ABSTRACT

Title of dissertation: FINDING OBJECTS IN COMPLEX SCENES

Jin Sun
Doctor of Philosophy, 2018

Dissertation directed by: Professor David Jacobs
Department of Computer Science

Object detection is one of the fundamental problems in computer vision that has great practical impact. Current object detectors work well under certain conditions. However, challenges arise when scenes become more complex. Scenes are often cluttered and object detectors trained on Internet collected data fail when there are large variations in objects' appearance.

We believe the key to tackle those challenges is to understand the rich context of objects in scenes, which includes: the appearance variations of an object due to viewpoint and lighting condition changes; the relationships between objects and their typical environment; and the composition of multiple objects in the same scene. This dissertation aims to study the complexity of scenes from those aspects.

To facilitate collecting training data with large variations, we design a novel user interface, ARLabeler, utilizing the power of Augmented Reality (AR) devices. Instead of labeling images from the Internet passively, we put an observer in the real world with full control over the scene complexities. Users walk around freely and observe objects from multiple angles. Lighting can be adjusted. Objects can be added and/or removed to the scene to create rich compositions. Our tool opens new possibilities to prepare data for complex scenes.

We also study challenges in deploying object detectors in real world scenes: detecting curb ramps in street view images. A system, Tohme, is proposed to combine detection results from detectors and human crowdsourcing verifications. One core component is a meta-classifier that estimates the complexity of a scene and assigns it to human (accurate but costly) or computer (low cost but error-prone) accordingly.

One of the insights from Tohme is that context is crucial in detecting objects. To understand the complex relationship between objects and their environment, we propose a standalone context model that predicts where an object can occur in an image. By combining this model with object detection, it can find regions where an object is missing. It can also be used to find out-of-context objects.

To take a step beyond single object based detections, we explicitly model the geometrical relationships between groups of objects and use the layout information to represent scenes as a whole. We show that such a strategy is useful in retrieving indoor furniture scenes with natural language inputs.

FINDING OBJECTS IN COMPLEX SCENES

by

Jin Sun

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2018

Acknowledgments

First and foremost I would like to thank my advisor, Professor David Jacobs for all his help and support during my PhD study. He cares so much about my research and is always ready to provide assistance for the best of my interest. His enthusiasm for solving fundamental problems in computer vision has greatly influenced my own research interest and attitude. He guided me step-by-step on how to become a responsible and independent researcher. I enjoyed meeting with him every week to talk about life, research, and everything. It has been a great pleasure working with and learning from him. This dissertation would not have been possible without his encouragement and endless patience.

I am grateful to my dissertation committee members: Professors Larry Davis, Rama Chellappa, Matthias Zwicker, and Yiannis Aloimonos, for their insightful comments and feedback on this dissertation. I also thank Dr. Kumar Sricharan, Dr. Raja Bala, and Dr. Matthew Shreve for their supervision and help during my internship at the Palo Alto Research Center (PARC).

I would like to thank Professor Jon Froehlich for his invaluable advices on my project and how to write a good scientific paper. I appreciate and enjoyed collaborations with my fellow student colleagues: Kotaro Hara, Ang Li, Joe Ng, and Hao Zhou.

I owe my thanks to my group mates for all the helpful, inspiring, and interesting discussions and conversations over the years: Angjoo Kanazawa, Soumyadip Sengupta, Abhay Yadav, Joao Soares, and Arijit Biswas. I also thank my other friends in the A.V. Williams Building including: Hao Li, Hui Miao, Hui Ding, Heng Zhang, and Jun-Cheng Chen, for all the great times we have shared together.

Last but not least, I am deeply thankful to my family for always believing in me, for their love, unconditional support, and sacrifice.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AABB	Axis Aligned Bounding Box
AR	Augmented Reality
BRN	Bounding box Refinement Network
COCO	Common Objects in Context
CNN	Convolutional Neural Network
CV	Computer Vision
DNN	Deep Neural Network
DPM	Deformable Parts Model
FN	False Negative
FP	False Positive
FPS	Frames Per Second
GSV	Google Street View
HIT	Human Intelligence Task
IOU	Intersection-Over-Union
NAABB	Non-Axis Aligned Bounding Box
RCNN	Regions with Convolutional Neural Network
SD	Standard Deviation
SFC	Siamese trained Fully convolutional Context
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machine
TP	True Positive
VGG	Visual Geometry Group
VIA	VGG Image Annotator
VOC	Visual Object Class
YOLO	You Only Look Once detector

Chapter 1: Introduction

1.1 Background

Computer vision is changing the world. With the ever growing enormous amount of visual data available (e.g., Google Street View, Flickr, and Instagram), new powerful dedicated hardware (e.g., Nvidia GPUs and Intel Deep Learning Chips), and evolving machine learning algorithms with highly nonlinear characteristics (Deep Neural Networks), we are standing at the dawn of a new era in which intelligent systems will substantially improve the quality of life for society. For example, self driving cars [1], which could reduce traffic and save lives once they hit the mass market, use object detection, semantic segmentation and 3D reconstruction algorithms from computer vision; grocery stores use facial recognition and tracking algorithms to enable a cashier-free shopping experience; segmentation and classification algorithms are used in medical imaging to diagnose diseases.

As one of the fundamental problems in computer vision, *Object Detection* attracts a lot of research attention and has great practical impact. The task is to find the positions of all objects of interest in input images. Usually, object locations are represented by rectangular boxes that tightly fit the objects: for example SSD [2] and YOLO [3]. Another choice is to produce a pixel level probability map of where an object is; for example human part detection often uses heat maps to account for uncertainties in predicting different parts.

Object detection is an essential building block in many computer vision applications. Once the location of an object is known, more sophisticated analysis can be performed on the identified object regions such as person identification or attributes analysis. Because such analysis takes more computational resources, it is crucial to reduce the computational load by focusing on true object locations. Being able to efficiently perform object detection is also important, especially for time-sensitive applications such as obstacle detection on self driving cars.

The main goal of this dissertation is to investigate and develop algorithms to improve object detection in complex scenes.

1.2 Challenges in Real-World Object Detection

Although extensively studied, object detection is still a difficult task in complex real-world scenes for the following reasons.

Standard object detection datasets such as PASCAL VOC and COCO collect images from the Internet. General purpose object detectors that are trained on those data are assumed to be useful in real world scenes. However, those detectors hardly work right out of the box for objects with arbitrary viewing angles and lighting conditions. There is a reality gap in terms of the objects **appearance variations** between images collected from the Internet and test images in real world applications. For example, there barely exists any back-view images of computer monitors in the PASCAL VOC data, while in a real use case, a detector should work for monitors in any orientation.

While state-of-the-art object detection systems certainly perform reasonably well, there are applications that require even higher accuracy. It is an open research problem on how to combine **human knowledge** in detection systems to help improve the performance of a fully automatic system.

Objects in an unconstrained environment often appear together with other objects, resulting in a **cluttered** scene. For example, a laptop might be on top of a desk and surrounded by monitors, books, and keyboards. Without knowledge of which objects tend to appear with each other, detecting each object individually can be difficult due to occlusion and distraction from similar looking objects.

The **relationships between objects** provide useful information for localizing them in a complex scene. Modeling objects' interactions, however, is not an easy task. One key challenge is that the parametric space is prohibitively large: $O(K^2)$ for pairwise relations, and $O(K^n)$ for n-tuple relations of K objects.

Most state-of-the-art object detectors focus on treating objects in a scene as individuals. For example, region proposal based approaches such as Faster RCNN discard information outside of the proposal boxes. For an end-to-end bounding box regression approach such as SSD, global information is used but the relationship between objects are not modeled explicitly.

Understanding interactions between multiple objects is a challenging task, yet it is naturally the next research step over the current single detection based methods.

1.3 Towards Better Performance in Complex Scenes

While object detection is often done by looking at isolated local regions of an image, we believe it is crucial to understand the elements in the whole scene that can affect our ways of finding objects.

When an object appears in a scene, there are viewpoint and lighting changes that alter its appearance. The existence of other objects provides information on the possibility that a related object coexists. Some objects have strict constraints on where other objects can occur: for example, a train should be on rail tracks.

Understanding the characteristics of a scene helps a detection system to: 1) improve performance on objects with large variations; 2) reduce false detections by considering a typical environment of objects; 3) detect object layouts as an intermediate representation of scenes. This dissertation aims to study these effects of scene complexity on object detection tasks.

To facilitate collecting training data with large variations in poses and lighting conditions, we design a novel user interface, ARLabeler, utilizing the power of Augmented Reality (AR) devices. Instead of labeling images from the Internet passively, we put an observer in the real world and collect training labels for object detection with full control over the scene complexities. Users walk around freely and observe objects from multiple angles. Lighting can be adjusted. Objects can be added and/or removed to the scene to create rich compositions. Our labeling tool opens new possibilities to prepare data for complex scenes.

We also study challenges in deploying state-of-the-art object detectors in real world scenes. In particular, our task is to detect curb ramps in street view images. This is challenging because the object of interest is not visually salient and street view images contain a lot of distractors. We propose a system, Tohme, that combines detection results from computer vision algorithms and human crowdsourcing verifications for this task. One core component is a meta-classifier that estimates the complexity of a scene and assigns detection tasks to human (accurate but costly) or computer (low cost but error-prone) accordingly.

One of the insights from Tohme is that context is very important in detecting some objects. For example, driveway ramps are visually almost identical to curb ramps but they are located in very different context: driveway ramps are attached to houses while curb ramps are in intersections. To understand the complex relationship between objects and their environment, we propose a standalone context model that predicts where an object can occur in an image. By combining this model with object detection, false positives can be reduced. It can also be used for novel tasks such as finding regions where an object is missing or finding out-of-context objects.

To take a step beyond single object based detections, we explicitly model the geometrical relationships between groups of objects and use the layout information to represent scenes as a whole. We show that such a strategy is useful in retrieving indoor furniture scenes with natural language inputs.

In the following subsections, we give detailed introductions to each work and their connections to the main goal of this dissertation: finding objects in complex scenes.

1.4 ARLabeler: Labeling and Detecting Objects in the Real World Using Augmented Reality

Accurate data labeling is a critical element in training any object detection or image recognition system. Detectors pre-trained with internet-collected data such as ImageNet or PASCAL VOC face challenges in complex real-world scenes due to large variations such as poses and lighting. We present three contributions to enhance the labeling process in real scenes to train better object detectors.

First, we develop a fast labeling tool using an Augmented Reality (AR) platform. The tool leverages the 3D geometry and tracking afforded by modern AR systems, reusing a single labeling of a natural scene from an initial perspective to obtain labels for any arbitrary number of different perspectives of the same scene. At 5% of the labeling effort relative to fully manual labeling, our tool achieves 90% label accuracy in Intersection-Over-Union ratio.

Furthermore, the data collected by our method exhibits rich diversity in pose and lighting, and when combined with standard datasets (e.g. PASCAL), significantly improves object detection performance.

Our second contribution is an efficient object detection pipeline that leverages the datasets created by the AR platform. Recognizing that rectangular bounding boxes in the original labeling perspective will result in skewed rectangles in other perspectives, we extend standard detection algorithms such as SSD [2] and YOLO [3] with a refiner network that predicts skewed quadrilateral bounding boxes while still maintaining fast ($>$ 15 FPS) detection rate.

We make available the AR labeling tool and object detection algorithm to enable practitioners to leverage this framework for their individual applications.

Our third contribution is a series of three datasets comprised of over 25,000 AR- labeled images containing man-made objects and object parts under a rich variety of poses, lighting, and background clutter.

1.5 Tohme: Detecting Curb Ramps in Google Street View Using Crowdsourcing, Computer Vision, and Machine Learning

In some complex scenes, even the best state-of-the-art detectors have unsatisfactory performance (e.g., 57% average precision in detecting potted plants with SSD). On the other hand, human performance on object detection are near perfect, but with a much higher cost than fully automatic systems. The key challenge is how to balance the cost and performance in an object detection workflow that utilizes human inputs.

Building on a prior work that combines Google Street View (GSV) and crowdsourcing to remotely collect information on physical world accessibility, we present the first smart system, Tohme, that combines machine learning, computer vision (CV), and custom crowd interfaces to find curb ramps remotely in GSV scenes.

Tohme consists of two workflows: a human labeling pipeline and a CV pipeline with human verification, which are scheduled dynamically based on predicted performance.

Using 1,086 GSV scenes (street intersections) from four North American cities and data from 403 crowd workers, we show that Tohme performs similarly in detecting curb ramps compared to a manual labeling approach alone (F- measure: 84% vs. 86% baseline) but at a 13% reduction in time cost.

Our work contributes the first CV-based curb ramp detection system, a custom machine-learning based workflow controller, a validation of GSV as a viable curb ramp data source, and a detailed examination of why curb ramp detection is a hard problem along with suggested steps forward.

1.6 Seeing What Is Not There: Learning Context to Determine Where Objects Are Missing

A scene is considered to be complex and challenging to object detectors usually when there are many objects appearing in the same image with cluttered layouts. However, these objects are not positioned arbitrarily. For example, it is highly likely that a keyboard is found near a computer monitor. The spatial relationship between an object and its typical environment, or context, contains valuable hints on where to find objects in complex scenes.

We propose to train a standalone object-centric context representation. Given an image, our context model can predict where objects should exist, even when no object instances are present. Combined with object detection results, we can perform a novel vision task: finding where objects are missing in an image. Our model is based on a convolutional neural network structure. With a specially designed training strategy, the model learns to ignore objects and focus on context only. It is fully convolutional thus highly efficient.

Experiments show the effectiveness of the proposed approach in one important accessibility task: finding city street regions where curb ramps are missing, which could help millions of people with mobility disabilities.

1.7 Generating Holistic 3D Scene Abstractions for Text-based Image Retrieval

Most current object detectors are designed for detecting single objects. We are interested in finding multiple objects that are grouped in certain spatial layouts in a scene.

Spatial relationships between objects provide important information for text-based image retrieval. As users are more likely to describe a scene from a real world perspective, using 3D spatial relationships rather than 2D relationships that assume a particular viewing direction, one of the main challenges is to infer the 3D structure that bridges images with users text descriptions. However, direct inference of 3D structure from images requires learning from large scale annotated data. Since interactions between objects can be reduced to a limited set of atomic spatial relations in 3D, we study the possibility of inferring 3D structure from a text description rather than an image, applying physical relation models to synthesize holistic 3D abstract object layouts satisfying the spatial constraints present in a textual description. We present a generic framework for retrieving images from a textual description of a scene by matching images with these generated abstract object layouts. Images are ranked by matching object detection outputs (bounding boxes) to 2D layout candidates (also represented by bounding boxes) which are obtained by projecting the 3D scenes with sampled camera directions.

We validate our approach using public indoor scene datasets and show that our method outperforms baselines built upon object occurrence histograms and learned 2D pairwise relations.

1.8 Summary

Object detection - the task of finding object instances in images - is a fundamental problem in computer vision. It serves as an essential step in practical computer vision systems such as the ones used in self-driving cars. Applying current object detectors in complex real-world scenes faces challenges including: large variations in poses and lighting conditions and multiple objects in cluttered scenes. In this dissertation, we summarize four research works that aims to improve performance of object detections in complex scenes. In the following chapters, we discuss these works in details.

Chapter 2: ARLabeler: Labeling and Detecting Objects in the Real World Using Augmented Reality

2.1 Introduction

Deep learning has achieved tremendous success in computer vision in the past decade, spanning a range of applications from image classification to object segmentation with human-level accuracy. A key practical challenge that remains is efficient annotation of large datasets required to train deep networks. The manual labeling process is a tedious but significant, time-consuming endeavor. There are two primary means for manual labeling: 1) Domain experts provide high precision labels, which involves significant expenditure in cost and time; 2) Crowdsourced workers (e.g., Amazon Mechanical Turks) provide labels at low cost and fast turnaround, but potentially with compromises in labeling accuracy and precision. Another potential drawback with standard large-scale dataset collection is a bias towards certain types of images [4], especially those collected from the Internet. Such images often have a preponderance of certain views and lighting conditions over others, thus resulting in potential overfitting and poor generalization to real world scenarios.

There has been some effort to ease the burden of manual labeling of images by

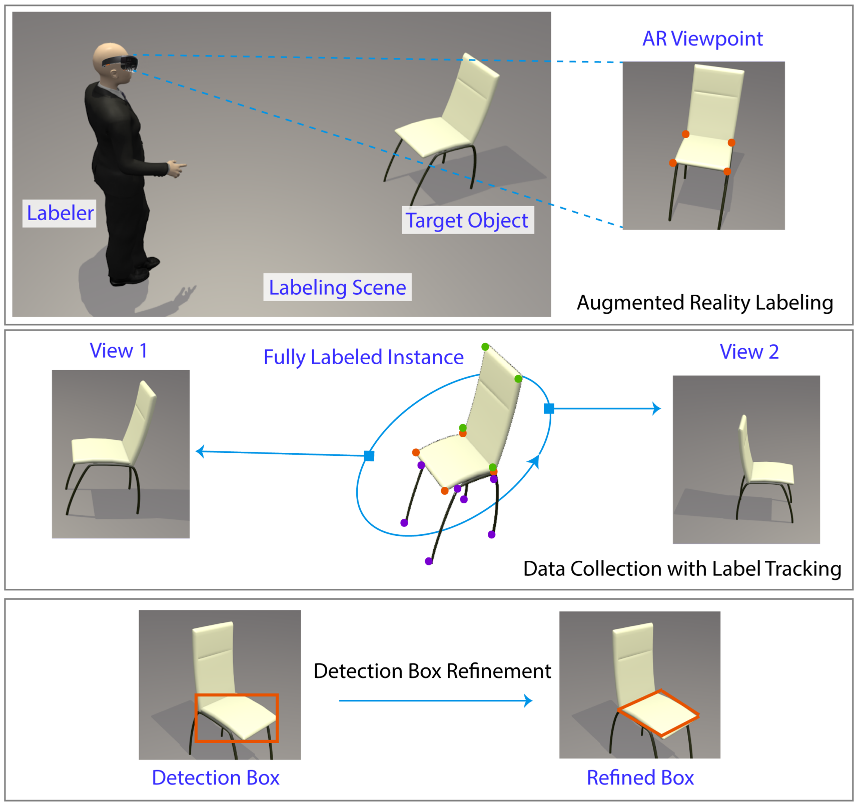


Figure 2.1: We propose a novel **Augmented Reality** (AR) based image labeling tool and an object detection refinement pipeline. In the **AR Labeling** stage, a human labeler uses our AR interface to label a scene with virtual markers. Next in the **Data Collection with Label Tracking** phase, these markers are propagated by the AR platform to help construct bounding box labels in a large number of images collected from different views. During testing, we propose a **Detection Box Refinement** pipeline to map axis-aligned detection boxes into skewed shapes that better fit the objects' spatial extent.

creating human-computer interfaces that provide annotation assistance; however, the majority of these tools only partially ease the labeling process. A few automatic labeling tools have also been proposed, but these suffer from poor accuracy in challenging scenes when captured frames have a lot of viewpoint changes, or from being specialized to particular datasets. These tools are discussed in detail in Section 2.2.

We take a fundamentally different approach to address these issues by proposing *ARLabeler*, an augmented reality (AR) based labeling interface to *annotate the real world*. In ARLabeler, users initially label a scene using virtual markers, which are then tracked by AR object localization as the user moves around the scene. The result is a large number of high quality labeled images that are rich in pose and lighting variations, collected at a small fraction of the time needed for fully manual labeling (Figure 2.1). Our experiments show that in a typical scenario, ARLabeler generates labels 22 times faster than fully manual labeling, at 90% accuracy (IOU). We also show that the data collected through ARLabeler exhibiting strong diversity in pose, lighting and other environmental factors, can be used to augment existing datasets such as PASCAL VOC to improve the performance of state-of-the-art object detection. This suggests that being able to efficiently collect a massive amount of data in a specific target domain (even with slight labeling imperfections) is critical to improving a computer vision system's performance in real-world scenarios.

It is worth mentioning that most current AR headsets are in the early stages of development and can be prohibitively expensive, however it is reasonable to believe that the cost will trend downward as the technology matures and adoption rates increase [5]. The primary focus of this work is to test the hypothesis that the tracking and mapping capabilities in current AR headsets are reliable enough to generate thousands of image labels with a single set of annotations under a wide range of adverse imaging conditions. ARLabeler can leverage any technical progress made by future headsets: for example, more robust object registration and tracking.

In addition, we propose an object detection pipeline specifically tuned for ARLabeler. We recognize that rectangular bounding boxes in the original perspective will result in skewed quadrilateral bounding boxes in other perspectives in the training data. This leads us to extend standard axis-aligned bounding box detection algorithms like YOLO and SSD to detect these skewed shapes while maintaining real time frame rates. We propose a light-weight refinement network that can be added to a standard object detector to predict transformations from standard axis aligned bounding boxes to the desired skewed non-axis aligned quadrilaterals. Our experiments show that the proposed refinement has significantly improved intersection-over-union (IOU) ratios with respect to the object's true outlines, when compared to axis aligned boxes.

To summarize, the contributions of this chapter are as follows:

1. A novel interactive AR interface that allows an annotator to precisely and efficiently label objects from different viewpoints and lighting conditions.
2. A light-weight refiner neural network that generates skewed bounding boxes to more tightly fit an object's appearance from arbitrary views.
3. Datasets generated by ARLabeler comprising over 25,000 AR-labelled images with man-made objects and object parts captured in three different scenarios under a rich variety of poses, lighting, and background clutter.

While we have developed ARLabeler using the Microsoft Hololens, the design framework and workflow are universal and should generalize to any AR device. We plan to release our labeling application tool and datasets to the community.

This work was done with Dr. Kumar Sricharan, Dr. Raja Bala, and Dr. Matthew Shreve during my internship at the Palo Alto Research Center (PARC).

2.2 Related Work

Manual Labeling Tools: The gold standard of data labeling is fully manual annotation with a visual user interface. In its basic form, a user clicks on an image to indicate landmark points or object boundaries. VGG Image Annotator (VIA) [6] is one such tool that has been used to label data for object detection and semantic segmentation. With crowdsourcing platforms such as Amazon Mechanical Turks, large amounts of labeled data can be acquired quickly and inexpensively as exemplified by the ImageNet [7] project comprising millions of images, classification labels and segmentation masks. One potential drawback of using crowdsourcing platforms is that additional protocols are often required to ensure quality and consistency of annotation; this is by itself an active research topic [8].

Assisted and Automatic Labeling Tools: Several methods have been proposed introducing some level of automation to ease labeling burden. ViPER [9, 10] uses key-frame animation style predictions to estimate (strictly linear) trajectories of objects moving in videos. LabelMe [11] interpolates future locations of bounding boxes based on feature-based tracking. Both approaches exhibit limitations. The first relies on a fixed camera perspective with objects moving at a relatively constant rate throughout the scene. The second approach does handle some shift in perspective, however the bounding boxes remain axis aligned and do not track accurately when the object undergoes rapid translations or shifts in perspective. Polygon RNN [12] guesses an initial set of polygon vertices defining the boundary of an object. A human annotator subsequently adjusts and refines this initial guess, thus requiring substantially less manual effort. Similarly, there have been methods that identify relevant segmentation masks from image descriptions [13], scribbles [14], as well as a single point [15]; however, none of these methods have been shown to generate ground truth level results [12]. Generating images and labels from computer generated scene models is also being actively investigated (unrealcv.org), however it is unclear how well such datasets generalize to real-world applications.

3D-Based Annotation Systems: Recently several methods have been proposed to collect semantic segmentation labels that leverage 3D information to speed up the process [16–18]. SemanticPaint [16] is an interactive VR approach that allows users to paint the surface of 3D reconstructed objects using a hand gesture that triggers a pixel-level label propagation algorithm. This system is designed to be fully online as a user provides live feedback of the labeling. Another interactive 3D labeling approach can be found in [17], wherein an initial 3D segmentation of the scene is performed using a combination of Markov Random Fields (MRF) and object localization, followed by refinement by a user. [18] uses a depth sensor and state-of-art algorithms to reconstruct a 3D indoor scene. Crowdsourced workers then annotate objects in the reconstructed 3D scenes.

To our knowledge, ARLabeler is one of the first AR headset based labeling approaches. One key difference between this and the aforementioned 3D methods is that our landmarks are environmentally anchored using the inverted tracking capabilities of AR headsets, and therefore remain stable even after significant appearance changes, and even after parts of objects have been articulated. This enables use cases beyond traditional object labeling. For example, a single AR-annotated bounding box for a region can indicate the two states of the same cabinet door (open / closed) that is useful for state classification. Also, utilizing the 3D registration and tracking functions in the hardware level of an AR device, ARLabeler does not require heavy 3D reconstructions such as [18] and instead focuses on user interactions in image labeling. Loaded in an AR headset, ARLabeler is light-weighted and easier to use in various environments, compared to a Kinect-based system such as [16].

Bounding Shapes for Object Detection: Bounding shapes range from highly descriptive semantic and pixel-level segmentation [19–22] to simple axis aligned bounding boxes (AABB) [2, 3, 23]. The pixel-level ground truth [24–26], while enabling precise segmentation, suffers from prohibitively high labeling cost and slow detection rates in practice. On the other hand, AABB methods, while achieving fast detection rates [2,3], are often coarse with respect to the objects' true visual extent. Our work falls in the middle of these two extremes, enjoying the advantage of the impressive computational performance achieved with methods such as YOLO [3] and SSD [2], while also providing a richer degree of localization that is beneficial to many real-world applications.

The most closely related approaches for locating and labeling objects with non-axis aligned bounding boxes (NAABB) can be found in scene text detection applications [27–30], wherein it is often necessary to detect the orientation/stroke of text and handwritten characters as a pre-processing step before further downstream processing can take place. In the work by Ma et al. [27], a modified proposal network (RRPN) is described that allows the network to learn a rotation θ in addition to the location (x,y,w,h) of a rectangular bounding box through the use of a loss function that is based on the intersection over union (IoU) as well as the intersection angle with the ground truth bounding box. A NAABB approach has also been proposed for hand pose detection [30]. Similar to [27], a proposer network is trained that jointly learns location and orientation, while also taking context into consideration. In contrast, our approach does not rely upon any task-specific constraints or context, and learns a more general parametric transform as a post-refinement step following any object detection pipeline. Additionally, our approach uses convolutional layers to learn the AABB to NAABB transformation through the use a spatial transform [31] based sampling layer to handle arbitrary AABB patch sizes. In contrast, [27–30] are restricted to using fully connected layers based on pooled features to detect the transformation. Also, unlike the work in [31] which uses a spatial transform to normalize the entire input image for classification, we use the spatial transform layer to scale the AABB image patch of interest.

2.3 ARLabeler Interface

In this section, we describe ARLabeler in greater detail. We begin by describing the overall design and critical components that we found necessary to create an effective and efficient annotation tool. We then describe a typical workflow for collecting and labeling images that are rich in pose variations and lighting conditions.

2.3.1 Design Requirements

1) Intuitive Interactions The majority of annotation tools available in the literature are restricted to a keyboard and mouse. However for ARLabeler, we leverage voice and gesture modes that are fully integrated with most commercially available AR devices. Our goal is to minimize the effort required by the annotator in terms of hand and gaze movements.

2) Low Cognitive Load Data annotation can be an exhausting task, and we want to reduce the cognitive burden on the labeler, i.e., there should be no difficulty disambiguating annotations once placed. We address this by leveraging depth perception and using easily distinguishable marker colors.

3) Flexible Object Boundary Labeling As a general tool, we want to be able to label not only bounding boxes, but also keypoints and the detailed boundaries of target objects. ARLabeler leverages the surface meshes that are produced by the AR device to initially guide placement of boundary points. These individual markers can then be adjusted using simple voice commands and/or gestures.

2.3.2 Interface Components

As shown in (Figure 2.2) ARLabeler is a virtual environment embedded in the real world from a user's point of view. It contains the following visual components

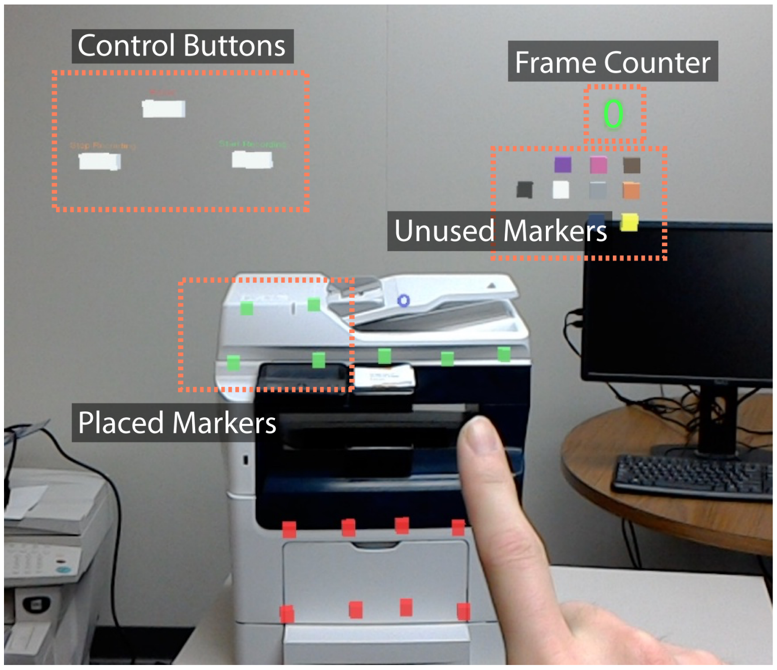


Figure 2.2: ARLabeler interface with control buttons, virtual markers, and the capture counter.

and functional modules that leverage a typical AR device's capabilities.

1) Virtual Markers are unit-length cubes that can be rendered onto the real world's surfaces. Depending on the number of markers being used, it is possible to annotate bounding boxes, keypoints, or complete outlines of an object. Markers are grouped by colors. For the default view, we have chosen the eleven basic colors based on the color naming theory of Berlin and Kay [32]. Each named color group represents a consistent visual concept: e.g., the outline of a car. An unlimited number of markers can be created one by one by voice command "New #COLOR", where #COLOR is one of the eleven color names.

2) World Surface Mesh Map is obtained from the AR device. The mesh is an estimate of the $[X, Y, Z]$ coordinates of solid surfaces in the real world, and is used as a guidance to let the user quickly attach markers on to real world objects. Because these meshes are produced with 3D sensors, they are robust to environmental changes in lighting and/or cosmetic appearance changes made to objects.

3) Interaction Mode Our application offers two modes by which a user can place markers: fast mode and refine mode. In fast mode, the user uses gaze to guide and attach markers onto the surface mesh. This is the most efficient way to quickly draw a large number of markers that are roughly aligned with the object. To improve marker placement accuracy, the user can invoke the refine mode, wherein hand gestures are used to move markers freely in three dimensions, without the constraint that they lie on a mesh. The AR device tracks the movement of the hand and replicates its motion with the target marker. The two modes are invoked by voice commands.

4) Controls There are three virtual buttons for controlling the data collection procedure: start, stop, and reset. Upon pressing the start button, the front facing camera starts capturing frames at a preset rate. At each time step, a photo is captured and all 3D marker coordinates are recorded, until the stop button is pressed. During capture, a counter is displayed to show how many photos have been captured.

5) Geometric Transformation Information is stored for every captured frame. The coordinates of markers in 3D are first transformed to camera coordinates, and then converted to the 2D image plane with the AR camera's projection matrix.

6) Visibility Test After placement, markers might be hidden in a particular view due to being out-of-sight or occluded. We check a marker's location in the 2D image plane to determine if it is out-of-sight. For occlusion, we utilize the surface mesh map and compare two vectors in the world coordinate system: 1) the vector from the camera center to the marker; 2) the vector from the camera center to the first point where the user's gaze hits the surface mesh map. If the difference between the two vectors exceeds a certain threshold, we assume there is an occluding object between the current camera position and the marker.

2.3.3 Data Collection Workflow

A typical workflow using ARLabeler for data collection is summarized as follows:

1. *Preparation.* The user identifies the target object(s) to be annotated. Walking around the scene may be required for the AR device to estimate and register the surface of each object.

2. *Interactive Labeling.* The user selects color markers and aligns them with the target object(s). To label a single object, the user places markers of the same color at the extreme points of that object to bound its volume, and then at multiple locations along its boundary. To label multiple objects in a room, the user can use colors as semantic labels for different object categories, or as instance level labels for different objects in the same semantic category.

3. *Photo Capture.* Once the initial labeling is done, the user starts the recording process and simply walks around the scene as ARLabeler takes photos from various viewing angles and lighting conditions (day, night, flashlights, etc.). Photos are captured at a preset frame rate, and all of the markers' positions for each frame are automatically calculated and stored. The process can be interrupted and resumed over an extended period of time, since the markers lock onto their respective locations.

4. *Post-processing.* Captured images and markers' coordinates are stored with their corresponding frame indices. Post-editing of annotations, e.g., merging or removing of markers is supported.

Through this interface and workflow, ARLabeler is able to achieve high quality labels similar to fully manual labeling, but with much less time and effort. Consider P scenes with K objects and N number of frames to be labeled. Manual labeling will take $O(N \cdot K \cdot P)$ human time, while ARLabeler takes $O(K \cdot P)$ time, because only the initial labeling phase requires human effort. This is a significant speed up especially when N is large (e.g. tens of thousands for a typical vision dataset). A quantitative comparison is provided in Section 2.5.2.

Discussion. The standard dataset collection pipeline in computer vision is a two stage process. First, images are collected either from the Internet or a target environment such as an indoor scene. Second, images are submitted for annotation by domain experts or crowdsourced workers (e.g. Amazon MTurkers). ARLabeler, on the other hand, couples both data collection and labeling into a single simultaneous operation. A modest overhead is introduced at the start of the data collection process where a user wears the AR device and places a few virtual markers at key points in the virtual scene. Once the data capture is completed, the images are already labeled, resulting in a much shorter turnaround time between data collection and deployment of real-world computer vision systems. Throughput efficiency will be demonstrated quantitatively in the experimental section.

2.4 Bounding Box Refinement Network

We now switch our attention to training of object detection systems that can leverage the data collected through ARLabeler. Recall that the user specifies virtual markers that are bound to objects during labeling, and as a result, axis aligned bounding box labels created by the user in the initial perspective transform to skewed, non-axis aligned quadrilaterals in later perspectives. This presents us with two options: (i) use the markers to construct loose, axis-aligned bounding boxes in all perspectives (see left image in Figure 2.3) and use a standard detector like SSD or YOLO, or (ii), use the markers to construct tighter, but skewed, non axis aligned quadrilaterals (see right figure in Figure 2.3), and build a new detection pipeline to detect the non axis aligned bounding boxes. We choose the latter approach for the reasons highlighted below.

2.4.1 Motivation

Two factors prompt a consideration for non-axis-aligned bounding boxes: task utility and IOU. We discuss each in turn.

In a real world AR application, the bounding box from the object detection step is often used to create virtual annotations that assist end-users for various tasks, including machine repair [33] or surgery [34]. Axis-aligned bounding boxes

(AABB) are not an ideal choice for this purpose, since they “float” on top of the object rather than aligning with its 3D orientation, and cover a large region not within the object’s extent. This becomes especially problematic in cluttered scenes where overlapping object labels impede the visual task, as shown in Figure 2.3.

We conducted a pilot study to understand how bounding box geometry affects visual task guidance. We collected twelve images of three parts of a large office printer and labeled them using both axis aligned bounding boxes (AABBs) and non-axis aligned bounding boxes (NAABBs) at varying perspectives. We then asked five people to identify which part was indicated by each bounding box in each image. Part identification accuracy was approximately 72% on AABB-labeled images, and as expected, 100% with the NAABB-labeled images (images in the latter category were primarily used as a control). In addition, for the AABB-labeled images, 3 out of 4 had at least one user indicating the wrong part. It is also worth noting that more mistakes were made on images where the parts had large out-of-plane rotations (for example, see the blue, yellow, and orange bounding boxes in Figure 2.3). We believe this result indicates that tighter fits to object shapes (higher IOU) result in higher visual identification, especially on real world objects exhibiting significant perspective distortion.

IOU is a standard metric to measure object detection performance. One approach to improve IOU is to estimate the full outline of objects, or instance-level segmentation. However, such segmentation is computationally expensive: Mask RCNN [35] has a frame rate of 5FPS, which creates notable lags in any real-time AR application. Instead, we hypothesize that simple geometric transformations of the AABB labels can significantly improve detection quality while maintaining the desired frame rate for real-time applications. To validate this hypothesis, we conducted a small study on the COCO dataset. We calculated IOU between the AABB and the ground truth pixel-level object labels. For the NAABB ground truth target, we find the optimal 3×3 affine transformation that yields the tightest fitting quadrilateral around the object's outline, and determine its IOU ratio. The result was an average IOU of 0.559 for AABB and 0.66 for the transformed NAABB shape. This quantitatively illustrates the potential gains that can be obtained by detecting NAABBs over AABBs.

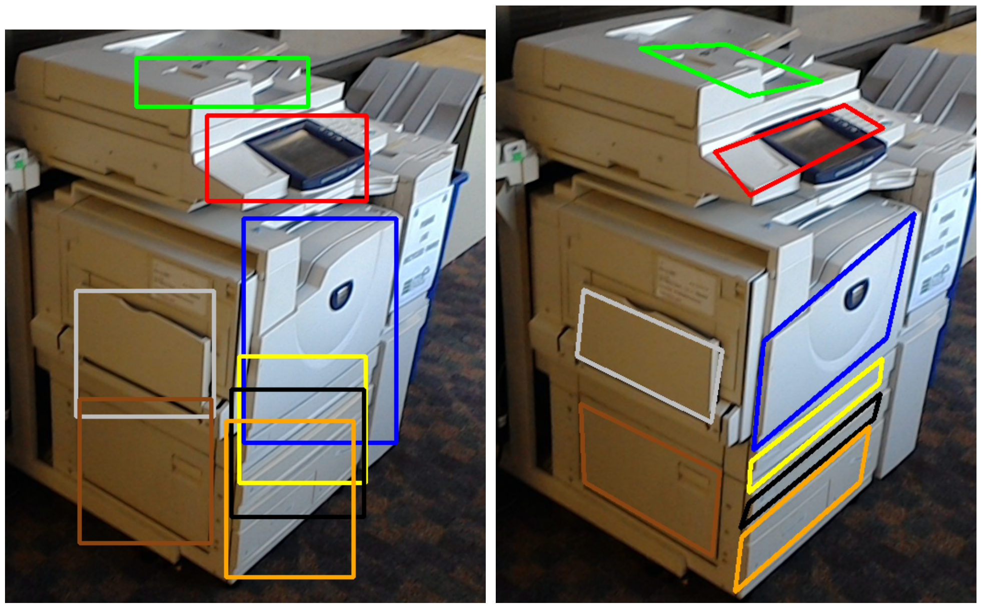


Figure 2.3: Left: A cluttered scene image with AABB. Right: The same scene image with NAABB. It is clear that the NAABB provides better guidance on where objects are.

2.4.2 Network Structure

We propose a Bounding box Refinement Network (BRN) that takes as input an image and a canonical AABB provided by a standard object detection network, and predicts the parameters of a spatial transformation that maps the AABB to a target NAABB that optimally fits the object's spatial extent. Figure 2.4 illustrates the network structure. We emphasize that BRN is a general purpose network that can be applied to arbitrary datasets beyond those collected through ARLabeler.

The BRN first samples pixels from the region of interest (ROI) defined by the AABB. This operation is similar to the pixel sampling mechanism used in the Spatial Transformer Network [31], and it has two benefits: 1) Sampling directly from the image makes BRN independent of any specific object detectors; 2) The sampling results in a fixed size image patch from which it is easy to extract features. Next, BRN extracts features through several convolutional and fully connected layers and regresses on these features to yield a parameterized geometric transformation. This transformation can be used to convert the coordinates of any bounding shape in the canonical space to a shape in the image space that better aligns with the target object.

The BRN, as a function of the input image I, minimizes $\|X_t - T(I)X_s\|_2$, where X_s are augmented coordinates in the axis-aligned canonical space and X_t are the target coordinates that are aligned with the object's orientation. $T(I)$ is a 3×3 transformation matrix that can be flattened to a 9-dimensional vector.

There are several transformation choices for $T(\cdot)$:

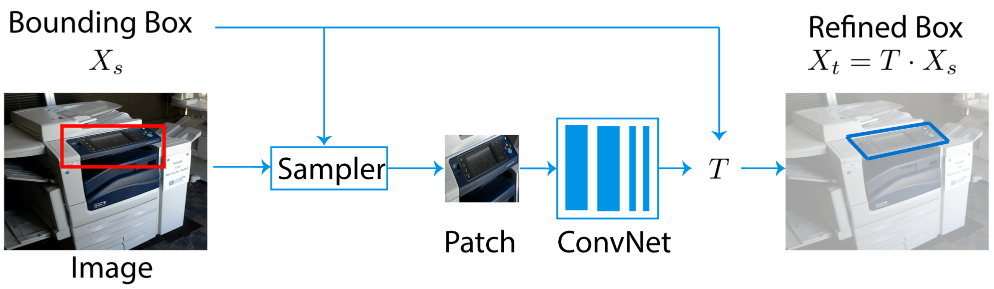


Figure 2.4: Network structure of BRN.

1. Rotation: $[\cos(\theta), -\sin(\theta), 0, \sin(\theta), \cos(\theta), 0, 0, 0, 0]$
2. Rotation + scale in x,y:

   $[s_x \cos(\theta), -s_y \sin(\theta), 0, s_x \sin(\theta), s_y \cos(\theta), 0, 0, 0, 0]$
3. Affine (2×3): $[a_1, a_2, a_3, a_4, a_5, a_6, 0, 0, 0]$
4. Homography (3×3): $[h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, 1]$

It is easy to modify the number of regression outputs in BRN to output any of the aforementioned transformations, of which homography is the most general. With efficiency in mind, the network is designed to be simple, and comprises two convolutional filter layers, batch normalization layers, and a fully connected regression layer. While our implementation is class-independent, it is straightforward to take class predictions and concatenate them with the fully connected layer to predict class-dependent transformations. Compared with an end-to-end coordinates regression approach, BRN has several advantages. 1) It learns a small set of parameters (only 6 for an affine transform), comparing to $O(k)$ number of parameters for a direct coordinates regression approach, where k is the number of keypoints along the object's outlines. 2) The geometric transformation matrix is easier to learn because the parameters are decoupled: learning the rotation angle is independent from learning the scaling factor. In contrast, direct coordinate regression has to predict all coordinates in a joint fashion. This generalizes the common practice in object detection that predicts (x, y, w, h) for a box instead of $(x, y)_4$ pairs. 3) The estimated geometric transformation can be independently applied to virtual AR content such as arrows or pop-up text boxes for user guidance.

2.5 Experiments

In this section, we first evaluate ARLabeler as a data collection and labeling tool, and measure its efficiency and accuracy in propagating manually placed markers to images captured at different perspectives. In the second part of this section, we show the effectiveness of BRN at transforming an AABB to better fit an object's boundary. Finally, we show that combining data collected by ARLabeler and existing datasets can improve object detection on both sets of data.

2.5.1 ARLabeler Dataset

We believe that the ability to collect images with wide variation in 3D object pose and orientation, as well as different types of data including key points, bounding boxes and boundaries, is crucial to boost object detection and recognition research. We collected three datasets of various types that demonstrate the flexibility of ARLabeler to this end.

1) Data-Room contains images of multiple objects in a room. For each image, object instances are labeled by markers; different colored markers are used to differentiate each of the objects and generate bounding boxes (AABBs or NAABBs). Labeling is performed as described in Section 2.3.3, with an annotator walking around a room capturing images of each object at various viewpoints, adjusting ambient lighting, and placing different distracting objects in the vicinity. The result is 9255 images captured in 4 room types: kitchen, office room, conference room, and office common area, with 30 object classes and under rich variations in pose and illumination.

2) Data-Part contains images of object parts labels. This is crucial in applications where the user needs to interact with specific parts of an object (e.g. locating the jammed paper tray of a printer). This dataset comprises 10872 images of 9 different parts of 7 office printers, captured under vastly different poses, illumination and background clutter.

3) Data-Shape contains images of single objects of complex shapes. This shows ARLabeler 's ability to collect labels of objects that are beyond rectilinear shapes. The dataset comprises four stuffed animals: lizard, cheetah, giraffe, and zebra. For each animal, a user puts virtual markers along its physical outlines from one viewpoint. Also, we put an additional four markers as a reference box to bound the shape. Subsequently, ARLabeler produces a large dataset with these objects at different orientations, lighting and clutter. The result is 5192 images with an average of 48 points per animal.

The three datasets in total contain over 25,000 labeled images; typical exam-

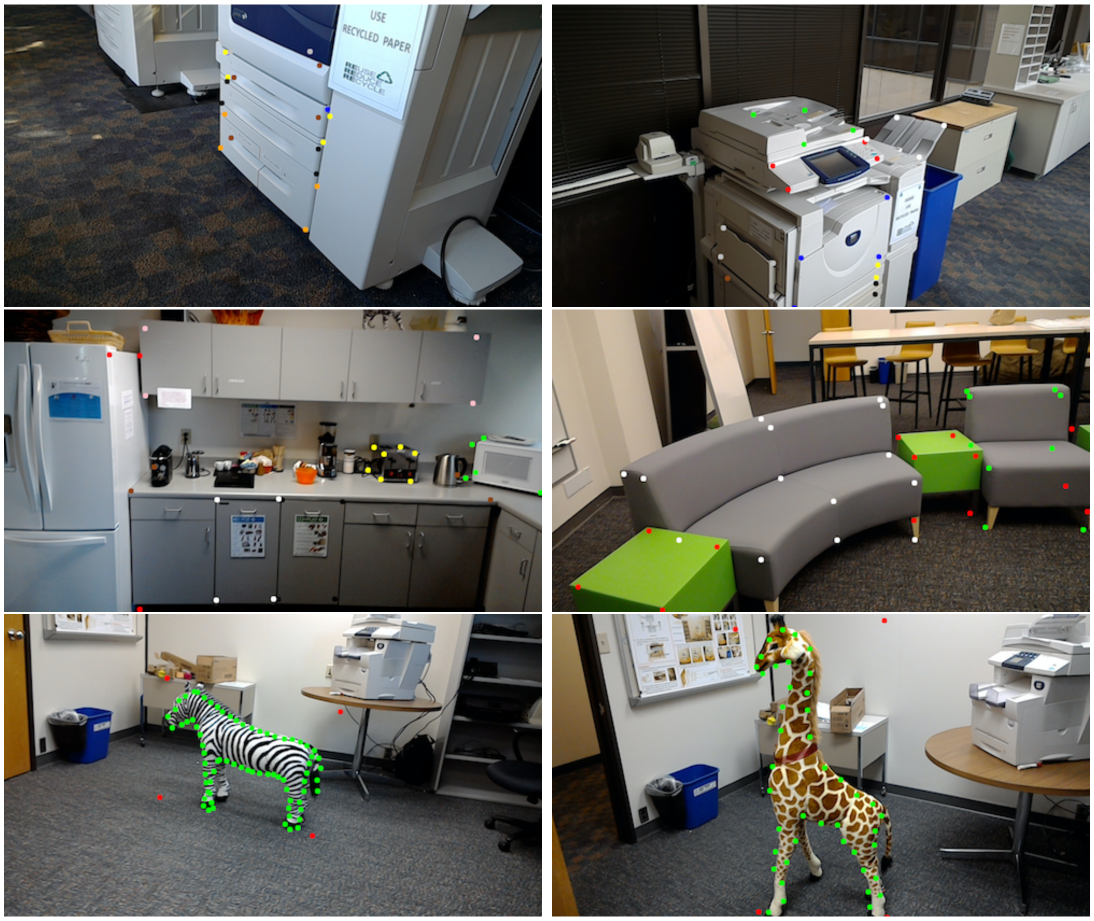


Figure 2.5: Example images and marker labels of our datasets. Top: Data-Part. Middle: Data-Room. Bottom: Data-Shape.

ples are shown in Figure 2.5.

2.5.2 ARLabeler Annotation Quality

In this experiment, we compare ARLabeler quantitatively with image labeling tools that are widely used in the computer vision community.

We randomly selected 50 frames from the Data-Parts dataset, containing on average 6 parts per frame and a total of approximately 300 regions. Each part is labeled by a polygon with 4 vertices. We compare ARLabeler with representative manual labeling, semi-automatic video labeling, and crowdsourced labeling methods. For manual labeling, we use an online tool (VIA) [6]. For semi-automatic video labeling, we use a popular video-based labeling method called VATIC [36], in which a user labels key frames and the software predicts the labels for the remaining frames by tracking visual features. One volunteer was hired to complete both tasks. For crowd-sourced labeling, we developed a web-based manual labeling interface that is similar to ARLabeler in function, and serves selected frames to Amazon Mechanical Turkers for annotation. The crowdsourced workers are guided to label the same object parts, using a few reference images as examples.

We focus on the trade-off between efficiency and accuracy when evaluating a labeling method. *Efficiency* is measured by time spent in the labeling process. For ARLabeler, the duration of the volunteer placing virtual 3D markers onto the objects is recorded. For manual labeling, the time spent drawing polygons on each frame is recorded and summed, while for video labeling, the time spent for labeling the key frames is recorded. *Accuracy* of the obtained labels is measured by intersection-over-union (IOU) ratio between the ground truth polygons and those obtained using each of the labeling methods. We believe that higher IOU indicates higher correlations between the labels and true object boundaries, which is important for superior object detection performance. Note that manual labeling is not 1.0 due to the discrepancy between the volunteer and the ground truth source.

1) Comparison with Manual Labeling Table 2.1 shows that the accuracy of ARLabeler is comparable to that of manual annotation (0.74/0.82 = 90%) with only a fraction (5%) of the time cost for labeling the selected frames. Moreover, the time required for manually labeling N frames is proportional to N; while ARLabeler requires no additional time cost for labeling additional frames for the same objects in the same scene.

	Manual	ARLabeler
IOU	0.82	0.74
Time (50 Frames)	133	6
Time (N Frames)	2.66*N	6

Table 2.1: Accuracy and efficiency comparison between manual labeling and ARLabeler. ARLabeler comes close to manual labeling in terms of accuracy, but at a fraction of the labeling time. Times are in minutes.

2) Comparison with Video Labeling For our data, the video labeling tool, VATIC, while being very time efficient, suffers from inferior labeling accuracy if used without heavy user interventions (Table 2.2). By only labeling the initial frame, VATIC loses track of part labels quickly after large viewpoint changes. By adding some key frame annotations in viewpoint transitioning scenes, VATIC is able to improve. With heavy key frame annotations and adjustments (60% of total frames), it is able to achieve about 57% (0.47/0.82) accuracy in IOU. With even more key frame annotations and adjustments, we expect VATIC to further improve accuracy, but with time cost approaching that of a fully manual process.

Also noteworthy is that ARLabeler can accurately track markers from various angles while the semi-automated video labeling fails with strong geometric changes across frames. In general we believe that feature-tracking based labeling works well only when there is no significant motion between frames, and object parts are visible across most frames. ARLabeler utilizes the built-in capabilities of an AR device to provide scene level registration and tracking that is highly robust to viewpoint changes.

	VATIC			
	Init	10% KF	60% KF	ARLabeler
IOU	0.19	0.27	0.47	0.74
Time (50 Frames)	1	2	10	6
Time (N Frames)	1	0.04*N	0.24*N	6

Table 2.2: Accuracy and efficiency comparison between VATIC and ARLabeler. KF stands for portion of added key frame annotations. VATIC needs a significant amount of annotations from users to achieve a good IOU level. Times are in minutes.

3) Comparison with MTurk We assigned 150 HITs on Amazon MTurks and recruited 26 workers to complete our annotation task in two days. On average, there were three workers annotating each image. The final polygon was generated by majority votes of labeled pixels. Table 2.3 shows the comparison of the quality of the MTurk labels and ARLabeler. MTurk labels have slightly lower label accuracy and a much higher turnaround time.

In terms of the cost in money, MTurk results are $0.15 for hiring three turkers per image. To collect a dataset of N images, the total cost goes linearly with N. The cost of ARLabeler is a constant of one-time initial investment on the AR device. We assume the cost of hiring a person to initially label the scene is negligible and considered as included in the total cost of the whole image capture process.

In general, there are many uncertain factors that come into play in a crowd-sourced labeling approach. The task requester has no accurate estimation of time to complete the HITs. For each new task, there is an unknown probability that a MTurker misunderstands the instruction and performs sub-optimally. Thus the time needed for post-processing and verification of the labels is difficult to quantify deterministically. In addition, complex tasks that require domain knowledge such as labeling functioning parts of a complicated machine are not suitable for MTurkers. These factors make MTurk annotation a less favorable choice compared with ARLabeler in scenarios requiring fast and accurate labeling.

	MTurk	ARLabeler
IOU	0.71	0.74
Time (50 Frames)	$\sim$ 48 hours	6 min

Table 2.3: Accuracy and efficiency comparison between MTurk labeling and ARLabeler.

Discussion The ARLabeler tools uniformly performs better than its competitors as demonstrated by above results. While this underlines the significant value of this approach, we also would like to point out that the ARLabeler has a few limitations. Specifically, due to hardware characteristics of the chosen Microsoft Hololens AR device, the whole 3D coordinate system shifts when a user undergoes large motion at times. We have also noticed that registration and tracking of virtual markers are not reliable for objects with reflective surfaces. With the advent of superior AR headset technology over time, we expect more robust data annotation performance using ARLabeler.

2.5.3 Bounding Box Refinement

In this experiment, we investigate BRN's ability to improve IOU over the standard axis-aligned bounding box. We first run the input image through a SSD object detector to obtain axis-aligned ROIs for each object, which are then supplied to BRN for refinement.

We compare different parametric BRN transformations with end-to-end coordinate regression and a rotated bounding box approach. The end-to-end method takes in an image region of interest and directly regresses coordinates of all points. The rotated bounding box approach predicts $[x1, y1, x2, y2, h]$ parameters [28] that represent a rectangular shape with arbitrary rotations. All methods are learning based, and for fair comparison, share the same convolutional feature extraction structure. Specifically, we employ two convolutional layers, two batch normalization layers and one regression layer. We use the Data-Part and Data-Shape datasets for evaluation. Each dataset is randomly split into 90% training and 10% testing. Implemented in Caffe, we train each method with the Adam optimizer until stability is achieved.

With the Data-Part dataset, there are four pairs of marker (x-y) coordinates to be estimated for each printer part. The result is summarized in Table 2.4. Example predictions are shown in Figure 2.6.

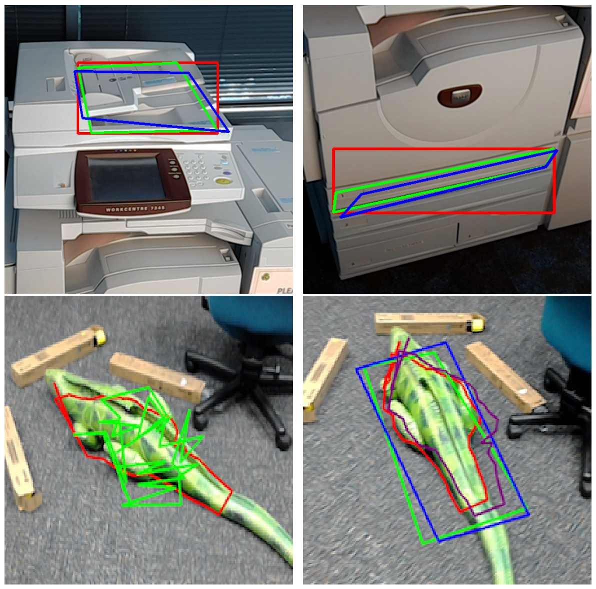


Figure 2.6: Top row: Example of Data-Part predictions. Red box: axis-aligned box. Green box: ground truth box. Blue box: BRN estimated box. Bottom row: Example of Data-Shape predictions. Left: End-to-End predicted shape (green) compared with ground truth (red). Right: BRN estimated shape (purple) compared with ground truth (red).

Method	AABB	End2End	xyxyh	BRN(R+S)	BRN(H)
IOU	0.44	0.63	0.53	0.60	0.68
MSE	0.50	0.17	0.75	0.19	0.13

Table 2.4: Comparison of different bounding box refinement methods on Data-Part dataset. BRN(R+S): BRN with rotation + scale. BRN(H): BRN with homography.

Method	AABB	End2End	BRN(H)
IOU (37 pts)	0.322	0.39	0.678
IOU (73 pts)	0.17	0.19	0.623

Table 2.5: Comparison of different bounding box refinement methods on Data-Shape dataset, with different number of shape points. NAABBs produced by BRN completely outperform AABBs.

For the Data-Shape set, for each object, BRN predicts the transformation matrix T that will be used to transform the reference bounding rectangle. Next, we extract the object category (e.g., lizard) output by SSD and use this to look up a prior object shape in its canonical space. The prior shape is obtained by averaging over all shapes of various views that are transformed back to the canonical space in the training data. The final object shape is then predicted by applying T on the prior shape. The result is summarized in Table 2.5. Figure 2.6 shows qualitative examples.

With respect to computational performance, BRN achieves high precision with little sacrifice in speed. When combined with SSD, the overall pipeline runs at about 20 FPS on a Titan 1080Ti GPU. As a reference, the more detailed instance segmentation, Mask RCNN, runs at 5 FPS on the COCO dataset.

Discussion On the Data-Part dataset, BRN with homography transformation outperforms all methods, with end-to-end training coming closest to a comparable performance. However on the Data-Shape dataset, BRN is significantly (70%) better than the end-to-end training. One critical reason is that learning four coordinates regression is much simpler than learning all fine-grain coordinates (Table 2.5). Figure 2.6 shows that the end-to-end training fails completely for complex shapes. On the other hand, BRN does not learn a direct coordinate mapping, but rather a simple parametric transformation between image spaces that is much easier to learn. Furthermore the BRN-learned transformation can be advantageously used to overlay text and pop-up animation in an AR setting - a feature that does not appear achievable by end-to-end training.

2.5.4 Data Augmentation for Enhanced Object Detection

As mentioned earlier, standard object detection datasets including PASCAL VOC [26], DAVIS [24] and COCO [25] are often biased in terms of pose and lighting conditions, leading to mixed results when used to train object detectors for real-world applications. In this experiment, we show that by using ARLabeler, it is possible to quickly collect a large number of labeled images that can then augment these existing datasets to train a more robust detector with superior performance. We selected samples from the 'monitor' class in the PASCAL 2012 VOC datasets. We then used the ARLabeler tool to collect 3000 additional images of six different types of monitors in the span of 2 hours. We combined this new dataset with PASCAL and trained an SSD detector with different combinations of training/testing splits. The result is summarized in Table 2.6. The first entry of this matrix [PASCAL (train) / PASCAL (test)] used a publicly available SSD model that was pre-trained on the PASCAL VOC dataset.

	PASCAL (test)	AR (test)
PASCAL (train)	0.52	0.39
PASCAL+AR (train)	0.66	0.49

Table 2.6: Average precision with different training configurations of a SSD detector trained on monitors.

The PASCAL trained detector does not work well on our AR data because many monitor images in PASCAL are taken from a frontal view, and as a result, the detector performs poorly on non-frontal views in the AR data. When jointly trained with PASCAL and AR data, the SSD detector is able to improve performance significantly on both the AR dataset and the PASCAL dataset by about 20%. This suggests that the AR data provides valuable pose and environmental variations in training an object detector to complement the rich object type variation in the PASCAL data. This experiment illustrates the promise of ARLabeler in efficiently filling the gap between the performance of detectors built on existing benchmark datasets and that of detectors applied in real world settings with enormous variations in pose, lighting and other confounding variables.

2.6 Summary

We propose an augmented reality application, ARLabeler, that enables intuitive and efficient large scale data annotation with greatly reduced labeling time, and increased samples diversity when compared to standard data collection and labeling methods. ARLabeler can be used to generate standard axis-aligned bounding boxes, as well as object key-points and detailed outlines quickly and accurately. It labels images 20 times faster than typical manual labeling, and is useful to rapidly create new datasets that can be used standalone to train accurate object detection systems, or combined with existing benchmark datasets to further improve state-of-the-art detectors. Additionally, we present BRN, an efficient neural network module with minimal computational overhead, that refines axis-aligned bounding boxes from a standard object detector to predict skewed boxes that more closely align with an object's visual extent. We demonstrate significantly improved IOU (50% improvement over axis-aligned bounding boxes) for general object detection applications. Finally we contribute three datasets collected using *ARLabeler* that exhibit rich diversity in pose, lighting, and background clutter. We plan to release the ARLabeler labeling tool to the scientific community to enable rapid creation of high volume real-world datasets for general purpose machine vision systems.

Chapter 3: Tohme: Detecting Curb Ramps in Google Street View Using Crowdsourcing, Computer Vision, and Machine Learning

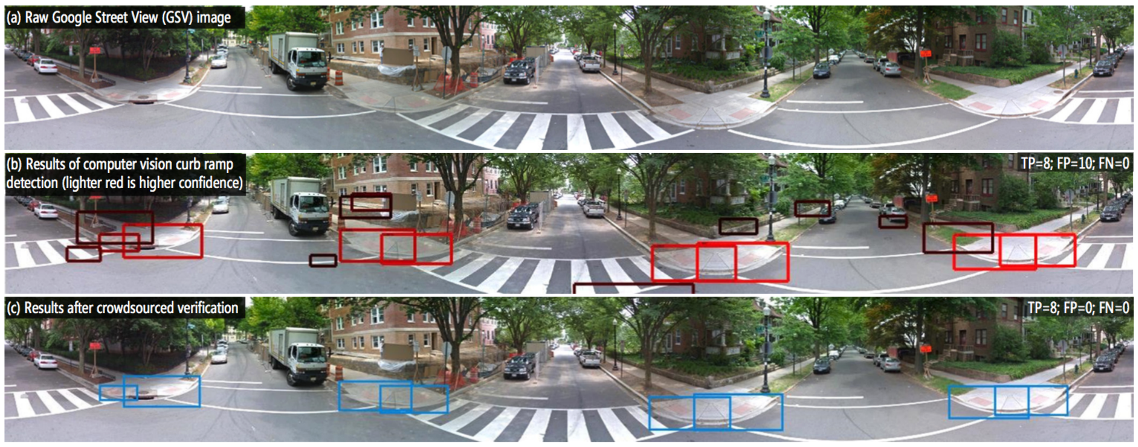

(a) Raw Google Street View (GSV) image
(b) Results of computer vision curb ramp detection (lighter red is higher confidence)
TP=8; FP=10; FN=0
(c) Results after crowdsourced verification
TP=8; FP=0; FN=0

Figure 3.1: In this chapter, we present Tohme, a scalable system for semi-automatically finding curb ramps in Google Streetview (GSV) panoramic imagery using computer vision, machine learning and crowdsourcing. The images above show an actual result from our evaluation.

3.1 Introduction

Recent work has examined how to leverage massive online map datasets such as Google Street View (GSV) along with crowdsourcing to collect information about the accessibility of the built environment [37–41]. Early results have been promising; for example, using a manually curated set of static GSV images, Hara et al. [39] found that minimally trained crowd workers in Amazon Mechanical Turk (turkers) could find four types of street-level accessibility problems with 81% accuracy. However, the sole reliance on human labor limits scalability.

In this chapter, we present Tohme[1], a scalable system for remotely collecting geo-located curb ramp data using a combination of crowdsourcing, Computer Vision (CV), machine learning, and online map data. Tohme lowers the overall human time cost of finding accessibility problems in GSV while maintaining result quality (Figure 3.1). As the first work in this area, we limit ourselves to sidewalk curb ramps (sometimes called "curb cuts"), which we selected because of their visual salience, geospatial properties (e.g., often located on corners), and significance to accessibility. For example, in a precedent-setting US court case in 1993, the court ruled that the "*lack of curb cuts is a primary obstacle to the smooth integration of those with disabilities into the commerce of daily life*" and that "*without curb cuts, people with ambulatory disabilities simply cannot navigate the city*" [42].

While some cities maintain a public database of curb ramp information (e.g., [43, 44]), this data can be outdated, erroneous, and expensive to collect. Moreover, it is not integrated into modern mapping tools. In a recent report, the National Council on Disability noted that they could not find comprehensive information on the "*degree to which sidewalks are accessible*" across the US [45]. In addition, the quality of data available in government systems is contingent on the specific policies

[1]Tohme is a Japanese word that roughly translates to "remote eye."

and technical infrastructure of that particular local administration (e.g., at the city and/or county level). While federal US legislation passed in 1990 mandates the use of ADA-compliant curb ramps in all new road construction and renovation [46], this is not the case across the globe. Our overarching goal is to design a scalable system that can remotely collect accessibility information for any city across the world that has streetscape imagery, which is now broadly available in GSV, Microsoft Bing Maps, and Nokia City Scene.

Tohme is comprised of four custom parts: (i) a web scraper for downloading street intersection data; (ii) two crowd worker interfaces for finding, labeling, and verifying the presence of curb ramps; (iii) state-of-the-art CV algorithms for automatic curb ramp detection; and (iv) a machine learning-based workflow controller, which predicts CV performance and dynamically allocates work to either a human labeling pipeline or a CV + human verification pipeline. While Tohme is purely a data collection system, we envision future work that integrates Tohmes output into accessibility-aware map tools (e.g., a heatmap visualization of a citys accessibility or a smart navigation system that recommends accessible routes).

To evaluate Tohme, we conducted two studies using data we collected from 1,086 intersections across four North American cities. First, to validate the use of GSV imagery as a reliable source of curb ramp knowledge, we conducted physical audits in two of these cities and compared our results to GSV-based audit data. As with previous work exploring the concordance between GSV and the physical world [37,41,47–49], we found high correspondence between the virtual and physical audit data. Second, we evaluated Tohmes performance in detecting curb ramps across our entire dataset with 403 turkers. Alone, the computer vision sub-system currently finds 67% of the curb ramps in the GSV scenes. However, by dynamically allocating work to the CV module or to the slower but more accurate human workers, Tohme performs similarly in detecting curb ramps compared to a manual labeling approach alone (F-measure: 84% vs. 86% baseline) but at a 13% reduction in human time cost.

In summary, the primary contribution of this study is the design and evaluation of the Tohme system as a whole, with secondary contributions being: (i) the first design and evaluation of a computer vision system for automatically detecting curb ramps in images; (ii) the design and study of a "smart" workflow controller that dynamically allocates work based on predicted scene complexity from GIS data and CV output; (iii) a comparative physical vs. virtual curb ramp audit study (Study 1), which establishes that GSV is a viable data source for collecting curb ramp data; and (iv) a detailed examination of why curb ramp detection is a hard problem and opportunities for future work in this domain.

This work was done with Kotaro Hara, Prof. David Jacobs, and Prof. Jon Froehlich; it was published in Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14) [50]. At the time of this work, Deformable Parts Model was the best performing object detection algorithm. Later on, deep convolution neural networks based methods outperform others and take the competition to a new level. We investigate ways of encoding context information into a deep convolutional neural network model in Chapter 4.

3.2 Related Work

We describe work in sidewalk assessment, crowdsourcing, computer vision, and dynamic workflow allocation.

3.3 Crowdsourcing

Recently, Bigham et al. argued that current technological infrastructure provides unprecedented access to large sources of human power that can be harnessed to address accessibility challenges [51] (e.g., via crowdsourcing). Examples include VizWiz [52] and Legion:Scribe [53]. Most relevant to this chapter is the recent exploration of combining GSV and crowdsourcing for collecting street-level accessibility data including sidewalks [39], bus stops [41], and intersections [37]. Though this prior work demonstrates GSV as a potential accessibility data source, the studies do not examine semi-automatic methods (e.g., using machine learning or CV) as we do here.

Tohmes performance is contingent on crowd workers speed and accuracy in processing GSV imagery. Prior work exists in studying how to efficiently collect image labels(e.g., [54,55]). Su et al. investigated cost-performance tradeoff between majority vote based labeling and verification based data collection [55], finding quality control via verification improves cost-effectiveness. Recent work by Deng [54] explores methods of efficiently collecting multiclass image annotations by incorporating heuristics such as correlation, hierarchy, and sparsity (e.g., the presence of a keyboard in an image also suggests the presence of correlated objects such as mouse and monitor); however, to our knowledge, no prior work exists on efficiently collecting image labels from crowd workers on large panoramic imagery.

3.4 Computer Vision

There is a growing body of research applying CV techniques to GSV [56–60]. For example, Xiao et al.introduced automatic approaches to model 3D structures of streetscape and building faades using GSV [56, 57]. Zamir et al. showed that large-scale image localization, tracking, and commercial entity identification are possible [58–60]. This work demonstrates the potential of combining CV with GSV; however, automatically detecting curb ramps or other accessibility features has not been studied.

Tohme builds on top of existing object detection algorithms from the CV community [61–63]. For example, we use *Deformable Part Models* (DPMs) [62, 64], one of the top-performing approaches in the PASCAL Visual Object Classes (VOC) challenge, a major object detection and recognition competition [65]. Despite a decade-long effort, however, object detection remains an open problem [66, 67]. For example, even the DPM, which won the Lifetime Achievement Prize at the aforementioned PASCAL VOC challenge, has reached 30% precision and 70% recall in car detection [62]. Due to their variation in size, shape, and appearance, curb ramps are similarly difficult to detect. Consequently, we incorporate a "smart" workflow algorithm that attempts to predict poor CV performance and, in those instances, route work to human labelers.

3.5 Dynamic Workflow Allocation

Tohme uses machine learning to control its workflow for efficiently collecting data from GSV. Typical workflow adaptions include: varying the number of workers to recruit for a task [67, 68], assigning stronger workers to harder versions of a task [69], and/or fundamentally changing the task an individual worker is given [70, 71]. These workflow decisions are made automatically by workflow controllers often by analyzing worker performance history, inferring task difficulty, or estimating cost.

Most relevant to our work is workflow adaptation research in crowdsourcing systems [67, 71, 72]. For example, Lin et al. and Welinder et al. rely on worker performance histories to either assign different tasks [71] or recruit different numbers of workers [67]. More similar to our work is [70, 72] who infer task difficulty via automated methods and adapt work accordingly. For example, Kamaret al. [72] analyzed image features with CV algorithms to predict worker behaviors a priori on image annotation tasks and used this to dynamically decide the number of workers to recruit.

Though similar, our work is different both in problem domain (finding curb ramps) as well as in approach. Rather than vary the number of workers per task, our workflow controller infers CV performance and decides whether to use crowd worker labor for verifications or labeling. In addition, we do not simply rely on image features or CV output to determine workflow but also contextual information such as intersection complexity and 3D-point cloud data.

3.6 Dataset

Because sidewalk infrastructure can vary in quality, design, and appearance across geographic areas, our study sites include a range of neighborhoods from four North American cities: Washington DC, Baltimore, Los Angeles, and Saskatoon, Saskatchewan (Figure 3.2; Table 3.1). For each city, we collected data from dense urban cores (shown in blue) and semi-urban residential areas (shown in red). We emphasized neighborhoods with potential high demand for sidewalk accessibility (e.g., areas with schools, shopping centers, libraries, and medical clinics).

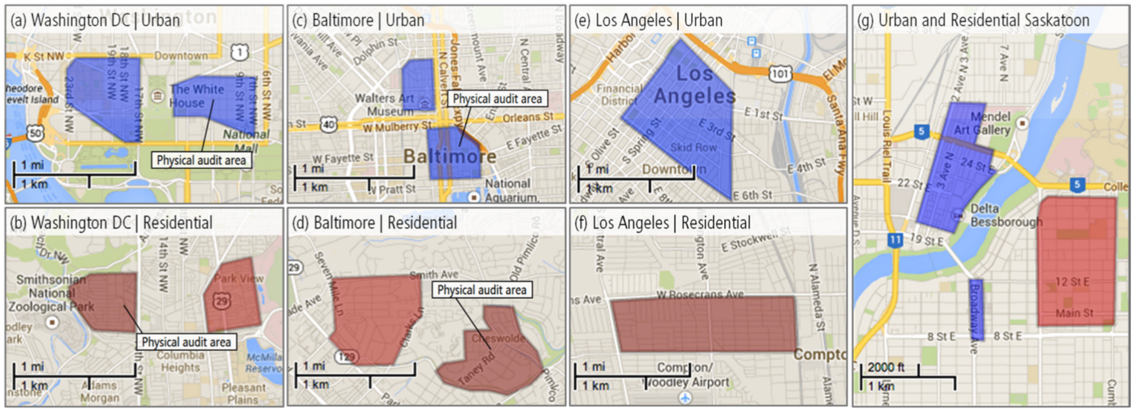

(a) Washington DC | Urban
Physical audit area
(b) Washington DC | Residential
Physical audit area
(c) Baltimore | Urban
Physical audit area
(d) Baltimore | Residential
Physical audit area
(e) Los Angeles | Urban
(f) Los Angeles | Residential
(g) Urban and Residential Saskatoon

Figure 3.2: The eight urban (blue) and residential (red) audit areas used in our studies from Washington DC, Baltimore, LA and Saskatoon. This includes 1,086 intersections across a total area of $11.3km^2$. Among these areas, we physically surveyed 273 intersections (see annotations in a-d).

We used two data collection approaches: (i) an automated web scraper tool that we developed called svCrawl, which downloads GIS-based intersection data,

	Washington DC		Baltimore		Los Angeles		Saskatoon		Overall
Region Type	Downtown	Residential	Downtown	Residential	Downtown	Residential	Downtown	Residential	
Total Area (km^2)	1.52	1.13	0.73	2.24	1.91	1.89	0.74	1.13	11.28
# of Intersections	140	124	132	139	132	132	141	146	1,086
# of Curb Ramps	818	352	476	229	358	186	321	137	2877
# of Missing Curb Ramps	8	35	32	69	43	214	24	222	647
Avg. GSV Data Age (SD)	1.9 yrs (0.77)	1.6 yrs (0.63)	2.1 yrs (0.75)	0.4 yrs (0.65)	2.0 yrs (0.31)	0.9 yrs (0.24)	4.0 yrs (0.0)	4.0 yrs (0.0)	2.2 (1.3)

Table 3.1: A breakdown of our eight audit areas. Age calculated from summer 2013. *These counts are based on ground truth data.

including GSV images, within a geographically defined region; and (ii) a physical surveyof a subset of our study sites (four neighborhoods totaling 273 intersections), which was used to validate curb ramp infrastructure found in the GSV images. In all, we used svCrawl to download data from 1,086 intersections across $11.3km^2$ (Table 3.1).

To create a ground truth dataset, two members of our research team independently labeled all 1,086 scenes using our custom labeling tool (svLabel). Label disagreements were resolved by consensus. From the ground truth data, we discovered 2,877 curb ramps and 647 missing curb ramps(Figure 3.3). Of the 1,086 scenes, 218 GSV scenes did not require marking a curb ramp or missing curb ramp because the location was not a traditional intersection (e.g., an alleyway with no vertical drop from the sidewalk). These 218 scenes are useful for exploring false positive labeling behavior and were kept in our dataset. The remaining 868 intersections had on average 3.3 curb ramps (SD=2.3) and 0.75 missing curb ramps (SD=1.3) per intersection. A total of 603/868 intersections were marked as not missing any curb ramps. We use the ground truth labels for training and testing our machine learning and CV algorithms and to evaluate crowd worker performance.

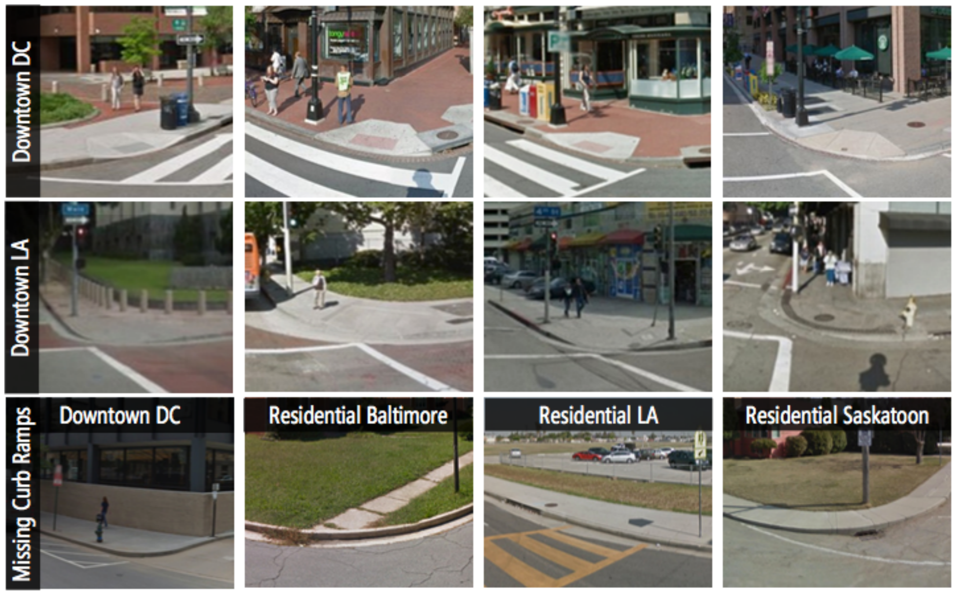


Figure 3.3: Example curb ramps (top two rows) and missing curb ramps (bottom row) from our GSV dataset.

At download time (summer 2013), the average age of the GSV images was 2.2 years (SD=1.3). As image age is one potential limitation in our approach, it is necessary to first show that GSV is a reasonable data source for deriving curb ramp information, which we do next.

3.7 Study 1: Assessing GSV as a Data Source

To establish GSV as a viable curb ramp data source, we must show: (i) that it presents unoccluded views of curb ramps, (ii) that the curb ramps can be reliably found by humans and, potentially, machines, and (iii) that the curb ramps found in GSV adequately reflect the state of the physical world. This study addresses each of these points. Multiple studies have previously demonstrated high concordance between GSV-based audits and audits conducted in the physical world [37,41,47,48]; however, prior work has not examined curb ramps specifically. Though this audit study was labor intensive, it is important to establish GSV as a reliable data source for curb ramp information, as it is the crux of our systems approach.

We conducted physical audits in the summer of 2013 across a subset of our GSV dataset: 273 intersections spanning urban and residential areas in Washington DC and Baltimore (Figure 3.2). We followed a physical audit process similar to Hara et al. [41]. Research team members physically visited each intersection, capturing geo-timestamped pictures (Mean=15 per intersection; SD=5). These images were analyzed post hoc for the actual audit. Surveying the 273 intersections took approximately 25 hours as calculated by image capture timestamps.

3.8 Auditing Methodology

For the auditing process itself, two additional research assistants (different from the above) independently counted the number of curb ramps and missing curb ramps at each intersection in both the physical and GSV image datasets. An initial visual codebook was composed based on US government standards for sidewalk accessibility [46,73]. Following the iterative coding method prescribed by Hruschka et al. [74], a small subset of the data was individually coded first (five intersections from each area). The coders then met, compared their count data, and updated the codebook appropriately to help reduce ambiguity in edge cases. Both datasets were then coded in entirety (including the original subset, which was recoded). This process was iterated until high agreement was reached.

3.9 Calculating Inter-Rater Reliability between Auditors

Before comparing the physical audit data to the GSV audit data, which is the primary goal of Study 1, we first calculated inter-rater reliability between the two coders for each dataset. We applied the Krippendorffs Alpha (α) statistical measure, which is used for calculating inter-rater reliability of count data (see [75]). Results after each of the three coding passes using the iterative scheme from [74] are shown in Table 3.2. Agreement was consistently high, withthe 3rd pass representing the reliability of codes in the final code set. There was initially greater inconsistency in coding missing curb ramps vs. coding existing curb ramps, perhaps because identifying a missing ramp requires a deeper understanding of the intersection and proper ramp placement.

3.10 Comparing Physical vs. GSV Audit Data

With high agreement verified within each dataset, we can now compare the count scores between the datasets. Similar to [41,49], we calculate a Spearman rank correlation between the two count sets (physical and GSV). This was done for both the curb ramp and missing curb ramp counts. To enable this calculation, however, we first merged the two auditors counts by taking the average of their counts for missing curb ramps and the average for present curb ramps at each intersection.

	Physical Audit Image Dataset			GSV Audit Image Dataset		
	1st Pass	2nd Pass	3rd Pass	1st Pass	2nd Pass	3rd Pass
Curb Ramp	0.959	0.960	0.989	0.927	0.928	0.989
Missing C. Ramp	0.647	0.802	0.999	0.631	0.788	0.999
Overall	0.897	0.931	0.996	0.883	0.917	0.996

Table 3.2: Krippendorff's alpha inter-rater agreement scores between two researchers on both the physical audit and GSV audit image datasets. Following Hruschka et al. iterative coding methodology, a 3rd audit pass was conducted with an updated codebook to achieve high-agreement scores.

Using these average counts, a Spearman rank correlation was computed, which shows high correspondence between datasets: ρ=0.996 for curb ramps and ρ=0.977 for missing curb ramps ($p < 0.001$). Overall, 1,008 curb ramps were identified in the virtual audit compared to 1,002 with the physical audit; differences were due to construction. The number of missing curb ramps was exactly the same for both datasets (89).

3.11 Study 1 Summary

Though the age of images in GSV remains a concern, Study 1 demonstrates that there is remarkably high concordance between curb ramp infrastructure in GSV and the physical world, even though the average image age of our dataset was 2.2 years. With GSV established as a curb ramp dataset source, we now move on to describing Tohme.

3.12 A Scalable System for Curb Ramp Detection

Tohme is a custom-designed tool for remotely collecting geo-located curb ramp information using a combination of crowdsourcing, CV, machine learning, and online map data.It is comprised of four parts depicted in Figure 3.4: (i) a web scraper, *Street View Crawl (svCrawl)*, for downloading street intersection data; (ii) two crowd worker interfaces for finding, labeling, and verifying the presence of curb ramps called *svLabel* and *svVerify*; (iii) state-of-the-art CValgorithms for automatically detecting curb ramps (*svDetect*); and (iv) a machine learning-based workflow, called *svControl*, which predicts CV performance on a scenes and allocates work accordingly.

We designed Tohme iteratively with small, informal pilot studies in our laboratory to test early interface ideas. We also performed larger experiments on Amazon Mechanical Turk (MTurk) with a subset of our data to understand how different interfaces affected crowd performance and, more generally, how well crowds could perform our tasks. The CV sub-system, svDetect, also evolved across multiple iterations, and was trained and evaluated using the aforementioned ground truth labels. While our ultimate goal is to deploy Tohme publicly on the web, the current prototype and experiments were deployed on MTurk. Below, we describe each Tohme sub-system.

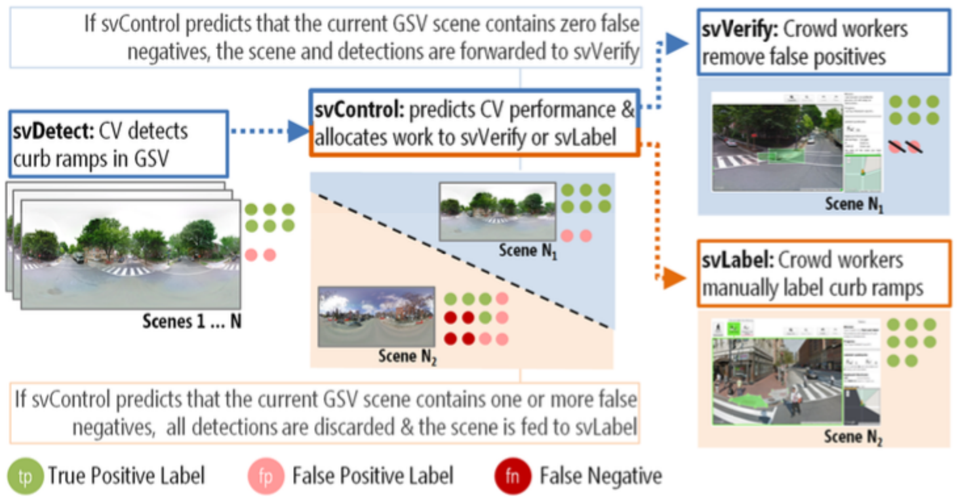

If svControl predicts that the current GSV scene contains zero false negatives, the scene and detections are forwarded to svVerify
svDetect: CV detects curb ramps in GSV
svControl: predicts CV performance & allocates work to svVerify or svLabel
svVerify: Crowd workers remove false positives
Scenes 1 ... N
Scene N1
Scene N2
svLabel: Crowd workers manually label curb ramps
If svControl predicts that the current GSV scene contains one or more false negatives, all detections are discarded & the scene is fed to svLabel
tp True Positive Label
fp False Positive Label
fn False Negative

Figure 3.4: A workflow diagram depicting Tohme's four main subsystems. In summary, svDetect processes every GSV scene producing curb ramp detections with confidence scores. svControl predicts whether the scene/detections contain a false negative. If so, the detections are discarded and the scene is fed to svLabel for manual labeling. If not, the scene/detections are forwarded to svVerify for verification. The workflow attempts to optimize accuracy and speed.

3.12.1 svCrawl: Automatic Intersection Scraping

svCrawl is a custom web scraper tool written in Python that downloads GIS-related intersection data over a predefined geographic region (Figure 3.2). It uses the Google Maps API (GMaps API) to enumerate and extract street intersection points within selected boundaries. For each intersection, svCrawl downloads four types of data:

1. **A GSV panoramic image** at its source resolution (13,312 x 6,656px). This is our primary data element (e.g., Figure 3.1).

2. **A 3D-point cloud**, which is captured by the GSV car using LiDAR [76]. The depth data overlays the GSV panorama but at a coarser resolution (512 x 256px; Figure 3.10). This is used by svDetect to automatically cull the visual search space and by svControl as an intersection complexity input feature.

3. **A top-down abstract map image** of the intersection obtained from the GMaps API (Figure 3.13), which is used as a training feature in our work scheduler, svControl, to infer intersection complexity (like the depth data).

4. **Associated intersection GIS metadata**, also provided by the GMaps API, such as latitude/longitude, GSV image age, street and city names, and intersection topology.

3.12.2 svLabel: Human-Powered GSV Image Labeling

In Tohme, intersections are labeled either manually, via svLabel, or automatically via svDetect. svLabel is a fully interactive online tool written in Javascript and PHP for finding and labeling curb ramps and missing curb ramps in GSV images (Figures 3.5,3.6,3.7). Unlike much previous crowd-sourcing GSV work, which uses static imagery to collect labels [37, 38, 72], our labeling tool builds on Bus Stop CSI [41] to provide a fully interactive 360 degree view of the GSV panoramic image. While this freedom increases user-interaction complexity, it allows the user to more naturally explore the intersection and maintain spatial context while searching for curb ramps. For example, the user can pan around the virtual 3D-space from one corner to the next within an intersection.

Using svLabel. When a turker accepts our HIT, they are immediately greeted by a three-stage interactive tutorial. The stages progressively teach the turker about the interface (e.g., the location of buttons and other widgets), user interactions (e.g., how to label, zoom, and pan), and task concepts (e.g., the definition of a curb ramp). If mistakes are made, our tutorial tool automatically provides corrective guidance. Turkers must successfully complete one tutorial stage before moving on to the next.

Once the tutorials are completed, we automatically position the turker in one of the audit area intersections and the labeling task begins in earnest. Similar to Bus Stop CSI, svLabel has two primary modes of interaction: Explorer Mode and Labeling Mode (Figure 3.6). When the user first drops into a scene, s/he defaults into Explorer Mode, which allows for exploration using Street Views native controls. Users are instructed to pan around to explore the 360 degree view of the intersection and visual feedback is provided to track their progress (bottom-right corner of Figure 3.6). Note: users movement is restricted to the drop location.

When the user clicks on either the Curb Ramp or Missing Curb Ramp buttons, the interface switches automatically to Labeling Mode. Here, mouse interactions no longer control the camera view. Instead, the cursor changes to a pen, allowing the user to draw an outline around the visual targeta curb ramp or lack thereof (Figure 3.5). We chose to have users outline the area rather than simply clicking or drawing a bounding box because the detailed outlines provide a higher degree of granularity for developing and experimenting with our CV algorithms. Once an outline is drawn, the user continues to search the intersection. Our tool automatically tracks the camera angle and repositions any applied labels in their correct location as the intersection view changes. In this way, the labels appear to "stick" to their associated targets. Once the user has surveyed the entire intersection by panning 360 degrees, s/he can submit the task and move on to the next task in the HIT, until all tasks are complete.

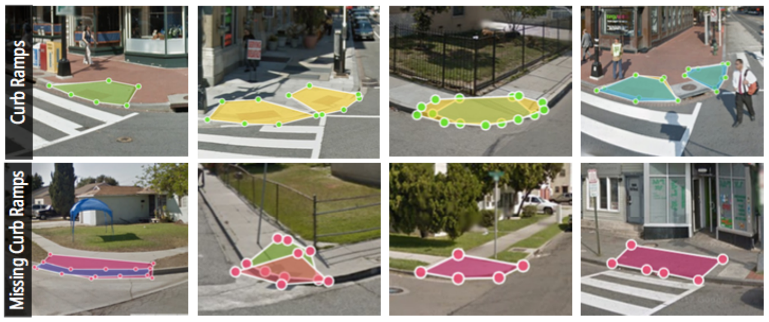
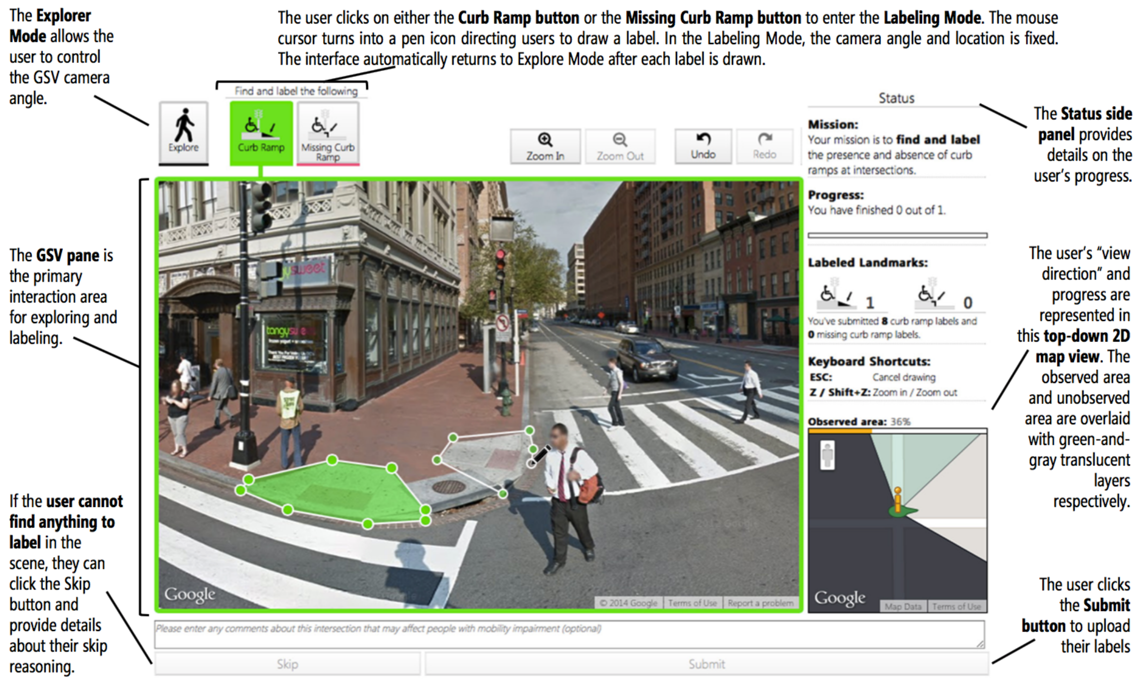


Figure 3.5: Example curb ramp and missing curb ramp labels from our turk studies. The green/pink outline points denote presence/absence.

Ground Truth Seeding. A single HIT is comprised of either five or six intersections depending on whether it contains a ground truth scene (a scene is just an intersection). This "ground truth seeding" [77] approach is commonly used to dynamically examine, provide feedback about, and improve worker performance. In our case, if a user makes a mistake at a ground truth scene, after hitting the submit button, we provide visual feedback about the error and show the proper corrective action. The user must correct all mistakes before submitting a ground truth task. If no mistakes are detected, the user is congratulated for their good performance. In our current system, there is a 50% chance that a HIT will contain one ground truth scene. The user is not able to tell whether they are working on a ground truth


The **Explorer Mode** allows the user to control the GSV camera angle.

The user clicks on either the **Curb Ramp button** or the **Missing Curb Ramp button** to enter the **Labeling Mode**. The mouse cursor turns into a pen icon directing users to draw a label. In the Labeling Mode, the camera angle and location is fixed. The interface automatically returns to Explore Mode after each label is drawn.

The **Status side panel** provides details on the user's progress.

The **GSV pane** is the primary interaction area for exploring and labeling.

The user's "view direction" and progress are represented in this **top-down 2D map view**. The observed area and unobserved area are overlaid with green-and-gray translucent layers respectively.

If the **user cannot find anything to label** in the scene, they can click the Skip button and provide details about their skip reasoning.

The user clicks the **Submit button** to upload their labels

Status

Mission: Your mission is to **find and label** the presence and absence of curb ramps at intersections.

Progress: You have finished 0 out of 1.

Labeled Landmarks: 1 0

You've submitted **8** curb ramp labels and **0** missing curb ramp labels.

Keyboard Shortcuts:
ESC: Cancel drawing
Z / Shift+Z: Zoom in / Zoom out

Observed area: 36%

Please enter any comments about this intersection that may affect people with mobility impairment (optional)

Figure 3.6: The svLabel interface. Crowd workers use the Explorer Mode to interactively explore the intersection (via pan and zoom) and switch to the Labeling Mode to label curb ramps and missing curb ramps. Clicking the Submit button uploads the target labels. The turker is then transported to a new location unless the HIT is complete.

scene until after they submit their work.

3.12.3 svVerify: Human-Powered GSV Label Verification

In addition to providing "curb ramp" and "missing curb ramp" labels, we rely on crowd workers to examine and verify the correctness of previously entered labels. This verification step is common in crowdsourcing systems to increase result quality (e.g., [39, 55]). svVerify (Figure 3.8) is similar to svLabel in appearance and general

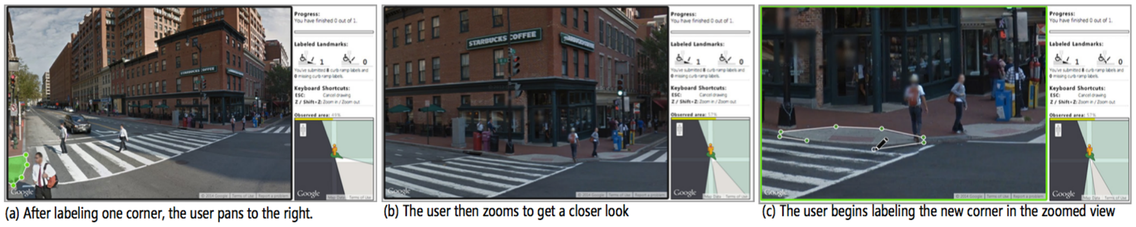

(a) After labeling one corner, the user pans to the right. (b) The user then zooms to get a closer look (c) The user begins labeling the new corner in the zoomed view

Figure 3.7: svLabel automatically tracks the camera angle and repositions any applied labels in their correct location as the view changes. When the turker pans the scene, the overlay on the map view is updated and the green 'explored' area increases (bottom right of interface). Turkers can zoom in up to two levels to inspect distant corners. Labels can be applied at any zoom level and are scaled appropriately.

workflow but has a simplified interaction (clicking and panning only) and is for an easier task (clicking on incorrect labels).

While we designed both svLabel and svVerify to maximize worker efficiency and accuracy, our expectation was that the verification task would be significantly faster than initially providing manual labels [55]. For verification, users need not perform a time-consuming visual search looking for curb ramps to label but rather can quickly scan for incorrect labels (false positives) to delete. And, unlike labeling, which requires drawing polygonal outlines, the delete interaction is a single click over the offending label. This enables users to rapidly eliminate false positive labels in a scene.

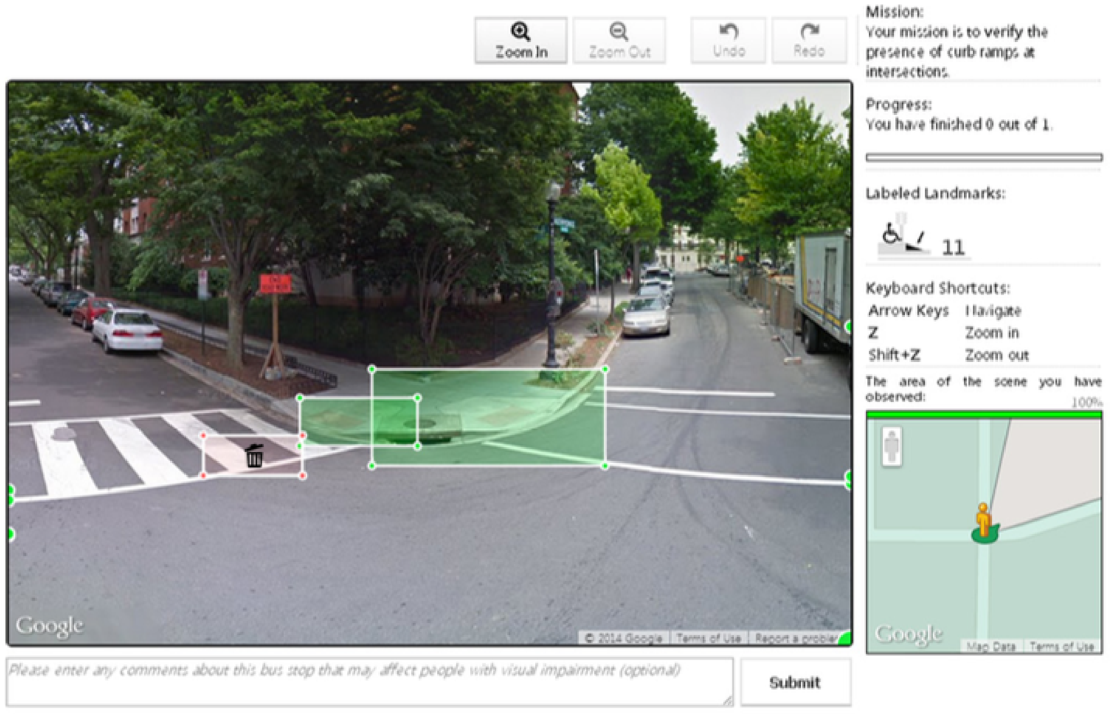
To maintain verification efficiency, however, we did not allow the user to spatially locate false negatives. This would essentially turn the verification task into a labeling task, by asking users to apply new "curb ramp" or "curb ramp missing" labels when they noticed a valid location that had not been labeled. Instead, svVerify gathers information on false negatives at a coarser-grained level by asking the user if the current scene was missing any labels after s/he clicks the submit button. Thus, svVerify can detect the presence of false negatives in an intersection but not their specific location or quantity.

Similar to svLabel, svVerify requires turkers to complete an interactive tutorial before beginning a HIT, which includes instructions about the task, the interface itself, and successfully verifying one intersection. Because verifications are faster than providing labels, we included 10 scenes in each HIT (vs. the 5 or 6 in svLabel). In addition, we inserted one ground truth scene into everysvVerify HIT rather than with 50% probability as was done with svLabel. Note that not all scenes are sent to svVerify for verification, as discussed in the svControl section below. We move now to describing the two more technical parts of Tohme: svDetect and svControl.

3.12.4 svDetect: Detecting Curb Ramps Automatically

While svLabel relies on manual labeling for finding curb ramps, svDetect attempts to do this automatically using CV. Because CV-based object detection is still an open problem–even for well-studied targets such as cars [64] and people [61]–our goal is to create a system that functions well enough to reduce the cost of curb ramp detection vs. a manual approach alone.

svDetect uses a three-stage detection process. First, we train a Deformable Part Model (DPM), one of the most successful recent approaches in object detection (e.g., [65]), as a first-pass curb ramp detector. Second, we post-process the resulting


Mission:
Your mission is to verify the presence of curb ramps at intersections.

Progress:
You have finished 0 out of 1.

Labeled Landmarks:
11

Keyboard Shortcuts:

Arrow Keys	Navigate
Z	Zoom in
Shift+Z	Zoom out

The area of the scene you have observed:
100%

Please enter any comments about this bus stop that may affect people with visual impairment (optional)

Figure 3.8: The svVerify interface is similar to svLabel but is designed for verifying rather than labeling. When the mouse hovers over a label, the cursor changes to a garbage can and a click removes the label. The user must pan 360 degrees before submitting the task.

bounding boxes using non-maximum suppression [78] and 3D-point cloud data to eliminate detector redundancies and false positives. Finally, the remaining bounding boxes are classified using a Support Vector Machine (SVM) [79], which uses features not leveraged by the DPM, further eliminating false positives.

svDetect was designed and tested iteratively. We attempted multiple algorithmic approaches and used preliminary experiments to guide and refine our approach. For example, we previously used a linear SVM with a Histograms of Oriented Gradients (HOG) feature descriptor [80] but found that the DPM was able to recognize curb ramps with larger variations. In addition, we found that though the raw GSV image size is 13,312 x 6,656 pixels, there were no detection performance benefits beyond 4,096 x 2,048px (the resolution used throughout this chapter). Because it helps explain our design rationale for Tohme, we include our evaluation experiments for svDetect in this section rather than later in the chapter.

First Stage: The Curb Ramp Deformable Part Model (DPM)

DPMs are comprised of two parts: a coarse-grained model, called a root filter, and a higher resolution parts model, called a parts filter. DPMs are commonly applied to human detection in images, which provides a useful example. For human detection, the root filter captures the whole human body while part filters are for individual body parts such as the head, hand, and legs [62]. The individual parts are learned automatically by the DPM–that is, they are not explicitly defined a priori. In addition, how these parts can be positioned around the body (the root filter) is also learned and modeled via displacement costs. This allows a DPM to recognize different configurations of the human body (e.g., sitting vs. standing).

In our case, the root filter describes the general appearance of a curb ramp while part filters account for individual components (e.g., edges of the ramp and transitions to the road). DPM creates multiple components for a single model (Figure 3.9) based on bounding box aspect ratios. We suspect that each component implicitly captures different viewpoints of a curb ramp. For our DPM, we used code provided by [81].

Second Stage: Post-Processing DPM Output

In the second stage, we post-process the DPM output in two ways. First,

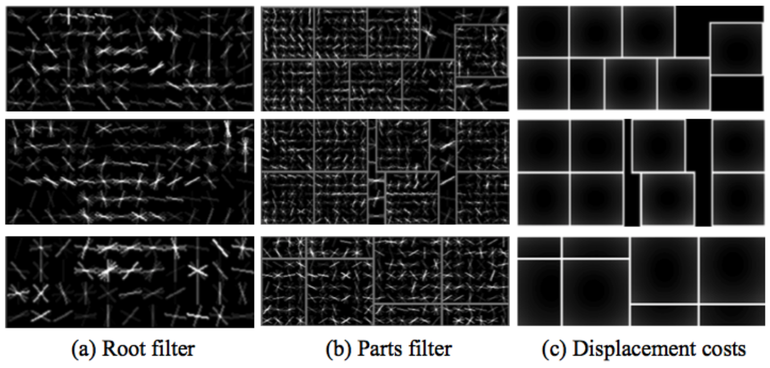


Figure 3.9: The trained curb ramp DPM model. Each row represents an automatically learned viewpoint variation. The root and parts filter visualize learned weights for the gradient features. The displacement costs for parts are shown in (c).

similar to [78], we use non-maximum suppression (NMS) to eliminate redundant bounding boxes. NMS is common in CV and works by greedily selecting bounding boxes with high confidence values and removing overlapping boxes with lower scores. Overlap is defined as the ratio of intersection of the two bounding boxes over the union of those boxes. Based on the criteria established by the PASCAL Visual Object Classes challenge [65], we set our NMS overlap threshold to 50%.

Our second post-processing step uses the 3D-point cloud data to eliminate curb ramp detections that occur above the ground plane (e.g., bounding boxes in the sky are removed). To do so, the 512 x 256px depth image is resized to the GSV image size (4096 x 2048px) using bilinear interpolation. For each pixel, we calculate a normal vector and generate a mask for those pixels with a strong vertical component. These pixels correspond to the ground plane. Bounding boxes outside of this pixel mask are eliminated(Figure 3.10 and 3.11).

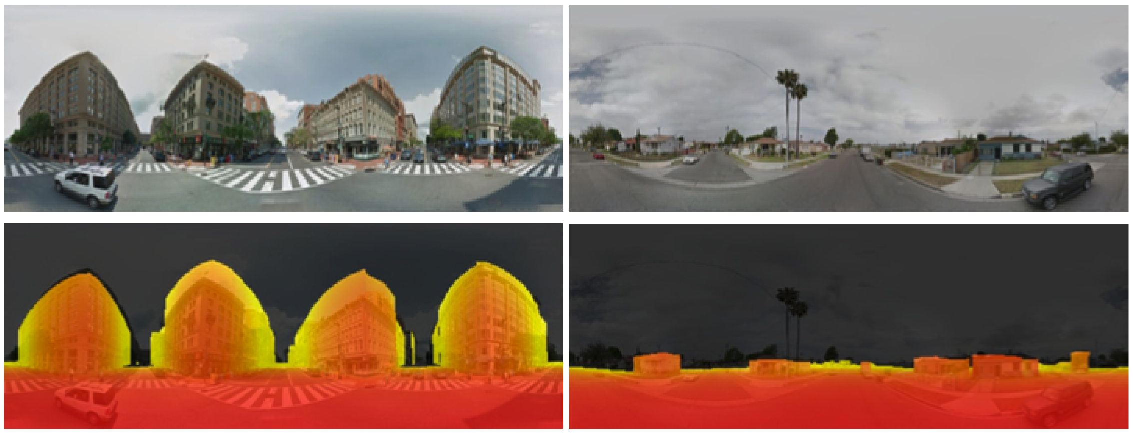


Figure 3.10: Using code from [38], we download GSV's 3D-point cloud data and use this to create a ground plane mask to post-process DPM output. The 3D depth data is coarse: 512 x 256px.

Third Stage: SVM-Based Classification

Finally, in the third stage, the remaining bounding boxes are fed into an additional classifier: an SVM. Because the DPM relies solely on gradient features in an image, it does not utilize other important discriminable information such as color or position of the bounding box. Given that street intersections have highly constrained geometrical configurations, curb ramps tend to occur in similar locations-so detection position is important. Thus, for each bounding box, we create a feature vector that includes: RGB color histograms, the top-left and bottom-right corner coordinates of the bounding box in the GSV image along with its width and height, and the detection confidence score from the DPM detector. We use the SVM as a binary classifier to keep or discard detection results from the second stage.

svDetect Training and Results

Two of the three svDetect stages require training: the DPM in Stage 1 and the

SVM in Stage 3. For training and testing, we used two-fold cross validation across the 1,086 GSVscenes and 2,877 ground truth curb ramp labels. The GSV scenes were randomly split in half (543 scenes per fold) with one fold initially assigned for training and the other for testing. This process was then repeated with the training and testing folds switched.

To train the DPM (Stage 1), we transform the polygonal ground truth labels into rectangular bounding boxes, which are used as positive training examples. DPM uses a sliding window approach, so the rest of the GSV scene is treated as negative examples (i.e., comprised of negative windows). For each image in the training set, the DPM produces a set of bounding boxes with associated confidence scores. The number of bounding boxes produced per scene is contingent on a minimum score threshold. This threshold is often learned empirically. A high threshold would produce a small number of bounding boxes, which would likely result in high precision and low recall; a low threshold would likely lead to low precision and high recall.

To train the SVM (Stage 3), we use the post-processed DPM bounding boxes from Stage 2. The bounding boxes are partitioned into positive and negative samples by calculating area overlap with the ground truth labels. Though there is no universal standard for evaluating good area overlap in object detection research, we use 20%overlap (from [82]). Prior work suggests that even 10-15% overlap agreement at the pixel level would be sufficient to confidently localize accessibility problems in images [39]. Thus, positive samples are boxes that overlap with ground truth by more than 20%; negative samples are all other boxes. We extract the aforementioned training features from both the positive and negative bounding boxes. Note that SVM parameters (e.g., coefficient for slack variables) are automatically selected by grid search during training.

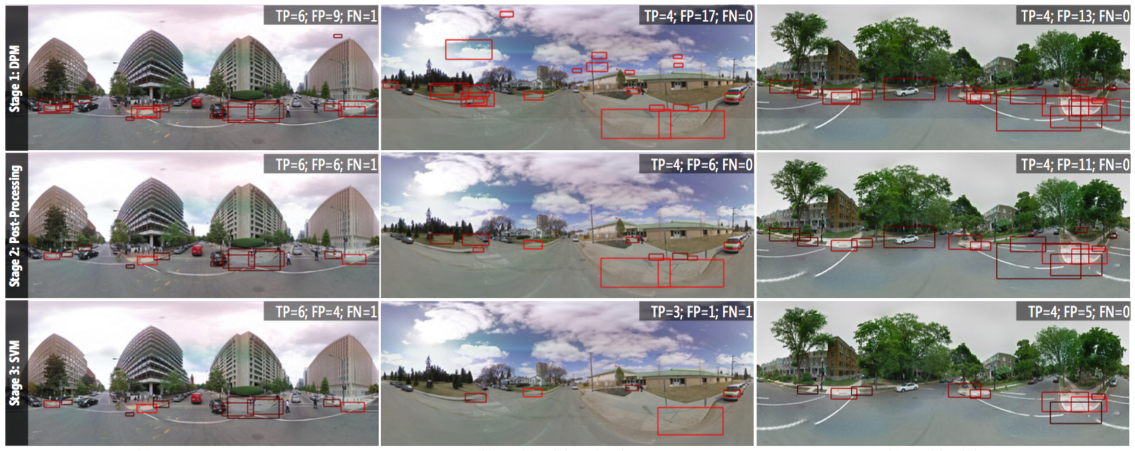


Figure 3.11: Example results from svDetects three-stage curb ramp detection framework. Bounding boxes are colored by confidence score (lighter is higher confidence). As this figure illustrates, setting the detection threshold to -0.99 results in a relatively low false negative rate at a cost of a high false positive rate (false negatives are more expensive to correct). Many false positives are eliminated in Stages 2 and 3. The effect of Stage 2s ground plane mask is evident in (b). Acronyms: TP=true positive; FP=false positive; FN=false negative.

Results. To analyze svDetects overall performance and to determine an appropriate confidence score cutoff for svDetect, we stepped through various DPM detection thresholds (from -3-to -3 with a 0.01 step) and measured the results. For each threshold, we calculated true positive, false positive, and false negative detections for each scene. True positives were assessed as bounding boxes that had

20% overlap with ground truth labels and that had a detection score higher than the currently set threshold. The results are graphed on a precision-recall curve in Figure 3.12. To balance the number of true positive detections and false positives in our system, we selected a DPM detection threshold of -0.99. At this threshold, svDetect generates an average of 7.0 bounding boxes per intersection (SD=3.7); see Figure 3.11 for examples. Note: svDetect failed to generate a bounding box for 15 of the 1,086 intersections. These are still included in our performance comparison.

In the ideal, our three-stage detection framework would have both high precision and high recall. As can be observed in Figure 3.12, this is obviously not the case as 20% of the curb ramps are never detected (i.e., the recall metric never breaches 80%). With that said, automatically finding curb ramps using CV is a hard problem due to viewpoint variation, illumination, and within/between class variation. This is why Tohme combines automation with manual labor using svControl.

svControl: Scheduling Work via Performance Prediction

svControl is a machine-learning module for predicting CV performance and assigning work to either a manual labor pipeline (svLabel) or an automated pipeline with human verification (svDetect + svVerify)-see Figure 3.4. We designed svControl based on three principles: first, that human-based verifications are fast and relatively low-cost compared to human-based labeling; second, CV is fast and inexpensive but error prone both in producing high false positives and false negatives; third, false negatives are more expensive to correct than false positives.

From these principles, we derived two overarching design questions: first, given the high cost of human labeling and relative low-cost of human verification, could

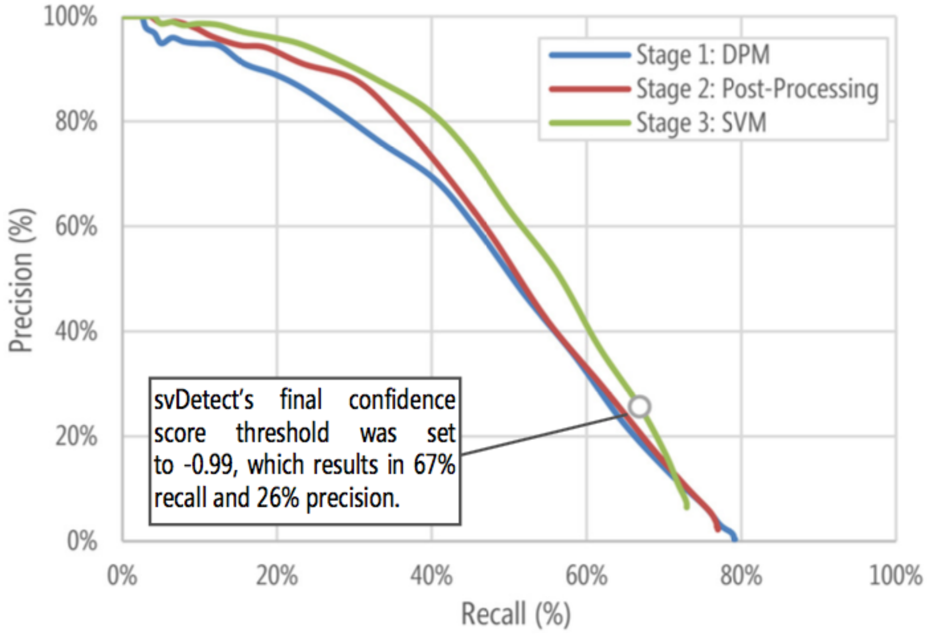


Figure 3.12: The precision-recall curve of the three-stage curb ramp detection process constructed by stepping through various DPM detection thresholds (from -3-to-3 with a 0.01 step). For the final svDetect module, we selected a DPM detection threshold of -0.99, which balances true positive detections with false positives.

we optimize CV performance with a bias towards a low false negative rate (even if it meant an increase in false positives)? Second, given that false negatives cannot be eliminated completely from svDetect, can we predict their occurrence based on features of an intersection and use this to divert work to svLabel instead for human labeling?

Towards the first question, biasing CV performance towards a certain rate of false negatives is trivial. It is simply a matter of selecting the appropriate threshold on the precision/recall curve (recall that the threshold that we selected was -0.99). The second question is more complex. We iterated over a number of prediction techniques and intersection features before settling on a linear SVM and Lasso regression model [83] with the following three types of input features:

- **svDetect results (16 features):** For each GSV image, we include the raw number of bounding boxes output from svDetect, the average, median, standard deviation, and range of confidence scores of all bounding boxes in the image, and descriptive statistics for their XY-coordinates. Importantly, we did not use the correctness of the bounding box as a feature since this would be unknown during testing.

- **Intersection complexity (2 features)**:We calculate intersection complexity via two measures: cardinality (i.e.,how many streets are connected to the target intersection) and an indirect measure of complexity, for which we count the number of street pixels in a stylized top-down Google Map. We found that high pixel counts correlate to high intersection complexity (Figure 3.13).

- **3D-point cloud data (5 features):** svDetect struggles to detect curb ramps that are distant in a scene-e.g., because the intersection is large or because the GSV car is in a sub-optimal position to photograph the intersection. Thus, we include descriptive statistics of depth information of each scene (e.g., average, median, variance).

We combine the above features into a single 23-dimensional feature vector for training and classification.

svControl Training and Test Results

We train and test svControl with two-fold cross validation using the same train and test data as used for svDetect. Given that the goal of svControl is to predict svDetect performance, namely the occurrence of false negatives, we define a svDetect failure as a GSV scene with at least one false negative curb ramp detection. The SVM model is trained to make a binary failure prediction with the aforementioned features. Similarly, the Lasso regression model is trained to predict the raw number of false negatives of svDetect (regression value > 0.5 is failure).

To help better understand the important features in our models, we present the top three correlation coefficients for both. For the SVM, the top coefficients were the labels x-coordinate variance (0.91), the mean confidence score of automatically detected labels (0.69), and the minimum scene depth (0.67). For the Lasso model, the top three were mean scene depth (0.69), median scene depth (-0.28), and, similar to the SVM, the mean confidence score of the automatically detected labels (0.21). If either the SVM or the Lasso model predicts failure on a particular GSV scene, svControl routes that scene to svLabel instead of svVerify.

svControl Results. We assessed svControls prediction performance across the 1,086 scenes. While not perfect, our results show that svControl is capable of identifying svDetect failures with high probabilitywe correctly predicted 397 of the 439 svDetect failures (86.3%); however, this high recall comes at a cost of precision: 404 of the total 801 scenes (50.4%) marked as failures were false positives. Given that we designed svControl to be conservative (i.e., pass more work to svLabel if in doubt about svDetect), this accuracy balance is reasonable. Below, we examine whether this is sufficient to provide performance benefits for Tohme.

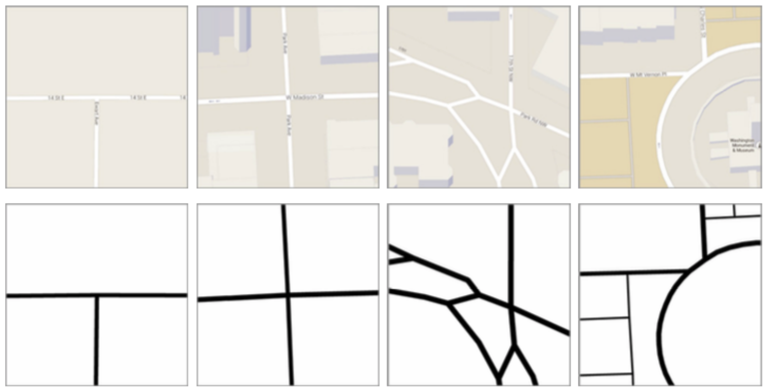


Figure 3.13: We use top-down stylized Google Maps (bottom row) to infer intersection complexity by counting black pixels (streets) in each scene. A higher count correlates to higher complexity.

3.13 Study 2: Evaluating Tohme

To examine the effectiveness of Tohme for finding curb ramps in GSV images and to compare its performance to a baseline approach, we performed an online study with MTurk in spring 2014. Our goal here is threefold: first, and most importantly, to investigate whether Tohme provides performance benefits over manual labeling alone (baseline); second, to understand the effectiveness of each of Tohmes sub-systems (svLabel, svVerify, svDetect, and svControl); and third, to uncover directions for future work in preparation for a public deployment.

3.13.1 Tohme Study Method

Similar to Hara et al. [39], we collected more data than necessary in practice so that we could simulate performance with different workflow configurations *post* *hoc.* To allow us to compare Tohme vs. feeding all scenes to either workflow on their own (svLabel and svDetect+svVerify), we ran all GSV scenes through both. To avoid interaction effects, turkers hired for one workflow (labeling) could not work on the other (verifying) and vice versa.

Second, to more rigorously assess Tohme and to reduce the influence of any one turker on our results, we hired at least three turkers per scene for each workflow and used this data to perform Monte Carlo simulations. More specifically, for both workflows, we randomly sampled one turker from each scene, calculated performance statistics (e.g., precision), and repeated this process 1,000 times. Admittedly, this is a more complex evaluation than simply hiring one turker per scene and computing the results; however, the Monte Carlo simulation allows us to derive a more robust indicator of Tohmes expected future performance.

Of the 1,086 GSV scenes (street intersections) in our dataset, we reserved 40 for ground truth seeding, which were randomly selected from the eight geographic areas (5 scenes from each). We calculated HIT payment rates based on MTurk pilot studies: \$0.80 for svLabel HITs (five intersections; \$0.16 per intersection) and \$0.80 for svVerify (ten intersections; \$0.08 per intersection). As noted in our system description, turkers had to successfully complete interactive tutorials before beginning the tasks.

3.13.2 Analysis Metrics

To assess Tohme, we used the following measures:

- **Label overlap compared to ground truth:** as described in the svDetect section, we use 20% overlap as our correctness threshold (from [39]).
- **We calculate standard object detection performance metrics** including precision, recall, and F-measure based on this 20% area overlapthe same overlap used by svDetect.
- **Human time cost**: cost is calculated by measuring completion times for each intersection in svLabel and svVerify.

3.13.3 Tohme Study Results

We first present high-level descriptive statistics of the MTurk HITs before focusing on the comparison between Tohme vs. our baseline approach (pure manual labeling with svLabel). We provide additional analyses that help explain the underlying trends in our results.

Descriptive Statistics of MTurk Work

To gather data for our analyses, we hired 242 distinct turkers for the svLabel pipeline and 161 turkers for the svVerify pipeline (Table 3.3). As noted previously, all 1,046 GSV scenes were fed through both workflows. For svLabel, turkers completed 1,270 HITs (6,350 labeling tasks) providing 17,327 curb ramp labels and 3,462 missing curb ramp labels. For svVerify, turkers completed 582 HITs (5,820 verification tasks) and verified a total of 42,226 curb ramp labels. On average, turkers eliminated 4.9 labels per intersection (SD=2.9). We hired an average of 6.1 (SD=0.6) turkers per intersection for svLabel and 5.6 (SD=0.6) for svVerify.

	Turkers	GSV Scenes	HITs	Tasks	Avg. Turkers / Intersection	Label Stats	Avg. Task Time
svLabel	242	1046	1270	6350	6.1 (0.6)	20789 labels	94.1s (144.4s)
svVerify	161	1046	582	5820	5.6 (0.6)	42226 verified labels	43.2s (48.7s)

Table 3.3: An overview of the MTurk svLabel and svVerify HITs. While Tohme's svControl system would, in practice, split work between the svLabel and svDetect+svVerify pipelines, we fed every GSV scene to both to perform our analyses. svVerify was 2.2x faster than svLabel.

Evaluating Tohmes Performance

To evaluate Tohmes overall performance, we first examined how well each pipeline would perform on its own across the entire dataset (1,046 scenes). This provides two baselines for comparison: (i) the svDetect + svVerifyresults show how well Tohme would perform if the svControl module passed all work to this pipeline and, similarly, (ii) the svLabel results show what would happen if we only relied on manual labor for finding and labeling curb ramps.

We found that Tohme achieved similar but slightly lower curb ramp detection results compared to the manual approach alone (F-measure: 84% vs. 86%) but with a much lower time cost (13% reduction); see Figure 3.14. As expected, while the svDetect + svVerify pipeline is relatively inexpensive, it performed the worst (F-measure: 63%). These findings show that the svControl module routed work appropriately to maintain high accuracy but at a reduced cost. Tohme reduces the average per-scene processing time by 12 seconds compared to svLabel alone.The overall task completion times were 12.3, 27.3, and 23.7 hours for svDetect + svVerify, svLabel, and Tohme respectively.

The above results were calculated using the aforementioned Monte Carlo method. If we, instead, use only the first turker to arrive and complete the task, our results are largely the same. The F-measures are 63%, 86%, and 85% respectively for svDetect + svVerify, svLabel, and Tohme with a 10% drop in cost for Tohme (rather than 13%). This includes 65 distinct turkers for svDetect + svVerify, 97 for svLabel, and 149 for Tohme.

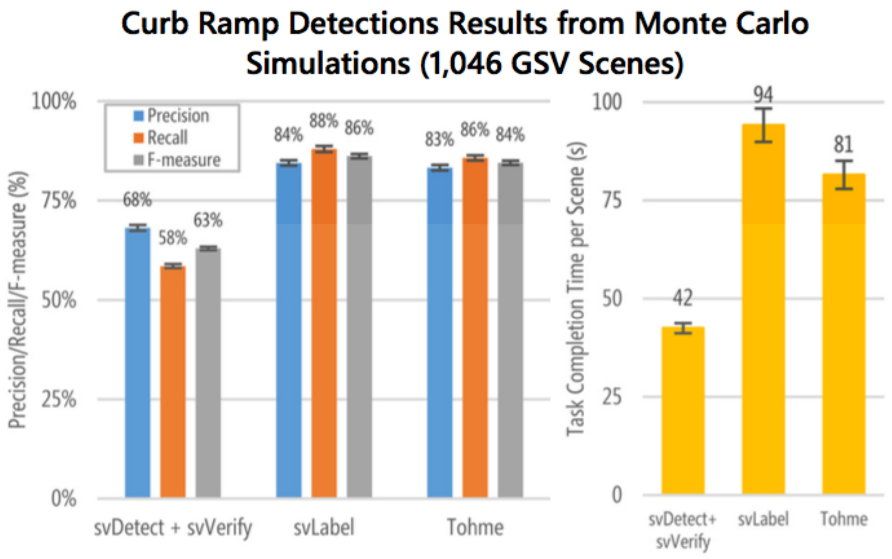


Figure 3.14: Tohme achieves comparable results to a manual labeling approach alone but with a 13% reduction in time cost. Error bars are standard deviation.

Task Allocation by svControl

As the workflow scheduler, the svControl module is a critical component of Tohme. Because the svVerify interface does not allow for labeling (e.g., correcting false negatives), the svControl system is conservativeit routes most of the work to svLabel otherwise many curb ramps would possibly remain undetected. Of the 1,046 scenes, svControl predicted svDetect to fail on 769 scenes (these results are the same as presented in the svControl section but with the 40 ground truth scenes removed). Thus, 73.5% of all scenes were routed to svLabel for manual work and the rest (277) were fed to svVerify for human verification (Figure 3.15). Again, svControls true positive rate is high: 86%. However, if svControl worked as a perfect classifier, 439 scenes would have been forwarded to svLabel and 607 to svVerify. In this idealized case, Tohmes cost drops to 27.7% compared to a manual labeling approach with the same F-measure as before (84%). Thus, assuming limited improvements in CV-based curb ramp detections in the near future, a key area for future work will be improving the workflow control system.

Where Humans and Computers Struggle

The key to improving both CV and human labeling performance is to understand where and why each sub-system makes mistakes. To assess the detection accuracy of human labelers, we calculated the average F-measure score per scene based on the average number of true positives(TP), false positives (FP), and false negatives (FN). For example, if the average for a scene was (TP, FP, FN) = (1, 1, 2), then (Precision, Recall, F-measure) = (0.5, 0.3, 0.4). For CV, we simply used the F-measure score for each scene based on our svDetect results. We sorted the two F-measure lists and visually inspected the best and worst performing scenes for each. For the top and bottom 10, the average F-measure scores were 99% and 0% for CV and 100% and 25% for human labeling respectively. Common problems are summarized in Figure 3.16.

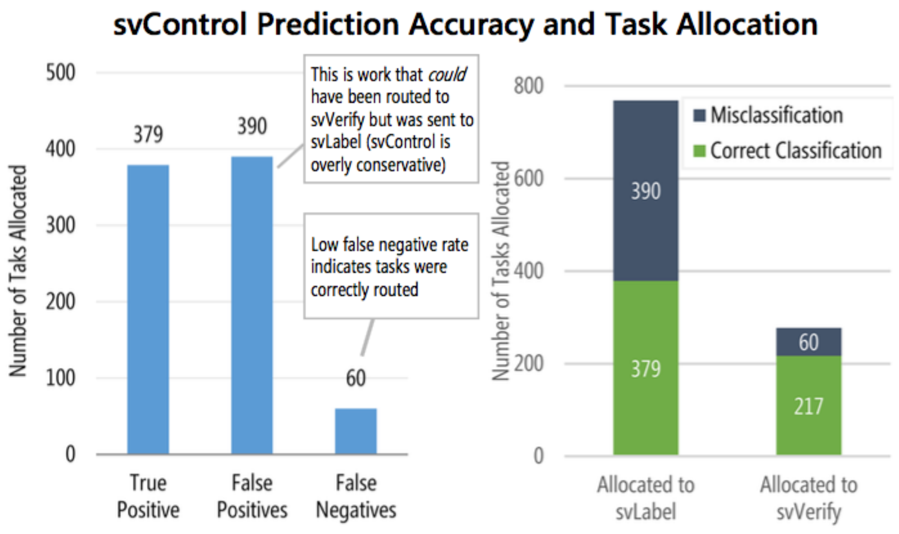


Figure 3.15: svControl allocated 769 scenes to svLabel and 277 scenes to svVerify. 379 out of 439 scenes (86.3%) where svDetect failed were allocated 'correctly' to svLabel. Recall that svControl is conservative in routing work to svVerify because false negative labels are expensive to correct; thus, the 86.3% comes at a high false positive cost (390).

Crowd workers struggled with labeling distant curb ramps (scale) or due to placement and angle (viewpoint variation). To mitigate this, future labeling interfaces could allow the worker to "walk" around the intersection to select better viewpoints (similar to [25]); however, this will increase user-interaction complexity and labeling time. Perhaps as should be expected, crowd workers were much more adept at dealing with occlusion than CVeven if a majority of a curb ramp was occluded, a worker could infer its location and shape (e.g., middle occlusion picture). CV struggled for all the reasons noted in Figure 3.16. Given the tremendous variation in curb ramp design and capture angles, a larger training set may have improved our results. Moreover, because multiple views of a single intersection are available n GSV via neighboring panoramas, these additional perspectives could be combined to potentially improve scene structure understanding and mitigate issues with occlusion, illumination, scale, and viewpoint variation. The semantic issues-e.g., confusing structures similar to curb ramps-are obviously much more difficult for CV than humans. We describe other areas for improvement in the Discussion.

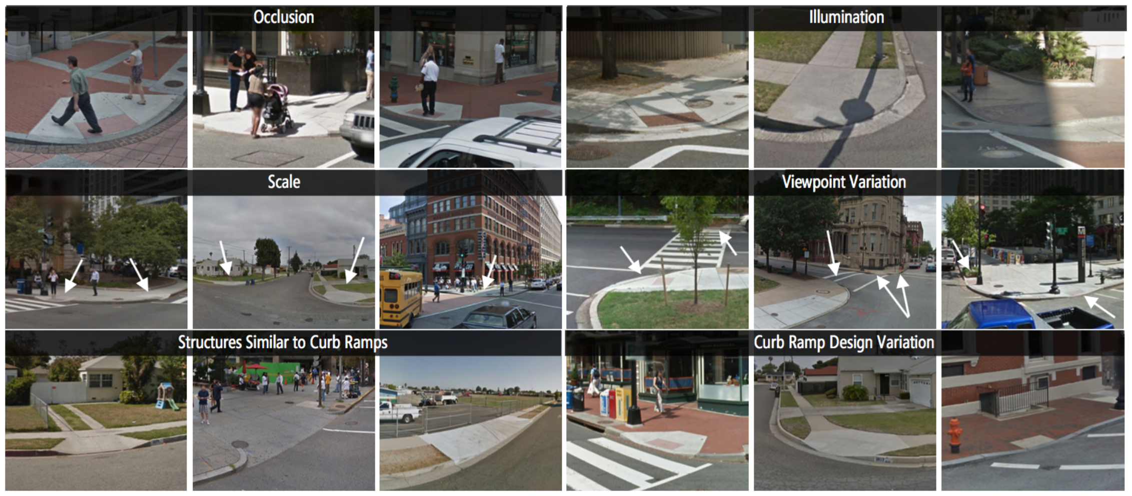


Figure 3.16: Finding curb ramps in GSV imagery can be difficult. Common problems include occlusion, illumination, scale differences because of distance, viewpoint variation (side, front, back), between class similarity, and within class variation. For between class similarity, many structures exist in the physical world that appear similar to curb ramps but are not. For within class variation, there are a wide variety of curb ramp designs that vary in appearance. White arrows are used in some images to draw attention to curb ramps. Some images contain multiple problems.

Effect of Area Overlap Threshold on Performance

As noted previously, there is no universal standard for selecting an area overlap threshold in CV; this decision is often domain dependent. To investigate the effect of changing the overlap threshold on performance, we measured precision, recall, and F-measure at different values from 0-50% at a step size of 10% (Figure 3.17). For overlap=0%, at least 1px of a detected bounding must overlap with a ground truth label to be considered correct.

A few observations: first, as expected, performance decreases as the overlap threshold increases; however, the relative performance difference between Tohme and baseline (svLabel) stays roughly the same. For example, at 0% overlap, the (Precision, Recall, F-measure) of Tohme is (85%, 89%, 87%) and (86%, 90%, 88%) for svLabel and at 50% overlap, (54%, 55%, 55%) vs. (57%, 59%, 58%).Thus, Tohmes relative performance is consistent regardless of overlap threshold (i.e., slightly poorer performance but cheaper). Second, there appears to be a more substantial performance drop starting at 30%, which suggests that obtaining curb ramp label agreement at the pixel level between human labelers and ground truth after this point is difficult. Finally, though svDetect + svVerify has much greater precision than svDetect alone, this increase comes at a cost of recalla gap which widens as the overlap threshold becomes more aggressive. So, though human verifiers help increase precision, they are imperfect and sometimes delete true positive labels.

3.14 Discussion

Our research advances recent work using GSV and crowdsourcing to remotely collect data on accessibility features of the physical world (e.g., [37–39, 41]) by in-

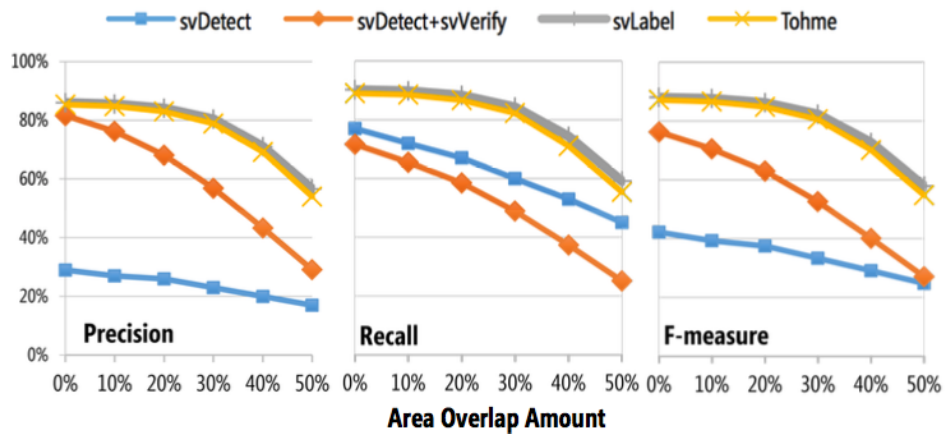


Figure 3.17: As expected, performance drops as the area overlap threshold increases; however, the relative difference between Tohme and baseline (svLabel) remains consistent.

tegrating CV and a machine learning-based workflow scheduler. We showed that a trained CV-based curb ramp detector (svDetect) found 63% of curb ramps in GSV scenes and fast, human-based verifications further improved the overall results. We also demonstrated that a novel machine-learning based workflow controller, svControl, could predict CV performance and route work accordingly. Below, we discuss limitations and opportunities for future work. Note that the future works discussed in this section is a general discussion on possible research directions to extending Tohme. The future works that are closely relevant to my thesis is summarized in Chapter 6.

3.14.1 Improving Human Interfaces

How much context is necessary for verification? We were surprised that verification tasks were only 2.2x faster than labeling tasks. Though we attempted to design both interfaces for rapid user interaction, there is some basic overhead incurred by panning and searching in the 360-degree GSV view. In an attempt to eliminate this overhead, we have designed a completely new type of verification interface, quickVerify, that simply presents detected bounding boxes in a grid view (Figure 3.18). Similar to the facial recognition verifier in Google Picasa, these boxes can be rapidly confirmed or rejected with a single-click and a new bounding box appears in its place. In a preliminary experiment using 160 GSV scenes and 59 distinct turkers, however, we found that accuracy with quickVerify dropped significantly. Unlike faces, we believe that curb ramps require some level of surrounding context to accurately perceive their existence. More work is needed to determine the appropriate amount of surrounding view context to balance speed and accuracy.

3.14.2 Improving human labeling.

Human labeling time could be reduced if point-and-click interactions were used for labeling targets rather than outlining; however, as demonstrated in Figure 3.16, curb ramps vary dramatically in size, scale, and shape. Clicking alone would be insufficient for CV training. Moreover, labeling will always be more costly than verification because it is a more difficult task (i.e., finding elements in an image requires visual search and a higher mental load). With that said, we currently discard all svDetect bounding boxeseven those with a high confidence scorewhen a scene is routed to svLabel. Future work should explore how to, instead, best utilize this CV data to improve worker performance (e.g., by showing detected bounding

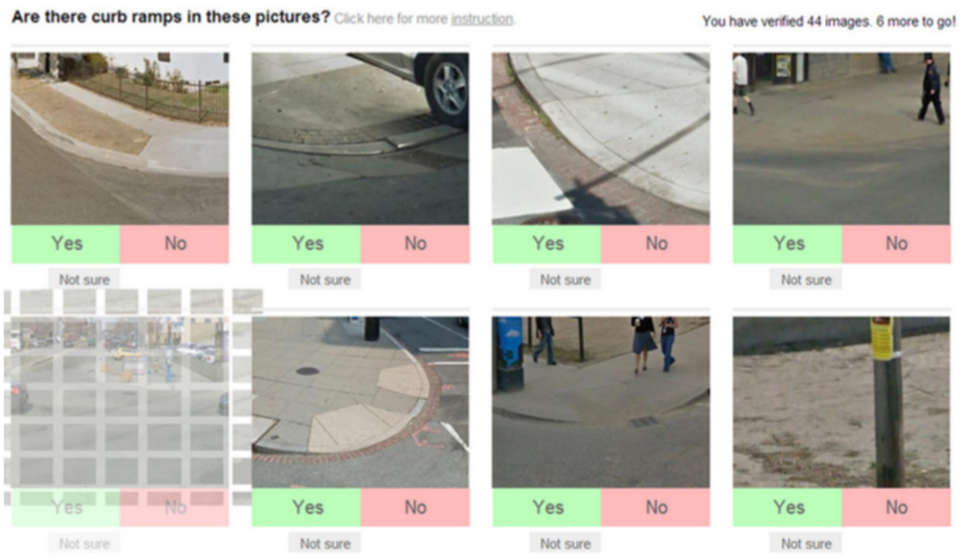

Are there curb ramps in these pictures? Click here for more instruction.
You have verified 44 images. 6 more to go!
Yes
No
Not sure

Figure 3.18: In the quickVerify interface, workers could randomly verify CV curb ramp detection patches. After providing an answer for a given detection, the patch would explode (bottom left) and a new one would load in its place. Though fast, verification accuracies went down in an experiment of 160 GSV scenes and 59 turkers.

boxes with high scores to the user or as a way to help verify human labels). Finally, similar to quickVerify, future work could explore GSV panorama labeling that is not projected onto a 3D-sphere but is instead flattened into a 2D zoomable interface (e.g., [84]) or specially rendered to increase focus on intersection corners.

3.14.3 Improving Automated Approaches

As the first work in automatically detecting curb ramps using CV, there are no prior systems with which to directly compare our performance. Having said that, there is much room for improvement and advances in CV will only increase the overall efficacy of our system.

Improving CV-based curb ramp detection. We are currently exploring three areas of future work: (i) Context integration. While we use some context information in Tohme (e.g., 3D-depth data, intersection complexity inference), we are exploring methods to include broader contextual cues about buildings, traffic signal poles, crosswalks, and pedestrians as well as the precise location of corners from top-down map imagery. (ii) 3D-data integration.Due to low-resolution and noise, we currently use 3D-point cloud data as a ground plane mask rather than as a feature to our CV algorithms. We plan to explore approaches that combine the 3D and 2D imagery to increase scene structure understanding (e.g., [85]). If higher resolution depth data becomes available, this may be useful to directly detect the presence of a curb or corner, which would likely improve our results. (iii) Training. Our CV algorithms are currently trained using GSV scenes from all eight city regions in our dataset. Given the variation in curb ramp appearance across geographic areas, we expect that performance could be improved if we trained and tested per city. However, in preliminary experiments, we found no difference in performance. We suspect that this is due to the decreased training set size. In the future, we would like to perform training experiments to study the effects of per-city training and to identify minimal training set size. Relatedly, we plan to explore active learning approaches where crowd labels train the system over time.

Improving the workflow controller. While our current workflow controller focuses on predicting CV performance, future systems should explore modeling and predicting human worker performance and adapting work assignments accordingly. For example, struggling workers could be fed scenes that are predicted to be easy, or hard scenes can be assigned to more than one worker to take majority vote [69, 72]. Similar to CV detection, per-city training and active learning should also be explored.

Who pays? The question of who will pay for data collection(or if payment is even necessary) in the future is an important, unresolved one. Our immediate plans are to build an open website where anyone can contribute voluntarily. From conversations with motor impaired (MI)persons and the accessibility community as a whole (e.g.,non-profit organizations, families of those with MI), we believe there is a strong demand for this system. For example, with a public version of Tohme, a concerned, motivated father could easily label over 100 intersections in his neighborhood in a few hours. A website akin to walkscore.com could then visualize the accessibility of that neighborhood using heatmaps and also calculate accessible pedestrian routes.

3.14.4 Limitations

There are two primary limitations to our work. First, there is a workload imbalance between svLabel and svDetect. svLabel gathers explicit data on both curb ramps and missing curb ramps while svDetect only detects the former. It is likely that if the svLabel task involved only labeling curb ramps, the labeling task completion time would go down, which would affect our primary results. And, while the lack of a detected curb ramp could be equated to a missing curb ramp label for svDetect, we have not yet performed this analysis. Clearly, more explorations are needed here but we believe our initial examinations are sufficient to show the potential of Tohme. Second, there is no assessment of how our curb ramp detection results compare to traditional auditing approaches (e.g., performed by city governments). Anecdotally, we have found many errors in the DC government curb ramp dataset [44]; however, more research is necessary to uncover whether our approach is faster, cheaper, and/or more accurate. Ultimately, Tohme must produce sufficiently good data to enable new types of accessibility-aware GIS applications (e.g., pedestrian directions routed through an accessible sidewalk path).

3.15 Summary

This chapter contributes the design and evaluation of a new tool, Tohme, for semi-automatically detecting curb ramps in GSV images using crowdsourcing, computer vision, and machine learning. To our knowledge, we are the first work to design and investigate CV algorithms for curb ramp detection, an important sidewalk accessibility attribute. We are also the first to combine crowdsourcing with automated methods for collecting accessibility information about the physical world in GSV scenes. Tohmes custom workflow controller predicts CV performance and routes work accordingly to balance accuracy and human labor. Through an MTurk study of 1,086 intersections across four North American cities, we showed that Tohme could provide comparable curb ramp detection accuracy at a 13% reduction in cost. As computer vision and machine learning algorithms continue to improve, Tohme should only become more efficient.

While this work focuses specifically on curb ramps, we believe a similar approach could be applied to analyze the accessibility of external building facades (e.g., the presence of stairways), the safety of intersections (e.g., the presence of painted cross walks), or even the accessibility of store aisles as mapping companies increasingly focus on the indoors (e.g., [20]).

One limitation of Tohme is that it is not designed to detect missing curb ramps. For that purpose, we need a standalone context model, which we introduce in the next chapter.

Chapter 4: Seeing What Is Not There: Learning Context to Determine Where Objects Are Missing

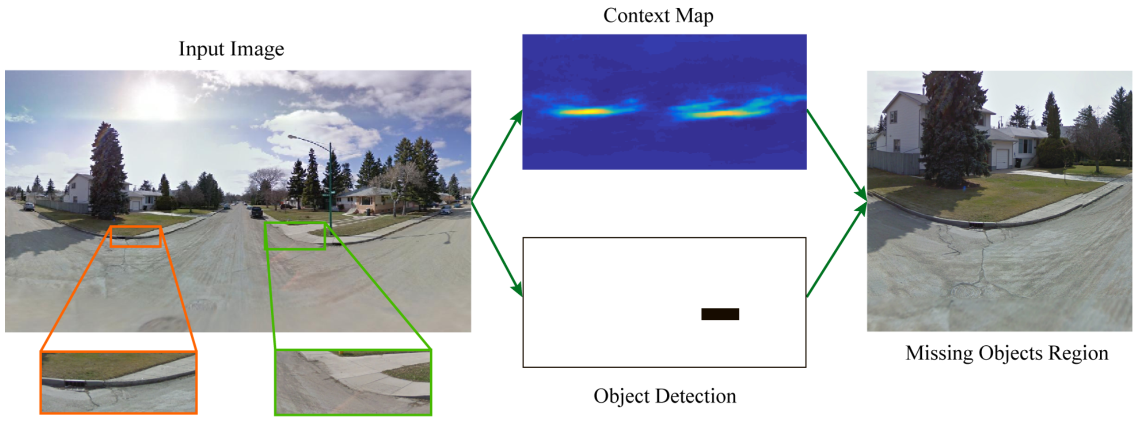


Figure 4.1: When curb ramps (green rectangle) are missing from a segment of sidewalks in an intersection (orange rectangle), people with mobility impairments are unable to cross the street. We propose an approach to determine where objects are missing by learning a context model so that it can be combined with object detection results.

4.1 Introduction

There are several limitations in Tohme (Chapter 3). 1) It uses a collection of simple cues for context. While our experiments show that such context information indeed can help improving detections, we want to model context more effectively and flexible. 2) Although Tohme has good performance in detecting constructed curb ramps, it is not designed to detect missing curb ramps (Figure 4.1). In this work, we investigate how to encode context using a deep convolutional neural network model and solve the missing curb ramp detection problem.

Knowing missing curb ramp information is highly valuable: people with disabilities can assess the accessibility of an area; navigation algorithms can calculate better routes for pedestrians; governments can plan for future renovations accordingly. This is an expensive and time consuming task for human labelers, which is partially the reason why such information is missing from public databases. Therefore, we are interested in developing an automatic algorithm that is effective and efficient. It can be used to scan a whole city to find regions where curb ramps are missing. In this scenario, the number of found true missing curb ramp regions (recall) is more important than precision because it is much more light-weight to ask humans to verify algorithm results than to label images from scratch. Moreover, even if the algorithm reports one true missing curb ramp region but mistakenly ignores three others in an image, it is still valuable as a preprocessing step. With the missing curb ramp regions data, government can prioritize intersections in a city to send physical auditors in a more efficient way.

Most fundamental computer vision tasks, e.g., image classification and object detection, focus on seeing what is there: for example, is there a curb ramp in this image, if yes, where is it? Using deep neural network models, computational approaches to such tasks are catching up to human performance in more and more benchmarks. However, humans can easily outperform algorithms in the task of inferring objects that are ‘not there’: for example, is there a curb ramp in this image, if no, where *could* it be?

We are interested in finding where objects are *missing* in an image: an object of interest is not there, even though the environment suggests it should be. From a computational perspective, an object can be defined as missing in an image region when: 1) an object detector finds nothing; 2) a predictor of the object’s typical environment, i.e. context, indicates high probability of its existence. Given an image, we want to detect all such regions efficiently. We summarize the relationship between an object’s detector and its context model in Table 4.1. While there are many existing works on utilizing context in object detection (Section 4.2), they mainly focus on improving performance on finding typical objects with contextual and object information entangled. In this work we propose to train a standalone object-centric context representation to find missing objects. By looking at the reverse conditions, it can be adapted to find out of context objects too.

One practical motivation for finding missing objects comes from the street view curb ramp detection problem (Figure 4.1). The task is to label curb ramps in a city’s intersections so that people with mobility impairments can plan their routes with confidence. Although existing work [50] shows good performance in detecting constructed curb ramps, it cannot detect missing curb ramps regions. Knowing this information is highly valuable: users can assess the accessibility of an area; navigation algorithms can calculate better routes for pedestrians; governments can plan for future renovations accordingly. This is a very expensive and time consuming task for human labelers, which is partially the reason why such information is missing from public databases. Therefore, we are interested in developing an automatic algorithm that is effective and efficient. It can be used to scan a whole city to find regions where curb ramps are missing. In this scenario, the number of found true missing curb ramp regions (recall) is more important than precision because it is much more light-weight to ask humans to verify algorithm results than to label images from scratch. Moreover, even if the algorithm reports one true missing curb ramp region but mistakenly ignores three others in an image, it is still valuable as a preprocessing step: governments can prioritize intersection assessments in a city and allocate auditors more efficiently.

We believe the key to tackle this problem is to learn a model that focuses on context only and works efficiently just like an object detector: it scans each image and generates a probability heat map in which each pixel represents the probability that an object exists, even when no object is in sight. One big advantage of the context and object decomposition is that we don't need abnormal object labels (missing/out-of-context) for training. A standalone context model can be learned from typical objects and later used for finding abnormal objects. This greatly simplifies training: normal objects are abundant and much easier to collect and label than abnormal objects.

In this study, we propose such a model based on convolutional neural networks and a novel training strategy to learn a standalone context representation of a target object. We start by introducing a base network in Section 4.3. It takes input images with explicit object masks and learns useful context from the remaining areas of the

Object Score	Context Score	Image Region Remark
High	High	Typical objects
Low	High	Missing objects
High	Low	Out of context objects

Table 4.1: Relationship between object and context. Object score is obtained from an object detector, while context score is from its context model.

images. Because of the limitations discussed in Section 4.4, we then propose a fully convolutional version of the network that learns an implicit object mask such that it ignores objects in an image and focuses purely on context. It does not require object masks during test time. Section 4.5 describes the procedure for using the context model to find missing objects regions and Section 4.6 presents experimental results.

The contributions of this work are as follows. First, we propose a method to learn an object-centric context representation by learning from object instances with masks. Second, we propose a training strategy to force the network to ignore objects and learn an implicit mask. The model is fully convolutional so it also speeds up probability heat map generation significantly. Finally we present promising results on the missing curb ramps detection problem in street view images, and a preliminary result on finding out-of-context faces.

This work was done with Prof. David Jacobs and published in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017.

4.2 Related Work

Context in Object Recognition. A large body of evidence has shown that contextual information affects human visual search and recognition of objects [87, 88]. In computer vision, recently it also has become a well accepted idea that context helps in object recognition algorithms [89–92]. Usually, context is represented by semantic labels around an object. [93] uses a Conditional Random Field to model contextual relations between objects' semantic labels to post-process object recognition results. [90] builds a deformable part model that incorporates context labels around an object as 'parts'. Because of the coupling between context and object information, these methods are unsuitable to detect missing object regions.

Torralba et al. proposed the Context Challenge [94] that consists in detecting an object using exclusively contextual information. They take the approach of learning the relation between global scene statistical features and object scale and position. Visual Memex [95] is a model that can either retrieve exemplar object instances or predict the semantic identity of a hidden region in an image. It uses hand-crafted features and models context as inter-category relations. Our approach can be seen as a general approach that attempts to address this challenge, without the need for designing hand-crafted features or using preset object classes.

Finding Missing Objects. Grabner et al. proposed to use the General Hough Transform to find objects that are missing in video frames during object tracking [96]. The idea is to estimate positions of a target object from surrounding objects with coupled motions.

Computer Vision with Masked Images. Recently Pathak et al. [97] proposed to learn a convolutional neural network context encoder for image inpainting. Both their work and ours train convolutional neural networks with masked images. But the purpose is very different as they try to learn a generative model to inpaint the mask while we learn a discriminative model to infer what is inside the mask. Also, our work uses an efficient fully convolutional structure.

Accessibility Tasks. With massive online resources such as the Google Street View (GSV) service, many computer algorithms are designed to help people with disabilities and improve their quality of life. CrossingGuard [37] is a system designed to help visually impaired pedestrians to navigate across intersections with help from Amazon Mechanical Turk. Tohme [50] is a semi-automated system that combines crowdsourcing and computer vision to collect existing curb ramp positions in city intersections using GSV images. It uses the Deformable Part Models [64] as a curb ramp detector and asks Mechanical Turkers to verify the results. They provide a street view curb ramp dataset with 1086 city intersection images, which we use in the experiment section.

4.3 Learning Context from Explicit Object Masks

In this section, we introduce a base version of the proposed context learning algorithm. If 'context' is defined to be everything that surrounds an object except the object itself, this model is learning context literally: every target object instance in training images is masked out. Here we assume an object's visual extent is fully

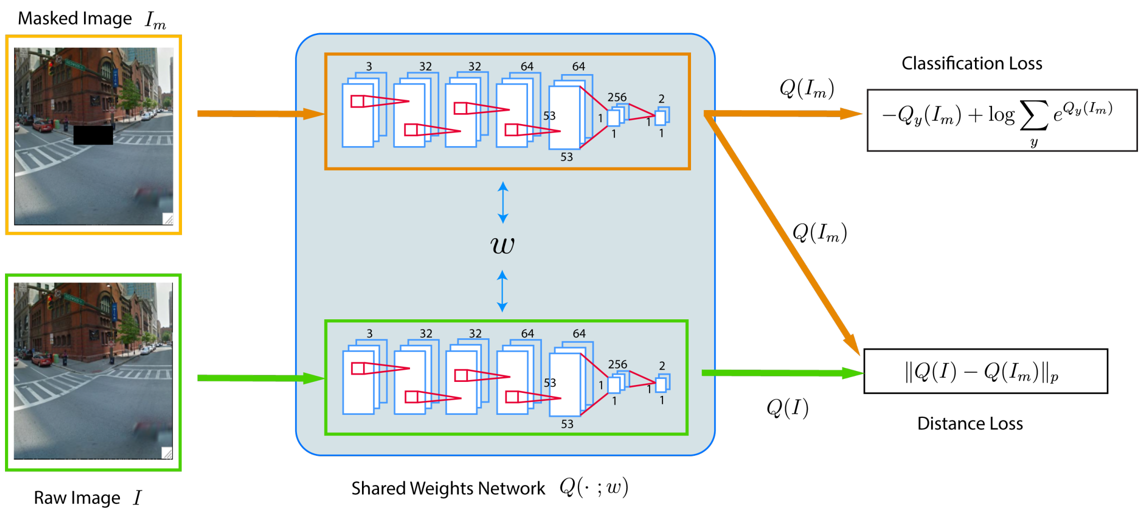


Figure 4.2: Training scheme of the **S**iamese trained **F**ully convolutional **C**ontext network (SFC). The intuition is to enforce the fully convolutional network Q to output similar results regardless of whether an image is masked or not. Additionally, the network should produce correct classification labels. The training is done in a Siamese network setting with shared weights w.

represented by its bounding box label.

This is a binary classification problem. Positive samples are collected so that each image sample has an object at its center, with a black mask (value equals zero after preprocessing) covering the object's full extent. The bounding box width to the whole image width ratio is set to 1/4 for the purpose of including a larger contextual area. Negative samples are random crops with similar black masks at their centers. The position of a negative crop is chosen so that the masked region will not cover any groundtruth labeled objects with more than a Jaccard index [1] of 0.2.

[1]Defined as the intersection-over-union ratio of two rectangles.

If there are multiple object instances in an image, we mask out one object at a time for positive samples. This is because the existence of other object instances could be useful context: for example, curb ramps often appear in pairs.

To prevent our context model trivially learning the particular mask shape, we force negative samples to share a similar distribution of mask dimensions with positive samples. The sampling strategy is to interleave the positive sampling and negative sampling processes, and use the previous positive sample's mask dimension in the next negative sample.

We train a convolutional neural network model Q. The network consists of four convolutional layers with pooling and dropout, and two fully connected layers. Its structure is summarized in Table 4.2. Cross entropy loss (Eq. 4.1) is used as the classification loss:

$$\mathcal{L}_c = -Q_y(I_m) + \log \sum_y e^{Q_y(I_m)}, \tag{4.1}$$

where $y \in \{1, 2\}$ is the groundtruth label for a masked image I_m (1 for positive, 2 for negative), $Q(I_m)$ is a 2x1 vector representing the output from the network Q, while $Q_y(I_m)$ represents its y-th element.

During test time, a sliding window approach is used to generate a probability heat map for a new image so that each pixel has a context score of how likely it is to contain an object. At each position, a fixed size (224x224 in our implementation) image patch is cropped with the center region masked out to be fed into the base network. The mask size is determined empirically from the training set.

4.4 A Fully Convolutional Model that Learns Implicit Masks

There are several issues with a network trained with masked images. First, the network tends to learn artifacts. [97] reports that training with rectangular mask makes a network learn “low level image features that latch onto the boundary of the mask”. They propose to use random mask shapes to prevent this issue. However, we cannot use random masks because our mask is defined over the visual extent of an object. Second, during testing time, the base network expects every input to have an explicit mask. This is highly inefficient when we evaluate the network at all positions and scales to generate a heat map. There are standard procedures to convert a convolutional neural network with fully connected layers into a fully convolutional one [98] so that the map generation is much more efficient for images of arbitrary sizes. However, in our case the situation is complicated. During training, the base network always sees input images with all zeros at the center, so the weights of neurons with receptive fields on this region can be arbitrary because no gradients are updated. If we apply the converted fully convolutional network to unmasked images, outputs from those neurons can affect the final map arbitrarily.

The question is then, can we train a network so that it is fully convolutional and learns context by ignoring the masked region ‘by heart’?

The answer is yes and we now propose a training strategy to make a network learn an implicit object mask. The intuition is that we want the network to output similar results regardless of whether an input image is masked or not. By enforcing this objective, the network should learn to find visual features that are shared in both masked and raw images: i.e. from the unmasked regions.

Formally, we want to minimize a distance loss in addition to the classification loss used in the base network:

$$\mathcal{L}_d = ||Q(I_m) - Q(I)||_p, \tag{4.2}$$

where $Q(I_m)$ is the output vector from the network Q with masked image I_m as the input, $Q(I)$ is the output vector from Q with the unmasked raw image I as the input, and $\|\cdot\|_p$ represents the L_p-norm.

Effectively, we have two shared-weight networks that are fed with masked and raw image pairs (Figure 4.2). The network is a fully convolutional version of the base network (Table 4.3). One stream of the network computation takes a masked image as input and outputs $Q(I_m)$. In parallel, the other stream of network computation takes the unmasked raw image as input and outputs $Q(I)$. The classification loss $\mathcal{L}_c$ is calculated based on $Q(I_m)$ alone, while the distance loss $\mathcal{L}_d$ is calculated by $Q(I_m)$ and $Q(I)$. This structure is known as a Siamese Network [99] so we call it the Siamese trained Fully convolutional Context (SFC) network. Following [99], we choose the L_1 norm in distance loss $\mathcal{L}_d$. We expect the SFC network to learn an implicit object mask by assigning zero weights to neurons whose receptive field falls onto the center object mask region. During test time, unlike the base network, we don't have to manually set the mask size: the SFC network has encoded this information in convolutional filters' weights.

Finally, the overall training objective is defined as a weighted sum of the two losses:

$$\mathcal{L} = \lambda \mathcal{L}_d + \mathcal{L}_c, \tag{4.3}$$

where $\lambda = 0.5$ in our implementation.

The benefits of this training strategy are three fold:

1) Because the SFC learns to ignore object mask regions, we can directly apply it to new unmasked images with arbitrary sizes: it is now highly efficient to generate a dense probability map. Figure 4.3 shows a comparison between heat maps generated by the base network and the SFC network. A 1024x2048 pixels image costs about 5 minutes to generate a heat map with the base network while the SFC network takes less than 4 seconds to generate a map with higher spatial resolutions.

2) The SFC network is less prone to artifacts. It is possible for the base network to learn artifact features along the boundary of masks. Since such features are not present in unmasked images, the SFC network learns to ignore them.

3) During training, we can perform hard negative mining efficiently. Between each training epoch, we can apply the SFC network on all training images to generate heat maps and find high score false positive regions. Because of the efficiency of fully convolutional networks, this step can be easily included in training. Section 4.6.2 shows that hard negative mining indeed improves the network performance by a large margin.

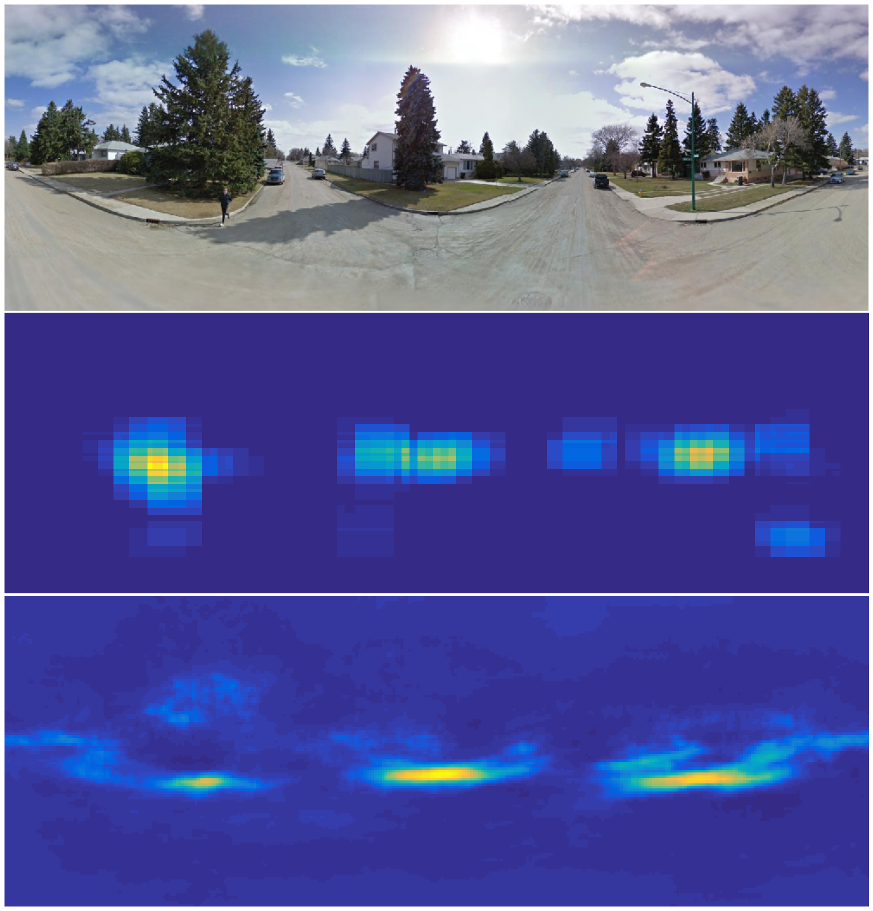


Figure 4.3: Top: an input street view panorama image. Middle: the heat map generated by the base network using a sliding window approach. Bottom: the dense heat map generated by the SFC network.

4.5 Finding Missing Object Regions Pipeline

With a trained standalone context network (base network or SFC network), we summarize the procedure for finding missing object regions in a test image.

1) Generate a context heat map using the context network Q. This map shows where an object should appear.

2) Generate object detection results using any object detector. Convert detec-

tion boxes into a binary map by assigning 0 to the detected box region, 1 otherwise. This binary map shows where no objects are found.

3) Perform element-wise multiplication between the context heatmap and the binary map. The resulting map shows the regions where an object should occur according to its context but the detector finds nothing.

4) Crop the high scored regions (above a preset threshold) from the image according to the resulting map. These are the regions where objects are missing.

4.6 Experiments

In this section, we first examine the characteristics of the base network and the SFC network in Subsection 4.6.1. Then we evaluate their effectiveness. With the decomposition of context and object information, we study two unique tasks that can be efficiently performed using a standalone context model. Subsection 4.6.2 shows experimental results of finding missing curb ramp regions in street view images. Subsection 4.6.3 shows preliminary results of detecting out of context faces.

4.6.1 Characteristics of the Trained Model

As a validation study, we first check the sensitivity of the base and the SFC networks with regard to small changes in input images. All experiments are conducted on the curb ramp street view dataset. A desirable model has small response variations to the center region of an input image, where a mask was put during training. For evaluation, we change one pixel value at a time in a test image, by adding a small noise. The L_2 distance between a network's output before and after the disturbance is recorded for each pixel. In the end we obtain a map that shows which region in the image has large impact on the network's output. This can be seen as an estimate of the first order derivative of a network with respect to its input. Figure 4.4 shows the result with comparison between the base network and the SFC network. This result is summed over 20 different image samples.

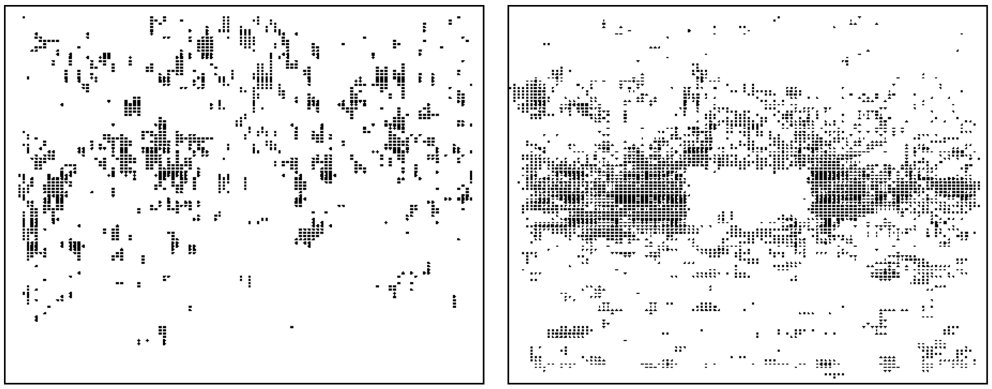


Figure 4.4: The sensitivity map of the base network (left) and the SFC network (right): a dark dot indicates a high sensitivity spot. Compared to the base network, the SFC map has a clear blank area at the center, which indicates that changes in this region have little effect on the network's output. The SFC network learns an implicit region mask.

From the result it is clear that the SFC network has small sensitivity at the center region of the input image. This is most likely due to the network learning to mute neurons whose receptive field falls at the center region of the input image. The blank region in the SFC's sensitivity map can be seen as a visualization of an approximation to the learned implicit region mask.

Next we check the distance loss $\mathcal{L}_d$ of the base network and the SFC network on test data. Using the same set of training hyper-parameters and setup (learning rate, training epochs) to train the two networks, the mean $\mathcal{L}_d$ loss is summarized in Table 4.4. It is clear that the SFC network is much more consistent in producing similar outputs regardless of object masks.

The SFC network works as intended: 1) it learns an implicit mask so it is less sensitive to any changes in the center region; 2) the useful features that it learns for the classification task are mainly from unmasked regions.

4.6.2 Finding Missing Curb Ramp Regions

Setup. We want to find missing curb ramps in the street view curb ramps dataset [50]. The dataset contains 1086 Google Street View panoramas which come from four cities in North America: Washington DC, Baltimore, Los Angeles and Saskatoon (Canada). Each panorama image has 1024x2048 pixels. It provides bounding box labels for existing curb ramps. On average there are four curb ramps per image. In addition, for our evaluation, an expert has labeled all missing curb ramp regions.

The dataset is split into half training and half testing. Each image is converted to YUV color space and normalized to be zero mean and one standard deviation in all channels. We use the curb ramp detector provided with the dataset, a Deformable Part Model, with default settings.

Training. For each epoch, 5000 samples are generated from training data, with half positives and half negatives. Figure 4.5 shows several examples. Each sample has 50% probability of being horizontally flipped for data augmentation purposes. Positive samples contain valid context around curb ramps. Negative samples are cropped randomly from areas not containing a curb ramp. To train the SFC network, each sample is prepared with two versions: raw and masked. We resize positive samples such that the object width is close to 55 pixels in a 224 pixels wide image. Each negative sample uses the same object mask and scale as the last positive sample to prevent the network overfitting to mask shapes.

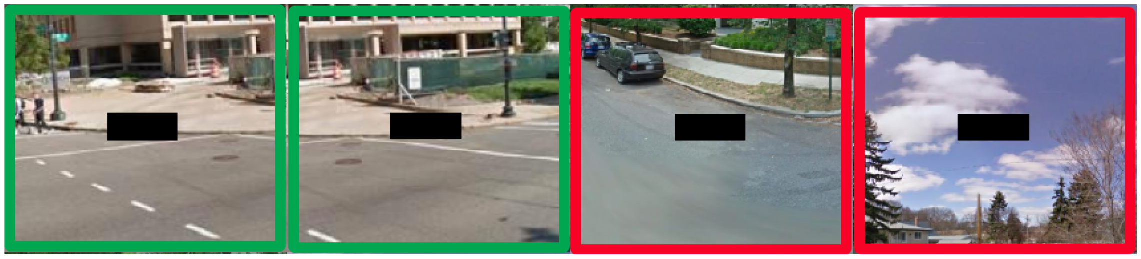


Figure 4.5: Training examples of curb ramps. Green rectangles represent positive samples, red rectangles represent negative samples.

We use the Keras/Tensorflow neural network software package [100]. The optimization algorithm uses Adadelta with default parameters. Since this is an adaptive learning rate method, there is no need to set a learning rate schedule during training. 20% of the training data is used as a validation set for an early stopping test. A base network and a SFC network are trained using the same hyper-parameters and training setup.

Results. Following the procedure described in Section 4.5, we run the two networks on test images to generate probability heat maps of where curb ramps should be in an image. For the base network, each heat map is generated in a sliding window scheme with a stride of 10 pixels, and various object mask widths of {50, 70, 100} pixels to generate multi-scale maps. The SFC network doesn't need an object mask size, so we resize the input panorama image with scales {0.5, 0.7, 1.0}. The numbers are chosen so that two networks see similar image pyramids. We use the DPM detector provided with the dataset to generate detection results. For each panorama, we generate a final map that combines the detection and context map and crop high scored regions (above a certain threshold) with size $d \times d$. According to preliminary empirical studies, we set the context threshold to 0.4 throughout the experiment.

We use human verification to evaluate the quality of the reported missing curb ramp regions. For that purpose, we develop a web based interface (Figure 4.6) that displays a gallery of found regions, ranked by their context scores. For each candidate region, a user provides feedback on whether it is truly a missing curb ramp region. We compare context maps generated by the base and the SFC networks with three baseline methods: random scores, spatial prior map, and a Faster RCNN [101] missing curb ramp detector.

Random scores assigns uniformly random context scores from $[0, 1]$ to all positions in an image. This is a reference baseline showing the performance by chance.

A spatial prior map is built using the prior positions of curb ramps in street view panoramas. We use the prior map as a replacement for the context map for comparison. We collect the prior spatial distribution of all curb ramps from the training images. The collected distribution is smoothed with a 30x30 pixel Gaussian kernel with sigma=10. Figure 4.7 shows the spatial prior map used in our experiment. Because most panoramas are at street intersections, there is strong

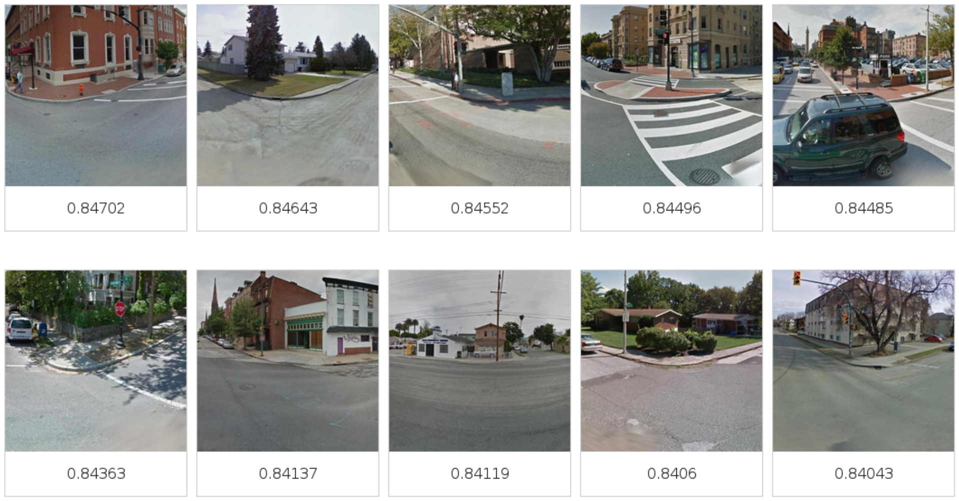


Figure 4.6: The web interface for verification. Each thumbnail shows one retrieved region, with its score displayed below. A user clicks on a thumbnail to verify it.

spatial structure consistency among the dataset. We expect this approach to be a reasonable baseline.

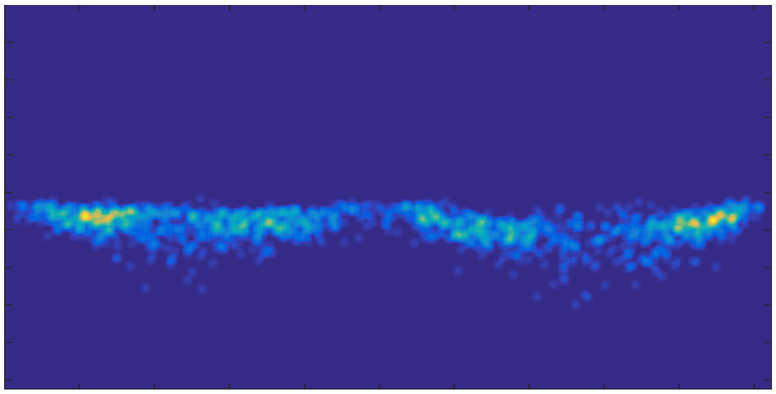


Figure 4.7: The spatial prior heat map generated from groundtruth locations of curb ramps in the training set. It shows that curb ramps are far from uniformly distributed.

With missing curb ramp region labels, we can treat this task as a standard object detection problem and directly train a Faster RCNN detector: the positive 'object' is a region labeled as missing curb ramps. Note that a Faster RCNN detector is capable of learning context because it's an end-to-end approach: potentially the detector can learn from the whole image to predict locations of missing curb ramp regions. We expect the Faster RCNN detector to be a strong baseline.

The verification of the missing curb ramp regions requires domain knowledge. One of the authors who has extensive experience with accessibility problems verified the results using our web interface. Figure 4.8 shows the comparison in recall of true missing curb ramp regions versus the number of visited regions (Recall@K). The retrieved region size is set to $d = 400$ pixels. 500 regions were retrieved from 543 test images.

The result shows that the SFC network with hard negative mining outperforms all other methods. We believe its superiority comes from the highly efficient fully convolutional structure that helps in training and generating high resolution context maps. Spatial prior map shows reasonable performance, which confirms the spatial bias of curb ramps locations in the dataset. Unlike the spatial prior map, the proposed methods can work well on other datasets that have no such bias. The Faster RCNN detector has significantly less recall compared with the SFC networks. With more missing curb ramp regions as training data, we expect the Faster RCNN detector to show improved performance; on the other hand, the SFC network does not even need missing curb ramp labels in training. The proposed methods learn useful context information from normal curb ramps, which are much easier to collect and label than missing curb ramp regions. Moreover, the SFC network is using detection results from a less advanced curb ramp detector (a DPM model shipped with the dataset): 77% of the false missing curb ramp retrievals are due to inaccurate curb ramp detections. Due to the page limit, we show more qualitative results of retrieved regions in the supplementary document.

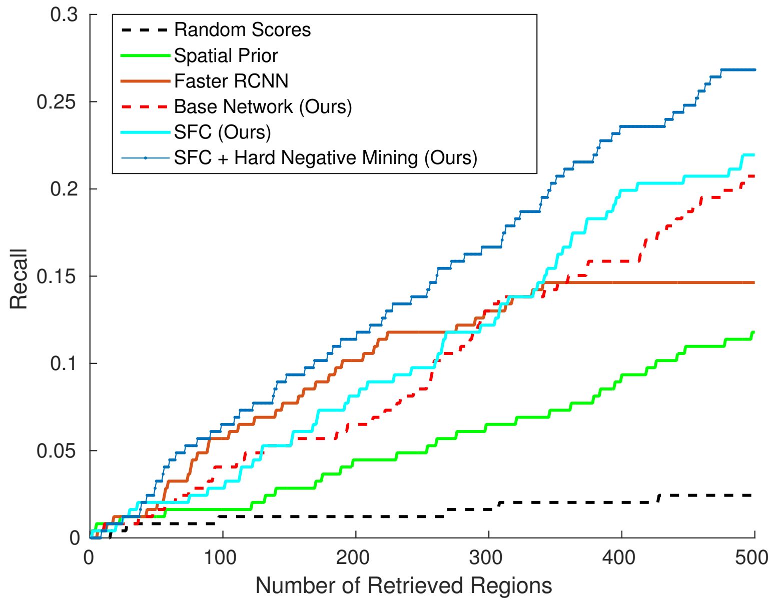


Figure 4.8: Recall of true missing curb ramp regions vs number of regions viewed (Recall@K). Our base and SFC networks outperform the two baseline methods (random scores and prior maps) by a large margin. The difference in recall between the Faster RCNN detector and the proposed method is substantial. The SFC network with hard negative mining has the best result among the proposed methods.

Additionally, we investigate the effects of the retrieved region size d on the number of true missing curb ramp regions. Specifically, we vary the cropped region size from 400 pixels in width to 100 pixels. With smaller region size, it becomes crucial that the region is accurately localized with missing curb ramps at the center. Table 4.5 shows that the SFC network is not affected too much by the reduced field of view. This is because the regions it found are very well localized (See

Figure 4.6). On the other hand, two baseline methods (random scores and prior maps) are performing poorly when the region size becomes small.

Discussion. Among 543 street view intersections in the test set, the SFC network is able to find 27% of the missing curb ramp regions by merely looking at 500 regions. This is an impressive result: 1) The whole process is very efficient (Table 4.6) such that it can be easily deployed to scan new city areas. For example, there are about 2,820 intersections in Manhattan, New York: it will take merely a few hours for our system to find missing curb ramps in a region with 1.6 million population; 2) Accessibility reports have shown that curb ramps condition (missing or not) shows high proximity consistency: if one intersection is missing curb ramps, it is highly likely that the nearby intersection has similar issues [86]. Our results can be used as an initial probe to quickly locate city areas that need special attention.

4.6.3 Finding Out of Context Faces

The pipeline in Section 4.5 for finding missing objects can be adapted to find out of context objects with just a few small modifications: change step 2 by assigning 1 to detected box regions and 0 for other regions; change step 4 to retrieve the lowest scored regions. Here we show a preliminary result of finding out of context faces to demonstrate both the generalization ability of the proposed method in different domains and possible future directions.

The task is to find out of context faces in the Wider face dataset [102]. Using a similar procedure as in finding missing objects and a state-of-the-art face detector [103], we retrieve the top 500 face regions that contain high face detector scores and low context scores from the validation set. For evaluation, we define an out of context face as a face without a visible body. Figure 4.9 shows qualitative results of the SFC network. We compare the SFC network results with random scoring. Out of 500 regions, the SFC network can find 27 out of context faces while random scoring found 14. While this result is preliminary, it suggests that the proposed method has the potential to be used in many other applications where finding out of context objects is important: for example, visual anomaly detection.

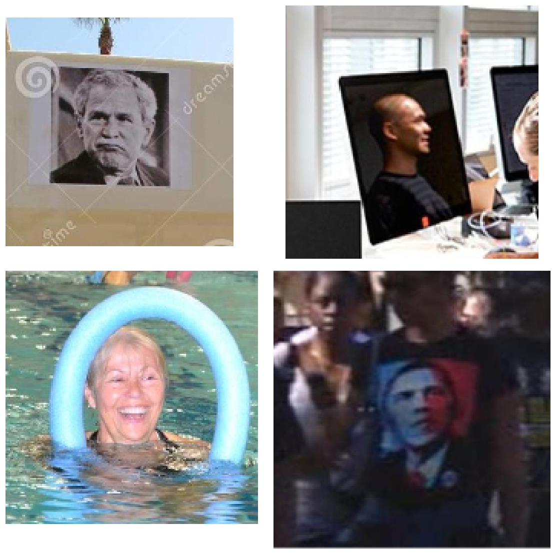


Figure 4.9: Retrieved out of context faces by a SFC network.

4.7 Summary

We present an approach to learn a standalone context representation to find missing objects in an image. Our model is based on a convolutional neural network structure and we propose ways to learn implicit masks so that the network ignores objects and focuses on context only. Experiments show that the proposed approach works effectively and efficiently on finding missing curb ramp regions.

Layer (type)	Shape	Param #
Convolution2D	(3, 3, 32)	896
Convolution2D	(3, 3, 32)	9248
MaxPooling2D	(2, 2)	0
Dropout	-	0
Convolution2D	(3, 3, 64)	18496
Convolution2D	(3, 3, 64)	36928
MaxPooling2D	(2, 2)	0
Dropout	-	0
FullyConnected	(53*53*64, 256)	46022912
Dropout	-	0
FullyConnected	(256, 2)	514
Total params: 46,088,994		

Table 4.2: Neural network structure summary for the base network. Convolution filter shapes are represented by (filter width, filter height, number of filters) tuples. The network expects to take an input image of size 224x224, with an explicit mask at the center.

Convolution2D	(53, 53, 256)	46022912
Dropout	-	0
Convolution2D	(1, 1, 2)	514

Table 4.3: Fully convolutional layers to substitute for the last three layers of the base network. This network can take arbitrary sized input, with no explicit mask needed.

	SFC network	Base network
$\mathcal{L}_d$ loss	0.041	2.27

Table 4.4: Mean $\mathcal{L}_d$ loss of the two networks on the curb ramp dataset test set. Lower loss means smaller changes between a network's outputs from masked and unmasked images.

Region Width	400	200	100
SFC	35	33	27
Spatial Prior	13	8	4
Random Scores	4	2	0

Table 4.5: Effect of retrieved region size on the raw number of found missing curb ramps with 255 regions (the higher the better). As the region width shrinks, SFC performs very consistently while the two baseline methods (random scores and prior maps) suffer from poor localizations.

	Context Map (*)	Detection	Verification
Cost	4s/image	22s/image	20min/500 ims

Table 4.6: Time costs for different steps in finding missing curb ramps. The whole process is efficient as context and detection maps can be generated in parallel. *Using the SFC network.

Chapter 5: Generating Holistic 3D Scene Abstractions for Text-based Image Retrieval

5.1 Introduction

In this chapter, we move beyond single object detections and investigate the problem of representing objects in certain geometrical relationships in a scene. In particular, we are interested in image retrieval tasks where a complex scene is represented as a composition of multiple objects.

Text-based image retrieval, dating back to the late 1970s, has evolved from a keyword-based task to a more challenging task based on natural language descriptions (e.g., sentences and paragraphs) [114, 115, 126]. Queries in the form of sentences rather than keywords refer to not only object categorical information but also interactions, such as spatial relationships, between objects. Those relationships are usually described in the real (3D) world due to the nature of human language. Intuitively, they can be the core feature for ranking images in many application scenarios, e.g., a user searching for images that are relevant to a particular mental image of a room layout. Not surprisingly, researchers have recently increased their focus on understanding spatial relationships from text input and retrieving semantically consistent visual information [114, 120, 127, 136].

Matching images with user provided spatial relations is challenging because humans naturally describe scenes in 3D while images are 2D projections of the world. Inferring 3D information from a single image is difficult. Most existing approaches learn from annotated data to map language directly to a probability distribution of pairwise relationships between object locations [114, 120]. However, such a distribution is non-convex and highly non-linear in the 2D image space because the (unknown) camera view affects the bounding box configurations. Consequently, the success of 2D learning based approaches naturally depends on the size of annotated training data. Also, the learner overfits easily since annotated spatial relations have a long-tailed distribution; many valid configurations happen rarely in the real world (e.g., a desk on another desk). With pairwise relations, it is also hard to enforce the fact that all objects are viewed from the same direction in an image. This argues for a holistic model for object relationships that jointly optimizes object configurations. Motivated by this, we explore an alternative model of spatial relations that generates 3D configurations explicitly based on physics.

We explore an approach that uses physical models and complex spatial relation semantics as part of an image retrieval system that generates 3D object layouts from text (rather than from images) and performs image retrieval by matching 2D projections of these layouts against objects detected in each database image. Our framework requires the a priori definition of a fixed set of object and spatial relation categories. Spatial relation terms are extracted from the dependency tree of the text. Objects are modeled using cuboids and spatial relations are modeled as inequality constraints on object locations and orientations. These inequality constraints can become very complex, containing nonlinear transformations represented using first order logic. Consequently, an interval arithmetic based 3D scene solver is introduced to search for feasible 3D spatial layout solutions. Camera orientations are constrained and sampled for obtaining 2D projections of candidate scenes. Finally, images are scored and ranked by comparing object detection outputs to a sampled set of 2D reference layouts.

Compared to 2D learning based approaches, our approach has the following advantages: (1) the mapping from language to 3D is simple since the text-based spatial constraints have a very concrete and simple meaning in 3D, simple enough to define with a few rules by hand; (2) no training data is needed to learn complex distributions over the spatial arrangement of 2D boxes given linguistic constraints (the non-linear mapping from language to 2D is handled by projective geometry) and (3) adding common sense constraints is easy when referring to physical relationships in 3D (Sec. 5.4.2.2), while it is hard if these constraints are specified and learned in 2D (due to the non-linearity of projective geometry). We evaluate our approach using two public scene understanding datasets [107, 131]. The results suggest that our approach outperforms baselines built upon object occurrence histograms and learned 2D relations.

This work was done with Ang Li, Joe Yue-Hei Ng, Ruichi Yu, Vlad I. Morariu, and Prof. Larry Davis. It was published in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017.

5.2 Related Work

Text-based image retrieval has been studied for decades [126]. As both computer vision and natural language processing have advanced, recent efforts have emerged that build connections between linguistic and visual information [116, 123]. Srivastava and Salakhutdinov [132] extend Deep Boltzman Machines (DBMs) to multimodal data for learning joint representations of images and text. They apply such representations to retrieving images from text descriptions. Their model learns mappings between objects with attributes and their corresponding visual appearances; however spatial relations are not modeled.

Spatial relationships play an important role in visual understanding. Previous works make use of text-extracted spatial relations in image retrieval. Zitnick *et al.* [136] generate and retrieve abstract cartoon images from text. Cartoon object models are pre-defined and 2D clipart images are composed according to the text. Siddiquie *et al.* [128] devise a multi-modal framework for retrieving images from sources including images, sketches and text by jointly considering objects, attributes and spatial relationships, and reducing all sources into 2D sketches. However, their framework handles text with only two or three objects and very limited 2D spatial relationships. Lin *et al.* [120] retrieve videos from textual queries. A set of motion text is defined with visual trajectory properties and parsed into a semantic graph to to match video segments via a generalized bipartite graph matching. All these works rely on 2D spatial relations while our work is based on real world physical models of 3D scenes to retrieve semantically consistent images.

Interesting recent work on retrieving images from text is based on the scene graph representation [114,127]. A scene graph is a graph-based representation which encodes objects, attributes and object relations. In Johnson *et al.* [114], text input is converted to a scene graph by a human and a CRF model is used to match scene graphs to images by encoding global spatial relations of objects rather than only pairwise relations. Their approach requires learning spatial relations from annotated image data. Our work differs in that we take a generative perspective and inject physical relation models and human knowledge into the retrieval system without the requirement of large-scale data annotation.

Many existing works utilize 3D geometry in vision tasks such as object recognition [112], image matching [119], object detection [134, 135], *etc.* However, to the best of our knowledge, the use of 3D geometry in relating images with language has not been exploited. While inferring the 3D structure from a single image is challenging and complicated in vision [107, 109, 113, 124, 125], the problem of rendering scenes from text is of interest in the graphics community. The wordseye system [108] renders scenes from text with given 3D object models. Chang *et al.* [106] generates 3D scenes from text by incorporating the spatial knowledge learned from data. In addition, some recent works cast computer vision as inverse graphics and try to incorporate computer graphics elements into visual understanding systems [117,118,133]. Our work also involves scene generation. However, our purpose is to retrieve similar images based on bounding boxes , which can be efficiently computed using off-the-shelf software during a database indexing step, so real object models are not required, although better scene generation could potentially improve image retrieval

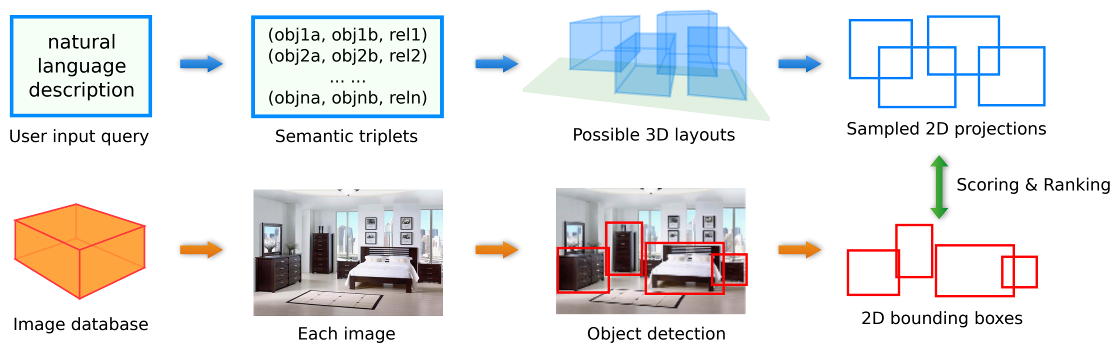


Figure 5.1: Framework overview: a textual description of the visual scene is parsed into semantic triplets which are used for solving feasible 3D layouts and their 2D projections as *reference configurations*. An object detector runs over each database image and generates a 2D bounding box layout, to be matched to reference configurations. All database images are ranked according to their configuration scores.

accuracy.

5.3 Preliminary: Interval Analysis

Our approach involves finding feasible solutions to a mathematical program where the variables are object coordinates and orientations, and the constraints are inequalities translated from user descriptions. Since small placement perturbations usually do not affect the fulfillment of constraints, feasible variables can naturally be represented by a set of intervals (any value within the interval is feasible).

Interval analysis represents each variable by its feasible interval, e.g., $[l, u]$ (with lower bound l and upper bound u) and the goal is to find the bound for each dimension that satisfies all constraints [130]. When an interval does not satisfy all the constraints, it is split into smaller intervals and evaluated recursively. Arithmetic operators are defined in terms of intervals, e.g.,

- *addition*: $[l_1, u_1] + [l_2, u_2] = [l_1 + l_2, u_1 + u_2]$;
- *subtraction*: $[l_1, u_1] - [l_2, u_2] = [l_1 - u_2, u_1 - l_2]$;
- *comparison*: $[l_1, u_1] < [l_2, u_2]$ equals $[0, 0]$ if $u_2 \leq l_1$ (definitely false); equals $[1, 1]$ if $u_1 < l_2$ (definitely true); equals $[0, 1]$ otherwise (maybe true).

The fulfillment of a constraint can be represented by any of the three logical intervals, *i.e.*, $[0, 0], [1, 1], [0, 1]$.

5.4 Our Approach

The proposed framework, as illustrated in Fig. 5.1, consists of several modules. First, the input text is parsed into a set of semantic triplets of object names and their spatial relationships. Second, the semantic triplets are used to solve possible 3D layouts of objects along with sampled camera locations and orientations. The 2D projections of the 3D scenes are used for generating 2D bounding boxes of objects, which we call *reference configurations*. Finally, the reference configurations are matched to the detected bounding boxes in each database image to score and rank according to their configuration similarity.

5.4.1 Text Parsing

The text parsing module translates text into a set of semantic triplets which encode the information about two object instances and their spatial interactions. How to robustly extract relations from text is still an open research problem in natural language processing [115], which is beyond the scope of this chapter. For our application, a simple rule-based pattern matching works sufficiently well, requiring a pre-defined dictionary of object and spatial relation categories. A text example and its parsing output is shown in Table 5.1.

The input text is processed by the Stanford CoreNLP library [122] with part-of-speech tagging and dependency tree. We implement a rule-based approach to extract spatial relations (such as *on*, *under*, *in front of*, *behind*, *above*, etc.) from the dependency tree and compose its corresponding semantic triplet representation *(target object, reference object, relation)*. The co-reference module in the CoreNLP library is used to aggregate multiple noun occurrences that correspond to the same object instance. Each object reference is represented by its category name and a unique ID within the category, e.g. `sofa-0` and `dining-table-2`.

Natural objects are usually composed of multiple sub-objects and there are often cases when a sub-object is referenced instead of the whole object. A bed, for instance, has its head and rear. And a chair has its back and seat. We take sub-objects into consideration and represent any sub-object reference by its object category name, unique in-category ID and sub-object name, e.g. "the rear of the bed" is represented as `bed-0:rear` if the ID is 0.

Besides object categories and spatial relationships, we also consider the count of each object, e.g. three chairs, two monitors, etc. The parser maintains a list of object ID and their counts. If the count of `chair-0` is 3, then the parser will expand `chair-0` to a set of three instances {`chair-0-0,chair-0-1,chair-0-2`} in the outputs.

#	Sentence → (object-1, object-2, relation)
1	A picture is above a bed.
	`(picture-0, bed-0, above)`
2	A night stand is on the right side of the head of the bed.
	`(night-stand-0, bed-0:head, right)`
3	A lamp is on the night stand.
	`(lamp-0, night-stand-0, on)`
4	Another picture is above the lamp.
	`(picture-1, lamp-0, above)`
5	A dresser is on the left side of the head of the bed.
	`(dresser-0, bed-0:head, left)`

Table 5.1: Semantic triplet parsing from an example query

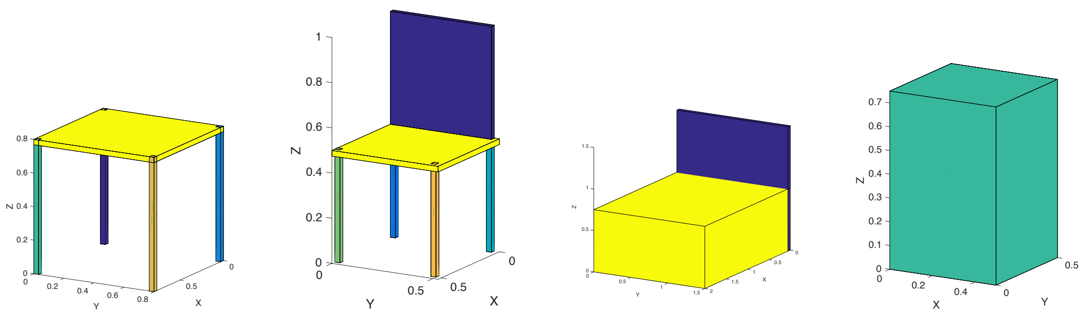
5.4.2 3D Abstract Scene Generation

The 3D abstract scene generation module is the central component in our image retrieval framework; it takes as input semantic triplets and generates a set of sampled possible 3D object layouts. We describe below the three core components of the scene generator: the cuboid based object model, the spatial relation model and the 3D scene solver.

5.4.2.1 Cuboid based object model

The basic *cuboid representation* of an object is $\mathbf{C} = (l_x, l_y, l_z, z_s)$ where (l_x, l_y, l_z) is the size of the cuboid that bounds the object in x, y, z directions respectively and z_s is the z-coordinate of the supporting surface of the object. We mostly use regular sizes but also set different sizes for objects with attributes such as `long-desk`, `triple-sofa`, etc. The supporting surface is usually the top face of the object cuboid, but it can sometimes be located elsewhere with respect to the cuboid, e.g., for a chair it is in the middle of the cuboid. Spatial relations such as `on` and `under` are modeled with respect to the surface of the object. Most of the objects can be modeled using this cuboid representation such as `garbage-bin`, `picture`, `night-stand`, etc.

However, the single cuboid representation is not sufficient for some object categories such as `chair` and `desk` since the under-surface area is empty. Considering the fact that most objects can be easily decomposed into smaller sub-objects, we represent these object categories as the union of a set of cuboids, which we call a *cuboid set representation*. Each sub-cuboid corresponds to a sub-object and is considered a simple object, whose top face is the supporting surface. The k-th sub-cuboid is represented by $\mathbf{S}^k = (d_x^k, d_y^k, d_z^k, l_x^k, l_y^k, l_z^k)$ where (d_x^k, d_y^k, d_z^k) is the offset from the lowest point of the sub-cuboid to the lowest point of the original object, and (l_x^k, l_y^k, l_z^k) is the size of the sub-cuboid. The sub-cuboid parameters $\mathbf{S}^k$ are computed as functions of the original object parameters $\mathbf{C}$. Four sampled cuboid based object models are visualized in Fig. 5.2.



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

Figure 5.2: Sample cuboid based object representations: (a) table (b) chair (c) bed (d) night-stand. Different colors represent different sub-objects. The night stand (d) is represented by a single cuboid.

5.4.2.2 Spatial relation model

The spatial location and orientation of each object is represented as $\mathbf{X} = (x, y, z, d)$ where (x, y, z) is the lowest point of the object cuboid and d is its orientation. The object rotation is around the z-axis.

Atomic relations. We model 8 basic spatial relations using the following mathematical expressions. Given the object pose and its size, the lowest point $\mathbf{p} = (x_p, y_p, z_p)^\intercal$ and highest point $\mathbf{q} = (x_q, y_q, z_q)^\intercal$ of the object cuboid can be computed by rotating the object models w.r.t. the object orientation such that

$$\begin{aligned} \mathbf{p} &= \mathbf{R}_d \left[-\frac{l_x}{2}, -\frac{l_y}{2}, -\frac{l_z}{2}\right]^\intercal + \left[x + \frac{l_x}{2}, y + \frac{l_y}{2}, z + \frac{l_z}{2}\right]^\intercal, \\ \mathbf{q} &= \mathbf{R}_d \left[\frac{l_x}{2}, \frac{l_y}{2}, \frac{l_z}{2}\right]^\intercal + \left[x + \frac{l_x}{2}, y + \frac{l_y}{2}, z + \frac{l_z}{2}\right]^\intercal \end{aligned} \tag{5.1}$$

where $\mathbf{R}_d$ is the z-axis rotation matrix w.r.t. to orientation d. So an object can be represented using tuple $(\mathbf{p}, \mathbf{q}, d)$. Letting the cuboid of object-1 be $\mathbf{O}_1(\mathbf{p}_1, \mathbf{q}_1, d_1)$

with support surface z_{s1} and the cuboid of object-2 be $\mathbf{O}_2(\mathbf{p}_2, \mathbf{q}_2, d_2)$ with support surface z_{s2}, we define 8 atomic relations as

- `near`: $\mathbf{O}_1 \cap (\mathbf{p}_2 - d_{\text{near}}\mathbf{e}_{d_2}, \mathbf{q}_2 + d_{\text{near}}\mathbf{e}_{d_2}, d_2) \neq \emptyset$;
- `on`: $z_{p1} = z_{s2} \wedge \frac{\mathbf{p}_1+\mathbf{q}_1}{2} \in_{xy} \mathbf{O}_2$;
- `above`: $z_{q2} + d_{\text{min-above}} \leq z_{p1} \leq z_{q2} + d_{\text{max-above}} \wedge \frac{\mathbf{p}_1+\mathbf{q}_1}{2} \in_{xy} \mathbf{O}_2$;
- `under`: $z_{s1} < z_{s2} \wedge \mathbf{O}_1 \cap_{xy} \mathbf{O}_2 \neq \emptyset$;
- `behind`: $\max(\mathbf{u}_{d_2}^\intercal \mathbf{p}_1, \mathbf{u}_{d_2}^\intercal \mathbf{q}_1) \leq \min(\mathbf{u}_{d_2}^\intercal \mathbf{p}_2, \mathbf{u}_{d_2}^\intercal \mathbf{q}_2)$;
- `front`: $\min(\mathbf{u}_{d_2}^\intercal \mathbf{p}_1, \mathbf{u}_{d_2}^\intercal \mathbf{q}_1) \geq \max(\mathbf{u}_{d_2}^\intercal \mathbf{p}_2, \mathbf{u}_{d_2}^\intercal \mathbf{q}_2)$;
- `on-left`: $\min(\mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{p}_1, \mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{q}_1)$
  $\geq \max(\mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{p}_2, \mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{q}_2)$;
- `on-right`: $\max(\mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{p}_1, \mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{q}_1)$
  $\leq \min(\mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{p}_2, \mathbf{u}_{d_2-\pi/2}^\intercal \mathbf{q}_2)$;

where $d_{\text{near}}, d_{\text{min-above}}, d_{\text{max-above}}$ are distance thresholds, $\mathbf{p} \in_{xy} \mathbf{C}$ means point $\mathbf{p}$ is inside the cuboid $\mathbf{C}$ on the x-y plane, $\cap$ represents the intersection of two cuboids and $\cap_{xy}$ the intersection of two cuboids on the x-y plane, and $\mathbf{u}_\theta = (\cos\theta, \sin\theta, 0)^\intercal$ is a unit direction vector and $\mathbf{e}_\theta = (\cos\theta - \sin\theta, \sin\theta + \cos\theta, 1)^\intercal$ is a vector that enlarges the effective object cuboid.

Composite relations. In natural language, there are far more spatial relation descriptions than the above mentioned 8 relations. However, most of the spatial relations can be defined based on the 8 atomic relations. Two examples are

- `next-to`: `on-left`$(\mathbf{O}_1, \mathbf{O}_2) \vee$ `on-right`$(\mathbf{O}_1, \mathbf{O}_2)$;

- `side-by-side`: $d_1 = d_2 \wedge \texttt{near}(\mathbf{O}_1, \mathbf{O}_2)$;

In addition, another relation is modeled which is usually used for a set of multiple instances $\{\mathbf{O}_1, \mathbf{O}_2, \dots, \mathbf{O}_k\}$ of the same object category, i.e.,

- `in-a-row`: $d_i = d_{i+1} \wedge \texttt{on-right}(\mathbf{O}_i, \mathbf{O}_{i+1}),\ \forall i$;

Group relations. If an object reference has a count more than 1, then all of its instances form a group, which often interacts with other objects as an entirety. If a group of k instances occurs in the triplet as the target, we create k new triplets with the same reference and relation. If the group occurs as the reference, then we create a new virtual object whose cuboid is bounded by all of its instances.

Prior constraints. An effective way to reduce the search space is to incorporate common sense and reasonable assumptions into the constraints. First, we make the following assumptions: (a) the room has two walls ($x = 0$ and $y = 0$); (b) the text description is coherent, i.e., the objects in each semantic triplet are close to each other; (c) objects are usually oriented along x-axis or y-axis directions. Second, no pair of objects overlap with each other, i.e.,

- `exclusive`: $\mathbf{S}_i^v \cap \mathbf{S}_j^w = \emptyset \forall i, j, v, w$

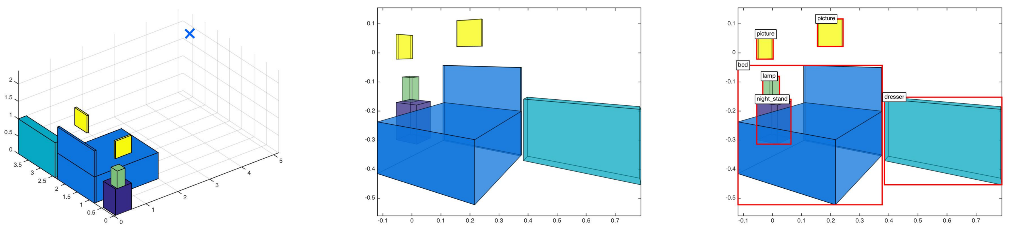
where $\mathbf{S}_i^v$ is the v-th component (sub-cuboid) of the i-th object. Many other constraints are related with object properties: (a) picture, door, mirror are on the wall, i.e. $x = 0 \vee y = 0$; (b) for relation next-to, in-a-row, side-by-side, if either reference or target is against the wall, the other ones are also against the wall and they should also have the same orientation; (c) bed, night-stand, sink are against the wall; (d) bed, night-stand, sofa are on the ground.

5.4.2.3 3D scene solver

Let $\mathbf{X} = \{x_1, y_1, z_1, d_1, \ldots, x_n, y_n, z_n, d_n\} \in \mathbb{R}^{4n}$ be a *layout state* representing the locations and orientations of all objects. We construct *constraint function* $F : \mathbb{R}^{4n} \rightarrow \{0, 1\}$ which evaluates all prior constraints and relational constraints. The goal is to find the feasible solution set $\mathbf{S}$ such that $F(\mathbf{X}) = 1$ for all $\mathbf{X} \in \mathbf{S}$.

Our solver is based on interval analysis [130] where any variable is represented by an interval (an uncertain value) instead of a certain value. We use a vector of size 2 to represent an interval, i.e., a lower bound and an upper bound. Under interval analysis, the domain of layout states becomes $\mathbb{R}^{4n\times 2}$ and the constraint function becomes $F : \mathbb{R}^{4n\times 2} \rightarrow \{[0, 0], [0, 1], [1, 1]\}$. Starting with a candidate queue containing an initial interval layout state $\{\mathbf{X}_0\}$, our solver examines the candidate states one at a time. For each state $\mathbf{X}_i \in \mathbb{R}^{4n\times 2}$, if $F(\mathbf{X}_i) = [1, 1]$, then $\mathbf{X}_i$ is feasible and appended to the solution set. If the constraint fullfillment is undecidable, i.e., $F(\mathbf{X}_i) = [0, 1]$, then $\mathbf{X}_i$ is divided into two equally sized intervals by splitting the variable with the largest uncertainty. The two new states are appended to the candidate queue. Otherwise, $F(\mathbf{X}_i) = [0, 0]$ and no feasible solution is within the space bounded by $\mathbf{X}_i$. In the end, any layout in the solution set is guaranteed to meet all constraints. An advantage of the method is that it does not require computing the gradient of constraint F. The pseudo-code is shown in Algorithm 1.

Interval shrinkage. The original interval analysis does not make full use of equality constraints, e.g., when a variable is constrained to equal another variable, it becomes redundant to divide both of their intervals since one can be directly



(a) (b) (c)

Figure 5.3: The generated scene geometry for the query in Table 5.1: (a) a sampled 3D layout with the sampled camera location (a blue cross in the figure), (b) 2D projections of the object cuboids and (c) 2D bounding boxes of the objects.

computed based on the other. In addition, many spatial relations are transitive, e.g., if object A is in front of object B and B is in front of C, then A is likely to be in front of C but with a larger distance. Such inferred constraint can benefit the solver with a better pruning power. Based on these observations, we develop the interval shrinkage operation which pre-computes lower bound matrices $\mathbf{L}^x, \mathbf{L}^y, \mathbf{L}^z \in \mathbb{R}^{n\times n}$ and upper bound matrices $\mathbf{U}^x, \mathbf{U}^y, \mathbf{U}^z \in \mathbb{R}^{n\times n}$ for pairwise coordinate differences, i.e., $L^x_{i,j} \le x_i - x_j \le U^x_{i,j} \wedge L^y_{i,j} \le y_i - y_j \le U^y_{i,j} \wedge L^z_{i,j} \le z_i - z_j \le U^z_{i,j}$. The bound matrices are initialized using the original constraints and updated once we find $L^*_{i,j} < L^*_{i,k} + L^*_{k,j}$ or $U^*_{i,j} < U^*_{i,k} + U^*_{k,j}$ $(* \in \{x, y, z\})$. Before evaluating each candidate interval layout state, we shrink its variables according to the bound matrices, e.g., $\mathbf{x}^{\text{shrink}}_i = \cap_j [x_j + L^x_{i,j}, x_j + U^x_{i,j}] \cap \mathbf{x}_i$ where $\mathbf{x}_i$ is the interval of variable x_i and $\mathbf{x}^{\text{shrink}}_i$ is the interval after shrinkage.

Early stopping. The feasible solution space can be large if the input constraints are weak. Since we sample K layouts in our framework for subsequent

Algorithm 1: 3D scene solver

```
Data: Initial bounds $\mathbf{X}_0 = [\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_1, \mathbf{d}_1, \dots, \mathbf{x}_n, \mathbf{y}_n, \mathbf{z}_n, \mathbf{d}_n] \in \mathbb{R}^{4n\times 2}$
Data: Constraint $F : \mathbb{R}^{4n\times 2} \to \{[0,0],[0,1],[1,1]\}$
Result: Feasible regions (or solution set) $\mathbf{S}$
1  initialization: $\mathbf{S} = \emptyset, \mathbf{Q} = \{\mathbf{X}_0\}$;
2  while $\mathbf{Q} \neq \emptyset$ do
3      read the first interval: $\mathbf{X}_i = \mathbf{Q}$.front();
4      remove the first interval: $\mathbf{Q}$.pop();
5      interval shrinkage: $\mathbf{X}_i$ = shrinkage($\mathbf{X}_i$);
6      if $F(\mathbf{X}_i)$=$[0,0]$ then
7          $\mathbf{X}_i$ is not feasible;
8      else if $F(\mathbf{X}_i)$=$[1,1]$ then
9          $\mathbf{X}_i$ is feasible: $\mathbf{S}$.append($\mathbf{X}_i$);
10     else if $\max_k |X_{ik}.\max - X_{ik}.\min| > tol$ then
11         $k = \arg\max_k |X_{ik}.\max - X_{ik}.\min|$;
12         half split $k$-th dimension of $\mathbf{X}_i$ into $\mathbf{X}_i^{(1)}$ and $\mathbf{X}_i^{(2)}$;
13         $\mathbf{Q}$.append($\mathbf{X}_i^{(1)}$);
14         $\mathbf{Q}$.append($\mathbf{X}_i^{(2)}$);
15 end
16 return $\mathbf{S}$;
```

image matching, the 3D scene solver stops when at least K layouts are found. The sampling behavior is achieved by implementing the candidate queue with Knuth shuffling, i.e., each time after appending a new element, the queue randomly pick an element and swaps it with the new element.

The problem is a combinatorial optimization which is NP-hard and interval analysis is essentially a breadth first search with pruning. As a result, the algorithm has no time limit guarantee. However, with interval shrinkage and early stopping, our algorithm is able to solve most queries in a reasonable amount of time. Without interval shrinkage, our MATLAB implementation can not find a solution for the query in Table 5.1 within 10 minutes, while it returns 5 solutions with only 6 seconds using the shrinkage operation.

5.4.3 Image Retrieval

To compare a query with image bounding boxes, we first sample feasible 3D layouts and potential camera locations and orientations to produce reasonable 2D projections of objects and then compute their bounding boxes. The whole image database is scored and ranked according to the similarity between bounding boxes detected by object detectors and those from sampled 2D layouts.

3D layout sampling. The 3D solver finds (continuous) interval solutions for 3D object coordinates; any solution within such intervals is feasible. However, the solutions within an interval are redundant; those object locations shift in tiny distances. So we sample only one layout within each interval, which results in a set of representative feasible 3D layouts. We further sample a few 3D layouts from this feasible set in order to generate their 2D projections.

2D layout projections. For each layout, we sample camera locations and orientations to obtain 2D projections which allows matching images under multiple views. Object bounding boxes are computed according to the 2D projections. Since we solve for scale and translation for each image individually during matching, in this step we only consider a canonical camera. Some heuristics are used for sampling camera locations and orientations. First, the camera always faces the objects and should be neither too close nor too far, so we sample its location from 5-10 meters from the origin. Second, the camera should not be located behind the wall, so the coordinates are positive. Third, when an object is on the wall, the camera direction should be within 60 degree offset from the object orientation. We assume the camera is 1.7 meters above the ground and situated horizontally. Fig. 5.3 shows an example of 3D layout, 2D projections and 2D bounding boxes for the query in Table 5.1.

2D layout similarity. Both detection outputs and 2D reference layouts can be represented by $\{\mathbf{b}_i, c_i\}$ where $\mathbf{b}_i$ is the 2D box of the i-th object and c_i is its category. Let $\{\mathbf{b}_i, c_i\}$ be a 2D reference layout and $\{\mathbf{b}'_i, c'_i\}$ be the detected boxes. Since scaling and translation are left as free variables, the bounding box matching involves optimizing

$$\max_{s,t,\mathbf{a}} \sum_i p(\mathbf{b}'_{a_i}) \cdot \textsc{iou}(s\mathbf{b}_i + t, \mathbf{b}'_{a_i}), \quad s.t.\ c_i = c'_{a_i}, \tag{5.2}$$

where $p(\mathbf{b}'_k)$ is the detection confidence, IOU is intersection-over-union and assignment vector $\mathbf{a}$ indicates the correspondence between two sets of bounding boxes. In our experiment, we evaluate two versions: (a) the *hard* version uses a threshold on detection outputs and uniform $p(\mathbf{b}'_k)$ and (b) the *soft* version makes $p(\mathbf{b}'_k)$ equal to the detection score. We use a sliding window to find the best matched transformation and assignment. Specifically, we uniformly sample 5 scale factors from 0.5 to 1 w.r.t. the image space and search with a 10-pixel stride. We use a greedy strategy to compute assignments and scores (Eq. 5.2). The score for a query is computed as the highest score among the scores of all its sampled 2D layouts.

5.5 Experiments

We validate our approach using two indoor scene datasets (SUN RGB-D [131] and 3DGP [107]). Although the original goal of the two datasets is not text-based image retrieval, both contain groundtruth object bounding boxes which enables evaluation in our image retrieval setting. We compare 3 baselines built upon object occurrence histogram and 2D spatial relation based scene graph matching.

5.5.1 Setup

Baseline (H). The first baseline is based on the histogram of object occurrences. Specifically, both the image and text are converted to a histogram representation, i.e., a vector $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, where x_i is the number of occurrences of the i-th object category. The similarity between occurrence histograms is measured by ℓ^1 distance.

Baseline (2D). The second baseline is based on learned object relations in 2D image space. Specifically, the baseline learns a bounding box distribution of the first object w.r.t. the second object box (normalized in both x and y coordinates). We have all eight atomic relations annotated in 1,000 images in the training set of SUN

RGB-D dataset and use IOU-based nearest neighbor (IOU-NN) classifier to score for each test image the spatial relationships between object pairs. Following [105], we convert the text to a simplified scene graph that maps all instances of an object category into a single node, and assign the count of each relation as an attribute of the corresponding edge. An image scene graph with relation probabilities on edges can be constructed for each test image by using the IOU-NN relation classifier upon each pair of detected object instances. To measure the similarity between text scene graph and image scene graph, we sum for each edge (u, v, r) in the text scene graph the top $k_{u,v,r}$ corresponding relation scores in the image scene graph, where $k_{u,v,r}$ is the count of the relation r between object categories u and v in text scene graph.

Baseline (CNN). The third baseline replaces the IOU-NN relation classifier in Baseline 2D with a Convolutional Neural Network (CNN). Following [121], we finetune the pretrained VGG-19 [129] to predict predicates from cropped union image regions of the two objects. The word2vec vectors of the two objects are concatenated with the response of layer *fc7*. We backpropagate through the whole network with initial learning rate 0.001 for 90 epochs.

Evaluation metric. We evaluate different approaches to retrieving indoor images from text descriptions by measuring the percentage of queries (recall) at least one of whose ground truth images are retrieved within top k ranked images ($R@k$). The median rank (median of the ranks of all ground truths) is used as a global measurement.

Parameter selection. We set the room size to be $5m \times 5m \times 5m$. $d_{\text{near}} = 0.5m, d_{\text{min-above}} = 0.25m, d_{\text{max-above}} = 0.5m$. The tolerance in 3D scene solver is

$0.2m$ because $20cm$ replacement of objects is unlikely to change the constraint fulfillment. We sample 5 reference layouts per query and 1 camera view per layout unless otherwise specified.

5.5.2 SUN RGB-D dataset with R-CNN detectors

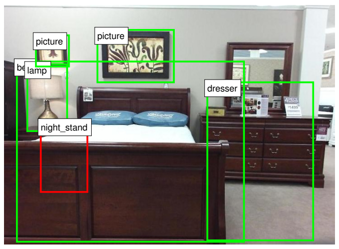
SUN RGB-D Dataset [131] is a recent dataset for scene understanding which contains $10,335$ RGBD images. We use only the RGB images without depth information. We follow the same protocol as [131] by using $5,285$ images for training the detectors and the remaining $5,050$ images as the evaluation set. We annotated text queries for 150 sampled test images. SUN RGB-D contains various objects and complex spatial relations. We choose 19 object categories in our evaluation: {*bed, chair, cabinet, sofa, table, door, picture, desk, dresser, pillow, mirror, tv, box, whiteboard, night_stand, sink, lamp, garbage_bin, monitor*}, which contains not only objects on the floor but also those off the ground or on the wall such as *picture* and *mirror*.

We use the $5,285$ training images and their ground truth object bounding boxes to train Fast R-CNN [111] detectors for the 19 object categories. The R-CNN approach is built upon object proposals; non-maximum suppression is not used in postprocessing. For each test image, R-CNN detectors generate probability-like scores for all object categories on each object proposal bounding box. The category with the highest score is chosen as the bounding box category and its score is used as the bounding box confidence.

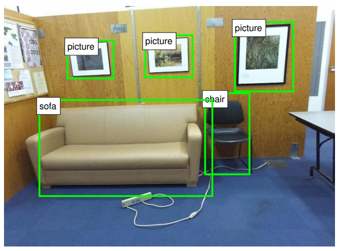
	R@1	**R@10**	**R@50**	**R@100**	**R@500**
Baseline H	1.3	4.0	14.0	20.0	43.3
Baseline 2D	2.7	15.3	35.3	44.0	64.0
Baseline CNN	2.7	16.7	30.7	36.0	63.3
Ours hard[5,1]	3.9	16.4	31.7	42.3	71.7
Ours soft[5,1]	4.5	16.7	34.0	46.4	76.0
Ours soft[5,5]	<u>4.9</u>	<u>18.7</u>	<u>37.9</u>	<u>48.1</u>	<u>76.9</u>
Ours soft[5,5] + 2D	**8.7**	**21.6**	**40.5**	**50.7**	**77.6**

Table 5.2: SUN RGB-D: Top-k retrieval accuracy for 150 queries. The retrieval candidate set contains 5,050 images. We evaluate the occurrence baseline (H), 2D relation baseline (2D), CNN baseline, the proposed hard version, proposed soft versions, and a combination between our soft version and the 2D baseline. The parameter of our model $[x, y]$ means sampling x 3D layouts and y camera views for each layout. All results of our model are averaged over 5 random trials. The threshold for detection outputs is 0.5. The best is shown in **bold** and the second best is shown with <u>underline</u>.

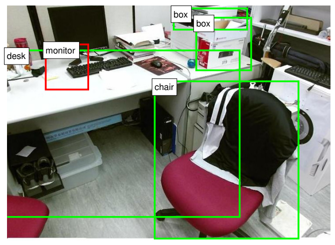
The top-k retrieval recalls are shown in Table 5.2. In addition with the baselines, two versions of our approach are evaluated. The baselines and our hard model use bounding boxes with over 0.5 confidence and weigh them equally, while our soft models use all bounding boxes and assign their confidences as weights in Eq. 5.2. The results suggest that the hard model with 5 layout samples outperforms the occurrence baseline and is on par with the 2D baseline. Our soft models perform even



(a) A picture is above a bed. A night stand is on the right side of the head of the bed. A lamp is on the night stand. Another picture is above the lamp. A dresser is on the left side of the head of the bed.



(b) There is a triple sofa. The sofa is against the wall. A chair is next to the sofa. And the chair is also against the wall. Two pictures are above the sofa. And another picture is above the chair.

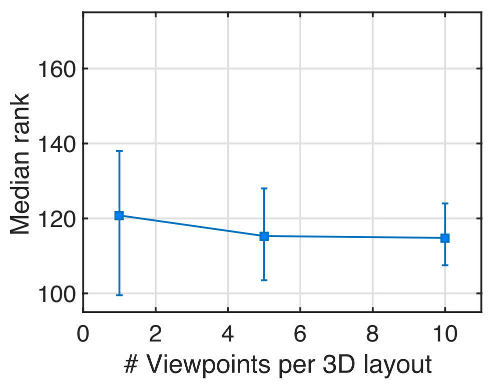


(c) A chair is in front of the desk. Some boxes are on the desk. A monitor is on the desk. The desk is against the wall.

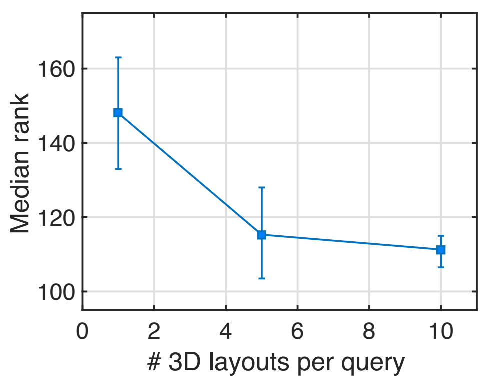
Figure 5.4: Matched object layouts based on our greedy 2D layout matching for three ground truth images that are ranked top 5 among all candidate images. Green bounding boxes are object detection outputs that match the 2D layouts generated from the text queries. Red bounding boxes represent a missing object (not detected by the object detector) within the expected region proposed by 2D layouts.

better than the hard one. With increased layout samples, our approach outperforms the baselines significantly. We also evaluate a combination between our soft model and the 2D baseline by adding their normalized scores. The result suggests that such combination further boost the accuracy and that our physical model based solution is complementary to learning based approaches.

Fig. 5.4 shows 3 examples whose ground truths are ranked top 5. The object bounding boxes that best match the generated 2D layouts are shown on the images. Green boxes are matched objects and red boxes are missing ones, expected in the



(a)



(b)

Figure 5.5: Influence of # viewpoint samples and # layout samples: (a) 5 3D layouts sampled for each query, and (b) 5 viewpoint sampled for each 3D layout. The y-axis is median rank of ground truths. We random 5 times for each data point. Lower is better.

generated 2D layout but unseen in the object detection output. The figure shows that our model has some level of tolerance on missing detections. A more interesting finding is that our model suggests potential locations for missing objects even though they could be heavily occluded.

To obtain 2D layouts, we sample 3D layouts and camera views. Fig. 5.5 shows how the sample size of both affects the the median rank of ground truths (keeping one and varying the other). Fig. 5.5 suggests that more samples generally yield better performance and the improvement saturates as the sample size increases. The improvement brought by more 3D layouts is more significant than that brought by more camera views. In addition, the performance uncertainty due to randomness decreases as the sample size increases.

5.5.3 3DGP dataset with DPM detectors

The 3DGP dataset [107] contains $1,045$ images with three scene types: living room, bedroom and dining room. Each image is annotated with bounding boxes for 6 object categories: *sofa*, *table*, *chair*, *bed*, *side table* and *dining table*. Following the same protocol as in [107], 622 training images are used to train the furniture detectors and the remaining 423 images are used as the retrieval image database. We use pre-trained Deformable Part Models (DPM) [110] of indoor furnitures provided by the 3DGP dataset and use the thresholds in the pre-trained models to cut off false alarms. Non-maximum suppression is used to remove duplicates.

3DGP dataset is less diverse than SUN RGB-D; many images have very similar layouts. We annotated 50 unique layout descriptions which cover 222 test images. The retrieval results are shown in Table 5.3. Because our method is agnostic about object detector algorithms, we split the results into two parts to separate the impact from using a specific detection algorithm: one using ground truth bounding boxes and the other using DPM detection outputs. The results suggest that our approach outperforms baseline algorithms under both bounding box settings and the improvement is independent from detector performances.

5.6 Summary

We present a general framework for retrieving images from a natural language description of the spatial layout of an indoor scene. The core component of our framework is an algorithm that generates possible 3D object layouts from text-

	w/ DPM bbox				**w/ GT bbox**			
	H	2D	CNN	Ours	H	2D	CNN	Ours
R@1	4.0	2.0	4.0	**4.4**	<u>4.0</u>	<u>4.0</u>	<u>4.0</u>	3.0
R@10	10.0	14.0	16.0	**16.8**	16.0	18.0	14.0	<u>20.2</u>
R@50	30.0	30.0	30.0	**31.2**	34.0	38.0	32.0	<u>41.4</u>
R@100	46.0	32.0	32.0	**52.0**	64.0	66.0	66.0	<u>68.0</u>

Table 5.3: 3DGP dataset: Top-K image retrieval accuracy. Left half is based on DPM (the best is with **bold**) and right half is based on ground truth bounding boxes (the best is in <u>underline</u>). The results of our approach (soft[5,5]) are averaged over 10 random trials.

described spatial relations and matching these layout proposals to the 2D image database. We validated our approach via the image retrieval task on two public indoor scene datasets and the result shows the possibility of generating 3D layout proposals for rigid objects and the effectiveness of our approach to matching them with images.

Chapter 6: Conclusion

With the ever growing enormous amount of visual data available, new powerful dedicated hardware, and evolving learning algorithms, computer vision becomes more and more practical and useful in real world applications. Object detection is one of the fundamental problems that has great practical impact in many computer vision systems.

Current object detectors are able to work well under certain conditions. However, challenges arise when scenes becomes more complex:

- Object detectors trained on data collected from the Internet fail when there are large variations in objects' poses and lighting conditions.
- Current detectors' performance are still suboptimal compared to human performance in some scenes.
- Scenes are often cluttered. Detecting each object individually can be difficult due to occlusion and distraction of similar looking objects.
- Understanding interactions between multiple objects is crucial for object localization tasks. Yet modeling object relationships is hard.

This dissertation presents four research works that aim to tackle those challenges from different angles.

1. We design a novel user interface based on Augmented Reality headsets. It can collect training data in real world scenes with large variations in poses and lighting conditions.

2. We build a system that combines computer vision algorithms detection results and human crowdsourcing verifications in detecting curb ramps. It improves detection performance over a fully automatic system but with a reduced cost compared to a fully manual process.

3. We propose a standalone context model to capture the complex relationship between objects and their environment.

4. We explicitly model the geometric relationship between objects of indoor scenes. The model can be used to retrieve objects with similar spatial layouts.

Future Directions. There are still many open questions and unsolved issues left in modeling objects and their context. One of the core challenges is how to build effective and flexible representations of context in complex scenes. Combined with standard object detectors, a desired context representation can help reduce false positive detections in unlikely regions, recover missed detections in highly plausible regions, and identify abnormal objects in unusual context. We list in the following a few directions that are worth exploring towards this goal.

Context for General Objects. SFC shows promising results on detecting missing curb ramp regions. We are interested in applying a similar strategy to learn context models for general objects such as those in the COCO dataset. This is not a trivial extension of the current work because there are different object categories sharing the same context in an image.

Non Object-centric Context. Currently, SFC is learning an object-centric context model, i.e., an object is always at the center of the input view. There are a few limitations: the ratio between region of objects and context has to be preset; objects that are close to image boundaries will have cut-off black regions in their context. We are interested in learning a more flexible context model that is not limited to the object-centric setting.

Learning Context from Positive-only Data. In the curb ramp dataset, we have information on which images do not contain curb ramps, i.e., the negative labels. However in learning context for other object categories, it is not common to have negative labels. For example in the KITTI dataset, there are regions with no pedestrians, but they are not necessarily regions where a pedestrian *cannot* occur, i.e., the true 'negative regions'. How to learn context models from positive-only data is thus an important and interesting research problem.

Bibliography

[1] Google self-driving car project, https://www.google.com/selfdrivingcar/.

[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[3] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[4] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1521–1528. IEEE, 2011.

[5] Business wire, 2017. Worldwide Shipments of Augmented Reality and Virtual Reality Headsets Expected to Grow.

[6] Vgg image annotator, 2017. `http://www.robots.ox.ac.uk/~vgg/software/via/`.

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[8] Alexander Sorokin and David Forsyth. Utility data annotation with amazon mechanical turk. In *In IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008.

[9] V.Y. Mariano, J. Min, J.-H. Park, R. Kasturi amd D. Mihalcik, D. Doermann, and T. Drayer. Performance evaluation of object detection algorithms. pages 965–969, 2002.

[10] Miguel A. Serrano, Jesús Gracía, Miguel A. Patricio, and José M. Molina. *Interactive Video Annotation Tool*, pages 325–332. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

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

[12] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4485–4493, 2017.

[13] J. Xu, A. G. Schwing, and R. Urtasun. Tell me what you see and i will show you where it is. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3190–3197, June 2014.

[14] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3159–3167, 2016.

[15] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. *What's the Point: Semantic Segmentation with Point Supervision*, pages 549–565. Springer International Publishing, Cham, 2016.

[16] Julien Valentin, Vibhav Vineet, Ming-Ming Cheng, David Kim, Jamie Shotton, Pushmeet Kohli, Matthias Nießner, Antonio Criminisi, Shahram Izadi, and Philip Torr. Semanticpaint: Interactive 3d labeling and learning at your fingertips. *ACM Transactions on Graphics (TOG)*, 34(5):154, 2015.

[17] Duc Thanh Nguyen, Binh-Son Hua, Lap-Fai Yu, and Sai-Kit Yeung. A robust 3d-2d interactive tool for scene segmentation and annotation. *CoRR*, abs/1610.05883, 2016.

[18] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niessner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[19] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Weakly-supervised semantic segmentation using motion cues. In *ECCV*, 2016.

[20] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[21] P. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1713–1721, June 2015.

[22] M. Rajchl, M. C. H. Lee, O. Oktay, K. Kamnitsas, J. Passerat-Palmbach, W. Bai, M. Damodaram, M. A. Rutherford, J. V. Hajnal, B. Kainz, and D. Rueckert. Deepcut: Object segmentation from bounding box annotations using convolutional neural networks. *IEEE Transactions on Medical Imaging*, 36(2):674–683, Feb 2017.

[23] Ross Girshick. Fast r-cnn. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 1440–1448. IEEE, 2015.

[24] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 724–732, June 2016.

[25] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[26] Mark Everingham, S. M. Eslami, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vision*, 111(1):98–136, January 2015.

[27] Jianqi Ma, Weiyuan Shao, Hao Ye, Li Wang, Hong Wang, Yingbin Zheng, and Xiangyang Xue. Arbitrary-oriented scene text detection via rotation proposals. *CoRR*, abs/1703.01086, 2017.

[28] Yingying Jiang, Xiangyu Zhu, Xiaobing Wang, Shuli Yang, Wei Li, Hua Wang, Pei Fu, and Zhenbo Luo. R2cnn: Rotational region cnn for orientation robust scene text detection. *CoRR*, abs/1706.09579, 2017.

[29] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. Multi-oriented text detection with fully convolutional networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4159–4167, 2016.

[30] Xiaoming Deng, Ye Yuan, Yinda Zhang, Ping Tan, Liang Chang, Shuo Yang, and Hongan Wang. Joint hand detection and rotation estimation by using CNN. *CoRR*, abs/1612.02742, 2016.

[31] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[32] Brent Berlin and Paul Kay. CSLI Publications, 1969.

[33] I. Wijesooriya, D. Wijewardana, T. De Silva, and C. Gamage. Demo abstract: Enhanced real-time machine inspection with mobile augmented reality for

maintenance and repair. In *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 287–288, April 2017.

[34] Y. Oyamada. A look into medical augmented reality. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–1, Sept 2014.

[35] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[36] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, pages 1–21. 10.1007/s11263-012-0564-1.

[37] Richard Guy and Khai Truong. CrossingGuard: Exploring Information Content in Navigation Aids for Visually Impaired Pedestrians. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 405–414, New York, NY, USA, 2012. ACM.

[38] Kotaro Hara, Victoria Le, and Jon Froehlich. A Feasibility Study of Crowdsourcing and Google Street View to Determine Sidewalk Accessibility. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '12, pages 273–274, New York, NY, USA, 2012. ACM.

[39] Kotaro Hara, Vicki Le, and Jon Froehlich. Combining Crowdsourcing and Google Street View to Identify Street-level Accessibility Problems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 631–640, New York, NY, USA, 2013. ACM.

[40] Kotaro Hara, Victoria Le, Jin Sun, David Jacobs, and J Froehlich. Exploring early solutions for automatically identifying inaccessible sidewalks in the physical world using google street view. *Human Computer Interaction Consortium (2013).*

[41] Kotaro Hara, Shiri Azenkot, Megan Campbell, Cynthia L. Bennett, Vicki Le, Sean Pannella, Robert Moore, Kelly Minckler, Rochelle H. Ng, and Jon E. Froehlich. Improving Public Transit Accessibility for Blind Riders by Crowdsourcing Bus Stop Landmark Locations with Google Street View. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '13, pages 16:1–16:8, New York, NY, USA, 2013. ACM.

[42] 3rd Circuit, C. of A. *Kinney v. Yerusalim*, (No. 93-1168), 1993.

[43] 311 Online, http://311.dc.gov.

[44] Data Catalog, http://data.dc.gov/.

[45] The Impact of the Americans with Disabilities Act: Assessing the Progress Toward Achieving the Goals of the ADA, 2007.

[46] United States Department of Justice, C.R.D. Americans with Disabilities Act of 1990, 1990.

[47] Hannah M. Badland, Simon Opit, Karen Witten, Robin A. Kearns, and Suzanne Mavoa. Can virtual streetscape audits reliably replace physical streetscape audits? *Journal of Urban Health: Bulletin of the New York Academy of Medicine*, 87(6):1007–1016, December 2010.

[48] Philippa Clarke, Jennifer Ailshire, Robert Melendez, Michael Bader, and Jeff Morenoff. Using Google Earth to Conduct a Neighborhood Audit: Reliability of a Virtual Audit Instrument. *Health & place*, 16(6):1224–1229, November 2010.

[49] Andrew G. Rundle, Michael D. M. Bader, Catherine A. Richards, Kathryn M. Neckerman, and Julien O. Teitler. Using Google Street View to audit neighborhood environments. *American Journal of Preventive Medicine*, 40(1):94–100, January 2011.

[50] Kotaro Hara, Jin Sun, Robert Moore, David Jacobs, and Jon Froehlich. Tohme: Detecting Curb Ramps in Google Street View Using Crowdsourcing, Computer Vision, and Machine Learning. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 189–204, New York, NY, USA, 2014. ACM.

[51] Jeffrey P. Bigham, Richard E. Ladner, and Yevgen Borodin. The Design of Human-powered Access Technology. In *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '11, pages 3–10, New York, NY, USA, 2011. ACM.

[52] Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and Tom Yeh. VizWiz: Nearly Real-time Answers to Visual Questions. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 333–342, New York, NY, USA, 2010. ACM.

[53] Walter Lasecki, Christopher Miller, Adam Sadilek, Andrew Abumoussa, Donato Borrello, Raja Kushalnagar, and Jeffrey Bigham. Real-time Captioning by Groups of Non-experts. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 23–34, New York, NY, USA, 2012. ACM.

[54] Jia Deng, Olga Russakovsky, Jonathan Krause, Michael S. Bernstein, Alex Berg, and Li Fei-Fei. Scalable Multi-label Annotation. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 3099–3102, New York, NY, USA, 2014. ACM.

[55] Hao Su, Jia Deng, and Li Fei-Fei. Crowdsourcing Annotations for Visual Object Detection. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, July 2012.

[56] Jianxiong Xiao, Tian Fang, Ping Tan, Peng Zhao, Eyal Ofek, and Long Quan. Image-based FaAde Modeling. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 161:1–161:10, New York, NY, USA, 2008. ACM.

[57] Jianxiong Xiao, Tian Fang, Peng Zhao, Maxime Lhuillier, and Long Quan. Image-based Street-side City Modeling. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 114:1–114:12, New York, NY, USA, 2009. ACM.

[58] Amir Roshan Zamir, Alexander Darino, and Mubarak Shah. Street View Challenge: Identification of Commercial Entities in Street View Imagery. In *Proceedings of the 2011 10th International Conference on Machine Learning and Applications and Workshops - Volume 02*, ICMLA '11, pages 380–383, Washington, DC, USA, 2011. IEEE Computer Society.

[59] Amir Roshan Zamir, Afshin Dehghan, and Mubarak Shah. GMCP-Tracker: Global Multi-object Tracking Using Generalized Minimum Clique Graphs. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part II*, ECCV'12, pages 343–356, Berlin, Heidelberg, 2012. Springer-Verlag.

[60] Amir Roshan Zamir and Mubarak Shah. Accurate Image Localization Based on Google Maps Street View. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 255–268, Berlin, Heidelberg, 2010. Springer-Verlag.

[61] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[62] Deva Ramanan, Pedro Felzenszwalb, and David McAllester. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[63] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001*, volume 1, pages I–511–I–518 vol.1, 2001.

[64] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, September 2010.

[65] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[66] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual Recognition with Humans in the Loop. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision ECCV 2010*, Lecture Notes in Computer Science, pages 438–451. Springer Berlin Heidelberg, September 2010. DOI: 10.1007/978-3-642-15561-1_32.

[67] Peter Welinder and Pietro Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. 2010.

[68] Mlanie Lagarrigue, Theodore Alexandrov, Gabriel Dieuset, Aline Perrin, Rgis Lavigne, Stphanie Baulac, Herbert Thiele, Benoit Martin, and Charles Pineau. New analysis workflow for MALDI imaging mass spectrometry: Application to the discovery and identification of potential markers of childhood absence epilepsy. *Journal of Proteome Research*, 11(11):5453–5463, November 2012.

[69] Peng Dai, Daniel S Weld, et al. Artificial intelligence for artificial artificial intelligence. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 1153–1159. AAAI Press, 2011.

[70] Suyog Dutt Jain and Kristen Grauman. Predicting Sufficient Annotation Strength for Interactive Foreground Segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[71] Christopher H. Lin, Mausam Mausam, and Daniel S. Weld. Dynamically Switching Between Synergistic Workflows for Crowdsourcing. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 87–93, Toronto, Ontario, Canada, 2012. AAAI Press.

[72] Ece Kamar, Severin Hacker, and Eric Horvitz. Combining Human and Machine Intelligence in Large-scale Crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 467–474, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[73] J.B. Kirschbaum, P.W. Axelson, P.E. Longmuir, K.M. Mispagel, J.A. Stein, and D.A. Yamada. Designing Sidewalks and Trails for Access, Part II of II: Best Practices Design Guide, Chapter 7. 2001.

[74] Daniel J. Hruschka, Deborah Schwartz, Daphne Cobb St.John, Erin Picone-Decaro, Richard A. Jenkins, and James W. Carey. Reliability in Coding Open-Ended Data: Lessons Learned from HIV Behavioral Research. *Field Methods*, 16(3):307–331, August 2004.

[75] Klaus Krippendorff. *Content analysis: An introduction to its methodology.* Sage, 2004.

[76] Dragomir Anguelov, Carole Dulong, Daniel Filip, Christian Frueh, Stphane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent, and Josh Weaver. Google Street View: Capturing the World at Street Level. *Computer*, 43(6):32–38, June 2010.

[77] Alexander J. Quinn and Benjamin B. Bederson. Human Computation: A Survey and Taxonomy of a Growing Field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1403–1412, New York, NY, USA, 2011. ACM.

[78] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 89–96, Washington, DC, USA, 2011. IEEE Computer Society.

[79] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, June 1998.

[80] Kotaro Hara, Jin Sun, Jonah Chazan, David Jacobs, and Jon Froehlich. An Initial Study of Automatic Curb Ramp Detection with Crowdsourced Verification Using Google Street View Images. In *HCOMP*, 2013.

[81] Ross B. Girshick, Pedro Felzenszwalb, and David McAllester. Discriminatively Trained Deformable Part Models, Release 5.

[82] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):36–51, 2008. 00372.

[83] Robert Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[84] Johannes Kopf, Billy Chen, Richard Szeliski, and Michael Cohen. Street Slide: Browsing Street Level Imagery. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 96:1–96:8, New York, NY, USA, 2010. ACM.

[85] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15, October 2008.

[86] Toward universal access: Americans with disabilities act sidewalk and curb ramp self-evaluation report. *City of Bellevue, WA*, September 2009.

[87] Moshe Bar. Visual objects in context. *Nature Reviews Neuroscience*, 5(8):617–629, August 2004.

[88] A Oliva and A Torralba. The role of context in object recognition. *Trends in cognitive sciences*, 11(12):520–527, 2007.

[89] S.K. Divvala, D. Hoiem, J.H. Hays, A.A. Efros, and M. Hebert. An empirical study of context in object detection. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009*, pages 1271–1278, 2009. 00155.

[90] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild.

[91] W. Ouyang, X. Zeng, and X. Wang. Single-Pedestrian Detection Aided by Two-Pedestrian Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1875–1889, September 2015.

[92] Peng Wang, Lingqiao Liu, Chunhua Shen, Zi Huang, Anton van den Hengel, and Heng Tao Shen. What's Wrong With That Object? Identifying Images of Unusual Objects by Modelling the Detection Score Distribution. pages 1573–1581, 2016.

[93] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *IEEE 11th International Conference on Computer Vision, 2007. ICCV 2007*, pages 1–8, 2007. 00365.

[94] Antonio Torralba. Contextual priming for object detection. *IJCV*, 53:2003, 2003. 00516.

[95] Tomasz Malisiewicz and Alyosha Efros. Beyond Categories: The Visual Memex Model for Reasoning About Object Relationships. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1222–1230. Curran Associates, Inc., 2009.

[96] H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1285–1292, 2010. 00062.

[97] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. 2016.

[98] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *ICLR*, 2014.

[99] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, June 2005.

[100] Keras: Deep learning library for theano and tensorflow.

[101] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[102] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Wider face: A face detection benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[103] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, October 2016.

[104] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision*, 2016.

[105] P. Anderson, B. Fernando, M. Johnson, and S. Gould. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision (ECCV)*, 2016.

[106] A. Chang, M. Savva, and C. D. Manning. Learning spatial knowledge for text to 3d scene generation. In *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2028–2038, Doha, Qatar, October 2014.

[107] W. Choi, Y. W. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[108] B. Coyne and R. Sproat. Wordseye: An automatic text-to-scene conversion system. In *28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 487–496, New York, NY, USA, 2001.

[109] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *International Conference on Computer Vision (ICCV)*, pages 2650–2658, Dec 2015.

[110] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on*, 32(9):1627–1645, Sept 2010.

[111] R. Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, 2015.

[112] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 2008.

[113] D. Hoiem and S. Savarese. *Representations and Techniques for 3D Object Recognition and Scene Interpretation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011.

[114] J. Johnson, R. Krishna, M. Stark, J. Li, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[115] P. Kordjamshidi, M. Van Otterlo, and M.-F. Moens. Spatial role labeling: Towards extraction of spatial relations from natural language. *ACM Trans. Speech Lang. Process.*, 8(3):4:1–4:36, Dec. 2011.

[116] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, pages 1–42, 2017.

[117] T. Kulkarni, I. Yildirim, P. Kohli, W. Freiwald, and J. B. Tenenbaum. Deep generative vision as approximate bayesian computation. In *NIPS 2014 ABC Workshop*, 2014.

[118] T. D. Kulkarni, V. K. Mansinghka, P. Kohli, and J. B. Tenenbaum. Inverse graphics with probabilistic CAD models. *CoRR*, abs/1407.1339, 2014.

[119] A. Li, V. I. Morariu, and L. S. Davis. Planar structure matching under projective uncertainty for geolocation. In *European Conference on Computer Vision (ECCV)*, 2014.

[120] D. Lin, S. Fidler, C. Kong, and R. Urtasun. Visual semantic search: Retrieving videos via complex textual queries. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[121] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei. Visual relationship detection with language priors. In *European Conference on Computer Vision*, 2016.

[122] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[123] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 3111–3119. 2013.

[124] L. D. Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard. Bayesian geometric modeling of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2719–2726, 2012.

[125] L. D. Pero, J. Bowdish, B. Kermgard, E. Hartley, and K. Barnard. Understanding bayesian rooms using composite 3d object models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 153–160, June 2013.

[126] Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39 – 62, 1999.

[127] S. Schuster, R. Krishna, A. Chang, L. Fei-Fei, and C. D. Manning. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *4th Workshop on Vision and Language*, pages 70–80, 2015.

[128] B. Siddiquie, B. White, A. Sharma, and L. S. Davis. Multi-modal image retrieval for complex queries using small codes. In *International Conference on Multimedia Retrieval (ICMR)*, 2014.

[129] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[130] J. M. Snyder. Interval analysis for computer graphics. *SIGGRAPH Comput. Graph.*, 26(2):121–130, July 1992.

[131] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[132] N. Srivastava and R. Salakhutdinov. Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research (JMLR)*, 15:2949–2980, 2014.

[133] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in Neural Information Processing Systems 28 (NIPS)*. 2015.

[134] Y. Xiang and S. Savarese. Object detection by 3d aspectlets and occlusion reasoning. In *International Conference on Computer Vision Workshops (ICCVW)*, Dec 2013.

[135] M. Z. Zia, M. Stark, and K. Schindler. Towards scene understanding with detailed 3d object representations. *International Journal of Computer Vision*, 112(2):188–203, 2015.

[136] C. L. Zitnick, D. Parikh, and L. Vanderwende. Learning the visual interpretation of sentences. In *International Conference on Computer Vision (ICCV)*, pages 1681–1688, 2013.