# From Restoring Human Vision to Enhancing Computer Vision

Thesis by

Yang Liu

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

Caltech

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2020
Defended June 2, 2020

# ACKNOWLEDGEMENTS

I wish to express my sincere appreciation for my adviser Dr. Markus Meister, who offered tremendous support and mentorship during my my entire PhD study at CalTech. I'm particularly thankful for his help with the conceptualization of the first project, which started my journey in vision research. I am also grateful for the freedom that Markus gave me to explore projects of different flavors.

I would also like thank my thesis committee chair, Dr. Pietro Perona, and committee members, Dr. Thanos Siapas and Dr. Yisong Yue, for their support, advice, and mentorship. I have enjoyed taking classes from each of them, and the knowledge from these classes helped me in building a firm foundation for my graduate research.

I would like to thank my family, lab members, classmates, and friends for their continued support, and for making my time at CalTech enjoyable. Last but not least, I would like to fiancée Dr. Kristina Dylla for always being there for me.

# ABSTRACT

The central theme of this work is enabling vision, which includes two subtopics: restoring vision for blind humans, and enhancing computer vision models in visual recognition. Chapter 1 first provides a gentle introduction to relevant high level principles of human visual computations and summarizes two fundamental questions that vision answers: “what" and “where." Chapters 2, 3, and 4 contain three published projects that are anchored by those two fundamental questions.

Chapter 2 introduces a cognitive assistant to restore visual function for blind humans by focusing on an interface powered by audio augmented reality. The assistant communicates the “what" and “where" aspects of visual scenes by a combination of natural language and spatialized sound. We experimentally demonstrated that the assistant enables many aspects of visual functions for naive blind users.

Chapters 3 and 4 develop data augmentation methods to address the data inefficiency problem in neural network based computer visual recognition models. In Chapter 3, a 3D-simulation based data augmentation method is developed for improving the generalization of visual classification models for rare classes. In Chapter 4, a fast and efficient data augmentation method is developed for the newly formulated panoptic segmentation task. The method improves performance of state-of-the-art panoptic segmentation models and generalizes across dataset domains, sizes, model architectures, and backbones.

# PUBLISHED CONTENT AND CONTRIBUTIONS

Beery, Sara et al. (Mar. 2020). "Synthetic Examples Improve Generalization for Rare Classes". In: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*. URL: `http://openaccess.thecvf.com/content_WACV_2020/papers/Beery_Synthetic_Examples_Improve_Generalization_for_Rare_Classes_WACV_2020_paper.pdf`.
Y.L participated in the conceptualization of the project, developed the data simulation pipeline, participated in training and evaluation of networks, data analysis and visualization, and writing of manuscript.

Liu, Yang, Pietro Perona, and Markus Meister (2019). "PanDA: Panoptic Data Augmentation". In: *arXiv preprint arXiv:1911.12317*. URL: `https://arxiv.org/abs/1911.12317`.
Y.L participated in all aspects of this project.

Meister, Markus and Yang Liu (July 2019). *Systems and methods for generating spatial sound information relevant to real-world environments*. US Patent 10,362,429.
Y.L participated in the conceptualization of the project, experimental design, hardware setup, software implementation, investigation, data analysis and visualization, and writing of manuscript.

Liu, Yang, Noelle RB Stiles, and Markus Meister (2018). "Augmented reality powers a cognitive assistant for the blind". In: *eLife* 7, e37841. DOI: `10.7554/eLife.3784`.
Y.L participated in the conceptualization of the project, experimental design, hardware setup, software implementation, investigation, data analysis and visualization, and writing of manuscript.

# TABLE OF CONTENTS

# LIST OF ILLUSTRATIONS

# LIST OF TABLES

*Chapter 1*

# INTRODUCTION

## ABSTRACT

This thesis consists of three main chapters. Each is a published research article with appropriate sections from introduction to conclusions. This introductory chapter provides a high-level introduction and overview of the common theme that connects the three research articles, namely enhancing human and computer vision. The chapter first briefly introduces some high-level principles of the human visual computation and challenges to restore vision. It then provides an overview of existing approaches to help people with vision loss and why they fail. In the second section, the idea of using modern computer vision to guide the blind is explained. An assistant for restoring visual functions of blind humans with a focus on intuitive low bandwidth user interface is proposed. In the last section, we identify the data inefficiency problem in modern computer visual recognition problems and two data augmentation methods are proposed to address the problem.

## 1.1 Human Vision and Blindness

About 216 million people are visually impaired globally, of which 36 million are blind (R. Bourne et al., 2017; Seth R Flaxman et al., 2017). For most sighted people like ourselves to appreciate the importance of our vision, one needs no more than to try a few minutes of simple daily routines, like having lunch and reading news, with eyes closed. Consistent with our subjective experience, studies have shown that vision loss drastically impacts people's quality of life (Evans, 1989; Frick et al., 2007).

To develop a deep understanding of visual impairment, it is useful to understand normal human vision first. There are many ways to understand biological vision, and a relevant, classic definition from David Marr describes vision as "knowing what is where by looking." (Marr, 1982) Much of vision neuroscience research has been inspired by this succinct definition of vision. The well-known two-stream hypothesis (Goodale, Milner, et al., 1992) argues that the human visual system has two separate streams of process that answer the "what" and "where" questions respectively. The ventral pathway that leads to the temporal lobe is primarily responsible for visual recognition and identification, and the dorsal pathway that leads to the parietal lobe processes the spatial location of objects relative to the viewer. Chapters in the rest of the thesis are anchored by answering the two fundamental questions of vision for either blind humans or computers.

Despite the ostensible straightforwardness of "what" and "where" questions, the underlying computation performed by our brain to answer these two questions is complex. From an information processing point of view, the human visual system takes in about 1 gigabit of raw image information every second at the the first stage – the retina. In sharp contrast, it is estimated that only tens of bits of information are extracted to guide our thoughts and actions (Pitkow and Meister, 2014). To perform this eight orders of magnitude information extraction, the human brain dedicates a large proportion of its cortical area to visual processing (Kandel et al., 2000).

With the dominant causes of blindness worldwide being age-related diseases of the eye, the normal flow of visual data from the eye to the brain is blocked, leaving the sophisticated visual area largely out of action despite the cortical plasticity. The number of people affected by the common causes of vision loss has increased substantially as the population increases and ages (Seth R Flaxman et al., 2017). A straightforward idea to fix blindness caused by malfunctioning eyes is to try and repair the eye. The first option is to repair the eye biologically, and a wide range of treatments involving gene therapy, stem cells, or transplantation are being explored(Scholl et al., 2016). The second option is to bring the image into the brain through alternate means. The most direct route is electrical stimulation of surviving cells in the retina (Stingl and Zrenner, 2013; Weiland and Humayun, 2014) or of neurons in the visual cortex (Dobelle, Mladejovsky, and Girvin, 1974). The clear advantages of such methods are that they make the powerful visual areas in the brain useful again and can potentially restore natural visual perception. However, such methods are likely to remain decades away from being widely accessible by the blind. Because, in addition to the need of further technological development, such methods are generally invasive medical procedures and therefore require government agency approval, and may come at prohibitively high cost.

A second class of approaches seeks to restore visual functions such as "knowing what is where" rather than visual sensation itself. This approach involves rerouting visual information to a different sensory channel, such as hearing(Meijer, 1992; Auvray, Hanneton, and O'Regan, 2007; Capelle et al., 1998) and touch (Stronks et al., 2016). Such approaches can often be made non-invasive, therefore they are generally safer and cheaper. However, two major challenges are present. First, the required bit rate of for raw visual imagery transmission is exceptionally high, and other sensory modalities often have lower bandwidths. Second, despite the adaptive plasticity of the human cortex, non-visual cortical areas are ill-equipped to process information in a raw imagery format. Due to a lack of sophisticated algorithms, all of the aforementioned approaches apply naive and aggressive pixel-space down sampling to reduce bandwidth to match that of the receiving channel. A lot of useful visual information is lost during this simple-minded down-sampling process. In addition, the down-sampled information often maintains the original format of images, which is unintuitive to understand and learn. In summary, these approaches failed to deliver any practical restoration of visual functions to the blind. It is apparent that a more intelligent way is needed to both reduce the data rate and translate the information into an intuitive format.

## 1.2 Enabling Vision for the Blind with Computer Vision

Fortunately, one of the major goals of computer vision is to engineer computer systems that understand high-level information as the human visual system does. This means translating visual images to descriptions of the world that can be used to guide behaviors. Since its inception in 1960s, computer vision has developed into a large scientific discipline that consists of many subdomains such as image restoration, scene reconstruction, and event detection.

In the past two decades, the thanks to the exponential growth of computational power, the development and popularization of various flavors of artificial neural networks, and the availability of large-scale annotated datasets, computer vision systems have seen revolutionary improvements. Modern deep convolutional neural network (DCNN) powered models often yield top performance across many computer vision benchmarks (Olga Russakovsky et al., 2014; K. He, X. Zhang, et al., 2015). In certain visual recognition tasks, they can even rival or outperform humans (Olga Russakovsky et al., 2014; K. He, X. Zhang, et al., 2015). Although, a general purpose computer vision model that matches every aspect of human vision has yet to be invented. Many domain-specific visual recognition models have proven useful in a wide range of real world applications, including agriculture (Barth et al., 2018), medicine (Poh, McDuff, and Picard, 2010; Litjens et al., 2017), wildlife preservation (Beery, Van Horn, and Perona, 2018), manufacturing (Lee, Cheon, and Kim, 2017), and transportation (B. Wu et al., 2017).

In 1985, long before the advent of powerful deep neural network based computer vision models which rival human performance, Collins (Collins, 1985) envisioned a future of assistive technology for the blind with the help of computer vision:

> For the ideal mobility system of the future, I strongly believe that we should take a more sophisticated approach, utilizing the power of artificial intelligence for processing large amounts of detailed visual information in order to substitute for the missing functions of the eye and much of the visual pre-processing performed by the brain. We should off-load the blind traveler's brain of these otherwise slow and arduous tasks which are normally performed effortlessly by the sighted visual system.

Decades later, this work aims to revitalize the prophetic idea from Collins with the help of modern computer vision. Specifically, the proposed system that helps the blind can be broken down to two components: 1) a front-end **computer vision** model that extracts high-level information from the scene, and 2) a back-end **user interface** that communicates such information to the blind user at a comfortable bandwidth and in an intuitive format. The next three chapters contain three independent projects that focus on the two components: restoration of human vision and enhancing computer vision.

In Chapter 2, we asked the question of what auditory user interface is efficient and intuitive for communicating visual information to the blind, given that all relevant information has already been extracted and made available by a hypothetical front-end system. Specifically, the interface uses natural language to communicate the identities of objects, which answers the “what" question of vision, and spatialized sound to inform object locations, which answers the“where" aspect of vision. The hardware platform is a self-contained wearable computer that not only serves as a surrogate of the hypothetical front-end system by provided simulate information, but also powers a real world cognitive augmented reality assistant (CARA) for the blind by being able to sample and store real world information. Behavior experiments were conducted with blind human subjects to test the efficacy of the proposed system. The system supports many aspects of visual cognition for the blind without training: from obstacle avoidance to formation and recall of spatial memories, to line-of-sight and multi-story building navigation.

## 1.3 Enhancing Computer Vision with Data Augmentation

Having established the effectiveness of CARA, a system focusing on the back-end user interface component, Chapters 3 and 4 move forward to improve the front-end component, namely computer vision models that extract high level information from the visual scene. This line of research is motivated not only by the special need of a front-end for the blind, but also the author’s general interest in enabling a computer to see as humans do.

Among many tasks of computer vision, visual recognition (Jia Deng et al., 2009; Olga Russakovsky et al., 2014) is a the classical one that aims to answer the “what" aspect of an image: to assign a class or category to the image. Its generalized variants such as object detection and segmentation also answer the “where" aspect of vision by localizing objects or points of interest in the form of indicating locations, boundaries, or providing pixel-level labels of object classes and identities (Lin, Maire, et al., 2014; Kirillov et al., 2019; Krasin et al., 2017). Together, these computer vision tasks and models are of particular interest to assisting the blind since they answer the fundamental “what" and “where" problems of vision.

The availability of high-quality, large-scale annotated datasets such as ImageNet (Jia Deng et al., 2009) and Microsoft COCO (Lin, Maire, et al., 2014) has been an indispensable driving force for supervised computer vision models. The quality and scale of the dataset often have decisive impact on computer vision models. Studies have shown that even mild label corruption can significantly reduce the generalization performance of DCNN-based models (C. Zhang et al., 2016; Sukhbaatar et al., 2014). State-of-the-art models in many visual benchmarks often need thousands of training samples to reach desirable performance (Xiong et al., 2019; Porzi et al., 2019; K. He, X. Zhang, et al., 2015; Szegedy, Ioffe, et al., 2017). Additionally, many real-world visual recognition problems have long-tailed (Van Horn and Perona, 2017) class distributions, and generalization performance is usually poor for rare classes (Beery, Y. Liu, Morris, Piavis, Kapoor, Joshi, et al., 2020).

In general, there are several approaches to address the “data hunger" issue. First, a technique called transfer-learning can be used to store knowledge gained while solving one problem and applying it to a related problem. Specifically, for visual recognition tasks such as object detection, image segmentation, and action recognition, a widely used method is to pre-train on large datasets such as ImageNet and fine-tune on target task datasets. In fact, many state-of-the-art visual recognition models are obtained with the help of this approach (Lin, Goyal, et al., 2018; Redmon and Farhadi, 2018; K. He, Gkioxari, et al., 2017; Xiong et al., 2019; Porzi et al., 2019). Second, more data efficient models that are capable of learning visual concepts from a few examples are being developed (Li, Fergus, and Perona, 2006). Several models have shown promising results on simple benchmarks with low capacity models (Hariharan and Girshick, 2017; Pahde et al., 2019) but have yet to prove their usefulness on state-of-the-art high-capacity models. Finally, data augmentation methods aim to supplement limited datasets by generating new data. This is usually achieved by either directly generating new data with additional resources, such as simulation and generative models (Varol, Romero, X. Martin, Mahmood, Michael J. Black, et al., 2017a; Pepik et al., 2015; Hinterstoisser et al., 2019; Rajpura, Bojinov, and Hegde, 2017; Goodfellow et al., 2014; Shrivastava et al., 2017; Stephan R. Richter et al., 2016a; Peng et al., 2018; Hattori et al., 2015), or synthesizing data by applying invariant transforms to existing data (Bousmalis et al., 2017; Gregor et al., 2015; Im et al., 2016; Radford, Metz, and Chintala, 2015; Tran et al., 2017; Luan et al., 2017; J.-Y. Zhu et al., 2017). In Chapters 3 and 4, two data augmentation methods were developed to improve state-of-the-art models (Beery, Y. Liu, Morris, Piavis, Kapoor, Joshi, et al., 2020; Y. Liu, Perona, and Meister, 2019).

Chapter 3 is focused on the data imbalance issue of real-world visual classification problems. Specifically, our testbed is animal species classification, which has a real-world long-tailed distribution. Driven largely by the entertainment industry, modern 3D game development engines can render near photo-realistic 3D environments in real time and can rapidly generate large-scale, high quality images with ground truth. Two natural world simulators powered by such game engines are developed, and effects of different axes of variation in simulation, such as pose, lighting, model, and simulation method, are analyzed. In addition, we prescribe best practices for efficiently incorporating simulated data for real-world performance gain. Experiments revealed three main conclusions: 1) synthetic data can significantly reduce error rates for rare classes, 2) target class error decreases as more simulated data are added to the training, and 3) high variation of simulated data provides maximum performance gain.

Chapter 4 addresses the data deficit problem of the newly formulated panoptic segmentation task which unifies semantic segmentation and instance segmentation tasks (Kirillov et al., 2019). The panoptic data augmentation (PanDA) method is inspired by two observations: 1) humans takes advantage of motion parallax cues for segmenting foreground from background, and 2) the unexpected effectiveness of pasting cropped foreground images on empty background images in Chapter 3. PanDA takes advantage of semantic- and instance-invariant transformations in pixel space and generates fully annotated new training images from the original training dataset without any additional data input. Unmodified state-of-the-art panoptic segmentation models were retrained on PanDA augmented datasets generated with a single frozen set of parameters. We experimentally showed robust performance gains in panoptic segmentation, instance segmentation, as well as object detection tasks across models, backbones, dataset domains, and scales. An important additional insight originating from the unrealistic-looking images generated by PanDA, which is in sharp contrast with existing data augmentation methods in this domain (S. Liu et al., 2019; Beery, Y. Liu, Morris, Piavis, Kapoor, Meister, et al., 2019; H.-S. Fang et al., 2019; R. Shetty, Schiele, and Fritz, 2019; Dvornik, Mairal, and Schmid, 2019). The effectiveness of such unrealistic-looking images suggests that photo-realism is not necessary for synthesis by data augmentation, and one should rethink optimizing for image realism for future data augmentation methods.

*Chapter 2*

# AUGMENTED REALITY POWERS A COGNITIVE ASSISTANT FOR THE BLIND

Liu, Yang, Noelle RB Stiles, and Markus Meister (2018). "Augmented reality powers a cognitive assistant for the blind". In: *eLife* 7, e37841. DOI: 10.7554/eLife.3784.

# ABSTRACT

To restore vision for the blind, several prosthetic approaches have been explored that convey raw images to the brain. So far, these schemes all suffer from a lack of bandwidth and the extensive training required to interpret unusual stimuli. Here we present an alternate approach that restores vision at the cognitive level, bypassing the need to convey sensory data. A wearable computer captures video and other data, extracts the important scene knowledge, and conveys that through auditory augmented reality. This system supports many aspects of visual cognition: from obstacle avoidance to formation and recall of spatial memories, to long-range navigation. Neither training nor modification of the physical environment are required: blind subjects can navigate an unfamiliar multi-story building on their first attempt. The combination of unprecedented computing power in wearable devices with augmented reality technology promises a new era of non-invasive prostheses that are limited only by software.

## 2.1 Introduction

About 36 million people are blind worldwide (R. Bourne et al., 2017). In industrialized nations, the dominant causes of blindness are age-related diseases of the eye, all of which disrupt the normal flow of visual data from the eye to the brain. In some of these cases, biological repair is a potential option, and various treatments are being explored involving gene therapy, stem cells, or transplantation (Scholl et al., 2016). However, the dominant strategy for restoring vision has been to bring the image into the brain through alternate means. The most direct route is electrical stimulation of surviving cells in the retina (Stingl and Zrenner, 2013; Weiland and Humayun, 2014) or of neurons in the visual cortex (Dobelle, Mladejovsky, and Girvin, 1974). Another option involves translating the raw visual image into a different sensory modality (Loomis, Klatzky, and Giudice, 2012; Maidenbaum, Abboud, and Amedi, 2014; Proulx et al., 2016), such as touch (Stronks et al., 2016) or hearing (Auvray, Hanneton, and O'Regan, 2007; Capelle et al., 1998; Meijer, 1992). So far, none of these approaches has enabled any practical recovery of the functions formerly supported by vision. Despite decades of effort, all users of such devices remain legally blind (Luo and Da Cruz, 2016; Katarina Stingl et al., 2017; Striem-Amit, Guendelman, and Amedi, 2012; Stronks et al., 2016). While one can certainly hope for progress in these domains, it is worth asking what the fundamental obstacles to restoration of visual function are.

The human eye takes in about 1 gigabit of raw image information every second, whereas our visual system extracts from this just tens of bits to guide our thoughts and actions (Pitkow and Meister, 2014). All the above approaches seek to transmit the raw image into the brain. This requires inordinately high data rates. Further, the signal must arrive in the brain in a format that can be interpreted usefully by the visual system or some substitute brain area to perform the key steps of knowledge acquisition, like scene recognition and object identification. None of the technologies available today deliver the high data rate required to retain the relevant details of a scene, nor do they produce a neural code for the image that matches the expectations of the human brain, even given the prodigious degree of adaptive plasticity in the nervous system.

Three decades ago, one of the pioneers of sensory substitution articulated his vision of a future visual prosthesis (Collins, 1985):

> I strongly believe that we should take a more sophisticated approach, utilizing the power of artificial intelligence for processing large amounts

> of detailed visual information in order to substitute for the missing functions of the eye and much of the visual pre-processing performed by the brain. We should off-load the blind travelers' brain of these otherwise slow and arduous tasks which are normally performed effortlessly by the sighted visual system.

Whereas at that time the goal was hopelessly out of reach, today's capabilities in computer vision, artificial intelligence, and miniaturized computing power are converging to make it realistic.

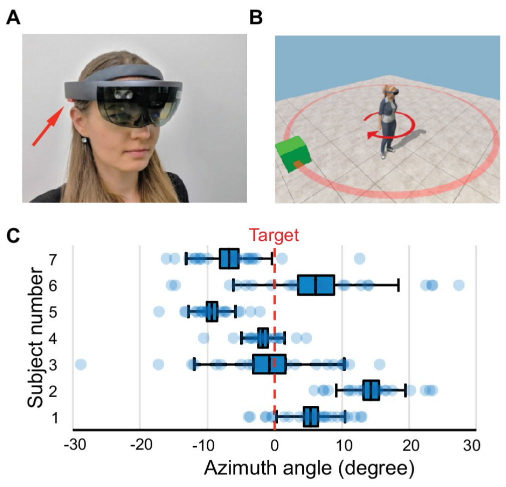
Here, we present such an approach that bypasses the need to convey the sensory data entirely, and focuses instead on the important high-level knowledge, presented at a comfortable data rate and in an intuitive format. We call the system CARA: a cognitive augmented reality assistant for the blind.

## 2.2 Results

### Design Principles

CARA uses a wearable augmented reality device to give voices to all the relevant objects in the environment (Fig. 2.1A). Unlike most efforts at scene sonification (Bujacz and Strumiłło, 2016; Csapó and Wersényi, 2013), our system communicates through natural language. Each object in the scene can talk to the user with a voice that comes from the object's location. The voice's pitch increases as the object gets closer. The user actively selects which objects speak through several modes of control (Fig. 2.6): In Scan mode, the objects call out their names in sequence from left to right, offering a quick overview of the scene. In Spotlight mode, the object directly in front speaks, and the user can explore the scene by moving the head. In Target mode, the user selects one object that calls repeatedly at the press of a clicker. In addition, any surface in the space emits a hissing sound as a collision warning when the user gets too close (Fig. 2.6).

The system is implemented on the Microsoft HoloLens (Fig. 2.1A), a powerful head-mounted computer designed for augmented reality (Hoffman et al., 2016). The HoloLens scans all surfaces in the environment using video and infrared sensors, creates a 3D map of the surrounding space, and localizes itself within that volume to a precision of a few centimeters (Fig. 2.7). It includes a see-through display for digital imagery superposed on the real visual scene; open-ear speakers that augment auditory reality while maintaining regular hearing; and an operating system that



D

| # | Age /gender | Blinding disease | Blindness type/onset | Visual perception | Hearing deficit | Hearing aid |
|---|---|---|---|---|---|---|
| 1 | 71 M | R.O.P. | Congenital | No | L,R | L,R |
| 2 | 58 F | A.O.I.D | 2000 | No | No | No |
| 3 | 64 M | R.O.P. | Congenital | No | R | No |
| 4 | 64 M | R.O.P. | Congenital | No | No | No |
| 5 | 66 F | R.O.P. | Congenital | No | No | No |
| 6 | 40 M | Unknown | Congenital | L.P. | L | No |
| 7 | 48 F | R.P. | 1987 | L.P. | L,R | L,R |

R.O.P. = Retinopathy of prematurity

A.O.I.D. = Auto optic immune deficiency

R.P. = Retinitis pigmentosa L.P. = Light perception

L/R = Left/right ear

Figure 2.1: **Hardware platform and object localization task.** (A) The Microsoft HoloLens wearable augmented reality device. Arrow points to one of its stereo speakers. (B) In each trial of the object localization task, the target (green box) is randomly placed on a circle (red). The subject localizes and turns to aim at the target. (C) Object localization relative to the true azimuth angle (dashed line). Box denotes s.e.m., whiskers s.d. (D) Characteristics of the seven blind subjects.

implements all the localization functions and provides access to the various sensor streams.

Any cognitive assistant must both acquire knowledge about the environment and then communicate that knowledge to the user. Tracking and identifying objects and people in a dynamic scene still presents a challenge, but those capabilities are improving at a remarkable rate (Jafri et al., 2014; Verschae and Ruiz-del-Solar, 2015), propelled primarily by interests in autonomous vehicles (see also Technical extensions below). Anticipating that the acquisition problems will be solved shortly, we focus here on the second task, the interface to the user. Thus, we populated the real-space volume scanned by the HoloLens with virtual objects that interact with the user. The applications were designed using the Unity game development platform which allows tracking of the user's head in the experimental space; the simulation of virtual objects; the generation of speech and sounds that appear to emanate from specific locations; and interaction with the user via voice commands and a clicker.

## Human Subject Tests

After a preliminary exploration of these methods, we settled on a fixed experimental protocol and recruited seven blind subjects (Fig. 2.1D). Subjects heard a short explanation of what to expect, then donned the HoloLens and launched into a series of four fully automated tasks without experimenter involvement. No training sessions were provided, and all the data were gathered within a 2 hr visit.

## Object Localization

Here, we tested the user's ability to localize an augmented reality sound source (Fig. 1B). A virtual object placed randomly at a 2 m distance from the subject called out “box" whenever the subject pressed the clicker. The subject was asked to orient the head towards the object and then confirm the final choice of direction with a voice command. All subjects found this a reasonable request and oriented surprisingly well, with an accuracy of 3–12 deg (standard deviation across trials, Fig. 2.1C). Several subjects had a systematic pointing bias to one or the other side of the target (9 to +13 deg, Fig. 2.1C), but no attempt was made to correct for this bias. These results show that users can accurately localize the virtual voices generated by HoloLens, even though the software used a generic head-related transfer function without customization.

## Spatial Memory

Do object voices help in forming a mental image of the scene (Lacey and Lawson, 2013) that can be recalled for subsequent decisions? A panel of five virtual objects was placed in the horizontal plane 2 m from the subject, spaced 30 degrees apart in azimuth (Fig. 2A). The subject scanned this scene actively using the Spotlight mode for 60 s. Then the object voices were turned off, and we asked the subject to orient towards the remembered location of each object, queried in random order. All subjects performed remarkably well, correctly recalling the arrangement of all objects (Fig. 2.2B, Fig. 2.8) with just one error (1/28 trials). Even the overall scale of the scene and the absolute positions of the objects were reproduced well from memory, to an average accuracy of  15 deg (rms deviation from true position, Fig. 2.2C–D). In a second round, we shuffled the object positions and repeated the task. Here three of the subjects made a mistake, presumably owing to interference with the memory formed on the previous round. Sighted subjects who inspected the scene visually performed similarly on the recall task (Fig. 2.8). These experiments suggest that active exploration of object voices builds an effective mental representation of the scene that supports subsequent recall and orientation in the environment.

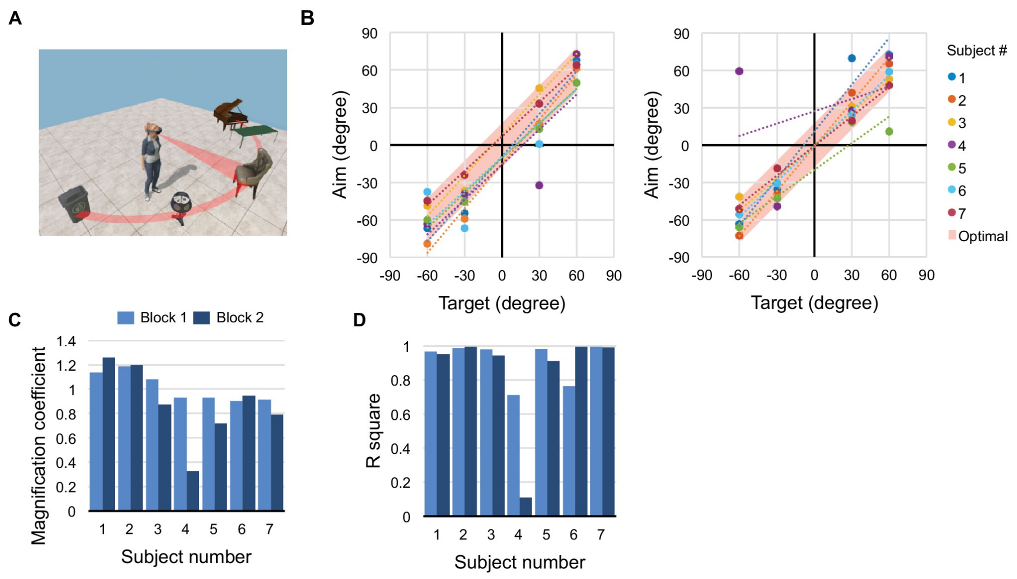


Figure 2.2: **Spatial memory task.** (A) Five objects are arranged on a half-circle; the subject explores the scene, then reports the recalled object identities and locations. (B) Recall performance during blocks 1 (left) and 2 (right). Recalled target angle potted against true angle. Shaded bar along the diagonal shows the 30 deg width of each object; data points within the bar indicate perfect recall. Dotted lines are linear regressions. (C) Slope and (D) correlation coefficient for the regressions in panel (B).

## Direct Navigation

Here, the subject was instructed to walk to a virtual chair, located 2 m away at a random location (Fig. 2.3A). In Target mode, the chair called out its name on every clicker press. All subjects found the chair after walking essentially straight-line trajectories (Fig. 2.3B–C, Fig. 2.9). Most users followed a two-phase strategy: first localize the voice by turning in place, then walk swiftly toward it (Fig. 2.9D–E). On rare occasions ( 5 of 139 trials), a subject started walking in the opposite direction, then reversed course (Fig. 2.9C), presumably owing to ambiguities in azimuthal sound cues (McAnally and R. L. Martin, 2014). Subject seven aimed consistently to the left of the target (just as in the task of Fig. 2.1) and thus approached the chair in a spiral trajectory (Fig. 2.3C). Regardless, for all subjects, the average trajectory was only 11–25% longer than the straight-line distance (Fig. 2.3E, Fig. 2.9A).

For comparison, we asked subjects to find a real chair in the same space using only their usual walking aid (Fig. 2.3D). These searches took on average eight times longer and covered 13 times the distance needed with CARA. In a related series of

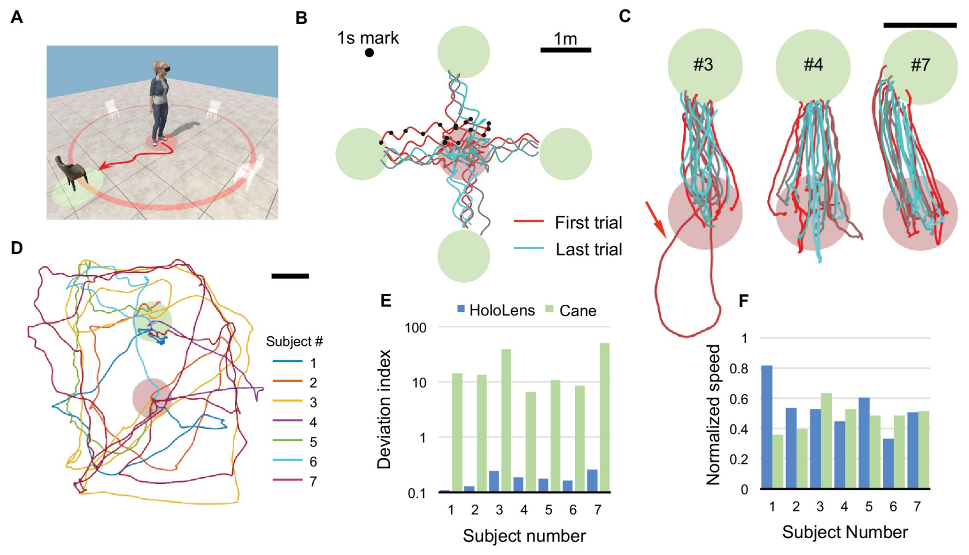


Figure 2.3: **Direct navigation task.** (A) For each trial, a target chair is randomly placed at one of four locations. The subject begins in the starting zone (red shaded circle), follows the voice of the chair, and navigates to the target zone (green shaded circle). (B) All raw trajectories from one subject (#6) including 1 s time markers. Oscillations from head movement are filtered out in subsequent analysis. (C) Filtered and aligned trajectories from all trials of 3 subjects (#3, 4, 7). Arrow highlights a trial where the subject started in the wrong direction. (D) Trajectories of subjects performing the task with only a cane and no HoloLens. (E) Deviation index, namely the excess length of the walking trajectory relative to the shortest distance between start and target. Note logarithmic axis and dramatic difference between HoloLens and Cane conditions. (F) Speed of each subject normalized to the free-walking speed.

experiments, we encumbered the path to the target with several virtual obstacles. Using the alarm sounds, our subjects weaved through the obstacles without collision (Fig. 2.10D). Informal reports from the subjects confirmed that steering towards a voice is a natural function that can be performed automatically, leaving attentional bandwidth for other activities. For example, some subjects carried on a conversation while following CARA.

## Long Range Guided Navigation

If the target object begins to move as the subject follows its voice, it becomes a “virtual guide". We designed a guide that follows a precomputed path and repeatedly calls out “follow me". The guide monitors the subject’s progress, and stays at most 1 m ahead of the subject. If the subject strays off the path, the guide stops and waits

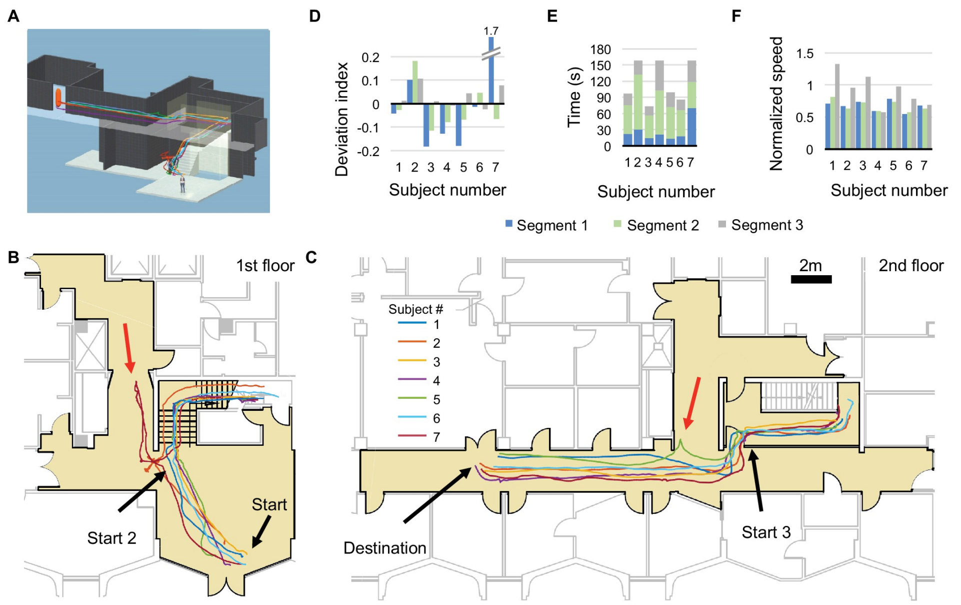


Figure 2.4: **Long-range guided navigation task.** (A) 3D reconstruction of the experimental space with trajectories from all subjects overlaid. (B and C) 2D floor plans with all first trial trajectories overlaid. Trajectories are divided into three segments: lobby (Start – Start 2), stairwell (Start 2 – Start 3), and hallway (Start 3 – Destination). Red arrows indicate significant deviations from the planned path. (D) Deviation index (as in Fig. 3E) for all segments by subject. Outlier corresponds to initial error by subject 7. Negative values indicate that the subject cut corners relative to the virtual guide. (E) Duration and (F) normalized speed of all the segments by subject.

for the subject to catch up. The guide also offers warnings about impending turns or a flight of stairs. To test this design, we asked subjects to navigate a campus building that had been pre-scanned by the HoloLens (Fig. 2.4A, Fig. 2.11). The path led from the ground-floor entrance across a lobby, up two flights of stairs, around several corners and along a straight corridor, then into a second floor office (Fig. 2.4B–C). The subjects had no prior experience with this part of the building. They were told to follow the voice of the virtual guide, but given no assistance or coaching during the task.

All seven subjects completed the trajectory on the first attempt (Fig. 2.4B–C). Subject seven transiently walked off course (Fig. 2.4B), due to her left-ward bias (Fig. 2.1C and Fig. 2.3C), then regained contact with the virtual guide. On a second attempt, this subject completed the task without straying. On average, this task required 119 s (range 73–159 s), a tolerable investment for finding an office in an unfamiliar building (Fig. 2.4E). The median distance walked by the subjects was 36 m (Fig. 2.4D), slightly shorter (1%) than the path programmed for the virtual guide, because the subjects can cut corners (Fig. 2.4C). The subjects' speed varied with difficulty along the route, but even on the stairs they proceeded at 60% of their free-walking speed (Fig. 2.4F). On arriving at the office, one subject remarked, "That was fun! When can I get one?" Other comments from subjects regarding user experience with CARA are provided in Supplementary Observations.

### Technical Extensions

As discussed above, the capabilities for identification of objects and people in a dynamic scene are rapidly developing. We have already implemented real-time object naming for items that are easily identified by the HoloLens, such as standardized signs and bar codes (Sudol et al., 2010) (Fig. 2.10A–B). Furthermore, we have combined these object labels with a scan of the environment to compute in real time a navigable path around obstacles toward any desired target (Fig. 2.10C). In the few months since our experimental series with blind subjects, algorithms have appeared that come close to a full solution. For example, YOLO (Redmon and Farhadi, 2018) will readily identify objects in a real time video feed that match one of 9000 categories. The algorithm already runs on the HoloLens, and we are adopting it for use within CARA (Fig. 2.10F).

### An Open-source Benchmarking Environment for Assistive Devices

The dramatic advances in mobile computing and machine vision are enabling a flurry of new devices and apps that offer assistive functions for the vision impaired. To coordinate these developments, one needs a reliable common standard by which to benchmark and compare different solutions. In several domains of engineering, the introduction of a standardized task with a quantitative performance metric has stimulated competition and rapid improvement of designs (Berens et al., 2018; Russakovsky et al., 2015).

On this background, we propose a method for the standardized evaluation of different assistive devices for the blind. The user is placed into a virtual environment implemented on the HTC Vive platform (*HTC Vive* n.d.).This virtual reality kit is widely used for gaming and is relatively affordable. Using this platform, researchers anywhere in the world can replicate an identical environment and use it to benchmark their assistive methods. This avoids having to replicate and construct real physical spaces.

At test time, the subject dons a wireless headset and moves freely within a physical space of 4 m x 4 m. The Vive system localizes position and orientation of the headset in that volume. Based on these data, the virtual reality software computes the subject's perspective of the virtual scene, and presents that view through the headset's stereo goggles. An assistive device of the experimenter's choice can use that same real-time view of the environment to guide a blind or blind-folded subject through the space. This approach is sufficiently general to accommodate designs ranging from raw sensory substitution – like vOICe (Meijer, 1992) and BrainPort (Stronks et al., 2016) – to cognitive assistants like CARA. The tracking data from the Vive system then serve to record the user's actions and evaluate the performance on any given task.

To illustrate this method, we constructed a virtual living room with furniture (Fig. 5A). Within that space, we defined three tasks that involve (1) scene understanding, (2) short-range navigation, and (3) finding a small object dropped on the floor. To enable blind subjects in these tasks we provided two assistive technologies: (a) the high-level assistant CARA, using the same principle of talking objects as described above on the HoloLens platform; and (b) the low-level method vOICe that converts photographs to soundscapes at the raw image level (Meijer, 1992). The vOICe system was implemented using software provided by its inventor (*Seeing with Sound* n.d.).

Here, we report performance of four subjects, all normally sighted. Each subject was given a short explanation of both CARA and vOICe. The subject was allowed to practice (10 min) with both methods by viewing the virtual scene while either CARA or vOICe provided translation to sound delivered by headphones. Then, the subjects were blindfolded and performed the three tasks with sound alone. Each task consisted of 20 trials with randomly chosen goals, and a time limit of 60 s was applied to each trial.

On the first task, the subject stood in the middle of the virtual living room and was asked to locate one of the objects and point at it with the head. With CARA, subjects mostly used the Target mode to efficiently find the desired object, and located it based on the 3D sound cues, with a typical aiming error of 10 degrees (Fig. 2.5B, bias of 0.2–13 deg, accuracy 4.5–27 deg). With vOICe, subjects reported great difficulty with identifying objects, despite the earlier opportunity to

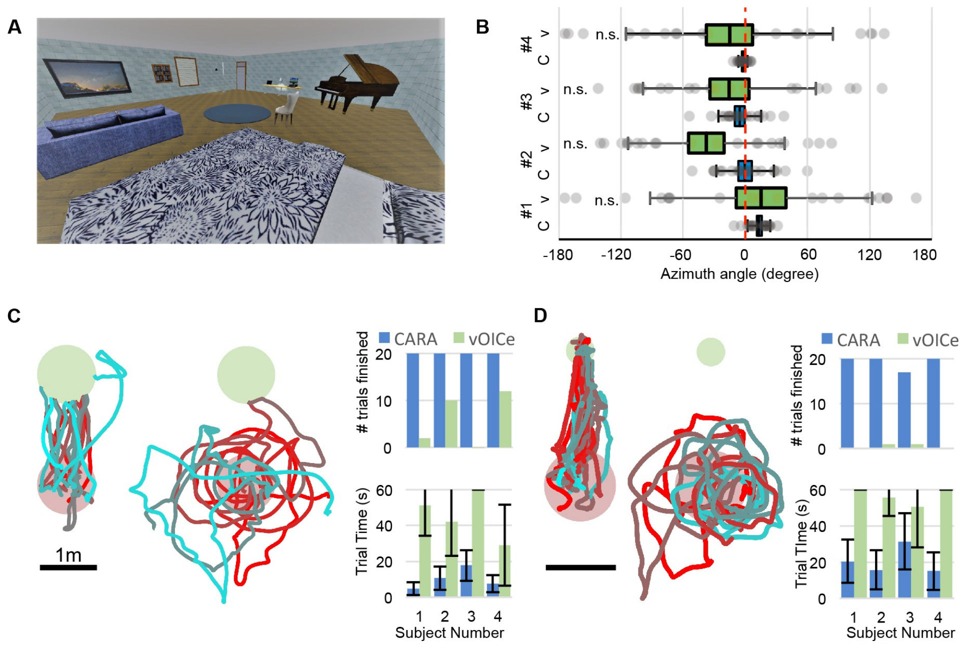


Figure 2.5: **Benchmark testing environment.** (A) A virtual living room including 16 pieces of furniture and other objects. (B) Localization of a randomly chosen object relative to the true object location (0 deg, dashed line) for four subjects using CARA (C) or vOICe (V). Box denotes s.e.m., whiskers s.d. For all subjects, the locations obtained with vOICe are consistent with a uniform circular distribution (Rayleigh z test, p>0.05). (C) Navigation toward a randomly placed chair. Trajectories from one subject using CARA (left) and vOICe (middle), displayed as in Fig. 2.3C. Right: Number of trials completed and time per trial (mean ±s.d.). (D) Navigation toward a randomly placed key on the floor (small green circle). Trajectories and trial statistics displayed as in panel C.

practice with visual feedback. Their aiming choices were statistically consistent with a uniform random distribution (Fig. 2.5B).

On the second task, the subject was asked to walk from the middle of the arena toward a chair placed randomly in one of four locations, as in the directed navigation task of Fig. 2.3. Using CARA, subjects found the chair efficiently, requiring only 10 s on average (Fig. 2.5C). Using vOICe, most subjects meandered through the arena, on occasion encountering the chair by accident. Only one subject was able to steer toward the chair (Fig. 2.12). None of the subjects were able to complete 20 trials within the 60 s time limit.

On the third task, the subject was asked to find a key that had fallen on the floor of the arena. To complete the task, the subject's head had to point toward the key at <1 m distance. Under those conditions, one can readily reach out and grasp the object. Using CARA, subjects found the key efficiently (Fig. 2.5D). Using vOICe, none of the subjects were able to locate the key (Fig. 2.5D), although two of them encountered it once by accident (Fig. 2.12).

These experiments illustrate the use of a standardized testing environment. Naive subjects performed well on these real-world tasks using CARA, but not using vOICe. It should be said that interpreting the vOICe sounds is very non-intuitive. Our subjects received the basic instructions offered on the vOICe web site and the Exercise mode of vOICe (*Seeing with Sound* n.d.), followed by a short period of practice. Extensive training with vOICe confers blind subjects with some ability to distinguish high contrast shapes on a clean background (Auvray, Hanneton, and O'Regan, 2007; Striem-Amit, Guendelman, and Amedi, 2012). Conceivably, an experienced vOICe user might perform better on the tests described here. Other investigators can attempt to demonstrate this using our published code and specifications (Y. Liu and Meister, 2018).

## 2.3 Discussion

Some components of what we implemented can be found in prior work (Botezatu et al., 2017; Ribeiro et al., 2012; Wang et al., 2017). Generally, assistive devices have been designed to perform one well-circumscribed function, such as obstacle avoidance or route finding (Loomis et al., 2012; Roentgen et al., 2008). Our main contribution here is to show that augmented reality with object voices offers a natural and effortless human interface on which one can build many functionalities that collectively come to resemble seeing. Our developments so far have focused on indoor applications to allow scene understanding and navigation. Blind people report that outdoor navigation is supported by many services (access vans, GPS, mobile phones with navigation apps), but these all fall away when one enters a building (Karimi, 2015). The present cognitive prosthesis can already function in this underserved domain, for example as a guide in a large public building, hotel, or mall. The virtual guide can be programmed to offer navigation options according to the known building geometry. Thanks to the intuitive interface, naïve visitors could pick up a device at the building entrance and begin using it in minutes. In this context, recall that our subjects were chosen without prescreening, including cases of early and late blindness and various hearing deficits (Fig. 1D): they represent a small but realistic sample of the expected blind user population. The functionality of this prosthesis can be enhanced far beyond replacing vision, by including information that is not visible. As a full service computer with online access, the HoloLens can be programmed to annotate the scene and offer ready access to other forms of knowledge. Down the line, one can envision a device that is attractive to both blind and sighted users, with somewhat different feature sets, which may help integrate the blind further into the community. By this point, we expect that the reader already has proposals in mind for enhancing the cognitive prosthesis. A hardware/software platform is now available to rapidly implement those ideas and test them with human subjects. We hope that this will inspire developments to enhance perception for both blind and sighted people, using augmented auditory reality to communicate things that we cannot see. "Seeing is knowing what is where by looking" (Marr, 1982). The prosthesis described here conveys "what" by the names of objects and "where" by the location from where each object calls. "Looking" occurs when the user actively requests these calls. The principal reason sighted people rely on vision much more than audition is that almost all objects in the world emit useful light signals almost all the time, whereas useful sound signals from our surroundings are few and sporadic. Our prosthesis can change this calculus fundamentally, such that all the relevant objects emit useful sounds. It remains to be seen whether prolonged use of such a device will fundamentally alter our perception of hearing to where it feels more like seeing.

## 2.4 Supplementary Materials

### Materials and Methods

#### General Implementation of CARA

The hardware platform for the cognitive assistant is the Microsoft HoloLens Development Edition, without any modifications. This is a self-contained wearable augmented reality (AR) device that can map and store the 3D mesh of an indoor space, localize itself in real time, and provide spatialized audio and visual display (Hoffman et al., 2016). We built custom software in Unity 2017.1.0f3 (64-bit) with HoloToolkit-Unity-v1.5.5.0. The scripts are written in C with MonoDevelop provided by Unity. The experiments are programmed on a desktop computer running Windows 10 Education and then deployed to Microsoft HoloLens. The software is versatile enough to be easily deployed to other hardware platforms, such as AR enabled smart phones.

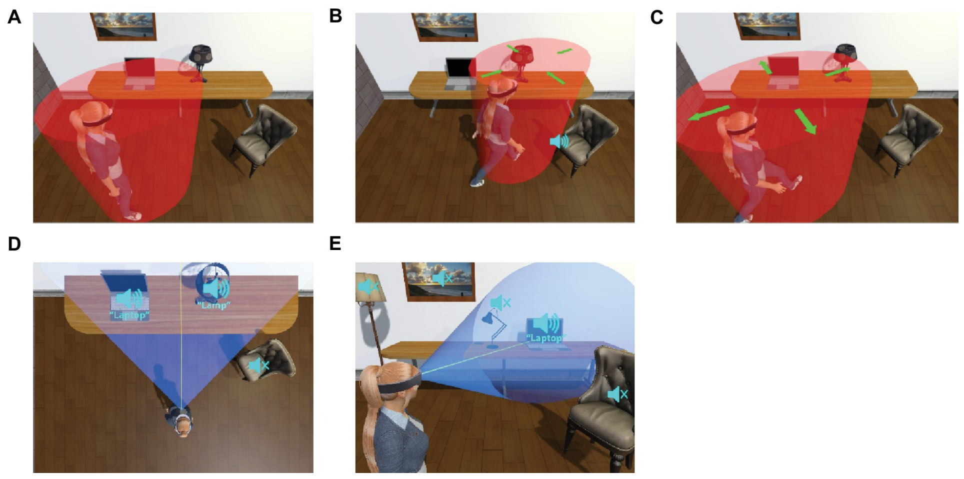


Figure 2.6: **Obstacle avoidance utility and active scene exploration modes.** (A) to (C) An object avoidance system is active in the background at all times. Whenever a real scanned surface or a virtual object enters a danger volume around the user (red in A), a spatialized warning sound is emitted from the point of contact (B). The danger volume expands automatically as the user moves (C), so as to deliver warnings in time. (D) to (E) Active exploration modes. In Scan mode (D), objects whose azimuthal angles fall in a certain range (e.g. between -60 and +60 deg) call themselves out from left to right. In Spotlight mode (E), only objects within a narrow cone are activated, and the object closest to the forward-facing vector calls out.

### User Interface

Before an experiment, the relevant building areas are scanned by the experimenter wearing the HoloLens, so the system has a 3D model of the space ahead of time. For each object in the scene, the system creates a voice that appears to emanate from the object's location, with a pitch that increases inversely with object distance. Natural spatialized sound is computed based on a generic head-related transfer function (Wenzel et al., 1993); nothing about the software was customized to individual users. Object names and guide commands are translated into English using the text-to-speech engine from HoloToolkit. The user provides input by moving the head to point at objects, pressing a wireless clicker, or using hand gesture commands or English voice commands.

In addition to instructions shown in the main body of the article, non-spatialized instructions are available at the user's request by voice commands. The user can use two voice commands (e.g. “direction", “distance") to get the direction of the current object of interest or its distance. Depending on the mode, the target object

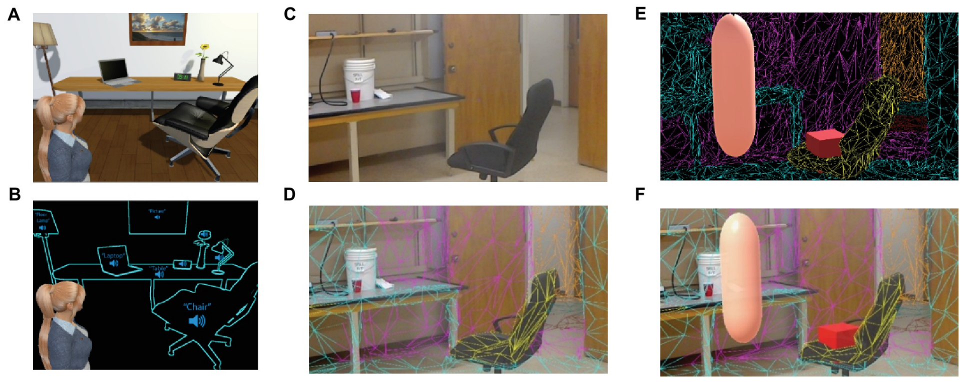


Figure 2.7: **Process of scene sonification.** The acquisition system should parse the scene (A) into objects and assign each object a name and a voice (B). In our study, this was accomplished by a combination of the HoloLens and the experimenter. The HoloLens scans the physical space (C) and generates a 3D mesh of all surfaces (D). In this digitized space (E) the experimenter can perform manipulations such as placing and labeling virtual objects, computing paths for navigation, and animating virtual guides (F). Because of the correspondence established in D, these virtual labels are tied to the physical objects in real space.

can be the object label of user's choice (Target mode) or the virtual guide. "Turn-by-turn" instructions can be activated by voice commands (e.g. "instruction"). The instruction generally consists of two parts, the distance the user has to travel until reaching the current target waypoint, and the turn needed to orient to the next waypoint (Fig. 2.10E).

### Experimental Design

All results in Figures 1–4 were gathered using a frozen experimental protocol, finalized before recruitment of the subjects. The tasks were fully automated, with dynamic instructions from the HoloLens, so that no experimenter involvement was needed during the task. Furthermore, we report performance of all subjects on all trials gathered this way. Some incidental observations and anecdotes from subject interviews are provided in Supplementary Observations. All procedures involving human subjects were reviewed and approved by the Institutional Review Board at Caltech. All subjects gave their informed consent to the experiments, and where applicable, to publication of videos that accompany this article.

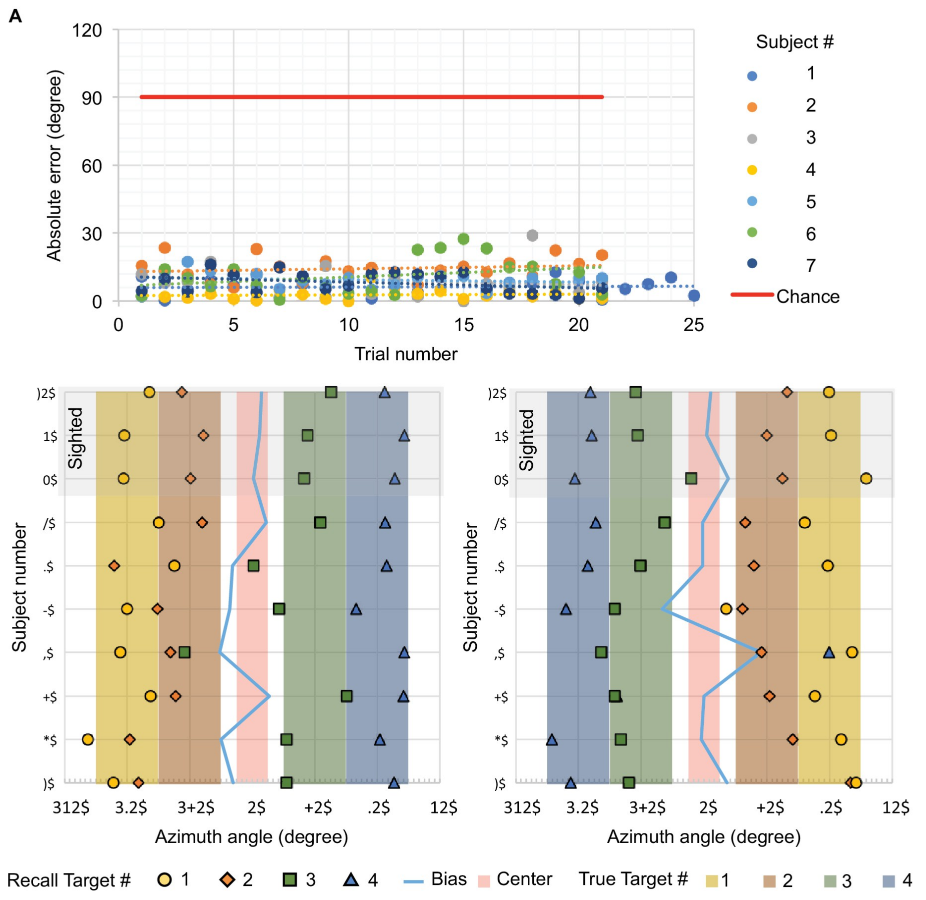


Figure 2.8: **Mental imagery task supplementary data.** Spatial memory data (Fig. 2.2) from blocks 1 (left) and 2 (right) by subject. Shaded areas indicate the true azimuthal extent of each object. Markers indicate recalled location. Most recalled locations overlap with the true extent of the object. Subjects 8–10 were normally sighted and performed the exploration phase using vision.

## Measurement

Timestamps are generated by the internal clock of the HoloLens. The six parameters of the subject's head location and orientation are recorded at 5 Hz from the onset to the completion of each trial in each task. All performance measures are derived from these time series. Localization errors of the HoloLens amount to <4 cm (Y. Liu, Dong, et al., 2018), which is insignificant compared to the distance measures reported in our study, and smaller than the line width in the graphs of trajectories in Fig. 2.3 and 2.4.

**Task Design**

Task 1, object localization (Fig. 2.1): In each trial, a single target is placed 1 m from the subject at a random azimuth angle drawn from a uniform distribution between 0 and 360 deg. To localize the target, the subject presses the clicker to hear a spatialized call from the target. After aiming the face at the object the subject confirms via a voice command (“Target confirmed"). When the location is successfully registered, the device plays a feedback message confirming the voice command and providing the aiming error. The subject was given 10–15 practice trials to learn the interaction with CARA, followed by 21 experimental trials. To estimate the upper limit on performance in this task, two sighted subjects performed the task with eyes open: this produced a standard deviation across trials of 0.31 and 0.36 degrees, and a bias of 0.02 and 0.06 degrees. That includes instrumentation errors as well as uncertainties in the subject’s head movement. Note that these error sources are insignificant compared to the accuracy and bias reported in Fig. 2.1 and 2.2.

Task 2, spatial memory (Fig. 2.2): This task consists of an exploration phase in which the subject scans the scene, followed by a recall phase with queries about the scene. Five objects are placed two meters from the subject at azimuth angles of 60 deg, 30 deg, 0 deg, 30 deg, 60 deg from the subject’s initial orientation. Throughout the experiment, a range between 7.5 deg and 7.5 deg in azimuth angle is marked by “sonar beeps" to provide the subject a reference orientation. During the 60 s exploration phase, the subject uses Spotlight mode: This projects a virtual spotlight cone of 30° aperture around the direction the subject is facing and activates object voices inside this spotlight. Typically subjects scan the virtual scene repeatedly, while listening to the voices. In the recall phase, Spotlight mode is turned off, and the subject performs four recall trials. For each recall trial, the subject presses the clicker, then a voice instruction specifies which object to turn to, the subject faces in the recalled direction, and confirms with a voice command (“Target confirmed"). The entire task was repeated in two blocks that differed in the arrangement of the objects. The object sequence from left to right was “piano," “table," “chair," “lamp," “trash bin" (block 1), and “trash bin," “piano," “table," “chair," “lamp" (block 2). The center object is never selected as a recall target because 0 deg is marked by sonar beeps and thus can be aimed at trivially.

Task 3, direct navigation (Fig. 2.3): In each trial, a single chair is placed at 2 m from the center of the arena at an azimuth angle randomly drawn from four possible choices: 0°, 90°, 180°, 270°. To start a trial, the subject must be in a starting zone of 1 m diameter in the center. During navigation, the subject can repeatedly press the Clicker to receive a spatialized call from the target. The trial completes when the subject arrives within 0.5 m of the center of the target. Then the system guides the subject back to the starting zone using spatialized calls emanating from the center of the arena, and the next trial begins. Subjects performed 19–21 trials. All blind subjects moved freely without a cane or guide dog during this task.

To measure performance on a comparable search without CARA, each subject performed a single trial with audio feedback turned off. A real chair is placed at one of the locations previously used for virtual chairs. The subject wears the HoloLens for tracking and uses a cane or other walking aid as desired. The trial completes when the subject touches the target chair with a hand. All blind subjects used a cane during this silent trial.

Task 4, long-range guided navigation (Fig. 2.4): The experimenter defined a guide path of 36 m length from the first-floor lobby to the second-floor office by placing nine waypoints in the pre-scanned environment. In each trial, the subject begins in a starting zone within 1.2 m of the first waypoint, and presses the Clicker to start. A virtual guide then follows the trajectory and guides the subject from the start to the destination. The guide calls out “follow me" with spatialized sound every 2 s, and it only proceeds along the path when the subject is less than 1 m away. Just before waypoints 2–8, a voice instruction is played to inform the subject about the direction of turn as well as approaching stairs. The trial completes when the subject arrives within 1.2 meters of the target. Voice feedback (“You have arrived") is played to inform the subject about arrival. In this task, all blind subjects used a cane.

Free walking: To measure the free walking speed, we asked subjects to walk for 20 m in a straight line in an unobstructed hallway using their preferred walking aid. Subjects 1 and 2 used a guide dog, the others a cane.

### Data Analysis and Visualization

MatLab 2017b (Mathworks) and Excel (Microsoft) were used for data analysis and visualization. Unity 5.6.1f1 was used to generate 3D cartoons of experiments and to visualize 3D trajectories. Photoshop CC 2017 was used for overlaying trajectories on floor plans.

Aiming: In Tasks 1 and 2, aiming errors are defined as the difference between the target azimuth angle and the subject’s front-facing azimuth angle. In Task 2, to correct for the delay of voice command registration, errors are measured at 1 s before the end of each trial.

Trajectory smoothing: The HoloLens tracks its wearer's head movement, which includes lateral movements perpendicular to the direction of walking. To estimate the center of mass trajectory of the subject, we applied a moving average with 2 s sliding window to the original trajectory.

Length of trajectory and deviation index: In the directed navigation task and the long-range guided navigation task, we computed the excess distance traveled by the subject relative to an optimal trajectory or the guide path. The deviation index, DI, is defined as

$$DI = \frac{Lexp - Lref}{Lref} \tag{2.1}$$

where L$exp$ is the length of the trajectory measured by experiment, and L$ref$ is the length of the reference trajectory. A value near 0 indicates that the subject followed the reference trajectory well.

In the direct navigation task, we divided each trial into an orientation phase where the subject turns the body to face the target, and a navigation phase where the subject approaches the target. We calculated head orientation and 2D distance to target in each frame, and marked the onset of the navigation phase when the subject's distance to target changed by 0.3 m. Note that with this criterion, the navigation phase includes the occasional trajectory where the subject starts to walk in the wrong direction. In this task, L$ref$ is defined as the length of the straight line from the subject's position at the onset of the navigation phase to the nearest point of the target trigger zone.

In the long-range guided navigation task, L$ref$ is the length of the guide trajectory. Due to variability in placing waypoints and tracking, the length of guide trajectories varied slightly across subjects (L$ref$=36.4 ± 0.7 m, mean ±s.d.). Negative DI values are possible in this task if the subject cuts corners of the guide trajectory.

Speed: Speed is calculated frame-by-frame using the displacements in the filtered trajectories. For the long-range guided navigation task, which includes vertical movements through space, the speed of translation is computed in three dimensions, whereas for the other tasks that occur on a horizontal plane, we did not include the vertical dimension. For all tasks, we estimated walking speed by the 90th percentile of the speed distribution, which robustly rejects the phases where the subject chooses an orientation. The normalized speed is obtained by dividing this value by the free walking speed.

**Supplementary Observations**

Here, we report incidental observations during experiments with CARA that were not planned in the frozen protocol, and comments gathered from blind subjects in the course of the experiments.

Subject 1: During navigation with the virtual guide, says, “Seems to me the ‘follow me’ sound means keep going straight.” Thinks addition of GPS services could make the system useful outdoors as well. Suggests experimenting with bone conduction headphones. Offers us 1 hr on his radio show.

Subject 2: During direct navigation, says, “Pitch change [with distance] was informative.” During navigation with the virtual guide says, “’Follow me’ was too much information.” Prefers to follow the explicit turn instructions. She could then transmit those instructions to her guide dog.

Subject 3: In addition to object voices, he likes instructions of the type “keep going forward for xx meters." During a previous visit using a similar system, he commented on possible adoption by the blind community: “I could see people spending in 4 figures for [something] light and reliable, and use it all the time.” Also supports the concept of borrowing a device when visiting a public building or mall. Devices in the form of glasses would be better, preferably light and thin. “Use the computing power of my phone, then I don’t have to carry anything else.” Likes the external speakers because they don’t interfere with outside sound. Finds it easy to localize the virtual sound sources.

Subject 4: After navigation with the virtual guide says, “That was fun. When can I get one?” Primarily used the “follow me" voice and the cane to correct for small errors. Reports that the turn instructions could be timed earlier (this is evident also in Video 1). On a previous visit using a similar system: “I’m very excited about all of this, and I would definitely like to be kept in the loop.” Also suggests the system could be used in gaming for the blind.

Subject 5: During navigation with the virtual guide, realized she made a wrong turn (see Fig. 4C) but the voice made her aware and allowed her to correct. Reports that the timing of turn instructions is a little off.

Subject 6: After all tasks says, “That was pretty cool,” and, “The technology is there.”

Subject 7: On the second trial with the virtual guide, reports that she paid more attention to the “follow me" sound (she strayed temporarily on the first trial, Fig. 4B). Wonders whether the object voices will be strong enough in a loud environment.

**Benchmarking Platform Using Virtual Reality**

The benchmarking platform runs on the HTC Vive VR headset and a Windows 10 desktop computer. A TPCast wireless adapter (CE-01H, https://www.tpcastvr.com/) replaces the standard headset cable so the subject can move freely within the 4 m x 4 m square arena. An Xbox One S controller is connected wirelessly to the host computer for the subject to start trials, confirm aiming, and control the modes of CARA. Audio is delivered by a pair of wireless headphones (SONY WH-1000XM2). All code and data to replicate this environment and the reported tests is publically available at `https://github.com/meisterlabcaltech/CARA_Public`.

All three benchmarking tasks are set in a 10 m x 10 m virtual environment that simulates a living room with 16 objects labeled with sound tags. The rendered scenes are close to photo-realistic, and computer vision algorithms trained on real scenes will correctly identify objects in the virtual scene as well (Fig. 2.10F). The first benchmark task (Fig. 2.5B) resembles Task 1 (see section on Task Design above), except that the aiming target is chosen at random from the 16 objects in the studio. CARA users have access to Spotlight and Target mode. The second benchmark task (Fig. 2.5C) resembles Task 3 with a 60 s time limit on each trial. CARA users only have Target mode available for this task. To facilitate detection with vOICe, the chair was made white and left unoccluded by any other objects. The third benchmark task (Fig. 2.5D) replaces the chair in the second benchmark task with a key on the floor. Within CARA, Target mode and Spotlight mode can be used in this task. Two conditions have to be met to finish a trial: (1) the head of the subject is within 1 m of the key and (2) the subject faces within 30 degrees of the key. To accomplish this, the subject must bend down or kneel facing the key. At this point, a simple reach with the hand would allow grasping a real object. A trial fails if not finished within 60 s. To avoid excess frustration among the subjects, a task is terminated after five failed trials.

**Voice Control on CARA**

In addition to the Clicker, subjects can also use natural language (e.g. English) as input to the system. Two subsystems of voice input are implemented: 1) keyword recognition (PhraseRecognitionSystem) monitors in the background what the user says, detects phrases that match the registered keywords, and activates corresponding functions on detection of keyword matches, and 2) dictation (DictationRecognizer) records what the user says and converts it into text. The first component enables subjects to confirm their aiming in the object localization task and mental imagery task with the voice command "target confirmed." It also enables the experimenter to control the experiment at runtime.

Keywords and their functions are defined through adding keywords to the keyword manager script provided by HoloToolkit and editing their responses. The KeywordRecognizer component starts at the beginning of each instance of the application and runs in the background throughout the instance of the application except for the time period in which dictation is in use.

To allow users to create object labels, the DictationRecognizer provided by HoloToolkit is used to convert natural language spoken by the user to English text. Due to the mutual exclusivity, KeywordRecognizer is shut down before DictationRecognizer is activated, and restarted after the dictation is finished.

**Automated Wayfinding**

In addition to hand-crafting paths, we implemented automated wayfinding by taking advantage of Unity's runtime NavMesh "baking." which calculates navigable areas given a 3D model of the space. At runtime, we import and update the 3D mesh of the scanned physical space and use it to bake the 3D mesh. When the user requests guided navigation, a path from the user's current location to the destination of choice is calculated. If the calculated path is valid, the virtual guide guides the user to the destination using the computer-generated path.

**Cost of the CARA System**

The hardware platform used in the research – Microsoft HoloLens Development Edition – currently costs \$ 3000. Several comparable AR goggles are in development, and one expects their price to drop in the near future. In addition, smart phones are increasingly designed with AR capabilities, although they do not yet match the HoloLens in the ability to scan the surrounding space and localize within it.

### Battery and Weight

The current HoloLens weighs 579 g. Like all electronic devices, this will be further miniaturized in the future. The current battery supports our system functions for 2–5 hr, sufficient for the indoor excursions we envision in public buildings, led by the “virtual guide". A portable battery pack can extend use to longer uninterrupted sessions.

### Tracking Robustness

While for most indoor scenarios that we have tested, the tracking of HoloLens was reliable and precise, but we have encountered occasional loss of tracking or localization errors. This occurs particularly when the environment lacks visual features, such as a narrow space with white walls.

### Extensions

Because this cognitive assistant is largely defined by software, its functionalities are very flexible. For example, the diverse recommendations from subjects noted above (Supplementary Observations) can be implemented in short order. In addition, one can envision hardware extensions by adding peripherals to the computer. For example, a haptic belt or vest could be used to convey collision alarms (Adebiyi et al., 2017), thus leaving the auditory channel open for the highly informative messages.

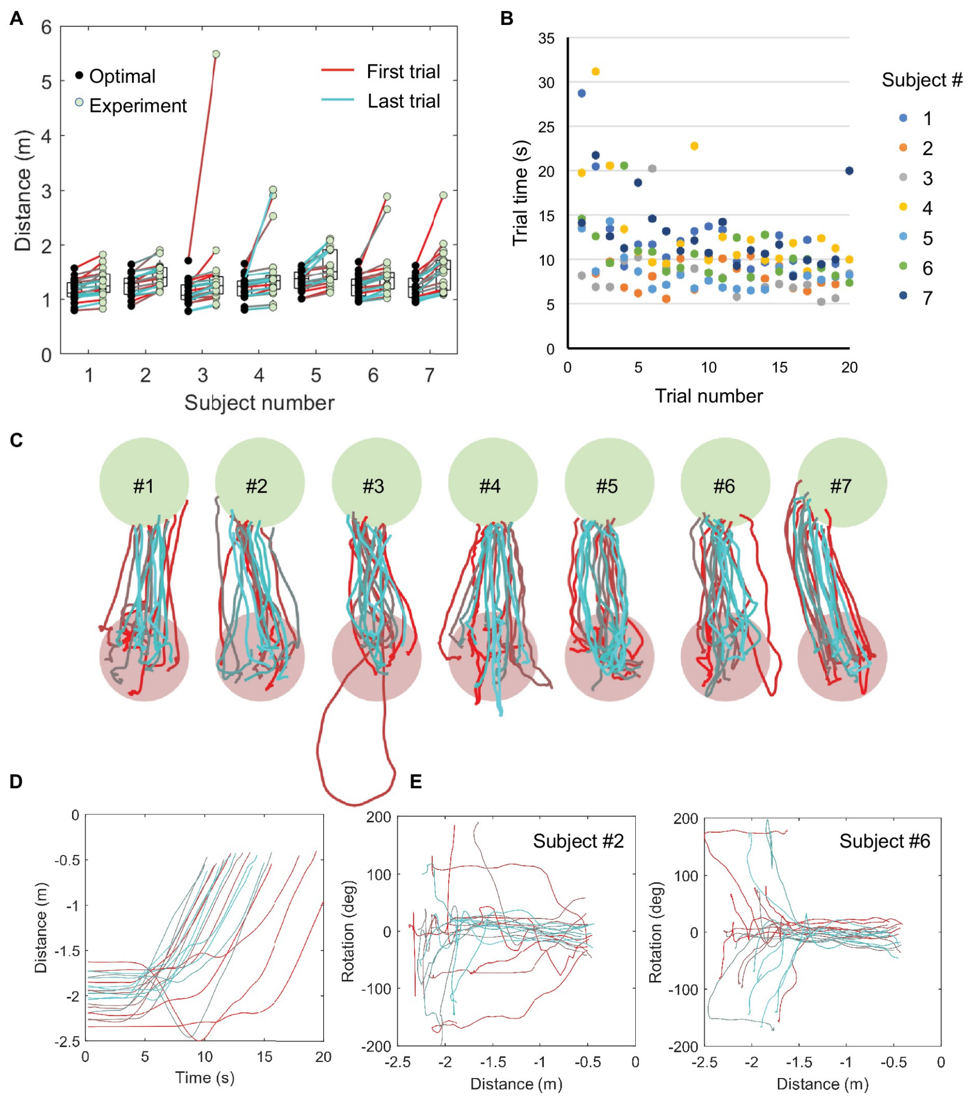


Figure 2.9: **Direct navigation task extended data.** Trial distance (A) and trial duration (B) for the first 20 trials of all subjects. A modest effect of practice on task duration can be observed across all subjects (B). (C) Low-pass filtered, aligned trajectories of all subjects. In most trials, subjects reach the target with little deviation. (D) Dynamics of navigation, showing the distance to target as a function of trial time for one subject. (E) Head orientation vs distance to target for two subjects. Note Subject 6 begins by orienting without walking, then walks to the target. Subject 2 orients and walks at the same time, especially during early trials.

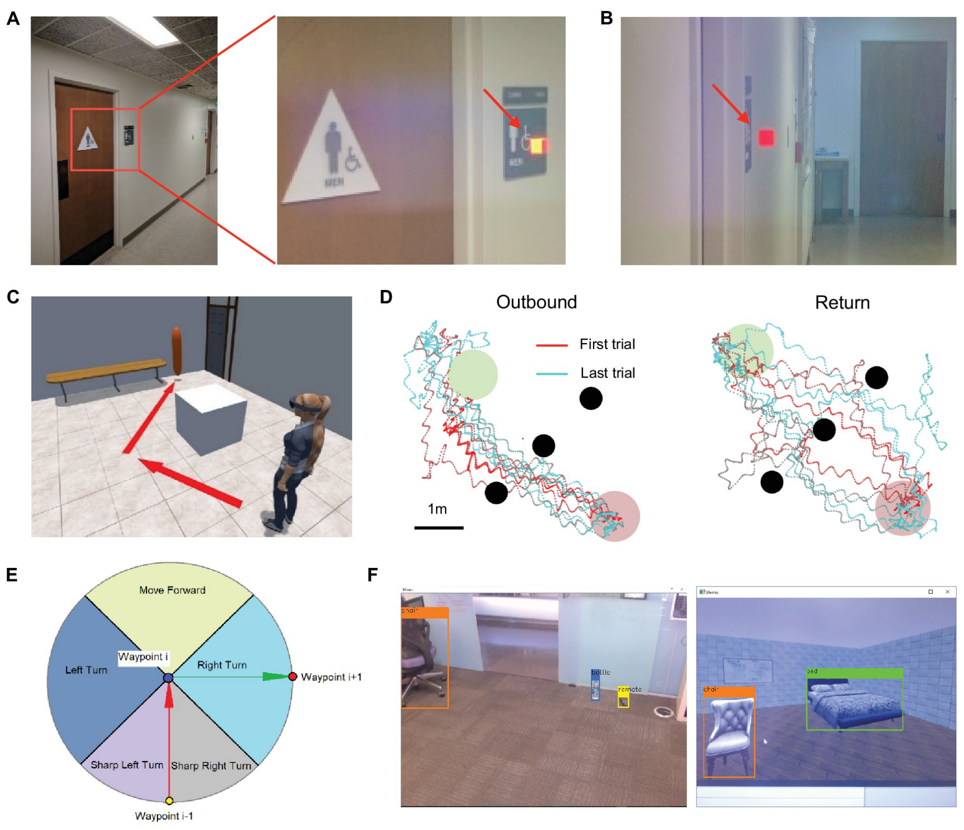
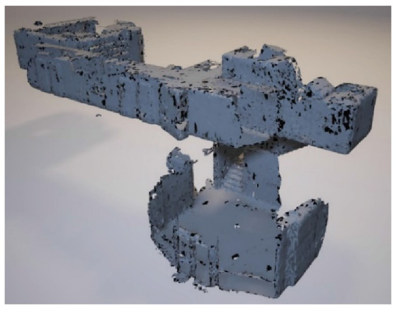


Figure 2.10: **Additional experimental functions.** (A) to (B) Automated sign recognition using computer vision. Using Vuforia software (*Vuforia* n.d.), the HoloLens recognizes a men's room sign (A) (image is viewed through HoloLens), and installs a virtual object (cube, arrow) next to the sign. (B) This object persists in the space even when the sign is no longer visible. (C) Automated wayfinding. The HoloLens generates a path to the target (door) that avoids the obstacle (white box). Then a virtual guide (orange balloon) can lead the user along the path. See Videos 2–3. (D) Navigation in the presence of obstacles. The subject navigates from the starting zone (red circle) to an object in the target zone (green circle) using calls emitted by the object. Three vertical columns block the path (black circles), and the subject must weave between them using the obstacle warning system. Raw trajectories (no filtering) of a blind subject (5) are shown during outbound (left) and return trips (right), illustrating effective avoidance of the columns. This experiment was performed with a version of the apparatus built around the HTC Vive headset. (E) Orienting functions of the virtual guide. In addition to spatialized voice calls, the virtual guide may also offer turning commands toward the next waypoint. In the illustrated example, the instruction is, "In x meters, turn right." (F) Real-time object detection using YOLO (Redmon and Farhadi, 2018). Left: A real scene. Note even small objects on a textured background are identified efficiently based on a single video frame. Right: A virtual scene from the benchmarking environment, rendered by Unity software.

A


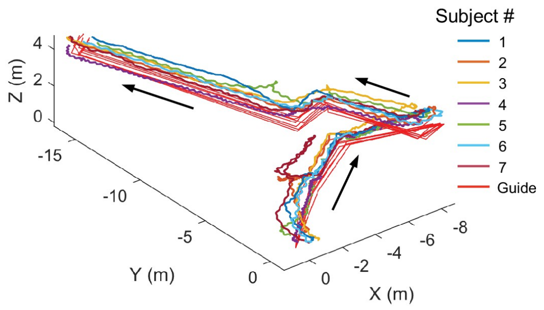
B


Figure 2.11: **Guided navigation trajectories.** (A) 3D model of the experimental space as scanned by the HoloLens. (B) Subject and guide trajectories from the long-range guided navigation task. Note small differences between guide trajectories across experimental days, owing to variations in detailed waypoint placement.

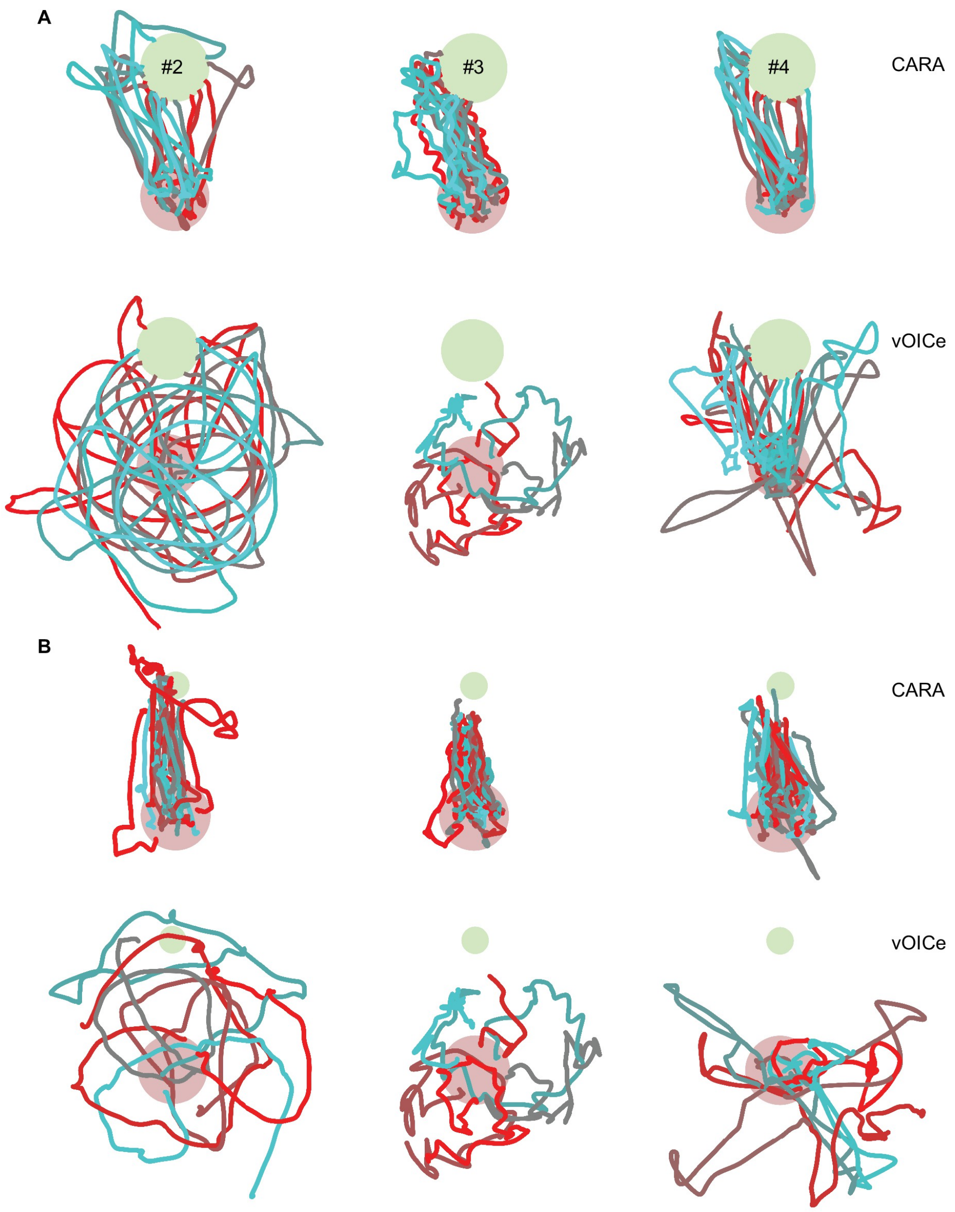


Figure 2.12: **Benchmark tests in a virtual environment.** Trajectories of three additional subjects. (A) Navigation to a randomly placed chair, using either CARA or vOICe, displayed as in Fig. 2.5C. Subject 4 exhibited some directed navigation using vOICe. (B) Finding a dropped key, as in Fig. 2.5D

.

*Chapter 3*

# SYNTHETIC EXAMPLES IMPROVE GENERALIZATION FOR RARE CLASSES

Beery, Sara et al. (Mar. 2020). “Synthetic Examples Improve Generalization for Rare Classes”. In: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*. URL: `http://openaccess.thecvf.com/content_WACV_2020/papers/Beery_Synthetic_Examples_Improve_Generalization_for_Rare_Classes_WACV_2020_paper.pdf`.

## ABSTRACT

The ability to detect and classify rare occurrences in images has important applications – for example, counting rare and endangered species when studying biodiversity, or detecting infrequent traffic scenarios that pose a danger to self-driving cars. Few-shot learning is an open problem: current computer vision systems struggle to categorize objects they have seen only rarely during training, and collecting a sufficient number of training examples of rare events is often challenging and expensive, and sometimes outright impossible. We explore in depth an approach to this problem: complementing the few available training images with ad-hoc simulated data.

Our testbed is animal species classification, which has a real-world long-tailed distribution. We present two natural world simulator and analyze the effect of different axes of variation in simulation, such as pose, lighting, model, and simulation method, and we prescribe best practices for efficiently incorporating simulated data for real-world performance gain. Our experiments reveal that synthetic data can considerably reduce error rates for classes that are rare, that as the amount of simulated data is increased, accuracy on the target class improves, and that high variation of simulated data provides maximum performance gain.

## 3.1 Introduction

In recent years, computer vision researchers have made substantial progress towards automated visual recognition across a wide variety of visual domains (Russakovsky et al., 2015; Esteva et al., 2017; Poplin et al., 2018; Van Horn, Mac Aodha, et al., 2017; Norouzzadeh et al., 2017; van Horn et al., 2017; Beery, Van Horn, and Perona, 2018). However, applications are hampered by the fact that in the real world, the distribution of visual classes is long-tailed, and state-of-the-art recognition algorithms struggle to learn classes with limited data (Van Horn and Perona, 2017). In some cases (such as recognition of rare endangered species) classifying rare occurrences correctly is crucial. Simulated data, which is plentiful and comes with annotation “for free," has been shown to be useful for various computer vision tasks (Varol, Romero, X. Martin, Mahmood, Michael J. Black, et al., 2017a; Pepik et al., 2015; Hinterstoisser et al., 2019; Rajpura, Bojinov, and Hegde, 2017; Goodfellow et al., 2014; Shrivastava et al., 2017; Stephan R. Richter et al., 2016a; Peng et al., 2018; Hattori et al., 2015; Han et al., 2017; Ji et al., 2019). However, an exploration of this approach in a long-tailed setting is still missing (see Section 3.2).

As a testbed, we focus on the effect of simulated data augmentation on the real-world application of recognizing animal species in camera trap images. Camera traps are heat- or motion-activated cameras placed in the wild to monitor animal populations and behavior. The processing of camera trap images is currently limited by human review capacity; consequently, automated detection and classification of animals is a necessity for scalable biodiversity assessment. A single sighting of a rare species is of immense importance. However, training data of rare species is, by definition, scarce. This makes this domain ideal for studying methods for training detection and classification algorithms with few training examples. We utilize a technique from (Beery, Van Horn, and Perona, 2018) which tests performance at camera locations both seen (cis) and unseen (trans) during training in order to explicitly study generalization (see Section 3.3 for a more detailed explanation).

We introduce two novel natural world simulators based on popular 3D game development engines for generalizable, realistic, and efficient synthetic data generation. We investigate the use of simulated data as augmentation during training, and how to best combine real data for common classes with simulated data for rare classes to achieve optimal performance across the class set at test time. We consider four different data simulation methods (see Fig.3.1) and compare the effects of each on classification performance. Finally, we analyze the effect of both increasing the number of simulated images and controlling for axes of variation to provide best practices for leveraging simulated data for real-world performance gain on rare classes.

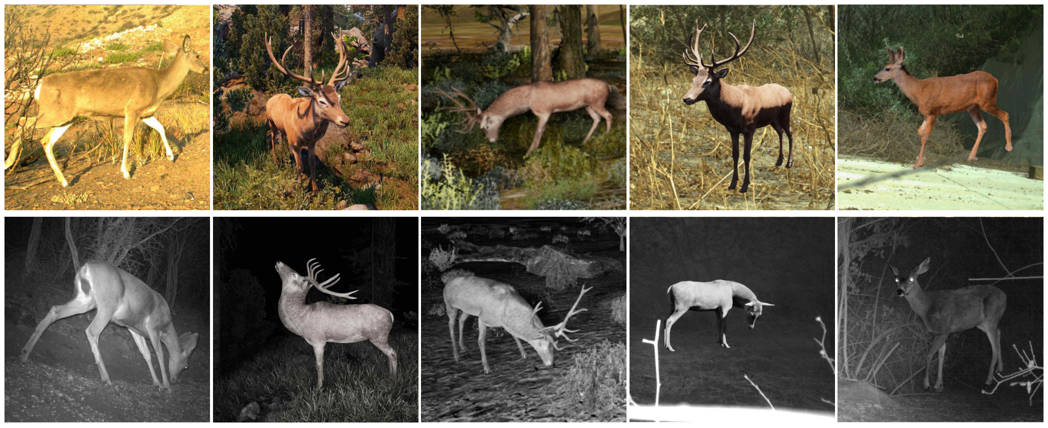


Figure 3.1: **Day (top) and night (bottom) examples for each simulation method.** We compare four different simulation methods and compare the effects of each on classification performance.

## 3.2 Related work

### Visual Categorization Datasets

Large and well-annotated public datasets allow scientists to train, analyze, and compare the performance of different methods, and have provided large performance improvements over traditional vision approaches (Szegedy, Vanhoucke, et al., 2016; J. Huang et al., 2017; K. He, X. Zhang, et al., 2016). The most popular datasets used for this purpose are ImageNet, COCO, PascalVOC, and OpenImages, all of which are human-curated from images scraped from the web (J. Deng et al., 2009; Lin, Maire, et al., 2014; Everingham et al., 2010; Krasin et al., 2017). These datasets cover a wide set of classes across both the manufactured and natural world, and are usually designed to provide “enough" data per class to avoid the low-data regime. More recently, researchers have proposed datasets that focus specifically on long-tailed distributions (Van Horn, Mac Aodha, et al., 2017; Beery, Van Horn, and Perona, 2018; Kumar et al., 2012). The Caltech Camera Traps dataset (Beery, Van Horn, and Perona, 2018) introduced the challenge of learning from limited locations, and generalizing to new locations.

### Handling Imbalanced Datasets

Imbalanced datasets lead to bias in algorithm performance toward well-represented classes (Buda, Maki, and Mazurowski, 2018). Algorithmic solutions often use a non-uniform cost per misclassification via weighted loss (Elkan, 2001; H. He and Garcia, 2008; H. He, Bai, et al., 2008). One example, focal loss, was recently proposed to deal with the large foreground/background imbalance in detection (Lin, Goyal, et al., 2018).

Data solutions employ data augmentation, either by 1) over-sampling the minority classes, 2) under-sampling the majority classes, or 3) generating new examples for the minority classes. When using mini-batch gradient descent, oversampling the minority classes is similar to weighted loss. Under-sampling the majority classes is non-ideal, as this reduces information about common classes. Our paper falls into the third category: generating new training data for rare classes. Data augmentation via pre-processing, using affine and photometric transformations, is a well-established tool for improving generalization (Krizhevsky, Sutskever, and G. E. Hinton, 2012; Howard, 2013). Data generation and simulation have begun to be explored as data augmentation methods, see Section 3.2.

Algorithmic and data solutions for imbalanced data are complementary, algorithmic advances can be used in conjunction with augmented training data.

### Low-shot Learning

Low-shot learning attempts to learn categories from few examples (Li, Fergus, and Perona, 2006). Wang and Herbert (Wang and Hebert, 2016) do low-shot classification by regressing from small-dataset classifiers to large-dataset classifiers. Hariharan and Girshick (Hariharan and Girshick, 2017) look specifically at ImageNet, using classes that are unbalanced, some with large amounts of training data, and some with little training data. Metric learning learns a representation space where distance corresponds to similarity, and uses this as a basis for low-shot solutions (Cui, F. Zhou, et al., 2016). We consider the low-shot regime with regard to *real* data for our rare target class, but investigate the use of added synthetic data based on a human-generated articulated model of the unseen class during training instead of additional class-specific attribute labels at training and test time. This takes us outside of the traditional low-shot framework into the realm of domain transfer from simulated to real data.

## Data Augmentation via Style Transfer, Generation, and Simulation

Image generation via generative adversarial networks (GANs) and recurrent neural networks (RNNs), as well as style transfer and image-to-image translation have all been considered as sources for data augmentation (Bousmalis et al., 2017; Gregor et al., 2015; Im et al., 2016; Radford, Metz, and Chintala, 2015; Tran et al., 2017; Luan et al., 2017; J.-Y. Zhu et al., 2017). These techniques require large amounts of data to generate realistic images, making them non-ideal solutions for low-data regimes. Though conditional generation allows for class-specific output, the results can be difficult to interpret or control.

Graphics engines such an Unreal (*UNREAL Game Engine* n.d.) and Unity (*Unity Game Engine* n.d.) leverage the expertise of human artists and complex physics models to generate photorealistic simulated images, which can be used for data augmentation. Because ground truth is known at generation, simulated data has proved particularly useful for tasks requiring detailed and expensive annotation, such as keypoints, semantic segmentations, or depth information (Varol, Romero, X. Martin, Mahmood, Michael J. Black, et al., 2017a; Pepik et al., 2015; Hinterstoisser et al., 2019; Rajpura, Bojinov, and Hegde, 2017; Goodfellow et al., 2014; Shrivastava et al., 2017; Stephan R. Richter et al., 2016a; Peng et al., 2018; Hattori et al., 2015). Varol et al. (Varol, Romero, X. Martin, Mahmood, Michael J. Black, et al., 2017a) use synthetically-generated humans placed on top of real image backgrounds as pretraining for human pose estimation, and suggest fine-tuning a synthetically-trained model on real data. (Shrivastava et al., 2017) use a combination of unlabeled real data and labeled simulated data of the same class to improve real-world performance on an eye-tracking task by using GANs (Goodfellow et al., 2014). This method requires a large number of unlabled examples from the target class. (Pepik et al., 2015; Hinterstoisser et al., 2019; Rajpura, Bojinov, and Hegde, 2017) find that simulated data improves detection performance, and the degree of realism and variability of simulation affects the amount of improvement. They consider only small sets of non-deformable man-made objects. Richter (Stephan R. Richter et al., 2016a) showed that a segmentation model for city scenes trained with a subset of their real dataset and a large synthetic set outperforms a model trained with the full real dataset. (Peng et al., 2018) proposes a dataset and benchmark for evaluating models for unsupervised domain transfer from synthetic to real data with all-simulated training data, as opposed to simulated data only for rare classes. While this literature is encouraging, a number of questions are left unexplored. The first is a careful analysis of when simulated data is useful and, in particular, if it is useful in generalizing to new scenarios. Second is whether simulated data can be useful in highly complex and relatively unpredictable scenes such as natural scenes, as opposed to indoors and urban scenes. Third is whether it is just the synthetic objects or also the synthetic environments that contribute to learning.

#### Simulated Datasets

Previous efforts on synthetic dataset generation focus on non-deformable man-made objects and indoor scenes (S. Song et al., 2017; Savva et al., 2017; Y. Wu et al., 2018; Hinterstoisser et al., 2019; Rajpura, Bojinov, and Hegde, 2017; Kolve et al., 2017), human poses/actions (Varol, Romero, X. Martin, Mahmood, Michael J. Black, et al., 2017a; Souza12 et al., 2017), or urban scenes (Ros et al., 2016; Gaidon et al., 2016; Stephan R. Richter et al., 2016a; Dosovitskiy et al., 2017; Han et al., 2017; Ji et al., 2019).

Bondi (Bondi et al., 2018) previously released the AirSim-w data simulator within the domain of wildlife conservation, focused on creating aerial infrared imagery. The resolution and quality of the assets is designed to replicate data from 100 meters in the air, but is not realistic close-up. We contribute the first image data generators specifically for the natural world with the ability to recreate natural environments and generate near-photorealistic images of animals within the scene, including real-world nuisance factors such as challenging pose, lighting, and occlusion.

## 3.3 Data and Simulation

#### Real Data

Our real-world training and test data comes from the Caltech Camera Traps (CCT) dataset (Beery, Van Horn, and Perona, 2018). CCT contains $243,187$ images from 140 camera trap locations covering 30 classes of animals, curated from data provided by the United States Geological Survey and the National Park Service. We follow the CCT-20 data split laid out in (Beery, Van Horn, and Perona, 2018), which was explicitly designed for in-depth generalization analysis. The split uses a subset of $57,868$ images from 20 camera locations covering 15 classes in CCT to simultaneously investigate performance on locations seen during training and generalization performance to new locations. Bounding-box annotations are provided for all images in CCT-20, whereas the rest of CCT has only class labels. In the CCT-20 data split, *cis-locations* are defined as locations seen during training and *trans-locations* as locations not seen during training (see Fig.3.3). Nine locations are used for trans-test data, one location for trans-validation data, and data from the remaining 10 locations is split between odd and even days, with odd days as cis-test data and even days as training and cis-validation data (a 95% of data from even days for training, 5% for testing).

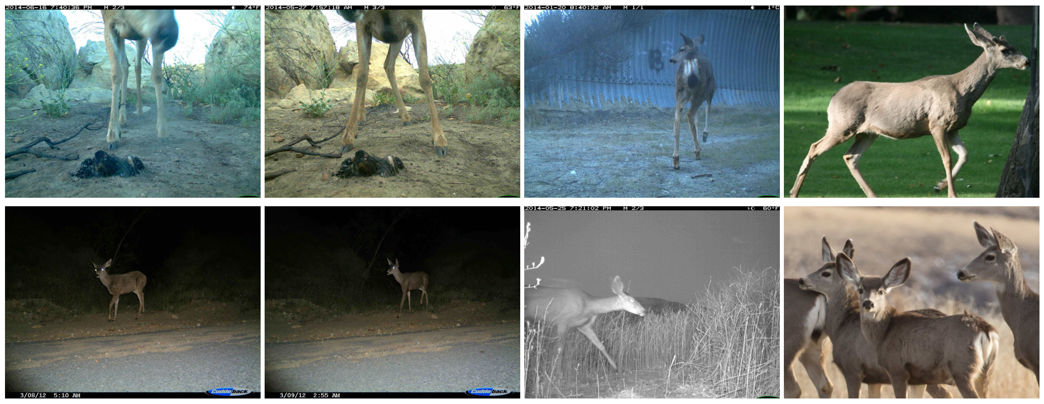
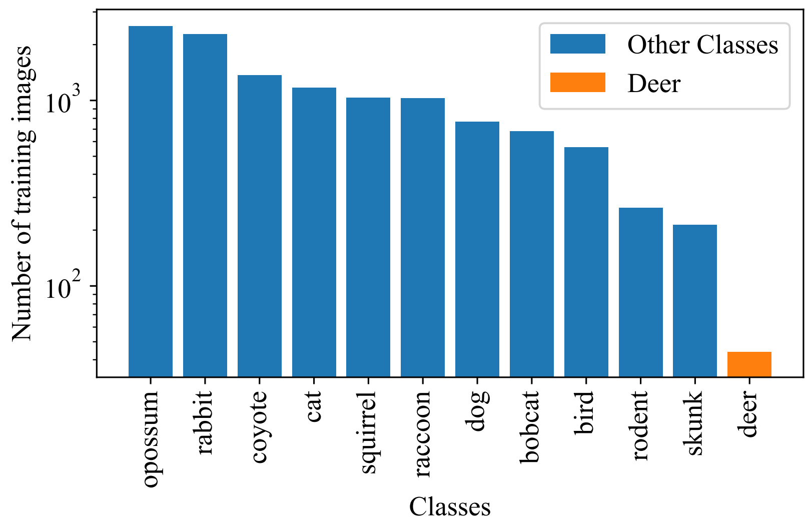


Figure 3.2: **Cis vs. Trans:** The cis-test data can be very similar to the training data: animals tend to behave similarly at a single location even across different days, so the images collected of each species are easy to memorize intra-location. The trans data has biases towards specific angles and lighting conditions that are different from those in the cis locations, and as such is very hard to learn from the training data. iNaturalist data represents a domain shift to human-curated images.

To study the effect of simulated data on rare species, we focus on deer, which are rare in CCT-20, with only 44 deer examples out of the 13, 553 images in the training set (see Fig.3.3). To focus on the performance of a single rare class, we remove the other two rare classes in CCT-20: badgers and foxes. We note that there are no deer images in the established CCT-20 trans sets. In reality, deer are far from uncommon: unlike a truly rare species, there exist sufficient images of deer in the CCT dataset outside of the CCT-20 locations to rigorously evaluate performance. To facilitate deeper investigation of generalization, we have collected bounding-box annotations for an additional 16K images from CCT across 65 new locations, which we add to the trans-validation and trans-test sets to cover a wider variety of locations and classes (including deer). We call this augmented trans set *trans*+ (see Fig.3.3) and will release the annotations at publication. To further analyze generalization, we also test on data containing deer from the iNaturalist 2017 dataset (Van Horn, Mac Aodha, et al., 2017), which represents a domain shift to human-captured and human-selected photographs. We consider *Odocoileus hemionus* (mule deer) and *Odocoileus* virginianus (white-tailed deer) images from iNaturalist, the two species



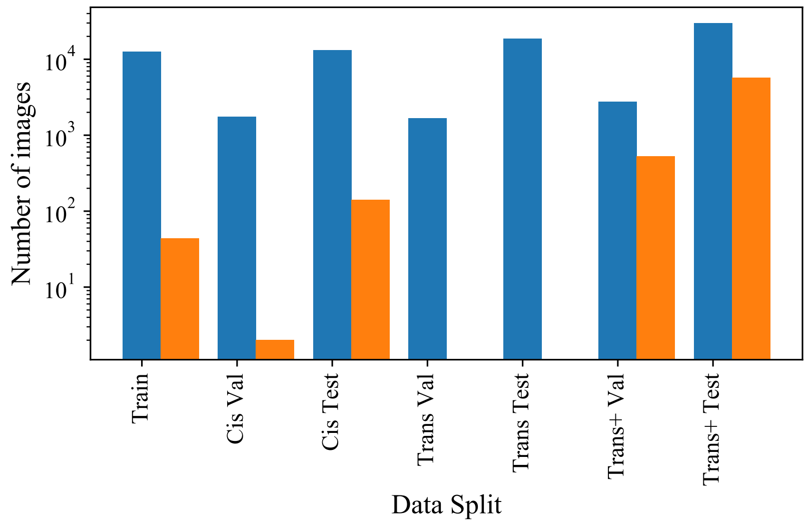


Figure 3.3: **(Top) Number of training examples for each class.** Deer are rare in the training locations from the CCT-20 data split. We focus on deer as a test species in order to investigate whether we can improve performance on a “rare" class. Since deer are not rare at other camera locations within the CCT dataset, we have enough test data to thoroughly evaluate the effect. **(Bottom) Number of examples for each data split, for deer and other classes.** In the CCT-20 data split, there were no trans examples of deer. We added annotations to the trans val and test sets for an additional 16K images across 65 new locations from CCT, including 6K examples of deer. We call these augmented sets *trans+*.

of deer seen in the CCT data. In Supplementary Material, we show results of an additional class, wolf.

## Synthetic Data

To assess generality, we leverage multiple collections of woodland and animal models to create two simulation environments, which we call TrapCam-Unity and TrapCam-AirSim. Both simulation environments and source code to generate images will be provided publicly, along with the data generated for this paper. To synthesize daytime images, we varied the orientation of the simulated sun in both azimuth and elevation. To create images taken at night, we used a spotlight attached to the simulated camera to simulate a white-light or IR flash and qualitatively match the low color saturation of the nighttime images. To simulate animals' eyeshine (a result of the reflection of camera flash from the back of the eye), we placed small reflective balls on top of the eyes of model animals.

**TrapCam-AirSim.** We create a modular natural environment within Microsoft AirSim (Shah et al., 2018) that can be randomly populated with flora and fauna. The distribution and types of trees, bushes, rocks, and logs can be varied and randomly seeded to create a diverse set of landscapes, from an open plain to a dense forest. We used various off-the-shelf components such as an animal pack from Epic Studios (*Epic Studios* n.d.) (Animals Vol 01: Forest Animals by GiM (*Forest Animals by GiM* n.d.)), background terrain also from Unreal Marketplace (*UNREAL Game Engine* n.d.), vegetation from SpeedTree (*SpeedTree* n.d.), and rocks/obstructions from Megascans (*Quixel Megascans Library* n.d.). The actual area of the environment is small, at 50 meters, but the modularity allows many possible scenes to be built.

**TrapCam-Unity.** Unity 3D game development engine is a popular game development tool that offers realistic graphics, real time performance, and abundant 3D assets. We take advantage of the “Book of The Dead" environment (*Unity Book of the Dead* n.d.), a near-photorealistic, open-source forest environment published by Unity to demonstrate its high definition rendering pipeline. This off-the-shelf environment is large and rich in details, and it has a diversity of subregions with significantly different statistics. We change the lighting and move throughout this large, static environment to collect data with various background scenes. We make use of 17 animated deer models from five off-the-shelf model sets, purchased from Unity Asset Store and originally developed for game development, including the GiM models used in TrapCam-AirSim. A single gaming PC (Core i7 5820K, 16GB RAM, GTX 1080Ti) generates over 300,000 full-HD images with pixel-level instance annotation per day and the throughput linearly scales to additional machines.

**Simulated animals on empty images.** Similar to the data generated in (Varol, Romero, X. Martin, Mahmood, Michael J. Black, et al., 2017a), we generate synthetic images of deer by rendering deer on top of real camera trap images containing no animals, which we call *Sim on Empty* (see Fig.3.1). We first generate animal foreground images by randomizing the location, orientation in azimuth, pose, and illumination of the deer, then paste the foreground images on top of the real empty images. A limitation is that the deer are not in realistic relationships or occlusion scenarios with the environment around them. We also note that the empty images used to construct this data come from both cis and trans locations, so Sim on Empty contains information about test-set backgrounds unavailable in the purely simulated sets. This choice is based on current camera trap literature, which first detects the presence of any animal, and then determines animal species (Norouzzadeh et al., 2017; Beery, Van Horn, and Perona, 2018). After the initial animal detection step, the empty images are known and can be utilized.

**Segmented animals on empty images.** We manually segment the 44 examples of deer from the training set and paste them at random on top of real empty camera trap images, which we call *Real on Empty* (see Fig.3.1). This allows us to analyze whether the generalization challenge is related to memorizing the training deer+background or memorizing the training deer regardless of background. Similar to the Sim on Empty set, these images do not have realistic foreground/background relationships, and the empty images come from both cis and trans locations.

## 3.4 Experiments

Beery, (Beery, Van Horn, and Perona, 2018) showed that detecting and localizing the presence of an “animal" (where all animals are grouped into a single class) both generalizes well to new locations and improves classification performance. We focus on classification of cropped ground-truth bounding boxes as opposed to training multi-class detectors in order to disambiguate classification and detection errors. We specifically investigate how added synthetic training data for rare classes effects model performance on both rare and common classes.

We find that the Inception-Resnet-V2 architecture (Szegedy, Ioffe, et al., 2017) works best for the cropped-box classification task, based on performance comparison across architectures (see Supplementary Material). Most classification systems are pretrained on Imagenet, which contains animal classes. To ensure that our “rare" class is truly something the model is unfamiliar with, as opposed to something seen

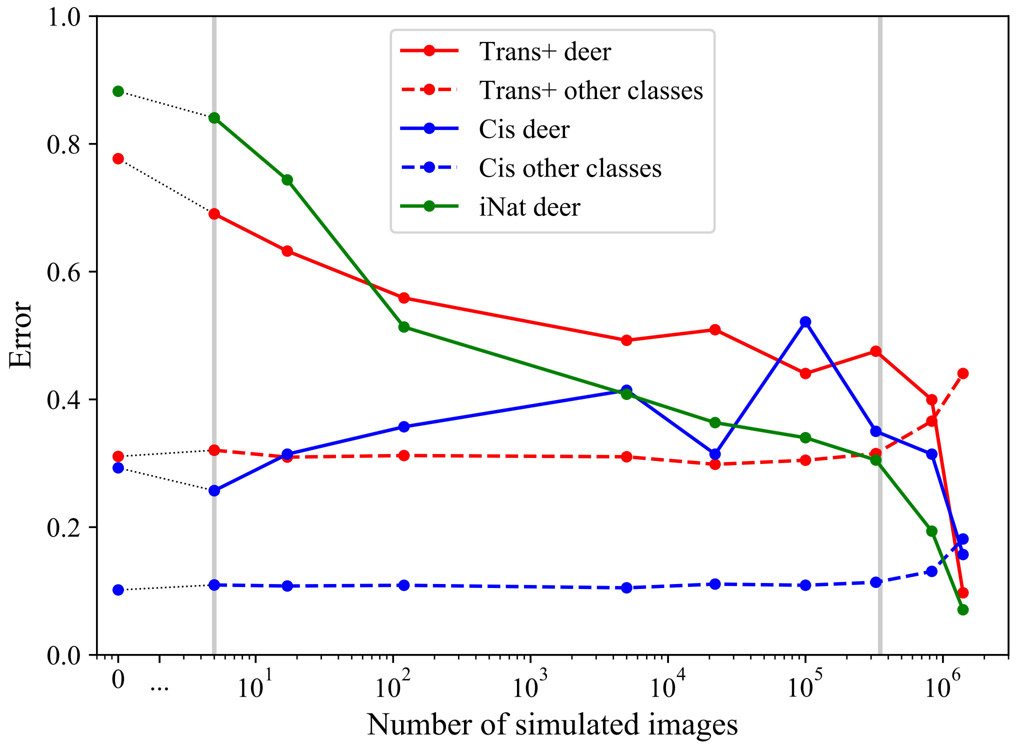
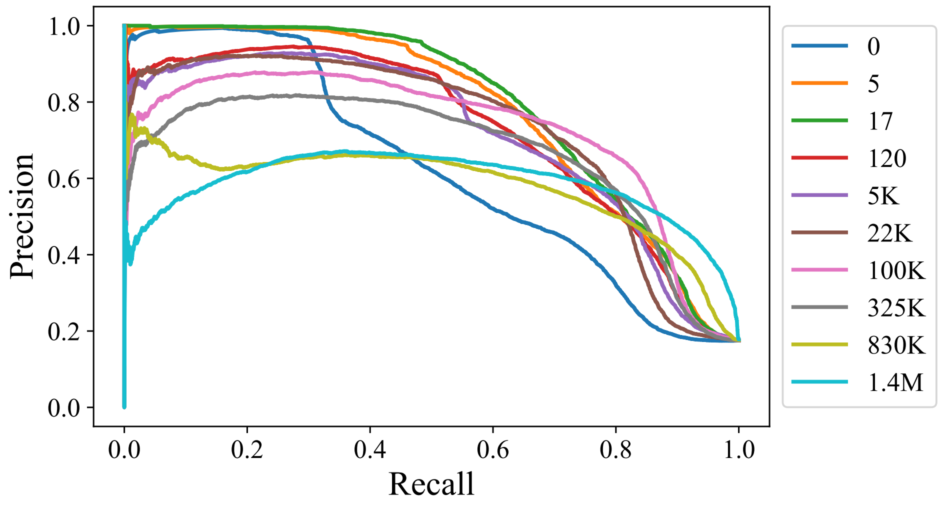
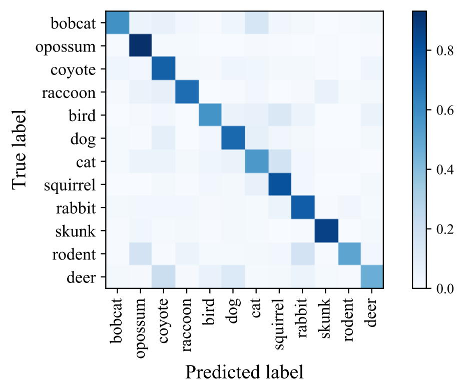


Figure 3.4: **Error as a function of number of simulated images seen during training. We divide this plot into three regions.** The leftmost region is the baseline performance with no simulated data, shown at x=0 (Note x-axis is in log scale). In the middle region, additional simulated training data increases performance on the rare class and does not harm the performance of the remaining classes (trend lines are visualized). The rightmost region, where many simulated images are added to the training set, results in a biased classifier, hurting the performance of the other classes (see Fig.3.5 (b-c) for details). We compare the class error for “deer" and “other classes" in both the “cis" and “trans+" testing regimes. Lines marked “deer" use only the deer test images for the error computation. Lines marked “other classes" use all the images in the other classes (excluding deer) for the error computation. Error is defined as the number of incorrectly identified images divided by the number of images.

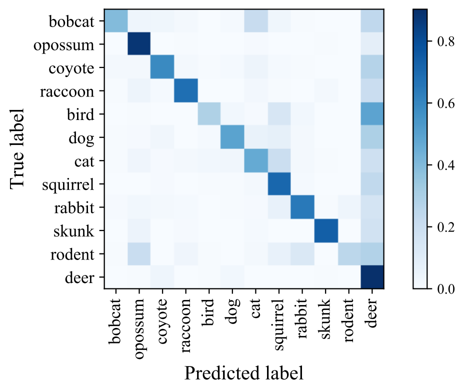
in pretraining, we pretrain our classifiers on *no-animal ImageNet*, a dataset we define by removing the “animal" subtree (all classes under synset node n00015388) from ImageNet. We use an initial learning rate of 0.0045, RMSprop with a momentum of 0.9 (Tieleman and G. Hinton, 2012), and a square input resolution of 299. We employ random cropping (containing at least 65% of the region), horizontal flipping, color distortion, and blur as data augmentation. Model selection is performed using early stopping based on trans+ validation set performance (Bengio, 2012).



**(a)** Trans+ deer precision-recall curves



**(b)** Confusion matrix: 100K



**(c)** Confusion matrix: 1.4M

Figure 3.5: **(a) Trans+ PR curves for the deer class:** Note the development of a biased classifier as we add simulated training data. The baseline model (in blue) has high precision but suffers low recall. The model trained with 1.4M simulated images (in grey) has higher recall, but suffers a loss in precision. **(b-c) Evidence of a biased classifier:** Compare the deer column in the confusion matrices, the model trained with 1.4M simulated images predicts more test images as deer.

### Effect of increase in simulated data

We explore the trade-off in performance when increasing the number of simulated images, from 5 to 1.4 million, spanning 5 log units (see Fig.3.4). Very little simulated data is needed to see a trans+ performance boost: with as few as 5 simulated images, we see a 10% decrease in per-class error on trans+ deer, with $< 0.5\%$ increase in average per-class error on the other trans+ classes. As we increase the number of simulated images, trans+ performance improves: with 100K simulated images, we see a 39% decrease in trans+ deer error, with $< 0.5\%$ increase in error for the

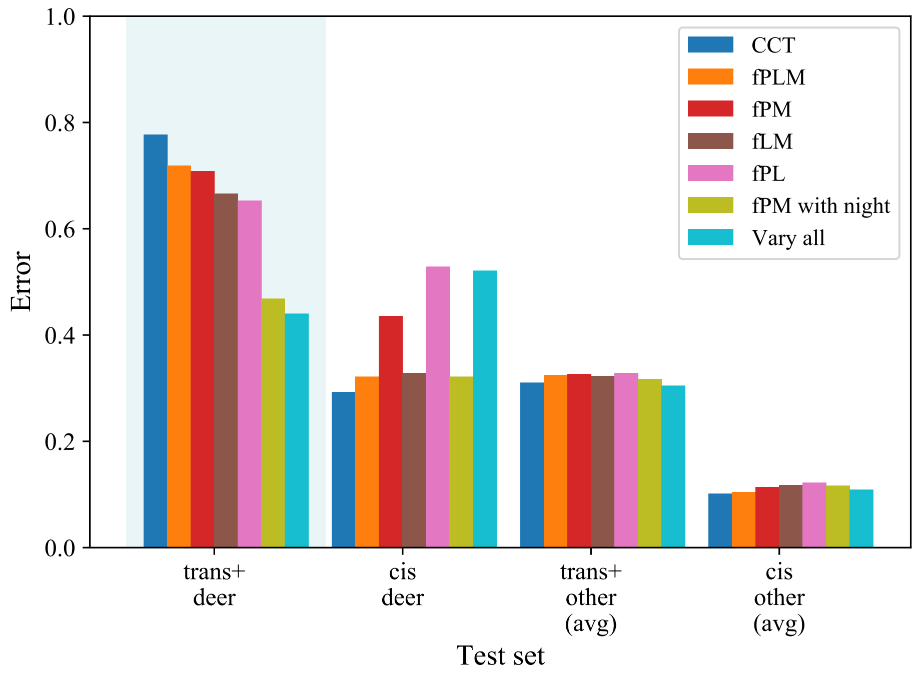


Figure 3.6: **Error as a function of variability of simulated images seen during training: 100K simulated deer images.** Error is calculated as in Fig.3.4, and all data is from TrapCam-Unity. Trans+ deer performance is highlighted. In the legend, “CCT" means the model was trained only on the CCT-20 training set with no added simulated dta. “P" means “pose," “L" means “lighting," and “M" means “model," while the prefix “f" for “fixed" denotes which of these variables were controlled for a particular experiment. For example “fPM" means the pose and the animal model were held fixed, while the lighting was allowed to vary. The variability of simulated data is extremely important, and that while all axes of variability matter, simulating nighttime images has the largest effect.

other trans classes. There exists some threshold ($> 325$K) where, if passed, an increase in simulated data noticeably biases the classifier towards the deer class (see Fig.3.5): with 1.4 million simulated images, our trans+ deer error decreases by 88%, but it comes at the cost of a 13% increase in average per-class error across the other classes. At this point, there is an overwhelming class prior towards deer: the next-largest class at training time would be opossums with $2,514$ images, 3 orders of magnitude less.

Unsurprisingly, cis deer performance decreases with added simulated data. Although the images were taken on different days (train from even days, cis-test from odd days) the animals captured were to some extent creatures of habit. Thus, training and test images can be nearly identical from within the same locations (see Fig.3.2). Almost all cis test deer images have at least one visually similar training image. As simulated data is added at training time, the model is forced to learn a more complex,

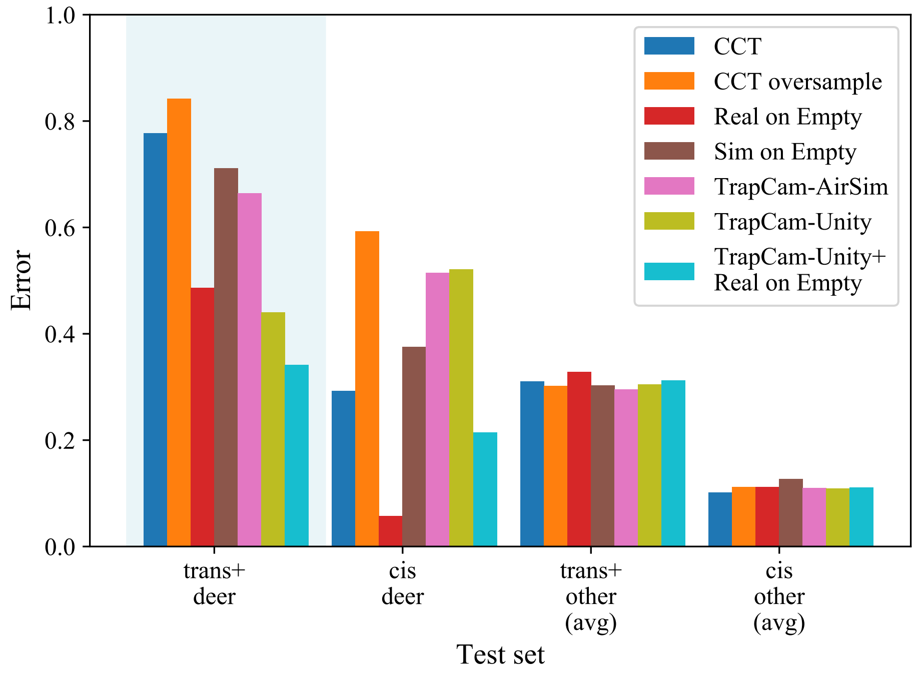
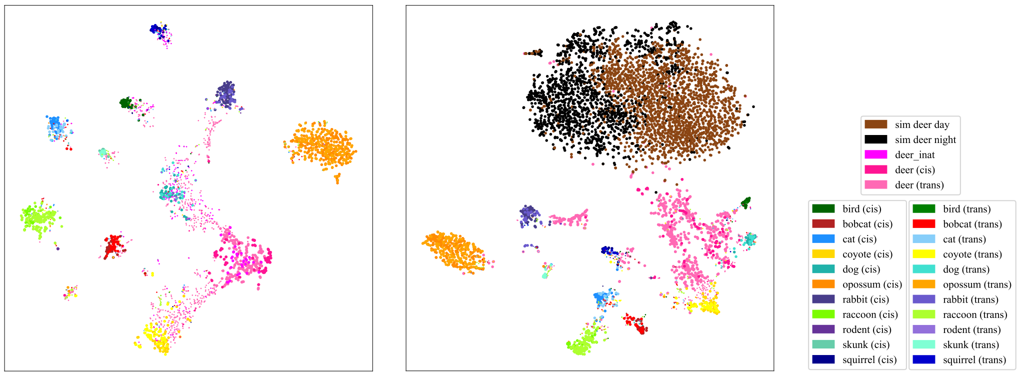


Figure 3.7: **Error as a function of simulated data generation method: 100K simulated deer images.** Per-class error is calculated as in Fig.3.4. Trans+ deer performance is highlighted. Oversampling decreases performance, and there is a large boost in performance from incorporating real segmented animals on different backgrounds (Real on Empty). TrapCam-Unity with everything allowed to vary (model, lighting, pose, including nighttime simulation) gives us slightly better trans+ performance, without requiring additional segmentation annotations. Combining Real on Empty with TrapCam-Unity (50K of each) gives us the best trans+ deer performance.

varied representation of deer. As a result, we see cis deer performance decrease. To quantify robustness, we ran the 100K experiment three times. We found that trans+ deer error had a standard deviation of 2% and cis deer error had a standard deviation of 4%, whereas the average error across other classes had a standard deviation of 0.2% for both cis and trans.

We also investigate performance on deer images from iNaturalist (Van Horn, Mac Aodha, et al., 2017), which are individually collected by humans and are usually relatively centered and well-focused (and therefore easier to classify) but represent a domain shift (see Fig.3.2). Adding simulated data improves performance on the iNaturalist deer images (see Fig.3.4), demonstrating the robustness and generality of the representation learned.



No simulated deer 1.4M simulated deer

Figure 3.8: **Visualization of network activations: more deer are classified correctly as we add synthetic data, despite the synthetic data being clustered separately.** The pink points are real deer, the brown are simulated day images and the grey are simulated night images. Large markers are points that are classified correctly, while small markers are points classified incorrectly. The plots were generated by running 200-dimensional PCA over the activations at the last pre-logit layer of the network when running inference on the test sets, and then running 2-dimensional tSNE over the resulting PCA embedding.

### Effect of variation in simulation

In order to understand which aspects of the simulated data are most beneficial, we consider three dimensions of variation during simulation: pose, lighting, and animal model. Using the TrapCam-Unity simulator, we generate 100K daytime simulated images for each of these experiments. As a control, we create a set of data where the pose, lighting, and animal model are all fixed. We then create sets with varied pose, varied lighting, and varied animal model, each with the other variables held fixed. An additional set of data is generated varying all of the above. Unsurprisingly, widest variation results in the best trans+ deer performance. The individual axes of variation do have an effect of performance, and some are more “valuable" than others (see Fig.3.6). There are many more dimensions of variation that could be explored, such as simulated motion blur or variation in camera perspective. For CCT data, we find adding simulated nighttime images has the largest effect on performance. We have determined that for deer 49% of training images, 53% of cis test images, and 56% of trans+ test images were captured at night, using either IR or white flash. Simulating only daytime images injects a prior towards deer being seen during the day. By training on half day and half night images, we match the day/night prior for deer in the data. Not all species occur equally during the day or night, some are strictly nocturnal. Our results suggest that a good strategy is to determine the appropriate ratio of day to night images using your training set and match that ratio when adding simulated data.

### Comparing simulated data generation methods

We compare performance gain from 4 methods of data synthesis, using 100K added deer images for each (see Fig.3.7. The animal model is controlled (each simulated set uses the same GiM deer model for these experiments) for fair comparison of the efficacy of each generation method. As an additional control, we consider oversampling the rare class. This creates the same sampling prior towards deer without introducing any new information. Oversampling performs worse than just training on the unbalanced training set by causing the model to overfit the deer class to the training images. By manually segmenting out the deer in the 44 training images and randomly pasting them onto empty backgrounds, we see a large improvement in performance. Cis error goes down to 6% with this method of data augmentation, which makes sense in the view of the strong similarities between the training and cis-test data (see Fig.3.2).

Real on Empty and Sim on Empty are able to approximate both “day" and “night" imagery, a deer pasted onto a nighttime empty image is actually a reasonable approximation of an animal illuminated by a flash at night (see Fig.3.1). They also have the additional benefit of using backgrounds from both cis and trans sets, giving them trans information not provided by the simulated datasets. TrapCam-Unity with all variability enabled is our best-performing model without requiring additional segmentation annotations. If segmentation information is available, Real on Empty combined with TrapCam-Unity (50K of each) improves both cis and trans deer performance: trans deer error decreases to 36% (a 54% decrease compared to CCT only), with $< 2\%$ increase in error on trans other classes.

### Visualizing the representation of data

In order to visualize how the network represents simulated data vs. real data, we use PCA and tSNE (Maaten and G. Hinton, 2008) to cluster the activations of the final pre-logit layer of the network. These visualizations can be seen in Fig.3.8. Interestingly, the model learns “deer" bimodally: simulated deer are clustered almost entirely separately from real deer, with a few datapoints of each ending up in the opposite cluster. Even though those clusters overlap only slightly, the network is surprisingly able to classify more deer images correctly.

## 3.5 Conclusions and Future Work

We present two fast, realistic natural world data simulators based on popular 3D game development engines. Our simulators have 3 major advantages. **First**, they are generalizable. Thanks to the abundant 3D assets available online in the game development community, integrating a new species in a new environment from off the shelf assets is simple and fast. **Second**, not only are the graphics near-photorealistic, the pipeline also generates animals with realistic pose, animation, and interactions with the environment. **Third**, data generation is efficient. A single gaming PC generates over 300,000 full-HD images with pixel-level instance annotation per day and the throughput linearly scales to additional machines.

We explore using the simulated data to augment rare classes during training. Towards this goal, we compare multiple sources of natural-world data simulation, explicitly measure generalization via the cis-vs-trans paradigm, examine trade-offs in performance as the number of simulated images seen during training is increased, and analyze the effect of controlling for different axes of variation and data generation methods.

From our experiments, we draw three main lessons. First: using synthetic data can considerably reduce error rates for classes that are rare, and with segmentation annotations we can reduce error rates even further by additionally randomly pasting segmented images of rare classes on empty background images. Second: as the amount of simulated data is increased, accuracy on the target class improves. However, with 1000x more simulated data than the common classes, we see negative effects on the performance of other classes due to the high class imbalance. Third: the variation of simulated data generated is very important, and maximum variation provides maximum performance gain.

While an increase in simulated data corresponds to an increase in target class performance, the representation of simulated data overlaps only rarely with real data (see Fig.3.8). It remains to be studied whether embedding techniques (Schroff, Kalenichenko, and Philbin, 2015), domain adaptation techniques (Ganin and Lempitsky, 2015; Zou et al., 2018), or style transfer (Goodfellow et al., 2014; Shrivastava et al., 2017) could be used to encourage a higher overlap in representation between the synthetic and real data, and if that overlap would lead to an increase in categorization accuracy. Additionally, the bias induced by adding large amounts of simulated data could be addressed with algorithmic solutions such as those in (Cui, Jia, et al., 2019; Elkan, 2001; H. He and Garcia, 2008; H. He, Bai, et al., 2008). We have not discussed the drawbacks related to model training with large quantities of synthetic data (epoch time, data storage, etc.). In the future, we will explore merging the simulator and classifier so that highly variable synthetic data could be requested "online" without storing raw frames.

Simulation is a fast, interpretable, and controllable method of data generation that is easy to use and easy to adapt to new classes. This allows for an integrated and evolving training pipeline with new classes of interest: simulated data can be generated iteratively based on needs or gaps in performance. Our analysis suggests a general methodology when using simulated data to improve rare-class performance: 1) generate small, variable sets of simulated data (even small sets can drive improvement), 2) add these sets to training and analyze performance to determine ideal ratios and dimensions of variation, and 3) take advantage of ease and speed of generation to create an abundance of data based on this ideal distribution, and determine an operating point of number of added simulated images to optimize performance between rare target class and other classes based on the project goal.

Further, the performance gains we have demonstrated, along with the data generation tools we contribute to the community, will allow biodiversity researchers focused on endangered species to improve classification performance on their target species. Adding each new species to the simulation tools currently requires the assistance of a graphics artist. However, automated 3D modeling techniques, such as those proposed in (Kanazawa et al., 2018; Reinert, Ritschel, and Seidel, 2016; Cashman and Fitzgibbon, 2013; Pahde et al., 2019), might eventually become an inexpensive and practical source of data to improve few-shot learning.

The improvement we have found in rare-class categorization is encouraging, and the release of our data generation tools and the data we have generated will provide a good starting point for other researchers studying imbalanced data, simulated data augmentation, or natural-world domains.

## 3.6 Supplementary Materials

### Architecture Selection

To select a single classification architecture to use across our experiments, we trained three classifiers: ResNet-101 V2, Inception V3, and Inception-ResNet V2. All three classifiers were pretrained on *no-animal ImageNet* then trained on the Caltech Camera Traps (CCT) training set (described in the main paper, section 3.1) with no added simulated images. We found that Inception-ResNet V2 performed best on deer in cis and trans scenarios (see Table 3.1), so we decided to use Inception-ResNet V2 as the base architecture for all further experiments.

Table 3.1: **Error for different architectures.** Error is defined as the number of incorrectly identified images divided by the number of images for each test set, where “Deer" contains only deer images and “Other" contains all non-deer images.

| | Cis Test | | Trans+ Test | |
|---|---|---|---|---|
| Architecture | Deer | Other | Deer | Other |
| Resnet 101 V2 | 47.86 | 11.18 | 88.63 | **29.76** |
| Inception V3 | 50.00 | 11.74 | 81.73 | 32.74 |
| Inception Resnet V2 | **29.28** | **10.17** | **77.69** | 31.07 |

**Additional analysis**

### Analyzing the value of real images

We find that our simulated data is sufficient to learn to recognize some deer even without real examples, though the real examples give a large boost in performance. The performance breakdown can be seen in Table 3.2. These results are promising for both researchers studying zero-shot learning and biologists studying highly endangered species: it is possible to learn a species with no real training data. This avenue remains open for further study.

### Comparing night and day performance

We further analyze the effect of day and night simulation by comparing three experiments: one trained with only simulated daytime images, one trained with only simulated nighttime images, and one trained with half day and half night (see Fig 3.9). We find that the models trained on only day and only night perform similarly on trans deer, and that the 50/50 split performs best on trans deer (highlighted region in Fig 3.9). Training on day or night alone gives us a 20% performance boost on trans deer, while training on both gives us a 40% performance boost. This suggests that the day and night simulated images help the classifier in complementary ways: day helps with day images and night helps with night images. Performance on other classes is not strongly effected. Cis performance is quite noisy, and performs best with no added simulated data, see Fig. 2 in the main paper for further analysis.

Table 3.2: **Error with and without the 44 real deer examples when adding 100K simulated deer images.** Error is computed as in Table 3.1.

| | Cis Test | | Trans+ Test | |
|---|---|---|---|---|
| Real Training Data | Deer | Other | Deer | Other |
| CCT train w/o deer | 94.29 | 18.64 | 68.56 | 34.42 |
| CCT train w/ deer | 52.14 | 10.91 | 44.05 | 30.47 |
| % decrease from real deer | 44.7 | 41.5 | 35.7 | 11.5 |

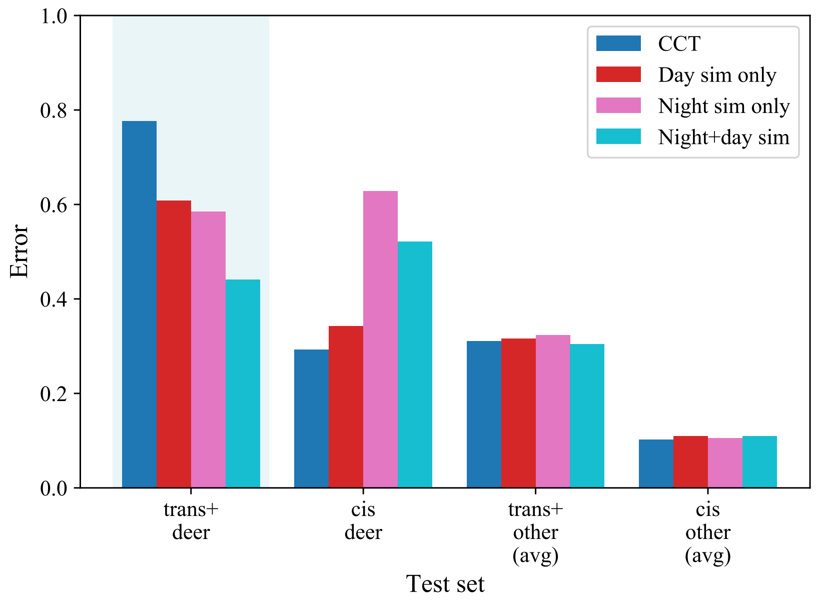


Figure 3.9: **Error as a function of day or night simulated images: 100K simulated deer images.** Error is calculated as in Fig. 4 in the main paper. Trans+ deer performance is highlighted. Models trained on added night- or day-only simulated data perform better on trans deer than CCT alone, but the best trans deer performance comes from the 50/50 day/night split of added simulated data.

**Investigating the effect of adding simulated data for a common class**

In order to investigate how added simulated data might effect a common class, as opposed to a rare one, we created "coyote" simulated data with TrapCam-Unity, using rendered models of wolves as a proxy for coyotes. Off-the-shelf, high-quality wolf models were more widely available, and wolves and coyotes are visually very similar (see Fig.3.11). This is a coarse-grained experiment, and it remains to be seen what would happen if simulated data from two visually similar classes (wolves and coyotes) was added at the same time.

We find that adding simulated "coyote" data improves trans+ coyote performance slightly, while cis coyote performance remains the same. Unsurprisingly, for the deer class (which has few training examples) adding a large amount of simulated coyote data harms both cis and trans+ deer performance.

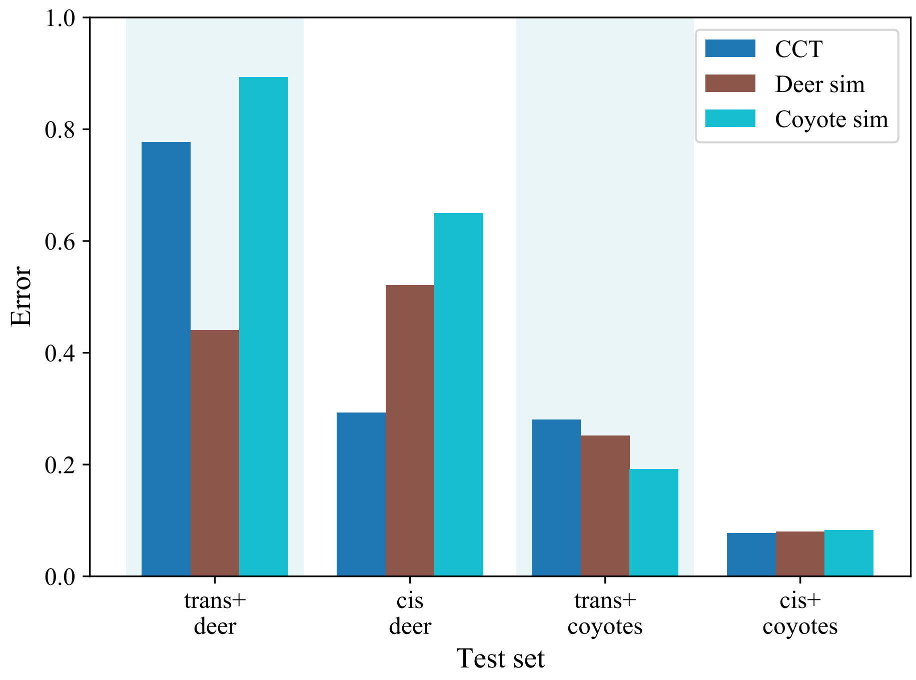
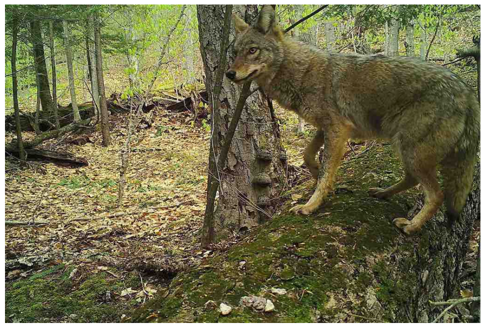
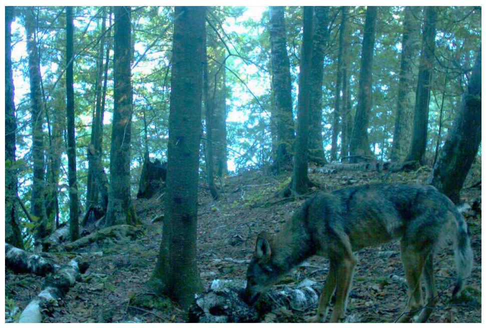


Figure 3.10: **Error as a function of deer or coyote simulated images: 100K simulated images.** Error is calculated as in Fig. 4 in the main paper. Trans+ deer and coyote performance are highlighted.



Coyote (*Coyote in a camera trap* n.d.)



Wolf (*Wolf in a camera trap* n.d.)

Figure 3.11: **Wolves and coyotes are visually similar.**

## Creating Sim and Real on Empty Data

Alternative to the full synthetic methods of data generation with AirSim and Unity, we generated synthetic images by overlaying either simulated deer or real cropped deer on real empty background images from the CCT dataset (see Fig. 3.12).

For the *Sim on Empty* dataset generation, we posed either a stag or a doe deer from the GiM model set in front of a simulated camera in Unity. We randomized the animation, orientation in azimuth (0-360 degrees), position, direction of light orientation in azimuth (0-360 degrees), and elevation (20-90 degrees).

For the *Real on Empty* dataset, we manually segmented and cropped out the 44 instances of deer from the CCT training set. Then we pasted the cropped deer foreground images on top of empty camera trap images in random locations.

**TrapCam-AirSim Details**

It took time and thought to derive the overall requirements for the AirSim TrapCam environment. With a sizable number of potential biomes globally, we narrowed the scope of what we intended to build to a SW United States environment similar to what is seen in the CCT data. Eventually we settled on a sub-alpine woodland scene that is readily found across most of the Western/ Southwest US. A major requirement and challenge was how to get the most data out of a relatively small, but detailed, area – this was key to the project without expanding the size of the area of interest. The overall intent was to leverage Microsoft AirSim's computer vision mode to move a pre-configured camera around the scene, providing varied background.

We used various off-the-shelf components such as an animal pack from Epic Studios (*Epic Studios* n.d.) (Animals Vol 01: Forest Animals by GiM (*Forest Animals by GiM* n.d.)), background terrain from Unreal Marketplace (*UNREAL Game Engine* n.d.), vegetation from SpeedTree (*SpeedTree* n.d.), and rocks/obstructions from Megascans (*Quixel Megascans Library* n.d.). In other AirSim environments, the general scenery is fairly static with exception of particle effects (snow/rain/dust/etc). For this effort, we wanted a method to vary the background, to replicate a variety of terrains within a single environment (see Fig.3.13). The actual area of the environment is small, at 50 meters long, but the modularity allows many possible scenes to be constructed. The randomization was designed to facilitate artists by allowing them to make a list of different objects to randomize from. Those objects are prioritized based on their order on the list. The BiomeTerrain class generates them by tracing random areas across the field based on a global seed. If there's space available it spawns the desired object. There are a number of object types available in TrapCam-AirSim; animal type, rocks, logs, grasses, shrubs, trees, and each type can be varied by density and distribution. Additionally, we provide 9 GiM animal models: deer (doe/stag), wolf, fox, rat, spider, bear, raccoon, and buffalo. The doe

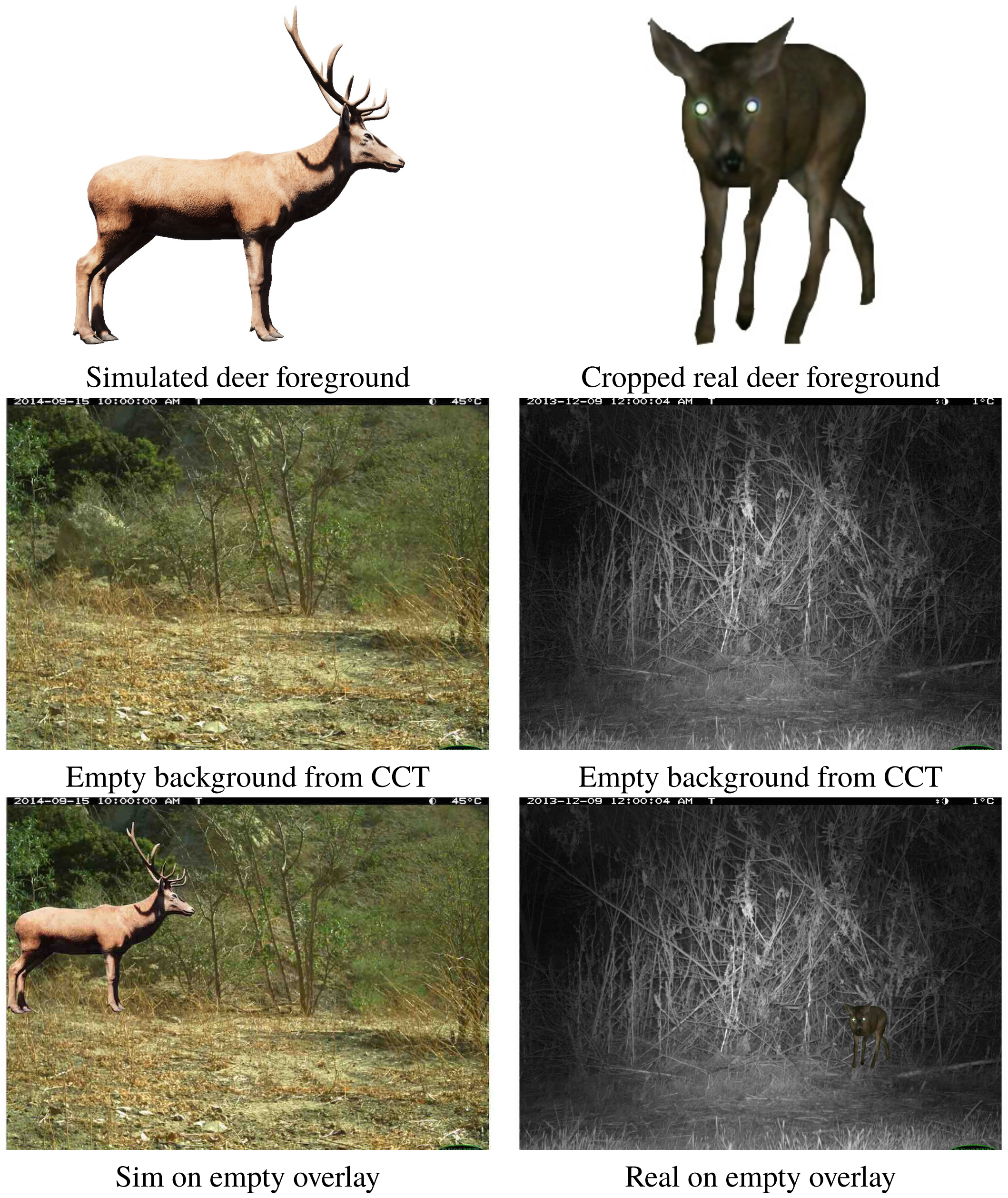

Simulated deer foreground
Cropped real deer foreground
Empty background from CCT
Empty background from CCT
Sim on empty overlay
Real on empty overlay

Figure 3.12: **Sim and Real on Empty Generation.** (a),(c),(e) demonstrate the process of overlaying a simulated deer on top of an empty background image from the CCT dataset. (b),(d),(f) show the process of overlaying a cropped real deer on top of an empty background image from the CCT dataset.

model was created by removing the antlers from the stag model with Maya (*Maya* n.d.), a common modeling tool. All animal objects were assigned segmentation IDs for efficient ground truth extraction.

We created a simple UI to vary parameters, along with a command line API for

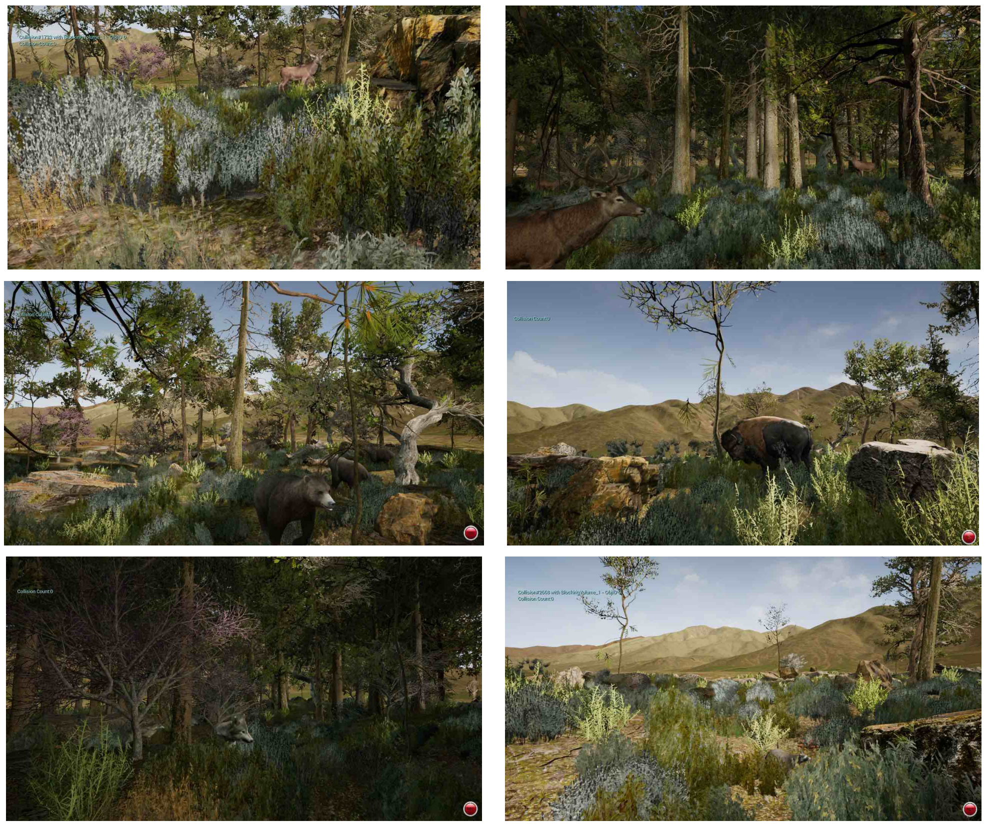
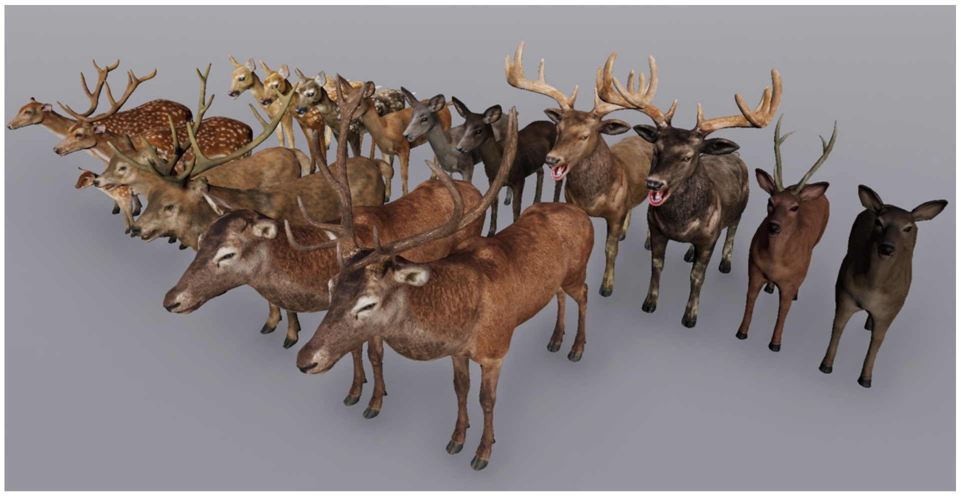


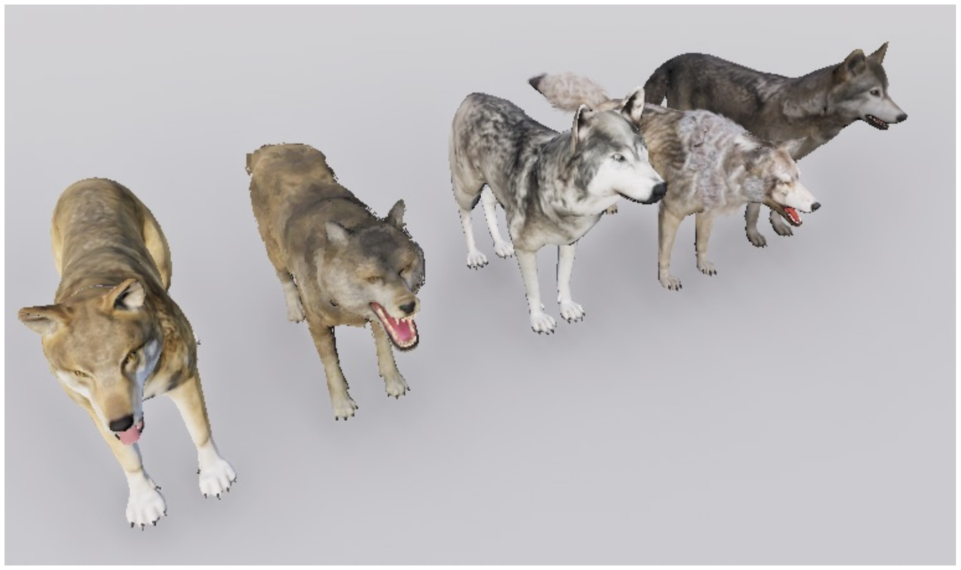
Figure 3.13: **TrapCam-AirSim environment.** The TrapCam-Airsim envionment was designed to be modular and randomizeable, which allows a variety of biomes to be synthesized within a limited simulated area.

parameter configuration. The UI was constructed with Unreal Motion Graphics (UMG) Widgets and allows for future flexibility for modifications, DPI resolutions and platforms. The main core functionalities were created with C++ for better performance as a parent class for data-only blueprints, which allows the technical artists to easily swap assets for different environments without re-compiling the C++ code.

We started the requirements and scoping in mid-August 2018 with a go-ahead of approximately September 6, and produced a working prototype two weeks later, with continued development and refining through mid-October. A second phase late in the year modified the camera system to include flash capability, and animals were updated to provide eye-shine, and the UI was modified to include variability for that eye-shine.



Models of deer



Models of wolves

Figure 3.14: **Models of deer and wolves.** In TrapCam-Unity, we used 17 different models of deer from 5 different artists and 5 models of wolves from 5 different artists. We used the wolf models as proxies for coyotes (see Section 3.6). Model details are available in section 3.6.

**TrapCam-Unity Details**

The “Book of The Dead" environment (*Book of the Dead Environment* n.d.) we use is published for free by Unity. As shown in Fig.3.15, the near-photorealistic environment simulates a large patch of forest in a valley with volumetric grass, a variety of high definition trees, logs, and bushes, as well as rocks and terrain. The environment is a irregular area of roughly 20,000 $m^2$. It runs on a desktop PC in real time and enables us to generate large amounts of images efficiently.

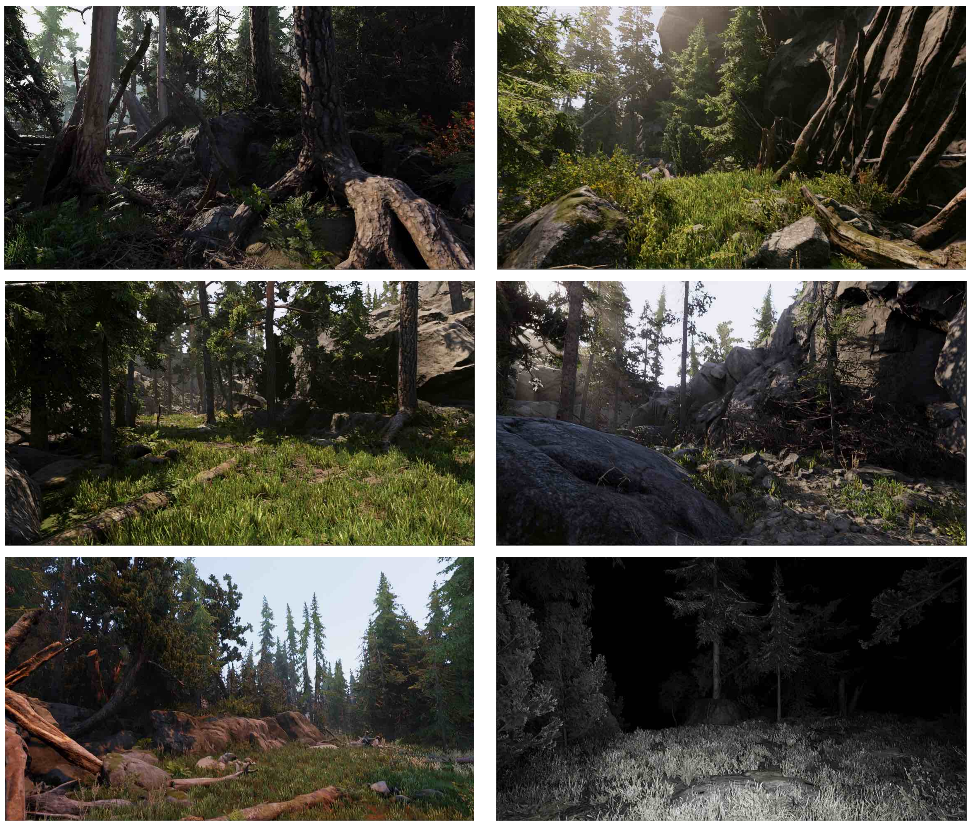


Figure 3.15: **TrapCam-Unity environment.** The Book of The Dead environment is a large natural environment with diverse sub regions.

To create daytime images, we varied the orientation of the simulated sun in both azimuth and elevation. To create images taken at night, we created a spotlight attached to the simulated camera to simulate a white-light or IR flash and qualitatively match the low color saturation of the night time images. To simulate animals' eyeshine (a result of the reflection of camera flash from the tapetum lucidum), we placed small reflective balls on top of the eyes of model animals (see Fig.3.16).

For deer simulation, we used 17 animated deer models from 5 publishers on Unity (GiM(*GiM Studio* n.d.), 4toon(*4toon studio* n.d.), Protofactor(*Protofactor Inc* n.d.), Red Deer(*Red Deer Studio* n.d.), and Janpec(*Janpec* n.d.)). For coyote simulation, we used 5 models from 5 publishers (GiM(*GiM Studio* n.d.), 4toon(*4toon studio* n.d.), Protofactor(*Protofactor Inc* n.d.), Janpec(*Janpec* n.d.), and WDallgraphics(*WDallgraphics studio* n.d.)). We created the GiM doe model by removing the antlers of the GiM stag model with Blender(*Blender* n.d.). For each of the animated models, we included an animation controller that contains several animation clips ranging from commonly seen behavior episodes like walking and eating, to rare occurrences like attacking and sleeping. During dataset generation, we randomly picked a clip for each instance of animals and froze it at a random time point, then we moved the cameras around to sample a static scene with animals and environment.

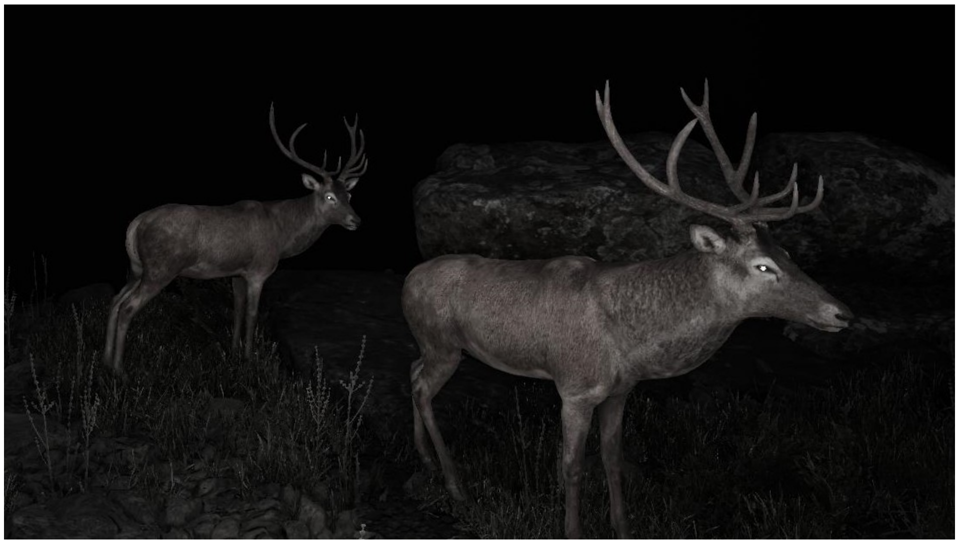


Figure 3.16: **Example of eyeshine simulation.**

We had 300 seed locations and randomly placed animals in the vicinity of a subset of the seed locations. This process was repeated multiple times to simulate animals in random locations within the environment. A similar random placement process was used to determine the locations of the cameras. All images generated are in full HD resolution (1980 x 1080).

For ground truth generation, we turned off the lighting and rendered each instance of the animal in a unique color by replacing the original animal shader with an unlit shader. We then used customized python scripts to extract animal bounding boxes by extracting pixels with these unique colors.

*Chapter 4*

# PANDA: PANOPTIC DATA AUGMENTATION

Liu, Yang, Pietro Perona, and Markus Meister (2019). "PanDA: Panoptic Data Augmentation". In: *arXiv preprint arXiv:1911.12317*. URL: `https://arxiv.org/abs/1911.12317`.

## ABSTRACT

The recently proposed panoptic segmentation task presents a significant challenge to image understanding with computer vision by unifying semantic segmentation and instance segmentation tasks. In this chapter, we present an efficient and novel panoptic data augmentation (PanDA) method which operates exclusively in pixel space, requires no additional data or training, and is computationally cheap to implement. By retraining original state-of-the-art models on PanDA augmented datasets generated with a single frozen set of parameters, we show robust performance gains in panoptic segmentation, instance segmentation, as well as object detection across models, backbones, dataset domains, and scales. Finally, the effectiveness of unrealistic-looking training images synthesized by PanDA suggests that one should rethink the need for image realism for efficient data augmentation.

## 4.1 Introduction

With the rapid development of convolutional neural networks (CNN) and the availability of large-scale annotated datasets like ImageNet (Russakovsky et al., 2015), modern computer vision models have reached or surpassed human performance in many domains of visual recognition (K. He, X. Zhang, et al., 2015; Cireşan, Meier, and Schmidhuber, 2012). Much of the interest of the community has since shifted from visual recognition to formulating and solving more challenging tasks such as object detection, semantic segmentation, and instance segmentation.

Recently, (Kirillov et al., 2019) proposed the panoptic segmentation task that unifies instance segmentation and semantic segmentation. The task requires a model to assign each pixel of an image a semantic label and an instance ID. Several panoptic datasets, such as the Cityscapes (Cordts et al., 2016), Microsoft COCO (Lin, Maire, et al., 2014), ADE20K (B. Zhou et al., 2017), and Mapillary Vistas (Neuhold et al., 2017), have been released. Much of research attention has focused on developing new models (Xiong et al., 2019; Porzi et al., 2019; Gao et al., 2019). So far, the best performing models on the leader boards of various major panoptic challenges are exclusively CNN based.

In this paper, insead of developing new models, we focus on the data augmentation aspect of the panoptic segmentation task, and develop a novel panoptic data augmentation method, PanDA, that further improves the performance of original top performing models from 2019.

We first identify the data deficiency and class imbalance problems and propose a pixel space data augmentation method for panoptic segmentation datasets that efficiently generates synthetic datasets from the original dataset. The method is computationally cheap and fast, and it requires no training or additional data. To demonstrate the robustness of PanDA, we use **a single frozen set of parameters** throughout the paper except for the Ablation Study section.

Next, we experimentally demonstrate that training with PanDA augmented Cityscapes further improves all performance metrics of original top-performing UPSNet (Xiong et al., 2019) and Seamless Scene Segmentation (Porzi et al., 2019) models. With PanDA, robust boosts in panoptic segmentation, instance segmentation, and detection with UPSNet with ResNet-50/101 backbones and Seamless Scene Segmentation model with ResNet-50 backbone are reported.

To further demonstrate the generalizability of PanDA across scales and domains, we apply it to Cityscapes subsets with 10 to 3,000 images, as well as a 10 times larger COCO subset with 30,000 images. We report performance gains across scales and datasets. By quantifying the log-linear relationship between number of training images and final performance metrics, we show that on average a PanDA generated image is 20-40% as effective as an original image.

Finally, results from the ablation study show that, contrary to common beliefs, less realistic looking images improve model performance more. It suggests that the level of image realism is not positively correlated with data augmentation efficiency.

## 4.2 Related Work

### Panoptic Segmentation

The panoptic segmentation task was first proposed in 2019 by Kirillov et al. (Kirillov et al., 2019) as an attempt to formulate a unified task that bridges the gap between semantic segmentation and instance segmentation. The task divides objects in to two super-categories: *things* and *stuff*. For classes in things, each pixel in the image is labeled with a class ID and an instance ID. For classes in stuff, each pixel is labeled with a class ID only. In addition to traditional metrics, such as mean intersection over union (mIoU) and average precision of instance segmentation (AP), the panoptic quality (PQ) is defined as,

$$PQ = \underbrace{\frac{\sum_{(p,g)\in TP} IoU(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}} \tag{4.1}$$

where true positives (TP) are predicted segments that have strictly greater than 0.5 overlap with ground truth segments. PQ is the product of segmentation quality (SQ) and recognition quality (RQ).

### Data Augmentation

Despite the efforts towards low-shot learning, modern CNN-based models are still data-hungry in that they have very large model capacity to even memorize randomly labeled large datasets (C. Zhang et al., 2016). One efficient way to regularize models and promote generalization is data augmentation.

Many methods still in use for detection and segmentation models (Xiong et al., 2019; Porzi et al., 2019; Gao et al., 2019) are largely inherited from the ImageNet visual recognition era (Krizhevsky, Sutskever, and G. E. Hinton, 2012). These methods take advantage of object invariances in pixel space: manipulations such as crop, horizontal flip, resize, color distortion, and noise injection do not usually change the identity of the object. These methods are simple in concept and computationally very cheap, but they do not take advantage of the more informative ground truth labeling that accompanies the harder detection and segmentation tasks. Some recent methods start to explore the use of bounding box and segmentation information (S. Liu et al., 2019; Beery, Y. Liu, Morris, Piavis, Kapoor, Meister, et al., 2019; H.-S. Fang et al., 2019; R. Shetty, Schiele, and Fritz, 2019; Dvornik, Mairal, and Schmid, 2019). InstaBoost (H.-S. Fang et al., 2019) is proposed as a method for instance segmentation where instances are cropped out and moved to a different location of the same image, then the hole in background is filled with a classical in-paint network (Bertalmio, Bertozzi, and Sapiro, 2001). Shetty et al. (R. Shetty, Schiele, and Fritz, 2019) augment images by removing certain instances that provide context and use a CNN based in-paint model (R. R. Shetty, Fritz, and Schiele, 2018) to fill the holes. Dvornik et al. (Dvornik, Mairal, and Schmid, 2019) use a CNN based context model to guide the addition of instances from an "instance-database" to original images, avoiding making holes in the original images and the need for in-painting. The Discussion Section offers a detailed comparison relating (H.-S. Fang et al., 2019), (R. Shetty, Schiele, and Fritz, 2019) and (Dvornik, Mairal, and Schmid, 2019) to the present work.

Data simulation is another flavor of data augmentation. Thanks to the development of graphical simulation engines, one can generate photo-realistic images together with pixel perfect ground truth. The method is proven to work for many tasks such as human pose estimate (Varol, Romero, X. Martin, Mahmood, Michael J Black, et al., 2017b), wildlife classification (Beery, Y. Liu, Morris, Piavis, Kapoor, Meister, et al., 2019), and object detection (Hinterstoisser et al., 2019). However, this method often requires handcrafted 3D models, and there usually is a significant domain gap between real images and simulated images.

Recently, with the popularization of generative models such as generative adversarial networks (GAN), researchers also add images generated by GANs to the training set (Frid-Adar et al., 2018; Bowles et al., 2018; S. Liu et al., 2019). However, as a type of neural network based model, the GAN itself requires training, which in turn is both computationally expensive and data hungry.

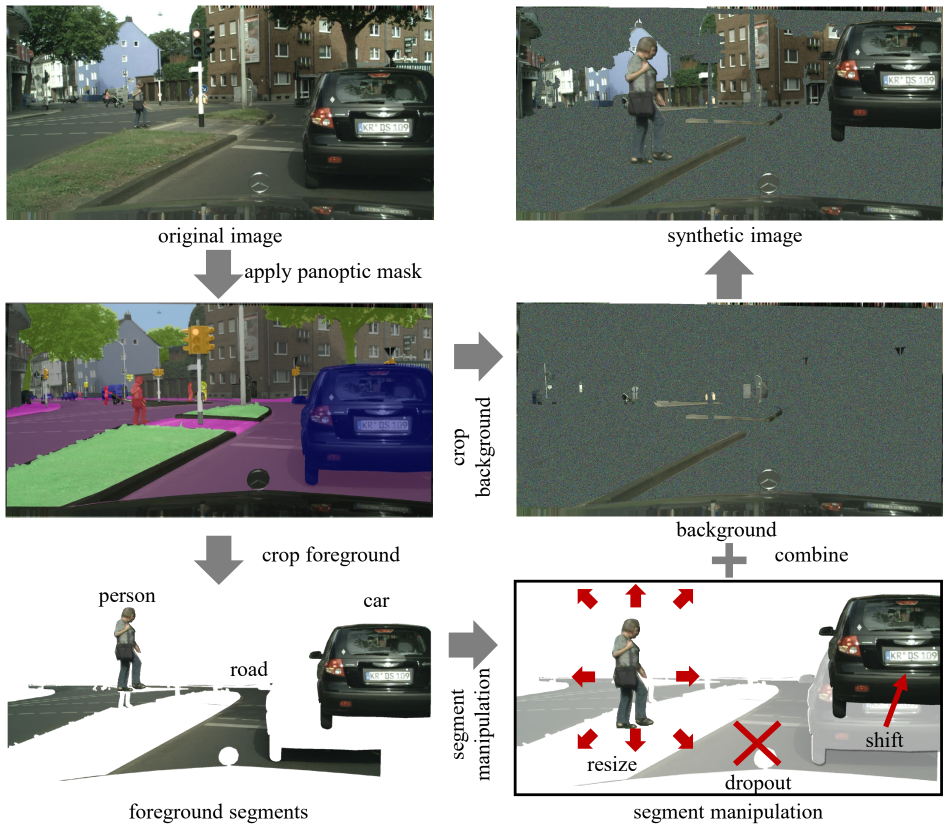


Figure 4.1: **Schematics of PanDA.** Foreground and background segments are greatly simplified for clear demonstration of the image decomposition and synthesis process. Note that background in the panoptic segmentation task usually takes a small percentage of pixels in an image

## 4.3 Panoptic Data Augmentation

Annotating images for panoptic segmentation can be costly, since every pixel in an image has to be semantically labeled, and pixels that belong to *things* classes must have an additional instance ID associated. One could argue that it is orders of magnitude harder for a human annotator to generate panoptic ground truth than to assign a single class ID to an image as needed in visual recognition. But with great challenges usually come great opportunities, PanDA takes advantage of the rich information embedded in panoptic annotations and uses it to synthesize new images with ground truth.

A fundamental feature that distinguishes PanDA from other data augmentation methods is that PanDA does not aim to create images that look realistic to human eyes. Many 3D simulation based methods (Beery, Y. Liu, Morris, Piavis, Kapoor,

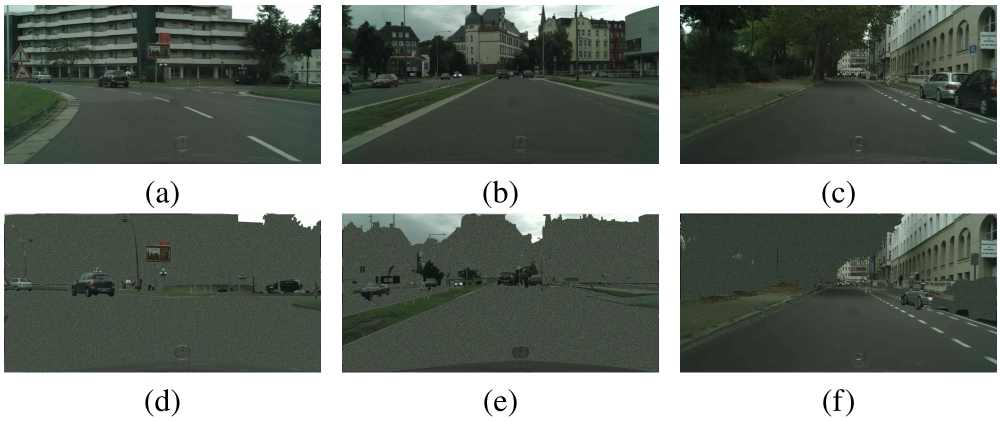


Figure 4.2: **Examples of original cityscapes images (*top row*) and PanDA generated images (*bottom row*).**

Meister, et al., 2019; Stephan R Richter et al., 2016b; Varol, Romero, X. Martin, Mahmood, Michael J Black, et al., 2017b) and cut-paste methods (R. Shetty, Schiele, and Fritz, 2019; Dvornik, Mairal, and Schmid, 2019; H.-S. Fang et al., 2019) implicitly or explicitly suggest that the level of realism is key to achieving high performances. Similary, GAN-based image synthesis methods (Frid-Adar et al., 2018; Bowles et al., 2018; R. Huang et al., 2017) also usually optimize for realism, since the very goal of the generator network in GANs is to synthesize images realistic enough to fool the discriminator network into thinking that they are real (Goodfellow et al., 2014). In contrast, PanDA takes more principled approaches to balance classes, break local pixel structures, and increase variations of objects, and the final images synthesized do not look fully realistic to human eyes.

As shown in Fig. 4.1, PanDA first decomposes a panoptically labeled training image by using the segmentation ground truth to divide it into foreground and background segments. It is worth noting that foreground segments not only include instances in *things* classes, but also segments in *stuff* classes which are considered background in InstaBoost (H.-S. Fang et al., 2019) and in Dvornik, Mairal, and Schmid, 2019. Unfilled pixels are padded with white noise patterns instead of in-painting as seen in (H.-S. Fang et al., 2019) and (R. Shetty, Schiele, and Fritz, 2019). Noise patterns are used because they do not belong to any known categories, making ground truth assignment unambiguous. Foreground segments are then overlaid on top of the new background image one by one. The same manipulation is applied to the ground truth segmentation image to generate the new ground truth segmentation.

For foreground segments, we can perform a series of pixel space operations such as dropout, resize, and shift to control different aspects of each individual object instance. **Dropout** is used to remove segments from an image. It serves to remove well segmented and classfied objects from the image. The **resize** operation changes the size of a segment while preserving the original aspect ratio, and it simulates object movement in depth and introduces more size variance in objects. It resembles zooming and multi-scale training (Xiong et al., 2019) on the individual object level. Random **shifting** moves the segment in x and y in pixel space. It prevents memorization of object locations by the model and breaks the local pixel relationship between the object and its background, thus creating new local contexts for the object. Resize and shift together simulate 3D translation of objects in space, and dropout controls the frequency of objects. Because the ground truth depth information to recover the depth ordering of segments is not available, and we do not want larger segments to occlude smaller ones, we sort the segments by their area and put largest segments on the bottom layer. It is worth noting that more operations can be added to the repertoire of the PanDA tool set, and different variants of the aforementioned operations may also be implemented. In this paper, we limit the scope to the operations discussed above to allow for in-depth experiments with PanDA.

## 4.4 Experiments

### UPSNet and Seamless Scene Segmentation Models

The UPSNet model and the Seamless Scene Segmentation (SSS) models are two independently developed top-performing panoptic segmentation models. To date, they are also the only two models with code published to reproduce the results.

The UPSNet model (Xiong et al., 2019) is one of the top performing single models on both Cityscapes and MS COCO panoptic challenges. It uses a shared CNN feature extractor backbone and multiple heads on top of it. The backbone is a Mask R-CNN feature extractor built on a deep residual network (ResNet) with a feature pyramid network (FPN). UPSNet has an instance head which follows the Mask R-CNN design and a semantic head that consists of a deformable convolution based sub-network. The outputs of the two heads are then fed into the panoptic head, which generates the final panoptic prediction.

The SSS model (Porzi et al., 2019), which was developed around the same time early in 2019, roughly follows the same meta-architecture with a Mask R-CNN instance head and a Mini Deeplab (MiniDL) semantic head taking input from a common FPN feature extractor and feeding into a fusion head for panoptic prediction. At the time of its publication, it became the top performing model in Cityscapes panoptic challenge with a ResNet-50 backbone (Porzi et al., 2019).

We used all the original hyper-parameters of UPSNet and SSS models for training and inference. We only scale (1) the number of training iterations to keep the number of epochs constant across a wide range of dataset sizes and (2) the learning rate based on the number of GPUs used in the training. All trainings start with ImageNet pretrained ResNet-50/101 models as reported in (Xiong et al., 2019) and (Porzi et al., 2019). The UPSNet models are trained with 8x Nvidia 2080Ti GPUs with 11GB VRAM. The SSS experiments are conducted on Amazon Web Services with 8x Nvidia Tesla V100 GPUs with 32GB VRAM. For the in-depth experiments, we choose UPSNet over SSS because it fits in our server's GPUs.

**Datasets**

The Cityscapes (Cordts et al., 2016) panoptic dataset has 2,975 training, 500 validation, and 1,525 test images of ego-centric driving scenes in urban environments in many cities in Europe. Examples are shown in Fig. 4.2 (a-c). The dense annotation covers 97% of the pixels, which consists of 9 *things* classes and 11 *stuff* classes. We choose Cityscapes as the main testbed for PanDA for several reasons. 1) It is one of the most popular panoptic datasets available, and it has a total of 19 diverse classes that covers a wide range of driving related scenarios. Many modern models(Xiong et al., 2019; Porzi et al., 2019; Gao et al., 2019) report performance on Cityscapes and have published code available. 2) Results on a relatively small set are likely to generalize to other specialized domains where annotated panoptic data is scarce. 3) The small size also makes both data synthesis and training cost manageable. As a result, it is suitable for exploratory studies like this.

To demonstrate the generalizability of PanDA, we also performed additional experiments on the MS COCO dataset, which has 118K training and 5k validation images including 53 *stuff* classes and 80 *things* classes of common objects. It is both orders of magnitude larger in size and more diverse regarding number of classes.

**Augmenting Cityscapes with PanDA**

The fact that multi scale inference and pretraining with additional data improves performance of the lightweight UPSNet-50 model suggests that model capacity is not the major limiting factor for the final performance and that data augmentation may further improve performance. To investigate the relationship between model performance and size of training dataset, we trained the UPSNet-50 model from the same ImageNet pretrained backbone on various subsets of the Cityscapes training set that consist of 10, 100, 1,000, and 2,975 images. Fig. 4.3 shows near perfect log-linear relationships between the number of training images and various performance metrics (PQ: $r^2$ = 0.9995, mean average precision for instance segmentation evaluated at 0.5:0.95 (AP): $r^2$ = 0.9744, mean average precision for detection evaluated at 0.5:0.95 (AP box): $r^2$ = 0.9948). It suggests that adding training images is likely to further improve model performance.

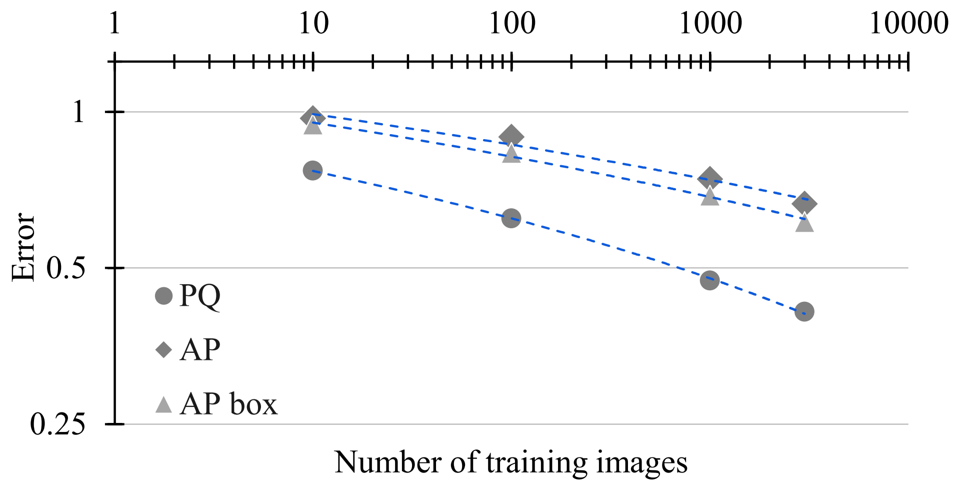


Figure 4.3: **Model performance vs training set size on Cityscapes.** We train UPSNet-50 models on various Cityscapes subsets ranging from 10 to 2,975 images. *PQ*: panoptic quality. *AP*: mean average precision of instance segmentation evaluated at 0.5:0.95. *AP box*: mean average precision of detection evaluated at 0.5:0.95. Dashed curves are log-linear fits. Panoptic segmentation, instance segmentation, and instance detection performance are summarized by PQ, AP, and AP box, respectively.

To test whether adding training images indeed helps, we used PanDA to generate synthetic images from the Cityscapes training set. A key discrepancy between model training and and the final PQ score is that classes with large areas provide overwhelming training signals during training and thus are favored during training, whereas SQ, the first term of the PQ formula, weights small and large objects equally. To mitigate the issue, we aim to balance average per class pixel count per image by applying random dropout of segments where the dropout rate is linearly proportional to the size of the segments. We also apply resizing and shifting to introduce size and location variance of objects.

We performed a grid search of the parameters of PanDA to optimize PQ of UPSNet-

50 model on the full Cityscapes dataset. To demonstrate the generalizability and robustness, we **froze PanDA's parameters** and **used it throughout the rest of the paper** with the exception of the Ablation Study section. Specifically, drop out is performed as follows: segments with areas between 10% and 50% of an image are dropped out with probability linearly proportional to their sizes. Segments occupying over 50% of an image are guaranteed to drop out, and segments smaller than 10% of the image frame are never dropped out. We couple resize and shift with a zooming operation: enlarged segments are pushed to the periphery and shrunk ones are pulled towards the center. This simulates an approaching motion of the object. Each segment is resized in the range of 0.5x to 1.5x, with scales drawn from a uniform distribution.

In original Cityscapes, *road* and *building* together occupy over 50% pixels of an image on average which supports our observation that some large classes dominate learning signals. While PanDA removes a large proportion of non-background pixels on average, it significantly reduces the dominance of common large classes while largely preserving small and rare classes (Supplementary Fig. 1 and 2).

Examples of original and synthetic image pairs are shown in Fig. 4.2 (and Supplementary Fig. 4.9): some of the commonly seen classes with large area such as *road* and *building* are more frequently dropped out and smaller instances such as *person* and *pole* are often kept. It has caught our attention that the synthetic images are no longer realistic and coherent scenes but rather nonsensical to human eyes. However we show in later sections that adding these synthetic images improves model performance.

To demonstrate the usefulness of PanDA, we first trained original UPSNet and SSS models from ImageNet pre-trained ResNet backbones to establish baseline performance metrics (Table 4.1). The reproduced results largely recapitulate those published in the original papers (for details, see Supplementary Table 4.5). Then we generated synthetic training sets with PanDA and trained original UPSNet and SSS models on the augmented training sets. We first used PanDA to double the Cityscapes training set and trained the original UPSNet-50 model on the PanDA augmented Cityscapes dataset with a total of 3,000 images. The model (UPSNet-50 with PanDA in Table 4.1 and Supplementary Fig. 4.7) outperforms the baseline model (as well as the one reported in (Xiong et al., 2019)) in all metrics. In summary, UPS-50 trained on PanDA augmented images improves on the baseline by 1.1 PQ, 0.9 AP, 1.1 AP box, evaluated at 0.5:0.95) and 1.1 mean Intersection over Union

Table 4.1: **Results on Cityscapes.** PanDA augmented Cityscapes datasets generated with a single frozen set of parameters improve all performance of UPSNet-50, UPSNet-101, SSS-50. *PQ*: panoptic quality. *SQ*: segmentation quality. *RQ*: recognition quality. $PQ^{th}$: PQ for *things* classes. $PQ^{st}$: PQ for *stuff* classes. *mIoU*: mean intersection over union. *AP*: mean average precision of instance segmentation evaluated at 0.5:0.95. $AP^{box}$: mean average precision of detection evaluated at 0.5:0.95.

| Model | PanDA | PQ | SQ | RQ | $PQ^{Th}$ | $PQ^{St}$ | mIoU | AP | $AP^{Box}$ |
|---|---|---|---|---|---|---|---|---|---|
| UPSNet-50 | | 58.8 | 79.5 | 72.6 | 54.5 | 62.0 | 75.1 | 33.5 | 38.7 |
| | ✓ | **59.9** | **79.9** | **73.8** | **55.4** | **63.3** | **76.2** | **34.6** | **39.6** |
| UPSNet-101 | | 59.8 | 80.0 | 73.5 | 55.6 | 62.9 | 76.7 | 33.0 | 38.2 |
| | ✓ | **60.5** | **80.2** | **74.1** | **56.3** | **63.6** | **77.1** | **35.0** | **40.6** |
| SSS-50 | | 60.3 | – | – | 55.5 | 63.8 | 77.4 | 32.4 | 36.3 |
| | ✓ | **60.9** | – | – | **56.4** | **64.2** | **78.0** | **33.8** | **37.0** |

(mIoU) (0.7 PQ, 1.0 AP, 1.3 AP box, and 1.0 mIoU improvements over (Xiong et al., 2019)) on the validation set. Additionally, overall and per class performance in both instance segmentation task (Supplementary Table 4.6 and detection task (Supplementary Table 4.7) improves upon baseline.

One commonly used method to improve model performance at the cost of increased computation is to use a larger and deeper backbone. We trained a UPSNet-101 model which uses the deeper ResNet-101 and observed performance gain for all metrics except for AP and AP box. We froze PanDA parameters and trained UPSNet-101 model with PanDA augmented images. Table 4.1 shows that the PanDA enhanced UPSNet-101 model in turn outperforms all other UPSNet models. Two observations can be made from the additional experiments with UPSNet-101. First, it is remarkable that with a half-sized back bone, the PanDA enhanced UPSNet-50 model outperforms the baseline UPSNet-101 model in five out of eight metrics: PQ, RQ, PQ stuff, AP and AP box. Second, the fact that PanDA further improves UPSNet-101 suggests that using a deeper backbone and training on PanDA augmented images help the model in complementary ways. In other words, it demonstrates that PanDA generalizes across backbones.

In addition to UPSNet, we also conducted experiments on SSS, which is the only other model with code available to date. We trained the SSS model on PanDA augmented images with the same frozen parameters. The last row in Table 4.1 shows all-around improvements of the SSS model trained with PanDA. Together, the set of experiments with UPSNet-50/101 and SSS shows that PanDA is effective across models and backbones.

**PanDA Across Scales and Datasets**

Table 4.2: **PanDA generalization results.** We used the same set of parameters for all experiments in this section. As shown in the table, PanDA generalizes well not only across scales of Cityscapes subsets, but also to COCO subsets that are 10 times larger than the original Cityscapes dataset.

| Dataset | # images | PanDA | PQ | SQ | RQ | $PQ^{Th}$ | $PQ^{St}$ | mIoU | AP | $AP^{Box}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Cityscapes | 10 | | 23.0 | **58.1** | 29.6 | 9.2 | 33.1 | **36.3** | 2.9 | 5.5 |
| | 10 | ✓ | **23.5** | 54.9 | **30.3** | **12.5** | **30.3** | 34.3 | **4.6** | **7.6** |
| | 100 | | 37.7 | **72.5** | 48.4 | 26.4 | 45.9 | 52.4 | 10.5 | 16.7 |
| | 100 | ✓ | **40.1** | 70.6 | **51.0** | **29.5** | **47.7** | **54.1** | **12.7** | **19.1** |
| | 1,000 | | 52.7 | 77.9 | 65.8 | 45.5 | 57.9 | 68.7 | 25.8 | 31.1 |
| | 1,000 | ✓ | **55.0** | **78.4** | **68.5** | **49.1** | **59.2** | **70.1** | **27.3** | **32.5** |
| | 3,000 | | 59.3 | 79.7 | 73.0 | 54.6 | 62.7 | 75.2 | 33.3 | 39.1 |
| | 3,000 | ✓ | **59.9** | **79.9** | **73.8** | **55.4** | **63.3** | **76.2** | **34.6** | **39.6** |
| COCO | 30,000 | | 36.5 | 76.2 | 45.7 | 41.6 | **28.8** | 45.3 | 26.5 | 29.0 |
| | 30,000 | ✓ | **37.4** | **76.9** | **46.6** | **43.2** | 28.7 | **45.9** | **28.0** | **31.2** |

To test how well PanDA generalizes across scales, we applied it to smaller subsets of Cityscapes. Such generalization would be particularly useful if one develops a new panoptic task in a new domain where annotated data is expensive and scarce. In Table 4.2 (and Table 4.8), we show the consistent performance improvement on UPSNet models trained with PanDA augmented Cityscapes subsets across scales of 10, 100, 1,000, and 3,000 images.

We then ask whether the improvement with PanDA is specific to the Cityscapes dataset. Arguably, the ego-centric driving scenarios in urban environments in Cityscapes are a specialized domain which may raise concerns of generalizability across domains. In addition, Cityscapes is one of the smaller panoptic datasets available. Although previous experiments demonstrate that PanDA performs well when the dataset scales down, it remains unknown how well it performs when scaled up. To address the concerns, we applied PanDA to a 30,000 image subset of COCO dataset which is obtained by going through a list of the original 118K training set with step size 4. The COCO 30K subset will not only test whether PanDA generalizes to a different domain (examples in Supplementary Fig. 4.10), but also breaks the 3,000-image upper limit of Cityscapes. The bottom two rows of Table 4.2 show that PanDA indeed works on the 10x larger COCO dataset, which further supports the claim that PanDA generalizes well across scales and domains.

**Data Efficiency**

In this section, we explore the data efficiency of PanDA generated images. As shown in Fig. 4.3 (and Supplementary Fig. 4.8), PQ, AP, and AP box scales linearly with the logarithm of number of training images. The log-linear regression functions are,

$$PQ = 6.3255ln(n) + 8.5413, (r^2 = 0.9995) \tag{4.2}$$

$$AP = 5.4384ln(n) - 11.494, (r^2 = 0.9744) \tag{4.3}$$

$$AP^{box} = 5.8271ln(n) - 8.7893, (r^2 = 0.9948) \tag{4.4}$$

where $n$ is the number of original images used to train the model, PQ, AP, and AP box are respective model performance. PQ, AP, and AP box values are plugged into the regression functions to in turn estimate **effective training set size** $N$ for experiments with either original images only or original plus PanDA images. We define **data efficiency (DE)** as,

$$DE = \frac{N^{aug} - N^{orig.}}{N^{orig.}} \times 100\% \tag{4.5}$$

where N$^{orig.}$ is the estimated effective training set size of models trained on original images only, and N$^{aug}$ is that of models trained on original and PanDA images. The definition is specific to the case where the ratio of original and synthetic images is 1. A higher DE suggests higher per image data efficiency, and one would expect a DE of 100% if a synthetic image is as informative as an original image.

Table 4.3: **Data efficiency.** Three sets of estimates of effective training set sizes are made per experiment with PQ, AP, and AP box, respectively. # original images: number of original images used in training. $DE$: data efficiency in percent. $N$: number of effective training images estimated by model performance. Superscripts *orig.* and *aug* denote model trained with original images only or original and PanDA images. Subscripts denote which performance metric is used to estimate $N$ or $DE$.

| # orig. img | $N_{PQ}^{orig.}$ | $N_{PQ}^{aug}$ | $DE_{PQ}$ | $N_{AP}^{orig.}$ | $N_{AP}^{aug}$ | $DE_{AP}$ | $N_{APbox}^{orig.}$ | $N_{APbox}^{aug}$ | $DE_{APbox}$ |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 9.8 | 10.6 | 8.2 | 14.1 | 19.3 | 36.7 | 11.6 | 16.7 | 43.4 |
| 100 | 100.5 | 146.8 | 46.1 | 57.1 | 85.5 | 49.9 | 79.4 | 119.8 | 51.0 |
| 1000 | 1076.1 | 1547.9 | 43.9 | 951.0 | 1253.1 | 31.8 | 939.6 | 1194.8 | 27.2 |
| 3000 | 2822.6 | 3358.7 | 19.0 | 3918.2 | 4796.6 | 22.4 | 3462.4 | 4040.7 | 16.7 |

Two conclusions can be drawn from results in Table 4.3. First, across scales and metrics, PanDA images are not as efficient as original images because none of DEs are near 100%. However, this is expected since we only reuse the object instances in the original images, and the synthetic images have only 40% non-background pixels per image on average. Second, the fact that a synthetic image can be half as efficient as a real one suggests that there is significant amount of information embedded in the original images that is not learned by current models. One can expect superior models to capture this additional information without the help of data augmentation.

### Ablation Study

Table 4.4: **Ablation study on Cityscapes dataset.** Training on the original dataset for more iterations does not improve model performance. Best performance is achieved by combining dropout, resize, and shift.

| Extra Iter. | Dropout | Resize | Shift | PQ | SQ | RQ | $PQ^{Th}$ | $PQ^{St}$ | mIoU | AP | $AP^{Box}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 58.8 | 79.5 | 72.6 | 54.5 | 62.0 | 75.1 | 33.5 | 38.7 |
| ✓ | | | | 58.7 | **80.1** | 71.9 | 53.5 | 62.4 | 75.6 | 33.2 | 38.1 |
| ✓ | ✓ | | | 59.3 | 79.6 | 73.1 | 54.1 | 63.1 | 75.8 | 33.4 | 39.2 |
| ✓ | ✓ | ✓ | | 59.0 | 79.8 | 72.6 | 55.3 | 61.7 | 75.8 | 33.4 | 39.3 |
| ✓ | ✓ | | ✓ | 59.3 | 79.9 | 72.8 | 53.9 | 63.2 | 76.0 | 34.1 | 39.4 |
| ✓ | ✓ | ✓ | ✓ | **59.9** | 79.9 | **73.8** | **55.4** | **63.3** | **76.2** | **34.6** | **39.6** |

We conducted an ablation study to evaluate the effectiveness of the individual operations of PanDA. We use the reproduced UPSNet-50 model trained on the original Cityscapes training set as the baseline (first row in Table 4.4). For each experiment group, certain subsets of PanDA image operations are ablated. We optimized PanDA's parameters under the ablation constraint and report the best performance obtained. Experiment groups are trained on their respective PanDA 1x augmented Cityscapes dataset. Doubling the number of training epochs without any augmented data (second row in Table 4.4) does not improve model performance which suggests that the original UPSNet training parameters are near optimal. PanDA with dropout operation only improves model performance, presumably by mitigating the data imbalance issue (Fig. 4.5 and 4.6). Without resize or shifting, original object relationships are preserved and thus still realistic (Fig 4.4(b)). As more operations are included in PanDA, the level of realism of synthesized images decreases (Fig. 4.4(c)). However, the fact that best performance is achieved with the least realistic image set suggests that, contrary to popular belief, the level of realism of synthetic images is not necessarily correlated with data augmentation performance.

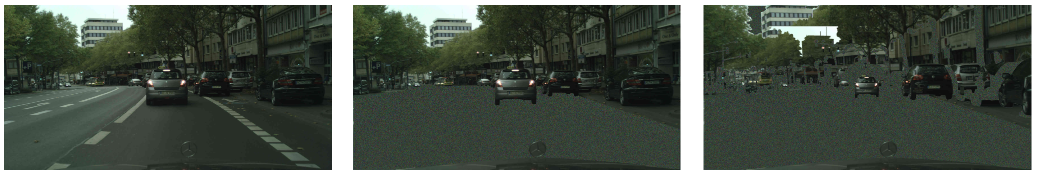


Figure 4.4: **Examples of image with decreasing levels realism.** (a) Original image. (b) Image with dropout only. The dominant segment - road is dropped out to balance per class pixel count. The composition of the picture is largely preserved and many objects still appear in realistic contexts. (c) Image with dropout, resize, and shift. In addition to dropout as seen in (b), segments are resized and moved to create new contexts. The additional operations make the new image unrealistic.

## 4.5 Discussion

Compared with GAN based and 3D simulation based data augmentation methods, PanDA has several advantages. First of all, PanDA does not require training. GAN based methods need to be trained to generate realistic images in the desired domain first before the generative network can be used for data augmentation. Second, no additional data is needed for PanDA. As mentioned before, training GANs not only takes time but also usually requires a large unlabeled dataset from the same domain. 3D simulation based methods almost always require hand crafted 3D models of classes of interest. Finally, it is computationally very cheap to use PanDA since it operates exclusively in pixel space. Many image operations such as cropping and resizing could be further parallelized and optimized, which allows for real time image synthesis at training time in an active learning fashion.

As mentioned in the Related Work section, InstaBoost (H.-S. Fang et al., 2019) relies on two operations, transform of instance segments and in-paint of background, to boost the performance of instance segmentation and detection. There are several key differences between PanDA and InstaBoost. First, in the panoptic segmentation task, a large part of the "background" in the instance segmentation task now belongs to the *stuff* superclass. The "background" in the panoptic segmentation images usually takes up less than 15% of the image, making it infeasible to use the same in-paint process. Secondly, the motivation behind in-painting the background is not explicitly explained and presumably exists to maintain the image's level of realism. In contrast, PanDA boosts panoptic segmentation performance despite losing image realism. Finally, different from the transforms in InstaBoost, PanDA included drop out operations which is shown to be essential for its effectiveness by the ablation study (Table 4.4). The method proposed by Dvornik et al. (Dvornik, Mairal, and Schmid, 2019) adds new instances to original images and avoids holes in the background from removing instances as seen in InstaBoost. However, the pasting of new instances has to be guided by a CNN-based context model to determine the location and class of instances to add, and that model has to be trained which requires data and time. Similar to InstaBoost, the method only applies to *things* classes, which take less than 50% of pixels in an image on average. Shetty et al. (R. Shetty, Schiele, and Fritz, 2019) propose removal of objects to break context information and help deep models to generalize in classification and semantic segmentation. This approach relies heavily on a trainable CNN based in-paint network to fill the holes in the background. Data and time are needed to train the in-paint CNN network in this method. Additionally, experiments in the original paper show that removing large objects such as mountains makes it hard for the in-paint network to fill the hole left behind and indeed hurts model performance. In contrast, PanDA removes the large objects like road and buildings with high probabilities, and we show that is essential for performance gain.

There are certain limitations within the current implementation of PanDA that leave doors open for more exploration. First, the functions used in background padding, dropout, resize, and shift can be further optimized or complemented with more optimized functions. For example, resizing can be drawn from a normal instead of uniform distribution. Operations such as rotation and warping can be added. Further optimizations can be performed in an automated way (Cubuk et al., 2018). Second, since the 3,000 training images in Cityscapes can be divided into 78,000 segments, if we allow the creation of hybrid images where segments in the new image can come from different original images, we could further expand the image variation combinatorially. Third, it remains unclear how many useful synthetic images can be generated with PanDA per original image. In principle, one could create an infinite number of synthetic images from the original dataset. Due to the high computation cost of training with very large datasets, this paper limits the scope to doubling the number of training images. It will be an interesting direction to investigate the limit of the number of useful synthetic images per original image.

It is worth noting that for challenges on ego-centric driving datasets like Cityscapes, it is also possible to improve model performance by pretraining on a larger dataset from a related domain. For example, better-performing models can be obtained by pretraining on MS COCO and fine tuning with Cityscapes (Xiong et al., 2019).

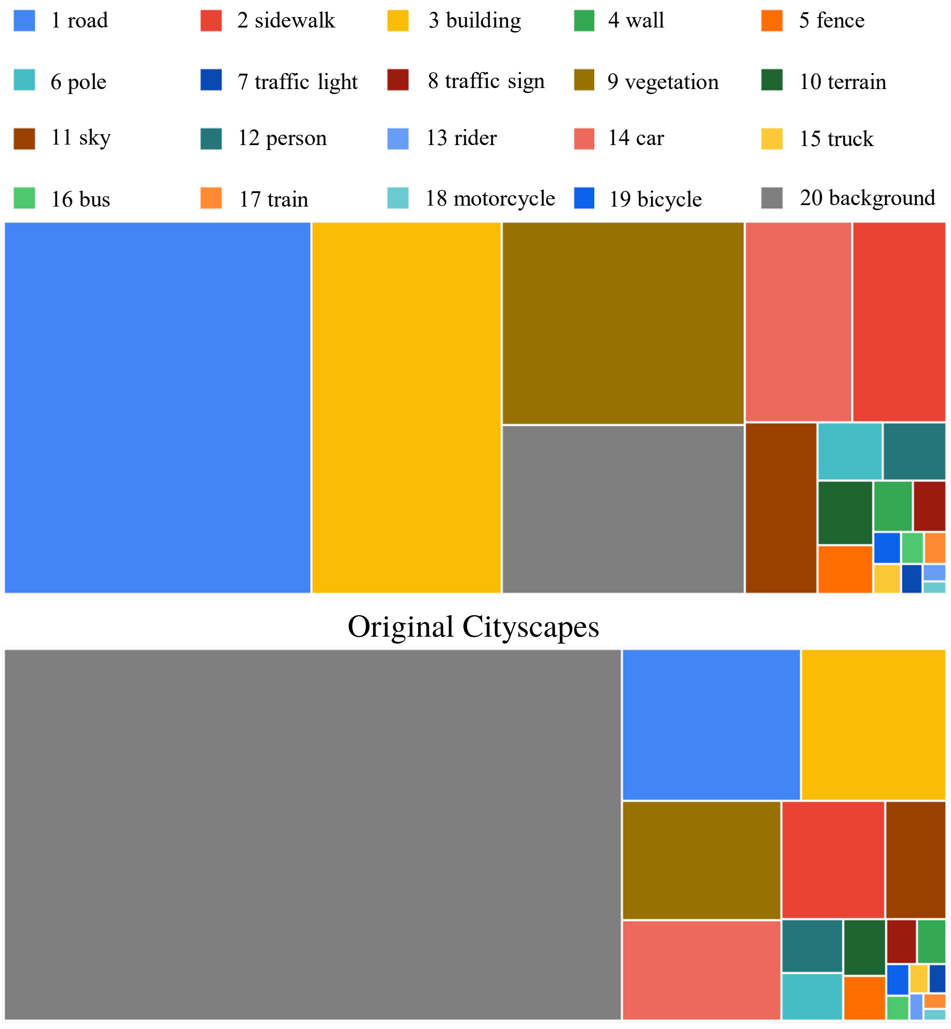
However, in a new and specialized domain, there might not exist a dataset available to pretrain on. In addition, for a large dataset like MS COCO, pretraining on smaller datasets is unlikely to help.

Finally, although PanDA operations are well justified from the standpoint of statistics, we were surprised that the best performing PanDA augmented datasets do not look natural or realistic to human eyes at all. Many PanDA generated images look qualitatively similar to the ones shown in Fig. 4.2 where positioning and occlusion of objects are not realistic. Many objects in the synthetic images appear to be “floating" on top of the noise background out of the original context (Fig. 4.2 (d), Fig. 4.4 (c)); sometimes they cluster and overlap each other in the wrong depth order (Fig. 4.2 (f)). It is known that contextual information helps human visual detection (Neider and Zelinsky, 2006). We suspect that by taking objects out of their original context, PanDA presents harder challenges to the model and therefore forces the model to pay more attention to the pixels within the object foreground. Secondly, despite the fact that PanDA drastically reduced the total non-background pixels per synthetic image, the augmented datasets are more balanced. It suggests that a balanced but small dataset might be more helpful than a large but unbalanced dataset.

## 4.6 Conclusions

We present a simple and efficient method for data augmentation of annotated panoptic images to improve panoptic segmentation performance. PanDA is computationally cheap, and requires no training or additional data. After training with PanDA augmented datasets, top performing panoptic segmentation models further improve performance on two popular datasets, Cityscapes and MS COCO. To the best of our knowledge, PanDA is the first pixel-space data augmentation method with demonstrated performance gain for leading models on panoptic segmentation tasks. Further improvement is possible with fine-tuned parameters. The effectiveness of unrealistic images suggests that we should reconsider maximizing realism in image synthesis for data augmentation. Finally, PanDA opens new opportunities for exploring efficient pixel space data augmentation methods for detection and segmentation datasets, and we believe the community would benefit from these explorations in data augmentation.

## 4.7 Supplementary Materials



Original Cityscapes

PanDA Cityscapes

Figure 4.5: **Treemap views of average pixel count by class per image of Cityscapes training set.** Class 20 is the background which is considered out of area of interest. Classes 1 and 3 together occupy the majority of the non-background pixels of an image in the original Cityscapes, and make up less than half the non-background pixels.

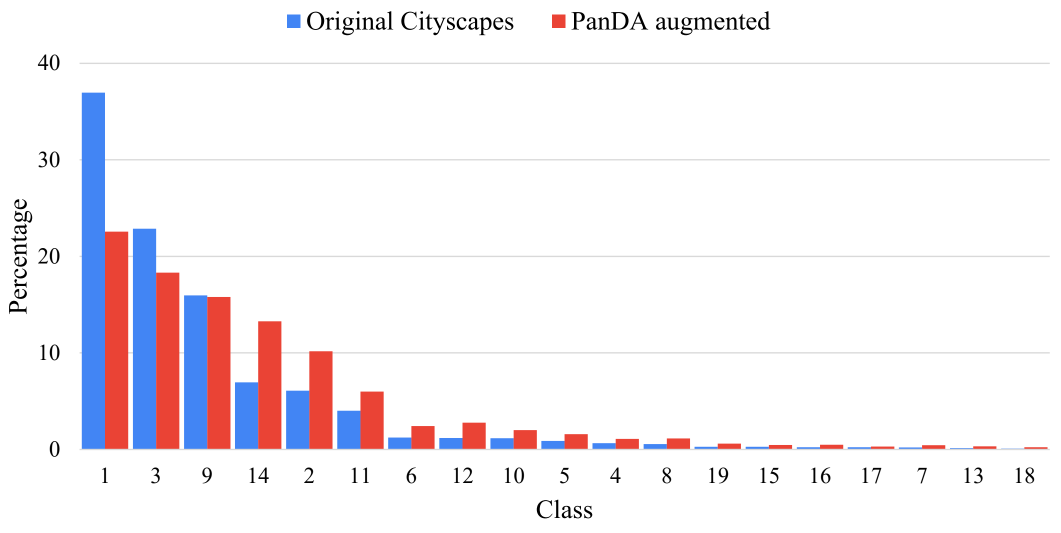


Figure 4.6: **Per class pixel percentage per image.** Bar graph of average per class pixel percentage of non-background classes per image. Each bar is computed by dividing the average number of pixels of a given class by the sum of the average number of pixels of non-background pixels. Pixel percentages of common classes are reduced and those of rare classes are increased, making the synthetic images more class-balanced.

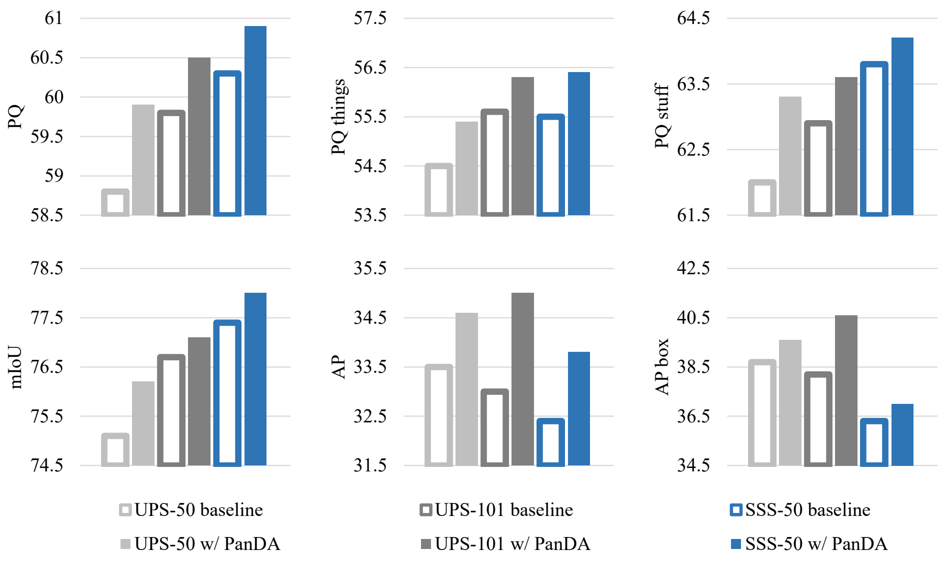


Figure 4.7: **Results on Cityscapes.** In each case, data augmentation with PanDA improves performance. *PQ*: panoptic quality. *PQ things*: PQ for *things* classes. *PQ stuff*: PQ for *stuff* classes. *mIoU*: mean intersection over union. *AP*: segmentation average precision evaluated at 0.5:0.95. $AP^{box}$: bounding box average precision evaluated at 0.5:0.95.

Table 4.5: **Extended Results on Cityscapes.** Despite the fact that our baseline performance (best of 5 runs) is lower than that reported in the original UPSNet paper Xiong et al., 2019, our models trained on PanDA augmented datasets outperform the original UPSNet-50 model without COCO pretraining in all metrics. PanDA 1x consists of 3,000 original Cityscapes images plus 3,000 PanDA synthetic images, PanDA 2x consists of 3,000 original Cityscapes images plus 6,000 PanDA images. PQ: panoptic quality. SQ: segmentation quality. RQ: recognition quality. $PQ^{th}$: PQ for *things* classes. $PQ^{st}$: PQ for *stuff* classes. *mIoU*: mean intersection over union. *AP*: segmentation average precision evaluated at 0.5:0.95. $AP^{box}$: bounding box average precision evaluated at 0.5:0.95.

| Model | PQ | SQ | RQ | $PQ^{Th}$ | $PQ^{St}$ | mIoU | AP | $AP^{Box}$ |
|---|---|---|---|---|---|---|---|---|
| UPSNet-50 Xiong et al., 2019 | 59.3 | 79.7 | 73.0 | 54.6 | 62.7 | 75.2 | 33.3 | 39.1 |
| UPSNet-50 baseline | 58.8 | 79.5 | 72.6 | 54.5 | 62.0 | 75.1 | 33.5 | 38.7 |
| UPSNet-50 w/ PanDA 1x | 59.9 | 79.9 | **73.8** | 55.4 | **63.3** | **76.2** | 34.6 | 39.6 |
| UPSNet-50 w/ PanDA 2x | **60.0** | **80.3** | 73.5 | **55.8** | 63.1 | **76.2** | **35.6** | **40.5** |
| UPSNet-101 baseline | 59.8 | 80.0 | 73.5 | 55.6 | 62.9 | 76.7 | 33.0 | 38.2 |
| UPSNet-101 w/ PanDA 1x | **60.5** | **80.2** | **74.1** | **56.3** | **63.6** | **77.1** | **35.0** | **40.6** |
| SSS-50 Porzi et al., 2019 | 60.3 | – | – | 56.1 | 63.3 | 77.5 | 33.5 | – |
| SSS-50 baseline | 60.3 | – | – | 55.5 | 63.8 | 77.4 | 32.4 | 36.3 |
| SSS-50 w/ PanDA 1x | **60.9** | – | – | **56.4** | **64.2** | **78.0** | **33.8** | **37.0** |

Table 4.6: **Per class instance segmentation results on Cityscapes.** Segmentation AP are reported. We observe not only large relative improvement on rare classes such as train and bicycle (18.4% and 13.1%, respectively), but also small gains on common classes such as car and person (3.5% and 3.5%, respectively).

| Models | person | rider | car | truck | bus | train | motorcycle | bicycle |
|---|---|---|---|---|---|---|---|---|
| UPSNet-50 Xiong et al., 2019 | 31.2 | 25.1 | 50.9 | 33.6 | 55.0 | 33.2 | 19.2 | 18.3 |
| UPSNet-50 our basline | 31.1 | 24.8 | 51.0 | 33.2 | 53.5 | 35.8 | 19.4 | 18.9 |
| UPSNet-50 w/ PanDA 1x | 31.5 | **25.7** | 52.2 | 34.3 | 54.4 | 37.9 | **20.8** | 20.3 |
| UPSNet-50 w/ PanDA 2x | **32.3** | 25.6 | **52.7** | **36.9** | **56.6** | **39.3** | 20.6 | **20.7** |

Table 4.7: **Per class detection results on Cityscapes.** AP box are reported.

| Models | person | rider | car | truck | bus | train | motorcycle | bicycle |
|---|---|---|---|---|---|---|---|---|
| UPSNet-50 Xiong et al., 2019 | 38.6 | 42.1 | 56.6 | 33.4 | 56.3 | 27.2 | **28.4** | 30.1 |
| UPSNet-50 our basline | 38.5 | 41.6 | 56.3 | 33.6 | 55.4 | 26.0 | 27.1 | 31.2 |
| UPSNet-50 w/ PanDA 1x | 39.0 | 43.4 | 57.8 | 33.6 | 55.8 | 27.2 | **28.4** | 31.2 |
| UPSNet-50 w/ PanDA 2x | **40.1** | **43.4** | **58.6** | **36.3** | **57.5** | **29.3** | 27.4 | **31.7** |

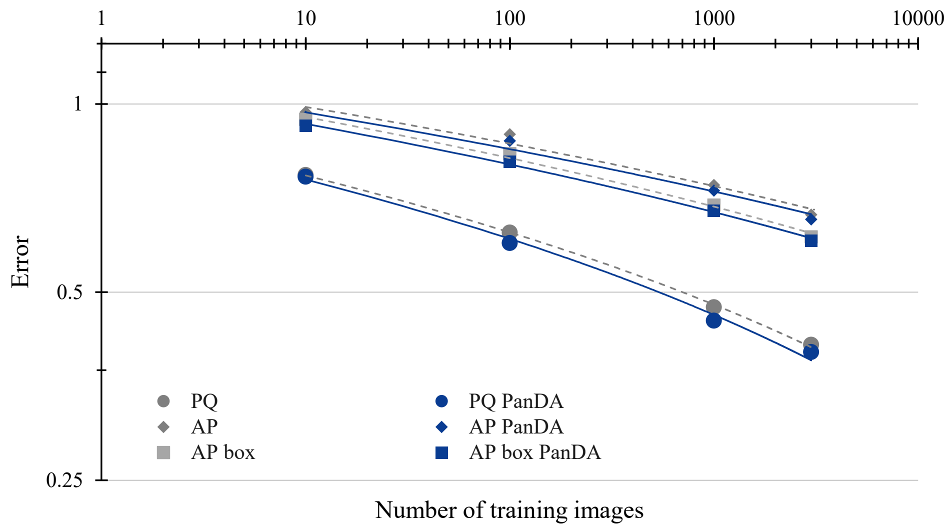


Figure 4.8: **Model performance vs training set size on Cityscapes.** We train UPSNet-50 models on various Cityscapes subsets ranging from 10 to 2,975 images. *PQ*: panoptic quality. *AP*: instance segmentation average precision evaluated at 0.5:0.95. *AP box*: bounding box average precision evaluated at 0.5:0.95. Dashed curves are log-linear fits of baseline experiments, solid curves are fits of PanDA experiments. Panoptic segmentation, instance segmentation, and instance detection performance are summarized by PQ, AP, and AP box, respectively. PanDA enhanced models consistently outperform original models across scales in all metrics. Data efficiency (DE) corresponds to the amount of right shift of PanDA curves to account for the improved performance with PanDA.

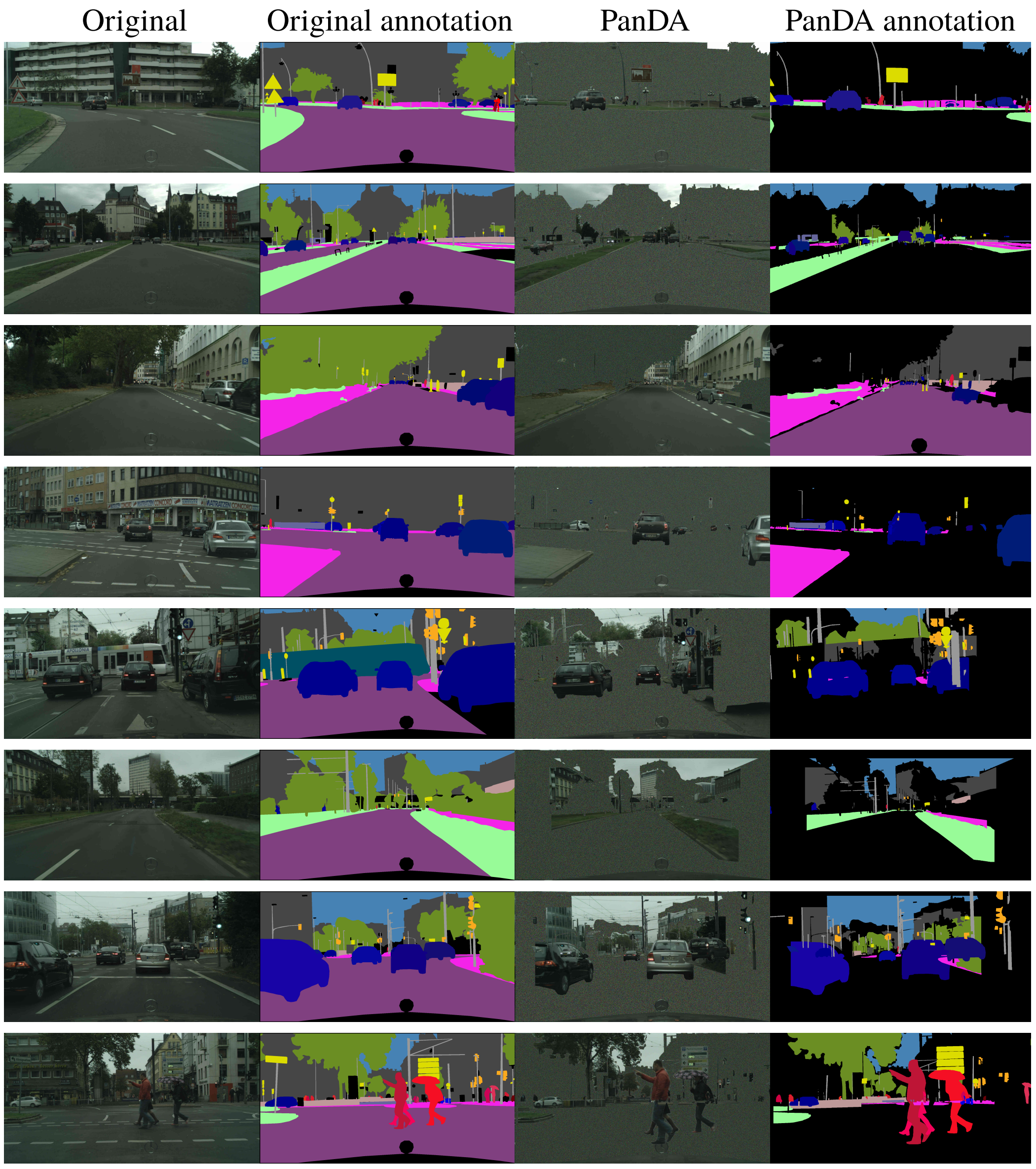


Figure 4.9: **Examples of original and PanDA Cityscapes images with ground truth annotation.** *Left two columns*: original Cityscapes images and ground truth annotation images. *Right two columns*: PanDA generated images and ground truth annotation images.

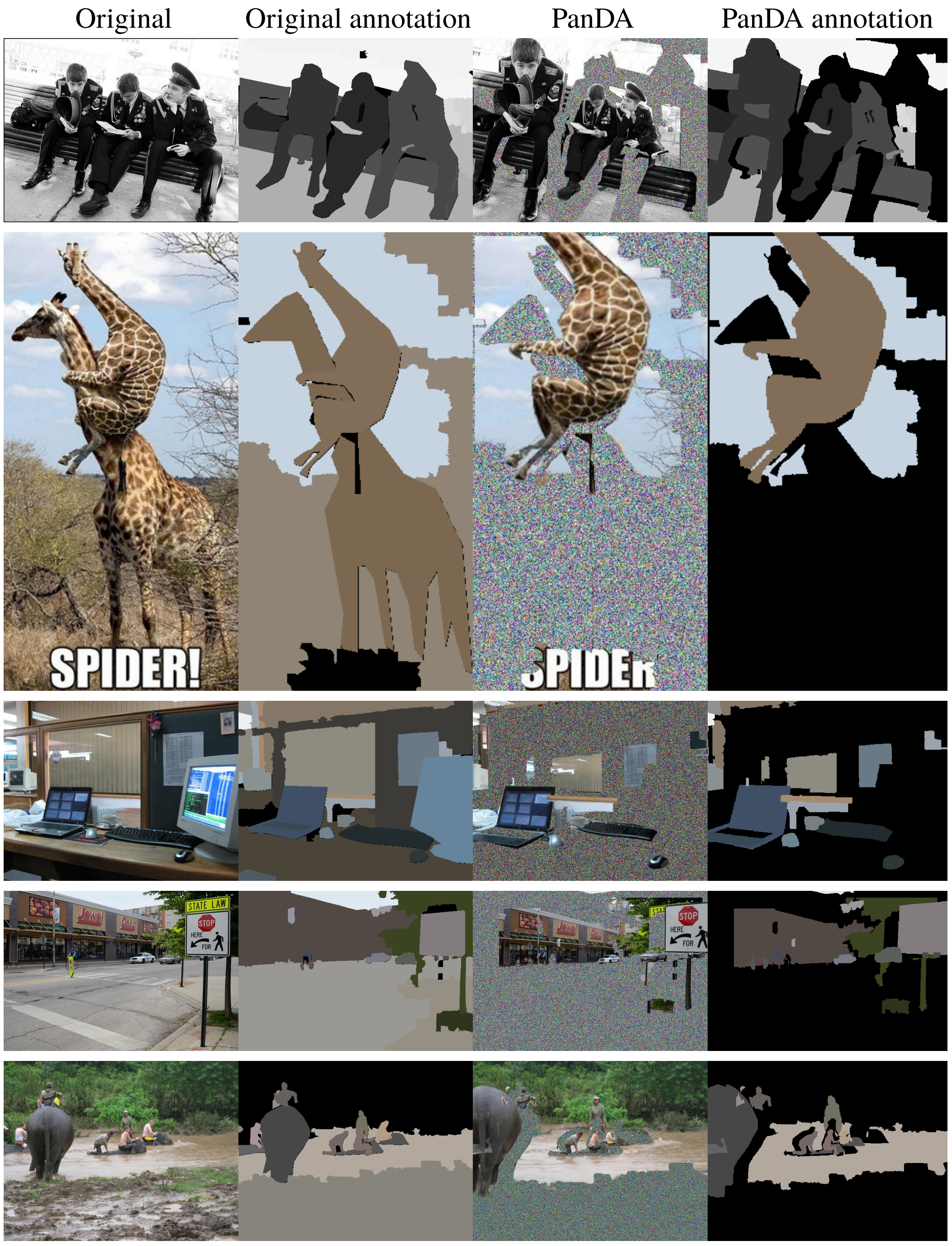


Figure 4.10: **Examples of original and PanDA COCO images with ground truth annotation.** *Left two columns*: original COCO images and ground truth annotation images. *Right two columns*: PanDA generated images and ground truth annotation images.

# BIBLIOGRAPHY

*4toon studio* (n.d.). `https://assetstore.unity.com/publishers/3695`. Accessed: 2019-03-27.

Adebiyi, Aminat et al. (2017). “Assessment of feedback modalities for wearable visual aids in blind mobility”. In: *PloS one* 12.2.

Auvray, Malika, Sylvain Hanneton, and J Kevin O’Regan (2007). “Learning to perceive with a visuo—auditory substitution system: localisation and object recognition with ‘The Voice’”. In: *Perception* 36.3, pp. 416–430.

Barth, Ruud et al. (2018). “Data synthesis methods for semantic segmentation in agriculture: A Capsicum annuum dataset”. In: *Computers and electronics in agriculture* 144, pp. 284–296.

Beery, Sara, Yang Liu, Dan Morris, Jim Piavis, Ashish Kapoor, Neel Joshi, et al. (Mar. 2020). “Synthetic Examples Improve Generalization for Rare Classes”. In: *The IEEE Winter Conference on Applications of Computer Vision (WACV)*. URL: `http://openaccess.thecvf.com/content_WACV_2020/papers/Beery_Synthetic_Examples_Improve_Generalization_for_Rare_Classes_WACV_2020_paper.pdf`.

Beery, Sara, Yang Liu, Dan Morris, Jim Piavis, Ashish Kapoor, Markus Meister, et al. (2019). “Synthetic examples improve generalization for rare classes”. In: *arXiv preprint arXiv:1904.05916*.

Beery, Sara, Grant Van Horn, and Pietro Perona (Sept. 2018). “Recognition in Terra Incognita”. In: *The European Conference on Computer Vision (ECCV)*.

Bengio, Yoshua (2012). “Practical recommendations for gradient-based training of deep architectures”. In: *Neural networks: Tricks of the trade*. Springer, pp. 437–478.

Berens, Philipp et al. (2018). “Community-based benchmarking improves spike rate inference from two-photon calcium imaging data”. In: *PLoS computational biology* 14.5, e1006157.

Bertalmio, Marcelo, Andrea L Bertozzi, and Guillermo Sapiro (2001). “Navier-stokes, fluid dynamics, and image and video inpainting”. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. IEEE, pp. I–I.

*Blender* (n.d.). `https://www.blender.org/`. Accessed: 2019-03-28.

Bondi, Elizabeth et al. (2018). “Airsim-w: A simulation environment for wildlife conservation with uavs”. In: *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*. ACM, p. 40.

*Book of the Dead Environment* (n.d.). `https://assetstore.unity.com/packages/essentials/tutorial-projects/book-of-the-dead-environment-121175`. Accessed: 2019-03-27.

Bourne, RRA et al. (2017). *Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis. Lancet Glob Health. 2017; 5 (9): e888-97.*

Bousmalis, Konstantinos et al. (2017). "Unsupervised pixel-level domain adaptation with generative adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3722–3731.

Bowles, Christopher et al. (2018). "GAN augmentation: augmenting training data using generative adversarial networks". In: *arXiv preprint arXiv:1810.10863*.

Buda, Mateusz, Atsuto Maki, and Maciej A Mazurowski (2018). "A systematic study of the class imbalance problem in convolutional neural networks". In: *Neural Networks* 106, pp. 249–259.

Bujacz, Michał and Paweł Strumiłło (2016). "Sonification: Review of auditory display solutions in electronic travel aids for the blind". In: *Archives of Acoustics* 41.3, pp. 401–414.

Capelle, Christian et al. (1998). "A real-time experimental prototype for enhancement of vision rehabilitation using auditory substitution". In: *IEEE Transactions on Biomedical Engineering* 45.10, pp. 1279–1293.

Cashman, Thomas J and Andrew W Fitzgibbon (2013). "What shape are dolphins? building 3d morphable models from 2d images". In: *IEEE transactions on pattern analysis and machine intelligence* 35.1, pp. 232–244.

Cireşan, Dan, Ueli Meier, and Jürgen Schmidhuber (2012). "Multi-column deep neural networks for image classification". In: *arXiv preprint arXiv:1202.2745*.

Collins, CC (1985). "On mobility aids for the blind, in Electronic Spatial Sensing for the Blind". In: *Warren and ER Strelow, Eds. Dordrecht, The Netherlands*, pp. 35–64.

Cordts, Marius et al. (2016). "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223.

*Coyote in a camera trap* (n.d.). `https://www.inaturalist.org/photos/7738216`. Accessed: 2019-03-28.

Csapó, Ádám and György Wersényi (2013). "Overview of auditory representations in human-machine interfaces". In: *ACM Computing Surveys (CSUR)* 46.2, pp. 1–23.

Cubuk, Ekin D et al. (2018). "Autoaugment: Learning augmentation policies from data". In: *arXiv preprint arXiv:1805.09501*.

Cui, Yin, Menglin Jia, et al. (2019). “Class-Balanced Loss Based on Effective Number of Samples”. In: *arXiv preprint arXiv:1901.05555*.

Cui, Yin, Feng Zhou, et al. (2016). “Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1153–1162.

Deng, J. et al. (2009). “ImageNet: A Large-Scale Hierarchical Image Database”. In: *CVPR09*.

Deng, Jia et al. (2009). “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, pp. 248–255.

Dobelle, William H, MG Mladejovsky, and JP Girvin (1974). “Artificial vision for the blind: electrical stimulation of visual cortex offers hope for a functional prosthesis”. In: *Science* 183.4123, pp. 440–444.

Dosovitskiy, Alexey et al. (2017). “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16.

Dvornik, Nikita, Julien Mairal, and Cordelia Schmid (2019). “On the importance of visual context for data augmentation in scene understanding”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Elkan, Charles (2001). “The foundations of cost-sensitive learning”. In: *International joint conference on artificial intelligence*. Vol. 17. 1. Lawrence Erlbaum Associates Ltd, pp. 973–978.

*Epic Studios* (n.d.). `http://epicstudios.com/`. Accessed: 2019-03-21.

Esteva, Andre et al. (2017). “Dermatologist-level classification of skin cancer with deep neural networks”. In: *Nature* 542.7639, p. 115.

Evans, Timothy (1989). “The impact of permanent disability on rural households: river blindness in Guinea”. In: *IDS bulletin* 20.2, pp. 41–48.

Everingham, Mark et al. (2010). “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88.2, pp. 303–338.

Fang, Hao-Shu et al. (2019). “Instaboost: Boosting instance segmentation via probability map guided copy-pasting”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 682–691.

Flaxman, Seth R et al. (2017). “Global causes of blindness and distance vision impairment 1990–2020: a systematic review and meta-analysis”. In: *The Lancet Global Health* 5.12, e1221–e1234.

*Forest Animals by GiM* (n.d.). `https://www.unrealengine.com/marketplace/en-US/animals-vol-01-forest-animals`. Accessed: 2019-03-21.

Frick, Kevin D et al. (2007). “Economic impact of visual impairment and blindness in the United States”. In: *Archives of ophthalmology* 125.4, pp. 544–550.

Frid-Adar, Maayan et al. (2018). “GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification”. In: *Neurocomputing* 321, pp. 321–331.

Gaidon, Adrien et al. (2016). “Virtual worlds as proxy for multi-object tracking analysis”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4340–4349.

Ganin, Yaroslav and Victor Lempitsky (2015). “Unsupervised domain adaptation by backpropagation”. In: *International Conference on Machine Learning*, pp. 1180–1189.

Gao, Naiyu et al. (2019). “SSAP: Single-Shot Instance Segmentation With Affinity Pyramid”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 642–651.

*GiM Studio* (n.d.). `https://assetstore.unity.com/publishers/18347`. Accessed: 2019-03-27.

Goodale, Melvyn A, A David Milner, et al. (1992). “Separate visual pathways for perception and action”. In:

Goodfellow, Ian et al. (2014). “Generative adversarial nets”. In: *Advances in neural information processing systems*, pp. 2672–2680.

Gregor, Karol et al. (2015). “Draw: A recurrent neural network for image generation”. In: *arXiv preprint arXiv:1502.04623*.

Han, Sanghui et al. (2017). “Efficient generation of image chips for training deep learning algorithms”. In: *Automatic Target Recognition XXVII*. Vol. 10202. International Society for Optics and Photonics, p. 1020203.

Hariharan, Bharath and Ross Girshick (2017). “Low-shot visual recognition by shrinking and hallucinating features”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3018–3027.

Hattori, Hironori et al. (2015). “Learning scene-specific pedestrian detectors without real data”. In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, pp. 3819–3827.

He, Haibo, Yang Bai, et al. (2008). “ADASYN: Adaptive synthetic sampling approach for imbalanced learning”. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, pp. 1322–1328.

He, Haibo and Edwardo A Garcia (2008). “Learning from imbalanced data”. In: *IEEE Transactions on Knowledge & Data Engineering* 9, pp. 1263–1284.

He, Kaiming, Georgia Gkioxari, et al. (2017). “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969.

He, Kaiming, Xiangyu Zhang, et al. (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

– (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hinterstoisser, Stefan et al. (2019). *An Annotation Saved is an Annotation Earned: Using Fully Synthetic Training for Object Instance Detection*. arXiv: 1902.09967 [cs.CV].

Hoffman, Judy et al. (2016). "Fcns in the wild: Pixel-level adversarial and constraint-based adaptation". In: *arXiv preprint arXiv:1612.02649*.

Howard, Andrew G (2013). "Some improvements on deep convolutional neural network based image classification". In: *arXiv preprint arXiv:1312.5402*.

*HTC Vive* (n.d.). https://en.wikipedia.org/wiki/HTC_Vive. Accessed: 2018-08-21.

Huang, Jonathan et al. (2017). "Speed/accuracy trade-offs for modern convolutional object detectors". In: *IEEE CVPR*.

Huang, Rui et al. (2017). "Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2439–2448.

Im, Daniel Jiwoong et al. (2016). "Generating images with recurrent adversarial networks". In: *arXiv preprint arXiv:1602.05110*.

Jafri, Rabia et al. (2014). "Computer vision-based object recognition for the visually impaired in an indoors environment: a survey". In: *The Visual Computer* 30.11, pp. 1197–1222.

*Janpec* (n.d.). https://assetstore.unity.com/publishers/1066. Accessed: 2019-03-27.

Ji, Shunping et al. (2019). "Building Instance Change Detection from Large-Scale Aerial Images using Convolutional Neural Networks and Simulated Samples". In: *Remote Sensing* 11.11, p. 1343.

Kanazawa, Angjoo et al. (2018). "Learning category-specific mesh reconstruction from image collections". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 371–386.

Kandel, Eric R et al. (2000). *Principles of neural science*. Vol. 4. McGraw-hill New York.

Kirillov, Alexander et al. (2019). "Panoptic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9404–9413.

Kolve, Eric et al. (2017). "AI2-THOR: An Interactive 3D Environment for Visual AI". In: *CoRR* abs/1712.05474. URL: http://arxiv.org/abs/1712.05474.

Krasin, Ivan et al. (2017). “OpenImages: A public dataset for large-scale multi-label and multi-class image classification.” In: URL: `%5Curl%7Bhttps://github.com/openimages%7D`.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*, pp. 1097–1105.

Kumar, Neeraj et al. (Oct. 2012). “Leafsnap: A Computer Vision System for Automatic Plant Species Identification”. In: *The 12th European Conference on Computer Vision (ECCV)*.

Lacey, Simon and Rebecca Lawson (2013). *Multisensory imagery*. Springer Science & Business Media.

Lee, Ki Bum, Sejune Cheon, and Chang Ouk Kim (2017). “A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes”. In: *IEEE Transactions on Semiconductor Manufacturing* 30.2, pp. 135–142.

Li, Fei-Fei, Rob Fergus, and Pietro Perona (2006). “One-shot learning of object categories”. In: *IEEE transactions on pattern analysis and machine intelligence* 28.4, pp. 594–611.

Lin, Tsung-Yi, Priyal Goyal, et al. (2018). “Focal loss for dense object detection”. In: *IEEE transactions on pattern analysis and machine intelligence*.

Lin, Tsung-Yi, Michael Maire, et al. (2014). “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer, pp. 740–755.

Litjens, Geert et al. (2017). “A survey on deep learning in medical image analysis”. In: *Medical image analysis* 42, pp. 60–88.

Liu, Shuangting et al. (2019). “Pixel Level Data Augmentation for Semantic Image Segmentation using Generative Adversarial Networks”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 1902–1906.

Liu, Yang, Haiwei Dong, et al. (2018). “Technical evaluation of HoloLens for multimedia: a first look”. In: *IEEE MultiMedia* 25.4, pp. 8–18.

Liu, Yang and Markus Meister (2018). *CARA*. URL: `https://github.com/meisterlabcaltech/CARA_Public`.

Liu, Yang, Pietro Perona, and Markus Meister (2019). “PanDA: Panoptic Data Augmentation”. In: *arXiv preprint arXiv:1911.12317*. URL: `https://arxiv.org/abs/1911.12317`.

Loomis, Jack M, Roberta L Klatzky, and Nicholas A Giudice (2012). “-Sensory Substitution of Vision: Importance of Perceptual and Cognitive Processing”. In: *Assistive technology for blindness and low vision*. CRC Press, pp. 179–210.

Luan, Fujun et al. (2017). “Deep photo style transfer”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4990–4998.

Luo, Yvonne Hsu-Lin and Lyndon Da Cruz (2016). “The Argus® II retinal prosthesis system”. In: *Progress in retinal and eye research* 50, pp. 89–107.

Maaten, Laurens van der and Geoffrey Hinton (2008). “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov, pp. 2579–2605.

Maidenbaum, Shachar, Sami Abboud, and Amir Amedi (2014). “Sensory substitution: closing the gap between basic research and widespread practical visual rehabilitation”. In: *Neuroscience & Biobehavioral Reviews* 41, pp. 3–15.

Marr, David (1982). “Vision: A computational investigation into the human representation and processing of visual information, henry holt and co”. In: *Inc., New York, NY* 2.4.2.

*Maya* (n.d.). `https://www.autodesk.com/products/maya/overview`. Accessed: 2019-03-28.

McAnally, Ken I and Russell L Martin (2014). “Sound localization with head movement: implications for 3-d audio displays”. In: *Frontiers in neuroscience* 8, p. 210.

Meijer, Peter BL (1992). “An experimental system for auditory image representations”. In: *IEEE transactions on biomedical engineering* 39.2, pp. 112–121.

Neider, Mark B and Gregory J Zelinsky (2006). “Scene context guides eye movements during visual search”. In: *Vision research* 46.5, pp. 614–621.

Neuhold, Gerhard et al. (2017). “The mapillary vistas dataset for semantic understanding of street scenes”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4990–4999.

Norouzzadeh, Mohammed Sadegh et al. (2017). “Automatically identifying wild animals in camera trap images with deep learning”. In: *arXiv:1703.05830*.

Pahde, Frederik et al. (2019). “Low-Shot Learning from Imaginary 3D Model”. In: *arXiv preprint arXiv:1901.01868*.

Peng, Xingchao et al. (2018). “Syn2real: A new benchmark forsynthetic-to-real visual domain adaptation”. In: *arXiv preprint arXiv:1806.09755*.

Pepik, Bojan et al. (2015). “What is holding back convnets for detection?” In: *German Conference on Pattern Recognition*. Springer, pp. 517–528.

Pitkow, Xaq and Markus Meister (2014). “Neural Computation in Sensory Systems”. In: *The Cognitive Neurosciences*, p. 305.

Poh, Ming-Zher, Daniel J McDuff, and Rosalind W Picard (2010). “Non-contact, automated cardiac pulse measurements using video imaging and blind source separation.” In: *Optics express* 18.10, pp. 10762–10774.

Poplin, Ryan et al. (2018). “Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning”. In: *Nature Biomedical Engineering*, p. 1.

Porzi, Lorenzo et al. (2019). “Seamless scene segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8277–8286.

*Protofactor Inc* (n.d.). `https://assetstore.unity.com/publishers/265`. Accessed: 2019-03-27.

Proulx, Michael J et al. (2016). “Other ways of seeing: From behavior to neural mechanisms in the online “visual” control of action with sensory substitution”. In: *Restorative neurology and neuroscience* 34.1, pp. 29–44.

*Quixel Megascans Library* (n.d.). `https://quixel.com/megascans`. Accessed: 2019-03-21.

Radford, Alec, Luke Metz, and Soumith Chintala (2015). “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434*.

Rajpura, Param S., Hristo Bojinov, and Ravi S. Hegde (2017). *Object Detection Using Deep CNNs Trained on Synthetic Images*. arXiv: `1706.06782 [cs.CV]`.

*Red Deer Studio* (n.d.). `https://assetstore.unity.com/publishers/12623`. Accessed: 2019-03-27.

Redmon, Joseph and Ali Farhadi (2018). “Yolov3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767*.

Reinert, Bernhard, Tobias Ritschel, and Hans-Peter Seidel (2016). “Animated 3D Creatures from Single-view Video by Skeletal Sketching.” In: *Graphics Interface*, pp. 133–141.

Richter, Stephan R. et al. (2016a). “Playing for Data: Ground Truth from Computer Games”. In: *Lecture Notes in Computer Science*, pp. 102–118. ISSN: 1611-3349. DOI: `10.1007/978-3-319-46475-6_7`. URL: `http://dx.doi.org/10.1007/978-3-319-46475-6_7`.

– (2016b). “Playing for data: Ground truth from computer games”. In: *European conference on computer vision*. Springer, pp. 102–118.

Ros, German et al. (2016). “The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3234–3243.

Russakovsky, O et al. (2015). “Imagenet large scale visual recognition challenge IJCV”. In: *arXiv preprint arXiv:1409.0575*.

Russakovsky, Olga et al. (2014). *ImageNet Large Scale Visual Recognition Challenge*. arXiv: `1409.0575 [cs.CV]`.

Savva, Manolis et al. (2017). “MINOS: Multimodal Indoor Simulator for Navigation in Complex Environments”. In: *arXiv:1712.03931*.

Scholl, Hendrik PN et al. (2016). “Emerging therapies for inherited retinal degeneration”. In: *Science translational medicine* 8.368, 368rv6–368rv6.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin (2015). “Facenet: A unified embedding for face recognition and clustering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823.

*Seeing with Sound* (n.d.). `https://www.seeingwithsound.com/`. Accessed: 2018-08-21.

Shah, Shital et al. (2018). “Airsim: High-fidelity visual and physical simulation for autonomous vehicles”. In: *Field and service robotics*. Springer, pp. 621–635.

Shetty, Rakshith R, Mario Fritz, and Bernt Schiele (2018). “Adversarial scene editing: Automatic object removal from weak supervision”. In: *Advances in Neural Information Processing Systems*, pp. 7706–7716.

Shetty, Rakshith, Bernt Schiele, and Mario Fritz (2019). “Not Using the Car to See the Sidewalk–Quantifying and Controlling the Effects of Context in Classification and Segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8218–8226.

Shrivastava, Ashish et al. (July 2017). “Learning From Simulated and Unsupervised Images Through Adversarial Training”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Song, Shuran et al. (2017). “Semantic Scene Completion from a Single Depth Image”. In: *IEEE Conference on Computer Vision and Pattern Recognition*.

Souza12, César Roberto de et al. (2017). “Procedural generation of videos to train deep action recognition networks”. In:

*SpeedTree* (n.d.). `https://store.speedtree.com/`. Accessed: 2019-03-21.

Stingl, K and E Zrenner (2013). “Electronic approaches to restitute vision in patients with neurodegenerative diseases of the retina”. In: *Ophthalmic research* 50.4, pp. 215–220.

Stingl, Katarina et al. (2017). “Interim results of a multicenter trial with the new electronic subretinal implant alpha AMS in 15 patients blind from inherited retinal degenerations”. In: *Frontiers in neuroscience* 11, p. 445.

Striem-Amit, Ella, Miriam Guendelman, and Amir Amedi (2012). “‘Visual’acuity of the congenitally blind using visual-to-auditory sensory substitution”. In: *PloS one* 7.3.

Stronks, H Christiaan et al. (2016). “Visual task performance in the blind with the BrainPort V100 Vision Aid”. In: *Expert review of medical devices* 13.10, pp. 919–931.

Sudol, Jeremi et al. (2010). “Looktel—A comprehensive platform for computer-aided visual assistance”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*. IEEE, pp. 73–80.

Sukhbaatar, Sainbayar et al. (2014). *Training Convolutional Networks with Noisy Labels*. arXiv: `1406.2080 [cs.CV]`.

Szegedy, Christian, Sergey Ioffe, et al. (2017). “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Thirty-First AAAI Conference on Artificial Intelligence*.

Szegedy, Christian, Vincent Vanhoucke, et al. (2016). “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.

Tieleman, Tijmen and Geoffrey Hinton (2012). “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2, pp. 26–31.

Tran, Toan et al. (2017). “A Bayesian data augmentation approach for learning deep models”. In: *Advances in Neural Information Processing Systems*, pp. 2797–2806.

*Unity Book of the Dead* (n.d.). `https://unity3d.com/book-of-the-dead`. Accessed: 2019-03-21.

*Unity Game Engine* (n.d.). `https://unity3d.com/`. Accessed: 2019-02-05.

*UNREAL Game Engine* (n.d.). `https://www.unrealengine.com/en-US/what-is-unreal-engine-4`. Accessed: 2019-02-05.

Van Horn, Grant, Oisin Mac Aodha, et al. (2017). “The iNaturalist Challenge 2017 Dataset”. In: *arXiv preprint arXiv:1707.06642*.

Van Horn, Grant and Pietro Perona (2017). “The Devil is in the Tails: Fine-grained Classification in the Wild”. In: *arXiv preprint arXiv:1709.01450*.

van Horn, Grant et al. (2017). *The Merlin Bird ID smartphone app*. URL: `%5Curl%7Bhttp://merlin.allaboutbirds.org/download/%7D` (visited on 2017).

Varol, Gul, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, et al. (July 2017a). “Learning From Synthetic Humans”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

– (2017b). “Learning from synthetic humans”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 109–117.

Verschae, Rodrigo and Javier Ruiz-del-Solar (2015). “Object detection: current and future directions”. In: *Frontiers in Robotics and AI* 2, p. 29.

*Vuforia* (n.d.). `https://www.vuforia.com/`. Accessed: 2018-08-21.

Wang, Yu-Xiong and Martial Hebert (2016). “Learning to learn: Model regression networks for easy small sample learning”. In: *European Conference on Computer Vision*. Springer, pp. 616–634.

*WDallgraphics studio* (n.d.). `https://assetstore.unity.com/publishers/5060`. Accessed: 2019-03-28.

Weiland, James D and Mark S Humayun (2014). “Retinal prosthesis”. In: *IEEE Transactions on Biomedical Engineering* 61.5, pp. 1412–1424.

Wenzel, Elizabeth M et al. (1993). “Localization using nonindividualized head-related transfer functions”. In: *The Journal of the Acoustical Society of America* 94.1, pp. 111–123.

*Wolf in a camera trap* (n.d.). `https://3c1703fe8d.site.internapcdn.net/newman/csz/news/800/2018/cameratrapst.jpg`. Accessed: 2019-03-28.

Wu, Bichen et al. (2017). “Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 129–137.

Wu, Yi et al. (2018). “Building generalizable agents with a realistic and rich 3D environment”. In: *arXiv preprint arXiv:1801.02209*.

Xiong, Yuwen et al. (2019). “Upsnet: A unified panoptic segmentation network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8818–8826.

Zhang, Chiyuan et al. (2016). *Understanding deep learning requires rethinking generalization*. arXiv: `1611.03530 [cs.LG]`.

Zhou, Bolei et al. (2017). “Scene parsing through ade20k dataset”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 633–641.

Zhu, Jun-Yan et al. (2017). “Unpaired image-to-image translation using cycle-consistent adversarial networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2223–2232.

Zou, Yang et al. (2018). “Unsupervised domain adaptation for semantic segmentation via class-balanced self-training”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 289–305.