructing the tree $\mathfrak{T}$ and searching for the best group to select in the tree. Setting these parameters too optimistically will result in the system suggesting groups that are very unlikely to be labeled together due to the system assuming that the cost of looking at and then rejecting a group is too low to be of concern. Setting these parameters too conservatively will result in the system preventing larger groups from even appearing on $\mathfrak{T}$ due to even a tiny risk of not sharing a label being penalized by a significant cost. Deriving these constants through human user studies is prohibitively expensive. Yet, as is the case with the AL interface, the GA interface makes object selections itself. Hence with the ground truth available, the necessary estimations can be carried out using user simulations.

5.4.1 Simulated user evaluations

The parameters that heavily influence the group search are processing speed factors a_{Id} and a_{Cmd} in Equations 5.6 and 5.15, and the fastest possible time to recognize that objects share a label T_{Verify} in Equation 5.9. Normally, these coefficients in the Hick's law are derived by fitting experimental observations of a simple task to the function. However, the model used in this work consists of a set of tasks of varying complexity and fitting observed data to acquire these values is both misleading and expensive. Furthermore, the tasks the users execute are contingent on the values of these constants used by the system, creating a loop dependency. Finally, since all elements of the benefit function in Equation 5.1 that these constants contribute to

are only needed to estimate the relative benefits of showing one group over another rather than evaluating exact times, their relative values are more relevant than the absolute ones.

To acquire a simpler expression for the relationships between a_{Id}, a_{Cmd}, and T_{Verify}, Equation 5.1 can be expanded in the following manner:

$$\begin{aligned}
B(G,L) &= P_{Label}(G)T_{1\times1}(G,L) - T_{Group}(G,L) \\
&= P_{Label}(G)(T_{Id}(G,L) + T_{Cmd}(L))|G| - (T_{Id}(G,L) + T_{Verify}(G) + T_{Cmd}(L)) \\
&= (P_{Label}(G)|G| - 1)(a_{Id} + a_{Cmd})H_{eq}(|L|) \\
&\quad - T_{Verify}\Sigma_{(o_i,o_j)\in MST_{Gest}}(A_{Gest}(o_i,o_j,G) + (1 - A_{Gest})d(o_i,o_j))H(P_{Label}(o_i,o_j))
\end{aligned}$$

To clarify the point of this operation we assume the following notation:

$$\begin{aligned}
f_1(G,L) &= (P_{Label}(G)|G| - 1)H_{eq}(|L|) \\
f_2(G,L) &= \Sigma_{(o_i,o_j)\in MST_{Gest}}(A_{Gest}(o_i,o_j,G) + (1 - A_{Gest})d(o_i,o_j))H(P_{Label}(o_i,o_j)) \\
\alpha &= \frac{T_{Verify}}{a_{Id} + a_{Cmd}}
\end{aligned}$$

and $B(G,L)$ scaled by a constant factor $(a_{Id} + a_{Cmd})$ takes the following form:

$$B_{Scaled}(G,L) = f_1(G,L) - \alpha f_2(G,L) \tag{5.18}$$

Since $B_{Scaled}(G,L)$ is the same benefit function scaled by a constant factor it is interchangeable with $B(G,L)$ for both the purposes of construction of the tree $\mathfrak{T}$ and the choice of the group with highest benefit within it. It has, however, the advantage of only depending on a single important constant α that needs to be decided upon. Intuitively, smaller α leads to the relative cost of pairwise similarity verifications being smaller than that of the label identification and provision. Useful values of α within the provided formalism and $\mathfrak{T}$ construction algorithm are bounded from above by the

value of $\log 3$. This is mandated by the fact that for higher values even groups of two object that are identical by shape and are part of a perfect regular pattern will show negative benefit, preventing these groups from appearing on $\mathfrak{T}$. If groups of two cannot be formed on the tree, then no larger groups can be formed hierarchically either, resulting in all available groups consisting of single objects.

Simulated users implement the same logic that a human user is expected to perform: given a selected group either label it according to the ground truth if the group if homogeneous or contract if otherwise. Expansion logic is not implemented in this simulation for two reasons. First, unlike contraction, modeled by a simple factor of presence of an outlier, expansion is a risk-taking behavior. Modeling this behavior is outside of the scope of this work. Second, practical implications of suggested groups requiring contraction are much more important for building a useful system. Presenting smaller groups that can still be labeled has some benefit while showing larger groups with outliers is a definite waste of users' time.

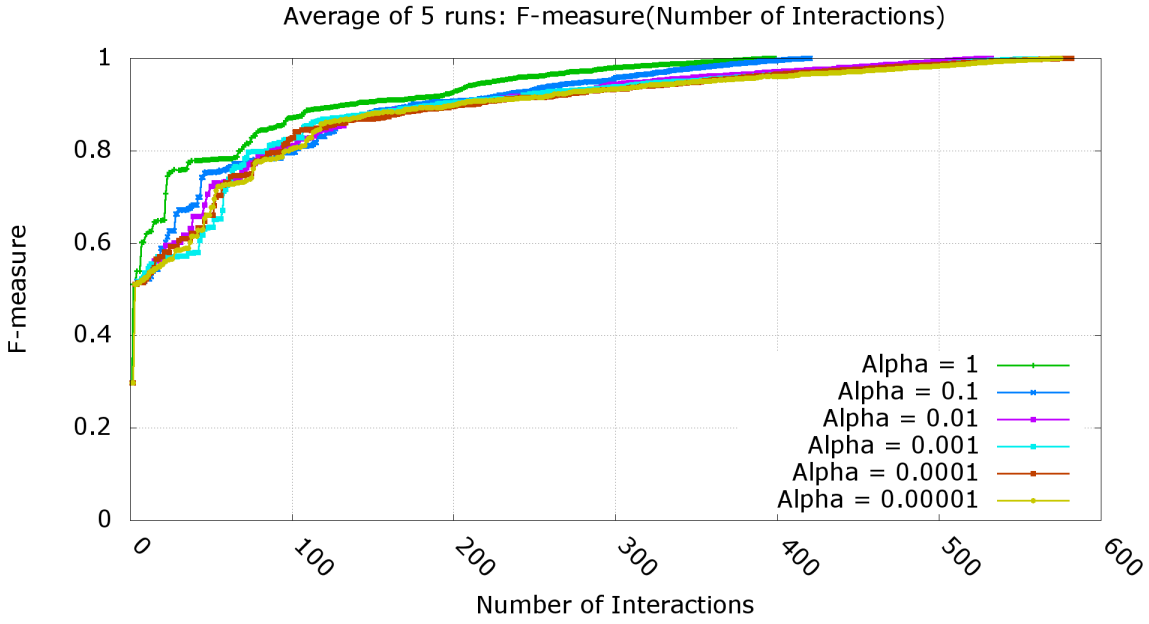


Figure 5.3: Annotation dynamic of simulated users for different α as function of number of interactions.

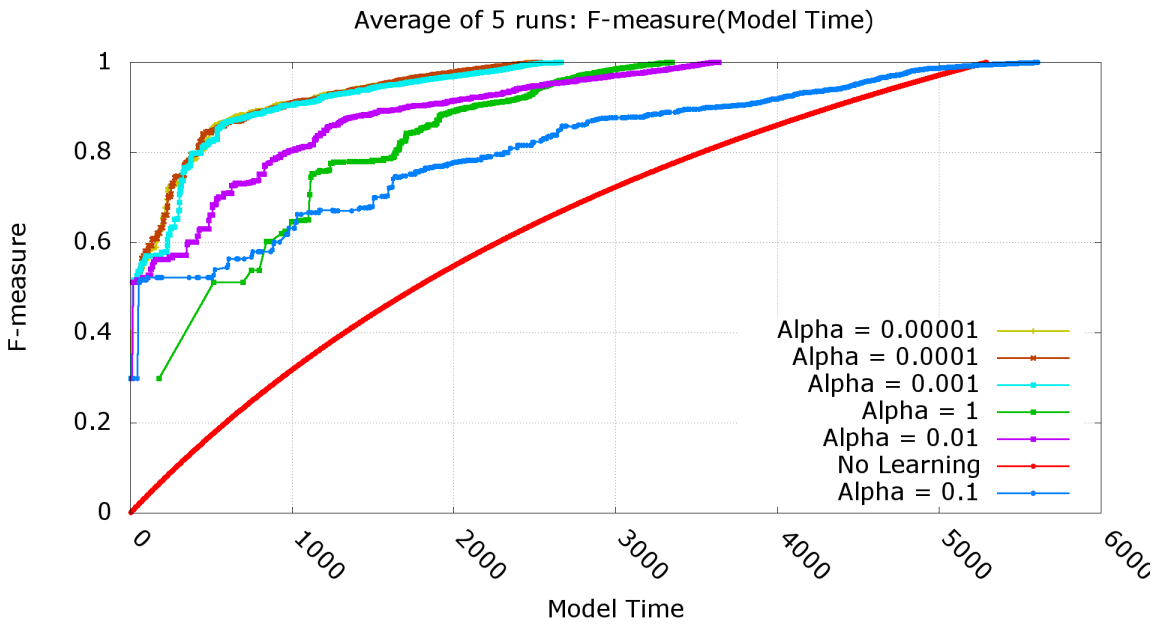


Figure 5.4: Annotation dynamic of simulated users for different α as function of model-estimated time.

Figure 5.3 shows the dynamic of the annotation process in therms of the number of interactions required by simulated users to perform. It indicates that higher α requires less user interactions. It is further supported by the interaction distribution histograms in Figures 5.5 and 5.6. For smaller α larger groups are suggested and more interactions are spent on contracting. Figure 5.4 presents the dynamic of the annotation process in terms of the model-estimated time. This is a self-referential evaluation, yet some observations about sensitivity of the model to α can be made. When α drops from 1 to 0.01 the model, although already becoming risky, still captures the fact that in the end the need for multiple contractions costs more. Yet as α drops further, the group evaluation costs are so small that the system sees no penalty for showing groups that need to be contracted most of the time. This is a wasteful behavior for an annotation system, and thus values of $\alpha = 1$ are used in the system during the human user studies.

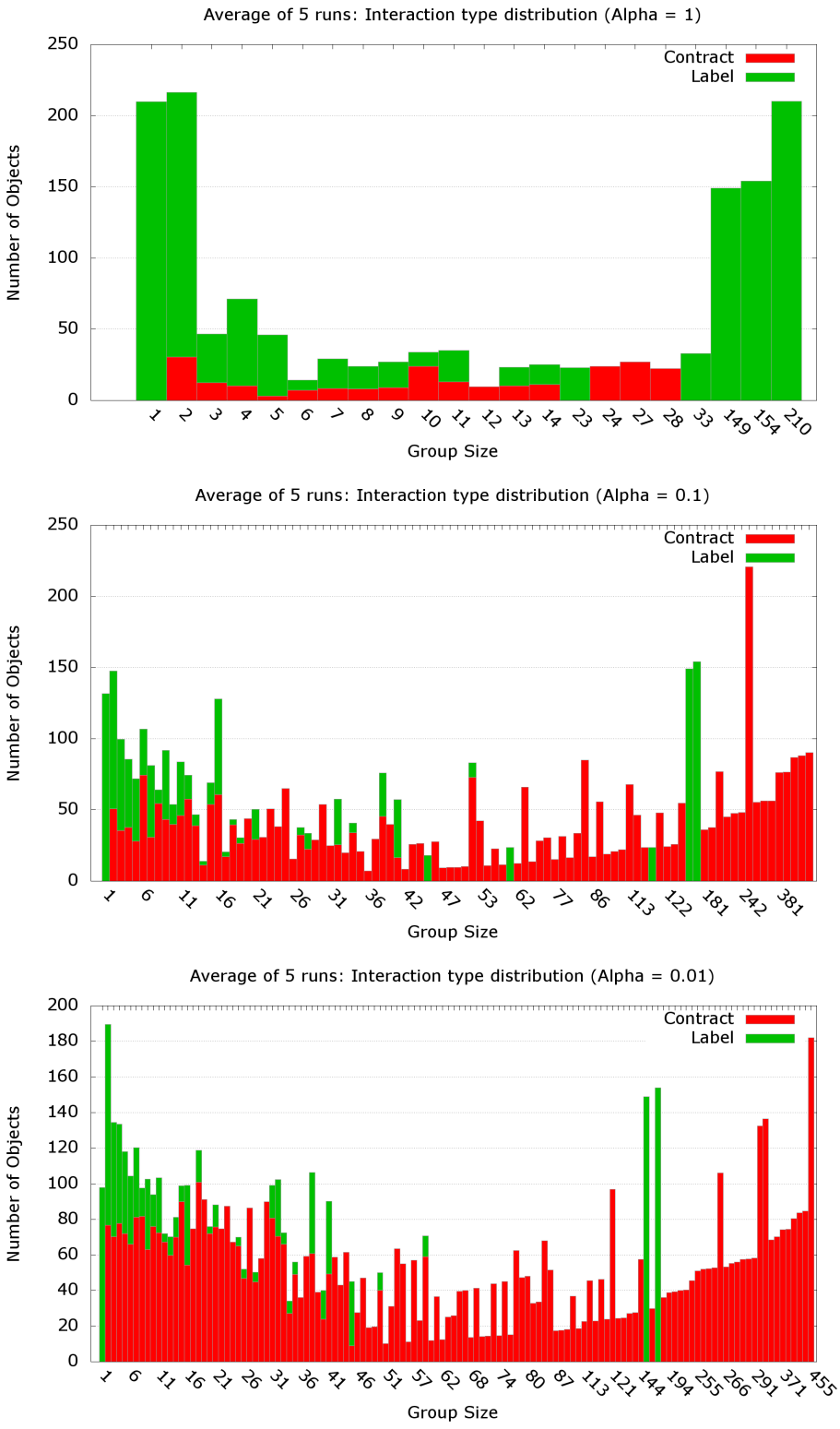


Figure 5.5: Distribution of objects that were in groups of given sizes when group was contracted or labeled for α between 1 and 0.01.

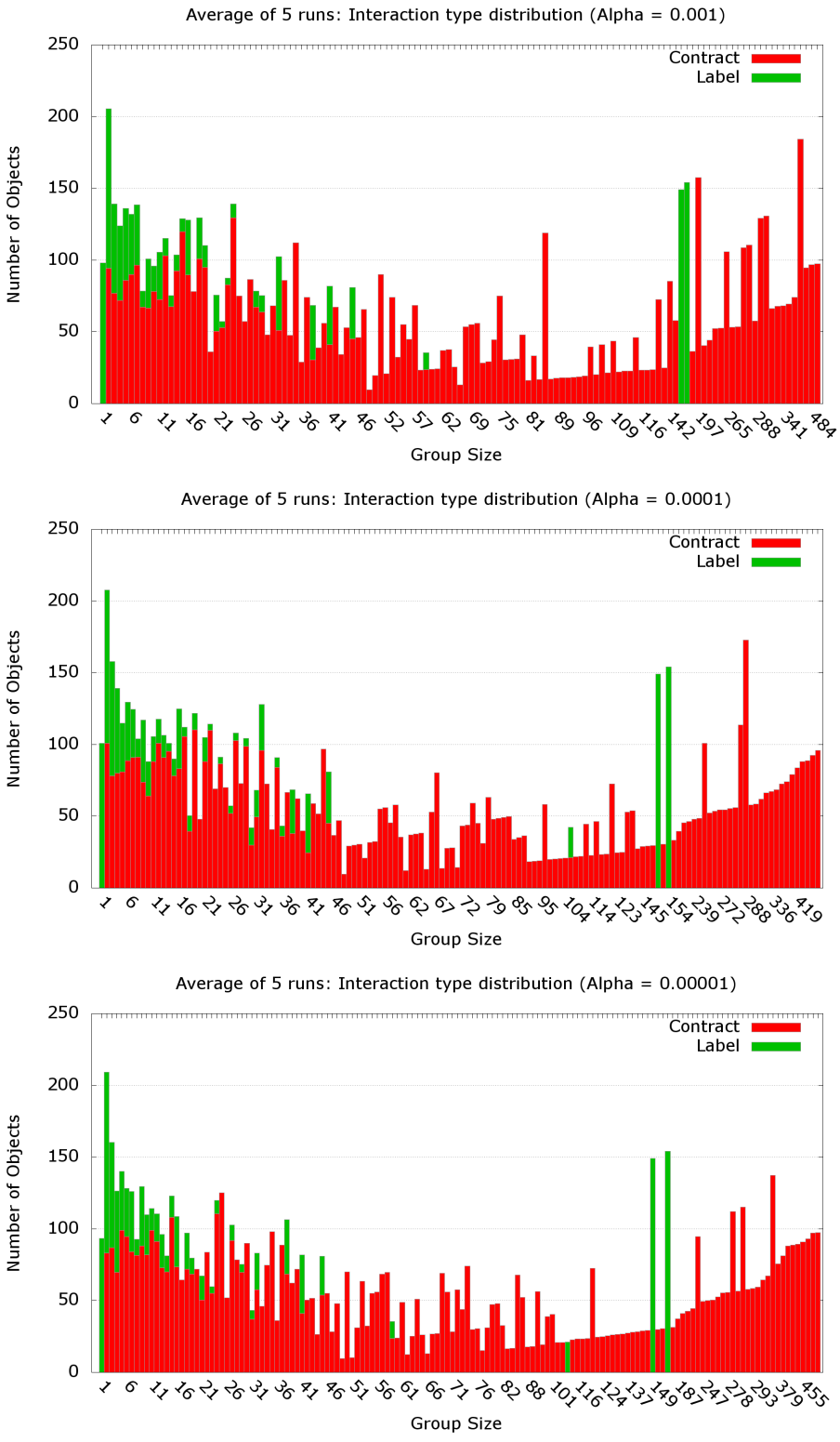


Figure 5.6: Distribution of objects that were in groups of given sizes when group was contracted or labeled for α between 10^{-3} and 10^{-5}.

5.4.2 Human user study

For user study 5 users between ages of 20 and 40, 3 females and 2 males, were recruited. The user study was performed on the same data set, using the set-up, instructions and user training similar to what is described in Section 3.3.2.

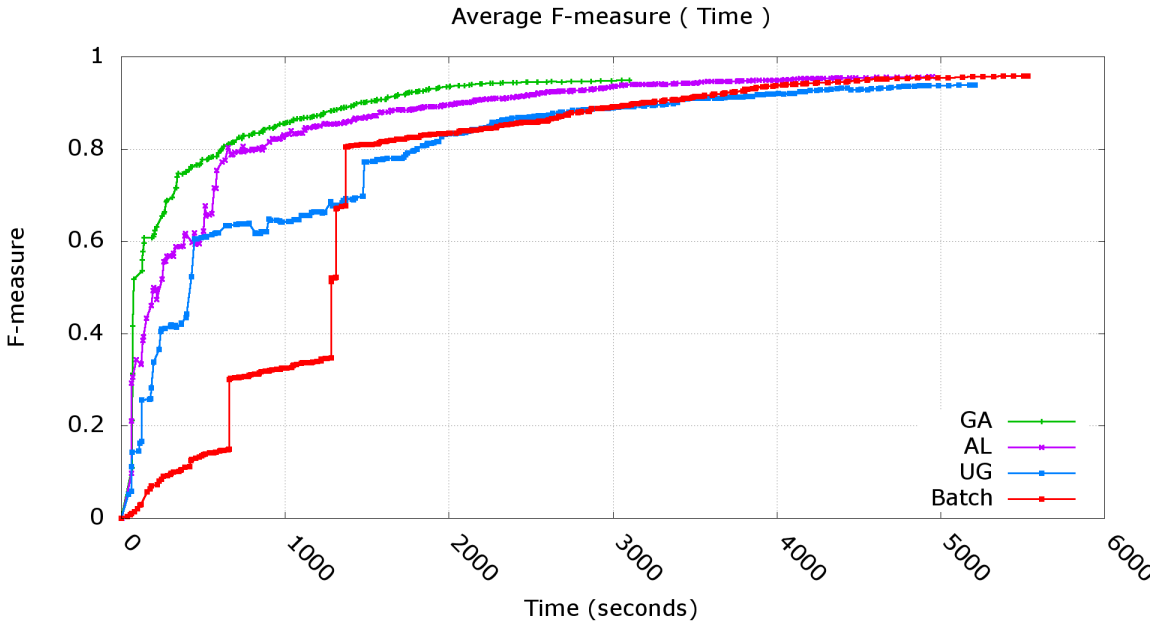


Figure 5.7: Annotation session accuracy as function of time for the GA interface (green) compared to the other interfaces.

Figure 5.7 shows a comparison of the results averaged across all five participants using the GA interface compared to the results of the users using other interfaces. This plot indicates that the GA interface provides faster times to task completion than the other interfaces. The green curve is consistently closer to the top left corner than any other curve and also terminates earlier. In particular, the average time to completion is 2281 ± 561 seconds with the GA interface. Compared to 3855 ± 837 seconds shown by the users of the closest second – AL interface – it is a difference that is statistically significant at the 1% confidence level according to an independent two-sample two-tailed t-test.

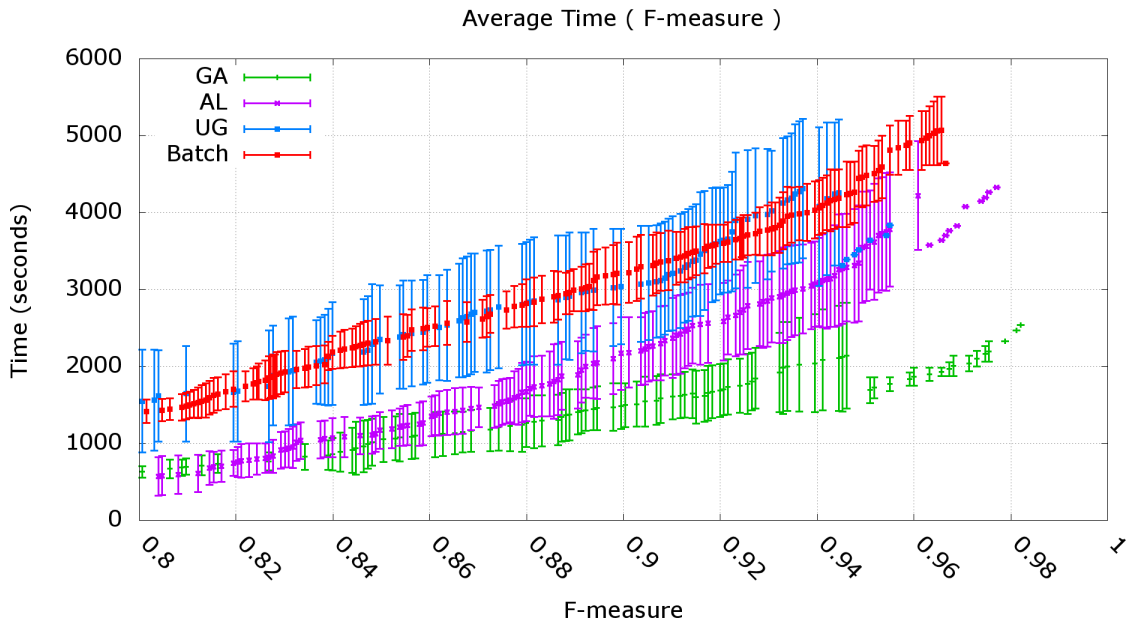


Figure 5.8: Average time required to achieve given levels of F-measure above 80% for the GA interface compared to other interfaces.

The results also indicate that the average number of labeling mistakes made by users of the GA interface is approximately the same as made with other interfaces. This can be seen from the fact that the F-measure of the rightmost point in all curves is almost the same in both plots. Specifically, users of the GA interface achieved on average F-measure of $95\% \pm 3\%$ by the end of the session, which closely overlaps with the levels achieved by the users of other interfaces. A more detailed illustration of the end of the process is given in Figure 5.8 that shows the average times for all interfaces to achieve given levels of F-measure above 80%. The tail of the process, which showed the slowest pace of accuracy improvement for other interfaces, proceeds much faster with the GA interface. Also, as the rightmost points of the green plot are further right than rightmost points of any other plot, some users of the GA interface actually achieved accuracies in excess of 98%, not attained by any of users of the competing interfaces.

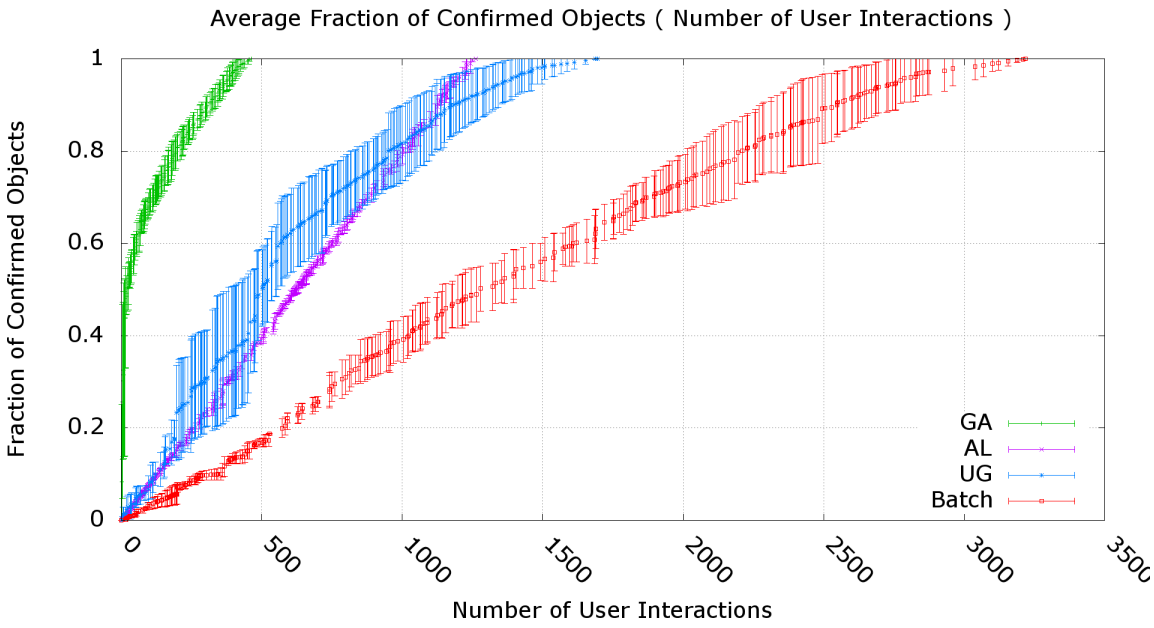


Figure 5.9: Average fraction of objects in the data set confirmed by the users of interfaces as function of number of necessary user interactions.

Further analysis of the results allows to understand what contributes to the improvement that the GA interface offers. Figure 5.9 shows the progress of confirming labels for objects by the users of different interfaces for certain number of interactions. It shows that the GA interface offers an overwhelming advantage over other interfaces. All users were capable to confirming all labels in under 500 interactions. For a data set consisting of 1224 objects it means that on average a little less than 3 labels were acquired per interaction. This is largely achieved by the fact that around 50% of the objects were confirmed in the first few interactions.

Figure 5.10 further addresses a breakdown of what type of commands users provided when presented groups of different sizes. The horizontal axis lists of the sizes of groups shown to the user. The vertical axis represents average numbers of objects in groups that were labeled/confirmed (green), contracted (red), or expanded (blue) for each group size. The plot shows that users labeled objects in groups as large as a few hundred objects at a time. For smaller groups the results show comparable

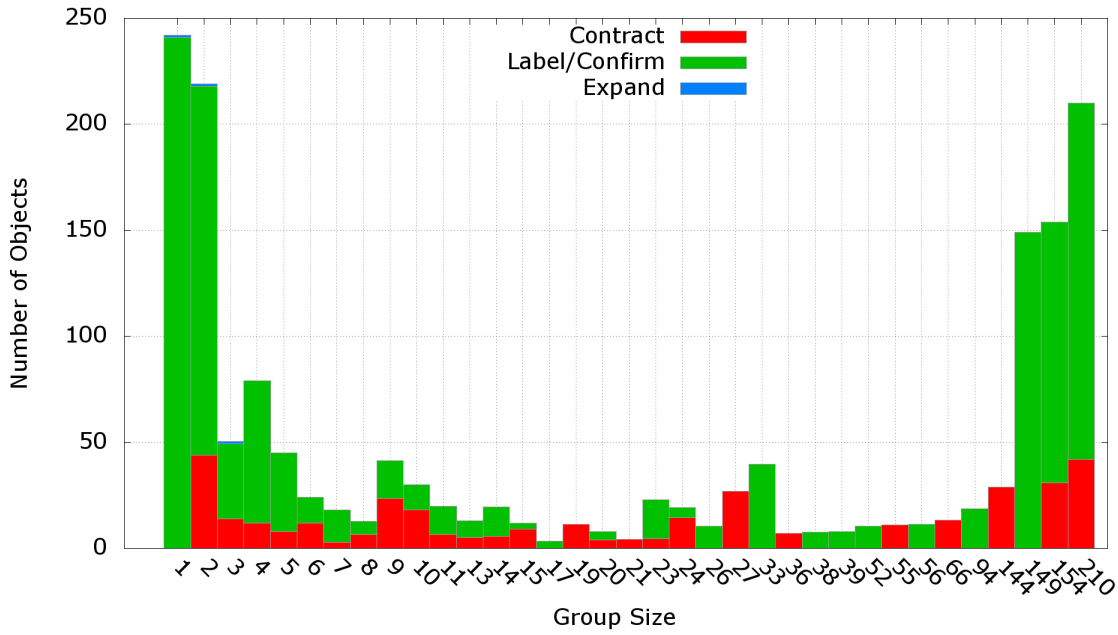


Figure 5.10: Average number of objects labeled (green), contracted (red), or expanded (blue) in groups of different sizes.

frequencies of both labeling and contraction, and closer to the lower sizes of groups users prefer labeling to contraction. According to exit surveys, the main reasons for group contraction was impurity of the group; only 1 user mentioned that it was due to inability to see the entire group in the field of view. As the left-most bar shows, only ∼20% of objects were labeled on their own, which is not surprising, since some object categories in this data set (e.g., mailboxes, garbage cans, fire hydrants, etc.) have instances scattered throughout the city, and thus are not presentable in groups. The number of objects labeled one-by-one, however, is still lower on average than that shown by the UG interface (in excess of 400 objects, Figure 3.14).

Does the size of the groups suggested affects the rate of incorrect labeling that occurs? Unlike the UG interface, where users manually assembled the groups they wanted to label and thus had an opportunity to analyze the objects as they added them, the GA interface presents the entire group to the user and the user needs to analyze it in its entirety. Figure 5.11 addresses this question by showing a breakdown

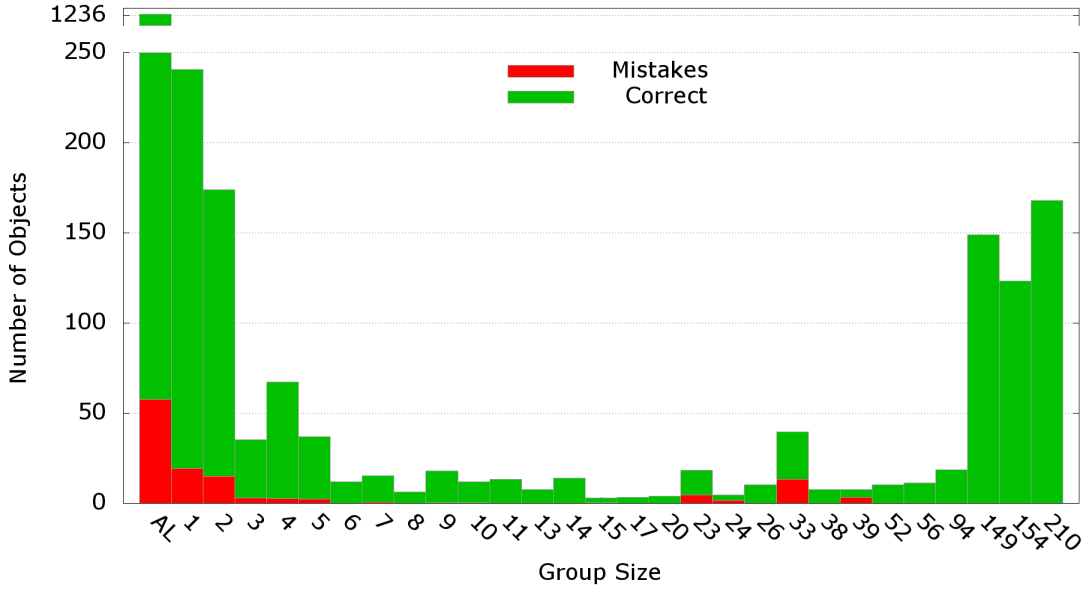


Figure 5.11: Average number of objects labeled correctly (green) or incorrectly (red) in groups of different sizes.

of correct versus incorrect labels provided by users for groups of different sizes. Again, the horizontal axis lists of the sizes of groups shown to the user. The vertical axis represents average numbers of objects in groups that were labeled correctly (green) or incorrectly (red) for each group size. The plot indicates that the fraction of mistakes is larger for smaller groups. There are two explanations for this observation. First, some objects are very difficult to recognize – so the user contracts groups containing them before providing a label, and then sometimes still makes a mistake. Second, large groups appearing in regular patterns are easier to recognize than individual objects – the pattern provides context that aids recognition.

Overall, the GA interface helps users label objects in less time and with as much accuracy as alternative interfaces do.

Chapter 6

Conclusion

6.1 Contribution

Several contributions were made in this work. First of all, unlike previous work in acquiring annotation for objects in 3D point clouds, this work asks the question of what interface tools and modes of operation allow users to provide high accuracy annotations more efficiently. Instead of focusing on models and algorithms that are geared towards annotating data sets with little user input, this work formulated the goal in a novel way: how an interface can help in acquiring completely confirmed by a human annotation in as little time as possible.

This work described and evaluated four prototype interactive systems that stress different approaches to the process of annotating small objects in 3D point clouds, three of which are introduced in this work. The Batch interface is designed following the logic of the most closely related works, which however did not describe or evaluate the properties of their specific interfaces to complete annotation. The UG (user-guided) interface takes a step in the direction of giving more information to the user and providing additional tools to empower the users to annotate more efficiently, while maintaining full control of the process. The AL (active learning) interface took

a step in a different direction, delegating most of the control over the process to the machine based on its understanding of the data and letting the user to only focus on sequential labeling as much as possible. Finally, the GA (group active) interface is a hybrid approach that still delegates most control to the machine, however, the system executes this control according to a model of how a human perceives the data.

To make the GA interface work as described a novel visual perception-based model has been proposed. This model explicitly estimates the relative complexity of the tasks involved in visual perception of groups of objects by leveraging Gestalt principles and dissecting complex tasks into microtasks, which it models as choice-reaction times using Hick's law. To employ this model in an interface at interactive rates, simplified recursive algorithms were described.

Finally, the proposed interactive systems have been evaluated in human user studies. These studies were performed on a large LiDAR point cloud with pre-segmented objects. Users were requested to perform a complete annotation, i.e. confirm a label for every object in the data set. This set-up allowed for the evaluation of the time commitments required from a human to perform an entire annotation session rather than evaluating the commitment in terms of number of annotations sufficient to achieve given accuracy of predictions achieved by a machine learning algorithm.

6.2 Discussion and future work

The results presented in this work are summarized in Table 6.1, which shows average completion times and final accuracies achieved by the users of the presented interfaces.

Interface	Completion (seconds)	Final F-measure (%)
Batch	4537 ± 743	96 ± 1
UG (User-guided)	4401 ± 787	94 ± 1
AL (Active learning)	3855 ± 837	96 ± 1
GA (Group active)	2281 ± 561	95 ± 3

Table 6.1: Average completion times and final F-measures of all interfaces.

These results show that with a right interface it is possible for novice users to label 1,224 objects with 95% accuracy in approximately 40 minutes after a few minutes of training. This is the time achieved by the users of the GA interface – that's approximately one object every 2 seconds. This is much faster than alternative 3D labeling interfaces tested. Simply providing the user with more information and interactive tools alone does not achieve a noticeable advantage because it adds to the decisions user needs to make and the interactions they need to perform. Taking the necessity of making navigation and selection decisions away from the user and allowing them to just label one object at a time requires less interactions. Yet it makes the process lengthy and tedious, requiring to confirm same labels for very similar objects over and over. Combining the simplification of the interface with intelligent grouping that allows for bulk submission of labels creates a very large improvement throughout the process.

This work provides an initial investigation and thus has many limitations that open up opportunities for future work. Firstly, the observed results are specific to the annotation of LiDAR data. However, the ideas and their implications may not be not specific to only this kind of data. Specifically, perception-based visual grouping of data for annotation is a general idea. It may be applicable to labeling any kind of data that can be presented in groups, such as cells in stained microscopic images, tu-

mors in medical images, snippets of sound. Exploring applicability of the approaches described in this work is an open question for future work.

Secondly, the proposed systems address only the problem of classification and thus evaluation is focused only on pre-segmented data. This choice is appropriate for LiDAR scans of cities because automatic segmentation algorithms can achieve 90+% precision and recall for this type of data [46]. For other types of data (e.g., Kinect scans of interior environments, biological imaging) this can be less reliable. Integration of interactive segmentation into labeling process to fix mistakes of automatic segmentation or produce new ones provides an extra challenge that needs to be addressed in future work.

Thirdly, the proposed attention model has only been evaluated with respect to it being able to serve good groups. However, a more direct evaluation of validity of the model and its underlying assumptions sets a direction for further exploration. Furthermore, the proposed model is based only on shape similarities and spatial patterns, but of course other factors (e.g., color) are important as well and should be considered in further studies.

Finally, an interesting question for future work is to investigate what other types of group annotation are most helpful. In this work the active interfaces ask a user to provide a single label for *all* objects in a group or to indicate that none is possible. Previous work has asked users to provide a single label for *any* object in a group [34] or to explicitly select outliers before providing a label [32]. Different alternatives provide different levels of information and require different amounts of time from a user. The method proposed in this work is one interesting point in this design space, but investigating other alternatives is an important topic for future work.

Bibliography

[1] George A Alvarez. Representing multiple objects as an ensemble enhances visual cognition. *Trends in cognitive sciences*, 2011.

[2] Saleema Amershi, James Fogarty, Ashish Kapoor, and Desney Tan. Overview based example selection in end user interactive concept learning. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 247–256, New York, NY, USA, 2009. ACM.

[3] Saleema Amershi, James Fogarty, Ashish Kapoor, and Desney Tan. Examining multiple potential models in end-user interactive concept learning. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 1357–1360, New York, NY, USA, 2010. ACM.

[4] Jaume Amores. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81 – 105, 2013.

[5] Dragomir Anguelov, Ben Taskar, Vassil Chatalbashev, Daphne Koller, Dinkar Gupta, Geremy Heitz, and Andrew Ng. Discriminative learning of markov random fields for segmentation of 3d scan data. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 169–176, 2005.

[6] Mihael Ankerst, Markus M. Breunig, Hans peter Kriegel, and Jrg Sander. Optics: Ordering points to identify the clustering structure. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99)*, pages 49–60. ACM Press, 1999.

[7] Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *SSD '99: Proceedings of the 6th International Symposium on Advances in Spatial Databases*, pages 207–226, London, UK, 1999. Springer-Verlag.

[8] Dan Ariely. Seeing sets: Representation by statistical properties. *Psychological Science*, 12(2):157–162, 2001.

[9] Franz Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[10] Brian P. Bailey and Joseph A. Konstan. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior*, 22(4):685 – 708, 2006.

[11] Jason Baldridge and Miles Osborne. Active learning and logarithmic opinion pools for hpsg parse selection. *Nat. Lang. Eng.*, 14(2):191–222, 2008.

[12] Holger Bekel, Ingo Bax, Gunther Heidemann, and Helge J. Ritter. Adaptive computer vision: Online learning for object recognition. In C. E. Rasmussen and et al, editors, *Proc. DAGM 2004*, pages 447–454. Springer, Springer, 2004.

[13] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 49–56. ACM, 2009.

[14] A. Borji and L. Itti. State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):185–207, 2013.

[15] Aleksey Boyko and Thomas Funkhouser. Cheaper by the dozen: Group annotation of 3d data. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 33–42, New York, NY, USA, 2014. ACM.

[16] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In *Proc. ECCV*, pages 438–451, 2010.

[17] M. Bredif, D. Boldo, M. Pierrot-Deseilligny, and H. Maitre. 3d building reconstruction with parametric roof superstructures. *International Conference on Image Processing, 2007*, 2:537–540, 2007.

[18] Razvan C. Bunescu and Raymond J. Mooney. Multiple instance learning for sparse positive bags. In *Proc. ICML*, 2007.

[19] Matthew Carlberg, James Andrews, Peiran Gao, and Avideh Zakhor. Fast surface reconstruction and segmentation with ground-based and airborne lidar range data. *3DPVT*, 2008.

[20] Jamie Carter, Keil Schmid, Kirk Waters, Lindy Betzhold, Brian Hadley, Rebecca Mataosky, and Jennifer Halleran. Lidar 101: An Introduction to Lidar Technology, Data, and Applications. Technical report, National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center, 2012. `http://lidar.cr.usgs.gov`. Accessed: 2014-10-29.

[21] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[22] Olivier Chapelle, Vladimir Vapnik, and Jason Weston. Transductive inference for estimating values of functions. In *NIPS*, pages 421–427, 1999.

[23] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In *Proc. 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

[24] Jie Chen and Baoquan Chen. Architectural modeling from sparsely scanned range data. *Int. J. Comput. Vision*, 78(2-3):223–236, 2008.

[25] Sang Chol Chong and Anne Treisman. Representation of statistical properties. *Vision Research*, 43(4):393–404, 2003.

[26] SangChul Chong and Anne Treisman. Attentional spread in the statistical processing of visual displays. *Perception & Psychophysics*, 67(1):1–13, 2005.

[27] Marc Christie, Rumesh Machap, Jean-Marie Normand, Patrick Olivier, and Jonathan Pickering. Virtual camera planning: A survey. In Andreas Butz, Brian D. Fisher, Antonio Krüger, and Patrick Olivier, editors, *Smart Graphics*, volume 3638 of *Lecture Notes in Computer Science*, pages 40–52. Springer, 2005.

[28] S. Clode, P. Kootsookos, and F. Rottensteiner. The automatic extraction of roads from lidar data. In *Proceedings of American Society of Photogrammetry and Remote Sensing*, volume 35, page 231237. ASPRS, May 2004.

[29] Brendan Collins, Jia Deng, Kai Li, and Li Fei-Fei. Towards scalable dataset construction: An active learning approach. In *Proc. ECCV*, volume 5302, 2008.

[30] Padraig Cunningham and Sarah Jane Delany. k-nearest neighbour classifiers. *Multiple Classifier Systems*, pages 1–17, 2007.

[31] A Delorme, G Richard, and M Fabre-Thorpe. Ultra-rapid categorisation of natural scenes does not rely on colour cues: a study in monkeys and humans. *Vision Research*, 40(16):2187 – 2200, 2000.

[32] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, pages 248–255, 2009.

[33] Jia Deng, Jonathan Krause, and Li Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *Proc. CVPR*, 2013.

[34] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[35] Peter Dorninger and Norbert Pfeifer. A comprehensive automated 3d approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds. *Sensors*, 8(11):7323–7343, 2008.

[36] Kun Duan, D. Parikh, D. Crandall, and K. Grauman. Discovering localized attributes for fine-grained recognition. In *CVPR*, pages 3474–3481, 2012.

[37] S. Ebert, M. Fritz, and B. Schiele. Ralf: A reinforced active learning formulation for object class recognition. In *Proc. CVPR*, 2012.

[38] Martin Ester, Hans peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[39] Jerry Alan Fails and Dan R. Olsen, Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, pages 39–45, New York, NY, USA, 2003. ACM.

[40] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. Cueflik: interactive concept learning in image search. In *Proc. SIGCHI*, pages 29–38, 2008.

[41] C. Frueh, S. Jain, and A. Zakhor. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *International Journal of Computer Vision, 2005*, 61(2):159–184, February 2005.

[42] Yifan Fu, Xingquan Zhu, and Bin Li. A survey on instance selection for active learning. *Knowledge and Information Systems*, pages 1–35, 2012.

[43] A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. In *Fourteenth conference on Uncertainty in artificial intelligence*, 1998.

[44] Caroline Gasperin. Active learning for anaphora resolution. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 1–8. Association for Computational Linguistics, 2009.

[45] Aleksey Golovinskiy and Thomas Funkhouser. Min-cut based segmentation of point clouds. In *IEEE Workshop on Search in 3D and Video (S3DV) at ICCV*, sep 2009.

[46] Aleksey Golovinskiy, Vladimir G. Kim, and Thomas Funkhouser. Shape-based recognition of 3D point clouds in urban environments. *International Conference on Computer Vision (ICCV)*, September 2009.

[47] Leonardo Gomes, Olga Regina Pereira Bellon, and Luciano Silva. 3d reconstruction methods for digital preservation of cultural heritage: A survey. *Pattern Recognition Letters*, 50(0):3 – 14, 2014. Depth Image Analysis.

[48] Antuan Goodwin. Nokia, Navteq show us how a map is made. `http://www.cnet.com/news/nokia-navteq-show-us-how-a-map-is-made`, November 2012. Accessed: 2014-10-29.

[49] Russell Greiner, Adam J. Grove, and Dan Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2):137 – 174, 2002.

[50] Yuhong Guo and Dale Schuurmans. Discriminative batch mode active learning. In *Advances in neural information processing systems*, pages 593–600, 2008.

[51] Jason Haberman and David Whitney. The visual system discounts emotional deviants when extracting average expression. *Attention, Perception, & Psychophysics*, 72(7):1825–1838, 2010.

[52] Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll, and Peter McClanahan. Assessing the costs of sampling methods in active learning for annotation. In *HLT-Short*, pages 65–68, 2008.

[53] Rob Harrap and Matt Lato. An Overview of LIDAR: collection to applications. `http://www.ngi.no/upload/48594/1%20What%20Is%20Lidar.pdf`, 2010. Accessed: 2014-10-29.

[54] Carsten Hatger and Claus Brenner. Extraction of road geometry parameters from laser scanning and existing databases. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34 (Part 3/W13):225–230, 2003.

[55] G. Heidemann, H. Bekel, I. Bax, and H. Ritter. Interactive online learning. *Pattern Recognition and Image Analysis*, 17:146–152, 2007. 10.1134/S105466180701018X.

[56] A. van den Hengel, A. Dick, T. Thormaehlen, P. Torr, and B. Ward. Building models of regular scenes from structure and motion. In *Proc. BMVC*, 2006.

[57] William E Hick. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4(1):11–26, 1952.

[58] M. Himmelsbach, T. Luettel, and H.-J. Wuensche. Real-time object classification in 3d point clouds using point feature histograms. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pages 994–1000, Piscataway, NJ, USA, 2009. IEEE Press.

[59] Huy Tho Ho. *3D Surface Matching from Range Images using Multiscale Local Features*. PhD thesis, The University of Adelaide Australia, 2009.

[60] L. Itti and C. Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, 2001.

[61] Anttoni Jaakkola, Juha Hyyppa, Hannu Hyyppa, and Antero Kukko. Retrieval algorithms for road surface modelling using laser-based mobile mapping. *Sensors*, 8(9):5238–5249, 2008.

[62] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE PAMI*, 21:433–449, 1999.

[63] Ajay J. Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *Proc. CVPR*, 2009.

[64] Ajay J. Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Breaking the interactive bottleneck in multi-class classification with active selection and binary feedback. In *Proc. CVPR*, pages 2995–3002, 2010.

[65] Wolfgang Juchmann and Ken Greenberg. Velodynes HDL-32E 3D LiDAR Sensor Spins on Multiple Mobile Mapping Systems at InterGEO in Berlin. `http://www.prweb.com/releases/2014/10/prweb12259559.htm`, October 2014. Accessed: 2014-10-29.

[66] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *Proc. ICCV*, 2007.

[67] Ashish Kapoor, Eric Horvitz, and Sumit Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *IJCAI*, pages 877–882, 2007.

[68] Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning Part-based Templates from Large Collections of 3D Shapes. *Transactions on Graphics (Proc. of SIGGRAPH)*, 32(4), 2013.

[69] Craig W. Kirkwood. *Decision Tree Primer*, chapter 3: The Value of Information. Arizona State University, 2002. `http://www.public.asu.edu/~kirkwood/DAStuff/decisiontrees/DecisionTreePrimer-3.pdf`.

[70] Adriana Kovashka, Devi Parikh, and Kristen Grauman. Whittlesearch: Image search with relative attribute feedback. In *Proc. CVPR*, 2012.

[71] Hans-Peter Kriegel, Peer Krger, Jrg Sander, and Arthur Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[72] F. Lafarge, X. Descombes, J. Zerubia, and M. Pierrot-Deseilligny. Building reconstruction from a single dem. *Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[73] Jean-Francois Lalonde, Ranjith Unnikrishnan, Nicolas Vandapel, and Martial Hebert. Scale selection for classification of point-sampled 3-d surfaces. In *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM 2005)*, pages 285 – 292, June 2005.

[74] Fei Fei Li, Rufin VanRullen, Christof Koch, and Pietro Perona. Rapid natural scene categorization in the near absence of attention. *PNAS*, 99(14):9596–9601, 2002.

[75] Hui Lin, Jizhou Gao, Yu Zhou, Guiliang Lu, Mao Ye, Chenxi Zhang, Ligang Liu, and Ruigang Yang. Semantic decomposition and reconstruction of residential scenes from lidar data. *ACM Transactions on Graphics, (Proc. of SIGGRAPH 2013)*, 32(4), 2013.

[76] Dong Liu, Xian-Sheng Hua, Linjun Yang, and Hong-Jiang Zhang. Multiple-instance active learning for image categorization. In *Proc. MMM*, volume 5371, pages 239–249, 2009.

[77] Oded Maron and Aparna Lakshmi Ratan. Multiple-instance learning for natural scene classification. In *Proc. ICML*, pages 341–349, 1998.

[78] Daniel Munoz, J. Andrew Bagnell, Nicolas Vandapel, and Martial Hebert. Contextual classification with functional max-margin markov networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[79] Daniel Munoz, Nicolas Vandapel, and Martial Hebert. Directional associative markov network for 3-d point cloud classification. In *Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, June 2008.

[80] Przemyslaw Musialski, Peter Wonka, Daniel G Aliaga, Michael Wimmer, Luc van Gool, and Werner Purgathofer. A survey of urban reconstruction. In *EUROGRAPHICS 2012 State of the Art Reports*, pages 1–28. Eurographics Association, 2012.

[81] Liangliang Nan, Andrei Sharf, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Smartboxes for interactive urban reconstruction. In *Proc. SIGGRAPH*, pages 93:1–93:10. ACM, 2010.

[82] Neptec. Wright state 100. `http://www.ws-arc.org/applied-research-corporation/ottawa-data-files.html`, 2009. Accessed: April, 2014.

[83] J Farley Norman, James T Todd, and Guy A Orban. Perception of three-dimensional shape from specular highlights, deformations of shading, and other types of visual information. *Psychological Science*, 15(8):565–570, 2004.

[84] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, C.-C. Chen, Jong Taek Lee, Saurajit Mukherjee, J. K. Aggarwal, Hyungtae Lee, Larry Davis, Eran Swears, Xioyang Wang, Qiang Ji, Kishore Reddy, Mubarak Shah, Carl Vondrick, Hamed Pirsiavash, Deva Ramanan, Jenny Yuen, Antonio Torralba, Bi Song, Anesco Fong, Amit Roy-Chowdhury, and Mita Desai. A large-scale benchmark dataset for event recognition in surveillance video. In *Proc. CVPR*, 2011.

[85] Aude Oliva and Antonio Torralba. The role of context in object recognition. *Trends in Cognitive Sciences*, 11(12):520 – 527, 2007.

[86] Mathias Ortner, Xavier Descombes, and Josiane Zerubia. Building outline extraction from digital elevation models using marked point processes. *Int. J. Comput. Vision*, 72(2):107–132, 2007.

[87] D. Parikh and K. Grauman. Interactively building a discriminative vocabulary of nameable attributes. In *CVPR*, pages 1681–1688, June 2011.

[88] A.I. Patterson, P. Mordohai, and K. Daniilidis. Object detection from large-scale 3D datasets using bottom-up andtop-down descriptors. In *Proc. ECCV*, pages IV: 553–566, 2008.

[89] D.G. Pelli, N.J. Majaj, C.J. Christian, E. Kim, and M.C. Palomares. Grouping in object recognition: the role of a Gestalt law in letter identification. *Cognitive Neuropsychology*, 26(1):36–49, February 2009.

[90] Zygmunt Pizlo. *3D shape: its unique place in visual perception.* Mit Press, 2010.

[91] D.V. Prokhorov. Object recognition in 3d lidar data with recurrent neural network. In *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pages 9 –15, june 2009.

[92] Ronald A. Rensink. The dynamic representation of scenes. *Visual Cognition*, 7(1-3):17–42, 2000.

[93] Ruth Rosenholtz. A simple saliency model predicts a number of motion popout phenomena. *Vision Research*, 39(19):3157 – 3163, 1999.

[94] Bryan C. Russell and Antonio Torralba. Building a database of 3d scenes from user annotations. In *Proc. CVPR*, pages 2711–2718, 2009.

[95] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Labelme: A database and web-based tool for image annotation. *IJCV*, 77(1-3):157–173, 2008.

[96] Ian Ruthven and Mounia Lalmas. A survey on the use of relevance feedback for information access systems. *Knowl. Eng. Rev.*, 18:95–145, June 2003.

[97] Thomas Sanocki and William Epstein. Priming spatial layout of scenes. *Psychological Science*, 8(5):374–378, 1997.

[98] Ruwen Schnabel, Roland Wahl, Raoul Wessel, and Reinhard Klein. Shape recognition in 3d point clouds. *The 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2008.

[99] Barry Schwartz. *The Paradox of Choice: Why More Is Less.* Harper Perennial, January 2005.

[100] B. Settles, M. Craven, and L. Friedland. Active learning with real annotation costs. In *NIPS*, 2008.

[101] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. `pages.cs.wisc.edu/~bsettles/active-learning`.

[102] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[103] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Proc. NIPS*, pages 1289–1296. MIT Press, 2008.

[104] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Trans. Graph.*, 31(6):136:1–136:11, 2012.

[105] Michael Shilman, Desney S. Tan, and Patrice Simard. Cuetip: a mixed-initiative interface for correcting handwriting errors. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 323–332, New York, NY, USA, 2006. ACM.

[106] B. Siddiquie and A. Gupta. Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *CVPR*, pages 2979–2986, 2010.

[107] Marc Simard. 3D Land Mapping. `http://lidarradar.jpl.nasa.gov`. Accessed: 2014-10-29.

[108] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. Beyond the point cloud: From transductive to semi-supervised learning. In *Proc. ICML*, pages 824–831, 2005.

[109] Hao Su, Jia Deng, and Li Fei-Fei. Crowdsourcing annotations for visual object detection. In *Proc. HCOMP*, 2012.

[110] David Suter and EeHui Lim. Conditional random field for 3d point clouds with adaptive data reduction. *Cyberworlds*, 2007.

[111] S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381:520, 1996.

[112] Antonio Torralba, Bryan C. Russell, and Jenny Yuen. Labelme: Online image annotation and applications. *Proc. IEEE*, 98(8):1467–1484, 2010.

[113] D. Tuia, E. Pasolli, and W.J. Emery. Using active learning to adapt remote sensing image classifiers. *Remote Sensing of Environment*, 115(9), 2011.

[114] USGS. USGS Center for LIDAR Information Coordination and Knowledge. http://lidar.cr.usgs.gov. Accessed: 2014-10-29.

[115] Brian Ussery. Google Maps Street View 2.0 LiDAR. https://www.beussery.com/blog/index.php/2010/04/google-streetview-lidar, April 2010. Accessed: 2014-10-29.

[116] A. Velizhev, R. Shapovalov, and K. Schindler. Implicit shape models for object detection in 3D point clouds. *ISPRS Annals*, I-3(2012):179–184, 2012.

[117] Frans A.J Verstraten, Patrick Cavanagh, and Angela T Labianca. Limits of attentive tracking reveal temporal properties of attention. *Vision Research*, 40(26):3651 – 3664, 2000.

[118] S. Vijayanarasimhan and K. Grauman. Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization. In *Proc. CVPR*, pages 1–8, 2008.

[119] S. Vijayanarasimhan and K. Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *Proc. CVPR*, pages 2262–2269, June 2009.

[120] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. In *Proc. CVPR*, 2011.

[121] S. Vijayanarasimhan, P. Jain, and K. Grauman. Far-sighted active learning on a budget for image and video recognition. In *Proc. CVPR*, pages 3035–3042, 2010.

[122] Sudheendra Vijayanarasimhan and Kristen Grauman. Multi-level active prediction of useful image annotations for recognition. In *Proc. NIPS*, pages 1705–1712, 2008.

[123] Sudheendra Vijayanarasimhan and Kristen Grauman. Cost-sensitive active visual category learning. *Int. J. Comput. Vision*, 91(1):24–44, January 2011.

[124] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proc. SIGCHI*, pages 319–326, 2004.

[125] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *IJCV*, 101(1):184–204, 2013.

[126] Carl Vondrick and Deva Ramanan. Video Annotation and Tracking with Active Learning. In *NIPS*, 2011.

[127] Carl Vondrick, Deva Ramanan, and Donald Patterson. Efficiently scaling up video annotation with crowdsourced marketplaces. In *Proc. ECCV*, 2010.

[128] Johan Wagemans, James H Elder, Michael Kubovy, Stephen E Palmer, Mary A Peterson, Manish Singh, and Rüdiger von der Heydt. A century of gestalt psychology in visual perception: I. perceptual grouping and figure–ground organization. *Psychological bulletin*, 138(6):1172, 2012.

[129] Johan Wagemans, Jacob Feldman, Sergei Gepshtein, Ruth Kimchi, James R Pomerantz, Peter A van der Helm, and Cees van Leeuwen. A century of gestalt psychology in visual perception: II. conceptual and theoretical foundations. *Psychological bulletin*, 138(6):1218, 2012.

[130] Yunhai Wang, Shmulik Asafi, Oliver van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Trans. Graph.*, 31(6):165:1–165:10, 2012.

[131] Yunsheng Wang, Holger Weinacker, and Barbara Koch. A lidar point cloud based procedure for vertical canopy structure analysis and 3d single tree modelling in forest. *Sensors*, 8:1424–8220, 2008.

[132] William J. Welch. Algorithmic complexity: Three NP-hard problems in computational statistics. *Journal of Statistical Computation and Simulation*, 15:17–25, 1982.

[133] Ryan Whitwam. How Googles self-driving cars detect and avoid obstacles. `http://www.extremetech.com/extreme/189486-how-googles-self-driving-cars-detect-and-avoid-obstacles`, September 2014. Accessed: 2014-10-29.

[134] Denis F. Wolf, Gaurav S. Sukhatme, Dieter Fox, and Wolfram Burgard. Autonomous terrain mapping and classification using hidden markov models. In *IEEE International Conference on Robotics and Automation*, pages 2038–2043, April 2005.

[135] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*, 2010.

[136] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *Proc. ICCV*, 2013.

[137] Hui Xu, Nathan Gossett, and Baoquan Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Graph.*, 26(4):19, 2007.

[138] Rong Yan, Jie Yang, and A. Hauptmann. Automatically labeling video data using multi-class active learning. In *Proc. ICCV*, 2003.

[139] C. Yang and T. Lozano-Perez. Image database retrieval with multiple-instance learning techniques. In *Proc. Conf. on Data Engineering*, pages 233–243, 2000.

[140] Jun Yang. Review of multi-instance learning and its applications. Technical report, School of Computer Science, Carnegie Mellon University, 2005.

[141] Angela Yao, Juergen Gall, Christian Leistner, and Luc J. Van Gool. Interactive object detection. In *Proc. CVPR*, pages 3242–3249, 2012.

[142] Sijie Yu, Sreenivas R. Sukumar, Andreas F. Koschan, and Mongi A. Abidi. 3d reconstruction of road surfaces using an integrated multi-sensory. *Optics and Lasers in Engineering*, 45 (7):808–818, February 2007.

[143] H. Zhang, J.E. Fritts, and S.A. Goldman. Image segmentation evaluation: A survey of unsupervised methods. *CVIU*, 110(2):260–280, May 2008.

[144] Zhi-Hua Zhou. Multi-instance learning: A survey. Technical report, AI Lab, Department of Computer Science &Technology, Nanjing University, 2004.

[145] Zhi-Hua Zhou and Min-Ling Zhang. Multi-instance multi-label learning with application to scene classification. In *Proc. NIPS*, pages 1609–1616, 2007.

[146] Qingsheng Zhu and Peng Yang. Density sensitive based spectral clustering. In *Computer and Information Sciences*, volume 62, pages 139–142. Springer, 2010.