UNIVERSITY OF CALIFORNIA,
IRVINE

Long-Term Tracking by Decision Making

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

James Supančič, III

Dissertation Committee:
Deva Ramanan, Chair
Charless Fowlkes
Alexander Ihler

2017

DEDICATION

To my family

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Foremost, I thank my adviser, Prof. Deva Ramanan. I am immensely grateful for his support and patience. He helped me understand the big picture, how individual approaches to computer vision relate with each other. From him, I learned to approach problems scientifically and to appreciate the value of theory and writing. Wherever you find my dissertation anything but incomprehensible, you have him to thank.

I also thank the many great teachers I've had throughout the years. In particular, I'm grateful to Profs. Charless Fowlkes and Alexander Ihler for guiding my research. I'm very grateful to Gregory Rogez, for his crucial guidance and collaboration. I also thank my high-school math & science teacher, Scott Bucher, and my art & music instructor, Joshua Rivera.

Next, I thank my friends and lab mates. I'm grateful to Yi, Dennis, Songfan, Xiangxin, Hamed, Chaitanya, and Golnaz for their mentorship. Thank you to Sam, Bailey, Shaofei, Carl, Maryam, Peiyun, Mohsen, Julian, Raul and Phuc for your friendship and company during the nights and weekends. And, I'm grateful to Shu, Minhaeng, and Zhe for bringing new ideas and energy to our lab.

Finally, I thank my family. First, I thank my grandparents for their love, support and inspiration. I give thanks to my parents for their support, love, and for pushing me when I felt like quitting. And to my little brothers, thank you for entertaining and distracting me, only when it was appropriate.

Thanks to Lars Otten for providing the LaTeX template for this thesis [163].

Funding for this research was provided by NSF Grant 0954083, ONR-MURI Grant N00014-10-1-0933, the Intel Science and Technology Center - Visual Computing, and he European Commission FP7 Marie Curie IOF grant "Egovision4Health" (PIOF-GA-2012-328288).

CURRICULUM VITAE

James Supančič, III

EDUCATION

Doctor of Philosophy in Computer Science	**2017**
University of California, Irvine	*Irvine, California*
Master of Science in Computer Science	**2015**
University of California, Irvine	*Irvine, California*
Bachelor of Science in Information and Computer Science	**2012**
University of California, Irvine	*Irvine, California*

RESEARCH EXPERIENCE

Graduate Research Assistant	**2012–2017**
University of California, Irvine	*Irvine, California*
Research Associate	**2016**
Disney Research	*Pittsburgh, Pennsylvania*
Research Intern	**2015**
Microsoft Research	*Redmond, Washington*

TEACHING EXPERIENCE

Teaching Assistant	**2013–2014**
University Of California, Irvine	*Irvine, California*

REFEREED JOURNAL PUBLICATIONS

Depth-based hand pose estimation: data, methods, and challenges **2017**
International Journal of Computer Vision

REFEREED CONFERENCE PUBLICATIONS

Self Paced Learning for Long-Term Tracking **2013**
Computer Vision and Pattern Recognition

First-Person Pose Recognition using Egocentric Workspaces **2015**
Computer Vision and Pattern Recognition

Understanding Everyday Hands in Action from RGB-D Images **2015**
International Conference on Computer Vision

Depth-based hand pose estimation: data, methods, and challenges **2015**
International Conference on Computer Vision

Tracking as Online Decision-Making: Learning a Policy from Streaming Videos with Reinforcement Learning **2017**
International Conference on Computer Vision

Why Can't We All Just Get Along? Effective Identity-Aware Multi-Object Tracking from Sensored Annotation? **2017**
International Conference on Computer Vision

SOFTWARE

Deep Hand Pose `https://github.com/jsupancic/deep_hand_pose`
A re-implementation of several deep learning based hand-pose estimation systems.

SPLTT `https://github.com/jsupancic/SPLTT-Release`
An implementation for our Self-Paced Learning Long-Term Tracker.

libhand-public `https://github.com/jsupancic/libhand-public`
A hand renderer for generating synthetic depth images.

ABSTRACT OF THE DISSERTATION

Long-Term Tracking by Decision Making

By

James Supančič, III

Doctor of Philosophy in Computer Science

University of California, Irvine, 2017

Deva Ramanan, Chair

Cameras can naturally capture sequences of images, or videos, and for computers to understand videos, they must track to connect the past with the present. We focus on two problems which challenge current state-of-the-art trackers. First, we address the challenge of long-term occlusion. For this challenge, a tracker must know when it has lost track and how to reinitialize tracking when the target reappears. We tackle reinitialization by building good appearance models for humans and hands, with a particular emphasis on robustness and occlusion. For the second challenge, appearance variation, the tracker must know when and how to re-learn (or update) an appearance model. Common solutions to this challenge encounter the classic problem of drift: aggressively learning putative appearance changes allows small errors to compound, as elements of the background environment pollute the appearance model. We propose two solutions. First, we consider self-paced learning, wherein a tracker begins by learning from frames it finds easy. As the tracker becomes better at recognizing the target, it begins to learn from harder frames. We also develop a data-driven approach in which we train a tracking policy to decide when and how to update an appearance model. To take this direct approach to learning when to learn, we exploit large-scale Internet data through reinforcement learning. We interpret the resulting policy and conclude with extensions for tracking multiple objects. By solving these tracking challenges, we advance applications in augmented reality, vehicle automation, healthcare, and security.

Chapter 1

Decisions in Long-Term Tracking

James Steven Supančič III

Tracking poses an important problem in computer vision; trackers estimate their targets' latent states (which may represent location, pose, appearance, etc.) over time, by observing video frames. Tracking is a fundamental task; we must track objects over time to understand how the past connects to the present. But in general, a vast number of latent states exist. That is, one state exists for every combination of position, pose, appearance, etc. Thus, computers cannot calculate (or even represent) the probability of every possible latent state. Therefore, practical state-of-the-art trackers only evaluate a finite number of candidate states [240, 241, 135, 129, 154, 54, 242, 98, 168, 216, 121, 249, 131]. Such trackers *decide* which states to explicitly compute, and which to omit.

Reinitialization and updates: This dissertation focuses on two such decisions. First, trackers must decide which locations they'll evaluate. Generally trackers evaluate locations near the target's most likely location in the previous frame. But in cases of long-term occlusion, the target may travel a great distance unseen. So, trackers sometimes prefer to

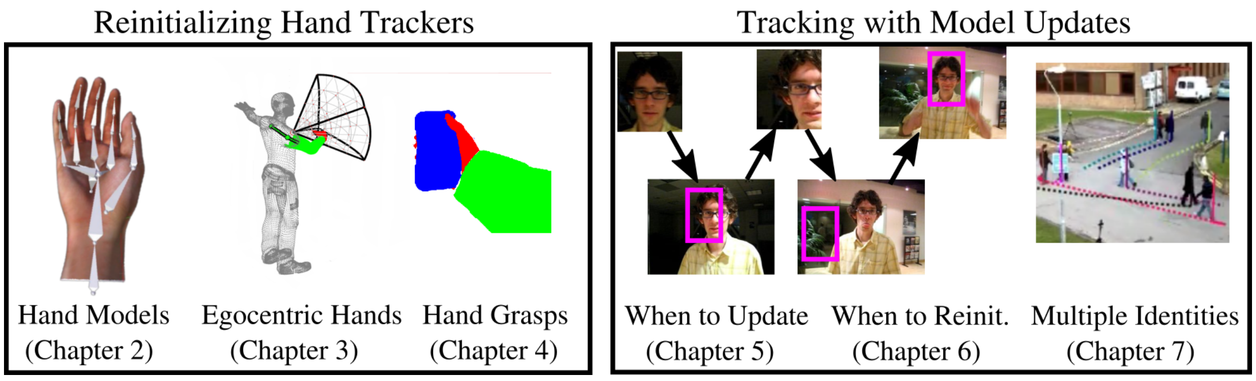

Reinitializing Hand Trackers
Tracking with Model Updates
Hand Models (Chapter 2)
Egocentric Hands (Chapter 3)
Hand Grasps (Chapter 4)
When to Update (Chapter 5)
When to Reinit. (Chapter 6)
Multiple Identities (Chapter 7)

Figure 1.1: **Dissertation overview:** We first focus on tracker reinitialization, using hand-pose estimation as an illustrative example (Chapter 2). Previous work focused on hand gesture recognition in clean scenes; we examine harder settings, where the hand may be occluded by clutter (Chapter 3) or held objects (Chapter 4). Second, we tackle tracking when appearance models must be updated at test time (Chapter 5). By formulating tracking as a decision making process, we develop a theoretical framework for developing (and training, from data) tracking algorithms which update at test time (Chapter 6). Finally, we use sensored annotation to collect large & detailed tracking data. Our new sensored data allows us to study and advance identity-aware multiple object tracking (Chapter 7).

evaluate locations globally distributed across the entire frame. We refer to this decision, that is when to globally evaluate the appearance model, as *reinitialization*. Second, trackers cannot compute (or represent) the likelihood of every possible target appearance model. So, they must decide how to *update* their appearance model, while tracking.

Organization: Beginning with a good appearance model makes these decisions easier. For example, a good appearance model tells the tracker when something occludes the target. When the target is occluded, the tracker should reinitialize. So, we first focus on good models for reinitialization, using hand tracking as our illustrative example. In subsequent chapters, we'll consider how to update appearance models. Fig. 1.1 gives an overview of this dissertation.

Tracker reinitialization: Tracking systems exploit consistency in their target's states. An airplane cannot fly faster than a certain speed; a person can only articulate their hand

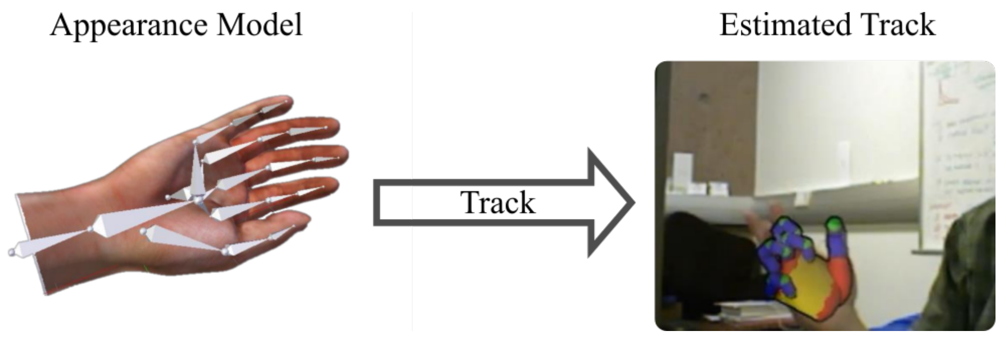


Figure 1.2: **Appearance model based tracking:** A tracker must infer its target's true, but latent, state (location, pose, appearance etc.). To update its previous estimate of the target's latent state, the tracker must interpret observations (i.e., video frames) using an appearance model; in fact, a tracker cannot interpret new observations without an appearance model. The importance of building better appearance models will be a recurring theme of this dissertation.

joints so fast. But, we cannot always depend on consistency with the previous state. Long-term occlusions present a major challenge in that we cannot reliably estimate the target's state while it's occluded. So, when a target first reappears, we must rely more on the current frame than our uncertain estimate of the target's previous state. Consider tracking the pose of a human hand which becomes occluded. Because hand pose changes rapidly and frequently, the pose before the occlusion might not provide good evidence for the pose after the occlusion. Instead, even when tracking, good single-frame appearance models are crucial because they rapidly identify occlusions and re-appearances (Fig. 1.2).

Single-frame hand tracker reinitialization: Recent model-based hand trackers have demonstrated impressive results [160]. However, they almost never recover after occlusions. In Chapter 2 we survey the state of single-frame hand-pose estimation. We found that, even for single-frame hand-pose estimation, almost every dataset and method focused on uncluttered scenes where the hand was the front-most object. We introduce new and more challenging test data, evaluate existing and new methods on our new data, and discuss the most salient conclusions. In particular, we discovered that prior work limited itself to

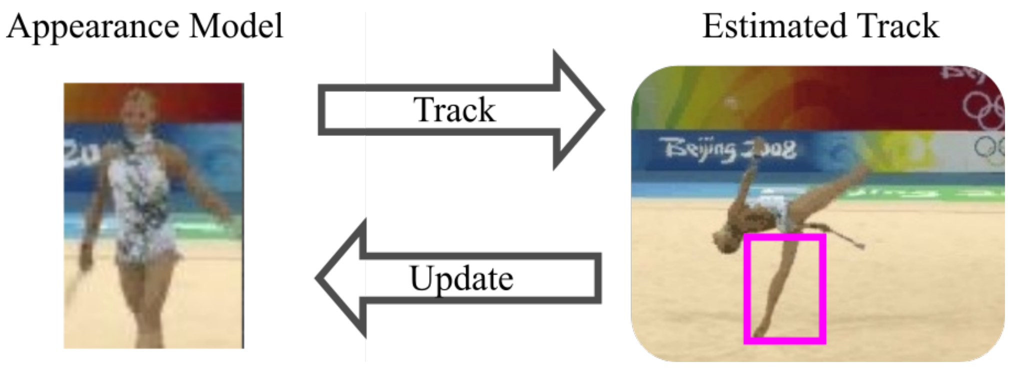


Figure 1.3: **Tracking and Updating:** For many tracking tasks, it becomes necessary for the tracker to update its appearance model at test time. But when updating a test time, a tracker does not have access to ground-truth. Appearance updates during tracking failures inevitably lead to further failures. To mitigate this, a tracker can *decide* if it has lost track (i.e., failed) and conclude how to update its appearance model.

uncluttered 3rd person scenes. To advance the field of hand-pose estimation, we posit a need for our new & more challenging datasets.

Reinitialization in egocentric scenes: We build the first training and test datasets for depth-based egocentric hand-pose estimation. In the egocentric setting, our system estimates the user's hand pose from a head (or chest) mounted depth camera. This setting enables many exciting applications, especially augmented reality interfaces where a depth camera can be integrated with a head-mounted display. Our approach (Chapter 3) obtains state-of-the-art robustness in cluttered egocentric scenes. To teach a user a new skill, or coach them at a game, we need our augmented reality system to understand how the user is holding tools or other objects. In Chapter 4 we explore jointly recognizing hand pose and object manipulation. But, by building models offline, before tracking, we limit our tracker; it cannot recognize new objects that were not seen prior to test-time tracking.

Online updates for learning appearance: To track a new object, a tracker must learn and update an appearance model online, during tracking (Fig. 1.3). This problem of tracking novel objects is often called "model-free tracking" because a tracker does not initialize with an appearance model for the new object. In fact, state-of-the-art trackers exploit online learning and updates [242, 166, 249, 95, 240, 82, 136, 216, 98] to mitigate a variety of challenges: Instance specific models work better than generic models [101], but we may not see the test time individuals at train time; a camera auto-focus will initialize with only a single view of the target [111] but must adapt to other viewpoints as the target rotates; or, an individual target might change their appearance by putting on sunglasses [25]. But, online updates do not use "ground truth", rather they use labels produced by the tracker. This leads to the classic problem of drift: small errors in tracking accumulate in the appearance model until the tracker fails. In Chapter 5 we propose a tracking algorithm which uses self-paced learning to avoid updating the appearance model with erroneous data.

Deciding when to update: While deciding when to update the appearance model has a big impact on tracker performance, it has received almost no serious attention in recent literature [242]. In Chapter 6 we specifically study the model update decision. We begin with standard heuristic decision rules and conclude by training decision rules using empirical data. Virtually no single-object trackers use data to learn when to update their models. One reason for this is that few fully annotated videos exist, because annotating a video requires much more effort than annotating an image. We propose an interactive tracking and learning algorithm. With only weak supervision, our algorithm uses the tracker to improve annotation and uses the annotation to improve the tracker. Our iterative algorithm produces good trackers by learning from weakly annotated data. But, weak annotation lacks precision and detail. So, to collect more precise and detailed data for training trackers, we resort to sensored annotation.

Sensored annotation: We collect a *massive* new dataset for Identity-Aware Multiple-Object Tracking (IAMOT) with pixel-level annotations. To do so, we use a sophisticated data capture setup which integrates Ultra-Wide Band (UWB) tags with color and depth cameras to obviate manual annotation. We refer to our use of sensors to automatically obtain pixel-level identity labels as “sensored annotation”, as opposed to manual annotation. In Chapter 7, we use sensored annotation to (1) gain new insights into the IAMOT problem and (2) improve state-of-the-art IAMOT systems. First, we demonstrate (perhaps surprisingly) that running k-Single-Object Trackers (k-SOT) outperformed two state-of-the-art IAMOT systems. Our analysis concludes that IAMOT only works when the appearance models are sufficiently good. Second, using our large dataset with pixel-level annotations, we build a new IAMOT which outperforms all prior work.

Conclusions: One crucial theme recurs throughout this research: long-term trackers benefit greatly from (1) self-reinitialization after occlusions or failures and (2) learning better appearance models. By formulating tracking as a decision making problem, we develop principled approaches to improving reinitialization and appearance model updates. We improve appearance models for hands, by developing new and more challenging scenarios for both training and testing (Chapters 2, 3, 4). Using decision heuristics (both hand-designed and learned from data) we help model-free trackers build better appearance models (Chapter 5, 6). Finally, sensored annotation helps us improve both reinitialization and appearance updates in IAMOT (Chapter 7). Throughout our work, the decision making perspective provides theoretical guidance for this array of practical improvements to long-term tracking. And, improving long-term tracking will enable a multitude of exciting applications in augmented reality, vehicle automation, healthcare, and security.

Chapter 2

Model-Based Hand Pose Estimation

James Steven Supančič III, Grégory Rogez, Yi Yang,
Jamie Shotton & Deva Ramanan

2.1 Introduction

In the previous chapter, we discussed the importance of reinitialization in long-term hand tracking. We choose to focus on hand tracking because it empowers many practical applications, for example sign language recognition [99], visual interfaces [144], and driver analysis [158]. Recently introduced consumer depth cameras have spurred a flurry of new advances [179, 99, 44, 125, 144, 251, 220, 228, 176, 211]. In this chapter we will evaluate single-frame hand-pose estimation methods. We can reinitialize trackers after long-term occlusions, using the best single-frame method.

Motivation: Recent single-frame methods have demonstrated impressive results. But differing (often in-house) test sets, varying performance criteria, and annotation errors impede reliable comparisons [155]. Indeed, a recent meta-level analysis of object tracking papers reveals that it is difficult to trust the "best" reported method in any one paper [166]. In the field of object recognition, comprehensive benchmark evaluation has been vital for progress [62, 52, 57]. Our goal is to similarly diagnose the state of affairs, and to suggest future strategic directions, for depth-based hand pose estimation.

Contributions: Foremost, we contribute the *most extensive* evaluation of depth-based hand pose estimators to date. We evaluate 13 state-of-the-art hand-pose estimation systems across 4 test sets under uniform scoring criteria. Additionally, we provide a *broad survey* of contemporary approaches, introduce a *new test set* that addresses prior limitations, and propose a *new baseline* for pose estimation based on nearest-neighbor (NN) exemplar volumes. Surprisingly, we find that NN exceeds the accuracy of most existing systems (Fig. 2.1). We organize our discussion along three axes: test data (Sec. 2.2), training data (Sec. 2.3), and model architectures (Sec. 2.4). We survey and taxonomize approaches for each dimension, and also contribute novelty to each dimension (e.g. new data and models). After explicitly describing our experimental protocol (Sec. 2.5), we end with an extensive empirical analysis (Sec. 2.6).

Preview: We foreshadow our conclusions here. When hands are easily segmented or detected, current systems perform quite well. However, hand "activities" involving interactions with objects/surfaces are still challenging (motivating the introduction of our new dataset). Moreover, in such cases even humans perform imperfectly. For reasonable error measures, annotators disagree 20% of the time (due to self and inter-object occlusions and low resolution). This has immediate implications for test benchmarks, but also imposes a challenge when collecting and annotating training data. Finally, our NN baseline illustrates some surprising points. Simple memorization of training data performs quite well, outperforming most existing systems. Variations in the training data often dwarf variations in the model architectures themselves (e.g. , decision forests versus deep neural nets). Thus, our analysis offers the salient conclusion that "it's all about the (training) data".

Prior work: Our work follows in the rich tradition of benchmarking [57, 53, 192] and taxiomatic analysis [198, 55]. In particular, Erol *et al.* [55] reviewed hand pose analysis in 2007. Contemporary approaches have considerably evolved, prompted by the introduction of commodity depth cameras. We believe the time is right for another look. We do extensive cross-dataset analysis, by training and testing systems on different datasets [229]. Human-level studies in benchmark evaluation [140] inspired our analysis of human-performance. Finally, our NN-baseline is closely inspired by non-parametric approaches to pose estimation [199]. In particular, we use volumetric depth features in a 3D scanning-window (or volume) framework, similar to [208]. But, our baseline does not need SVM training or multi-cue features, making it simpler to implement.

2.2 Testing Data

Test scenarios for depth-based hand-pose estimation have evolved rapidly. Early work evaluated on synthetic data, while contemporary work almost exclusively evaluates on real data. However, because of difficulties in manual annotation (a point that we will revisit), evaluation was not always quantitative - instead, it has been common to show selected frames to give a qualitative sense of performance [51, 28, 160, 170]. We fundamentally assume that quantitative evaluation on real data will be vital for continued progress.

Test set properties: We have tabulated a list of contemporary test benchmarks in Table 2.1, giving URLs on our website[1]. We refer the reader to the caption for a detailed

[1]`http://www.ics.uci.edu/~jsupanci/#HandData`

Dataset	Chal.	Scn.	Annot.	Frms.	Sub.	Cam.	Dist. (mm)
ASTAR [251]	A	1	435	435	15	ToF	270-580
Dexter 1 [210]	A	1	3,157	3,157	1	Both	100-989
MSRA-2014 [176]	A	1	2,400	2,400	6	ToF	339-422
ICL [220]	A	1	1,599	1,599	1	Struct	200-380
FORTH [160]	AV	1	0	7,148	5	Struct	200-1110
NYU [228]	AV	1	8,252	8,252	2	Struct	510-1070
HandNet [244]	AV	1	202,198	202,198	10	Struct	200-650
MPG-2014 [231]	AV	1	2,800	2,800	1	Struct	500-800
FingerPaint [200]	AV	1	113,800	113,800	5	ToF	400-700
CVAR-EGO [157]	AV	2	2,166	2,166	1	ToF	60-650
MSRA [215]	AV	1	76,375	76,528	9	ToF	244-530
KTH [170]	AVC	1	NA	46,000	9	Struct	NA
LISA [158]	AVC	1	NA	3,100	1	Struct	900-3780
UCI-EGO [185]	AVC	4	364	3,640	2	ToF	200-390
Ours	AVC	10+	23,640	23,640	10	Both	200-1950

Challenges (Chal.): A-Articulation V-Viewpoint C-Clutter

Table 2.1: **Testing data sets:** We group existing benchmark test sets into 3 groups based on `challenges` addressed - articulation, viewpoint, and/or background clutter. We also tabulate the number of captured `scenes`, number of `annotated` versus `total frames`, number of `subjects`, `camera` type (structured light versus time-of-flight), and `distance` of the hand to camera. We introduce a new dataset (**Ours**) that contains a significantly larger range of hand depths (up to 2m), more scenes (10+), more annotated frames (24K), and more subjects (10) than prior work.

summary of specific dataset properties. Per dataset, Fig. 2.2 visualizes the pose-space covered using multi-dimensional scaling (MDS). We embed both the camera viewpoint angles and joint angles (in a normalized coordinate frame that is centered, scaled and rotated to the camera viewpoint). We conclude that previous datasets make different assumptions about articulation, viewpoint, and perhaps most importantly, background clutter. Such assumptions are useful because they allow researchers to focus on particular aspects of the problem. However it is crucial to make such assumptions explicit [229], which much prior work does not. We do so below.

Articulation: Many datasets focus on pose estimation with the assumption that detection and overall hand viewpoint is either given or limited in variation. Example datasets include

MSRA-2014 [176], A-Star [251], and Dexter [210]. While these test sets focus on estimating hand articulation, not all test sets contain the same amount of pose variation. For example, a sign language test set will exhibit a small number of discrete poses. To quantify articulation, we fit a multi-variate Gaussian distribution to a test set's finger joint angles. Then we compute the *differential entropy* for the test set's distribution:

$$h(\Sigma) = .5 \log \left((2\pi e)^N \det(\Sigma) \right) \tag{2.1}$$

where Σ is the covariance of the test set's joint angles and N is the number of joint angles in each pose vector. This analysis suggests that our proposed test set contains greater pose variation (entropy, $h = 89$) than the ICL ($h = 34$), NYU ($h = 82$), FORTH ($h = 65$) or A-STAR ($h = 79$) test sets. We focus on **ICL** [220] as a representative example for experimental evaluation because it has been used in multiple prior published works [220, 44, 155].

Art. and viewpoint: Other test sets have focused on both viewpoint variation and articulation. FORTH [160] provides five test sequences with varied articulations and viewpoints, but these are unfortunately unannotated. The CVAR-EGO [157] dataset provides highly precise joint annotations but contains fewer frames and only one subject. In our experiments, we analyze the **NYU** dataset [228] because of its wide pose variation (see Fig. 2.2), larger size, and accurate annotations (see Sec. 2.3).

Art. + View. + Clutter: The most difficult datasets contain cluttered backgrounds that are not easy to segment away. These datasets tend to focus on "in-the-wild" hands performing activities and interacting with nearby objects and surfaces. The KTH Dataset [170] provides a rich set of 3rd person videos showing humans interacting with objects. Unfortunately, annotations are not provided for the hands (only the objects). Similarly, the LISA [158] dataset provides cluttered scenes captured inside vehicles. However, joint po-

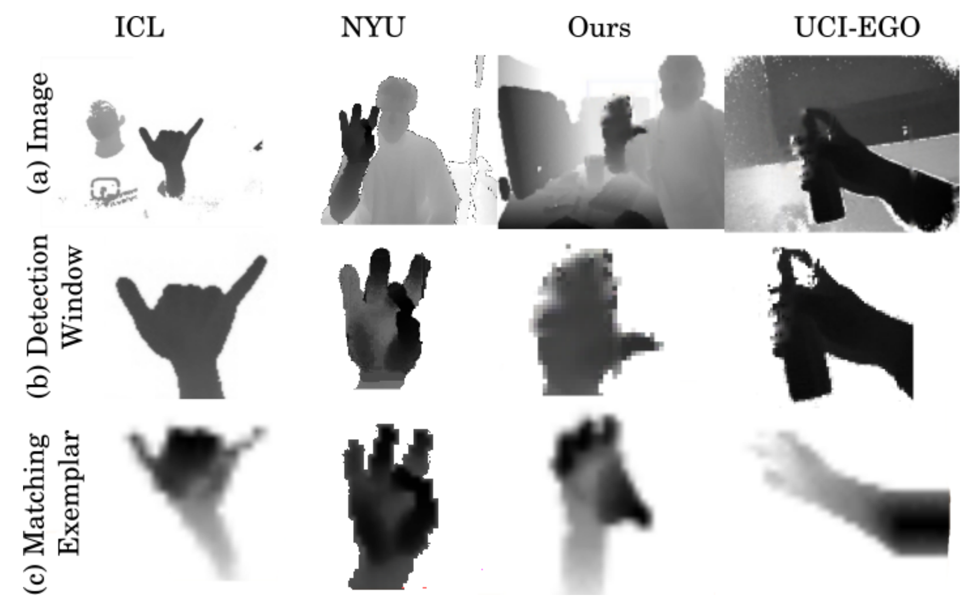


Figure 2.1: **NN Memorization:** We evaluate a broad collection of hand pose estimation algorithms on different training and test sets under consistent criteria. Test sets which contained limited variety, in pose and range, or which lacked complex backgrounds were notably easier. To aid our analysis, we introduce a simple 3D exemplar (nearest-neighbor) baseline that both detects and estimates pose suprisingly well, outperforming most existing systems. We show the best-matching detection window in (**b**) and the best-matching exemplar in (**c**). We use our baseline to rank dataset difficulty, compare algorithms, and show the importance of training set design. We provide a detailed analysis of which problem types are currently solved, what open research challenges remain, and provide suggestions for future model architectures.

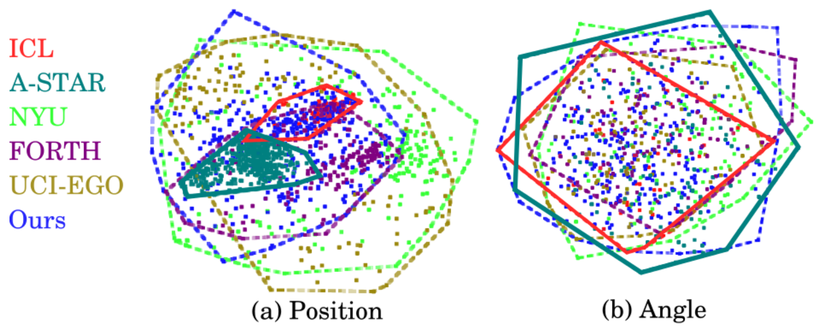


Figure 2.2: **Pose variation:** We use MDS (multi-dimensional scaling) to plot the pose space covered by a set of hand datasets with compatible joint annotations. We split the pose space into two components and plot the camera viewpoint angles (a) and finger joint angles (b). For each test set, we plot the convex hull of its poses. In terms of joint angle coverage, most test sets are similar. In terms of camera viewpoint, some test sets consider a smaller range of views (e.g. , ICL and A-STAR). We further analyze various assumptions made by datasets in the text.

sitions are not annotated, only coarse gesture. The **UCI-EGO** [185] dataset provides challenging sequences from an egocentric perspective with joint level annotations, and so is included in our benchmark analysis.

Our test set: Our empirical evaluation will show that *in-the-wild hand activity* is still challenging. To push research in this direction, we have collected and annotated our own test set of real images (labeled as **Ours** in Table 2.1, examples in Fig. 2.3). As far as we are aware, our dataset is the first to focus on hand pose estimation *across multiple subjects and multiple cluttered scenes*. This is important, because any practical application must handle diverse subjects, scenes, and clutter.

2.3 Training Data

Here we discuss various approaches for generating training data (ref. Table 2.2). Real annotated training data has long been the gold standard for supervised learning. However, the generally accepted wisdom (for hand pose estimation) is that the space of poses is too large to manually annotate. This motivates approaches to leverage synthetically generated training data, discussed further below.

Real data + manual annotation: Arguably, the space of hand poses exceeds what can be sampled with real data. Our experiments identify a second problem: perhaps surprisingly, human annotators often disagree on pose annotations. For example, in our test set, human annotators disagree on 20% of pose annotations (considering a 20mm threshold) as plotted in Fig. 2.20. These disagreements arise from limitations in the raw sensor data, either due to poor resolution or occlusions. We found that low resolution consistently corresponds to annotation ambiguities, across test sets. See Sec. 2.5.2) for further discussion and examples.

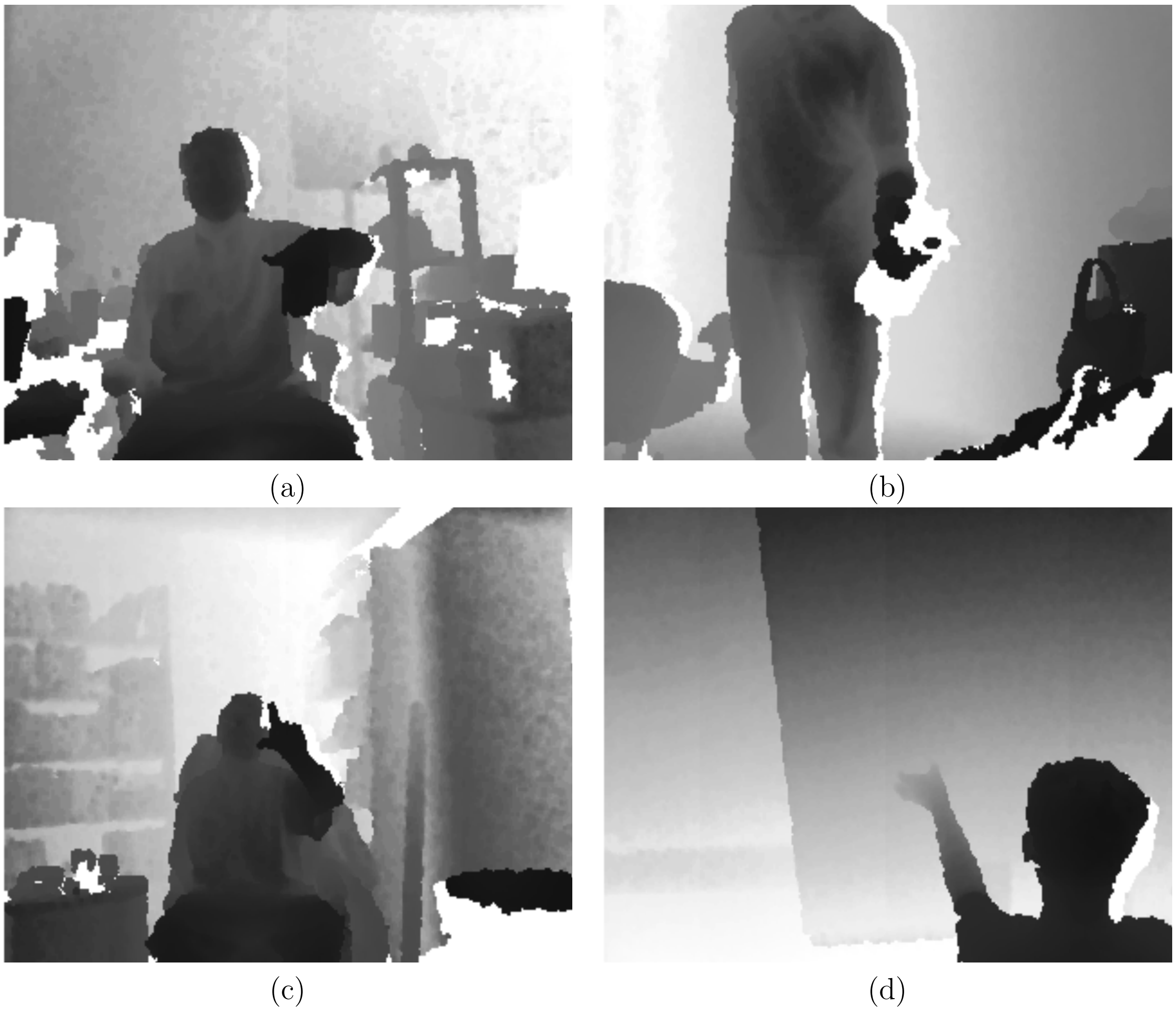


Figure 2.3: **Our new test data** challenges methods with clutter (a), object manipulation (b), low-res (c), and various viewpoints (d). We collected data in diverse environments (8 offices, 4 homes, 4 public spaces, 2 vehicles, and 2 outdoors) using time-of-flight (Intel/Creative Gesture Camera) and structured-light (ASUS Xtion Pro) depth cameras. Ten (3 female and 7 male) subjects were given prompts to perform natural interactions with objects in the environment, as well as display 24 random and 24 canonical poses.

These ambiguities are often mitigated by placing the hand close to the camera [251, 220, 176, 157]. As an illustrative example, we evaluate the **ICL** training set [220].

Real data + automatic annotation: Data gloves directly obtain automatic pose annotations for real data [251]. However, they require painstaking per-user calibration. Magnetic markers can partially alleviate calibration difficulties [244] but still distort the hand shape that is observed in the depth map. When evaluating depth-only systems, colored markers can provide ground-truth through the RGB channel [200]. Alternatively, one could use a "passive" motion capture system. We evaluate the larger **NYU** training set [228] that annotates real data by fitting (offline) a skinned 3D hand model to high-quality 3D measurements. Finally, integrating model fitting with tracking lets one leverage a small set of annotated reference frames to annotate an entire video [157].

Quasi-synthetic data: Augmenting real data with geometric computer graphics models provides an attractive solution. For example, one can apply geometric transformations (e.g. , rotations) to both real data and its annotations [220]. If multiple depth cameras are used to collect real data (that is then registered to a model), one can synthesize a larger set of varied viewpoints [211, 228]. Finally, mimicking the noise and artifacts of real data is often important when using synthetic data. Domain transfer methods [44] learn the relationships between a small real dataset and large synthetic one.

Synthetic data: Another hope is to use data rendered by a computer graphics system. Graphical synthesis sidesteps the annotation problem completely: precise annotations can be rendered along with the features. One can easily vary the size and shape of synthesized training hands, a fact which allows us to explore how user-specific training data impacts accuracy. Our experiments (ref. Sec. 5.5) verify that results may be optimistic when the training and test datasets contain the same individuals, as non-synthetic datasets commonly do (ref. Table 2.2). When synthesizing novel exemplars, it is important define a good sampling distribution. A common strategy for generating a sampling distribution is to collect pose samples with motion capture data [35, 64]. The **UCI-EGO** training set [185] synthesizes data with an egocentric prior over viewpoints and grasping poses.

2.3.1 libhand training set:

To further examine the effect of training data, we created a massive custom training set of 25,000,000 RGB-D training instances with the open-source **libhand** model (some examples are shown in Fig. 2.7). We modified the code to include a forearm and output depth data, semantic segmentations, and keypoint annotations. We emphasize that this *synthetic* training set is distinct from our new test dataset of *real* images.

Synthesis parameters: To avoid omitting possible but unlikely poses from our synthetic training set, we do not use motion capture data. Instead, we take a brute-force approach based on rejection-sampling. We uniformly and independently sample joint angles (from a bounded range), and throw away invalid samples that yield self-intersecting 3D hand poses. Specifically, using the libhand joint identifiers shown in Fig. 2.4, we generate poses by uniformly sampling from bounded ranges, as shown in Table. 2.3.

Quasi-Synthetic backgrounds: Hand synthesis engines commonly under-emphasize the importance of image backgrounds [237, 160, 228]. For methods operating on pre-segmented images [99, 210, 176], this is likely not an issue. However, for active hands "in-the-wild", the choice of synthetic backgrounds, surfaces, and interacting objects becomes important. Moreover, some systems require an explicit negative set (of images not containing hands) for training. To synthesize a robust background/negative training set, we take a quasi-synthetic approach by applying random affine transformations to 5,000 images of real scenes, yielding

Dataset	Generation	Viewpoint	Views	Size	Subj.
ICL [220]	Real + manual annot.	3rd Pers.	1	331,000	10
NYU [228]	Real + auto annot.	3rd Pers.	3	72,757	1
HandNet [244]	Real + auto annot.	3rd Pers.	1	12,773	10
UCI-EGO [185]	Synthetic	Egocentric	1	10,000	1
libhand [237]	Synthetic	Generic	1	25,000,000	1

Table 2.2: **Training data sets:** We broadly categorize training datasets by the method used to `generate` the data and annotations: real data + manual annotations, real data + automatic annotations, or synthetic data (and automatic annotations). Most existing datasets are `viewpoint`-specific (tuned for 3rd-person or egocentric recognition) and limited in `size` to tens of thousands of examples. NYU is unique in that it is a `multiview` dataset collected with multiple cameras, while ICL contains shape variation due to multiple (10) `subjects`. To explore the effect of training data, we use the public libhand animation package to generate a massive training set of 25 million examples.

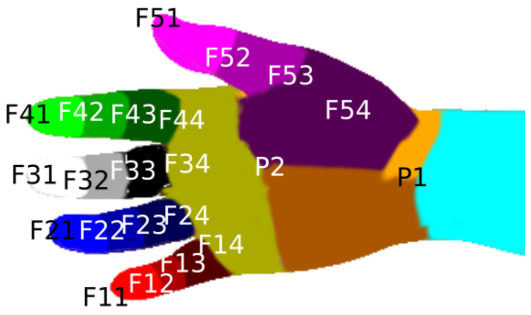


Figure 2.4: **libhand joints:** We use the above joint identifiers to describe how we sample poses (for libhand) in table 2.3. See `http://www.libhand.org/` for more details on the joints and their parameters.

Description	Identifiers	bend	side	elongation
Intermediate and Distal Joints	$F_{1:4,2:3}$	$U(\frac{-\pi}{2}^r, \frac{\pi}{7}^r)$	0	0
Proximal-Carpal Joints	$F_{1:4,4}$	$U(\frac{-\pi}{2}^r, \frac{\pi}{7}^r)$	$U(\frac{-\pi}{8}^r, \frac{\pi}{8}^r)$	0
Thumb Metacarpal	$F_{5,4}$	$U(-1^r, .5^r)$	$U(-.7^r, 1.2^r)$	$U(.8^r, 1.2^r)$
Thumb Proximal	$F_{5,3}$	$U(-1^r, -.6^r)$	$U(-.2^r, .5^r)$	0
Wrist Articulation	P_1	$U(-1^r, 1^r)$	$U(-.5^r, .8^r)$	0

Table 2.3: **Synthetic hand distribution:** We render synthetic hands with joint angles sampled from the above uniform distributions. `bend` refers to the natural extension-retraction of the finger joints. The proximal-carpal, wrist and thumb joints are additionally capable of `side-to-side` articulation. We do not consider a third type of articulation, `twist`, because it would be extremely painful and result in injury. We model anatomical differences by `elongating` some bones fanning out from a joint. Additionally, we apply an isotropic global metric scale factor sampled from the range $U(\frac{2}{3}, \frac{3}{2})$. Finally, we randomize the camera viewpoint by uniformly sampling tilt, yaw and roll from $U(0, 2\pi)$.

Method	Approach	Model-drv.	Data-drv.	Detection	Implementation	FPS
Simulate [144]	Tracker (simulation)	Yes	No	Initialization	Published	50
NiTE2 [175]	Tracker (pose search)	No	Yes	Initialization	Public	> 60
Particle Swarm Opt. (PSO) [160]	Tracker (PSO)	Yes	No	Initialization	Public	15
Hough Forest [251]	Decision forest	Yes	Yes	Decision forest	Ours	12
Random Decision Forest (RDF) [99]	Decision forest	No	Yes	-	Ours	8
Latent Regression Forest (LRF) [220]	Decision forest	No	Yes	-	Published	62
DeepJoint [228]	Deep network	Yes	Yes	Decision forest	Published	25
DeepPrior [155]	Deep network	No	Yes	Scanning window	Ours	5000
DeepSegment [59]	Deep network	No	Yes	Scanning window	Ours	5
Intel PXC [87]	Morphology (convex detection)	No	No	Heuristic segment	Public	> 60
Cascades [185]	Hierarchical cascades	No	Yes	Scanning window	Provided	30
Ego. WS. [183]	Multi-class SVM	No	Yes	Whole volume classif.	Provided	275
EPM [261]	Deformable part model	No	Yes	Scanning window	Ours	1/2
Volumetric Exemplars	Nearest neighbor (NN)	No	Yes	Scanning volume	Ours	1/15

Table 2.4: **Summary of methods:** We broadly categorize the pose estimation systems that we evaluate by their overall `approach`: decision forests, deep models, trackers, or others. Though we focus on single-frame systems, we also evaluate trackers by providing them manual initialization. `Model-driven` methods make use of articulated geometric models at test time, while `data-driven` models are trained beforehand on a training set. Many systems begin by `detect`ing hands with a Hough-transform or a scanning window/volume search. Finally, we made use of public source code when available, or re-`implement`ed the system ourselves, verifying our implementation's accuracy on published benchmarks. 'Published' indicates that published performance results were used for evaluation, while 'public' indicates that source code was available, allowing us to evaluate the method on additional test sets. We report the fastest speeds (in `FPS`), either reported or our implementation's.

a total of 1,000,0000 pseudo-synthetic backgrounds. We found it useful to include human bodies in the negative set because faces are common distractors for hand models.

2.4 Methods

Next we survey existing approaches to hand pose estimation (summarized in Table 2.4). We conclude by introducing a novel volumetric nearest-neighbor (NN) baseline.

2.4.1 Taxonomy

Trackers versus detectors: We focus our analysis on single-frame methods. For completeness, we also consider several tracking baselines [160, 175, 87] needing ground-truth initialization. Manual initialization may provide an unfair advantage, but we will show that single-frame methods are still nonetheless competitive, and in most cases, outperform tracking-based approaches. One reason is that single-frame methods essentially "reinitialize" themselves at each frame, while trackers cannot recover from an error.

Discrete versus continuous pose: We further concentrate our analysis on the continuous pose regression problem. Historically however, much prior work tackled the problem from a discrete gesture classification perspective [149, 175, 174, 159]. Yet, these perspectives are closely related because one can tackle continuous pose estimation using a large number of discrete classes. As such, we evaluate several discrete classifiers in our benchmark [152, 183].

Data-driven versus model-driven: Historic attempts to estimate hand pose optimized a geometric model to fit observed data [51, 28, 213]. Recently, Oikonomidis *et al.* [160] demonstrated hand tracking using GPU accelerated Particle Swarm Optimization (PSO). However, such optimizations remain notoriously difficult due to local minima in the objective function. As a result, model driven systems have historically found their successes mostly limited to the tracking domain, where initialization constrains the search space [210, 144, 176]. For single image detection, various fast classifiers and regressors have obtained real-time speeds [99, 87, 155, 156, 221, 215, 130, 239]. Most of the systems we evaluate fall into this category. When these classifiers are trained with data synthesized from a geometric model, they can be seen as efficiently approximating model fitting.

Multi-stage pipelines: Systems commonly separate their work into discrete stages: detecting, posing, refining and validating hands. Some systems use special purpose detectors as a "pre-processing" stage [68, 160, 99, 41, 251, 87, 187, 228]. A segmentation pre-processing stage has been historically popular. Typically, RGB skin classification [233] or morphological operations on the depth image [174] segment the hand from the background. Such segmentation allows computation of Zernike moment [41] or skeletonization [174] features. While RGB features compliment depth [185, 74], skin segmentation appears difficult to generalize across subjects and scenes with varying lighting [176]. We evaluate a depth-based segmentation system [87] for completeness. Other systems use a model for inverse-kinematics/IK [228, 251], geometric refinement/validation [144, 221], or collaborative filtering [36] during a "post-processing" stage. For highly precise hand pose estimation, recent hybrid pipelines compliment data-driven per-frame reinitialization with model-based refinement [226, 15, 211, 176, 255].

2.4.2 Architectures

In this section, we describe popular architectures for hand-pose estimation, placing in bold those systems that we empirically evaluate.

Decision forests: Decision forests constitute a dominant paradigm for estimating hand pose from depth. **Hough Forests** [251] take a two-stage approach of hand detection followed by pose estimation. **Random Decision Forests (RDFs)** [99] and **Latent Regression Forests (LRFs)** [220] leave the initial detection stage unspecified, but both make use of coarse-to-fine decision trees that perform rough viewpoint classification followed by detailed pose estimation. We experimented with several detection front-ends for RDFs and LRFs, finally selecting the first-stage detector from Hough Forests for its strong performance.

Part model: Pictorial structure models have been popular in human body pose estimation [253], but they appear somewhat rarely in the hand pose estimation literature. For completeness, we evaluate a deformable part model defined on depth image patches [65]. We specifically train an **exemplar part model (EPM)** constrained to model deformations consistent with 3D exemplars [261].

Deep models: Recent systems have explored using deep neural nets for hand pose estimation. We consider three variants in our experiments. **DeepJoint** [228] uses a three stage pipeline that initially detects hands with a decision forest, regresses joint locations with a deep network, and finally refines joint predictions with inverse kinematics (IK). **DeepPrior** [155] is based on a similar deep network, but does not require an IK stage and instead relies on the network itself to learn a spatial prior. **DeepSeg** [59] takes a pixel-labeling approach, predicting joint labels for each pixel, followed by a clustering stage to produce joint locations. This procedure is reminiscent of pixel-level part classification of Kinect [202], but substitutes a deep network for a decision forest.

2.4.3 Volumetric exemplars

We propose a nearest-neighbor (NN) baseline for additional diagnostic analysis. Specifically, we convert depth map measurements into a 3D voxel grid, and simultaneously detect and estimate pose by scanning over this grid with volumetric exemplar templates. We introduce several modifications to ensure an efficient scanning search.

Voxel grid: Depth cameras report depth as a function of pixel (u, v) coordinates: $D(u, v)$. To construct a voxel grid, we first re-project these image measurements into 3D using known camera intrinsics f_u, f_v.

$$(x, y, z) = \left(\frac{u}{f_u} D(u, v), \frac{v}{f_v} D(u, v), D(u, v)\right) \tag{2.2}$$

Given a test depth image, we construct a binary voxel grid $V[x, y, z]$ that is '1' if a depth value is observed at a quantized (x, y, z) location. To cover the rough viewable region of a camera, we define a coordinate frame of M^3 voxels, where $M = 200$ and each voxel spans

10mm^3. We similarly convert training examples into volumetric exemplars $E[x, y, z]$, but instead use a smaller N^3 grid of voxels (where $N = 30$), consistent with the size of a hand.

Occlusions: When a depth measurement is observed at a position $(x', y', z') = 1$, all voxels behind it are occluded $z > z'$. We define occluded voxels to be '1' for both the test-time volume V and training exemplar E.

Distance measure: Let V_j be the j^{th} subvolume (of size N^3) extracted from V, and let E_i be the i^{th} exemplar. We simultaneously detect and estimate pose by computing the best match in terms of Hamming distance:

$$(i^*, j^*) = \underset{i,j}{\operatorname{argmin}} \operatorname{Dist}(E_i, V_j) \qquad \text{where} \tag{2.3}$$

$$\operatorname{Dist}(E_i, V_j) = \sum_{x,y,z} \mathcal{I}(E_i[x, y, z] \neq V_j[x, y, z]), \tag{2.4}$$

such that i^* is the best-matching training exemplar and j^* is its detected position.

Efficient search: A naïve search over exemplars and subvolumes is prohibitively slow. But because the underlying features are binary and sparse, there exist considerable opportunities for speedup. We outline two simple strategies. First, one can eliminate subvolumes that are empty, fully occluded, or out of the camera's field-of-view. Song *et al.* [208] refer to such pruning strategies as "jumping window" searches. Second, one can compute volumetric Hamming distances with 2D computations:

$$\operatorname{Dist}(E_i, V_j) = \sum_{x,y} |e_i[x, y] - v_j[x, y]| \qquad \text{where} \tag{2.5}$$

$$e_i[x, y] = \sum_z E_i[x, y, z], \qquad v_j[x, y] = \sum_z V_j[x, y, z].$$

Intuition for our encoding: Because our 3D volumes are projections of 2.5D measurements, they can be sparsely encoded with a 2D array (see Fig. 2.5). Taken together, our two simple strategies imply that a *3D volumetric search can be as practically efficient as a 2D scanning-window search*. For a modest number of exemplars, our implementation still took tens of seconds per frame, which sufficed for our offline analysis. We posit faster NN algorithms could yield real-time speed [151, 152].

Comparison: Our volumetric exemplar baseline uses a scanning volume search and 2D depth encodings. It is useful to contrast this with a "standard" 2D scanning-window template on depth features [90]. First, our exemplars are defined in metric coordinates (Eq. 2.2). This means that they will *not* fire on the small hands of a toy figurine, unlike a scanning window search over scales. Second, our volumetric search ensures that the depth encoding from a local window contain features only within a fixed N^3 volume. This gives it the ability to segment out background clutter, unlike a 2D window (Fig. 2.6).

2.5 Protocols

We've surveyed hand-pose estimation methods. However, methods typically use slightly different evaluation criteria, which makes comparison tricky. So first (in Sec. 2.5.1), we describe our protocol for evaluating hand-pose estimators, which supports fair comparisons between different methods. But, evaluation requires accurate ground truth. Because annotating hand-pose test data is notoriously error prone [157], we second (Sec. 2.5.2) propose a protocol for both annotating hand-pose test data and for evaluating the accuracy of said annotations. After establishing criteria for evaluating methods and for assessing the reliability of our evaluation, we finally (Sec. 2.5.3) interpret how our quantitative performance measurements compare to qualitative and practical results.

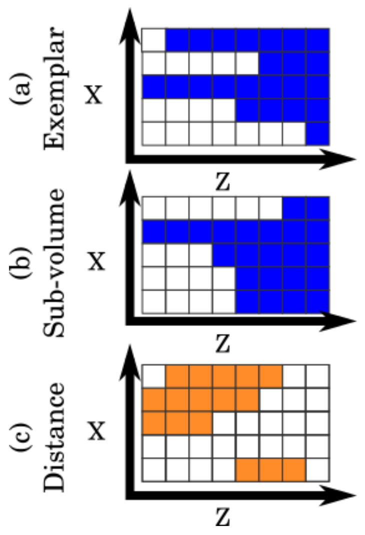


Figure 2.5: **Volumetric Hamming distance:** We visualize 3D voxels corresponding to an exemplar (a) and subvolume (b). For simplicity, we visualize a 2D slice along a fixed y-value. Because occluded voxels are defined to be '1' (indicating they are occupied, shown in blue) the total Hamming distance is readily computed by the L1 distance between projections along the z-axis (c), mathematically shown in Eq.(2.5).

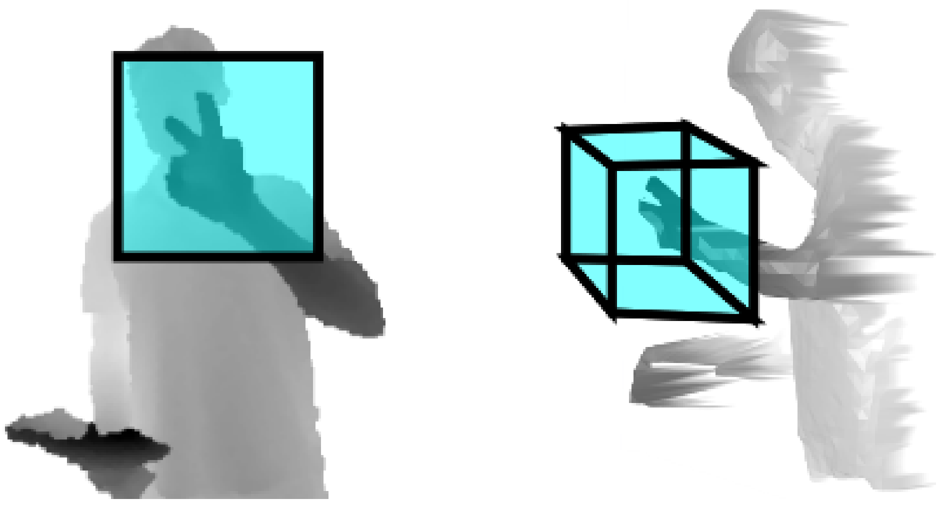


Figure 2.6: **Windows versus volumes:** 2D scanning windows (a) versus 3D scanning volumes (b). Volumes can ignore background clutter that lies outside the 3D scanning volume but still falls inside its 2D projection. For example, when scoring the shown hand, a 3D volume will ignore depth measurements from the shoulder and head, unlike a 2D window.

2.5.1 Evaluation

Implementations: We evaluate public code when available [160, 175, 87]. Some authors responded to our request for their code [185]. When software was not available, we attempted to re-implement methods ourselves. We were able to successfully reimplement [99, 251, 155], matching the accuracy on published results [220, 155]. In other cases, our in-house implementations did not suffice [228, 220]. For these latter cases, we include published performance reports, but unfortunately, they are limited to their own datasets. This partly motivated us to perform a multi-dataset analysis. In particular, previous benchmarks have shown that one can still compare algorithms across datasets using head-to-head matchups (similar to approaches that rank sports teams which do not directly compete [166]). We use our NN baseline to do precisely this. Finally, to spur further progress, *we have made our implementations publicly available, together with our evaluation code.*

Reprojection error: Following past work, we evaluate pose estimation methods as a regression task that predicts a set of 3D joint locations [160, 99, 176, 225, 220]. Given a predicted and ground-truth pose, we compute both the average and max 3D reprojection error (in mm) across all joints. We use the skeletal joints defined by libhand [237]. We then summarize performance by plotting the proportion of test frames whose average (or max) error falls below a threshold.

Detection issues: Reprojection error is hard to define during detection failures: that is, false positive hand detections or missed hand detections. Such failures are likely in cluttered scenes or when considering scenes containing zero or two hands. If a method produced zero detections when a hand was present, or produced one if no hand was present, this was treated as a "maxed-out" reprojection error (of ∞ mm). If two hands were present, we scored each method against both and took the minimum error. Though we have released our evaluation software[2], we give pseudocode in Alg. 1.

Missing data: Another challenge with reprojection error is missing data. First, some methods predict 2D screen coordinates for joints, not 3D metric coordinates [174, 87, 59, 228]. Approximating $z \approx D(u, v)$, inferring 3D joint positions should be straightforward with Eq. 2.2. But, small 2D position errors can cause significant errors in the approximated depth, especially around the hand silhouette. To mitigate, we instead use the centroid depth of a segmented/detected hand when the measured depth lies outside the segmented volume. Past comparisons appear not to do this [155], somewhat unfairly penalizing 2D approaches [228]. Second, some methods may predict a subset of joints [87, 174]. To ensure a consistent comparison, we force such methods to predict the locations of visible joints with a post-processing inverse-kinematics (IK) stage [228]. We fit the libhand kinematic model to the predicted joints, and infer the location of missing ones. Third, ground-truth joints may be occluded. By convention, we only evaluate visible joints in our benchmark analysis.

2.5.2 Annotation

We now describe both how we collect ground truth annotations and how we asses the accuracy of our annotations. We present the annotator with cropped RGB and depth images. They then click semantic key-points, corresponding to specific joints, on either the RGB or depth images. To ease the annotator's task and to get 3D keypoints from 2D clicks we invert the forward rendering (graphics) hand model provided by libhand which projects model parameters θ to 2D keypoints $P(\theta)$. While they label joints, an inverse kinematic solver minimizes the distance between the currently annotated 2D joint labels, $\forall_{j \in J} L_j$, and those projected from the libhand model parameters, $\forall_{j \in J} P_j(\theta)$. Formally, our inverse kinematic

[2]`https://github.com/jsupancic/AStar_Dual_Tree_HandPose`

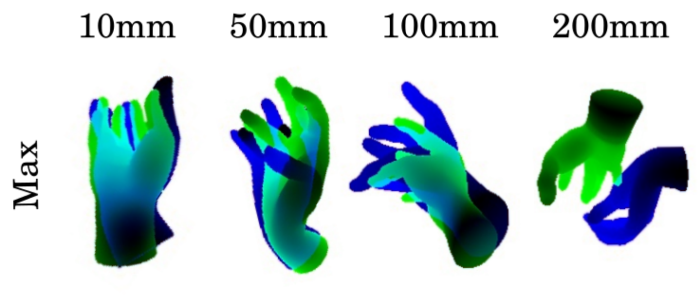


Figure 2.7: **Our error criteria:** For each predicted hand, we calculate the average and maximum distance (in mm) between its skeletal joints and a ground-truth. In our experimental results, we plot the fraction of predictions that lie within a distance threshold, for various thresholds. This figure visually illustrates the misalignment associated with various thresholds for max error. A 50mm max-error seems visually consistent with a "roughly correct pose estimation", and a 100mm max-error is consistent with a "correct hand detection".

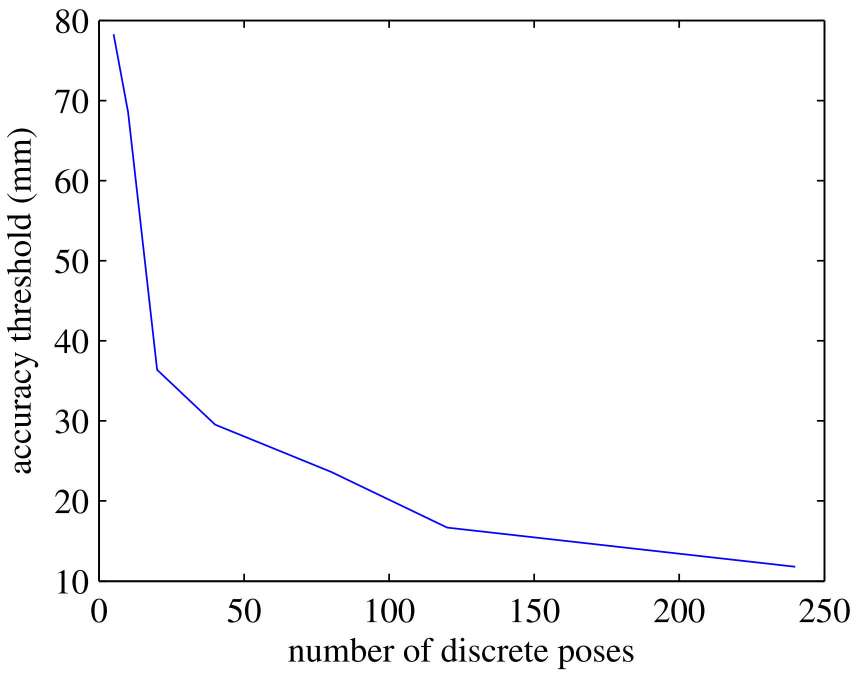


Figure 2.8: **Required precision per discrete pose:** Larger pose vocabularies require more precision. We plot this relationship by considering the sparsest distribution of N poses. A max-joint-error precision of 20mm suffices to perfectly disambiguate a vocabulary of 100 discrete poses, while 10mm roughly disambiguates 240 poses. If perfect classification is not needed, one can enlarge the effective vocabulary size.

```
input  : predictions and ground truths for each image
output: a set of errors, one per frame
forall test_images do
    $P \leftarrow$ method's most confident prediction;
    $G \leftarrow$ ground truths for the current test_image;
    if $G = \emptyset$ then
        /* Test Image contains zero hands                                   */
        if $P = \emptyset$ then
            errors $\leftarrow$ errors $\cup \{0\}$;
        else
            errors $\leftarrow$ errors $\cup \{\infty\}$;
        end
    else
        /* Test Image contains hand(s)                                      */
        if $P = \emptyset$ then
            errors $\leftarrow$ errors $\cup \{\infty\}$;
        else
            best_error $\leftarrow \infty$;
            /* Find the ground truth best matching the method's prediction
               */
            forall $H \in G$ do
                /* For mean error plots, replace $\max_i$ with $\text{mean}_i$   */
                /* $V$ denotes the set of visible joints                    */
                current_error $\leftarrow \max_{i \in V} ||H_i - P_i||_2$;
                if current_error < best_error then
                    best_error $\leftarrow$ current_error;
                end
            end
            errors $\leftarrow$ errors $\cup$ {best_error};
        end
    end
end
```
Algorithm 1: Scoring Procedure: For each frame we compute a max or mean reprojection error for the ground truth(s) G and prediction(s) P. We later plot the proportion of frames with an error below a threshold, for various thresholds.

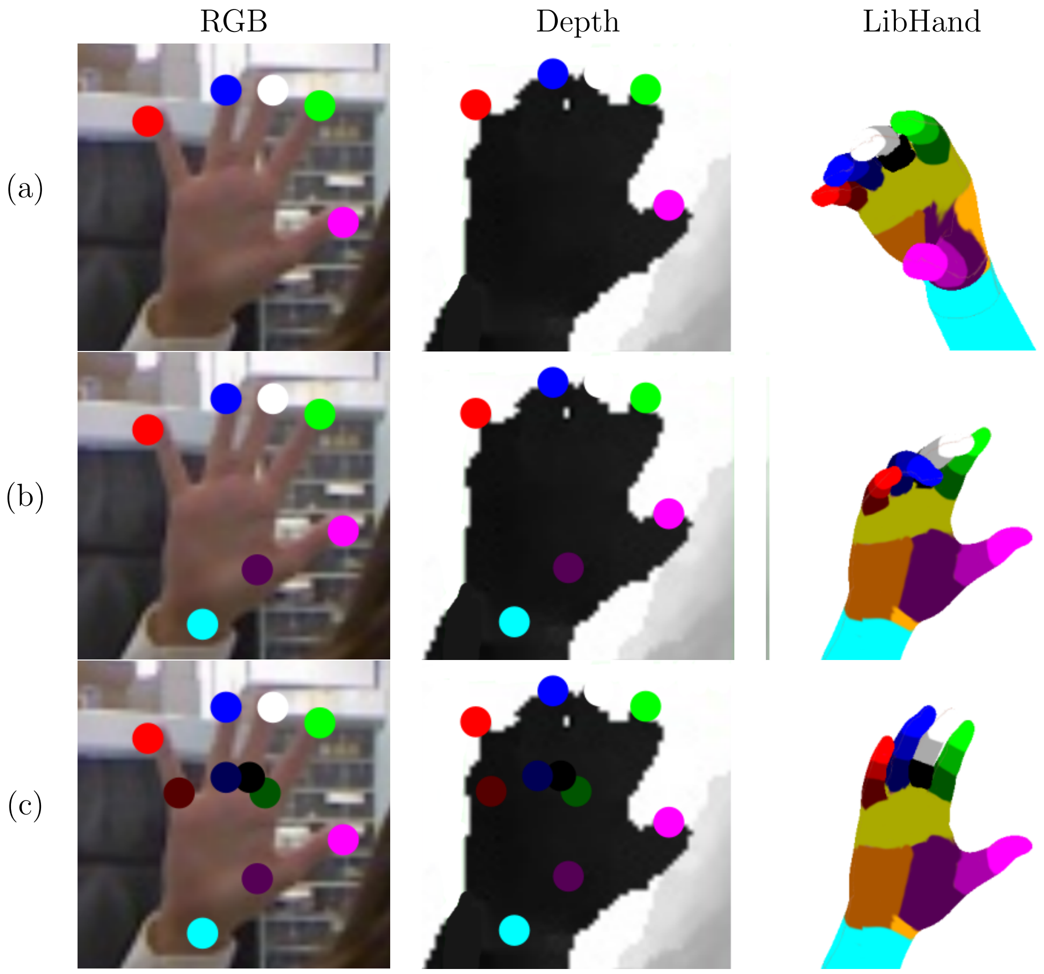


Figure 2.9: **Annotation procedure:** We annotate until we are satisfied that the fitted hand pose matches the RGB and depth data. The first two columns show the image evidence presented and keypoints received. The right most column shows the fitted libhand model. The IK solver is able to easily fit a model to the five given keypoints (a), but it doesn't match the image well. The annotator attempts to correct the model (b), to better match the image, by labeling the wrist. Labeling additional finger joints finally yields and acceptable solution (c).

solver attempts to solve the following:

$$\min_{\theta} \sum_{j \in J} \|L_j - P_j(\theta)\|_2 \tag{2.6}$$

The currently fitted libhand model, shown to the annotator, updates online as more joints are labeled. When the annotator indicates satisfaction with the fitted model, we proceed to the next frame. We give an example of the annotation process in Fig. 2.9.

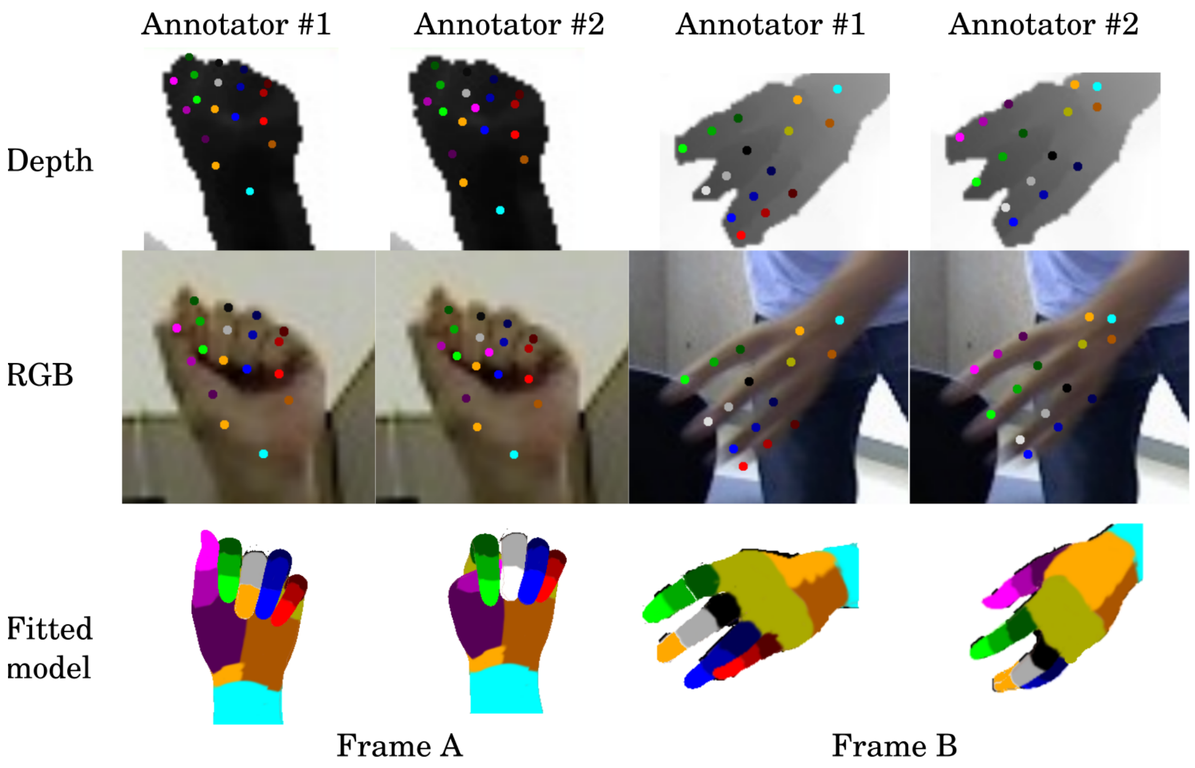


Figure 2.10: **Annotator disagreements:** With whom do you agree? We show two frames where annotators disagree. The top two rows show the RGB and depth images with annotated keypoints. The bottom row shows the `libhand` model fit to those annotations. In *Frame A*, is the thumb upright or tucked underneath the fingers? In *Frame B*, is the thumb or pinky occluded? Long-range (low resolution) makes this important case hard to decide, In one author's opinion, annotator 1 is more consistent with RGB evidence while annotator 2 is more consistent with depth evidence (we always present annotators with both).

Strengths: Our annotation process has several strengths. First, kinematic constraints prevent some keypoint combinations, so it is often possible to fit the model by labeling only a subset of keypoints. Second, the fitted model provides annotations for occluded keypoints. Third and most importantly, the fitted model provides 3D (x,y,z) keypoint locations given only 2D (u,v) annotations.

Disagreements: As shown in in Fig. 2.20, annotators disagree substantially on the hand pose, in a surprising number of cases. In applications, such as sign language [214] ambiguous poses are typically avoided. We believe it is important to acknowledge that, in general, it may not be possible to achieve full precision. For our proposed test set (with an average hand

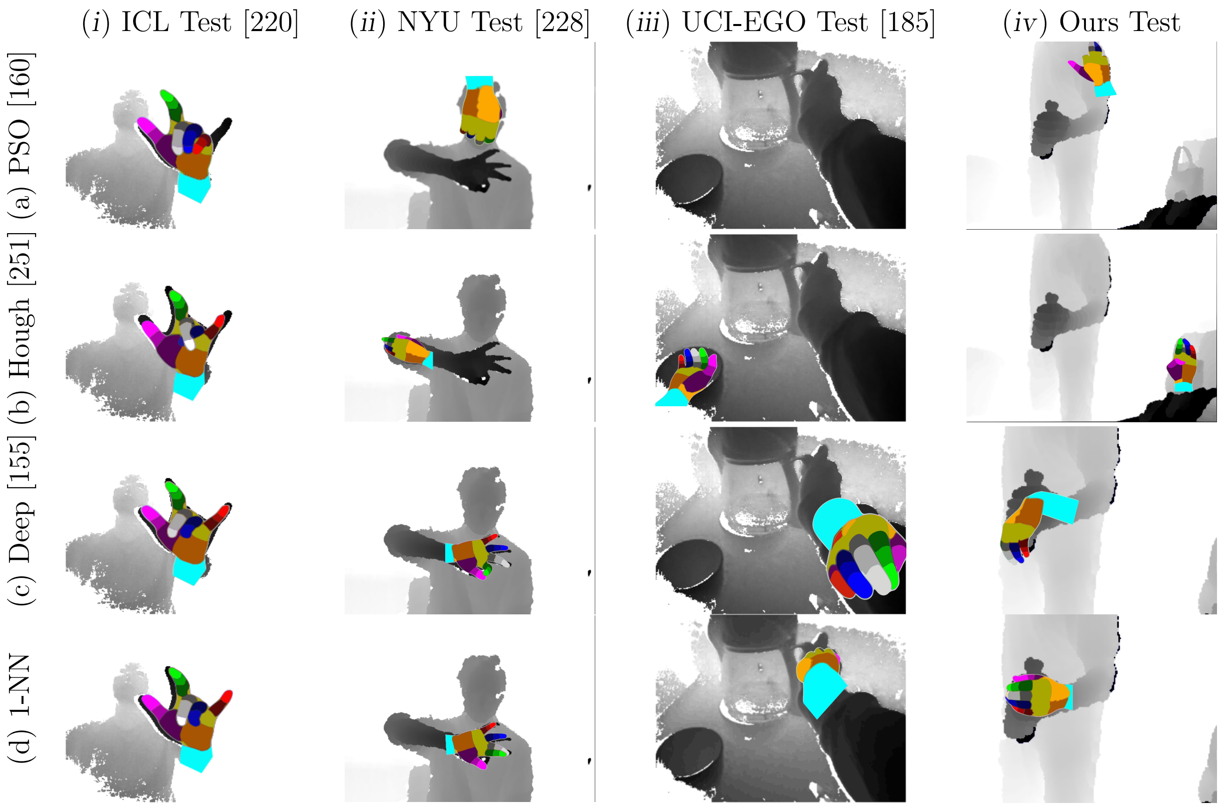


Figure 2.11: **Characteristic results:** The PSO [160] tracker tends to miss individually extended fingers, in this case the pinky (a,*i*), due to local minima. Faces are common distractors for all methods. But, the PSO tracker in particular never recovers once it locks onto a face. The first-stage Hough forest [251] detector can recover from failures. But, the trees vote independently for global orientation and location using only local patch evidence. This local evidence seems insufficient to differentiate hands from elbows (b,*ii*) and other hand sized clutter (b,*iv*). The second-stage Hough [251] forests typically provide poorer finger-tip localization deeper inside the hand silhouette; here (b,*i*) they confuse the ring and middle finger because without global context the local votes are noisy and unspecific. NN exemplars most often succeeded in localizing the hand while the deep model [155] more accurately estimated precise hand pose. See Sec. 5.5 for further discussion.

distance of 1100mm), we encountered an average annotation disagreement of about 20mm. For only nearby hands ($\leq$ 750mm from the camera, with an average distance of 550mm) we encountered an average annotation disagreement of about 10mm. The ICL dataset [220] exhibits similar annotation inconsistencies at similar ranges [155]. For hands at an average distance 235mm from the camera, [157] reduced annotation disagreements to approximately 4mm. This suggests that distance (which is inversely proportional to resolution) directly relates to annotation accuracy. Fig. 2.10 illustrates two examples of annotator disagreement on our test set.

2.5.3 Interpretation

We've described our proposed quantitative evaluation protocol. But, how can someone new to field of hand-pose estimation interpret these numbers? First, we provide visualizations and qualitative interpretations for our proposed metric. Second, we examine the quantitative performance required for various hand gesture recognition tasks.

Error thresholds: Much past work considers performance at fairly low error thresholds, approaching 10mm [251, 220, 228]. Interestingly, [155] show that established benchmarks such as the **ICL** test set include annotation errors of above 10mm in over a third of their frames. Ambiguities arise from manual labeling of joints versus bones and centroids versus surface points. We rigorously evaluate human-level performance through inter-annotator agreement on our new test set (Fig. 2.20). Overall, we find that max-errors of **20mm** approach the limit of human accuracy for closeby hands. We present a qualitative visualization of max error at different thresholds in Fig. 2.7. **50mm** appears consistent with a roughly correct pose, while an error within **100mm** appears consistent with a correct detection. Our qualitative analysis is consistent with empirical studies of human grasp [30] and gesture communication [214], which also suggest that a max-joint difference of 50mm differentiates common gestures and grasps. But in general, precision requirements depend greatly on the application; So we plot each method's performance across a broad range of thresholds (Fig. 2.8). We highlight 50 and 100mm thresholds for additional analysis.

Vocabulary size versus threshold: To better interpret max-error-thresholds, we ask "for a discrete vocabulary of N poses, what max-joint-error precision will suffice?". Intuitively, larger pose vocabularies require greater precision. To formalize this notion, we assume the user always perfectly articulates one of N poses from a discrete vocabulary Θ, with $|\Theta| = N$. Given a fixed vocabulary Θ, a recognition system needs to be precise within `prec` mm to avoid confusing any two poses from Θ:

$$\texttt{prec} < \min_{\theta_1 \in \Theta, \theta_2 \in \Theta} \frac{\text{dist}(P(\theta_1) - P(\theta_2))}{2} \tag{2.7}$$

where θ_1 and θ_2 represent two poses in Θ, $P(\theta)$ projects the pose θ's joints into metric space, and dist gives the maximum metric distance between the corresponding joints from each pose. To find the minimum precision required for each N, we construct a maximally distinguishable vocabulary Θ by maximizing the value of `prec`, subject to the kinematic constraints of libhand. Finding this most distinguishable pose vocabulary is an NP-hard problem. So, we take a greedy approach to optimize a vocabulary Θ for each vocabulary size N.

2.6 Results

We now report our experimental results, comparing datasets and methods. We first address the "state of the problem": what aspects of the problem have been solved, and what remain open research questions? Fig. 2.11 qualitatively characterizes our results. We conclude by discussing the specific lessons we learned and suggesting directions for future systems.

Mostly-solved (distinct poses): Fig. 2.17 shows that coarse hand pose estimation is viable on datasets of uncluttered scenes where hands face the camera (i.e. ICL). Deep models, decision forests, and NN all perform quite well, both in terms of articulated pose estimation (85% of frames are within 50mm max-error) and hand detection (100% are within 100mm max-error). Surprisingly, NN outperforms decision forests by a bit. However, when NN is trained on other datasets with larger pose variation, performance is considerably worse. This suggests that the test poses closely resemble the training poses. Novel poses (those not seen in training data) account for most of the remaining failures. More training data (perhaps user-specific) or better model generalization should correct these remaining failures. Yet, this may be reasonable for applications targeting sufficiently distinct poses from a small and finite vocabulary (e.g., a gaming interface). These results suggest that *the state-of-the-art can accurately predict distinct poses* (i.e., 50 mm apart) *in uncluttered scenes.*

Major progress (unconstrained poses): The NYU test set still considers isolated hands, but includes a wider range of poses, viewpoints, and subjects compared to ICL (see Fig. 2.2). Fig. 2.18 reveals that deep models perform the best for both articulated pose estimation (96% accuracy) and hand detection (100% accuracy). While decision forests struggle with the added variation in pose and viewpoint, NN still does quite well. In fact, when measured with average (rather than max) error, NN nearly matches the performance of [228]. This suggests that exemplars get most, but not all fingers, correct (see Fig. 2.13 and *cf.* Fig. 2.11 (c,*ii*) versus (d,*ii*)). Overall, *we see noticeable progress on unconstrained pose estimation since 2007* [55].

Unsolved (low-res, objects, occlusions, clutter): When considering our test set (Fig. 2.20) with distant (low-res) hands and background clutter consisting of objects or interacting surfaces (Fig. 2.14), results are significantly worse. Note that many applications [202] often

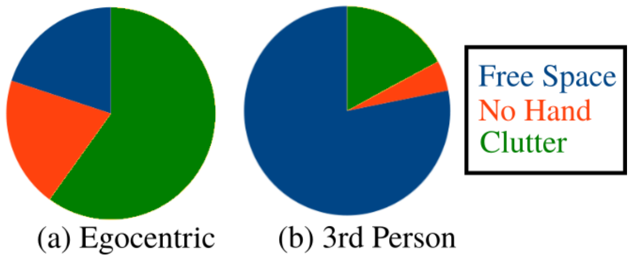


Figure 2.12: **Egocentric versus 3rd person challenges:** A robust hand-pose estimator must contend with isolated hands in `free space`, frames with `no hands` visible, and hands grasping objects in `cluttered` scenes. Uniformly sampling frames from the test data in Table 2.1 we show the distribution of challenges for both Egocentric (UCI-EGO and CVAR-EGO) and 3rd person test sets. Empirically, egocentric data contains more object manipulation and occlusion. In general, egocentric datasets target applications which involve significant clutter[183, 125, 61, 184]. While, 3rd person test sets historically focus on gesture recognition, involving less clutter.

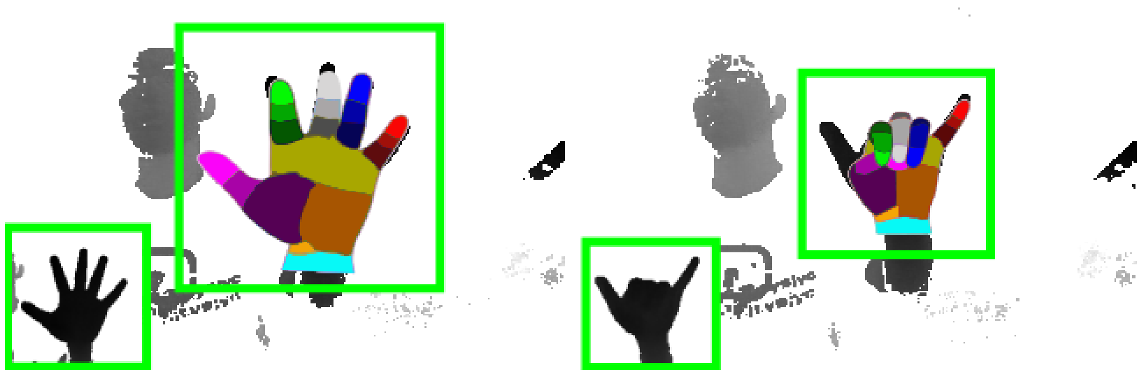


Figure 2.13: **Min versus max error:** Compared to state-of-the-art, our 1-NN baseline often does relatively better under the average-error criterion than under the max-error criterion. When it can find (nearly) an exact match between training and test data (left) it obtains very low error. However, it does not generalize well to unseen poses (right). When presented with a new pose it will often place some fingers perfectly but others totally wrong. The result is a reasonable mean error but a high max error.

demand hands to lie at distances greater than 750mm. For such scenes, hand detection is still a challenge. Scanning window approaches (such as our NN baseline) tend to outperform multistage pipelines [99, 59], which may make an unrecoverable error in the first (detection and segmentation) stage. We show some illustrative examples in Fig. 2.15. Yet, overall performance is still lacking, particularly when compared to human performance. Notably, human (annotator) accuracy also degrades for low-resolution hands far away from the camera (Fig. 2.20). This annotation uncertainty ("Human" in Fig. 2.20) makes it difficult to compare methods for highly precise pose estimation. As hand pose estimation systems become more precise, future work must make test data annotation more precise [157]. Our results suggest that *scenes of in-the-wild hand activity are still beyond the reach of the state of the art*.

Unsolved (Egocentric): The egocentric setting commonly presents (Fig. 2.19) the same problems discussed before, with the exception of low-res. While egocentric images do not necessarily contain clutter, most datasets in this area target applications with significant clutter (see Fig. 2.12). And, in some sense, egocentric views make hand detection fundamentally harder. We cannot merely assume that the nearest pixel in the depth image corresponds to the hand, as we can with many 3rd person gesture test sets. In fact, the *forearm* often provides the primary salient feature. In Fig. 2.11 (c-d,*iii*) both the deep and the 1-NN models need the arm to estimate the hand position. But, 1-NN wrongly predicts that the palm faces downwards, not towards the coffee maker. With such heavy occlusion and clutter, these errors are not surprising. The deep model's detector [228, 155] proved less robust in the egocentric setting. Perhaps it developed sensitivity to changes in noise patterns, between the synthetic training and real test datasets. But, the NN and deep detectors wrongly assume translation-invariance for egocentric hands. Hand appearance and position are linked by perspective effects coupled with the kinematic constraints imposed by the arm. As a result, an egocentric-specific whole volume classification model [183] outperformed both.

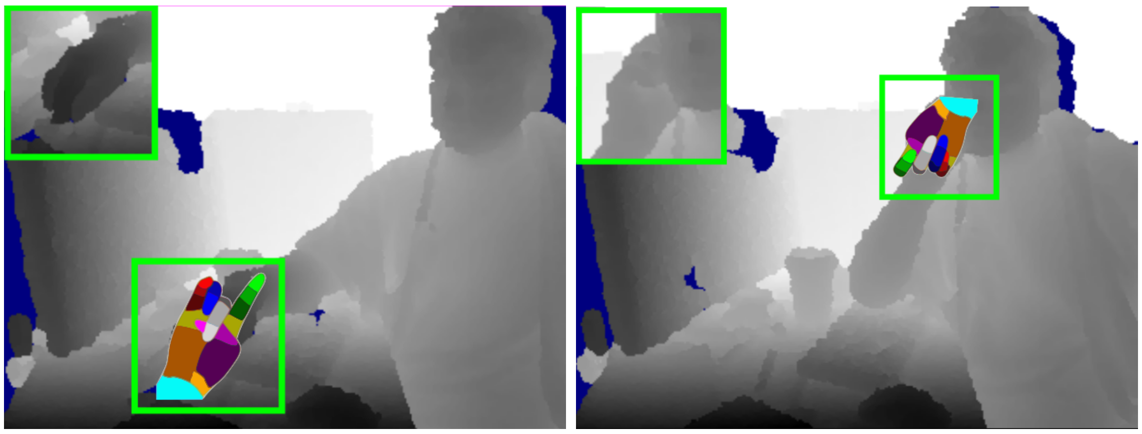
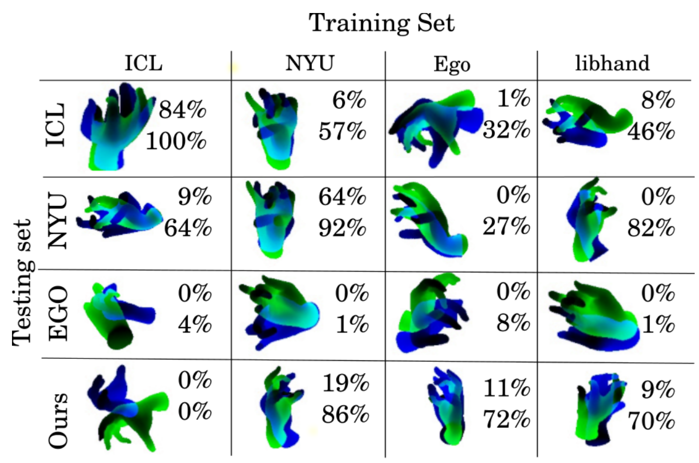


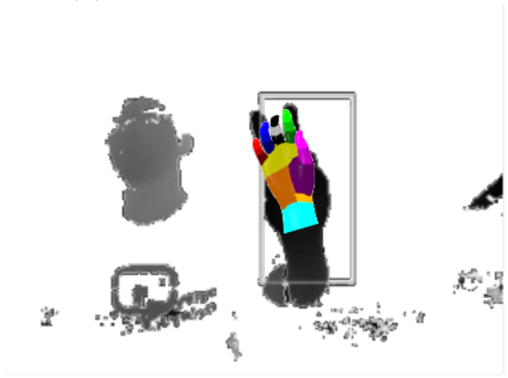
Figure 2.14: **Complex backgrounds:** Most existing systems, including our own 1-NN baseline, fail when challenged with complex backgrounds which cannot be trivially segmented. These backgrounds significantly alter the features extracted and processed and thus prevent even the best models from producing sensible output.



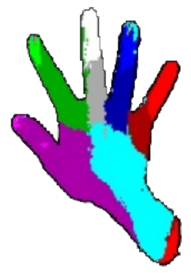
		Training Set			
		ICL	NYU	Ego	libhand
Testing set	ICL	84% 100%	6% 57%	1% 32%	8% 46%
	NYU	9% 64%	64% 92%	0% 27%	0% 82%
	EGO	0% 4%	0% 1%	0% 8%	0% 1%
	Ours	0% 0%	19% 86%	11% 72%	9% 70%

Table 2.5: **Cross-dataset generalization:** We compare training and test sets using a 1-NN classifier. Diagonal entries represent the performance using corresponding train and test sets. In each grid entry, we denote the percentage of test frames that are correct (50mm max-error, above, and 50mm average-error, below) and visualize the median error using the colored overlays from Fig. 2.7. We account for sensor specific noise artifacts using established techniques [33]. Please refer to the text for more details.

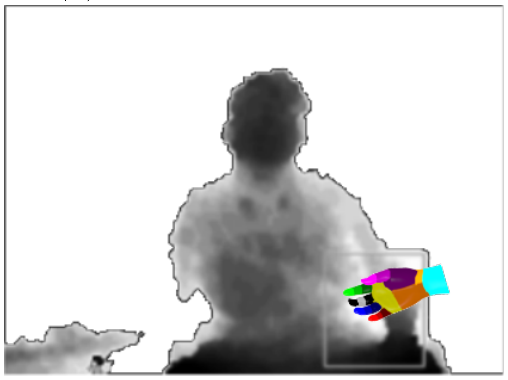
(a) Latent Hough detection



(c) per-pixel classification



(b) Hough orientation failure



(d) hard segmentation

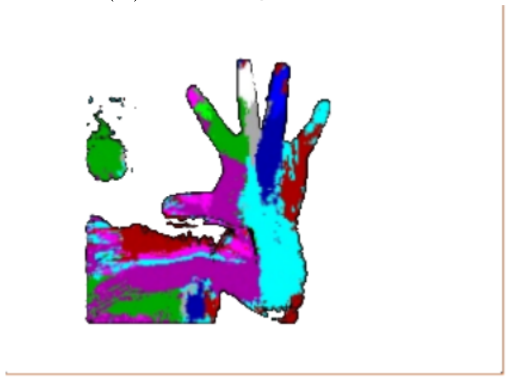


Figure 2.15: **Risks of multi-phase approaches:** Many approaches to hand pose estimation divide into three phases: (1) detect and segment (2) estimate pose (3) validate or refine [99, 87, 251, 228, 220]. However, when an earlier stage fails, the later stages are often unable to recover. When detection and segmentation are non-trivial, this becomes to root cause of many failures. For example, Hough forests [251] (a) first estimate the hand's location and orientation. They then convert to a cardinal translation and rotation before estimating joint locations. (b) When this first stage fails, the second stage cannot recover. (c) Other methods assume that segmentation is solved [99, 59], (d) when background clutter is inadvertently included by the hand segmenter, the finger pose estimator is prone to spurious outputs.

Training data: We use our NN-baseline to analyze the effect of training data in Table 2.5. Our NN model performed better using the NYU training set [228] (consisting of real data automatically labeled with a geometrically-fit 3D CAD model) than with the libhand training set. While enlarging the synthetic training set increases performance (Fig. 2.16), computation fast becomes intractable. This reflects the difficulty in using synthetic data: one must carefully model priors [155], sensor noise, [74] and hand shape variations between users [225, 101]. In Fig. 2.21 we explore the impact of each of these factors to uncover two salient conclusions: First, training with the test-time user's hand geometry (user-specific training data) showed modestly better performance, suggesting that results may be optimistic when using the same subjects for training and testing. Second, for synthetic hand data, modeling the pose-prior (i.e. , choosing likely poses to synthesize) overshadows other considerations. Finally, in some cases, the variation in the performance of NN (dependent on the particular training set) exceeded the variation between model architectures (decision forests versus deep models) - Fig. 2.17. Our results suggest the diversity and realism of the *training set is as important as the model learned from it.*

Surprising NN performance: Overall, our 1-NN baseline proved to be surprisingly potent, outperforming or matching the performance of most prior systems. This holds true even for moderately-sized training sets with tens of thousands of examples [228, 220], suggesting that simple memorization outperforms much prior work. To demonstrate generalization, future work on learning based methods will likely benefit from more and better training data. One contribution of our analysis is the notion that *NN-exemplars provides a vital baseline for understanding the behavior of a proposed system in relation to its training set.*

NN versus Deep models: In fact, DeepJoint [228] and DeepPrior [155] were the sole approaches to significantly outperform 1-NN (Figs. 2.17 and 2.18). This indicates that deep architectures generalize well to novel test poses. Yet, the deep model [155] did show greater sensitivity to objects and clutter than the 1-NN model. We see this qualitatively in Fig. 2.11 (c-d,*iii*-*iv*) and quantitatively in Figs. 2.20 and 2.19. But, we can understand the deep model's failures: we did not train it with clutter, so it "generalizes" that the bottle and hand are a single large hand. This may contrast with existing folk wisdom about deep models: that the need for large training sets suggests that these models essentially memorize. Our results indicate otherwise. Finally, the deep model performed worse on more distant hands; this is understandable because it requires a larger canonical template (128x128) than the 1-NN model (30x30).

Conclusion: The past several years have shown tremendous progress regarding hand pose: training sets, testing sets, and models. Some applications, such as gaming interfaces and sign-language recognition, appear to be well-within reach for current systems. Less than a decade ago, this was not true [55, 174, 41]. Thus, we have made progress! But, challenges remain nonetheless. Specifically, when segmentation is hard due to active hands or clutter, many existing methods fail. To illustrate these realistic challenges we introduce a novel test set. We demonstrate that realism and diversity in training sets is crucial, and can be as important as the choice of model architecture. Thus, future work should investigate building large, realistic, and diverse training sets. In terms of model architecture, we perform a broad benchmark evaluation and find that deep models appear particularly well-suited for pose estimation. Finally, we demonstrate that NN using volumetric exemplars provides a startlingly potent baseline, providing an additional tool for analyzing both methods and datasets.

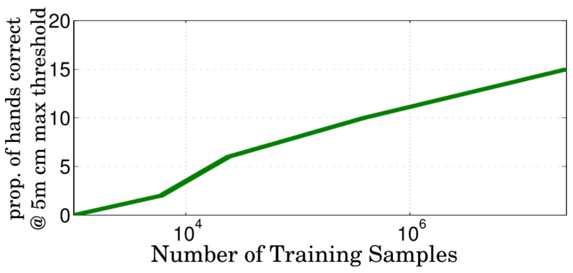


Figure 2.16: **Synthetic data versus accuracy**: Synthetic training set size impacts performance on our test test set. Performance grows logarithmically with the dataset size. Synthesis is theoretically unlimited, but practically becomes unattractively slow.

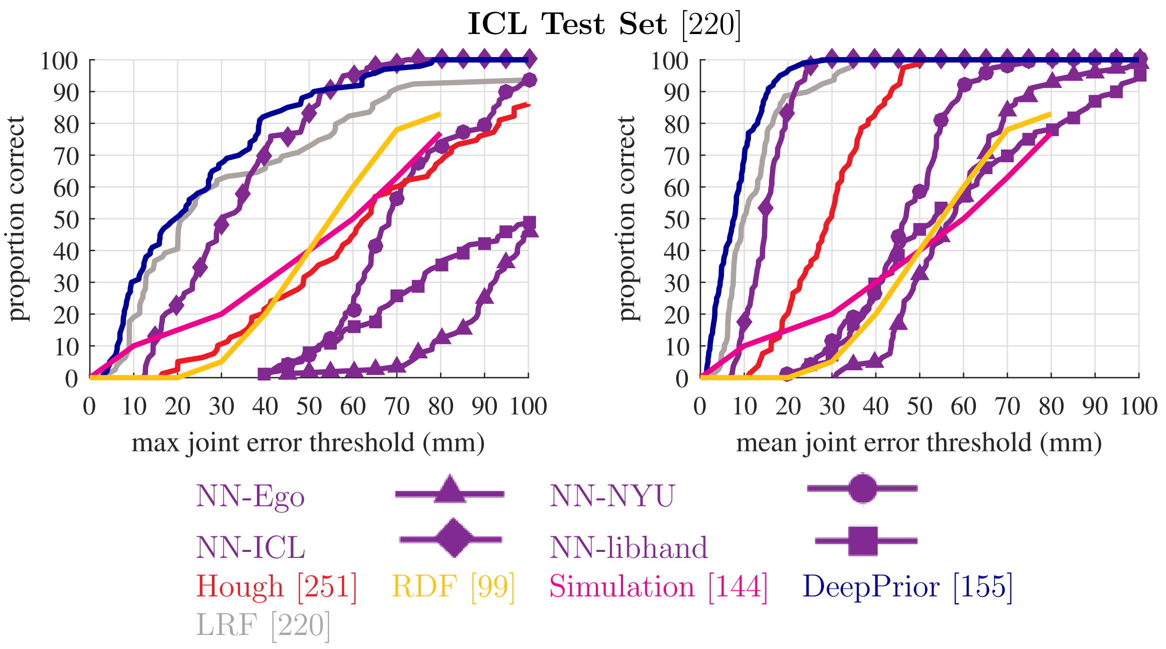


Figure 2.17: We plot results for several systems on the ICL test set using max-error (top) and average-error (bottom). Except for 1-NN, all systems are trained on the corresponding train set (in this case ICL-Train). To examine cross-dataset generalization, we also plot the performance of our NN-baseline constructed using alternate sets (NYU, EGO, and libhand). When trained with ICL, NN performs as well or better than prior art. One can find near-perfect pose matches in the training set (see Fig. 2.1). See the text for further discussion.

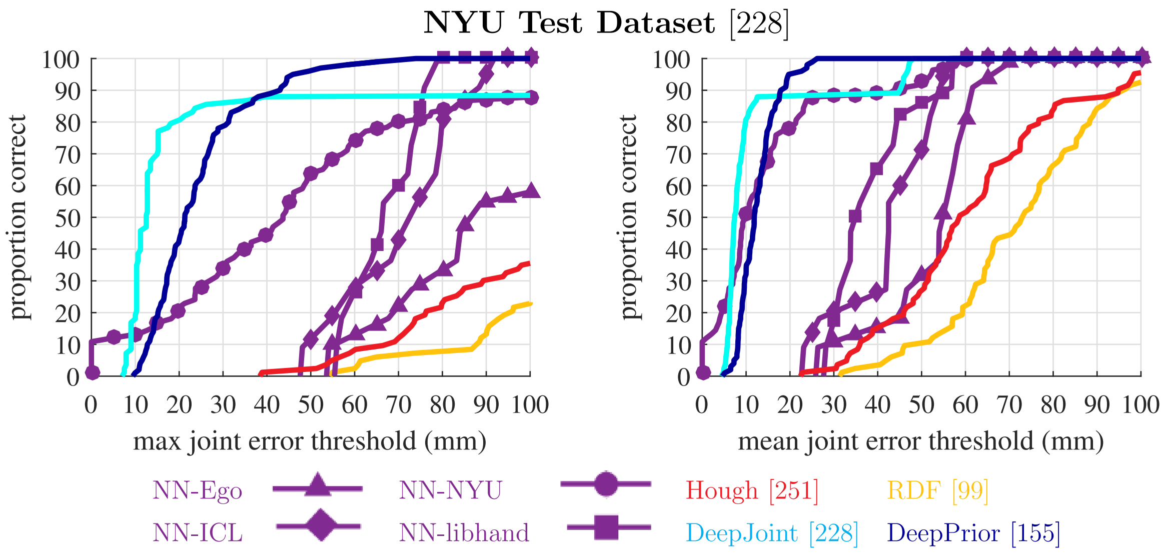


Figure 2.18: Deep models [228, 155] perform noticeably better than other systems, and appear to solve both articulated pose estimation and hand detection for uncluttered single-user scenes (common in the NYU test set). However, the other systems compare more favorably under average error. In Fig. 2.13, we interpret this disconnect by using 1-NN to show that each test hand commonly matches a training example in all but one finger. See the text for further discussion.

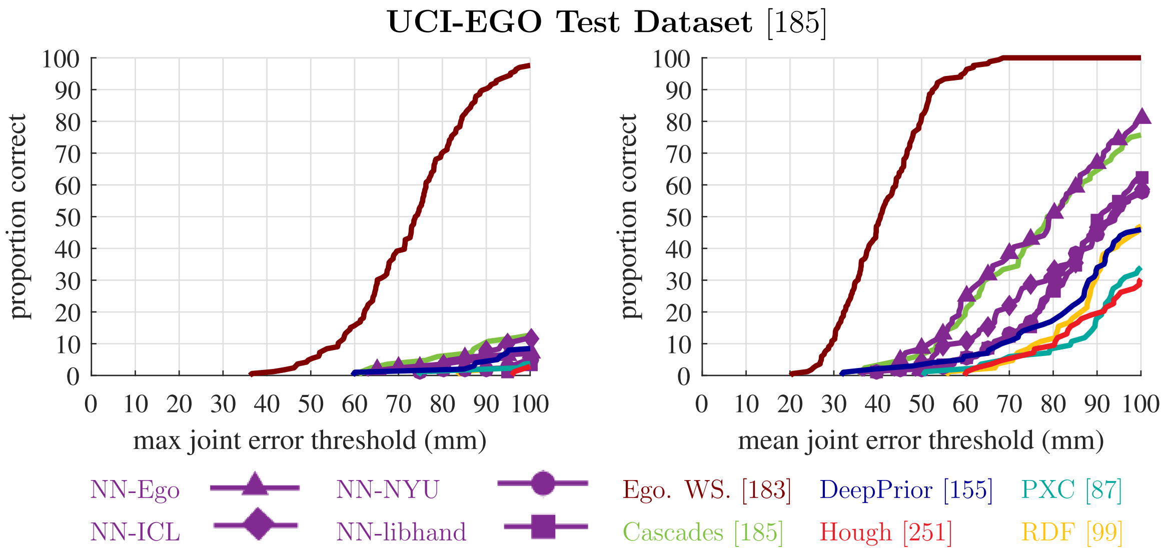


Figure 2.19: For egocentric data, methods that classify the global scene [183] tend to outperform local scanning-window based approaches (including both deep and NN detectors). Rogez at al [183] make the argument that kinematic constrains from the arm imply that the location of the hand (in an egocentric coordinate frame) effects its local orientation and appearance, which in turn implies that recognition should not be translation-invariant. Still overall, performance is considerably worse than on other datasets. Egocentric scenes contain more background clutter and object/surface interactions, making even hand detection challenging for most methods.

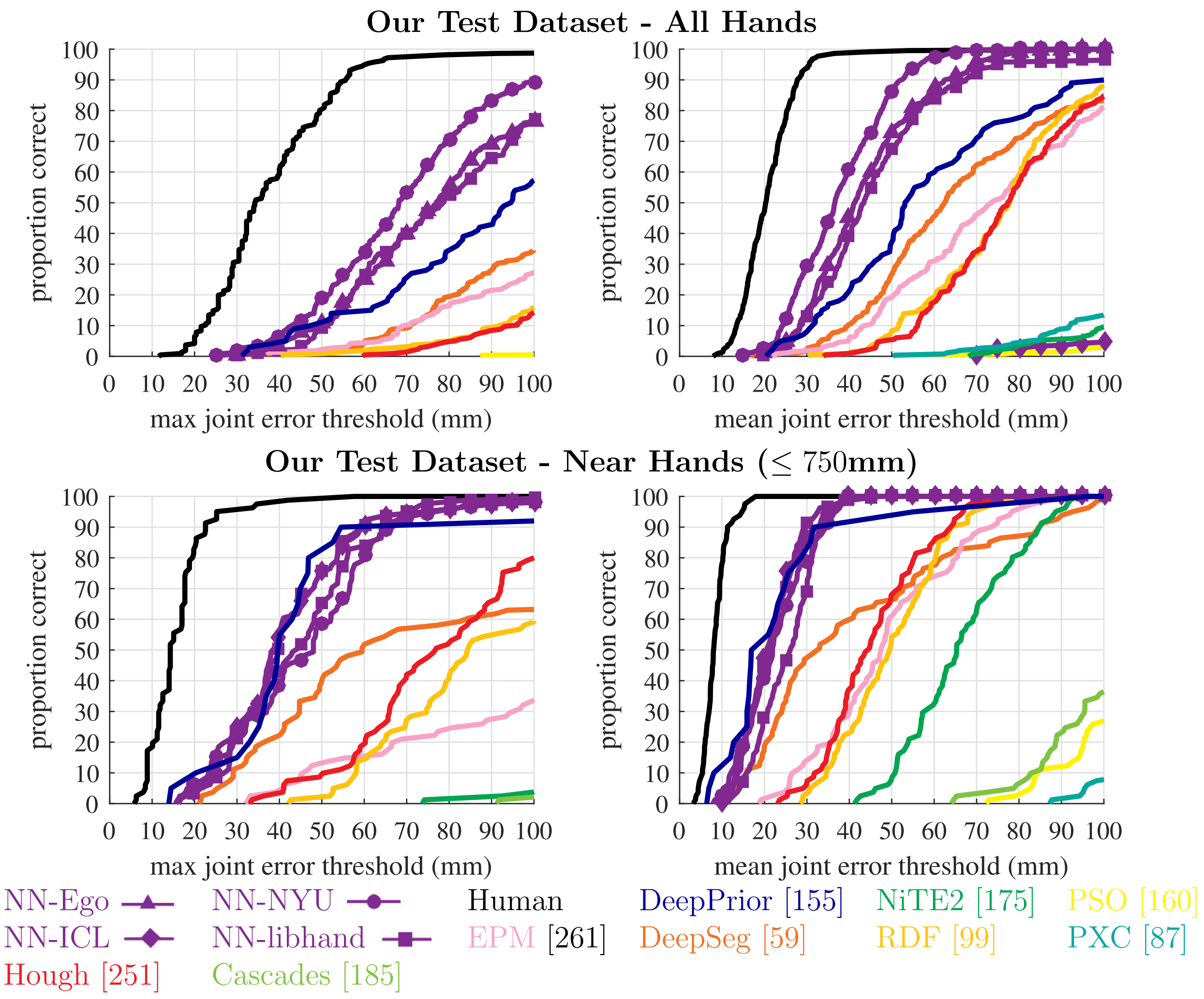


Figure 2.20: We designed our dataset to address the remaining challenges of in "in-the-wild" hand pose estimation, including scenes with low-res hands, clutter, object/surface interactions, and occlusions. We plot human-level performance (as measured through inter-annotator agreement) in black. On nearby hands (within 750mm, as commonly assumed in prior work) our annotation quality is similar to existing test sets such as ICL [155]. This is impressive given that our test set includes comparatively more ambiguous poses (see Sec. 2.5.2). Our dataset includes far away hands, for which even humans struggle to accurately label. Moreoever, several methods (Cascades,PXC,NiTE2,PSO) fail to correctly localize any hand at any distance, though the mean-error plots are more forgiving than the max-error above. In general, NN-exemplars and DeepPrior perform the best, correctly estimating pose on 75% of frames with nearby hands.

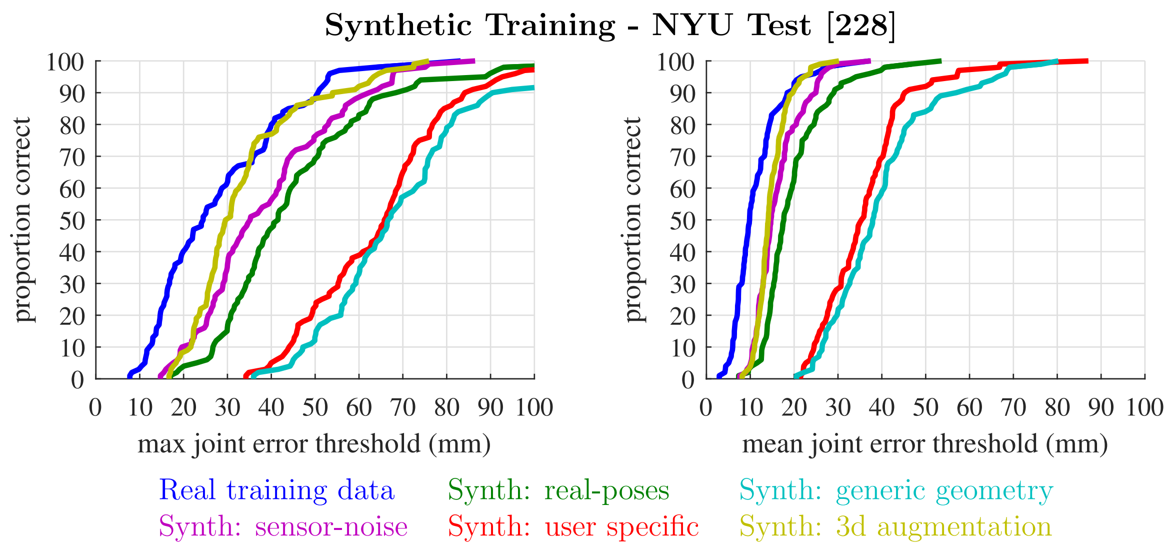


Figure 2.21: **Challenges of synthetic data:** We investigate possible causes for our synthetic training data's lackluster performance. To do so, we synthesize a variety of training sets for a deep model [155] and test on the NYU test set. Clearly, real training data (blue) outperforms our generic synthetic training set (cyan), as described in Sec. 2.3.1). By fitting our synthesis model's geometry to the test-time users we obtain a modest gain (red). However, the largest gain by far comes from synthesizing training data using only "realistic" poses, matching those from the NYU training set. By additionally modeling sensor noise [74] we obtain the magenta curve. Finally, we almost match the real training data (yellow versus blue) by augmenting our synthetic models of real-poses with out-of-plane rotations and foreshortening.

Chapter 3

Models for Egocentric Hand Pose

Grégory Rogez, James Steven Supančič III & Deva Ramanan

3.1 Introduction

Our evaluation (in the previous chapter) showed that egocentric scenes remain a challenge. As opposed to 3rd person hand-pose recognition, egocentric views may be more difficult due to additional occlusions (from manipulated objects, or self-occlusions of fingers by the palm) and the fact that hands interact with the environment and often leave the field-of-view. The latter necessitates constant re-initialization, precluding the use of a large body of hand trackers which typically perform well given manual initialization.

Previous work for egocentric hand analysis tends to rely on local 2D features, such as pixel-level skin classification [128, 127] or gradient-based processing of depth maps with scanning-window templates [181]. Our approach follows in the tradition of [181], who argue that near-field depth measures obtained from a egocentric-depth sensor considerably simplifies hand analysis. Interestingly, egocentric-depth is not "cheating" in the sense that humans make

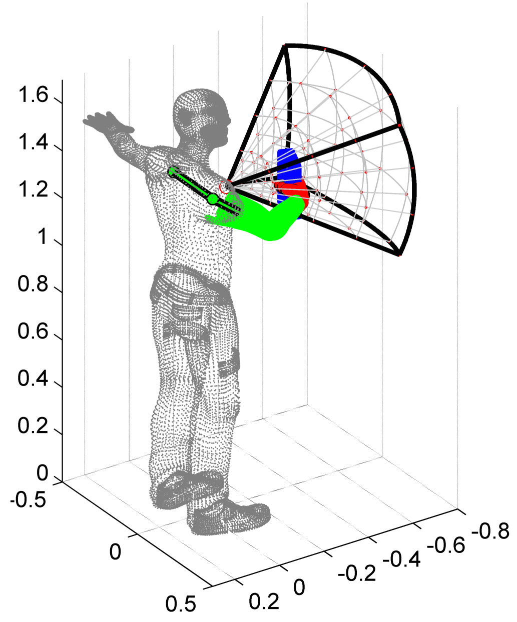


Figure 3.1: **Egocentric workspaces**. We directly model the observable egocentric workspace in front of a human with a 3D volumetric descriptor, extracted from a 2.5D egocentric depth sensor. In this example, this volume is discretized into $4 \times 3 \times 4$ bins. This feature can be used to accurately predict shoulder, arm, hand poses, even when interacting with objects. We describe models learned from synthetic examples of observable egocentric workspaces obtained by placing a virtual Intel Creative camera on the chest of an animated character.

use of stereoscopic depth cues for near-field manipulations [66]. We extend this observation by building an explicit 3D map of the observable near-field workspace.

Contributions: In this chapter, we describe a new computational architecture that makes use of **global** egocentric views, **volumetric** representations, and **contextual** models of interacting objects and human-bodies. Rather than detecting hands with a local (translation-invariant) scanning-window classifier, we process the entire global egocentric view (or *work-space*) in front of the observer (Fig. 6.1). Hand appearance is not translation-invariant due to perspective effects and kinematic constraints with the arm. To capture such effects, we build a library of synthetic 3D egocentric workspaces generated using real capture conditions (see examples in Fig. 3.2). We animate a 3D human character model inside virtual scenes with objects, and render such animations with a chest-mounted camera whose intrinsics match our physical camera . We simultaneously recognize arm and hand poses while interacting with objects by classifying the whole 3D volume using a multi-class Support Vector Machine (SVM) classifier. Our real-time recognition architecture is simple enough to be implemented in 4 lines of code.

3.1.1 Related work

Hand-object pose estimation: While there is a large body of work on hand-tracking [115, 109, 108, 9, 161, 212, 243], we focus on hand pose estimation during object manipulations. Object interactions both complicate analysis due to additional occlusions, but also provide additional contextual constraints (hands cannot penetrate object geometry, for example). Hamer et al. [77] describe articulated tracking with soft anti-penetration constraints, increasing robustness to occlusion. They later describe contextual priors for hands in relation to objects [76], and demonstrate their effectiveness for increasing tracking accuracy. Objects are easier to animate than hands because they have fewer pose parameters. With this intuition, object motion can be used as an input signal for estimating hand motions [75]. Romero et al. [189] use a large synthetic dataset of hands manipulating objects, similar to our work. We differ in our focus on single-image and egocentric analysis.

Egocentric Vision: Previous studies have focused on activities of daily living [171, 60]. Long-scale temporal structure was used to handle complex hand object interactions, exploiting the fact that objects look different when they are manipulated (active) versus not manipulated (passive) [171]. Much previous work on egocentric hand recognition make exclusive use of RGB cues [127, 126], while we focus on volumetric depth cues. Notable exceptions include Damen et al. [46], who employ egocentric RGB-D sensors for personal workspace monitoring in industrial environments and Mann et al. [138], who employ such sensors to assist blind users in navigation.

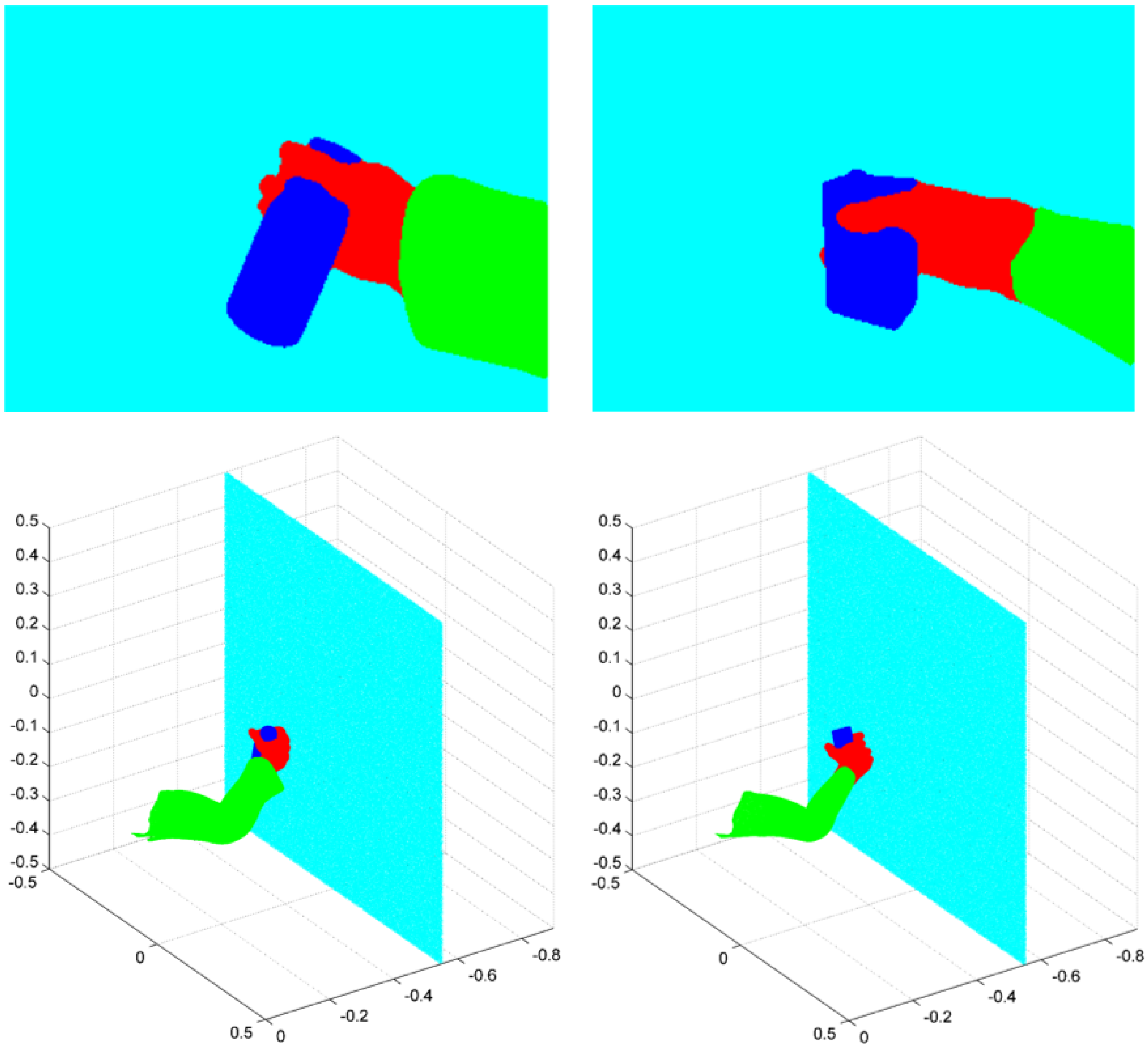


Figure 3.2: **Synthesis:** We generate a training set by sampling different dimensions of a workspace model, yielding a total number of $N_{arm} \times N_{hand} \times N_{object} \times N_{background}$ samples. We sample N_{arm} arm poses, a fixed set of hand-object configurations ($N_{hand} \times N_{object} = 100$) and a fixed set of $N_{background}$ background scenes captured with an Intel Creative depth camera. For each hand-object model, we randomly perturb shoulder, arm and hand joint angles to generate physically possible arm+hand+object configurations. We show 2 examples of a bottle-grasp (left) and a juice-box-grasp (right) rendered in front of a flat wall.

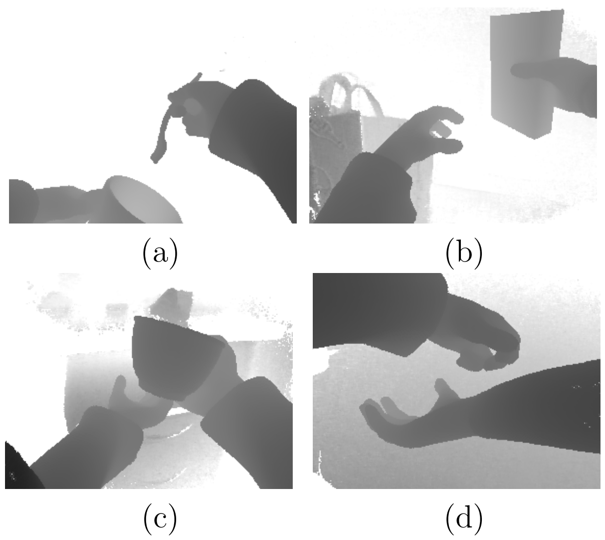


Figure 3.3: **Examples of synthetic training images**. Our rendering pipeline produces realistic depth maps consisting of multiple hands manipulating household objects in front of everyday backgrounds.

Depth features: Previous work has shown the efficacy of depth cues [201, 252]. We compute volumetric depth features from point clouds. Previous work has examined point-cloud processing of depth-images [254, 202, 252]. A common technique estimates local surface orientations and normals [254, 252], but this may be sensitive to noise since it requires derivative computations. We use simpler volumetric features, similar to [209] except that we use a spherical coordinate frame that does not slide along a scanning window (we want to measure depth in an egocentric coordinate frame).

Non-parametric recognition: Our work is inspired by non-parametric techniques that make use of synthetic training data [189, 199, 77, 44, 230]. Shakhnarovich et al. [199] make use of pose-sensitive hashing techniques for efficient matching of synthetic RGB images rendered with Poser. We generate synthetic depth images, mimicking capture conditions of our actual camera.

3.2 Training data

We begin by generating a training set of realistic 3D egocentric workspaces. Specifically, we render synthetic 3D hand-object data (generated from a 3D animation system) on top of real 3D background scenes, making use of the test camera projection matrix. Because egocentric scenes involve objects that lie close to the camera, we found it useful to model camera-specific perspective effects.

Poser models. Our in-house grasp database is constructed by modifying the commercial Everyday hands Grasp Poser library [49]. We vary the objects being interacted with, as well as the clothing of the character, i.e., with and without sleeves. We use more than 200 grasping postures and 49 objects, including kitchen utensils, personal bathroom items, office/classroom objects, fruits, etc. Additionally we use 6 models of empty hands: wave, fist, thumbs-up, point, open/close fingers. Some objects can be handled with different canonical grasps. For example, one can grip a bottle by its body or by its lid when opening it. We manually add such variants.

Kinematic model. Let θ be a vector of arm joint angles, and let ϕ be a vector of grasp-specific hand joint angles, obtained from the above set of Poser models. We use a standard forward kinematic chain to convert the location of finger joints $\mathbf{u}$ (in a local coordinate system) to image coordinates:

$$\begin{aligned} \mathbf{p} &= C\prod_i T(\theta_i)\prod_j T(\phi_j)\mathbf{u}, \quad \text{where} \quad T, C \in \mathbb{R}^{4\times4}, \\ \mathbf{u} &= \begin{bmatrix} u_x & u_y & u_z & 1 \end{bmatrix}^T, \quad (x,y) = (f\frac{p_x}{p_z}, f\frac{p_y}{p_z}), \end{aligned} \tag{3.1}$$

where T specifies rigid-body transformations (rotation and translation) along the kinematic chain and C specifies the extrinsic camera parameters. Here $\mathbf{p}$ represents the 3D position of point $\mathbf{u}$ in the camera coordinate system. To generate the corresponding image point, we assume camera intrinsics are given by identity scale factors and a focal length f (though it is straightforward to use more complex intrinsic parameterizations). We found it important to use the f corresponding to our physical camera, as it is crucial to correctly model perspective effects for our near-field workspaces.

Pose synthesis: We wish to generate a large set of postured hands. However, building a generative model of grasps is not trivial. One option is to take a data-driven approach and collect training samples using motion capture [173]. Instead, we take a model-driven approach that perturbs a small set of manually-defined canonical postures. To ensure that physically plausible perturbations are generated, we take a simple *rejection sampling* approach. We fix ϕ parameters to respect the hand grasps from Poser, and add small Gaussian perturbations to arm joint angles

$$\theta_i' = \theta_i + \epsilon \quad \text{where} \quad \epsilon \sim N(0, \sigma^2).$$

Importantly, this generates hand joints $\mathbf{p}$ at different translations and viewpoints, correctly modeling the dependencies between both. For each perturbed pose, we render hand joints using (3.1) and keep exemplars that are 90% visible (e.g., their projected (x, y) coordinates lie within the image boundaries). We show examples in Fig. 3.2.

Depth maps. Associated with each rendered set of keypoints, we would also like a depth map. To construct a depth map, we represent each rigid limb with a dense cloud of 3D vertices $\{\mathbf{u}_i\}$. We produce this cloud by (over) sampling the 3D meshes defining each rigid-body shape. We render this dense cloud using forward kinematics (3.1), producing a set of points $\{\mathbf{p}_i\} = \{(p_{x,i}, p_{y,i}, p_{z,i})\}$. We define a 2D depth map $z[u, v]$ by ray-tracing. Specifically, we cast a ray from the origin, in the direction of each image (or depth sensor) pixel location

(u, v) and find the closest point:

$$z[u, v] = \min_{k \in \text{Ray}(u,v)} ||\mathbf{p}_k|| \tag{3.2}$$

where $\text{Ray}(u, v)$ denotes the set of points on (or near) the ray passing through pixel (u, v). We found the above approach simpler to implement than hidden surface removal, so long as we projected a sufficiently dense cloud of points.

Multiple hands: Some object interactions require multiple hands interacting with a single object. Additionally, many views contain the second hand in the "background". For example, two hands are visible in roughly 25% of the frames in our benchmark videos. We would like our training dataset to have similar statistics. Our existing Poser library contains mostly single-hand grasps. To generate additional multi-arm egocentric views, we randomly pair 25% of the arm poses with a mirrored copy of another randomly-chosen pose. We then add noise to the arm joint angles, as described above. Such a procedure may generate unnatural or self-intersecting poses. To efficiently remove such cases, we separately generate depth maps for the left and right arms, and only keep pairings that produce compatible depth maps:

$$|z_{left}[u, v] - z_{right}[u, v]| > \delta \quad \forall u, v \tag{3.3}$$

We find this simple procedure produces surprisingly realistic multi-arm configurations (Fig. 3.3). Finally we add background clutter from depth maps of real egocentric scenes (not from our benchmark data). We use the above approach to generate over 100,000 multi-hand(+arm+objects) configurations and associated depth-maps.

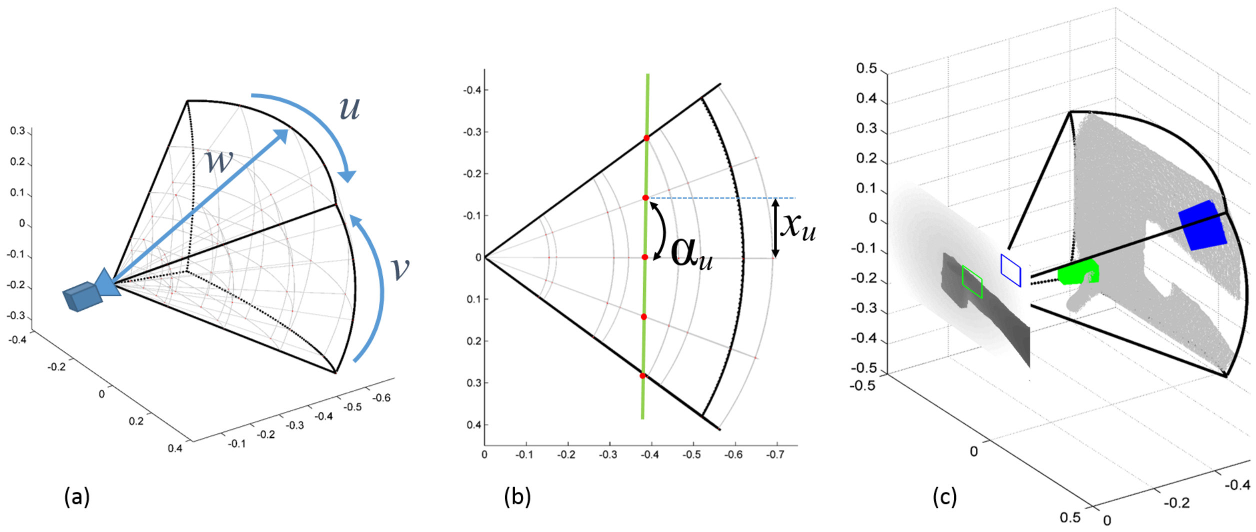


Figure 3.4: **Volume quantization.** We quantize those points that fall within the egocentric workspace (observable volume within z_{max} = 70cm) into a binary spherical voxel grid of $N_u \times N_v \times N_w$ voxels (a). We vary the azimuth angle α to generate equal-size projections on the image plane (b). Spherical bins ensure that voxels at different distances project to same image area (c). This allows for efficient feature computation and occlusion handling, since occluded voxels along the same line-of-sight can easily be identified.

3.3 Formulation

3.3.1 Perspective-aware depth features

It may seem attractive to work in orthographic (or scaled orthographic) coordinates, as this simplifies much of 3D analysis. Instead, we posit that perspective distortion is useful in egocentric settings and should be exploited: objects of interest (hands, arms, and manipulated things) tend to lie near the body and exhibit perspective effects. Specifically, parts of objects that are closer to the camera project to a larger image size. To model such effects, we construct a spherical bin histogram by gridding up the egocentric workspace volume by varying azimuth and elevation angles (Fig. 3.4). We demonstrate that this feature outperforms orthographic counterparts, and is also faster to compute.

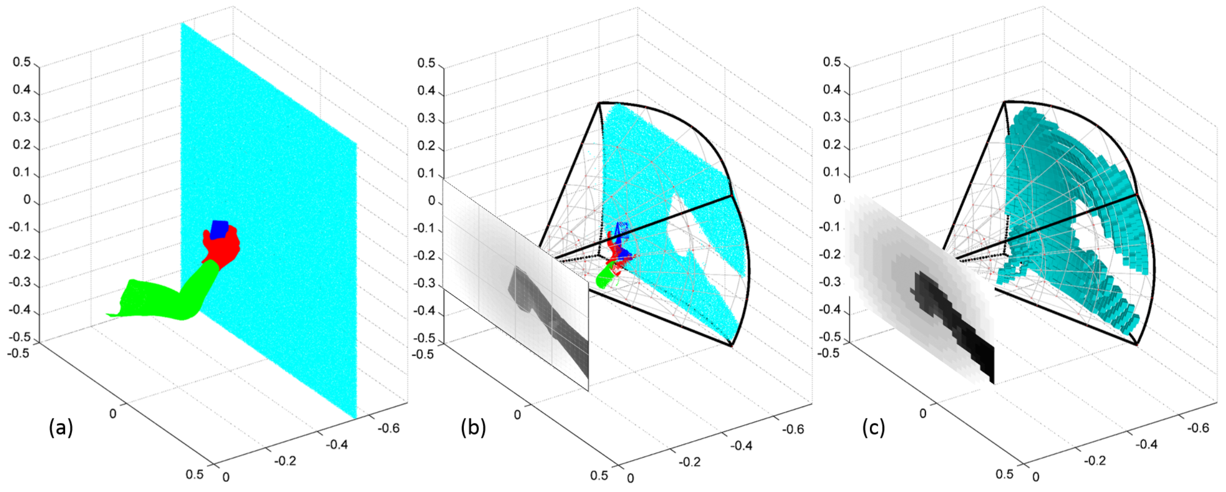


Figure 3.5: **Binarized volumetric feature.** We synthesize training examples by randomly perturbing shoulder, arm and hand joint angles in a physically possible manner (a). For each example, a synthetic depth map is created by projecting the visible set of dense 3D points using a real-world camera projection matrix (b). The resulting 2D depth map is then quantized with a regular grid in x-y directions and binned in the viewing direction to compute our new binarized volumetric feature (c). In this example, we use a $32 \times 24 \times 35$ grid. Note that for clarity we only show the sparse version of our 3D binary feature. We also show the quantized depth map $z[u, v]$ as a gray scale image (c).

Binarized volumetric features: Much past work processes depth maps as 2D rasterized sensor data. Though convenient for applying efficient image processing routines such as gradient computations (e.g., [222]), rasterization may not fully capture the 3D nature of the data. Alternatively, one can convert depth maps to a full 3D point cloud [119], but the result is orderless, making operations such as correspondence-estimation difficult. We propose encoding depth data in a 3D volumetric representation, similar to [209]. To do so, we can back-project the depth map from (3.2) into a cloud of visible 3D points $\{\mathbf{p}_k\}$, visualized in Fig. 3.5-(b). They are a subset of the original cloud of 3D points $\{\mathbf{p}_i\}$ in Fig. 3.5-(a). We now bin those visible points that fall within the egocentric workspace in front of the camera (observable volume within $z_{max} = 70$cm) into a binary voxel grid of $N_u \times N_v \times N_w$ voxels:

$$b[u, v, w] = \begin{cases} 1 & \text{if} \quad \exists k \text{ s.t.} \quad \mathbf{p}_k \in F(u, v, w) \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

where $F(u, v, w)$ denotes the set of points within a voxel centered at coordinate (u, v, w).

Spherical voxels: Past work tends to use rectilinear voxels [209, 119]. Instead, we use a spherical binning structure, centering the sphere at the camera origin (Fig. 3.4). At first glance, this might seem strange because voxels now vary in size – those further away from the camera are larger. The main advantage of a "perspective-aware" binning scheme is that all voxels now project to the same image area in pixels (Fig. 3.4-(c)). We will show that this both increases accuracy (because one can better reason about occlusions) and speed (because volumetric computations are sparse).

Efficient quantization: Let us choose spherical bins $F(u, v, w)$ such that they project to a single pixel (u, v) in the depth map. This allows one to compute the binary voxel grid $b[u, v, w]$ by simply "reading off" the depth value for each $z(u, v)$ coordinates, quantizing it to z', and assigning 1 to the corresponding voxel:

$$b[u, v, w] = \begin{cases} 1 \quad \text{if} \qquad w = z'[u, v] \\ 0 \qquad \text{otherwise} \end{cases} \tag{3.5}$$

This results in sparse volumetric voxel features, as visualized in Fig. 3.5-(c). Crucially, a spherical parameterization allows one to efficient reason about occlusions: once a depth measurement is observed at position $b[u', v', w'] = 1$, all voxels behind it are occluded for $w \geq w'$. This arises from the fact that single camera depth measurements are, in fact, 2.5D. By convention, we define occluded voxels to be "1". Note that such occlusion reasoning is difficult with orthographic parameterizations because voxels are not arranged along line-of-sight rays.

In practice, we consider a coarse discretization of the volume to make the problem more tractable. The depth map $z[x, y]$ is resized to $N_u \times N_v$ (smaller than depth map size) and quantized in the z-direction. To minimize the effect of noise when counting the points which fall in the different voxels, we quantize the depth measurements by applying a median filter on the pixel values within each image region:

$$\begin{aligned} &\forall u, v \in [1, N_u] \times [1, N_v], \\ z'[u,v] &= \tfrac{N_w}{z_{max}} \text{median}(z[x,y] : (x,y) \in P(u,v)), \end{aligned} \tag{3.6}$$

where $P(u,v)$ is the set of pixel coordinates in the original depth map corresponding to pixel coordinate (u,v) coordinates in the resized depth map.

3.3.2 Global pose classification

We quantize the set of poses from our synthetic database into K coarse classes for each limb, and train a K-way pose-classifier for pose-estimation. The classifier is linear and makes use of our sparse volumetric features, making it quite simple and efficient to implement.

Pose space quantization: For each training exemplar, we generate the set of 3D keypoints: 17 joints (elbow + wrist + 15 finger joints) and the 5 finger tips. Since we want to recognize coarse limb (arm+hand) configurations, we cluster the resulting training set by applying K-means to the elbow+wrist+knuckle 3D joints. We usually represent each of the K resulting clusters using the average 3D/2D keypoint locations of both arm+hand (See examples in Fig. 3.6). Note that K can be chosen as a compromise between accuracy and speed.

Global classification: We use a linear SVM for a multi-class classification of upper-limb poses. However, instead of classifying local scanning windows, we classify global depth maps quantized into our binarized depth feature $b[u,v,w]$ from (3.5). Global depth maps allow the classifier to exploit contextual interactions between multiple hands, arms and objects. In particular, we find that modeling arms is particularly helpful for detecting hands. For each class $k \in \{1, 2, ...K\}$, we train a one-vs-all SVM classifier to obtain a weight vector

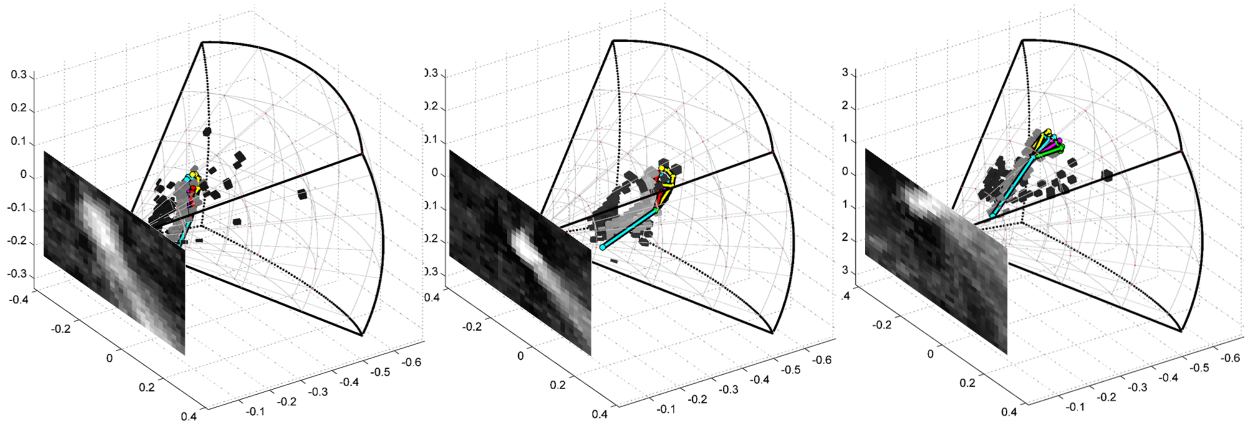


Figure 3.6: **Pose classifiers.** We visualize the linear weight tensor $\beta_k[u, v, w]$ learnt by the SVM for a $32 \times 24 \times 35$ grid of binary features for 3 different pose clusters. We plot a 2D (u, v) visualization obtained by computing the max along w. We also visualize the corresponding average 3D pose in the egocentric volume together with the top 500 positive (light gray) and negative weights (dark gray) within $\beta_k[u, v, w]$.

which can be re-arranged into a $N_u \times N_v \times N_w$ tensor $\beta_k[u, v, w]$. The score for class k is then obtained by a simple dot product of this weight and our binarized feature $b[u, v, w]$:

$$\text{score}[k] = \sum_u \sum_v \sum_w \beta_k[u, v, w] \cdot b[u, v, w]. \tag{3.7}$$

We visualize projections of the learned weight tensor $\beta_k[u, v, w]$ in Fig. 3.6 and slices of the tensor in Fig. 3.7.

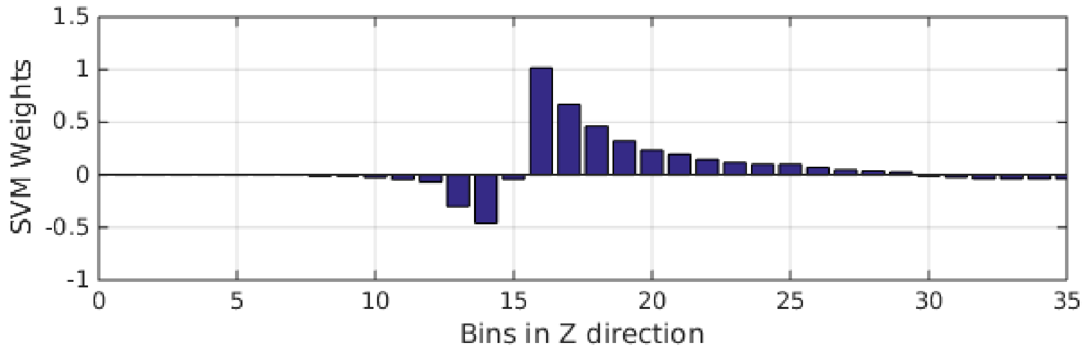


Figure 3.7: **Weights along w**. We visualize the SVM weights $\beta_k[u, v, w]$ for a particular (u, v) location. Our histogram encoding allows us to learn smooth nonlinear functions of depth values. For example, the above weights respond positively to depth values midway into the egocentric volume, but negatively to those closer.

3.3.3 Joint feature extraction and classification

To increase run-time efficiency, we exploit the sparsity of our binarized volumetric feature and jointly implement feature extraction and SVM scoring. Since the final score is a simple dot product with binary features, one can readily extract the feature and update the score on the fly. Because all voxels behind the first measurement are backfilled, the SVM score for each class k from (5.1) can be written as:

$$\text{score}[k] = \sum_u \sum_v \beta'_k[u, v, z'[u, v]], \tag{3.8}$$

where $z'[u, v]$ is the quantized depth map and tensor $\beta'_k[u, v, w]$ is the cumulative sum of the weight tensor along dimension w:

$$\beta'_k[u, v, w] = \sum_{d>=w} \beta_k[u, v, d]. \tag{3.9}$$

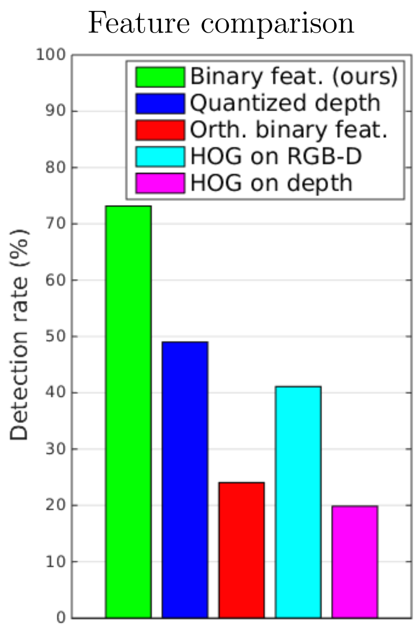
Note that the above cumulative-sum tensors can be precomputed. This makes test-time classification quite efficient (Eq. 3.8). Feature extraction and SVM classification can be computed jointly following the algorithm presented in Alg. 2. Our implementation runs at 275 frames per second.

3.4 Experiments

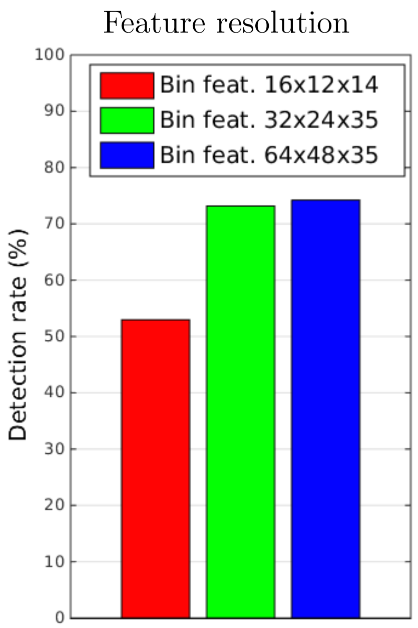
For evaluation, we use the recently released UCI Egocentric dataset [181] and score hand pose detection as a proxy for limb pose recognition (following the benchmark criteria used in [181]) . The dataset consists of 4 video sequences (around 1000 frames each) of everyday egocentric scenes with hand annotations every 10 frames.

```
input  : Quantized depth map $z'[u,v]$.
         Cumsum'ed weights $\{\beta'_k[u,v,w]\}$.
output: score$[k]$
for $u \in \{0,1,...N_u\}$ do
    for $v \in \{0,1,...N_v\}$ do
        for $k \in \{0,1,...K\}$ do
            score$[k] += \beta'_k[u,v,z'[u,v]]$
        end
    end
end
```

Algorithm 2: Joint feature extraction & classification. We jointly extract binarized depth features and evaluate linear classifiers for all quantized poses k. We precompute a "cumsum" β'_k of our SVM weights. At each location (u,v), we add all the weights corresponding to the voxels behind $z[u,v]$, i.e. such that $w \geq z[u,v]$.



(a)



(b)

Figure 3.8: **Feature evaluation**. We compare our feature encoding to different variants (for $K = 750$ classes) in **(a)**. Our feature outperforms HOG-on-depth and HOG-on-RGBD. Our feature also outperforms orthographic voxels and the raw quantized depth map, which surprisingly itself outperforms all other baselines. When combined with a linear classifier, our sparse encoding can learn nonlinear functions of depth (see Fig. 3.7), while the raw depth map can only learn linear functions. We also vary the resolution of our feature in **(b)**, again for $K = 750$. A size of $32 \times 24 \times 35$ is a good trade-off between size and performance. Doubling the resolution in u, v marginally improves accuracy.

Feature evaluation: We first compare hand detection accuracy for different K-way SVM classifiers trained on HOG on depth (as in [181]) and HOG on RGB-D, thus exploiting the stereo-views provided by RGB and depth sensors. To evaluate our voxel encoding, we also trained a SVM directly on the quantized depth map $z[u, v]$ (without constructing a sparse binary feature). To evaluate our perspective voxels, we compare to an orthographic version of our binarized volumetric feature (similar to past work [209, 119]). In that case, we quantize those points that fall within a 64x48x70 cm^3 egocentric workspace in front of the camera into a binary grid of square voxels:

$$b_{\perp}[u, v, w] = \begin{cases} 1 \quad \text{if} \quad \exists i \text{ s.t.} \quad (x_i, y_i, z_i) \in N(u, v, w) \\ 0 \qquad \text{otherwise} \end{cases} \tag{3.10}$$

where $N(u, v, w)$ specifies a $2 \times 2 \times 2cm$ cube centered at voxel (u, v, w). Note that this feature is considerably more involved to calculate, since it requires an explicit backprojection and explicit geometric computations for binning. Moreover, identifying occluded voxels is difficult because they are not arranged along line-of-sight rays.

The results obtained with $K = 750$ pose classes are reported in Fig. 3.8-(a). Our perspective binary features clearly outperforms other types of features. We reach 72% detection accuracy while the state of the art [181] reports 60% accuracy. Our volumetric feature has empirically strong performance in egocentric settings. One reason is that it is robust to small intra-cluster misalignment and deformations because all voxels behind the first measurement are backfilled. Second, it is sensitive to variations in apparent size induced by perspective effects (because voxels have consistent perspective projections). In Fig. 3.8-(b), we also show results varying the resolution of the grid. Our choice of $32 \times 24 \times 35$ is a good trade-off between feature dimensionality and performance.

We compare primarily to [181], as that method was already shown to outperform commercial (Intel PXC [87]) and fully-featured tracking systems [160]. Such systems perform poorly due

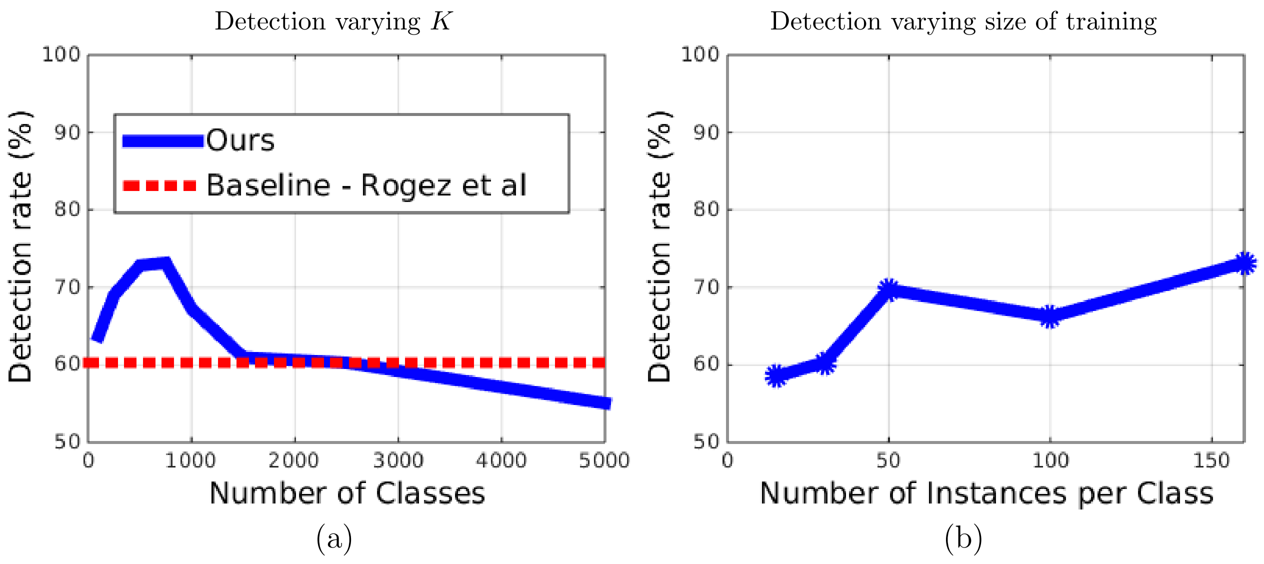


Figure 3.9: **Clustering and size of training set.** In **(a)**, we plot performance as a function of K (the number of discretized pose classes) for a fixed-size training set. For reference, we also plot the state-of-the-art method from [181]. In **(b)**, we plot performance as we increase the amount of training data for K. Both results suggest that our system may perform better with more training data and more quantized poses. Please see text for further discussion.

to occlusions inherent in egocentric viewpoints. Notably, [181] use local part templates in a scanning window fashion. Our global approach captures correlations between pose and spatial location, and better deals with occlusion where local appearance can be misleading.

Training data and clustering: We evaluated the performance of our algorithm when varying the number of quantized pose classes K and the amount of training data. Fig. 3.9-(a) varies K for a fixed training set of 120,000 training images. Performance maxes out relatively quickly at $K = 750$, suggesting that our model may be overfitting due to lack of training data. Fig. 3.9-(a) fixes $K = 750$ and increases the amount of training data per quantized class. Here, we see a more consistent increase in accuracy. These results suggest that a massive training set and larger K may produce better results.

Qualitative results: We illustrate successes in difficult scenarios in Fig. 3.10 and analyze common failure modes in Fig. 3.11. Please see the figures for additional discussion.

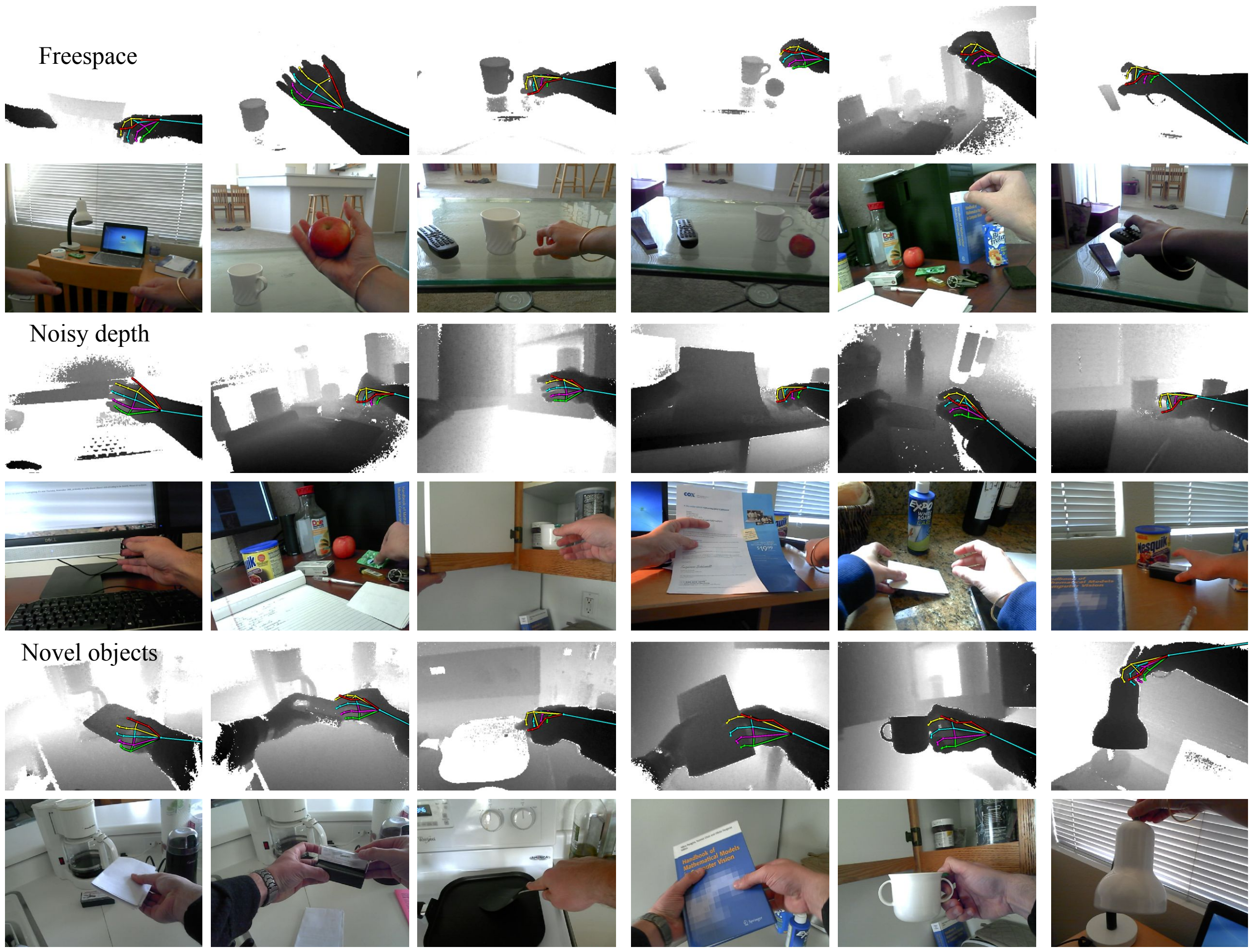


Figure 3.10: **Good detections**. We show frames where arm and hand are correctly detected. First, we present some easy cases of hands in free-space (**top row**). Noisy depth data and cluttered background cases (**middle row**) showcases the robustness of our system while novel objects (**bottom row:** envelope, staple box, pan, double-handed cup and lamp) require generalization to unseen objects at train-time.

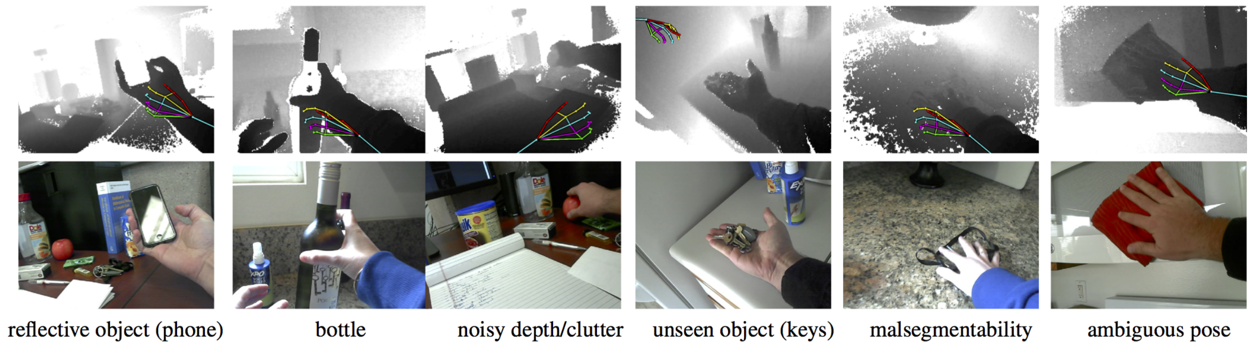


Figure 3.11: **Hard cases**. We show cases where the pose is not correctly recognized (sometimes not even detected): excessively-noisy depth data, hands manipulating reflective material (phone or bottle of wine) or malsegmentability cases of hands touching background.

3.5 Conclusions

We have proposed a new approach to the problem of egocentric 3D hand pose recognition during interactions with objects. Instead of classifying local depth image regions through a typical translation-invariant scanning window, we have shown that classifying the global arm+hand+object configurations within the "whole" egocentric workspace in front of the camera allows for fast and accurate results. We train our model by synthesizing workspace exemplars consisting of hands, arms, objects and backgrounds. Our model explicitly reasons about perspective occlusions while being both conceptually and practically simple to implement (4 lines of code). We produce state-of-the-art real-time results for egocentric pose estimation in real-time.

Chapter 4

Models for Hand Grasps

Grégory Rogez, James Steven Supančič III & Deva Ramanan

4.1 Introduction

Recall that, in the preceding chapter, we developed a method to (re-)initialize a hand tracker by estimating pose from a single egocentric depth depth image. But, there is more to hand-object manipulation than pose. Humans can interact with objects in complex ways, including grasping, pushing, or bending them. By watching humans, an autonomous robot might learn the forces to apply in novel object manipulation tasks. In this chapter, we address the perceptual problem of parsing such manipulations, with a focus on handheld, manipulatable objects. Most previous work on hand analysis tends to focus only on kinematic pose estimation [79, 56]. Interestingly, the same kinematic pose can be used for dramatically different *functional manipulations* (Fig. 6.1), where differences are manifested in terms of distinct contact points and force vectors. Thus, contact points and forces play a crucial role when parsing such interactions from a functional perspective.

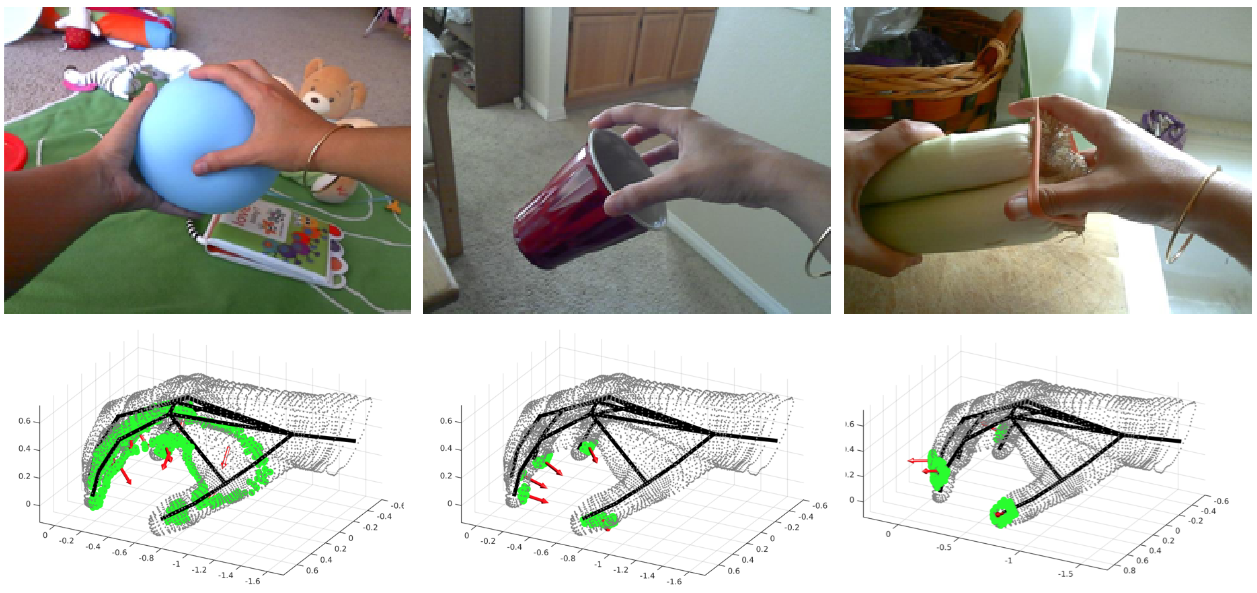


Figure 4.1: **Same kinematic pose, but different functions:** We show 3 images of near-identical kinematic hand pose, but very different functional manipulations, including a wide-object grasp (**a**), a precision grasp (**b**), and a finger extension (**c**). Contact regions (green) and force vectors (red), visualized below each image, appear to define such manipulations. This chapter (1) introduces a large-scale dataset for predicting pose+contacts+forces from images and (2) proposes an initial method based on fine-grained grasp classification.

Problem setup: Importantly, we wish to analyze human-object interactions *in situ*. To do so, we make use of wearable depth cameras to ensure that recordings are mobile (allowing one to capture diverse scenes [171, 31]) and passive (avoiding the need for specialized pressure sensors/gloves [27, 112]). We make no explicit assumption about the environment, such as known geometry [169]. However, we do make explicit use of depth cues, motivated by the fact that humans make use of depth for near-field analysis [66]. Our problem formulation is thus: given a first-person RGB-D image of a hand-object interaction, predict the 3D kinematic hand pose, contact points, and force vectors.

Motivation: We see several motivating scenarios and applications. Our long-term goal is to produce a truly functional description of a scene that is useful for an autonomous robot. When faced with a novel object, it will be useful to know how if it can be pushed or grasped, and what forces and contacts are necessary to do so [197]. A practical application of our work is imitation learning or learning by demonstration for robotics [8, 70], where a robot can be taught a task by observing humans performing it. Finally, our problem formulation has direct implications for assistive technology. Clinicians watch and evaluate patients performing everyday hand-object interactions for diagnosis and evaluation [7]. A patient-wearable camera that enabled automated parsing of object manipulations would allow for long-term monitoring.

Why is this hard? Estimating forces from visual signals typically requires knowledge of object mass and velocity, which is difficult to reliably infer from a single image or even a video sequence. Isometric forces are even more difficult to estimate because no motion may be observed. Finally, even traditional tasks such as kinematic hand pose estimation are now difficult because manipulated objects tend to generate significant occlusions. Indeed, much previous work on kinematic hand analysis considers isolated hands in free-space [218], which is a considerably easier problem.

Approach: We address the continuous problem of pose+contact+force prediction as a discrete fine-grained classification task, making use of a recent 73-class taxonomy of fine-grained hand-object interactions developed from the robotics community [134]. Our approach is inspired by prototype-based approaches for continuous shape estimation that treat the problem as a discrete categorical prediction tasks, such as shapemes [178] or poselets [26]. However, rather than learning prototypes, we make use of expert domain knowledge to quantize the space of manipulations, which allows us to treat the problem as one of (fine-grained) classification. A vital property of our classification engine is that it is data-driven rather than model-based. We put forth considerable effort toward assembling a large collection of diverse images that span the taxonomy of classes. We experiment with both parametric and exemplar-based classification architectures trained on our collection.

Our contributions: Our primary contributions are (1) a new "in-the-wild", large-scale dataset of fine-grained grasps, annotated with contact points and forces. Importantly, the data is RGB-D and collected from a wearable perspective. (2) We develop a pipeline for fine-grained grasp classification exploiting depth and RGB data, training on combinations of both real and synthetic training data and making use of state-of-the-art deep features. Overall, our results indicate that grasp classification is challenging, with accuracy approaching 20% for a 71-way classification problem. (3) We describe a simple post-processing exemplar framework that predicts contacts and forces associated with hand manipulations, providing an initial proof-of-concept system that addresses this rather novel visual prediction task.

4.2 Related Work

Hand pose with RGB(D): Hand pose estimation is a well-studied task, using both RGB and RGB-D sensors as input. Much work formulates the task as articulated tracking over time [115, 109, 108, 9, 161, 212, 243], but we focus on single-image hand pose estimation during object manipulations. Relatively few papers deal with object manipulations, with the important exceptions of [186, 189, 118, 117]. Most similar to our method is [169], who estimate contact forces during hand-object interactions, but do so in a "in-the-lab" scenario where objects of known geometry are used. We focus on single-frame "in-the-wild" footage where the observer is instrumented, but the environment (and its constituent objects) are not.

Egocentric hand analysis: Spurred by the availability of cheap wearable sensors, there has been a considerable amount of recent work on object manipulation and grasp analysis from egocentric viewpoints [47, 32, 86, 31, 61]. The detection and pose estimation of human hands from wearable cameras was explored in [180]. [32] propose a fully automatic vision-based approach for grasp analysis from a wearable RGB camera, while [86] explores unsupervised clustering techniques for automatically discovering common modes of human hand use. Our work is very much inspired by such lines of thought, but we take a data-driven perspective, focusing on large-scale dataset collection guided by a functional taxonomy.

Grasp taxonomies: Numerous taxonomies of grasps have been proposed, predominantly from the robotics community. Early work by Cutkosky [43] introduced 16 grasps, which were later extended to 33 by Feix et al. [63], following a definition of a grasp as a "static hand postures with which an object can be held with one hand". Though this excluded two-handed, dynamic, and gravity-dependent grasps, this taxonomy has been widely used [188, 32, 31]. Our work is based on a recent fine-grained taxonomy proposed in [134], that significantly broadens the scope of manipulations to include *non-prehensile* object interactions (that are technically not grasps, such as pushing or pressing) as well as other gravity-dependent interactions (such as lifting). The final taxonomy includes 73 grasps that are annotated with various qualities (including hand shape, force type, direction of movement and effort).

Datasets. Because grasp understanding is usually addressed from a robotics perspective, the resulting methods and datasets developed for the problem tend to be tailored for that domain. For example, robotics platforms often require an unavoidable real-time constraint, limiting the choice of algorithms, which also (perhaps implicitly) limited the difficulty of the data in terms of diversity (few subjects, few objects, few scenes). We overview the existing grasp datasets in Table 4.1 and tailor our new dataset to "fill the gap" in terms of overall scale, diversity, and annotation detail.

4.3 GUN-71: Grasp UNderstanding Dataset

We begin by describing our dataset, visualized in Fig. 4.2. We start with the 73-class taxonomy of [134], but omit grasps 50 and 61 because of their overly-specialized nature (holding a ping-pong racket and playing saxophone, respectively), resulting in 71 classes.

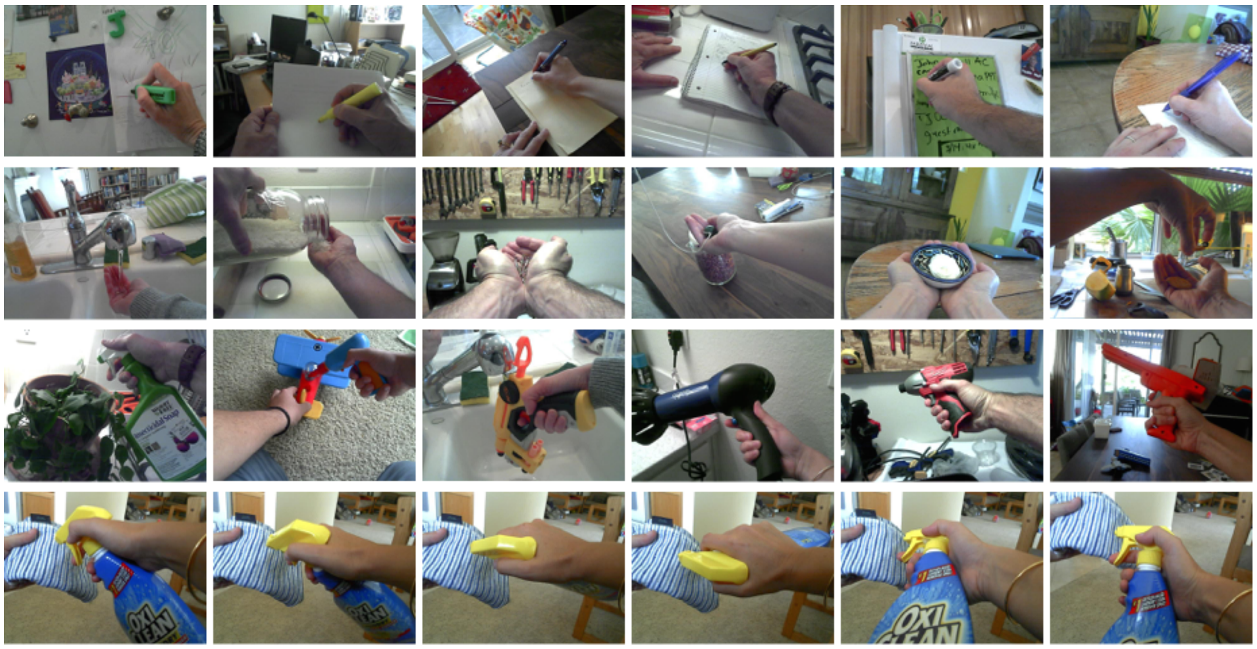


Figure 4.2: **GUN-71: Grasp Understanding dataset:** We have captured (from a chest-mounted RGB-D camera) and annotated our own dataset of fine-grained grasps, following the recent taxonomy of [134]. In the **top** row, the “writing tripod” grasp class exhibits low variability in object and pose/view across 6 different subjects and environments. In the **second** row, “flat hand cupping” exhibits high variability in objects and low variability in pose due to being gravity-dependent. In the **third** row, “trigger press” exhibits high variability in objects and pose/view. Finally, in **bottom**, we show 6 views of the same grasp captured for a particular object and a particular subject in our dataset.

4.3.1 Data capture

To capture truly in-the-wild data, we might follow the approach of [31] and monitor unprompted subjects behaving naturally throughout the course of a day. However, this results in a highly imbalanced distribution of observed object manipulations. Bullock et al. [31] shows that 10 grasps suffice to explain 80% of the object interactions of everyday users. Balanced class distributions arguably allow for more straightforward analysis, which is useful when addressing a relatively unexplored problem. Collecting a balanced distribution in such a unprompted manner would be prohibitively expensive, both in terms of raw data collection and manual annotation. Instead, we prompt users using the scheme below.

Capture sessions: We ask subjects to perform the 71 grasps on personal objects (typical for the specific grasp), mimicking real object manipulation scenarios in their home environment. Capture sessions were fairly intensive and laborious as a result. We mount Intel's Senz3D, a wearable time-of-flight sensor [89, 46, 138], on the subjects's chest using a GoPro harness (as in [180]). We tried to vary the types of objects as much as possible and considered between 3 and 4 different objects per subject for each of the 71 grasps. For each hand-object configuration, we took between 5 and 6 views of the manipulation scene. These views correspond to several steps of a particular action (opening a lid, pouring water) as well as different 3D locations and orientation of the hand holding the object (with respect to the camera).

Diversity: This process led to the capture of roughly 12,000 RGB-D images labeled with one of the 71 grasps. We captured 28 objects per grasp, resulting in $28 \times 71 = 1988$ different hand-object configurations with 5-6 views for each. We consider 8 different subjects (4 males and 4 females) in 5 different houses, ensuring that "house mates" avoid using the same objects to allow leave-one-out experiments (we can leave out one subject for testing and ensure that the objects will be novel as well). Six of our eight subjects were right handed. To ensure consistency, we asked the two left-handed subjects to perform grasps with their right hand. We posited that body shape characteristics might effect accuracy and generalizability, particularly in terms of hand size, shape, and movement. To facilitate such analysis, we also measured arm and finger lengths for each subject.

4.4 Synthetic (training) data generation

3D hand-object models: In addition to GUN-71, we construct a synthetic training dataset that will be used during our grasp-recognition pipeline. To construct this synthetic dataset, we make use of synthetic 3D hand models. We obtain a set of 3D models by extending the

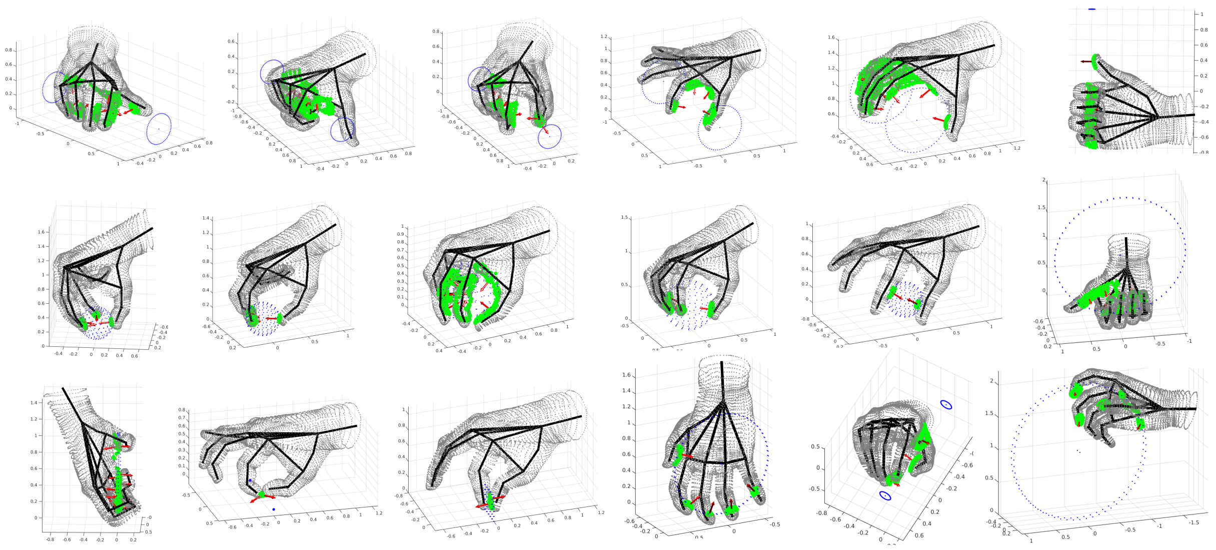


Figure 4.3: **Contact point and forces:** We show the 3D hand model for 18 grasps of the considered taxonomy. We also show the contact points (in green) and forces (in red) corresponding to each grasp. The blue points help visualize the shape of the typical object associated with each of these 18 grasps. We can observe that power grasps have wider contact areas on finger and palm regions, while precision grasps exhibit more localized contact points on finger tips and finger pads.

publicly-available Poser models from [186] to cover the selected grasps from [134]'s taxonomy (by manually articulating the models to match the visual description of grasp).

Contact and force annotations: We compute contact points and applied forces on our 3D models with the following heuristic procedure. First, we look for physical points of contact between the hand and object mesh. We do this by intersecting the triangulated hand and object meshes with the efficient method of [150]. We produce a final set of contact regions by connected-component clustering the set of 3D vertices lying within an intersection boundary. To estimate a force vector, we assume that contact points are locally stable and will not slide along the surface of the object (implying the force vector is normal to the surface of the object). We estimate this normal direction by simply reporting the average normal of vertices within each contact region. Note this only produces an estimate of the force direction, and not magnitude. Nevertheless, we find our heuristic procedure to produce surprisingly plausible estimates of contact points and force directions for each 3D model (Fig. 4.3).

Synthetic training data: We use our 3D models to generate an auxiliary dataset of synthetic depth data, annotated with 3D poses, grasp class label, contacts, and force direction vectors. We additionally annotate each rendered depth map with a segmentation mask denoting background, hand, and object pixels. We render over $200,000$ training instances ($3,000$ per grasp). We will release our models, rendering images, as well as GUN-71 (our dataset of real-world RGB-D images) to spun further research in the area.

4.5 Recognition pipeline

We now describe a fairly straightforward recognition system for recognizing grasps given real-world RGB-D images. Our pipeline consists of two stages; hand segmentation and fine-grained grasp classification.

4.5.1 Segmentation

The first stage of our pipeline is segmenting the hand from background clutter, both in the RGB and depth data. Many state-of-the-art approaches [32, 189, 186] employ user-specific skin models to localize and segment out the hand. We want a system that does not require such a user-specific learning stage and could be applied to any new user and environment, and so instead make use of depth cues to segment out the hand.

Depth-based hand detection: We train a P-way classifier designed to report one of $P = 1500$ quantized hand poses, using the approach of [182]. This classifier is trained on the synthetic training data, which is off-line clustered into P pose classes. Note that the set of

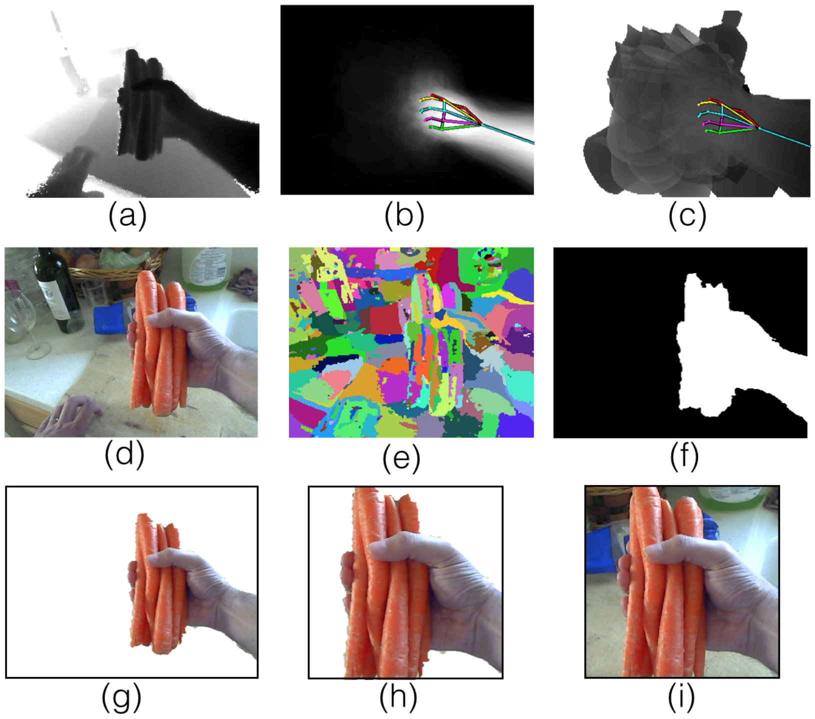


Figure 4.4: **Segmentation:** We show the different steps of our segmentation stage: the depth map (**a**) is processed using a K-way pose classifier [182], which reports a quantized pose detection k and associated foreground prior b_{ik} (**b**) and mean depth μ_{ik}(**c**) (used to compute a posterior following Eq. 4.1). To incorporate bottom-up RGB cues, we first extract superpixels (**e**) and then label superpixels instead of pixels to produce a segmentation mask (**f**). This produces a segmented RGB image in (**g**) , which can then be cropped (**h**) and/or unsegmented (**i**). We concatenate (deep) features extracted from (d), (g), (h), and (i) to span a variety of resolutions and local/global contexts.

pose classes P is significantly larger than the set of fine-grained grasps $K = 71$. We use the segmentation mask associated with this coarse quantized pose detection to segment out the hand (and object) from the test image, described further below.

Pixel model: We would like to use hand detections to generate binary segmentation masks. To do so, we use a simple probabilistic model where x_i denotes the depth value of pixel i and $y_i \in \{0, 1\}$ is its binary foreground/background label. We write the posterior probability of label y_i given observation x_i, all conditioned on pose class k as:

$$p(y_i|x_i, k) \propto p(y_i|k)p(x_i|y_i, k) \tag{4.1}$$

which can easily derived from Bayes rule . The first term on the right-hand-side is the "prior" probability of pixel i being fg/bg, and the second term is a "likelihood" of observing a depth value given a pose class k and label:

$$p(y_i = 1|k) = b_{ik} \qquad Bernoulli \tag{4.2}$$

$$p(x_i|y_i = 1, k) = N(x_i; \mu_{ik}, \sigma_{ik}^2) \qquad Normal \tag{4.3}$$

$$p(x_i|y_i = 0, k) \propto \text{constant} \qquad Uniform \tag{4.4}$$

We use a pixel-specific Bernoulli distribution for the prior, and an univariate Normal and Uniform (uninformative) distribution for the likelihood. Intuitively, foreground depths tend to be constrained by the pose, while the background will not be. Given training data of depth images x with foreground masks y and pose class labels k, it is straightforward to estimate model parameters $\{b_{ik}, \mu_{ik}, \sigma_{ik}\}$ with maximum likelihood estimation (frequency counts, sample means, and sample variances). We visualize the pixel-wise Bernoulli prior b_{ik} and mean depth μ_{ik} for a particular class k in Fig. 4.4-b and Fig. 4.4-c.

RGB-cues: Thus far, our segmentation model does not make use of RGB-based grouping cues such as color changes across object boundaries. To do so, we first compute RGB-based superpixels [2] on a test image and reason about the binary labels of superpixels rather than pixels:

$$label_j = I\Big(\frac{1}{|S_j|}\sum_{i \in S_j} p(y_i|x_i, k) > .5\Big) \tag{4.5}$$

where S_j denotes the set of pixels from superpixel j. We show a sample segmentation in Fig. 4.4. Our probabilistic approach tends to produce more reliable segmentations than existing approaches based on connected-component heuristics [87].

4.5.2 Fine-grained classification

We use the previous segmentation stage to produce features that will be fed into a $K = 71$-way classifier. We use state-of-the-art deep networks – specifically, Deep19 [204] – to extract a 3096 dimensional feature. We extract off-the-shelf deep features extracted for (1) the entire RGB image, (2) a cropped window around the detected hand, and (3) a segmented RGB image (Fig. 4.4 (d,g,h,i)). We resize each window to a canonical size (of 224 x 224 pixels) before processing. The intuition behind this choice is to mix high and low resolution features, as well as global (contextual) and local features. The final concatenated descriptors are fed into a linear multi-class SVM for processing.

Exemplar matching: The above stages return an estimate for the employed grasp and a fairly accurate quantized pose class, but it is still quantized nonetheless. One can refine this quantization by returning the closest synthetic training example belonging to the recognized grasp and the corresponding pose cluster. We do this by returning the training example n from quantized class k with the closest foreground depth:

$$NN(x) = \min_{n \in Class_k} \sum_i y_i^n (x_i^n - \mu_{ik})^2 \tag{4.6}$$

We match only foreground depths in the n^{th} synthetic training image x^n, as specified by its binary label y^n. Because each synthetic exemplar is annotated with hand-object contact points and forces from its parent 3D hand model, we can predict forces and contact points by simply transferring them from the selected grasp model to the exemplar location in the 3D space.

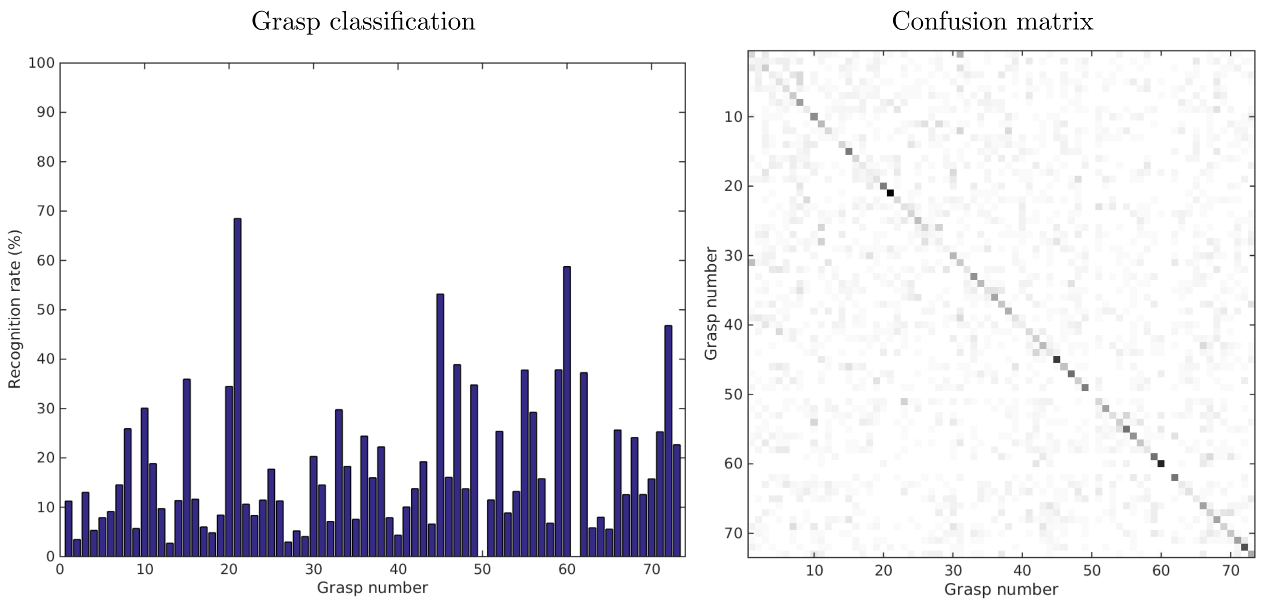
4.6 Experiments

For all the experiments of this section, we use a leave-one-out approach where we train our 1-vs-all SVM classifiers on 7 subjects and test on the last 8^{th} subject. We repeat that operation with the 8 subjects and average the results. When analyzing our results, we refer to grasps by their id#. In the supplementary material, we include a visualization of all grasps in our taxonomy.

Baselines: We first run some "standard" baselines: HOG-RGB, HOG-Depth, and an off-the-shelf deep RGB feature [204]. We obtained the following average classification rate: HOG-RGB (3.30%), HOG-Depth (6.55%), concatenated HOG-RGB and HOG-Depth (6.55%) and Deep-RGB (11.31%). Consistent with recent evidence, deep features considerably outperform their hand-designed counterparts, though overall performance is still rather low (Table 4.2).

Segmented/cropped data: Next, we evaluate the role of context and clutter. Using segmented RGB images marginally decreases accuracy of deep features from 11.31% to 11.10%, but recognition rates appear are more homogeneous. Looking at the individual grasp classification rates, segmentation helps a little for most grasps but hurts the accuracy of "easy" grasps where context or object shape are important (but removed in the segmentation). This includes non-prehensile "pressing" grasps (interacting with a keyboard) and grasps associated with unique objects (chopsticks). Deep features extracted from a cropped segmentation and cropped detection increase accuracy to 12.55% and 13.67%, respectively, suggesting that some amount of local context around the hand and object helps.

Competing methods: [189, 32] make use of HOG templates defined on segmented RGB images obtained with skin detection. Because skin detectors did not work well on our (in-the-wild) dataset, we re-implemented [32] using HOG templates defined on our depth-based segmentations and obtained 7.69% accuracy. To evaluate recent non-parametric methods



(a) (b)

Figure 4.5: **RGB Deep feature + SVM:** We show the individual classification rates for the 71 grasps in our dataset (**a**) and the corresponding confusion matrix in (**b**).

[189], we experimented with a naive nearest neighbor (NN) search using the different features extracted for the above experiments and obtained 6.10%, 6.97%, 6.31% grasp recognition accuracy using Deep-RGB, cropped-RGB and cropped+segmented-RGB. For clarity, these replace the K-way SVM classifier with a NN search. The significant drop in performance suggests that the learning is important, implying that our dataset is still not big enough to cover all possible variation in pose, objects and scenes.

Cue-combination: To take advantage of detection and segmentation without hurting classes where context is important, we trained our SVM grasp classifier on the concatenation of all the deep features. Our final overall classification rate of 17.97% is a considerable improvement over a naive deep model 11.31% as well as (our reimplementation of) prior work 7.69%. The corresponding recognition rates per grasp and confusion matrices corresponding to this classifier are given in Fig. 4.5.

Easy cases: High-performing grasp classes (Fig. 4.5) tend to be characterized by limited variability in terms of viewpoint (i.e., position and orientation of the hand w.r.t camera) and/or object: eg. opening a lid (#10), writing (#20), holding chopsticks (#21), measuring with a tape (#33), grabbing a large sphere such as a basketball (#45), using screwdriver (#47), trigger press (#49), using a keyboard (#60), thumb press (#62), holding a wine cup (#72). Other high-performing classes tend to exhibit limited occlusions of the hand: hooking a small object(#15) and palm press (#55).

Common confusions: Common confusions in Fig. 4.6 suggest that finger kinematics are a strong cue captured by deep features. Many confusions correspond to genuinely similar grasps that differ by small details that might be easily occluded by the hand or the manipulated object: "Large diameter" (#1) and "Ring" (#31) are both used to grasp cylindrical objects, except that "Ring" only uses thumb and index finger. When the last three fingers are fully occluded by the object, it is visually impossible to differentiate them (see Fig. 4.6-c). "Adduction-Grip" (#23) and "Middle-over-Index"(#51) both involve grasping an object using the index and middle finger. Abduction-Grip holds the object between the two fingers, while Middle-over-Index holds the object using the pad of the middle finger and nail of the index finger (see Fig. 4.6-f).

Best view: To examine the effect of viewpoint, we select the top-scoring view for each grasp class, increasing accuracy from 17.97% to 22.67% (Table 4.3). Comparing the two sets of recognition rates, best-view generally increases the performance of easy grasps significantly more than difficult ones - e.g., the average recognition rate of the top 20 grasps grow from 36.20% to 47.53%, while the top 10 grasps grows from 44.97% to 59.04%. This suggests that some views may be considerably more ambiguous than others.

Comparison to state-of-the-art. We now compare our results to those systems evaluated on previous grasp datasets. Particularly relevant is [32], which presents visual grasp recognition results in similar settings, i.e., egocentric perspective and daily activities. In their case, they consider a reduced 17-grasp taxonomy from Cutkosky [43], obtaining 31% with HOG features overall and 42% on a specific "machinist sequence" from [31]. Though these

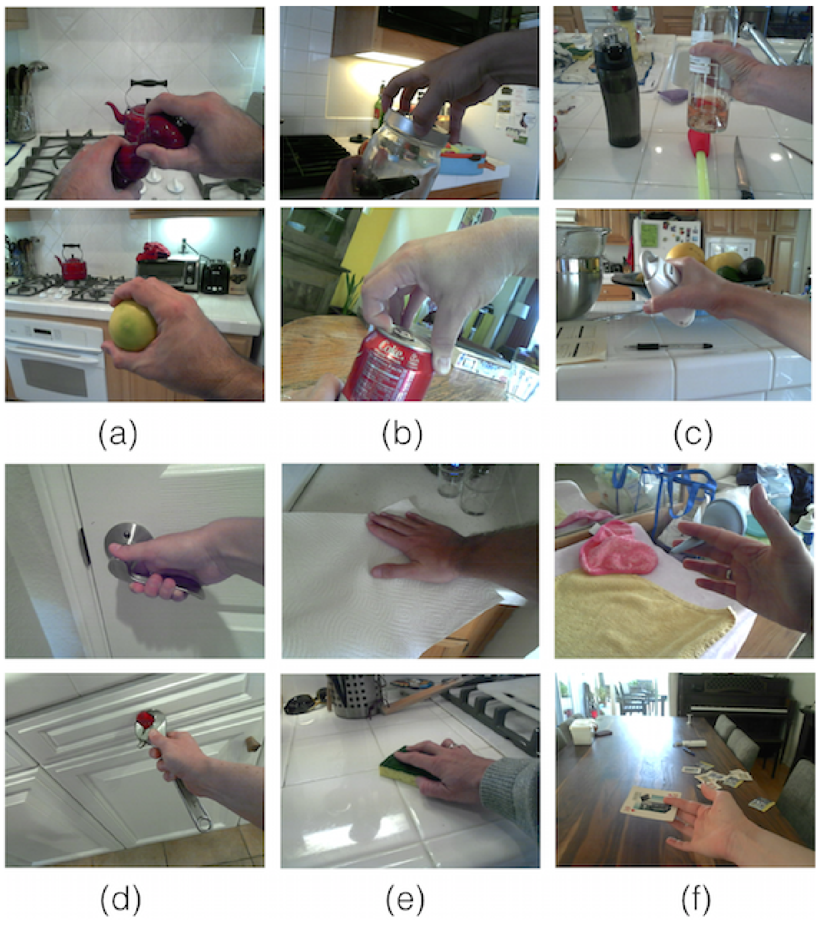


Figure 4.6: **Common confusions:** The confusions occur when some fingers are occluded (**a** and **c**) or when the poses are very similar but the functionality (associated forces and contact points) is different (**b**, **d**, **e** and **f**).

results appear more accurate than ours, its important to note that their dataset contains less variability in the background and scenes, and, crucially, their system appears to require training a skin detector on a subset of the test set. Additionally, it is not clear if they (or indeed, other past work) allow for the same subject and/or scene to be included across the train and testset. If we allow for this, recognition rate dramatically increases to 85%. This highly suggestive of overfitting, and can be seen a compelling motivation for the distinctly large number of subjects and scenes that we capture in our dataset.

Evaluations on limited taxonomies: If we limit our taxonomy to the 17 grasps from Cai et al. [32], i.e., by evaluating only the subset of 17 classes, we obtain 20.53% and 23.44%

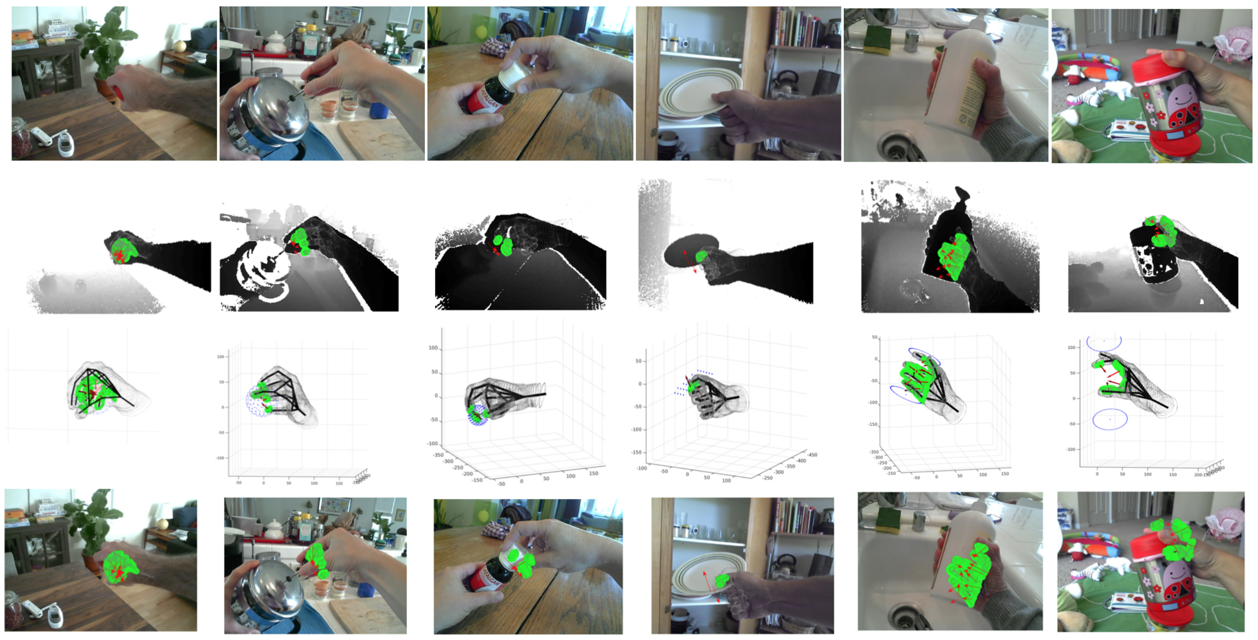


Figure 4.7: **Force and contact points prediction:** We show frames for which the entire pipeline (detection + grasp recognition + exemplar matching) led to an acceptable prediction of forces and contact points. For each selected frame, we show from top to bottom: the RGB image, the depth image with contact points and forces (respectively represented by green points and red arrows, the top scoring 3D exemplar with associated forces and contact points, and finally the RGB image with overlaid forces and contact points.

(best view). See Table 4.4. These numbers are comparable to those reported in Cai et al. [32]. Best-view may be a fair comparison because Bullock et al. [31] used data that was manually labelled, where annotators were explicitly instructed to only annotate those frames that were visually unambiguous. In our case, subjects were asked to naturally perform object manipulations, and the data was collected "as-is". Finally, if we limit our taxonomy to the 33 grasps from Feix et al. [63], we obtained 20.50% and 21.90% (best view). The marginal improvement when evaluating grasps from smaller taxonomies suggests that the new classes are not much harder to recognize. Rather, we believe that overall performance is somewhat low because our dataset is genuinely challenging due to diverse subjects, scenes, and objects.

Force and contact point prediction: Finally, we present preliminary results for force and contact prediction. We do so by showing the best-matching synthetic 3D exemplar from the detected pose class, along with its contact and force annotations. Fig. 4.7 shows frames for which the entire pipeline detection + grasp recognition + exemplar matching led to an acceptable prediction. Unfortunately, we are not able to provide a numerical evaluation as obtaining ground-truth annotation of contact and forces is challenging. One attractive option is to use active force sensors, either embedded into pressure-sensitive gloves worn by the user or through objects equipped with force sensors at predefined grasp points (as done for a simplified cuboid object in [169]). While certainly attractive, active sensing somewhat violates the integrity of a truly in-the-wild, everyday dataset.

4.7 Conclusions

We have introduced the challenging problem of understanding hands in action, including force and contact point prediction, during scenes of in-the-wild, everyday object manipulations. We have proposed an initial solution that reformulates this high-dimensional, continuous prediction task as a discrete fine-grained (functional grasp) classification task. To spur further research, we have captured a new large scale dataset of fine-grained grasps that we will release together with 3D models and rendering engine. Importantly, we have captured this dataset from an egocentric perspective, using RGB-D sensors to record multiple scenes and subjects. We have also proposed a pipeline which exploits depth and RGB data, producing state-of-the-art grasp recognition results. Our first analysis show that depth information is crucial for detection and segmentation, while the richer RGB feature allows for a better grasp recognition. Overall, our results indicate that grasp classification is challenging, with accuracy approaching 20% for a 71-way classification problem.

We have used a single 3D model per grasp. In future work, it would be interesting to (1) model within-grasp variability, capturing the dependence of hand kinematics on object shape and size and (2) consider subject-specific 3D hand shape models [100], which could lead to more accurate set of synthetic exemplars (and associated forces and contacts).

Dataset	View	Cam.	Sub.	Scn	Frms	Label	Tax.
YALE [31]	Ego	RGB	4	4	9100	Gr.	33
UTG [32]	Ego	RGB	4	1	?	Gr.	17
GTEA [61]	Ego	RGB	4	4	00	Act.	7
UCI-EGO [180]	Ego	RGB-D	2	4	400	Pose	?
Ours	Ego	RGB-D	8	> 5	$12,000$	Gr.	71

Table 4.1: **Object manipulation datasets:** [31] captured 27.7 hours but labelled only 9100 frames with grasp annotations. While our dataset is balanced and contains the same amount of data for each grasp, [31] is imbalanced in that common grasps appear much more often than rare grasps (10 grasps suffice to explain 80% of the data). [32] uses the same set of objects for the 4 subjects.

Features	Acc.	top 20	top 10	min	max
HOG-RGB	3.30	7.20	9.59	0.00	28.54
HOG-Depth	6.55	12.96	15.74	0.66	26.18
HOG-RGBD	6.54	13.76	19.24	0.00	45.62
Deep-RGB [204]	11.31	25.92	35.28	0.69	61.39
Deep-RGB(segm.)	11.10	21.56	26.51	0.69	29.46
HOG-RGB (cropped)	5.84	11.22	14.03	0.00	27.85
Deep-RGB (cropped)	13.67	27.32	36.95	1.22	55.35
HOG-RGB (crop.+segm.) [32]	7.69	15.23	18.65	0.69	30.77
HOG-Depth (crop.+segm.)	10.68	22.04	27.99	0.52	42.40
Deep-RGB (crop.+segm.)	12.55	22.89	27.85	0.69	37.49
Deep-RGB (All)	**17.97**	**36.20**	**44.97**	**2.71**	**68.48**

Table 4.2: **Grasp classification results.** We present the result obtained when training a K-way linear SVM (K=71) with different types of features: HOG-RGB, HOG-Depth and Deep-RGB features, on the whole workspace, i.e., entire image, on a cropped detection window or on cropped and segmented image.

View	Acc.	top 20	top 10	min	max
All (All)	17.97	36.20	44.97	**2.71**	68.48
Best scoring view	**22.67**	**47.53**	**59.04**	0	**79.37**

Table 4.3: **View selection:** We present grasp recognition results obtained when training a K-way linear SVM on a concatenation of Deep features. We present the results obtained when computing the average classification rate over 1) the entire dataset and 2) over the top scoring view of each hand-object configuration.

	71 Gr. [134]	33 Gr. [63]	17 Gr. [43]
All views	17.97	20.50	20.53
Best scoring view	22.67	21.90	23.44

Table 4.4: **Grasp classification for different sized taxonomies:** We present the results obtained for $K = 71$ [134], $K = 33$ [63] and $K = 17$ [43], smaller taxonomies being obtained by selecting the corresponding subsets of grasps.

Chapter 5

Self-Paced Learning for Tracking

James Steven Supančič III & Deva Ramanan

5.1 Introduction

In the previous three chapters, we discussed pre-constructing models offline, specifically for hand tracking. But, hands are not the only things we want to track. Sometimes we must track an object which we have not seen before, so we cannot use a pre-constructed appearance model. This is the problem of model-free tracking: we specifically consider the scenario where one must track an unknown object, given a known bounding box in a single frame. We focus on long-term tracking, where the object may become occluded, significantly change scale, and leave/re-enter the field-of-view.

Our approach builds on two key observations made by past work. The first is the importance of learning an appearance model. We learn adaptive discriminative models that implicitly encode the difference in appearance between the object and the background. Such methods allow for the construction of highly-tuned templates that are resistant to background clutter.

However, a well-known danger of adaptively *updating* (or re-learning) a template over time is the tendency to drift [142]; drift occurs when small errors in tracking accumulate in the appearance model until the tracker fails. Our main contribution is an algorithm that minimizes drift by carefully *deciding* which frames to learn from, using the framework of self-paced learning [19, 114].

The second observation is the importance of detection, where an object template is globally scanned across the entire frame. This allows one to *re-initialize* lost tracks, but requires detectors resistant to background clutter at all spatial regions, including those far away from the object. Most prior approaches learn a detector using a small set of negatives. We show that using a large set of negatives significantly improves performance, but also increases computation. We address the latter issue through the use of solvers that can be warm-started from previous solutions [58].

Self-paced learning: Curriculum learning is an approach inspired by the teaching of students, where easy concepts (say, a model learned from un-occluded frames) are taught before complex ones (a model learned from frames with partial occlusions) [19]. In self-paced learning, the student learner must determine what is easy versus complex [114]. One natural application of such a strategy would label frames as easy or complex as they are encountered by an online tracker. One could then update appearance models after the easy frames. We show that it is crucial to revisit old frames when learning. In terms of self-paced learning, a student might initially think a concept is hard; however, once that student learns other concepts, it may become easy *in retrospect.*

Transductive learning: A unique aspect of our learning problem is that it is transductive rather than inductive [232]: when tracking a face, the learned model need not generalize to all faces, but only separate the particular face and background in the video. In some sense, we want to "over-fit" to the video. Following [232], we use a transductive strategy for selecting frames: rather than choosing a frame that scores well under the current model (as

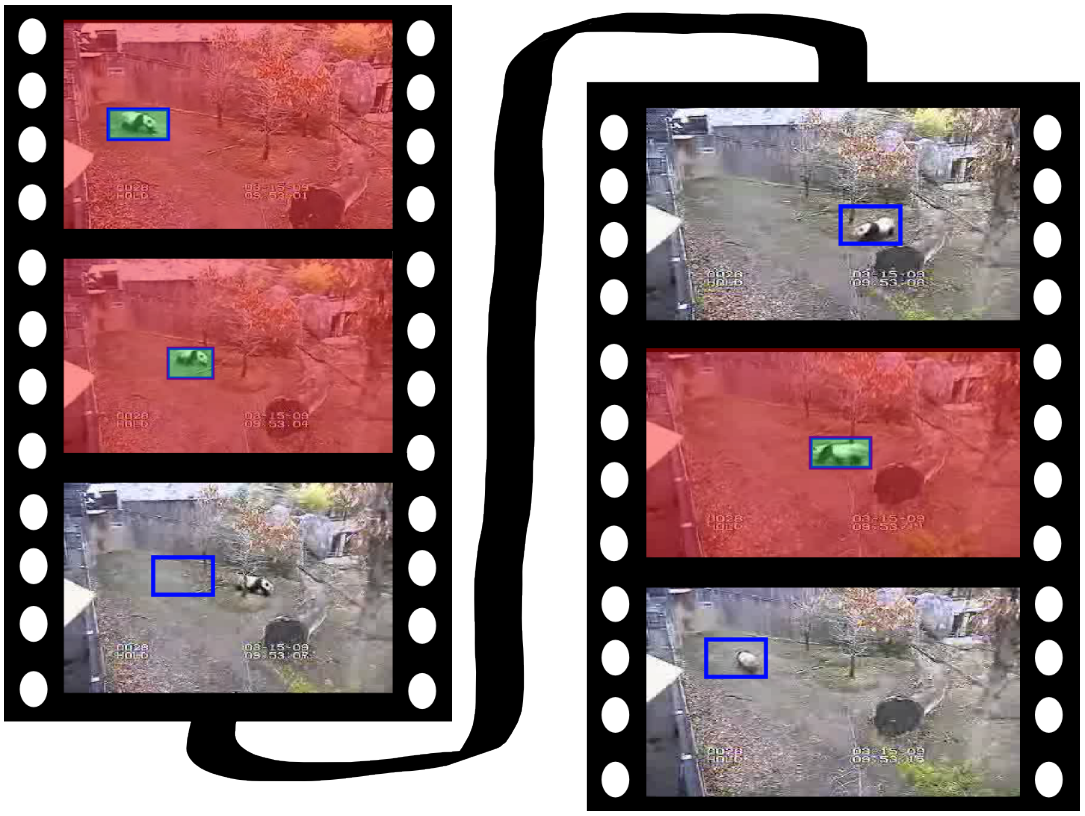


Figure 5.1: Our tracker uses self-paced learning to select reliable frames from which to extract additional training data as it progresses (shown in **red**). We use such frames to define both positive examples and a very-large set of negative examples (all windows that do not overlap each positive). By re-learning a model with this additional data, and re-tracking with that model, one can correct the errors shown above. We show that it is crucial to revisit old frames when adding training data; in terms of self-paced learning, a concept (frame) that initially looks hard may become easy in hindsight.

most prior work does), we choose a frame that when selected for learning, produces a model that well-separates the object from the background. We demonstrate that the latter scheme performs better because it is naturally retrospective.

Evaluation: We evaluate our method on a large-scale benchmark suite of videos. Part of our contribution is a baseline detector that tracks by detection without any online learning or temporal reasoning; the detector is learned from the first labeled frame. Surprisingly, we demonstrate that a simple linear template defined on HOG features *outperforms* state-of-the-art trackers, including the well-known Predator TLD-Tracker [97] and MIL-Tracker [12]. We believe this disparity exists because detection is an undervalued aspect of tracking; invariant gradient descriptors and large-scale negative training sets appear crucial for building good object detectors [45], but are insufficiently used in tracking. We use this baseline as a starting point, and show that one can reduce error by a **factor of 4** with our proposed self-paced transduction framework.

Computation: Our detection-based approach learns appearance models from large training sets with hundreds of thousands of negative examples. Our self-paced learning scheme requires learning putative appearance models for each candidate image. To address these computational burdens, we make use of dual coordinate descent SVM solvers that can be "warm-started" from previous solutions. Our solvers are efficient enough that detection (implemented as a convolution) is the computational bottleneck, which can further be ameliorated with parallel computations. This means that our tracker is near real-time at present, and could readily be real-time with hardware implementations.

5.1.1 Related work

We refer the reader to the survey from [256] for a broad description of related work. Many object trackers can be differentiated between their choice of appearance models and inference algorithms.

Appearance models: Because object appearance is likely to change over time, many tracks update appearance models through color histogram tracking [40] and online adaption [91]. Generative subspace models of appearance are common [190, 122], including recent work that makes use of sparse reconstructions [143, 133]. Other methods have focused on discrimative appearance models [39], often trained with boosting [72, 11] or SVMs [10, 78]. Our work is similar to these latter approaches, though we focus on the problem of carefully choosing a subset of frames from which to learn a classifier.

Inference algorithms: To capture multiple hypotheses, many trackers use sampling-based schemes such as particle filtering [88, 162] or Markov-chain Monte-Carlo techniques [116, 167]. Such methods may require a large number of particles to track objects in clutter. We show that a discrete first-order dynamic model (which is straightforward to optimize with dynamic programming) can accurately reason about multiple hypotheses. Moreover, our experiments suggest that multiple hypotheses may not even be necessary given a good appearance model; in such cases, tracking by detection is a simple and effective inference strategy.

Semi-supervised tracking: Semi-supervised and transductive approaches have been previously used in tracking. Semi-supervised [97, 12, 13] trackers tend to proceed in a greedy online fashion, not revisiting past decisions. We show retrospective learning is important for correcting errors in the past. Transductive approaches [246, 260] are limited by the fact that the general transductive problem is highly non-convex and hard to solve. We show that transduction can be effectively applied for the isolated sub-problem of frame selection (for self-paced learning).

5.2 Approach

Our tracker operates by iterating over three stages. First, it *learns* a detector given a training set of positives and negatives. Second it *tracks* using that learned detector. Third, it *selects* a subset of frames from which to re-learn a detector for the next iteration. After describing each stage, we describe our final algorithms in Sec. 5.3.

5.2.1 Learning appearance with a SVM

Assume we are given a set of labeled frames, where we are told the location of the object. Initially, this is the first frame of a video. We would like to learn a detector to apply on the

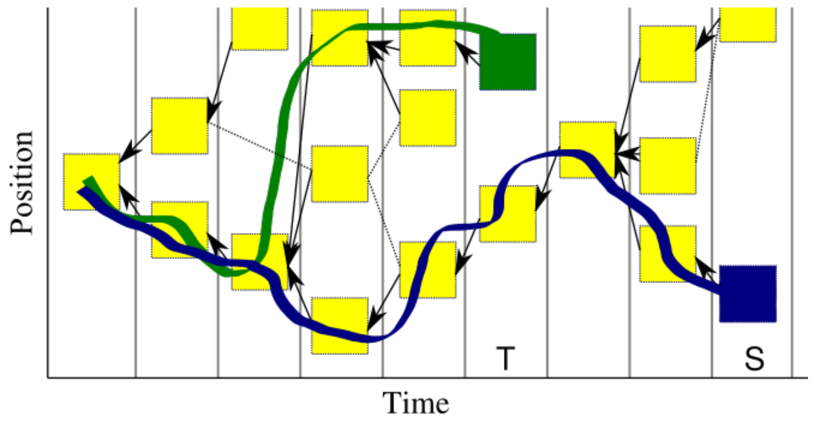


Figure 5.2: We use dynamic programming to maintain multiple track hypothesis over time. We visualize detections as boxes, the best previous transition leading to a give detection as a solid arrow, and non-optimal legal transitions with a dotted line. Note that the most-likely track at frame T (in green) can be revised to an alternate track hypothesis at a later frame S (in blue). We find that such reasoning provides a modest increase in performance.

remaining unlabeled frames. We write Λ for a set of frame-bounding box $\{(t_i, b_i)\}$ pairs. We extract positive and negative examples from Λ, and use them to learn a linear SVM:

$$LEARN(\Lambda) = \operatorname*{argmin}_{w} \frac{\lambda}{2} w \cdot w + \tag{5.1}$$
$$\sum_{t_i \in \Lambda} \Big[\max(0,1 - w \cdot \phi_{t_i}(b_i)) + \sum_{b \neq b_i} \max(0, 1 + w \cdot \phi_{t_i}(b)) \Big]$$

where $\phi_{t_i}(b_i)$ extracts appearance (e.g. , HOG) features from bounding box b_i in frame t_i. For each frame t_i in Λ, we extract a single positive example at bounding box b_i and extract a large set of negative examples at all other non-overlapping bounding boxes. We use a fixed regularization parameter λ for all our experiments. We solve the above minimization using a quadratic programming (QP) solver [58]. For convenience, we define $OBJ(\Lambda)$ to be the min objective value corresponding to the argmin from (5.1).

5.2.2 Tracking as shortest-paths

We formulate the problem of finding the optimal track $y_{1:N} = \{y_1, \dots y_N\}$ given the known location in the first fame y_1 and model w by solving a shortest path problem on a trellis graph shown in Fig. 5.2. For each frame, we have a set of nodes, or states, representing possible positions of the object in that frame. Between each pair of frames, we have a set of edges representing the cost of transitioning from a particular location to another location:

$$TRACK(y_1, w, N) = \operatorname*{argmin}_{y_{1:N}} \sum_{t=2}^{N} \pi(y_t, y_{t-1}) - w \cdot \phi(t, y_t) \tag{5.2}$$

Local cost: The second term defines the local cost of placing the object at location y_t in frame x_t as the negative SVM score of w. We experimented with calibrating the score to return a probability, but did not see a significant change in performance.

Pairwise cost: We experimented with many different definitions for the pairwise cost π. In general, we found a thresholded motion model to work well in most scenarios (where the pairwise cost is 0 for transitions that are consistent with measured optical flow and ∞ otherwise). Finally, to model occlusions, we augment y_t with a dummy occlusion state with a fixed local cost.

We compute the best track by solving the shortest-path problem using dynamic programming [29]. We also experimented with an uninformative prior $\pi(y_t, y_{t-1})$; in this case, the best track is given by independently selecting the highest scoring location in each frame ("tracking by detection").

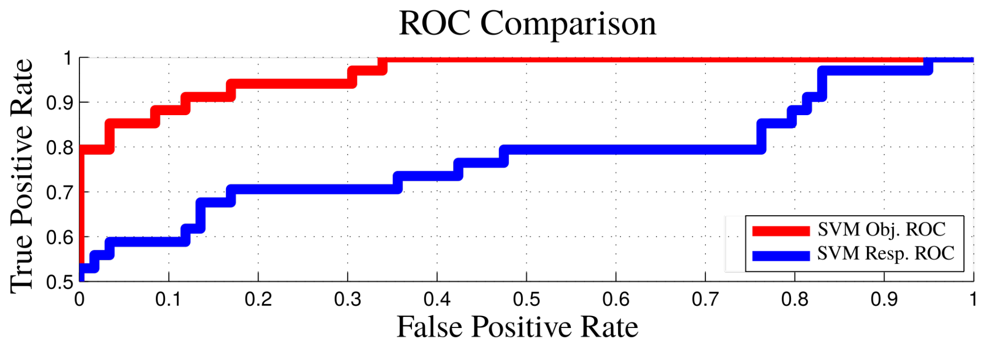


Figure 5.3: Given an initial detector w, we consider different methods for selecting frames in our SELECT stage. We deem a selected frame as a true positive if the estimated track location correctly overlaps the ground-truth. Frames selected based on the SVM objective value in (5.3) greatly outperform frames selected on SVM response.

5.2.3 Selecting good frames

Our tracker operates by sequentially re-learning a model from previously-tracked frames. To avoid template drift, we find it crucial to select "good" frames from which to learn. Given a set of frames used for learning Λ and an estimated track $y_{1:N}$, we estimate a new set of good frames:

$$\begin{aligned} SELECT(\Lambda, y_{1:N}) &= \Lambda \cup (t, y_t) \quad \text{where} \\ t &= \operatorname*{argmin}_{t' \notin \Lambda} OBJ(\Lambda \cup (t', y_{t'})) \end{aligned} \tag{5.3}$$

where we define $t \notin \Lambda$ to refer to frames that are not in any frame-location pair in Λ. The above select function computes the frame, that when added to the training set Λ, produces the lowest SVM objective. We generalize the above function to return a K-element set by independently finding the $(K - |\Lambda|)$ frames with the smallest increase in the SVM objective, written as $SELECT_K(\Lambda, y_{1:N})$.

Why use OBJ? Our approach directly follows from strategies for data selection in self-placed learning [114] and label assignment in transductive learning [232]. A more standard approach may be to simply select the frame with the strongest model response $w \cdot \phi(t, y_t)$;

Fig. 5.3 shows that this a poorer predictor of correct frames that should be used for learning. To build intuition as to why, consider tracking a face that rotates from frontal to profile. A model learned on frontal poses may score poorly on a profile face. However, a model (retrospectively) learned from frontal and profile faces may still produce a good discriminant boundary (and hence a low SVM objective).

5.3 Algorithm

We now describe an online and an offline algorithm that make use of the previously-defined stages.

5.3.1 Online (causal) tracker

Our online algorithm is outlined in Algorithm 3. Intuitively, at each frame: we re-estimate the best track from the first to the current frame using the current model. We then select half of the observed frames to learn/update an appearance model, which is used for the next frame. A crucial aspect of our algorithm is that can select frames that were previously rejected (unlike, for example [97]).

Efficiency: A naive implementation of an online algorithm would be very slow because it involves solving a shortest-path problem and learning an SVM at every timestep. Moreover, the *SELECT* function requires learning (t) SVMs at each iteration (in order to evaluate OBJ for each possible frame to add). Assuming one can train linear SVMs in linear time, this would make computation $\mathcal{O}(N^3)$. We make two modifications to considerably reduce computation. First, we only apply these expensive operations on batches of frames that double in size (Line 8). Secondly, we re-use solutions of previously-solved SVMs to warm-start new SVM problems. We do this by initializing the dual coordinate descent solvers of

```
Input: $y_1$
$t \leftarrow 1$;
$\Lambda \leftarrow \{(t, y_t)\}$;
$w \leftarrow LEARN(\Lambda)$;
while $t < N$ do
    $y_{1:t} \leftarrow TRACK(y_1, w, t)$;
    $\Lambda \leftarrow SELECT_{t/2}(\Lambda, y_{1:t})$;
    $w \leftarrow LEARN(\Lambda)$;
    $t \leftarrow 2t$
end
```

Algorithm 3: For each frame t, our online algorithm outputs the best track found up until then. However, every power-of-two frames, it learns a new model while revisiting past frames to correct mistakes made under previous models. This allows our tracker be linear time $\mathcal{O}(N)$ while still being a "retrospective" learner.

[58] with previously-computed dual variables. In practice, we find that evaluating OBJ for each possible frame is constant time because only a single convolution is required (discussed further in Sec. 5.4). This means that the $SELECT$ function scales linearly with t, making each iteration of the main loop $O(t)$. Because our batch procedure requires only $\log_2 N$ iterations, total computation is $N + N/2 + N/4 + \ldots = \boxed{\mathcal{O}(N)}$. We also observe a linear scaling of computation in practice.

5.3.2 Offline tracker

Alg. 4 describes our off-line algorithm. It operates similarly to our online algorithm, but it has access to the entire set of frames in a video. We iterate between tracking (over the whole video) and learning (from a select subset of frames) for a fixed number of iterations K. As in curriculum learning, we found it useful to learn from the easy cases first. We exponentially grow the number of selected frames such that at the last iteration, 50% of all frames are selected. We found that little is gained by iterating more than $K = 4$ times (shown qualitatively in Fig. 5.4 and quantitatively in Fig. 5.6).

```
**Input:** $y_1$
$\Lambda \leftarrow \{(1, y_1)\}$;
$w \leftarrow LEARN(\Lambda)$;
**for** $i = 1 : K$ **do**
    $y_{1:N} \leftarrow TRACK(y_1, w, N)$;
    $\Lambda \leftarrow SELECT_r(\Lambda, y_{1:N})$   where   $r = \frac{N}{2}^{\frac{i}{K}}$;
    $w \leftarrow LEARN(\Lambda)$;
**end**
```

Algorithm 4: Our offline algorithm performs a fixed number (K) of iterations. During each iteration, it updates the track, selects the r "easiest" new frames of training examples, and re-trains using these examples.

5.4 Implementation

We now discuss various implementation details for applying our algorithms.

Tracking: We implement the $TRACK$ subroutine by using dynamic programming, which requires $\mathcal{O}(s^2 N)$ time where N is the number of frames and s is the number of states per-frame. To reduce computational costs we limit s to be the top 25 new detections. Thus, assuming all videos are of some fixed resolution, the complexity of our tracking algorithm is $\boxed{\mathcal{O}(N)}$ given a fixed model.

Repeatedly-learning SVMs: Each call to $LEARN$ requires training a single SVM, and each call to $SELECT$ requires training (t) SVMs, needed to evaluate the SVM objective OBJ for each possible frame to select. Training each SVM from scratch would be prohibitively slow due to our massive negative training sets. We now show how to use the dual coordinate descent method of [58] to "warm-start" SVM training using previous solutions.

Dual QP: We write the dual of (5.1) by writing a training example as x_i and its label $y_i \in \{-1, 1\}$:

$$\max_{0 \leq \alpha \leq C} -\frac{1}{2} \sum_{i,j} \alpha_i y_i x_i \cdot x_j y_j \alpha_j + \sum_i \alpha_i \tag{5.4}$$

the KKT conditions allow us to reconstruct the primal weight vector as $w = \sum \alpha_i y_i x_i$. We represent our models both with a weight vector w, and implicitly through a cached collection of non-zero dual variables $\{\alpha_i\}$ and support vectors $\{x_i\}$. We include all positive examples in our cache (even if they are not a support vector) because they require essentially no storage, and may become support vectors during subsequent optimizations as described below.

Warm-start: Given a model w and its dual variables α_i and support vectors x_i, we can quickly learn a new model and estimate the increase in OBJ due to adding an additional frame t. We perform one pass of coordinate descent on examples from this new frame as follows: we run the model w on frame t and cache examples with a non-zero gradient in the dual objective (5.4). One can show this is equivalent to finding margin violations; e.g., negative examples that score greater than -1 and positive examples that score less than 1 [58]. This can be done with a single convolution that evaluates the current model w on frame t. In practice, we find that our dual QP converges after a small fixed number of coordinate descent passes over the cache, making the overall training time dominated by the single convolution.

5.5 Results

Benchmark evaluation: We define a test suite of videos and ground truth labelings by merging the test videos of [12, 97]. We show a sampling of frames in Fig. 5.5. Previous approaches evaluate mean displacement in pixels or thresholded detection accuracy. In our experience, displacement error can be ambiguous because it is not scale-invariant and can be somewhat arbitrary once an algorithm looses track. Furthermore, pixel displacement is undefined for frames where the object is occluded or leaves the camera view (common in long-scale tracking). Instead, we use follow [97] and define an estimated object location as correct if it sufficiently overlaps the ground-truth. We then compute true positive and

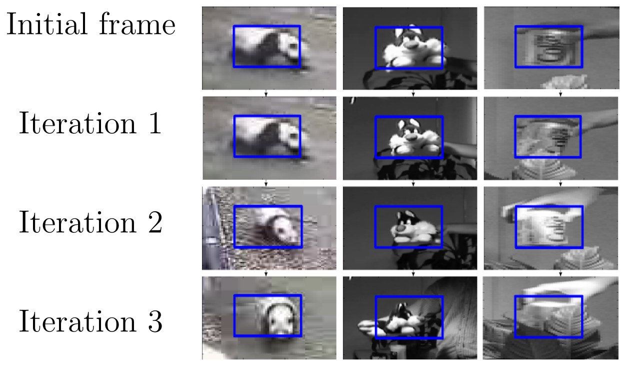


Figure 5.4: On the **top**, we show the initial labeled frame for 3 videos. In the next **3 rows**, we show specific examples that are added over 3 iterations of offline learning. Our model slowly expands to capture more difficult appearances (or "concepts" in curriculum learning).

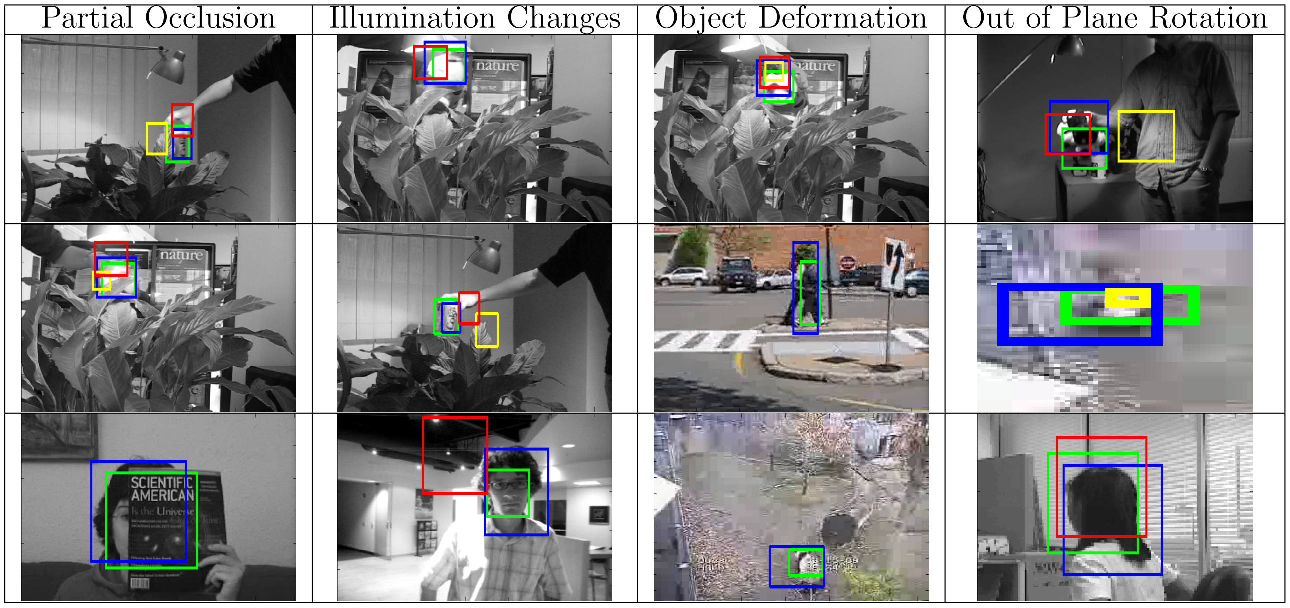


Figure 5.5: Our test videos present numerous challenges, including (from left to right): partial occlusion, illumination changes, object deformation, and out of plane rotation. Ground Truth is Green . TLD is Yellow ; MIL Track is Red ; Our Tracker is Blue . If there is no rectangle for a tracker, then that tracker signaled occlusion for the frame. In general, our system is able to track correctly through such challenging scenarios. Our large negative training sets and retrospective learning greatly reduce the probability of false positives.

missed detections, producing a final F_1 score. To further spur innovation, we have released our combined benchmark dataset, along with all our source code (available through the author's website).

Overall performance: We evaluate our performance against state-of-the-art trackers with published source (TLD and MIL Track) code in Fig. 5.7 and Table 5.1. TLD requires a minimum window size for the learned model; the default value (of 24×24 pixels) was too large for many of the benchmark videos. We manually tuned this to 16 pixels, but this reduced performance on other videos (because this increases the number of possible candidate locations). We show results for both parameter settings, including results for tuning this parameter on a *per-video* basis. On two videos, MILTrack loses track by frame 500 and so we only report accuracy over those initial 500 frames. Even when giving past work these unfair advantages, our final system (without any per-video tuning) significantly outperforms past work $F_1 = 93\%$ vs $F_1 = 76\%$. Our online algorithm slightly underperforms our offline variant, with an average F_1 of 91%. For completeness, we also include results for mean displacement error (for the subset of videos with no occlusion or field-of-view exits) in Table 5.3. In terms of displacement error, our method compares favorably to much recent work, but does not quite match the recent performance of [167]. We hypothesize this gap exists because we resolve object location up to a HOG cell and not an individual pixel. We posit that overlapping cells or post-processing temporal smoothing would likely reduce our displacement error.

Speed: Our final algorithm runs at about $\frac{1}{20}^{th}$ real-time. In the next diagnostic section, we describe variants, many of which are real-time while still outperforming prior art. The current bottleneck of our algorithm is the repeated evaluation of detectors on image frames. Because this convolution operation is straightforward to parallelize, we believe our full system could also operate at real-time given additional optimizations.

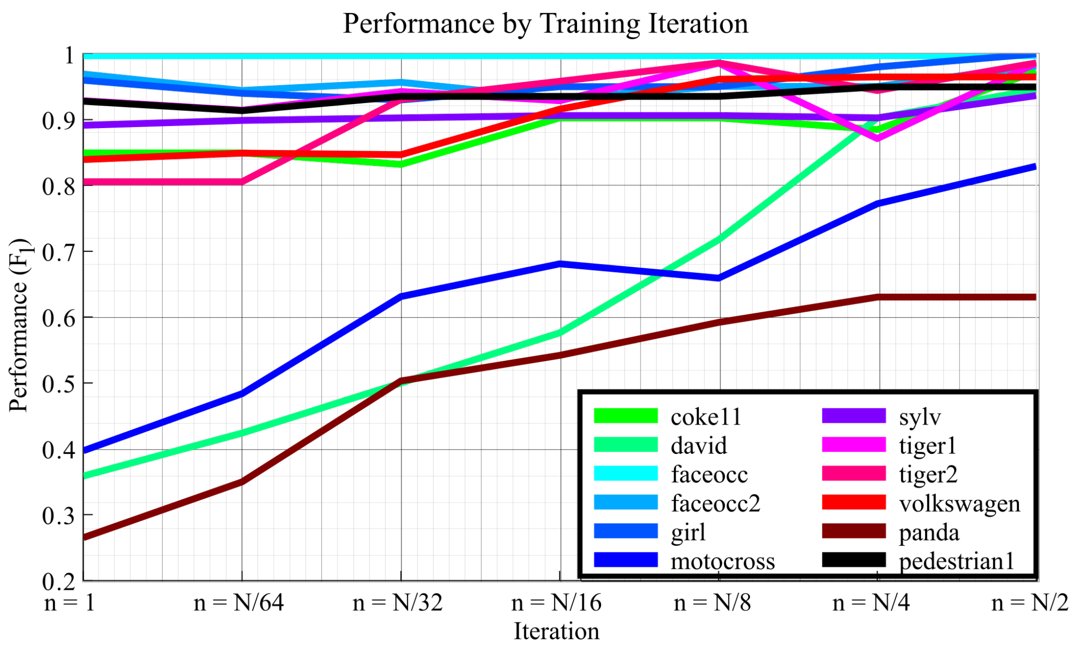


Figure 5.6: Offline learning in action. As we increase the number of latently labeled training frames (from 1 to 50%), performance generally increases. For many videos, the initial model learned on the first frame is already quite accurate. We discuss this somewhat surprising observation at length in the text.

5.5.1 Diagnostics

In Table 5.2, we analyze various aspects of our system that are important for good results. We point out those observations that seem inconsistent with the accepted wisdom in the tracking literature. We refer to specific rows in the table using its label in the first column (e.g., "D1").

Detection: We begin with a "naive" tracking-by-detection baseline. Our baseline $LEARN$s a detector from the initial labeled frame, and simply returns the highest scoring location at each subsequent frame. Surprisingly, *even without additional learning or tracking*, our baseline (D1) produces a F_1 score of 73%, outperforming MILTrack and comparable to TLD. Most prior work on learning for detection uses a small set of negatives, usually extracted from windows near the object. We compare to such an approach (D2), and show that using a large set of negatives is crucial for good performance. We see this observation as emphasizing an under-appreciated connection between tracking and detection; it is well-known in the object detection community that large training sets of negatives are crucial for good performance [45].

Motion: Adding a motion model to our baseline detector improves performance from 73% (D1) to 76% (D5,D6). However, a standard first-order motion model (that favors stationary objects) is not particularly effective, improving performance to only 74% (D4). Our optical-flow-based motion model works much better. A single-hypothesis greedy tracker – that greedily enforces the dynamic model in (5.2) given the best location in the previous frame – improves performance to 76% (D3). This suggests that multiple hypothesis tracking may not be crucial for good performance. Furthermore, the improvement due to motion in our final system with learning is even lower; without *any* motion model, we still perform at 89% (D8). We find that *the better the detector, the less advantage can be gained from a motion model.*

Learning: Learning is the most crucial aspect of our system, improving performance from 76% (D5) to 91% (C6) for our online algorithm (and even more for our off-line). We construct a restricted version of our online algorithm that does not require revisiting previous frames. We do this by not allowing SELECT to accept a previously-rejected frame, and not allowing TRACK to edit a previously-estimated location. This restriction virtually eliminates all benefits of learning, producing a F_1 of 77% (D7). This suggests that its vital to edit previous tracks to produce better examples for *retrospective* learning. Secondly, naively $SELECT$ing all previously-seen frames for learning also significantly decreases performance to 84% (D9). This suggests that selecting a good subset of reliable frames is also important. Finally, our OBJ-based criteria for frame selection outperforms the traditional SVM response, 91% (C6) vs 89% (D10). The performance increase is largest (up to 10%) for difficult videos such as Panda and Motocross.

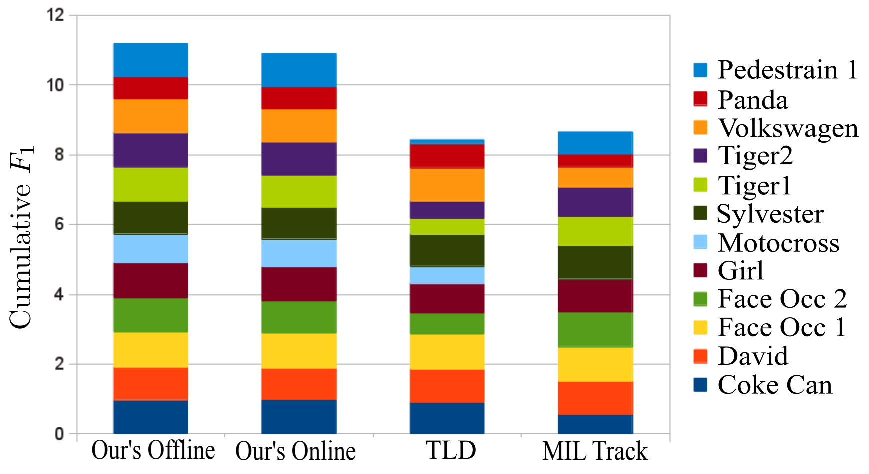


Figure 5.7: We compare F_1 scores, accumulated across 12 benchmark test videos. The maximum accumulated score is 12. Our trackers significantly reduce error, compared to prior work. Most of the improvement comes from a few "challenging" videos containing significant occlusions and scale challenges, including Pedestrian, Panda, and Motorcross. See Table 5.1 for additional numeric analysis.

5.5.2 Conclusion

We have a described a simple but effective system for tracking based on the selection of trustworthy frames for learning appearance. We have performed an extensive diagnostic analysis across a large set of benchmark videos that reveals a number of surprising results. We find the task of learning good appearance models to be crucial, as compared to say, maintaining multiple hypothesis for tracking. To learn good appearance models, we find it important to use large sets of negative training examples, and to *retrospectively* edit and select previous frames for learning. To do so in a principled and efficient manner, we use the formalism of self-paced learning and online solvers for SVMs.

Our tracker handles long videos with periods of occlusion/absence and large scale changes. Our benchmark analysis suggests we do so with significantly better accuracy than prior art. Our algorithms are linear time and so asymptotically efficient. We believe that a parallelized implementation could easily lead to real-time performance with significant real world applications.

Benchmark video comparison (F_1 score)

#		Coke	David	Face1	Face2	Girl	Moto-x	Sylvester	Tiger1	Tiger2	VW	Panda	Pedestrian	**Mean**
C1	LK-FB [96]	40.00	100.0	99.43	99.43	100.0	00.04	72.38	28.16	27.39	04.00	05.13	10.71	**42.09**
C2	MIL Track[12]	55.00	95.00	99.44	100.0	93.00	01.00†	97.00	82.00	85.00	58.00†	36.27	63.76	**72.12** †
C3	TLD (min=24) [97]	ERR	96.77	41.96	99.43	88.29	ERR	94.40	62.85	48.64	ERR	ERR	ERR	**ERR**
C4	TLD (min=16)	90.26	95.45	100.0	60.81	85.08	48.58	92.50	44.06	50.48	95.24	69.68	11.04	**70.26**
C5	TLD (min=better)	90.26	96.77	100.0	99.43	88.29	48.58	94.40	62.85	50.48	95.24	69.68	11.04	**75.59**
C6	Us [On-line]	98.21	90.32	99.71	93.25	98.02	79.08	91.01	91.01	95.83	95.46	62.09	95.71	**90.81**
C7	Us [Off-line]	97.39	94.62	99.71	98.16	100.0	82.95	93.65	98.59	98.63	96.48	63.05	95.00	**93.18**

Table 5.1: Comparison of methods (F_1, higher is better). ERR indicates a tracker did not run given the size of the initial object bounding box. † indicates a tracker was evaluated only on the initial (500) frames before it lost track. Our trackers place special emphasis on long term tracking and can thus recover from such failures. Both online and offline versions of our algorithm significantly outperform prior work, including various versions of TLD tuned with different hyper-parameters.

Diagnostic analysis (without learning)

#	Version	Coke	David	Face1	Face2	Girl	Moto-x	Sylvester	Tiger1	Tiger2	VW	Panda	Pedestrian	**Mean**
D1	Tracking by Detection	88.69	33.33	99.71	96.31	95.05	25.41	88.80	90.14	83.56	67.43	31.97	75.71	**73.01**
D2	TBD with sub-sampling	73.04	17.20	37.08	38.65	35.64	33.80	81.72	50.70	47.95	70.99	7.89	12.86	**42.29**
D3	Single track hypothesis	78.00	24.24	99.71	96.31	96.04	37.26	90.67	92.95	86.30	82.95	28.79	95.00	**75.68**
D4	On-line w/o flow	85.21	36.55	96.31	99.71	96.04	26.26	89.92	92.95	80.28	83.22	30.29	69.74	**73.87**
D5	On-line	85.96	33.51	99.71	96.93	91.08	34.60	89.55	92.95	84.72	83.11	29.63	91.75	**76.12**
D6	Off-line	85.21	36.55	99.71	96.93	96.04	39.76	89.17	92.95	80.82	83.94	26.52	92.85	**76.70**
	Diagnostic analysis (with learning)													
D7	Fix TRACK & SELECT	91.43	35.03	99.72	93.25	95.05	29.76	91.15	91.18	80.00	89.71	40.79	89.61	**77.22**
D8	On-line w/o motion	97.39	88.17	99.72	93.87	99.01	64.04	92.16	94.37	93.15	95.66	53.06	95.71	**88.86**
D9	On-line w/ SelectAll	97.39	91.40	99.72	88.34	100.00	73.35	80.46	97.87	73.38	70.18	43.74	90.97	**83.90**
D10	On-line w/ RespSelect	98.21	89.24	99.71	95.70	94.05	72.06	88.67	98.57	93.70	92.07	52.64	93.57	**89.02**

Table 5.2: Diagnostic analysis of our system (F_1, higher is better), with and without learning. Please see Sec. 5.5.1 for a detailed discussion.

Mean Center Displacement Error (MCDE)

Tracker	Us[C6]	Us[C7]	ART[167]	LRSVT[13]	TLD	MIL[12]	Frag[3]	PROST[196]	NN[73]	OAB1[71]
Code	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes
Sylv.	12	12	6	9	14	11	11	-	-	25
Girl	17	29	11	-	21	32	27	19	18	48
David	10	7	3	7	7	23	46	15	16	49
Face1	10	11	8	12	18	27	6	7	7	44
Face2	14	13	6	10	17	20	45	17	17	21
Tiger1	4	4	5	8	15	15	40	-	-	35
Tiger2	7	5	5	13	30	17	38	-	-	34
Coke	7	7	-	12	9	21	63	-	-	25
Mean	**10**	**11**	**6***	**10***	**16**	**21**	**35**	**15***	**15***	**35**

Table 5.3: Mean Center Displacement Error (MCDE), where lower is better. * indicates a given tracker did not report results on all videos. Though we believe the F_1 score (see Table 5.1) is a more accurate measure, we report MCDE for completeness. Our trackers generally compare favorably to other state of the art algorithms. We conjecture that our pixel displacement error would decrease if we resolved object locations up to pixels and not just HOG cells. See Sec. 5.5 for more discussion.

Chapter 6

Learning to Make Decisions

James Steven Supančič III & Deva Ramanan

6.1 Introduction

In the previous chapter, we looked at the problem of long-term model-free tracking. Model-free tracking is one of the basic computational building blocks of video analysis, relevant for tasks such as general scene understanding and robot perception. Recall that in model-free tracking's standard formulation, a tracker must follow an unknown object given only a single bounding box for initialization [111, 247]. In model-free tracking, most of the recent state-of-the-art advances make heavy use of online machine learning at test time [111, 242, 166, 249, 95]. Systems often produce impressive results by improving core components, such as appearance and motion models.

Challenges: We see two significant challenges that limit further progress in model-free tracking. First, the *limited quantity of annotated video data* impedes both training and evaluation. While image datasets involve millions of images for training and testing, tracking

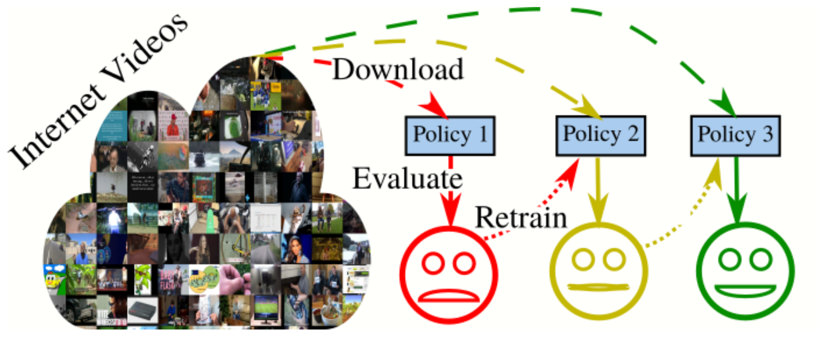


Figure 6.1: **Streaming training & evaluation**: We propose an iterative procedure for interactively training trackers from data. We *download* a new video from the Internet and run the current tracker on it, *evaluate* the tracker's performance with interactive rewards, and then *retrain* the tracker policy with the reward signals. This scheme allows us to train and evaluate our tracker on massive streaming datasets.

datasets have hundreds of videos. Lack of data seems to arise from the difficulty of annotating videos, as opposed to images. Second, as vision (re)-integrates with robotics, video processing must be done in an online, streaming fashion. In terms of tracking, this requires a tracker to make *on-the-fly decisions* such as when to *re-initialize* itself [240, 82, 136, 216, 98] or *update* its appearance model (the so-called template-update problem [242, 98]). Such decisions are known to be crucial in terms of final performance, but are typically hand-designed rather than learned.

Contribution 1 (interactive video processing): We show that reinforcement learning (RL) can be used to address both challenges in distinct ways. In terms of data, rather than requiring videos to be labeled with detailed bounding-boxes at each frame, we interactively train trackers with far more limited supervision (specifying binary rewards/penalties only when a tracker fails). This allows us to train on massive video datasets that are 100× larger than prior work. Interestingly, RL also naturally lends itself to streaming "open-world" evaluation: when running a tracker on a never-before-seen video, the video can be used for both evaluation of the current tracker and for training (or refining) the tracker for future use (Fig. 6.1). This streaming evaluation allows us to train and evaluate models in an integrated fashion seamlessly. For completeness, we also evaluate our learned models on standard tracking benchmarks.

Contribution 2 (tracking as decision-making): In terms of tracking, we model the tracker itself as an *active agent* that must make online decisions to maximize its reward, which is (as above) the correctness of a track. Decisions ultimately specify where to devote finite computational resources at any point of time: should the agent process only a limited region around the currently predicted location (e.g.,"track"), or should it globally search over the entire frame ("reinitialize")? Should the agent use the predicted image region to update its appearance model for the object being tracked ("update"), or should it be "ignored"? Such decisions are notoriously complicated when image evidence is ambiguous (due to say, partial occlusions): the agent may continue tracking an object but perhaps decide not to update its model of the object's appearance. Rather than defining these decisions heuristically, we will ultimately use data-driven techniques to learn good *policies* for active decision-making (Fig. 6.2).

Contribution 3 (deep POMDPs): We learn tracker decision policies using reinforcement learning. Much recent work in this space assumes a Markov Decision Process (MDP), where the agent observes the true state of the world [148, 249], which is the true (possibly 3D) location and unoccluded appearance of the object being tracked. In contrast, our tracker only assumes that it receives *partial* image observations about the world state. The resulting partially-observable MDP (POMDP) violates Markov independence assumptions: actions depend on the entire history of observations rather than just the current one [93, 194]. As in [80, 103], we account for this partial observability by maintaining a *memory* that captures *beliefs* about the world, which we update over time (Sec. 6.3). In our case, beliefs capture object location and appearance, and action policies specify how and when to update those beliefs (e.g., how and when should the tracker update its appearance model) (Sec. 6.4). However, policy-learning is notoriously challenging because actions can have long-term

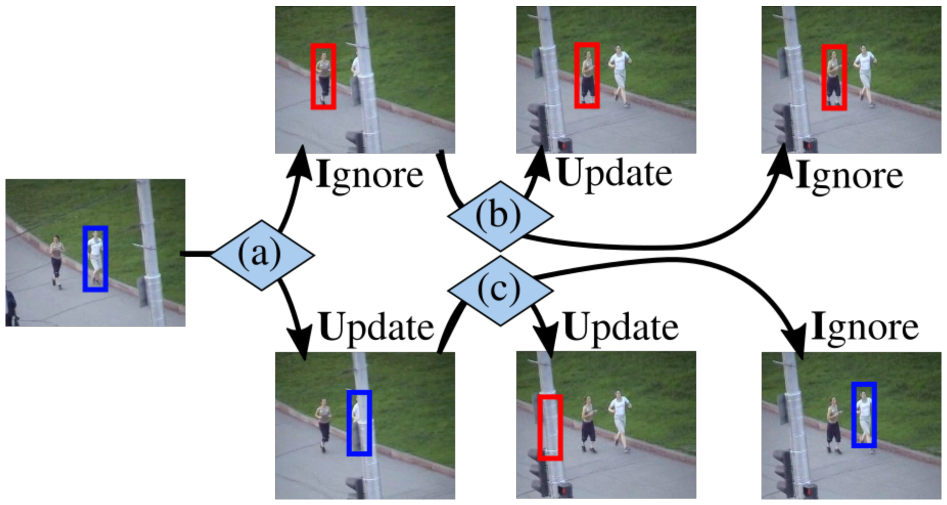


Figure 6.2: **Decisions in tracking**: Long-term trackers must decide when to update their appearance models and when to re-initialize. This requirement leads us to formulate tracking as a sequential decision problem. In this example, we isolate the update decision. A tracker should often update using good localizations (blue boxes) because such updates improve the tracker's appearance model (a)-Update. Without such updates, the appearance model may struggle detect occlusions and to avoid distractors during reinitialization (a)-Ignore. But a tracker cannot always update: Updates during tracking failures (red boxes) virtually always reduce tracking accuracy (b). And if a tracker always updates, it may drift and begin tracking occluders (e.g. the pole) (c). Thus, a tracker must have a decision policy which strikes a careful balance. We learn such a policy from data.

effects on future beliefs. To efficiently learn policies, we introduce frame-based heuristics that provide strong clues as to the long-term effects of taking a particular action (Sec. 6.5).

6.2 Related Work

Tracking datasets: Several established benchmarks exist for evaluating trackers [247, 111]. Interestingly, there is evidence to suggest that many methods tend to overfit due to aggressive tuning [166]. Withholding test data annotation and providing an evaluation server addresses this to some extent [124, 193]. Alternatively, we propose to evaluate on an open-world stream of Internet videos, making over-fitting impossible by design. It is well-known that algorithms trained on "closed-world" datasets (say, with centered objects against clean backgrounds [165, 21]) are difficult to generalize to "in-the-wild" footage [229]. We invite the reader to compare our videos in the supplementary material to contemporary video benchmarks for tracking.

Interactive tracking: Several works have explored interactive methods that use trackers to help annotate data. The computer first proposes a track. Then a human corrects major errors and retrains the tracker using the corrections [29, 5, 235]. Our approach is closely inspired by such active learning formalisms but differs in that we make use of minimal supervision in the form of a binary reward (rather than a bounding box annotation).

Learning-to-track: Many tracking benchmarks tend to focus on short-term tracking ($<$ 2000 frames per video) [247, 111]. In this setting, a central issue appears to be modeling the appearance of the target. Methods that use deep features learned from large-scale training data perform particularly well [240, 241, 135, 129, 154]. *Our focus on tracking over longer time frames poses additional challenges - namely, how to reinitialize after cuts, occlusions and failures, despite changes in target appearance [54, 242].* Several trackers address these challenges with hand-designed policies for model updating and reinitialization - TLD [98], ALIAN [168] and SPL [216] explicitly do so in the context of long-term tracking. On the other hand, our method takes a data-driven approach and learns policies for model-updating and reinitialization. Interestingly, such an "end-to-end" learning philosophy is often embraced by the multi-object tracking community, where strategies for online reinitialization and data association are learned from data [121, 249, 131]. Most related to our work are Xiang et al. [249], who use an MDP for multi-object tracking, and [103], who use RL for single target tracking. Both works use heuristics to reduce policy learning to a supervised learning task, avoiding the need to reason about rewards in the far future. The robotics community has developed techniques to accelerate RL using human demonstrations [123], interactive feedback [107, 227], and hand-designed heuristics [23]. Using heuristic functions

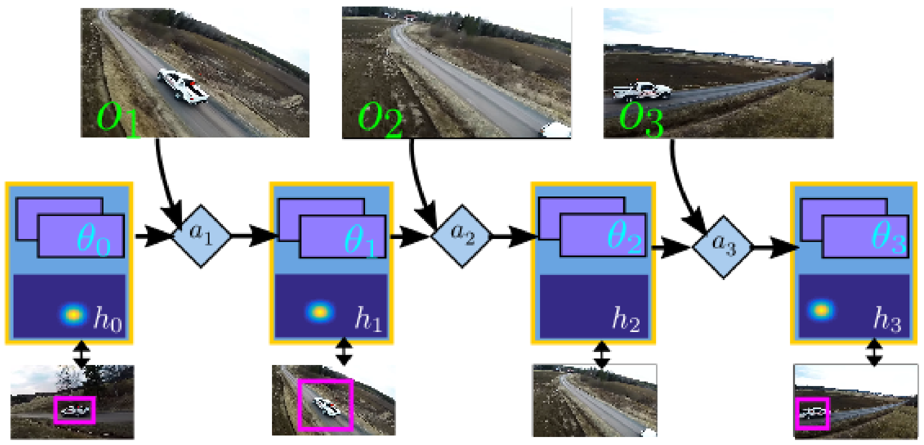


Figure 6.3: **Tracker architecture:** At each frame i, our tracker updates a location heatmap h_i for the target using the current image observation o_i, a location prior given by the previous frames' heatmap h_{i-1}, and the previous appearance model θ_{i-1}. Crucially, our tracker learns a policy for actions a_i that optimally update h_i and θ_i (6.2).

for initialization [23, 107], our experimental results show that explicit Q-learning outperforms supervised reductions because it can learn to capture long-term effects of taking particular actions.

Real-time tracking through attention: An interesting (but perhaps unsurprising) phenomenon is that better trackers tend to be slower [111]. Indeed, on the VOT benchmark, most recent trackers do not run in real time. Generally, trackers that search locally [206, 234] run faster than those that search globally [216, 136, 85]. To optimize visual recognition efficiency, one can learn a policy to guide selective search or attention. Inspired by recent work which finds a policy for selective search using RL [95, 147, 164, 37, 92, 16, 141], we aslo learn a policy that decides whether to track (i.e. , search positions near the previous estimate) or reinitialize (i.e. , search globally over the entire image). But in contrast to this prior work, we additionally learn a policy to decide when to update a tracker's appearance model. To ensure that our tracker operates with a fixed computational budget, we implement reinitialization by searching over a random subset of positions (equal in number to those examined by track).

6.3 POMDP Tracking

We now describe our POMDP tracker, using standard notation where possible [194]. For our purposes, a POMDP is defined by a tuple of states (Ω, O, A, B): At each frame i, the world state $\omega_i \in \Omega$ generates a noisy observation $o_i \in O$ that is mapped by an agent into an action $a_i \in A$, which in turn generates a reward.

In our case, the state ω_i captures the true location and appearance of the object being tracked in frame i. To help build intuition, one can think of the location as 2D pixel coordinates and appearance as a 2D visual template. Instead of directly observing this world state, the tracking agent maintains a belief over world states, written as

$$b_i = (\theta_i, h_i), \quad \text{where} \quad \theta_i \in R^{h\times w\times f}, h_i \in R^{H\times W}$$

where θ_i is a distribution over appearances (we use a point-mass distribution encoded by a single $h \times w$ filter defined on f convolutional features), and h_i is a distribution over pixel positions (encoded as a spatial heatmap of size $H \times W$). Given the previous belief b_{i-1} and current observed video frame o_i, the tracking agent updates its beliefs about the current frame b_i. Crucially, *tracker actions a_i specify how to update its beliefs*, that is, whether to update the appearance model and whether to reinitialize by disregarding previous heatmaps. From this perspective, our POMDP tracker is a memory-based agent that learns a *policy* for when and how to update its own memory (Fig. 6.3).

Specifically, beliefs are updated as follows:

$$h_i = \begin{cases} TRACK(h_{i-1}, \theta_{i-1}, o_i) & \text{if} \quad a_i^{(1)} = 1 \\ REINIT(\theta_{i-1}, o_i) & \text{otherwise} \end{cases} \tag{6.1}$$

$$\theta_i = \begin{cases} UPDATE(\theta_{i-1}, h_i, o_i) & \text{if} \quad a_i^{(2)} = 1 \\ \theta_{i-1} & \text{otherwise} \end{cases}$$

where $a_i = (a_i^{(1)}, a_i^{(2)})$. Object heatmaps h_i are updated by running the current appearance model θ_{i-1} on image regions o_i near the target location previously predicted using h_{i-1} ("tracking"). Alternatively, if the agent believes it has lost track, it may globally evaluate its appearance model ("reinit"). The agent may then decide to "update" its appearance model using the currently-predicted target location, or it may leave its appearance model unchanged. In our framework, the tracking, reinitialization, and appearance-update modules can be treated as black boxes given the above functional form. We will discuss particular implementations shortly, but first, we focus on the heart of our RL approach: a principled framework for learning to take appropriate actions. To do so, we begin by reviewing standard approaches for decision-making.

Online heuristics: The simplest approach to picking actions might be to pre-define heuristics functions which estimate the correct action. For example, many trackers reinitialize whenever the maximum confidence of the heatmap is below a threshold. Let us summarize the information available to the tracker at frame i as a "state" s_i, which includes the previous belief b_{i-1} (the previous heatmap and appearance model) and current image observation o_i.

$$a_i^* = \text{Heur}_{on}(s_i), \quad \text{where} \quad s_i = (b_{i-1}, o_i) \tag{6.2}$$

Offline heuristics: A generalization of the above is to use *offline* training data to build better heuristics. Crucially, one can now make use of ground-truth training annotations as well as future frames to better gauge the impact of possible actions. We can write this knowledge as the true world state $\{\omega_i, \forall i\}$. For example, a natural heuristic may be to reinitialize whenever the predicted object location does not overlap the ground-truth position for that frame. Similarly, one may update appearance whenever doing so improves the confidence of ground-truth object locations across future frames:

$$a_i^* = \text{Heur}_{off}(s_i, \{\omega_i : \forall i\}) \tag{6.3}$$

Crucially, these heuristics cannot be applied at test time because ground truth is not known! However, they can generate per-frame target action labels a_i^* on training data, effectively reducing policy learning to a standard *supervised-learning problem.* Though offline heuristics appear to be a simple and intuitive approach to policy learning, we have not seen them widely applied for learning tracker action policies.

Q-functions: Offline heuristics can be improved by unrolling them forward in time: the benefit of a putative action can be better modeled by applying that action, processing the next frame, and using the heuristic to score the goodness of possible actions in that next frame. This intuition is formalized through the well-known *Bellman equations* that recursively define Q-functions to return a score (the expected future reward) for each putative action a:

$$Q(s_i, a_i) = R(s_i) + \gamma \max_{a_{i+1}} Q(s_{i+1}, a_{i+1}), \tag{6.4}$$

where s_i includes both the tracker belief state and image observation, and $R(s_i)$ is the reward associated with the reporting the estimated object heatmap h_i. We let $R(s_i) = 1$ for a correct prediction and 0 otherwise. Finally, $\gamma \in [0, 1]$ is a discount factor that trades off immediate vs. future per-frame rewards. Given a tracker state and image observation s_i, the optimal action to take is readily computed from the Q-function:

$$a^* = \arg\max_a Q(s_i, a)$$

Q-learning: Traditionally, Q-functions are iteratively learned with Q-learning [194]:

$$\begin{aligned} Q(s_i, a_i) \Leftarrow & Q(s_i, a_i) + \dots \\ & \alpha\Big(R(s_i) + \gamma \max_{a_{i+1}} Q(s_{i+1}, a_{i+1}) - Q(s_i, a_i)\Big) \end{aligned} \tag{6.5}$$

where α is a learning rate. To handle continuous belief states, we approximate the Q-function with a CNNs:

$$Q(s_i, a_i) \approx CNN(s_i, a_i)$$

that processes states s_i and binary actions a_i to return a scalar value capturing the expected future reward for taking that action. Recall that a state s_i encodes a heatmap and an appearance model from previous frames and an image observation from the current frame.

6.4 Interactive Training and Evaluation

In this section, we describe our procedure for interactively learning CNN parameters w (that encode tracker action policies) from streaming video datasets. To do so, we gradually build up a database of *experience replay memories* [4, 148], which are a collection of state-action-reward-nextstate tuples $\mathcal{D} = \{(s_i, a_i, r_i, s_{i+1})\}$:

$$L(w, \mathcal{D}) = \Big(r_i + \max_{a_{i+1}} Q_w(s_{i+1}, a_{i+1}) - Q_w(s_i, a_i)\Big)^2 \tag{6.6}$$

Q-learning reduces policy learning to a *supervised regression problem* by unrolling the current policy one step forward in time. Gradient descent on the above objective is performed as follows: given a training sample (s_i, a_i, r_i, s_{i+1}), first perform a forward pass to compute the current estimate $Q_w(s_i, a_i)$ and the target: $r_i + \max_{a_{i+1}} Q_w(s_{i+1}, a_{i+1})$. Then backpropagate through the weights w to reduce $L(w)$.

Dataset	# Videos	# Frames	Annotations	Type
OTB-2013 [247]	50	29,134	29,134	AABB
PTB [207]	50	21,551	21,551	AABB
VOT-2016 [111]	60	21,455	21,455	RBB
ALOV++ [205]	315	151,657	30,331	AABB
NUS-PRO [124]	365	135,310	135,310	AABB
Ours	16,384	10,895,762	108,957	Rewards

Table 6.1: **Our streaming** dataset and evaluation enable us to evaluate on a larger number of videos than prior work. We annotate rewards, while the other datasets provide Axis Aligned (AABB) or Rotated (RBB) Bounding Boxes.

```
while True do
    Download random video;
    $\theta_1 \leftarrow$ UPDATE$(h_1, o_1)$; /* manually init.                    */
    forall $i \in$ video do
        $s_i = ((\theta_{i-1}, h_{i-1}), o_i)$;
        /* track or reinitialize?                                              */
        if $Q_w(s_i, \boldsymbol{t}) > Q_w(s_i, \boldsymbol{r})$ then
            $h_i \leftarrow$ TRACK$(h_{i-1}, \theta_{i-1}, o_i)$; $a_i^{(1)} \leftarrow 1$;
        end
        else $h_i \leftarrow$ REINIT$(\theta_{i-1}, o_i)$; $a_i^{(1)} \leftarrow 0$ ;
        /* update or ignore?                                                   */
        if $Q_w(s_i, \boldsymbol{m}) > Q_w(s_i, \boldsymbol{i})$ then
            $\theta_i \leftarrow$ UPDATE$(\theta_{i-1}, h_i, o_i)$; $a_i^{(2)} \leftarrow 1$;
        end
        else $\theta_i \leftarrow \theta_{i-1}$ ;$a_i^{(2)} \leftarrow 0$ ;
        /* manually evaluate the performance                                   */
        $r_i \leftarrow$ annotated frame correct? ;
        /* update experience database                                          */
        $\mathcal{D} \leftarrow \mathcal{D} \cup (s_i, a_i, r_i, s_{i+1})$
    end
    $w \leftarrow \arg\min L(w; \mathcal{D})$
end
```

Algorithm 5: Our final learning algorithm interactively labels a streaming dataset of videos while learning a tracker action policy Q_w. Given a video, steps 5 through 11 run a tracker according to the current policy. An annotator then assesses the binary reward r_i (correctness) for the highest-scoring bounding box extracted from the heatmap h_i by using an intersection-over-union threshold. Annotated frames (and their associated state-action-reward-nextstate tuples) are added to our experience replay database D. We then sample a minibatch of replay memories and update the action policy w with backprop (Eq. 6.6).

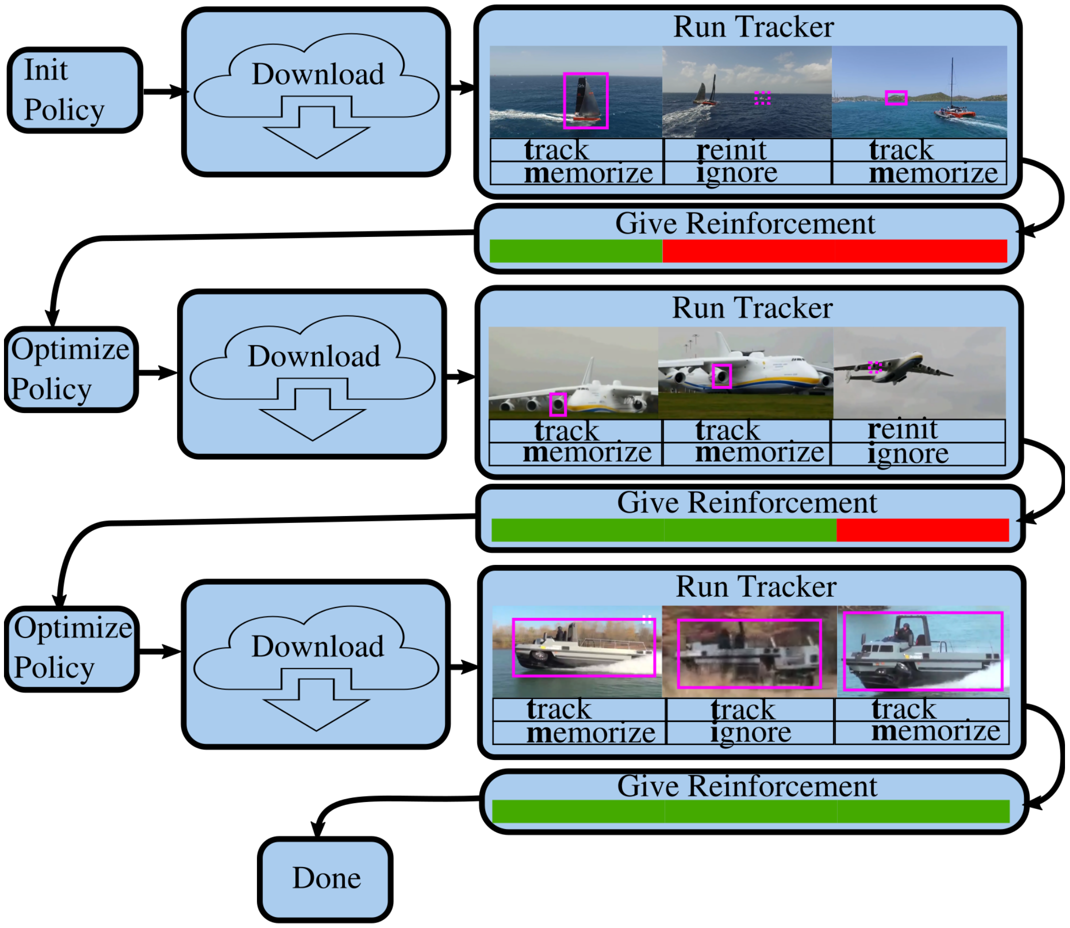


Figure 6.4: **Our final reinforcement learning** algorithm's (Alg. 5) flowchart is shown above.

The complete training algorithm is written in Alg. 5 and illustrated in Fig. 6.4. We choose random videos from the Internet by sampling phrases using WordNet [146]. Given the sampled phrase and video, an annotator provides an initialization bounding box and begins running the existing tracker. After tracking, the annotator marks those frames (in strides of 50) where the tracker was incorrect using a standard 50% intersection-over-union threshold. Such binary annotation ("correct" or "failed") requires less time per frame than bounding-box annotation: we can perform 135 annotations per minute versus 22. By May 23^{rd} of 2017, our dataset had grown to over 10 million frames (used for *both* training and evaluation), dramatically exceeding prior work (Table 6.1). After running our tracker and interactively marking failures, we use the annotation as a reward signal to update the policy parameters for the next video. Thus each video is used to both evaluate the current tracker and train it for future videos. To avoid any possibility of "training on test data", we always evaluate performance on videos not yet used for training the tracker.

6.5 Implementation

In this section, we discuss several implementation details. We begin with a detailed overview of our tracker's test time pipeline. The implementation details for the TRACK, REINIT, UPDATE, and Q-functions follow. Finally, we describe the offline heuristics used at train time.

Overview of p-tracker's pipeline: In Fig. 6.5 we show a flowchart for our tracker's test time pipeline. We use a UML Activity Diagram [191]. We describe each step in detail in Alg. 6. We now describe how we chose to implement the TRACK, REINIT, UPDATE and Q-functions.

TRACK/REINIT functions: Our TRACK function (Eq. 6.2) takes as input the previous heatmap h_{i-1}, appearance model θ_{i-1}, and image observation o_i, and produces a new heatmap for the current frame. We make use of the state-of-the-art fully-convolutional tracker FCNT [240] and refer to the reader to that work for precise implementation details, but summarize them here: TRACK crops the current image o_i to a region of interest (ROI) around the most likely object location from the previous frame (the argmax of h_{i-1}). This ROI is resized to a canonical image size (e.g., 224×224) and processed with a CNN (VGG16) to produce a convolutional feature map. The object appearance model θ_{i-1} is represented as a filter on this feature map, allowing one to compute a new heatmap with a convolu-

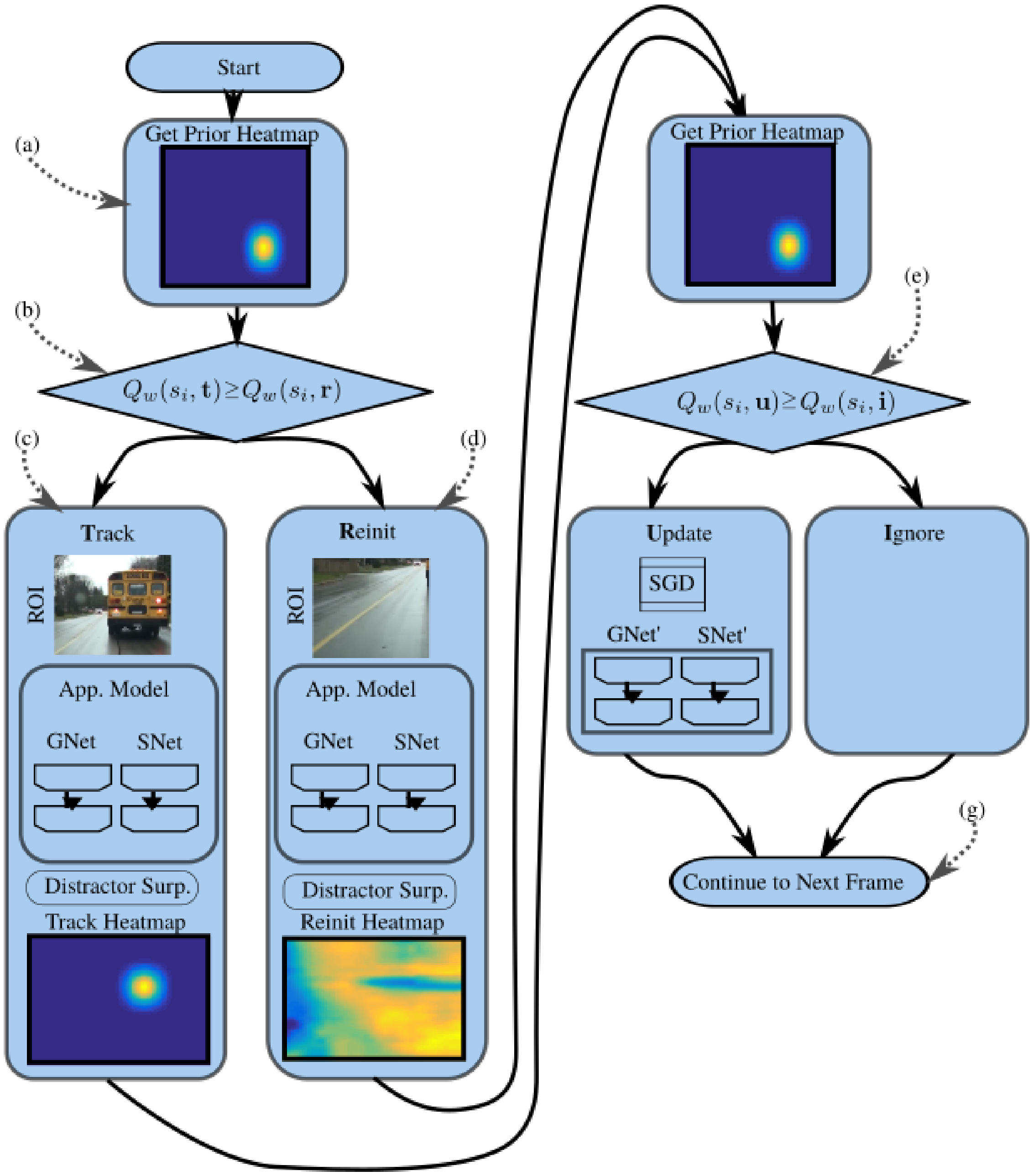


Figure 6.5: **Our tracker's (p-track's)** test time pipeline is shown above as a standard flow chart. The tracker invokes the pipeline once per frame. We explain each step in Alg. 6.

For each frame,

a Define the tracker's input state as $s_i = ((\theta_{i-1}, h_{i-1}), o_i)$. This state consists of the previous appearance model θ_{i-1}, previous localization heatmap h_{i-1} and the current observation o_i. Importantly, extract the localization heatmap predicted for the previous frame h_{i-1} from the tracker's input state s_i. We will use h_{i-1} as the argument of our decision function.

b Evaluate the Q-function on this heatmap for the track and reinit actions. Compare the value of $Q_w(s_i, \mathbf{t})$ to the value of $Q_w(s_i, \mathbf{r})$ and choose the action with the greater value.

c If we decide to track

- Extract a ROI from the current observation image, centered at the mode of the previous localization heatmap.
- Evaluate the FCNT [240] (Sec 4.2) GNet and SNet models to predict two localization heatmaps.
- Based on FCNT's distractor selection (Sec 4.2 of [240]), decide which heatmap to use. This becomes h_i.

d If we decide to reinit

- Extract a random ROI from the current observation image.
- Evaluate the FCNT [240] (Sec 4.2) GNet and SNet models to predict two location prediction heatmaps.
- Based on FCNT's distractor selection (Sec 4.2 of [240]), decide which heatmap to use. This becomes h_i.

e Now evaluate the Q-function for each of the candidate actions (update or ignore) using the old heatmap h_{i-1}. We could instead use the newly predicted heatmap h_i, but we don't. Using h_{i-1} is computationally faster because our implementation actually evaluates (b) and (e) in parallel. And, empirically using h_i does not improve performance.

f If we decide to update

- Assume the tracker correctly localized the target. Use this assumption to define new labeled training examples for the appearance model.
- Update the appearance model as described in [240] (Sec 4.3).

g Repeat on the next frame.

Algorithm 6: Above, we describe each step in our p-tracker's test time pipeline (cf. Fig. 6.5).

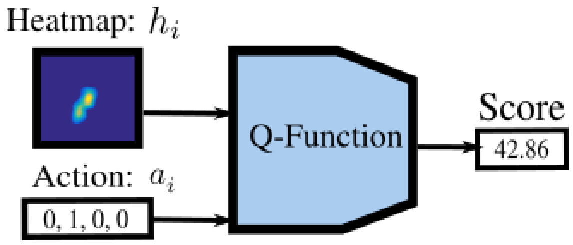


Figure 6.6: Q**-CNNs:** A Q-function predicts a score (the expected future reward) as a function of (1) the localization heatmap and (2) an action encoded using a one-hot encoding.

tion. When the tracker decides that it has lost track, the REINIT model simply processes a random ROI.

UPDATE function: We update the current filter θ using positive and negative patches extracted from the current frame i. We extract a positive patch from the maximal location in the reported heatmap h_i, and extract negative patches from adjacent regions with less than 30% overlap. We follow FCNT and represent θ as a two-layer perceptron defined over convolutional features, which we update with a small fixed number of iterations (10) of gradient descent.

Q-function CNN: Recall that our Q-functions process a tracker state $s_i = ((h_{i-1}, \theta_{i-1}), o_i)$ and a candidate action a_i, to return a scalar representing the expected future reward of taking that action (Fig. 6.6 and Eq. 6.6). We define two independent Q-functions for our two binary decisions (TRACK/REINIT and UPDATE/IGNORE). While independent functions support development & ablative analysis, the experience replay Q-learning algorithm [4, 148] is defined using a single Q function. For training, we define our single Q-function, $Q_w = Q_w^{(1)} + Q_w^{(2)}$, with $Q_w^{(1)}$ being for track vs. reinit and $Q_w^{(2)}$ being for update vs. ignore. We deliberately use addition to define our unified Q function because it makes our two Q-functions independent at test time:

$$\arg\max_{a_i^{(1)}, a_i^{(2)}} Q_w(s_i, (a_i^{(1)}, a_i^{(2)})) = \left(\arg\max_{a_i^{(1)}} Q_w^{(1)}(s_i, a_i^{(1)}), \arg\max_{a_i^{(2)}} Q_w^{(2)}(s_i, a_i^{(2)}) \right) \tag{6.7}$$

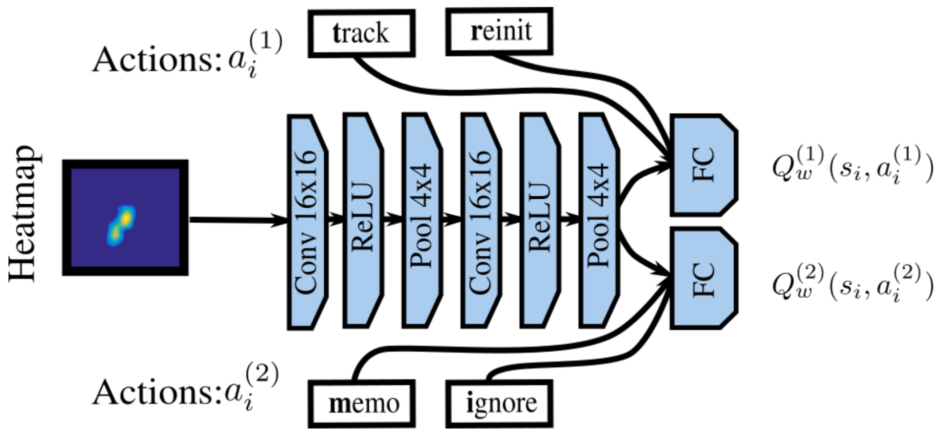


Figure 6.7: ***Q*-CNNs:** We implement our two Q-functions (for track vs. reinit and update vs. ignore) using the architecture shown above. Note that $Q_w^{(1)}$ and $Q_w^{(2)}$ share the first two convolutional layers. But, each has its own fully-connected (FC) layer.

where we have re-encoded our unified Q-function's argument a_i by decomposing it into two arguments, $a_i^{(1)}$ and $a_i^{(2)}$. In Fig. 6.7 we show a more detailed schematic of the multi-layer CNN we train to approximate $Q(s_i, a_i)$. We found it sufficed to condition on the heatmap h_{i-1} and implemented each function (of h_{i-1}) as a CNN, where the first two hidden layers are shared between the two functions. Sharing weights helps to mitigate overfitting. Each shared hidden layer consists of $16 \times 16 \times 4$ convolution followed by ReLU and 4×4 max pooling. For each decision, an independent fully-connected layer ultimately predicts the expected future reward. When training the Q-function using experience-replay, we use $\gamma = .95$, a learning rate of 1e-4, a momentum of .9 and 1e-8 weight decay.

Offline heuristics: We initialize the Q-learning optimization in Eq. 6.6 with an offline heuristic (Eq. 6.3) which assumes the tracker achieves the maximum possible future reward ($\sum_{j \geq i} \gamma^{j-i}$) if it chooses the action selected by the heuristic a_i^*:

$$Q_{\text{init}}(s_i, a_i) \Leftarrow \mathbf{I}[a_i = a_i^*] \sum_{j \geq i} \gamma^{j-i} \tag{6.8}$$

In practice, we found it useful to minimize a weighted average of the true loss in Eq. 6.6 and a supervised loss $(Q_w - Q_{\text{init}})^2$, an approach related to heuristically-guided Q-learning [95, 24].

Defining a heuristic for deciding when to track vs. reinitialize is straightforward: a_i^* should indicate track whenever the peak of the reported heatmap overlaps the ground-truth object. Defining a heuristic for update vs. ignore is more subtle. Intuitively, we should update the appearance template whenever doing so improves the confidence of ground-truth object locations across that video. Specifically, let us tentatively assume that the tracker correctly localized the target at frame i. We then fine-tune the current appearance model θ_i on samples labeled using this frame. Let Δ_+ indicate the number of future frames where updating the appearance model increases confidence at correct localizations, and let Δ_- be the number of future frames where the update decreases confidence at the incorrect localizations. We set a_i^* to update when $\Delta_+ + \Delta_- > .5N$, where N is the total number of future frames. A more formal description of the update heuristic follows.

Formalizing the update heuristic Recall that our offline (train-time) heuristic decides when to set the target action a_i^* to update. Our tracker could retrain (i.e. , update) its appearance model (as described in Sec 4.3 of [240]) using its localization at frame i to get new parameters θ' for GNet and SNet. Or, it may choose not to update its appearance model θ. We let Δ_+ indicate the number of future frames where updating the appearance model increases confidence at correct localizations, let Δ_- be the number future frames where the update decreases confidence at the incorrect localizations, and let N be the total number of remaining frames. Our heuristic sets a_i^* to update if and only if (on average) updating increases confidence for correct localizations and reduces confidence for incorrect localizations: $\Delta_+ + \Delta_- > .5N$. During training, we have the reward r_j which tells us which frames the tracker got correct. So we may formally define Δ_+ and Δ_- as

$$\Delta_+ = \frac{1}{\sum_{j>i} r_j} \sum_{j>i} r_j \mathbf{I}\left[\rho_j^{\theta'} > \rho_j^{\theta}\right] \tag{6.9}$$

Online Tracking Benchmark [247]

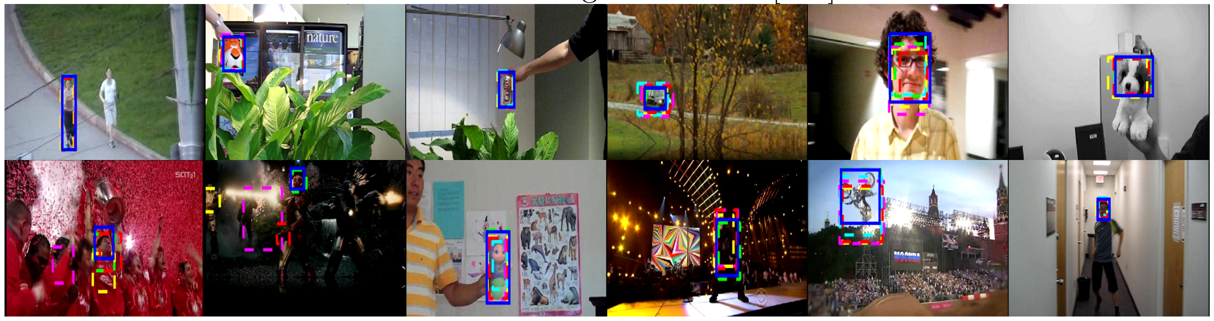


Figure 6.8: **OTB-2013:** While this dataset contains a number of challenging sequences, many are quite easy. We show results for our p-tracker plus FCNT [240], MUSTer [82], LTCT [136], TLD [98] and SPL [216]

$$\Delta_- = \frac{1}{\sum_{j>i} \bar{r}_j} \sum_{j>i} \bar{r}_j \mathbf{I} \left[\rho_j^{\theta'} < \rho_j^{\theta} \right] \quad \text{where} \quad \bar{r}_j = 1 - r_j \tag{6.10}$$

where ρ_j^θ is the confidence of the appearance model θ at the tracker's test-time localization in frame j.

6.6 Experiments

Evaluation metrics: Following established protocols for long-term tracking [98, 216], we evaluate $F_1 = \frac{2pr}{p+r}$, where precision (p) is the fraction of predicted locations that are correct and recall (r) is the fraction of ground-truth locations that are correctly predicted. Because Internet videos vary widely in difficulty, we supplement average statistics with boxplots to better visualize performance skew.

Baselines: We compare results to two classic approaches that explicitly tackle long-term tracking: TLD [98] and SPL [216]. Additionally, we also compare against short-term trackers with public code that we were able to adapt: FCNT [240], MUSTer [82], and LTCT [136].

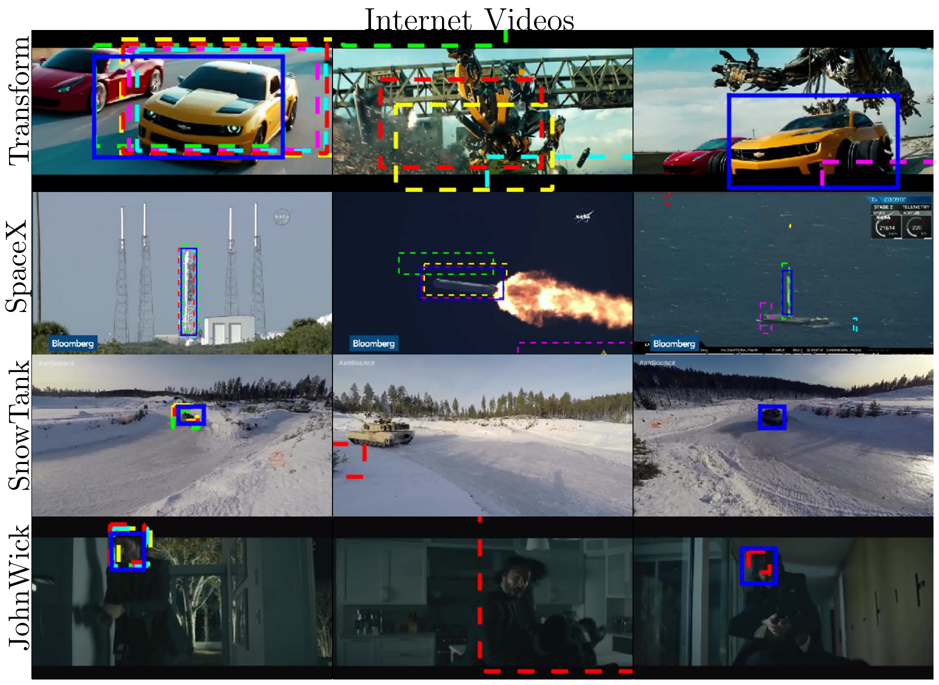


Figure 6.9: **Internet videos** contain new challenges, such as cuts, strange and interesting behaviors, fast motion and complex illumination. We show select results for our p-tracker plus FCNT [240], MUSTer [82], LTCT [136], TLD [98] and SPL [216].

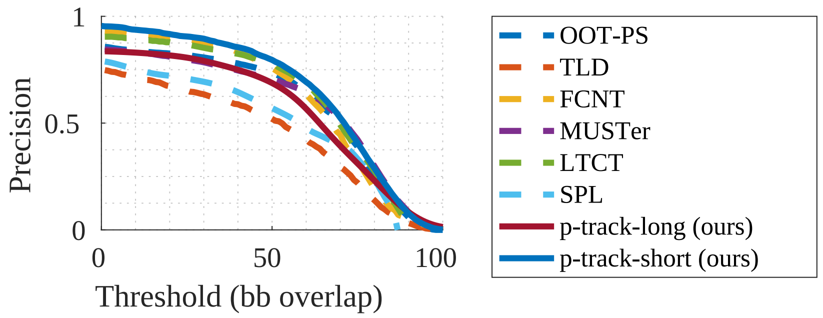


Figure 6.10: **OTB-2013 [247] results**: Our learned policy tracker (p-track) performs competitively on standard short-term tracking benchmarks. We find that a policy learned for long-term tracking (p-track-long) tends to select the ignore action more often (appropriate during occlusions, which tend be more common in long videos). Learning a policy from short-term videos significantly improves performance, producing state-of-the-art results: compare our p-track-short vs. OOT-PS [85], TLD [98], FCNT [240], MUSTer [82], LTCT [136], and SPL [216]

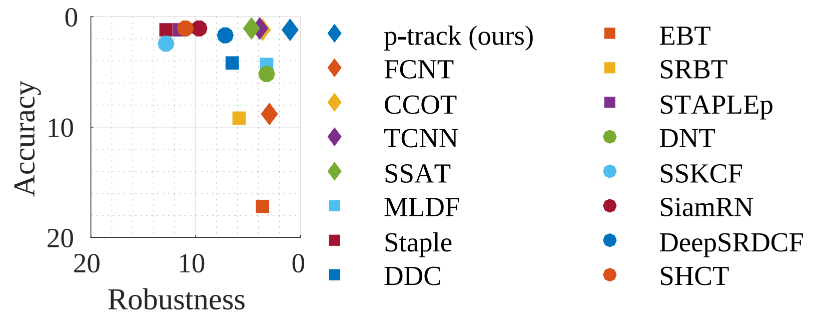


Figure 6.11: **VOT-2016 [111] results**: Our learned policy tracker (p-track-short) is as accurate as the state-of-the-art but is considerably more robust. Robustness is measured by a ranking of trackers according to the number of times they fail, while accuracy is the rank of a tracker according to its average overlap with the ground truth. Notably, p-track-short significantly outperforms FCNT [240] in terms of accuracy and robustness, even though its TRACK and UPDATE modules follow directly from that work.

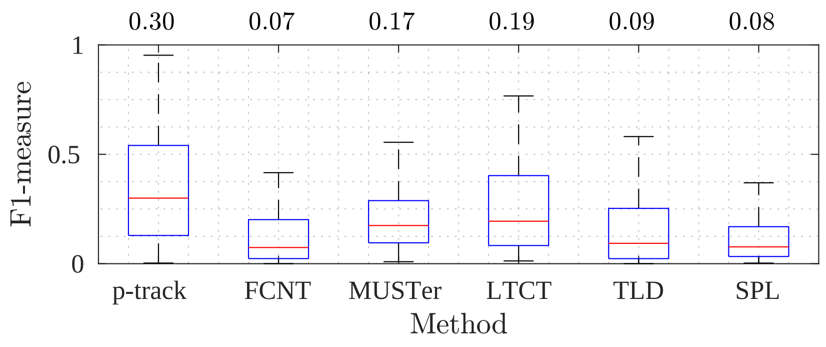


Figure 6.12: **Versus state-of-the-art**: Our learned policy (*p-track*) performs better than state-of-the-art baselines [240, 82, 136, 98, 216] on a hold-out test set. See Sec. 6.6.1 for discussion.

Notably, all these baselines use hand designed appearance update and reinitialization strategies.

6.6.1 Comparative Evaluation

Short-term benchmarks: While our focus is long-term tracking, we begin by presenting results on existing benchmarks that tend to focus on the short-term setting – the Online Tracker Benchmark (OTB-2013) [247] and Visual Object Tracking Benchmark (VOT-2016) [111]. Our **policy tracker** (*p-track-long*), trained on Internet videos, performs com-

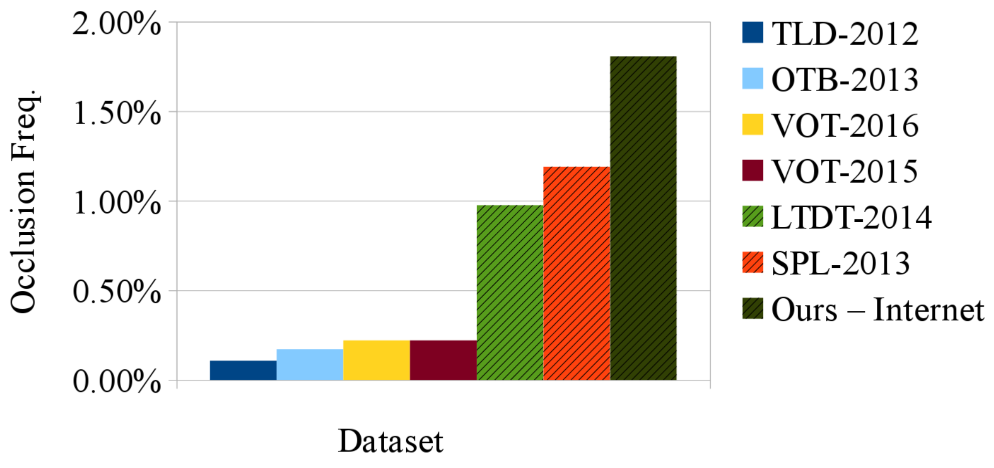


Figure 6.13: **Occlusion Frequency:** We compare how frequently targets become occluded ($\frac{\text{\# occlusions}}{\text{\# frames}}$) on various short-term (left, solid) and long-term tracking datasets (right, hatched). The long-term tracking datasets contain more frequent occlusions. To handle these occlusions, trackers must typically reinitialize. This explains why a policy learned specifically for long-term tracking outperforms a generic policy when evaluated on long-term tracking data: it can learn a policy which exploits the fact that occlusion is much more common in long-term tracking data. This fact supports the value of adapting short-term trackers to long-term tracking by learning different decision policies.

petitively (Fig. 6.10), but tends to over-predict occlusions (which rarely occur in short-term videos, as shown in Fig. 6.13). Fortunately, we can learn a dedicated policy for short-term tracking (*p-track-short*) by applying reinforcement learning (Alg. 5) on short-term training videos. For each test video in OTB-2013, we learn a policy using the 40 most dissimilar videos in VOT-2016 (and vice-versa). We define similarity between videos to be the correlation between the average (ground-truth) object image in RGB space. This ensures that, for example, we do not train on *Tiger1* when testing on *Tiger2*. Even under this controlled scenario, *p-track-short* significantly outperforms prior work on both OTB-2013 and VOT-2016 (Figs. 6.10 and 6.11).

Long-term results: To evaluate results for long-term tracking, we define a new 16-video held-out test set of long-term Internet videos that is *never used for training*. Each video has approximately 5,000 frames and contains numerous challenges: cuts, truncations, poor illumination, fast motion, and heavy occlusion. Please see the supplementary material for

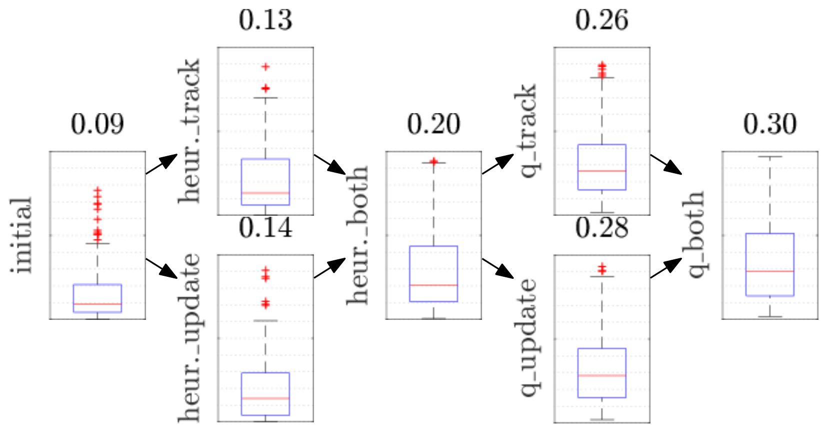


Figure 6.14: **System diagnostics:** Beginning with the *initial* policy of FCNT [240], we evolve towards our final data-driven policy objective. As shown, each component of our objective measurably improves performance on the hold-out test-data. See Sec. 6.6.2 for discussion.

visualizations. We compare our method to various baselines in Fig. 6.12. Comparisons to FCNT [240] are particularly interesting since we make use of its TRACK and UPDATE modules. While FCNT performs quite well in the short-term (Fig. 6.10), it performs poorly on long-term sequences (Fig. 6.12). However, by learning a policy for updating and reinitialization, we produce a state-of-the-art long-term tracker. We visualize the learned policy in Fig. 6.16.

Internet videos: We compare our method to long-term tracking baselines on a large suite of Internet videos in Fig. 6.12. Clearly, our final system (and indeed all methods) do worse on Internet videos than on OTB-2013 (which also uses F_1 as an evaluation criteria). Qualitatively speaking, internet videos are much more difficult than standard benchmarks (cf. Fig 6.8 and Fig. 6.9). First, many benchmarks tend to contain videos that are easy for most modern tracking approaches, implying that a method's rank is largely determined by performance on a small number of challenging videos. The easy videos focus on iconic [165, 21] views with slow motion and stable lighting conditions [124], featuring no cuts or long-term occlusions [54]. Internet videos are significantly more complex. One major reason is the presence of frequent cuts. In theory, long-term trackers should be able to re-detect the target after a cut, but there is still much room for improvement. Also, many strange things happen in the wild world of Internet videos. For example, in *Transform* the car transforms into a robot before transforming back to car form. All trackers fail to recognize the car when in robot form. In *SnowTank* the tracker must contend with many distractors (tanks of different colors and type) and widely varying viewpoint and scale. Meanwhile, *JohnWick* contains poor illumination, fast motion, and numerous distractors.

6.6.2 System Diagnostics

We now provide a diagnostic analysis of various components of our system. We begin by examining several alternative strategies for making sequential decisions (Fig. 6.14).

Online vs. offline heuristics: We begin by analyzing the online heuristic actions of our baseline tracker, FCNT. FCNT updates an appearance model when the predicted heatmap location is above a threshold, and always tracks without reinitialization. This produces a F1 score of .09. Next, we use offline heuristics to learn the best action to take. These correspond to tracking when the predicted object location is correct, and updating if the appearance model trained on the new patch produces higher scores for ground-truth locations. We train a classifier to predict these actions using the current heatmap. When this offline trained classifier is run at test time, F1 improves to .13 with the track heuristic alone, .14 with the update heuristic alone, and .20 if both are used.

Q-learning: Finally, we use Q-learning to refine our heuristics (Eq. 6.6), noticeably improving the F1 score to .30. Learning the appearance update action seems to have the most significant effect on performance, producing an F1 score of .28 by itself. During partial occlusions, the tracker learns to delicately balance between appearance update and drift while accepting a few failures to avoid the cost and risk of reinitialization. Overall, the learned

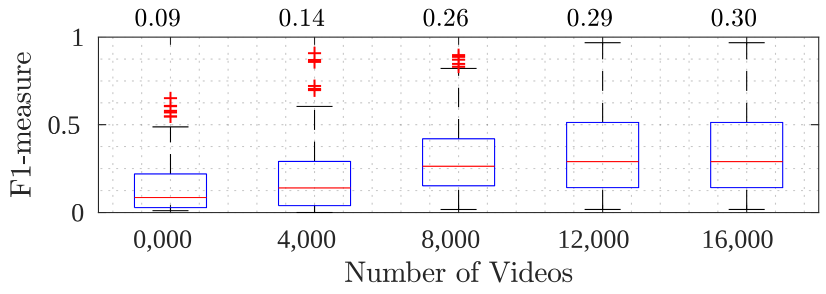


Figure 6.15: **Our p-tracker's performance increases** as it learns its policy using additional Internet videos. Above, we plot the distribution of F_1 scores on our hold-out test data, at various stages of training. At initialization, the average $F1$ score was 0.08. After seeing 16,000 videos, it achieves an average $F1$ score of 0.30. We stop training after seeing 16,000 videos because the marginal improvement on our hold-out test set becomes minuscule.

policy dramatically outperforms the default online heuristics, **tripling** the F_1 score from 9% to 30%!

Training iterations: In theory, our tracker can be interactively trained on a never-ending stream. However, in our experiments, Q-learning appeared to converge after seeing between 8,000 and 12,000 videos. Thus, we choose to stop training after seeing 16,000 videos. In Fig. 6.15, we plot performance versus training iteration.

Computation: As mentioned previously, comparatively slower trackers typically perform better [111]. On a Tesla K40 GPU, our tracker runs at approximately 10 fps. While computationally similar to [240], we add the ability to recover from tracking failures by reinitializing through detection. To do so, we learn an attention policy that efficiently balances tracking versus reinitialization. Tracking is fast because only a small region of interest (ROI) need be searched. Rather than searching over the whole image during reinitialization, we select a random ROI (which ensures that our trackers operate at a fixed frame rate). In practice, we find that target is typically found in ≈ 15 frames.

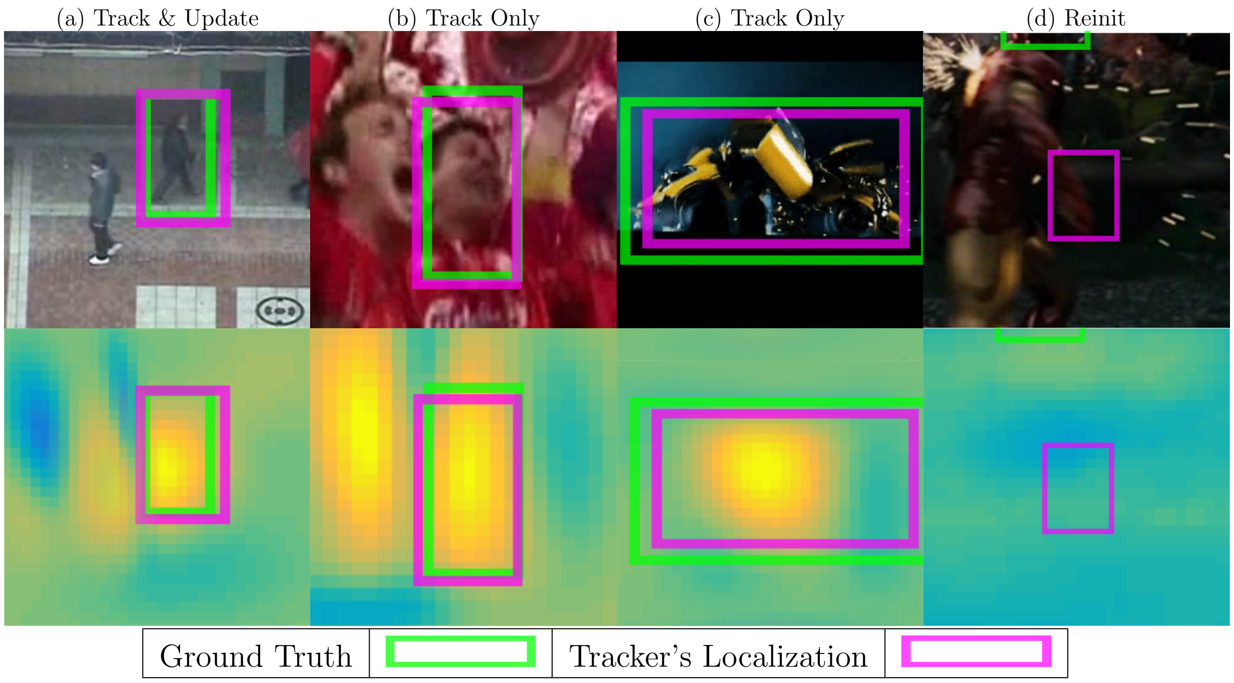


Figure 6.16: **What does p-track learn?** We show the actions taken by our tracker given four heatmaps. P-track learns to track and update appearance even in cluttered heatmaps with multiple modes (**a**). However, if the confidence of other modes becomes high, p-track learns not to update appearance to avoid drift due to distractors (**b**). If the target mode is heavily blurred, implying the target is difficult to localize (because of a transforming robot), p-track also avoids model update (**c**). Finally, the lack of mode suggests p-track will reinitialize (**d**).

Conclusions: We formulate tracking as a sequential decision-making problem, where a tracker must update its beliefs about the target, given noisy observations and a limited computational budget. While such decisions are typically made heuristically, we bring to bear tools from POMDPs and reinforcement learning to learn decision-making strategies in a data-driven way. Our framework allows trackers to learn action policies appropriate for different scenarios, including short-term and long-term tracking. One practical observation is that offline heuristics are an effective and efficient way to learn tracking policies, both by themselves and as a regularizer for Q-learning. Finally, we demonstrate that reinforcement learning can be used to leverage massive training datasets, which will likely be needed for further progress in data-driven tracking.

Chapter 7

Multiple-Object Tracking with Sensored Annotation

James Steven Supančič III, Deva Ramanan & Peter Carr

7.1 Introduction

The previous chapter proposed a method to evaluate & train single object trackers. Invariably however, scenes contain more than a single object. In this chapter, we consider tracking more than one object, a task often called *multiple object tracking* (MOT). MOT has a rich literature [104, 224, 145, 18, 67, 1, 258, 137, 20, 120]. Such works treat tracking as a spatiotemporal *grouping* task, where detections (i.e. , possible object positions) from the entire video are partitioned into disjoint subsets representing individual object tracks. However, when the number of objects K is known, tracking can be treated as a *classification* task: each detection is assigned an identity label $k \in [\emptyset, 1, \ldots, K]$. While similar, these two perspectives have distinct approaches and evaluations. The latter task, which is the focus of this chapter, is often denoted as *identity-aware* multiple object tracking (IAMOT) [257, 259, 50].

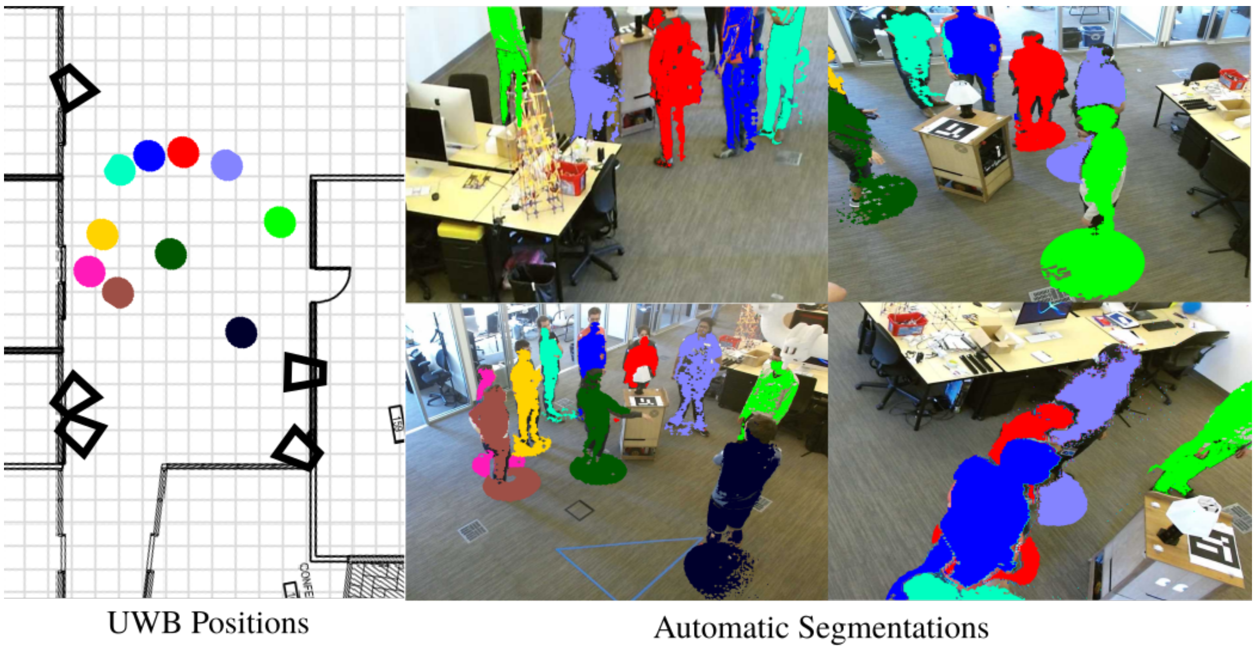


Figure 7.1: **Our new dataset for IAMOT** provides highly accurate pixel-level ground-truth annotations (we show four of our five cameras above) while being much larger than existing public datasets. We use highly accurate ultra-wideband (UWB) sensors to automatically & precisely (within 10cm) locate people in 3D space. We use these tags to label the depth points in our five calibrated depth cameras, which we then project onto the RGB image to obtain individual ground-truth segmentation masks. The scale and detail, of our new data, enables more detailed analysis of the IAMOT problem.

The MOT perspective originates from the radar community [177] and is well suited for tracking an unknown number of visually indistinguishable objects, like ants [102] or bats [22], but can also leverage appearance information when available. For efficiency, the MOT clustering objective typically only considers similarities between pairs of consecutive detections, which makes reliable long-term tracking through occlusion difficult [38]. IAMOT is the more natural setting for applications where *a priori* knowledge about appearance is available and maintaining correct identity is crucial, such as surveillance (e.g., determining who abandoned a piece of luggage) or team sports. Each tracking approach has its own metrics (Sec. 7.3), but intuitively, identity swaps are catastrophic in IAMOT (all future detections have the wrong label). In contrast, MOT only assigns an instantaneous penalty. This distinction becomes evident when tracking over long sequences.

IAMOT is a generalization of classic *single object tracking* (SOT): following an object when given its initial bounding box [238, 240]. The key difference between running K independent SOTs and one IAMOT is that the latter enforces *mutual exclusion* which prevents two tracks from claiming the same detection.

Although IAMOT is an excellent formulation for long-term multiple object tracking (e.g. , combining MOT with *re-identification* [17]), it has not been well studied. One reason is the lack of large-scale, long-duration datasets with sufficient annotation for detailed evaluation. The second is that IAMOT involves three critical components (detecting objects, identifying objects, and mutual exclusion), but the interplay between all three is not well understood (SOT and MOT only consider two of these aspects simultaneously). The final challenge is scalability. IAMOT algorithms must be efficient because they are evaluated on long-duration videos with many object identities. In this chapter, we address all three concerns.

Contribution 1 (data): We have collected a new dataset (Sec. 7.3) that is orders of magnitude larger and more challenging than existing scenarios (Table 7.2). Crucially, the annotations of true object location, identity, and segmentation mask are generated automatically using a heterogeneous sensor network of RGB cameras, Kinect cameras, and cutting edge ultra-wideband RF tags [195] (Fig. 7.1). We will release our dataset to spur further research.

Contribution 2 (analysis): We introduce new methodology (Sec. 7.4) for simulating IAMOT performance by hallucinating better detectors and appearance classifiers. We illustrate how the TINF [50] IAMOT algorithm can be converted into K independent SOTs, which makes it possible to assess the merit of improving specific stages of the pipeline. One interesting discovery is that mutual exclusion is only beneficial when detectors and appearance classifiers are sufficiently reliable. Surprisingly, running K independent SOTs outperformed two state-of-the-art IAMOT approaches (Fig. 7.11 and Table 7.1)!

Contribution 3 (algorithm): Finally, we use our large-scale annotated dataset to learn data-driven models (Sec. 7.2.2) for improving each stage of the IAMOT pipeline. We make heavy use of our extensive and detailed automatic annotations to learn geometry-aware detectors and occlusion-aware appearance models. We demonstrate that these detectors generalize to other datasets (Table 7.1). We also introduce a variant of the min-cost max-flow formulation which reasons about the compatibility of predicted occlusion masks between targets in close proximity. Motivated by the strong performance of our K independent SOTs baseline, we introduce greedy online algorithms that enforce mutual exclusion in a scalable manner and achieve competitive performance with their offline counterparts (Sec. 7.6).

We first review the core aspects of the three tracking paradigms MOT, SOT, and IAMOT; and then describe our investigations into IAMOT using our new dataset.

7.2 Related Work

7.2.1 Literature Review

Multiple-Object Tracking (MOT): is most commonly viewed as a *data association* problem [258, 137, 20, 120]. Trackers are tasked with grouping detections into coherent trajectories. In this formulation, the input is a set of spatiotemporal detections and associated appearance features [104, 224, 172]. Notably, this assumes prior appearance and detection models. These scenarios stand out in that the tracker is not provided with any initializa-

Dataset	Type	Tracker	Year	f_1	MTBF	MOTA
Ours	IAMOT	TINF [50]	2015	.42	.10	.35
Ours	IAMOT	SP [259]	2016	.27	.08	.42
Ours	SOT	FCNT [240]	2015	.60	.18	.48
Ours	IAMOT	Ours - Online	Proposed	.85	.85	.69
Ours	IAMOT	Ours - Offline	Proposed	.89	.86	.74
MOT3D	IAMOT	TINF [50]	2015	.43	.48	.32
MOT3D	IAMOT	SP [259]	2016	.33	.21	.23
MOT3D	SOT	FCNT [240]	2015	.50	.59	.33
MOT3D	IAMOT	Ours - Online	Proposed	.69	.60	.46
MOT3D	IAMOT	Ours - Offline	Proposed	.78	.68	.46
MOT3D	MOT	DBN [105]	2015	—	—	.51†
MOT3D	MOT	GPDBN [106]	2016	—	—	.50†
MOT3D	MOT	LPSFM [120]	2011	—	—	.36†
MOT3D	MOT	LP3D [145]	2016	—	—	.36†
MOT3D	MOT	KalmanSFM [48]	1993	—	—	.25†

Table 7.1: **Single vs. Multiple Object Trackers**: We compare K single object trackers (SOT) and identity-aware multi-object trackers (IAMOTs) across our new dataset and an existing public dataset with a similar 3D capture setup (MOT3D). Our focus is on f_1 accuracy, which scores the fraction of correctly-classified identities. See Sec. 7.3 for an explanation of why this is a more appropriate metric for identity-aware applications than MOTA. On both datasets, off-the-shelf SOTs outperform off-the-shelf IAMOTs. After improving detection, appearance modeling, and occlusion reasoning, our IAMOT trackers (blue) significantly outperforms prior work (red). †For completeness, we also include published MOT systems on MOT3D, but note that this is not a fair comparison as they are not manually initialized in the first frame.

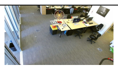
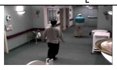
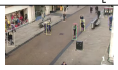
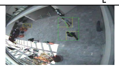
	Ours	SP-Data [258]	OxfordTC [18]	MOT16 [145]	MOT3D [145]	KITTI [67]	CAVIAR [1]
Image							
Frames	1,270,010	850,000	4,500	11,235	1,860	17,112	28,185
Annot.	1,270,010	5,000	4,500	5,316	974	8,008	28,185
Boxes	579,544	11,475	71,460	110,407	5,632	11,470	45,626
Segm.	579,544	0	0	0	0	0	0
Views	5	15	1	1	1	1	1
Imgs.	Yes	No	Yes	Yes	Yes	Yes	Yes

Table 7.2: **Public datasets for IAMOT** universally lack pixel-level annotations, and most contain only short sequences labeled at the bounding-box level. The only exception is the long nursing home sequence of [257]. However, privacy issues preclude its public release and it is only annotated at one frame per minute. In contrast, we have released our dataset without legal restrictions and have labels at 30fps.

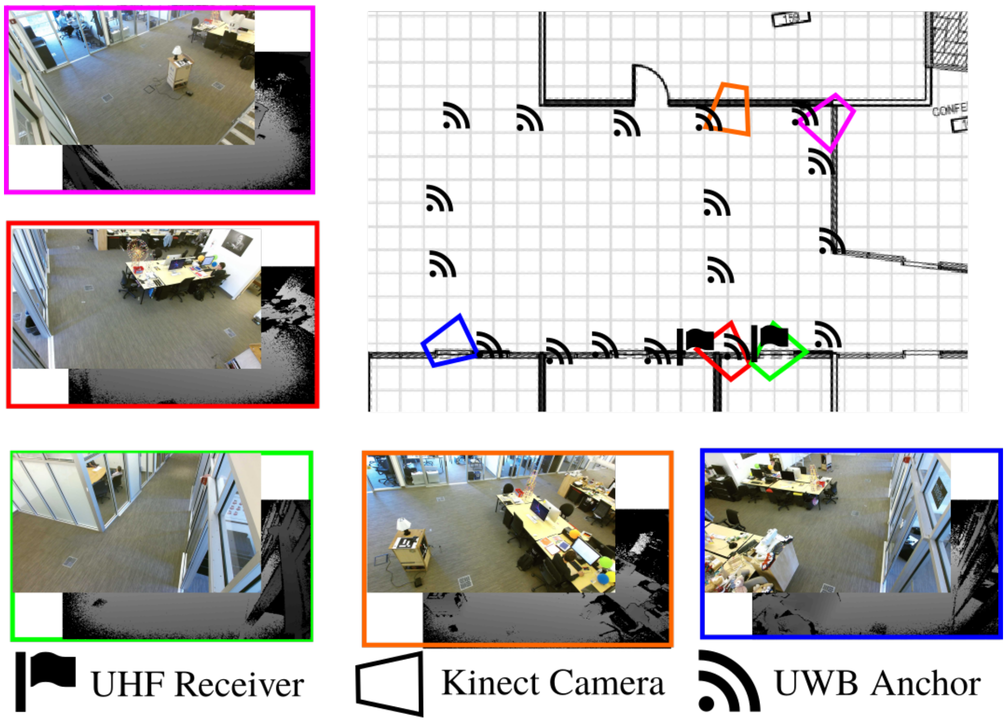


Figure 7.2: **Our sensor network** uses three types of sensors. We have five Kinect RGB+D cameras, two UHF receivers, and seventeen UWB base stations. The UWB system localizes tags to an accuracy of 10cm. We can use the UWB tag locations to label the depth point cloud, which we project onto the RGB image to obtain per-individual segmentation masks.

tion [145, 18, 67, 1]. Thus, the tracker must not only decide which detections belong to the same trajectory but also how many trajectories exist.

Single-Object Trackers (SOT): are often provided with a bounding-box in the first-frame, but are often not provided with a pre-trained detector [247, 110]. Unlike multi-object tracking, single-object trackers tend to be evaluated on shorter-length sequences where full occlusions are less common, implying reinitialization is not a key issue [247, 248]. That said, numerous SOTs explore extensions for automatic reinitialization, typically through learning object-specific detectors on the fly [82, 240, 216, 136, 98]. Such SOTs are natural candidates for "one-at-a-time" multi-object tracking.

Identity-Aware Multiple-Object Tracking (IAMOT): has received recent attention [50, 259, 258]. One reason is that many real-world MOT scenarios require correct identities – e.g., health-care [259], sports [50], and surveillance [257]. Formally, IAMOT requires labeling detections with identities, rather than merely clustering them. This problem requires some form of supervision at test time, typically provided as initial bounding boxes for learning identity-specific appearance models (similar to SOT). We explore two IAMOT systems in detail. The Target Identity-Aware Multi-Object Tracker (TINF) [50] uses global per-identity SVM appearance models and alternates between solving a network-flow based linear program [258] and updating the global appearance models. The Solution Path (SP) [258] algorithm uses quadratic constraints which are optimized through successive tightening of a relaxation. The *Marauder's map* [257] procedure tackles this problem using Non-Negative Discretization (NND).

MOT datasets: We summarize the available data in Table 7.2. Our dataset combines the features of many MOT datasets. We capture data with multiple camera views and collect a very large number of frames (which, importantly, are *automatically* annotated both densely and accurately). Additionally, our data focuses on realistic and crowded scenes. While our entire dataset requires 200GB of storage, we include some short clips in the supplementary material. Finally, we will release the raw image data and annotations, as well as precomputed detections with associated features. Virtually every tracking algorithm has some (hyper-)parameters. Rather than manually tune these on a test set, most MOT datasets provide an explicit training set [94, 249]. We similarly divide our dataset into training & test subsets. However, because our training set is much larger than most, we can train generic models (e.g., detectors and appearance classifiers) for general-purpose use.

Occlusion-aware trackers: Our dataset contains very crowded scenes and plentiful occlusion, which a tracker must reason about, either explicitly or implicitly. The idea of explicitly reasoning about occlusion patterns is not new [84, 6, 82, 245]. In particular, the work of Shu et al. [203] reasons about part visibility to avoid updating part appearances during occlusion. Prior work has also shown that training detectors to handle multiple, overlapping people improves performance [223]. Our work differs in that we use pixel-level occlusion reasoning to improve mutual-exclusion constraints.

7.2.2 Theoretical Background

The following unified overview of Single-Object and Identity-Aware Multi-Object Tracking should help describe the components studied in our analysis. In terms of Single-Object tracking, we focus on FCNT [240], as it is a state-of-the-art model based on classic tracker components (motion models, appearance models, and template updates) integrated with contemporary deep features. In terms of multi-object tracking, we focus on TINF [50], which tracks multiple objects by decomposing IAMOT into multiple independent tracking subproblems (making it particularly attractive for comparisons to single-object trackers).

Single-Object Tracker (FCNT): Loosely put, all SOTs make use of a motion model, appearance model, and template update strategy. FCNT [240] is a state-of-the-art SOT that makes use of a standard Brownian motion model and instead focuses on tracking with a strong appearance model based on multiscale deep features. Tracking is specifically done with a template that is implemented as a two-layer convolutional classifier, which is evaluated at all pixel positions in the next frame to return a heatmap of candidate object positions. FCNT itself does not include any mechanism for detecting occlusions, nor for automatically reinitializing after occlusion or tracking failures. We extend FCNT to handle both. First, we threshold the category level confidence to determine if the object has become occluded or has left the frame. Second, when the object is not found, we search globally for possible re-initializations. As our empirical results show, these simple modifications render FCNT a formidable technique for long-term IAMOT.

Solution Path (SP) tracker: Theoretically, tracking all the target identities through a single optimization should outperform running FCNT once per identity. In one formulation of IAMOT, the SP [259] tracker tries to find a binary matrix F assigning detections (rows) to identities (columns) which minimizes a quadratic loss function. Its quadratic loss function considers both appearance costs L and spatial affinity costs C. We refer to reader to Yu et al. [259] for details but will briefly review the SP objective to demonstrate that it can be trivially converted into K independent SOT problems as follows:

$$\min_{\mathbf{F}} \quad Tr(\mathbf{F}^T(\mathbf{L}+\mathbf{C})\mathbf{F}) \tag{7.1}$$

$$\text{s.t.} \quad \forall (d,k) \in \mathcal{Y}, \mathbf{F}_{dk} = 1 \tag{7.2}$$

$$\forall (i,j) \in \mathcal{T}, ||[\mathbf{F}_{il} \quad \mathbf{F}_{jl}||_0 \leq 1, \forall_l \tag{7.3}$$

$$||\mathbf{F}_d||_0 \leq 1, \forall_d \quad \text{(dropped in the k-SOT reduction)} \tag{7.4}$$

with the notation

- $F_{d,k}$: binary indicator for detection d assigned to identity k.
- L : normalized Laplacian encoding appearance costs, as defined in [259].
- C : normalized Laplacian encoding spatial costs, as defined in [259].
- $(d,k) \in \mathcal{Y}$: provided ground truth assignment of detection d to identity k.
- $\mathcal{T} = \{(i,j)|v_{i,j} > V\}$: set of detections which cannot be the same person due to velocity constraints. This is equivalent to the absence of a transition edge in TINF.

The first constraint (Eq. (7.2)) integrates the identity initialization provided in $\mathcal{Y}$. The second constraint (Eq. (**??**)) enforces a maximum velocity motion model. If we drop the mutual-exclusion constraints (Eq. (7.4) the last line in the objective), the SP reduces to k-SOT. We next describe an alternative IAMOT formulation and how to reduce it to K independent SOTs.

Identity-Aware Multiple-Object (TINF) tracker: As shown in Fig. 7.3, TINF [50] formulates IAMOT as a min-cost max-flow problem. The objective decomposes as unary and binary potentials (corresponding to observation and transition edges). Each detection d_n corresponds to K observation edges, and transition edges link pairs of subsequent detections. TINF enforces appearance consistency using a K-way classifier which is trained (from the detections initially labeled with identities) using a structured SVM. Later, these per-identity appearance models are refined using the passive aggressive algorithm [42]. The weights on each observation edge can be viewed as the negative log probability of detection d_n being object k based on the associated appearance features a_n. For each identity $k = 1 \dots K$ the cost associated with a track f^k becomes:

$$C(f^k) = \sum_{k=1}^{K} \sum_{(i,j) \in E} c_{i,j}^k f_{i,j}^k, \qquad s.t. \qquad \sum_{k=1}^{K} f_{i,j}^k \leq 1 \tag{7.5}$$

where $c_{i,j}^k$ is the cost of an edge and $f_{i,j}^k$ indicates the flow through that edge.

The linear inequality constraint from Eq. 7.5 enforces mutual exclusion: no more than one track can claim a single detection, or in other terms, two people cannot exist at the same location at the same time. While target co-locating might be possible for 2D image locations, this is less permissible when reasoning in 3D positions (as we do). Crucially, without this constraint, finding the optimal tracks according to Eq. 7.5 reduces to solving K shortest path problems, equivalent to solving the tracking problems independently. Akin to the original paper [50], we iteratively solve the constrained problem using Lagrange multipliers. Essentially, we run the trackers independently but whenever they claim the same detection we add a penalty ($\lambda_{i,j}$) for all but the most confident tracker claiming that detection (and repeat).

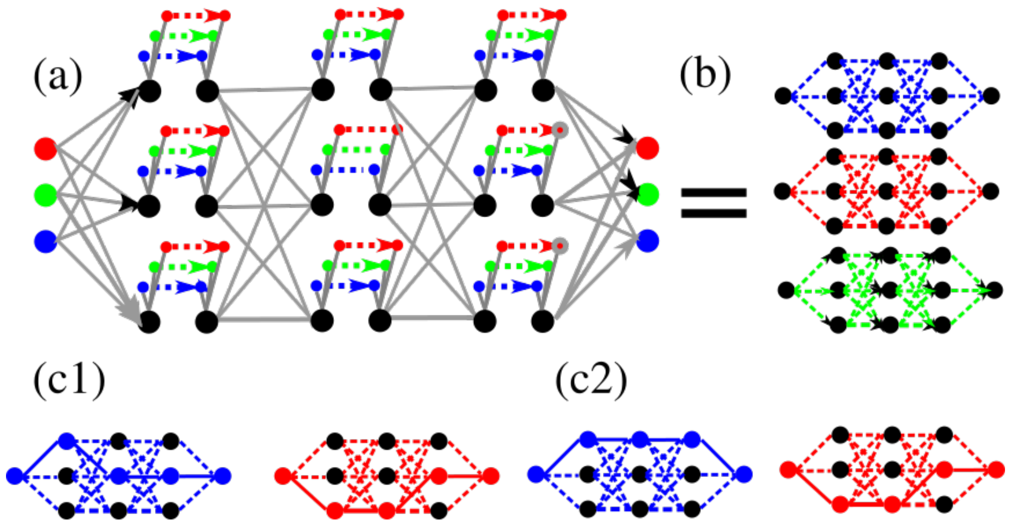


Figure 7.3: **TINF:** In each iteration, TINF (1) runs k-single-object trackers independently, (2) adjusts the costs in the single-target tracking objectives to enforce interaction constraints and (3) updates per-identity appearance models. The first two operations constitute a flow problem which could be reduced to a linear program. However, the appearance model updates induce higher-order interactions which motivates the coordinate descent algorithm described here. (a) shows the original TINF min-cost max-flow network. We replace the flow network (b) with k-independent shortest path computations. If we run the trackers independently, (c1) they might both claim the same detections. The TINF algorithm increases the costs for nodes when there's a conflict. After re-running the detector with the new cost graph, (c2) the conflicted interaction is resolved.

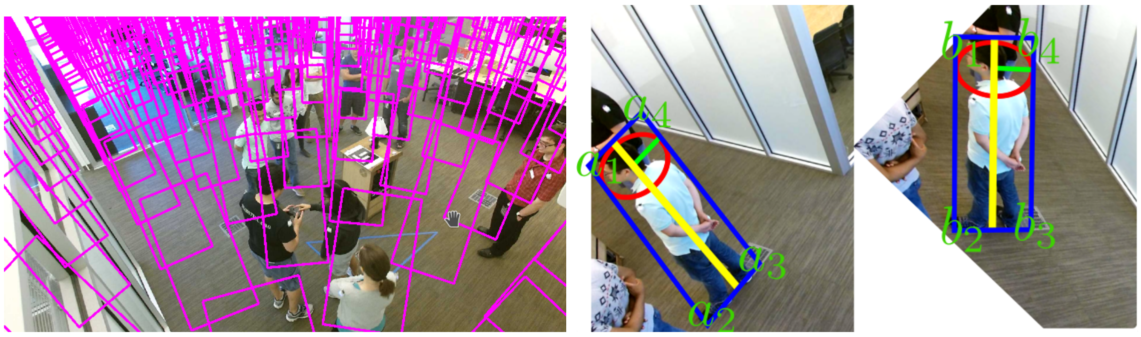


Figure 7.4: **Det. windows** for downward looking cameras are not translation invariant. Rather, we found rectifying windows before feature computation crucial. For each 3D head detection, we first compute the rotated rectangle most tightly fitting the detection's 3d bounding cylinder. Then, we fit an affine transform from each corner of the bounding rectangle to their corresponding coordinates in the rectified image. Using this affine transform, we warp the image before computing features.

7.3 Dataset, Task, and Evaluation

We now describe our new dataset (Contribution 1) and review the evaluation criteria we used for conducting our detailed analysis of IAMOT vs. SOT methods. Importantly, we make use of a sensored environment that provides automatic high-quality ground-truth annotations (Fig. 7.2).

Scenarios: Our dataset captures several realistic scenarios: (1) a crowd gathered to listen to a presentation, (2) a demonstration of an interactive robot, (3) a simulation of people walking through a busy intersection and crossing paths, (4) socializing in small groups, and (5) a simulation of queuing for an event.

Ground-truthing: We make use of three sensor types: Kinect RGB+D, UHF [83], and UWB [69]. The UWB system uses active (battery powered) tags to achieve precise localization performance. However, passive (energy harvesting) tags are attractive for some applications, although the localization performance is often not as precise as UWB. We do not use the UHF sensor data in our analysis, but it is included in our public dataset. We register and fuse depth, UWB, and image data (using the Kuhn-Munkres algorithm to match detections) to obtain high-quality 3D annotations that are accurate up to 10cm. By combining our UWB tags with depth cameras and simple depth-based threshold heuristics, one can automatically generate segmentations.

Task: Our dataset can be used to both train and evaluate trackers across multiple modalities (RGB/Depth/RF), single/multiple cameras, and 2D/3D tracking formulations. To explore occlusion reasoning and appearance modeling in a practical setting, we focus on the task of identity-aware 3D tracking from a single 2D RGB camera. We use the (known) camera parameters to reason about object locations on the world ground-plane.

Evaluation (f_1): The main criteria for IAMOT tracking is f_1-accuracy. We evaluate identity-aware K object tracks as K object *detection* problems: for each identity, we compute the number of false negatives (FN), false positives (FP) and true positives (TP) using standard 50% intersection-over-union overlap thresholds. We then sum up FN, FP, and TP across identities and use the aggregates to compute f_1 as the harmonic mean of precision and recall: $p = \frac{TP}{TP+FP}, r = \frac{TP}{TP+FN}$. In some cases, we examine the accuracy of the underlying detector, which can be measured using detector f_1. Detector f_1 is identity agnostic; it counts detections which are correct but labeled with the wrong identity as TP, instead of FP.

Evaluation (MTBF): Recent work has suggested that Mean-Time Between Failures (MTBF) provides a more interpretable measure of tracker performance [34]. While MTBF can be used in an identity-agnostic setting, we adapt it to measure identity-aware performance: if the tracker's prediction is within 0.5m of the identity-aware target, we consider it correct. MTBF is the average run length of a correct output. We then divide by the length of the identity's ground truth track to normalize to the $[0, 1]$ range and average the result across all identities.

Evaluation (MOTA): Even though the MOTA (Multiple-Object Tracking Accuracy) evaluation metric was designed for the identity-agnostic MOT regime [145], we include it in our analysis for historical reasons. MOTA averages over three types of errors: identity-*agnostic* FP, identity-*agnostic* FN, and Identity Swaps (IDSW):

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t}. \tag{7.6}$$

One crucial difference between identity-aware measures (such as f_1) and MOTA is the way identity swaps are penalized. Eq. 7.6 counts the number of identity swaps as one of many factors when measuring accuracy, while identity swaps in IAMOT (and SOT) are *catastrophic* in that all future frames are wrong. We argue that such extreme penalties are appropriate for applications that require identity maintenance (such as sports and surveillance). The same arguments apply to fragmentation (where a tracker cannot correctly link a series of shorter tracks into one long track).

7.4 Preliminary Analysis

We use our annotated dataset to explore the performance of off-the-shelf SOTs and IAMOTs, and discover the surprising result that running K independent SOTs can outperform IAMOTs; As shown in Table 7.1, on our dataset the best SOT obtains an f_1 score of .60 while the best IAMOT obtains an f_1 score of 0.42. In this section, we seek to understand why. Because SOT isn't concerned with mutual-exclusion constraints (there is only one object), research efforts have focused on developing good (deep) feature representations and learning effective instance-specific detectors online. In contrast, much of the MOT literature has focused on object interactions, and employed classic appearance features like RGB histograms. For these reasons, we suspect K independent SOTs (with no mutual exclusion) can outperform a single IAMOT because they use more effective detectors and appearance features.

To investigate this hypothesis, we note that the method of [50] is equivalent to K independent SOTs if no Lagrange multipliers are used (for details see Sec. 7.2.2). As a result, we can hold all other parameters fixed and only vary whether mutual exclusion is enforced or ignored. This allows a direct comparison between IAMOT and SOT.

To understand how the reliability of the detection and identification stages affect the final tracking results with and without mutual exclusion, we hallucinate new detection and identification outputs by interpolating between the actual results of an off-the-shelf detector and identity classifier, and the corresponding ground truth generated from the UWB tags. It is important to note that this process does not specify how to make a better detector or appearance-based identity classifier. Instead, it estimates how tracking performance should increase *if* the performance of the detector and/or identity classifier is improved. This insight also allows us to focus our efforts on the components of the tracker that should lead to the biggest overall improvement. As such, we first explain our simulation process and its key findings (see Sec. 7.6 for a full analysis) and then describe how we develop better classifiers, identifiers and exclusion models in Section 7.2.2. Finally, we interpret our results and discuss some interesting conclusions in Sec. 7.6.

7.5 Methods

We now describe our simulation experiments and several improvements to the core modules of IAMOT. To do so, we make use of data-driven methods that take advantage of our large-scale dataset for training.

7.5.1 Hallucinating Better Detections

To examine the effect of detector quality on tracker performance, we first evaluate the performance of a baseline detector [139]. Given an input set of detections $\mathbf{D}$ and the corresponding ground truth detections $\mathbf{D}^* = \{d_1^*, \dots, d_M^*\}$, we use the Kuhn-Munkres algorithm [113, 153] to determine which elements in $\mathbf{D}$ are true positives and false positives, and which elements in $\mathbf{D}^*$ are false negatives (missed detections)

$$\mathbf{D} = \mathbf{D}_{\mathsf{TP}} \cup \mathbf{D}_{\mathsf{FP}}, \tag{7.7}$$

$$\mathbf{D}^* = \mathbf{D}_{\mathsf{TP}} \cup \mathbf{D}_{\mathsf{FN}}. \tag{7.8}$$

We hallucinate a better set of detections $\hat{\mathbf{D}}$ by interpolating between $\mathbf{D}$ and $\mathbf{D}^*$. We interpolate between sets by sampling elements for each set. Specifically, we start with the set of true positives $\mathbf{D}_{\mathsf{TP}}$, and for a specific interpolation value $\gamma \in [0, 1]$ add a portion of false positives and a portion of the missed detections (false negatives),

$$\hat{\mathbf{D}} = \mathbf{D}_{\mathsf{TP}} \cup (1-\gamma)\mathbf{D}_{\mathsf{FP}} \cup \gamma\mathbf{D}_{\mathsf{FN}}. \tag{7.9}$$

Effectively, we are producing a better set of detections by suppressing false positives and inserting missed detections (false negatives). But, we must carefully consider how we choose detections to suppress and insert. Randomly sampling results in an unrealistic set of hallucinated detections. In reality, detector errors occur in clusters: occurrences of occlusions and confusing background clutter are rarely instantaneous. In practice, we sample false positives and false negatives based on confidence scores (i.e. , discarding the least confident false positives, and adding the most confident false negatives). We plot the performance of IAMOT versus SOT as a function of the hallucinated detector's performance and present the results in Fig. 7.5.

7.5.2 Hallucinating Better Identifications

For each detection d_n there is an associated vector $\mathbf{I}_n$ encoding the probabilities for each of the K identities and a background class (for false detections). Following [50] we derive the baseline probabilities using color histograms. Similarly, the ground truth identity of each detection can be encoded as a "one-hot" vector $\mathbf{I}^*_n$. Like detections, we hallucinate new identification results using linear interpolation

$$\hat{\mathbf{I}}_n = (1-\gamma)\mathbf{I}_n \cup \gamma\mathbf{I}^*_n. \tag{7.10}$$

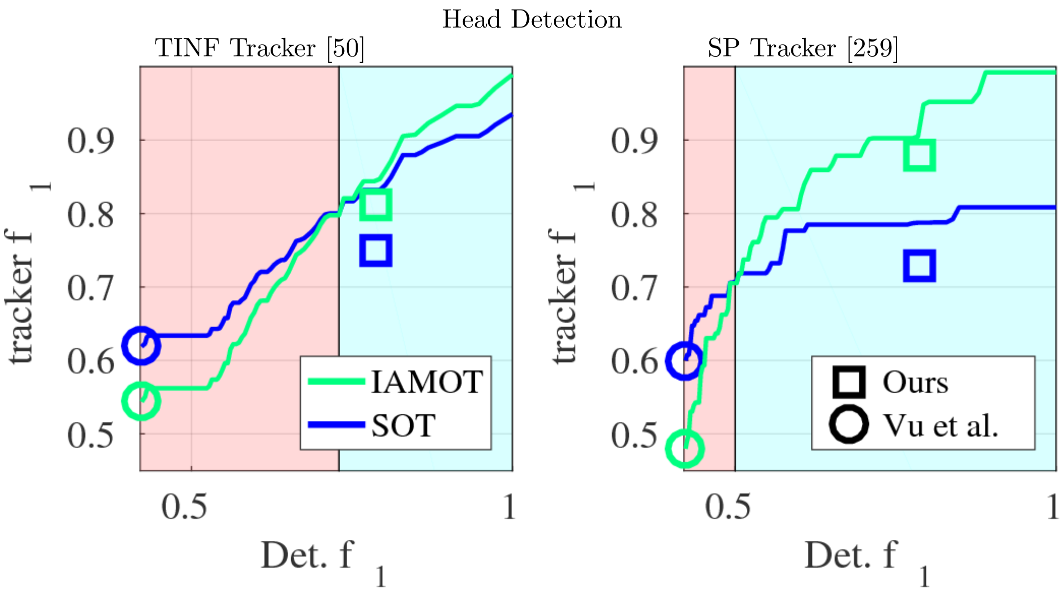


Figure 7.5: **Increasing detection** performance increases tracker performance. The marked points correspond to four actual systems: either a SOT or IAMOT using either the state-of-the-art CNN detector [238] or our new head detector. On the left, the IAMOT is TINF [50]. On the right, we use SP [259]. For both TINF and SP, we remove the mutual-exclusion constraints, which results in the corresponding green SOT. The lines plot tracking performance as a function of an interpolated detection set's quality. This plot shows that standard IAMOT systems outperform in the blue region while k SOT outperform in the red region.

Here, we use random sampling to identify a set of detections which should get the vector indicating the correct identities $\mathbf{I}_n^*$, instead of the probabilities $\mathbf{I}_n$ estimated from appearance features. We show the results in Fig. 7.6.

7.5.3 Extensions to TINF

Motivated by our simulation experiments, we now describe several improvements to the core modules of IAMOT.

k-Independent SOTs: In its initial iteration, the TINF tracker essentially runs multiple independent trackers with no mutual-exclusion constraints. Any SOT (e.g. , FCNT) could be used for this. Then, by updating penalties (in the form of Lagrange Multipliers) the independent trackers interact with each other. If we optimize Eq. (7.5), adjusting the Lagrange multipliers λ until convergence, the hard mutual-exclusion constraints will be satisfied **??**. Alternatively, we could reduce TINF back to a SOT by simply dropping the mutual-exclusion constraint entirely. We use this fact in order to make direct comparisons between IAMOT and K-SOT, keeping all other parameters fixed. We reimplemented TINF, making several additional changes. First, our implementation tracks people in 3D space instead of 2D scanning windows. This allows it to fuse detections from multiple calibrated cameras and enforce 3D mutual-exclusion. Additionally, as will be discussed later, we can swap in better features for detection and identification to obtain significant gains in performance. These changes make TINF comparable to K-SOT, in terms of effective appearance features. We refer to this improved implementation as TINF++.

Detector: We posit that existing head detectors are not trained on sufficiently difficult data. For crowded surveillance data, we want training examples exhibiting the partial occlusions that are frequent at test time. We build a better head detector using our large-scale annotated tracking dataset (six times larger than prior work [238]). First, we note that surveillance data commonly contains downward looking cameras with wide fields of view. From these camera perspectives, people can appear rotated by almost 45 degrees near the boundaries (Fig. 7.4). Following prior work on re-identification [14], we rectify the individual windows with affine transformations (i.e. , using the approximation that people are planar). We then train a binary classifier on deep features extracted from these rectified windows: specifically, we use state-of-the-art deep networks (ResNet) [81]. Fig. 7.7 shows our improved detector's performance at various precision-recall thresholds. The improved detector performance (Det. f_1 **.42 to .79**) translated to better tracker performance (Track. f_1 **.54 to .81**). This confirms what our hallucination experiments suggested: improving the detector will greatly increase tracking performance.

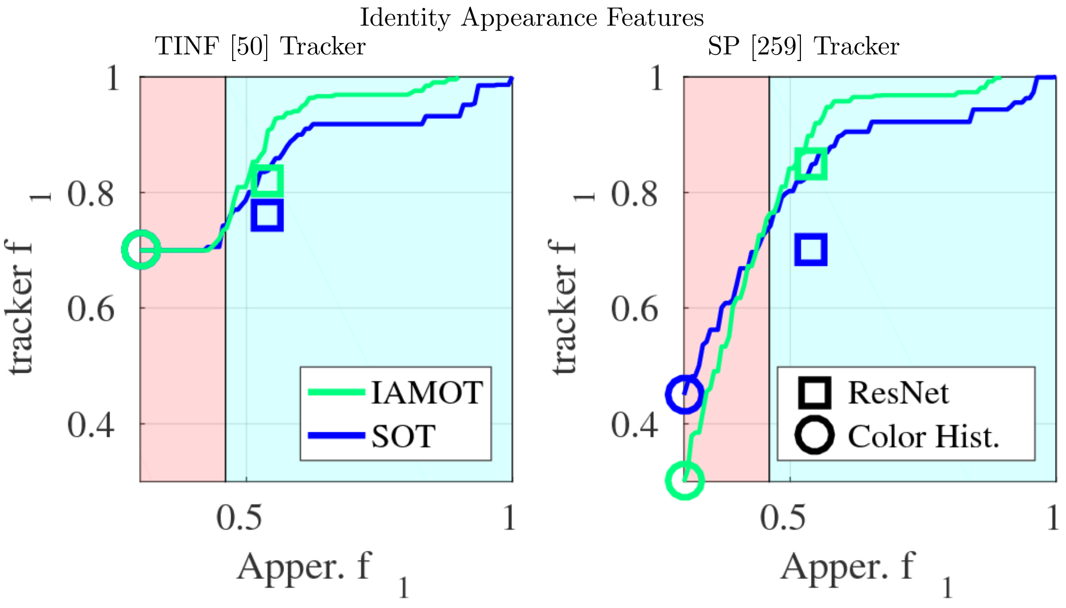


Figure 7.6: **Appearance feature quality:** Here we consider four real systems: a SOT and an IAMOT combined with color features or ResNet features. The marked points indicate the real system's performance. While, the lines correspond to simulated performance (as described in the text), for IAMOT and SOT, respectively. In the red region, SOT beats out IAMOT. But, IAMOT outperforms SOT in the blue region.

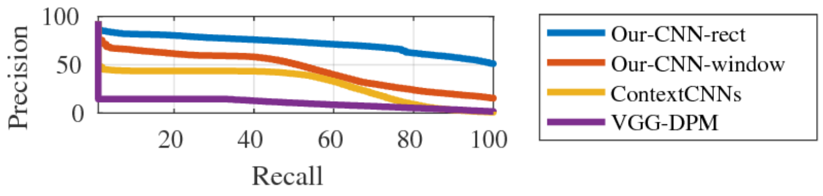


Figure 7.7: **Our new head detector** , trained from our large-scale annotated training set, performs much better than prior art on surveillance videos. We make two improvements over state-of-the-art deep head detection [238]. First, we adopt the deep appearance features of ResNet [81], labled as `Our-CNN-window`. Second, we adopt a scanning rectified window detector [14] instead of a translation invariant object proposal architecture as `Our-CNN-rect`.

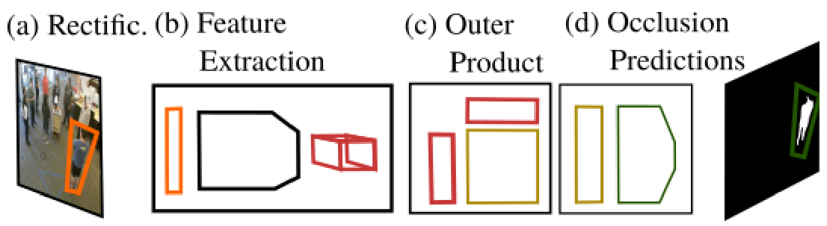


Figure 7.8: **Our deep detector** predicts a segmentation mask and overall confidence for each (a) rectified window using a CNN. (b) We extract pre-trained ResNet [81] features (`res4fx`), shown in red and written as $x[i, j, k]$ in (7.12). (c) We found an outer-product layer ($b[i, j, k]$ in (7.13), shown in gold) helpful because it allows the detector to reason about similarities in appearance between spatial locations. For example, all features on the chest should have the same appearance (i.e. the shirt color). (d) We train a simple two-layer fully-connected model to make the final predictions (shown in green and written as a hidden layer $c[u]$ in (7.14) and final predictor $d[p]$ in (7.15)).

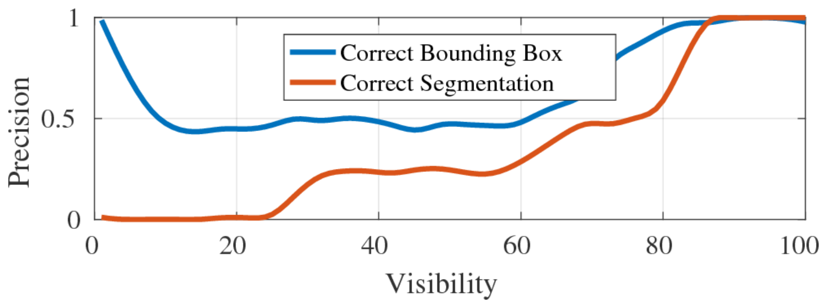


Figure 7.9: **Occlusion impacts detection**, as shown above. We define a detection as correct if it sufficiently overlaps a bounding-box (blue) or if its segmentation mask sufficiently overlaps the ground-truth segmentation mask (red). Note, that when considering bounding-boxes, precision goes to one when the person is totally visible or totally occluded by another person, in which case we're likely detecting the occluder.

Appearance features: Instead of using RGB histograms, we characterize appearance using deep ResNet features [81]. In Fig. 7.6, moving from color histogram features to deep appearance features demonstrates a clear boost in performance for all tracking algorithms, evaluation metrics, and datasets. Prior work has shown improvement by training MOT specific deep appearance features [250]. However, our results show that off-the-shelf deep features readily adapt to IAMOT. This modification improves the f_1 score of the $K+1$ way identity classifier from **.30 to .54** which increased the tracking f_1 score from **.69 to .81**.

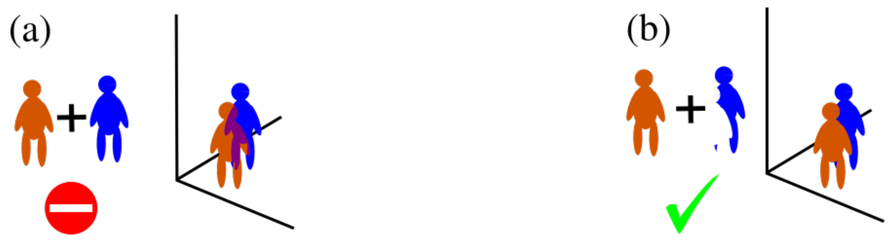


Figure 7.10: **Segmentation masks** can improve mutual-exclusion constraints. We propose extended mutual-exclusion constraints which enforce this rule: *two detections can overlap only if one is occluded*. Therefore, two trackers cannot claim two overlapping fully-visible detections, as shown in (a). But in (b), the blue mask was predicted as partially occluded. In this case, the blue and orange detections can be in close proximity and still both be correct.

Pixel-level segmentation: We augment our detection framework to also predict a segmentation mask, as shown in Fig. 7.8. Inspired by [132], our segmentation network uses an outer-product layer to reason about correlations between spatial locations. For example, the color of a person's left-arm skin tends to be similar to their right-arm skin. In Fig. 7.9 we evaluate our segmentation network on our test set's pixel-level ground truth — segmentation remains significantly harder than bounding-box prediction. We provide details for our training & evaluation procedures in Sec. 7.5.4. But, first we describe how we use segmentation to improve tracking.

Occlusion-Aware Min-Cost Flows (OAMCF): As our preliminary results suggested, our better detector improves tracking performance. But, we can also exploit the predicted segmentation masks to improve mutual-exclusion reasoning by defining a min-cost max-flow algorithm, much like TINF (Fig. 7.3), but with different mutual-exclusion constraints. Broadly speaking, two detections can only coexist if the intersection-over-union of the two masks is less than 15%. Our OAMCF algorithm outperforms our improved TINF++ (Table 7.1), proving the value of incorporating segmentation mask compatibility into mutual exclusion.

Formalizing mutual exclusion: Recall Eq. 7.5, where the mutual exclusion constraint limits the flow through any detection edge to be at most one: In practice, because many detections can overlap, TINF enforces the linear inequality constraint across *sets* of overlapping detections. This allows the MCF problem to naturally enforce non-maximal suppression during network-flow optimization. We formally write this as follows: let $M = \{m_k\}$ be the set of overlapping detection pairs, which can be represented by the incompatible edges $\{(i, j) \in m\}$:

$$C(f^k) = \sum_{k=1}^{K} \sum_{(i,j)\in E} c_{i,j}^k f_{i,j}^k \qquad s.t. \quad \forall_{m\in M} \left(\sum_{(i,j)\in m} \sum_{k=1}^{K} f_{i,j}^k \right) \leq 1 \tag{7.11}$$

The off-the-shelf implementation of TINF uses a 50% bounding-box (intersection-over-union) overlap to define incompatible detection pairs in M. Instead, we make use of our predicted segmentation masks to define a much more precise notion of overlap: a pair of detections are incompatible if their segmentation masks overlap by more than 15%. Fig. 7.10 illustrates how we use segmentation masks (as opposed to bounding-box overlap) to prevent two targets from claiming the same pixels. The optimization algorithm used by the TINF tracker [50] *actually* solves this modified objective. Further, we can even modify it for online tracking.

Online optimization: K-SOT solutions are often trivially scalable, being linear in the number of frames and linear in the number of objects, as well as being online algorithms (so that they do not require storing the entire video). Traditional IAMOT solutions are typically offline and require access to an entire video (or at least the detections from all frames) at once, making them cumbersome to apply to long-duration video. Motivated by the strong performance of K-SOT, we explore a simple online variant of TINF that optimizes Lagrange multipliers one frame at a time, holding all Lagrange multipliers and flow estimates from previous frames fixed. In practice, this is equivalent to running the Kuhn-Munkres algorithm at each frame to extend the previous tracks. In theory, one could generalize the approach to optimizing over a fixed memory of past M frames, but we found a single frame performed quite well.

7.5.4 Implementing Deep Detection and Occlusion Prediction

Using our large-scale automatically-annotated dataset, we train a CNN to predict if a given rectified image patch contains a person (as shown in Fig. 7.8). If the patch does contain a person, our model attempts to predict the person's segmentation mask. For both tasks, we begin with the pretrained ResNet [81] features from layer `res4fx`. Given that we rectify patches to a fixed size of 224×224 pixels, the resulting `res4fx` feature can be written as a multidimensional array $x \in R^{5\times5\times256}$, where the first two dimensions correspond to spatial locations and the last dimension corresponds to feature "channels". Our dataset provides a large collection of training examples x with labels $y \in \{-1, 1\}$ (indicating person or not-person), as well as binary segmentation masks $y[p] \in \{-1, 1\}$, for each pixel p in a rectified image of a person.

Detection: Given training examples of features x and binary labels y, we learn a fully-connected layer with a hinge loss objective function:

$$L_{\text{det}}(x, y) = \max(0, 1 - y \sum_{ijk} w[i, j, k]\, x[i, j, k]), \qquad w, x \in R^{5\times5\times256} \tag{7.12}$$

Given that we use off-the-shelf features, weights can be learned with a standard linear SVM package (we use liblinear).

Segmentation: Our segmentation model is similar, but predicts a *heatmap* of foreground confidences instead of a scalar detection confidence. We write $d[p]$ for the confidence that pixel p should be foreground. To capture second-order appearance statistics across different spatial locations, we make use of an "outer-product" layer inspired by [132]. These might capture the fact that the left and right shoulder have similar appearances. Specifically, for each feature channel k and image location (i, j), we compute a weighted sum of correlations with other locations (l, m):

$$b[i,j,k] = \sum_{l,m} w_b[i,j,l,m,k]\, x[i,j,k]\, x[l,m,k], \qquad b \in R^{5\times5\times256} \tag{7.13}$$

To produce our final heatmap, we process our outer-product layer with 2 additional fully-connected layers with rectified linear units:

$$c[u] = \max(0, \sum_{ijk} w_c[u,i,j,k]\, b[i,j,k]), \qquad c \in R^{1024} \tag{7.14}$$

$$d[p] = \sum_{u} w_d[p,u]\, C[u], \qquad d \in R^{80\times30} \tag{7.15}$$

where $c[u]$ contains our hidden layer activations and $d[p]$ contains our final foreground confidences for each pixel p. We produce predictions for pixels on an 80x30 output grid. We again use a simple hinge loss to train this model:

$$L_{\text{seg}}(x, y[p]) = \max(0, 1 - y[p]d[p]) \tag{7.16}$$

where we resize the segmentation masks $y[p]$ to be the same size as our heatmap $d[p]$. We train the parameters of our segmentation model (w_b, w_c, w_d) using MatConvNet, restricting our train set to patches x containing people (with an associated segmentation mask $y[p]$).

7.6 Analysis

We now examine some of the interesting empirical results that motivated our large-scale analysis. Recall that Table 7.1 revealed the surprising result that multiple instances of single-object tracking (specifically, FCNT [240]) outperformed state-of-the-art IAMOT [259, 50]. But, after improving our detection and appearance models, we see the "natural order" restored (Fig. 7.3). We further analyze the state-of-affairs here.

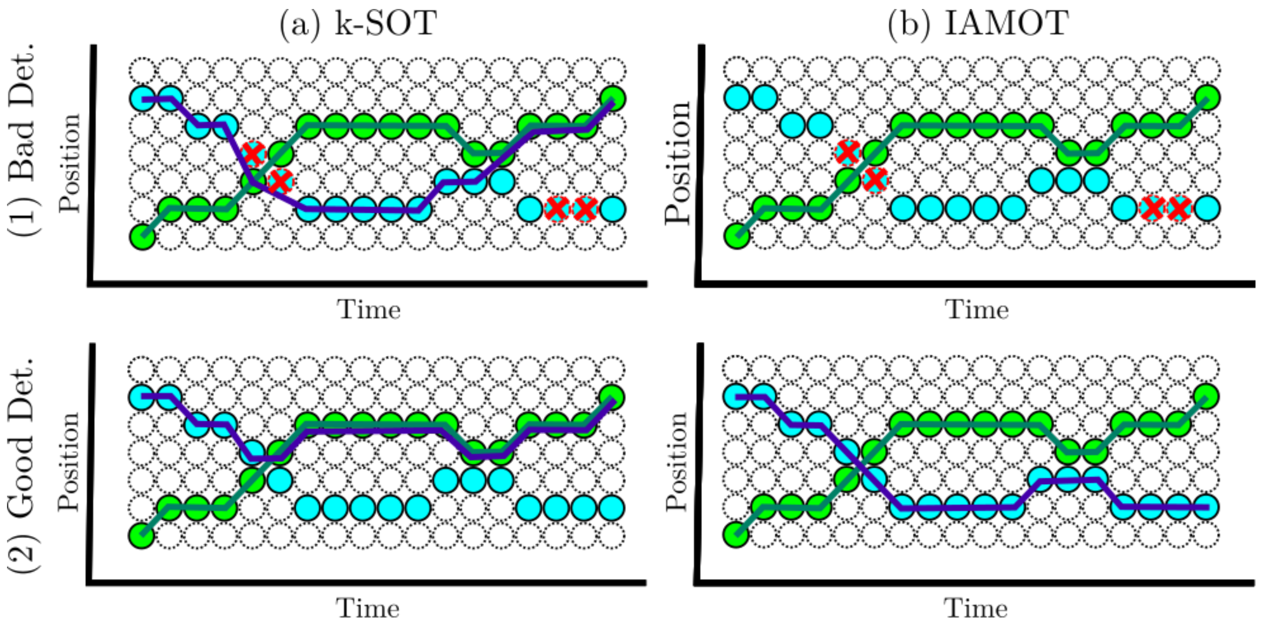


Figure 7.11: **Mutual exclusion** can be harmful! With bad detections (top), enforcing exclusion (IAMOT) finds only a single track (1b) — it misses two detections at the cross-over point (indicated by the red Xs), so only one track could pass through. Instead, allowing for overlapping detections (k-SOT) may outperform (1a). However, with good detections, IAMOT (2b) beats k-SOT because it never produces impossible tracks, where two target identities coexist at the same place and time (2a).

Simulated versus actual accuracy: Our analysis relies heavily on simulated detectors and appearance features. So we first verify the accuracy of our simulations; for verification, we compare results using actual components *versus* cross sections (of the simulated performance) on the same plots. Fig. 7.5 illustrates the simulated performance of IAMOT as the detector is improved and shows the actual IAMOT performance using our new (in-house) trained detector. Fig. 7.6 illustrates the improvement from using better appearance features. In this case, we use our improved detections and swap in the new deep appearance features. Recall that we can convert either SP or TINF into a SOT (Sec. 7.2.2) with trivial modifications to their objective functions. Our plots compare simulated performance of both the SOT and MOT versions of the SP and TINF trackers (holding all else constant). In all cases, the impact of the improved detector or appearance feature is fairly accurately simulated, though the simulated f_1 score is sometimes over-estimated.

Crossover of SOT vs. IAMOT: We use the aforementioned plots (Fig. 7.5 and 7.6) also to delineate the regime for which IAMOT outperforms SOT. As we can see, without accurate detection (with $f_1 \lesssim .72$) and appearance model classification (with $f_1 \lesssim .45$), independent single-object tracking outperforms global methods for multi-object tracking. This is because mutual-exclusion constraints may force inaccurate detections to be mislabeled (Fig. 7.11). Remarkably, the advent of deep features has fundamentally "changed the game". We now have access to strong detectors and appearance models that surpass the cross-over point in our simulations, (finally!) revealing the benefit of truly global tracking.

Human experiments: Because SOT can sometimes (surprisingly!) outperform IAMOT, we conclude that single-object trackers provide an important baseline when benchmarking multiple object trackers. SOTs are trivially scalable and often make use of state-of-the-art detectors and appearance features. Inspired by the annotation experiments from [236], we conduct an in-house psychophysics experiment verifying the appeal of SOT: we ask human subjects to track 15 objects moving in a video in one of two ways: (1) repeatedly view the video 15 times, tracking one object at a time (with a frame rate of 1 fps) or (2) view the video once and track all objects simultaneously (with a frame rate of 15 fps). Human subjects found "one-at-a-time" tracking significantly easier, both qualitatively and in terms of performance (Table 7.4). We argue that these results support the viability of using SOT for IAMOT tasks (at least until computers outperform humans).

Full simulation results: Finally, we itemize some general insights from our complete simulation experiments (Fig. 7.12). We begin with an off-the-shelf detector [139] and the original appearance model proposed by TINF, which is a color histogram model. Starting from these models, we plot the results of simulating performance by interpolating with the ground truth (Sec. 7.4). We investigated the impact of better detectors and identity classifiers for both the MOT3D datasets, as well as our new dataset. Additionally, for our more complex dataset, we also investigated how sensitive each of the identity-aware trackers (SP & TINF) were to more reliable detectors and identification features. Last, we conducted simulations with respect to two starting points: RGB and deep appearance features. The simulation results reveal some interesting trends.

- The performance of TINF on MOT3D is quite good. Although tracking performance improves as detector and identifier performance increases, the performance at the origin (off the shelf detector with RGB features) is high. In contrast, on our much more cluttered dataset, the initial performance is extremely low, showing a massive gap between current state-of-the-art performance and the performance obtainable with improved detection, identification, and mutual-exclusion models.

- Generally, the improvement from better identification performance is marginal. Many graphs exhibit minimal variation along the identity axis. Instead, there are significant changes along the detection axis implying there is a bigger benefit to pursuing a better detector.

- In general, improving detector performance has a much greater impact than the choice of the IAMOT algorithm. Perhaps SP and TINF suffer similarly from bad detections because they both enforce the same hard mutual-exclusion constraint.

7.7 Conclusion

In this chapter, we focused on the problem of identity-aware multiple-object tracking. We contributed a new dataset for IAMOT with very accurate, pixel-level labels automatically generated from UWB tags and depth maps. Our new data allowed us to glean new insights into the problem. One surprising finding was that ignoring mutual exclusion constraints can sometimes help performance. Inspired by our simulation analysis, we use our dataset to train new detection, appearance, and scalable mutual-exclusion models that significantly improve performance on both our data and standard benchmarks. Finally, we enable practical applications by demonstrating an online greedy variant of our approach which achieves similar performance at substantially higher frame rates (Table 7.3).

Dataset	Type	Tracker	Year	f_1	MTBF	MOTA	FPS
Ours	IAMOT	TINF [50]	2015	.42	.10	.35	6
Ours	IAMOT	SP [259]	2016	.27	.08	.42	13
Ours	SOT	FCNT [240]	2015	.60	.18	.48	7
Ours	SOT	FCNT++	Proposed	.67	.23	.51	7
Ours	IAMOT	TINF++	Proposed	.81	.76	.71	6
Ours	IAMOT	OAMCF-Offline	Proposed	.89	.86	.74	6
Ours	IAMOT	OAMCF-Online	Proposed	.85	.85	.69	81

Table 7.3: **Diagnostic analysis**: We summarize the improvements of our various modules. Starting with a single-object-tracker FCNT that achieves a f_1 of .60, we improve its accuracy to .69 by adding the ability to detect occlusions (and prevent appearance model updates) reinitialize. Adding mutual-exclusion reasoning and geometry-aware detectors then significantly improves accuracy to .81. Adding segmentation masks to facilitate mutual exclusion further boosts performance to .89. Finally, making the algorithm online marginally reduces performance (.85), but increases speed and reduces memory storage. See supplementary material for further diagnostics of individual components, as well as a similar evaluation on the MOT3D dataset.

Dataset	Type	Tracker	MOTA	f_1	MTBF
Ours	SOT	Human	.51	.98	.97
Ours	MOT	Human	.11	.46	.23
MOT3D	SOT	Human	.71	.91	.87
MOT3D	MOT	Human	.33	.43	.33

Table 7.4: **Humans** are quite good a single-target tracking but fail at multiple object tracking.

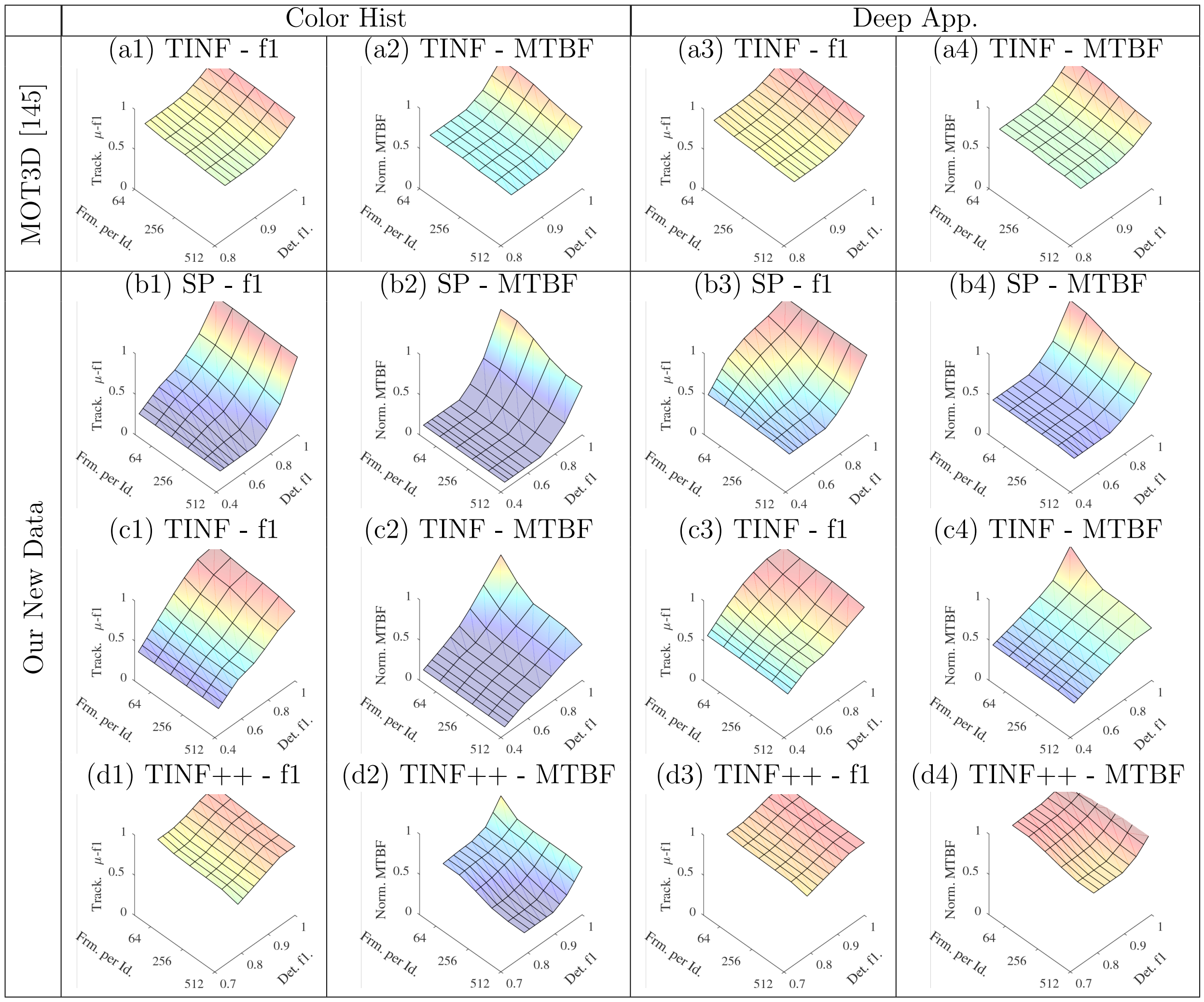


Figure 7.12: **Simulation Results**: The expected improvement in IAMOT performance as better detectors and appearance features† are used is shown for two different datasets. The origin (near-side/bottom-center) of each plot corresponds to the baseline solution (e.g. , current detector results) and the opposite corner (back-center/far-side) uses ground-truth. Contrasting with the first column, we see that our collected data poses considerably more difficulty for state-of-the-art trackers than the previous MOT3D challenge. † In these performance simulations, "Frm. per Id." serves as a proxy for identifier performance. When Frm. per Id. is one, we train our appearance model using all frames; this leads to 100% identification performance. When Frm. per Id. is 512, we use one of every 512 frames for training our identity model; this is a more realistic scenario for the initial identity models.

Chapter 8

Conclusions

James Steven Supančič III

In this thesis, we have examined the task of long-term tracking in detail. In conclusion, we posit that we must strongly consider two decisions: First, when should a tracker reinitialize? And second, when should a tracker update its appearance model? In the past, these two decisions have been under appreciated [242].

Deciding when to reinitialize: Looking at the specific problem of hand tracking, we explored reinitialization under new and more challenging scenarios (such as cluttered and egocentric scenes). In particular, our work investigated synthetic training data and semi-automatic annotation [183, 218, 184]. We proposed the first approaches for robust reinitialization in these difficult scenarios. However, much work remains to be done: better training data and better model generalization will continue to push hand tracking forward. More precise and robust hand models will ultimately enable exciting applications in augmented reality.

Deciding when to update: Long-term trackers must also decide when to adapt their appearance model at test time, for several reasons: to build instance specific models [101], track new objects [111], or to track objects whose appearances change [25]. Traditionally, trackers used hand-designed heuristics to decide when to update their appearance model during tracking [242, 216]. We propose this: trackers should instead update appearance models according to learned decision policies. However, learning such a policy is hard because annotating training videos is hard; labeling a single training video requires labeling hundreds, if not thousands, of individual image frames. As a result, training data is scarce. To train policies from data, we mitigate this lack of training data by using a reinforcement learning formulation [217] and sensored annotation [219].

Conclusion: Throughout this research, we have repeatedly observed the crucial importance of appearance models during long-term tracking. By introducing appearance models for new and more challenging scenarios, we have advanced the fields of hand tracking and identity-aware multiple-object tracking. We specifically trained *reinitialization* models for cluttered scenes (Chapter 2), egocentric viewpoints (Chapter 3), and object manipulations (Chapter 4). But labeling training videos proved difficult. So, we explored trackers which *update* and improve their appearance models without supervision. Specifically, we proposed to learn from unlabeled data using self-paced learning (Chapter 5), reinforcement learning (Chapter 6) and sensored annotation (Chapter 7). By using these techniques to obtain more precise appearance models, our future work will integrate hand-pose estimation and long-term tracking with exciting applications in virtual and augmented reality.

Bibliography

[1] CAVIAR dataset. 2004. URL `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012.

[3] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 798–805. IEEE, 2006.

[4] Sander Adam, Lucian Busoniu, and Robert Babuska. Experience replay for real-time reinforcement learning control. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(2):201–212, 2012.

[5] Aseem Agarwala, Aaron Hertzmann, David H Salesin, and Steven M Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics (ToG)*, 23(3):584–591, 2004.

[6] Irshad Ali and Matthew N Dailey. Multiple human tracking in high-density crowds. *Image and vision computing*, 30(12):966–977, 2012.

[7] S. Allin and D. Ramanan. Assessment of Post-Stroke Functioning using Machine Vision. *IAPR Machine Vision and Applications (MVA), Tokyo, Japan*, 2007.

[8] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[9] Vassilis Athitsos and Stan Sclaroff. Estimating 3d hand pose from a cluttered image. In *CVPR (2)*, pages 432–442, 2003.

[10] S. Avidan. Support vector tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(8):1064–1072, 2004.

[11] S. Avidan. Ensemble tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):261–271, 2007.

[12] B. Babenko, M.H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1619–1632, 2011.

[13] Y. Bai and M. Tang. Robust tracking via weakly supervised ranking svm. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1854–1861. IEEE, 2012.

[14] Slawomir Bak, Sofia Zaidenberg, Bernard Boulay, and François Bremond. Improving person re-identification by viewpoint cues. In *Advanced Video and Signal Based Surveillance (AVSS), 2014 11th IEEE International Conference on*, pages 175–180. IEEE, 2014.

[15] Luca Ballan, Aparna Taneja, Jürgen Gall, Luc J. Van Gool, and Marc Pollefeys. Motion capture of hands in action using discriminative salient points. In *ECCV (6)*, 2012.

[16] Loris Bazzani, Nando de Freitas, and Jo-Anne Ting. Learning attentional mechanisms for simultaneous object tracking and recognition with deep networks. In *NIPS 2010 Deep Learning and Unsupervised Feature Learning Workshop*, volume 32, 2010.

[17] Apurva Bedagkar-Gala and Shishir K Shah. A survey of approaches and trends in person re-identification. *Image and Vision Computing*, 32(4):270–286, 2014.

[18] Ben Benfold and Ian Reid. Stable multi-target tracking in real-time surveillance video. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3457–3464. IEEE, 2011.

[19] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, pages 41–48. ACM, 2009.

[20] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1806–1819, 2011.

[21] Tamara Berg and Alexander C Berg. Finding iconic images. *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[22] M. Betke, D. E. Hirsh, A. Bagchi, N. I. Hristov, N. C. Makris, and T. H. Kunz. Tracking large variable numbers of objects in clutter. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[23] Reinaldo AC Bianchi, Carlos HC Ribeiro, and Anna HR Costa. Accelerating autonomous learning by using heuristic selection of actions. *Journal of Heuristics*, 14(2): 135–168, 2008.

[24] Reinaldo AC Bianchi, Carlos HC Ribeiro, and Anna Helena Reali Costa. Heuristically accelerated reinforcement learning: Theoretical and experimental results. *European Conference on Artificial Intelligence*, pages 169–174, 2012.

[25] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2544–2550. IEEE, 2010.

[26] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *International Conference on Computer Vision*, 2009.

[27] Mourad Bouzit, Grigore Burdea, George Popescu, and Rares Boian. The Rutgers master ii-new design force-feedback glove. *Mechatronics, IEEE/ASME Transactions on*, 7(2):256–263, 2002.

[28] Matthieu Bray, Esther Koller-Meier, Pascal Müller, Luc Van Gool, and Nicol N Schraudolph. 3D hand tracking by rapid stochastic gradient descent using a skinning model. In *1st European Conference on Visual Media Production (CVMP)*, 2004.

[29] Aeron Buchanan and Andrew Fitzgibbon. Interactive feature tracking using kd trees and dynamic programming. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 626–633. IEEE, 2006.

[30] Ian M Bullock, Student Member, Joshua Z Zheng, Sara De La Rosa, Charlotte Guertler, and Aaron M Dollar. Grasp Frequency and Usage in Daily Household and Machine Shop Tasks. *Haptics, IEEE Transactions on*, 2013.

[31] Ian M. Bullock, Thomas Feix, and Aaron M. Dollar. The Yale human grasping dataset: Grasp, object, and task data in household and machine shop environments. *I. J. Robotic Res.*, 34(3):251–255, 2015.

[32] Minjie Cai, Kris M. Kitani, and Yoichi Sato. A scalable approach for understanding the visual structures of hand grasps. In *ICRA*, 2015.

[33] Massimo Camplani and Luis Salgado. Efficient spatio-temporal hole filling strategy for kinect depth maps. In *Proceedings of SPIE*, 2012.

[34] Peter Carr and Robert T Collins. Assessing tracking performance in complex scenarios using mean time between failures. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–10. IEEE, 2016.

[35] Claudio Castellini, Tatiana Tommasi, Nicoletta Noceti, Francesca Odone, and Barbara Caputo. Using object affordances to improve object recognition. *Autonomous Mental Development, IEEE Transactions on*, 2011.

[36] Chiho Choi, Ayan Sinha, Joon Hee Choi, Sujin Jang, and Karthik Ramani. A collaborative filtering approach to real-time hand pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2336–2344, 2015.

[37] Leanne Chukoskie, Joseph Snider, Michael C Mozer, Richard J Krauzlis, and Terrence J Sejnowski. Learning where to look for a hidden target. *Proceedings of the National Academy of Sciences*, 110(Supplement 2):10438–10445, 2013.

[38] Robert T. Collins. Multitarget data association with higher-order motion models. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[39] R.T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10): 1631–1643, 2005.

[40] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5), 2003.

[41] Helen Cooper. Sign language recognition using sub-units. *The Journal of Machine Learning Research*, 2012.

[42] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar): 551–585, 2006.

[43] Mark R. Cutkosky. On grasp choice, grasp models, and the design of hands for manufacturing tasks. *IEEE T. Robotics and Automation*, 5(3):269–279, 1989.

[44] T.H. Yu D. Tang and T-K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *International Conference on Computer Vision (ICCV)*, 2013.

[45] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[46] Dima Damen, Andrew P. Gee, Walterio W. Mayol-Cuevas, and Andrew Calway. Egocentric real-time workspace monitoring using an RGB-D camera. In *IROS*, 2012.

[47] Dima Damen, Teesid Leelasawassuk, Osian Haines, Andrew Calway, and Walterio W. Mayol-Cuevas. You-do, I-learn: Discovering task relevant objects and their modes of interaction from multi-user egocentric video. In *British Machine Vision Conference*, 2014.

[48] Grace Davie. Youll never walk alone. In *Pilgrimage in popular culture*, pages 201–219. Springer, 1993.

[49] Daz3D. Every-hands pose library. `http://www.daz3d.com/everyday-hands-poses-for-v4-and-m4`, 2013.

[50] Afshin Dehghan, Yicong Tian, Philip HS Torr, and Mubarak Shah. Target identity-aware network flow for online multiple target tracking. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1146–1154. IEEE, 2015.

[51] Quentin Delamarre and Olivier Faugeras. 3D articulated models and multiview tracking with physical forces. *Computer Vision and Image Understanding*, March 2001. ISSN 10773142.

[52] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009.

[53] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2012.

[54] Mario Edoardo Maresca and Alfredo Petrosino. The Matrioska tracking algorithm on LTDT2014 dataset. *CVPR workshop on LTDT*, 2014.

[55] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, October 2007. ISSN 10773142.

[56] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *CVIU*, 2007.

[57] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International journal of computer vision*, 2010.

[58] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[59] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2013.

[60] A. Fathi, A. Farhadi, and J.M. Rehg. Understanding egocentric activities. In *International Conference on Computer Vision*, 2011.

[61] Alireza Fathi, Xiaofeng Ren, and James M Rehg. Learning to recognize objects in egocentric activities. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference On*, pages 3281–3288. IEEE, 2011.

[62] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, April 2007. ISSN 10773142.

[63] T. Feix, R. Pawlik, H. Schmiedmayer, J. Romero, and D. Kragic. A comprehensive grasp taxonomy. In *RSS Workshop on Understanding the Human Hand for Advancing Robotic Manipulation*, 2009. URL `http://grasp.xief.net`.

[64] T. Feix, J. Romero, C. H. Ek, H Schmiedmayer, and D. Kragic. A Metric for Comparing the Anthropomorphic Motion Capability of Artificial Hands. *Robotics, IEEE Transactions on*, February 2013.

[65] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2010.

[66] Alistair R Fielder and Merrick J Moseley. Does stereopsis matter in humans? *Eye*, 10 (2):233–238, 1996.

[67] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, page 0278364913491297, 2013.

[68] Michael Girard and Anthony A Maciejewski. Computational Modeling for the Computer Animation of Legged Figures. *ACM SIGGRAPH Computer Graphics*, 1985.

[69] Afzal Godil, Roger Bostelman, Kamel Saidi, Will Shackleford, Geraldine Cheok, Michael Shneier, and Tsai Hong. 3D ground-truth systems for object/human recognition and tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 719–726, 2013.

[70] Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275, 2007.

[71] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proc. BMVC*, volume 1, 2006.

[72] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. *ECCV*, pages 234–247, 2008.

[73] S. Gu, Y. Zheng, and C. Tomasi. Efficient visual object tracking with online nearest neighbor classifier. *ACCV*, pages 271–282, 2011.

[74] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *European Conference on Computer Vision (ECCV)*. Springer, 2014.

[75] H. Hamer, J. Gall, R. Urtasun, and L. Van Gool. Data-driven animation of hand-object interactions. In *2011 IEEE International Conference on Automatic Face Gesture Recognition and Workshops (FG 2011)*, pages 360–367, .

[76] H. Hamer, J. Gall, T. Weise, and L. Van Gool. An object-dependent hand pose prior from sparse training data. In *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 671–678, .

[77] H. Hamer, K. Schindler, E. Koller-Meier, and L. Van Gool. Tracking a hand manipulating an object. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1475–1482, .

[78] S. Hare, A. Saffari, and P.H.S. Torr. Struck: Structured output tracking with kernels. In *International Conference on Computer Vision*, 2011.

[79] Randall P Harrison. Nonverbal communication. *Human Communication As a Field of Study: Selected Contemporary Views*, 113, 1989.

[80] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable MDPs. *arXiv:1507.06527*, 2015.

[81] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[82] Zhibin Hong, Zhe Chen, Chaohui Wang, Xue Mei, Danil Prokhorov, and Dacheng Tao. Multi-store tracker (muster): a cognitive psychology inspired approach to object tracking. *Computer Vision and Pattern Recognition (CVPR)*, pages 749–758, 2015.

[83] Ren-Ching Hua and Tzyh-Ghuang Ma. A printed dipole antenna for ultra high frequency (uhf) radio frequency identification (rfid) handheld reader. *IEEE Transactions on Antennas and Propagation*, 55(12):3742–3745, 2007.

[84] Yang Hua, Karteek Alahari, and Cordelia Schmid. Occlusion and motion reasoning for long-term tracking. In *European Conference on Computer Vision*, pages 172–187. Springer, 2014.

[85] Yang Hua, Karteek Alahari, and Cordelia Schmid. Online object tracking with proposal selection. *International Conference on Computer Vision*, pages 3092–3100, 2015.

[86] De-An Huang, Wei-Chiu Ma, Minghuang Ma, and Kris M. Kitani. How do we use our hands? discovering a diverse set of common grasps. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[87] Intel. Perceptual computing SDK, 2013.

[88] M. Isard and A. Blake. Condensation-conditional density propagation for visual tracking. *International Journal of Computer Vision*, 1998.

[89] Youngkyoon Jang, Seungtak Noh, Hyung Jin Chang, Tae-Kyun Kim, and Woontack Woo. 3d finger CAPE: clicking action and position estimation under self-occlusions in egocentric viewpoint. *IEEE Trans. Vis. Comput. Graph.*, 21(4):501–510, 2015.

[90] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3d object dataset: Putting the kinect to work. In *Consumer Depth Cameras for Computer Vision*. Springer London, 2013.

[91] A.D. Jepson, D.J. Fleet, and T.F. El-Maraghi. Robust online appearance models for visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25 (10):1296–1311, 2003.

[92] Ming Jiang, Xavier Boix, Gemma Roig, Juan Xu, Luc Van Gool, and Qi Zhao. Learning to predict sequences of human visual fixations. *IEEE transactions on neural networks and learning systems*, 27(6):1241–1252, 2016.

[93] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[94] Samira Ebrahimi Kahou, Vincent Michalski, and Roland Memisevic. RATM: Recurrent attentive tracking model. *arXiv preprint arXiv:1510.08660*, 2015.

[95] Samira Ebrahimi Kahou, Vincent Michalski, and Roland Memisevic. RATM: Recurrent attentive tracking model. *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[96] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-Backward Error: Automatic Detection of Tracking Failures. *ICPR*, 2010.

[97] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2011.

[98] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1409–1422, 2012.

[99] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *European Conference on Computer Vision (ECCV)*. 2012.

[100] S. Khamis, Taylor J., Shotton J., Keskin C., Izadi S., and A. Fitzgibbon. Learning an efficient model of hand shape variation from depth images. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[101] Sameh Khamis, Jonathan Taylor, Jamie Shotton, Cem Keskin, Shahram Izadi, and Andrew Fitzgibbon. Learning an efficient model of hand shape variation from depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2540–2548, 2015.

[102] Zia Khan, T. Balch, and F. Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(11):1805–1819, 2005.

[103] Sarang Khim, Sungjin Hong, Yoonyoung Kim, and Phill kyu Rhee. Adaptive visual tracking using the prioritized q-learning algorithm: MDP-based parameter learning approach. *Image and Vision Computing*, 32(12):1090–1101, 2014.

[104] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4696–4704, 2015.

[105] T Klinger, F Rottensteiner, and C Heipke. Probabilistic multi-person tracking using dynamic bayes networks. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2015.

[106] T Klinger, F Rottensteiner, and C Heipke. A Gaussian process based multi-person interaction model. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 271–277, 2016.

[107] W Bradley Knox and Peter Stone. Augmenting reinforcement learning with human feedback. In *ICML 2011 Workshop on New Developments in Imitation Learning (July 2011)*, volume 855, 2011.

[108] Mathias Kölsch. An appearance-based prior for hand tracking. In *ACIVS (2)*, pages 292–303, 2010.

[109] Mathias Kölsch and Matthew Turk. Hand tracking with flocks of features. In *CVPR (2)*, 2005.

[110] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking VOT2015 challenge results. *International Conference on Computer Vision*, pages 1–23, 2015.

[111] Matej Kristan, Aleš Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Čehovin, Tomas Vojir, Gustav Häger, Alan Lukežič, and Gustavo Fernandez. The visual object tracking vot2016 challenge results. Springer, Oct 2016. URL `http://www.springer.com/gp/book/9783319488806`.

[112] Paul G Kry and Dinesh K Pai. Interaction capture and synthesis. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 872–880. ACM, 2006.

[113] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[114] M.P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. *NIPS*, 2010.

[115] Takeshi Kurata, Takekazu Kato, Masakatsu Kourogi, Keechul Jung, and Ken Endo. A functionally-distributed hand tracking method for wearable visual interfaces and its applications. In *MVA*, pages 84–89, 2002.

[116] J. Kwon and K.M. Lee. Visual tracking decomposition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1269–1276. IEEE, 2010.

[117] Nikolaos Kyriazis and Antonis A. Argyros. Physically plausible 3d scene tracking: The single actor hypothesis. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[118] Nikolaos Kyriazis and Antonis A. Argyros. Scalable 3d tracking of multiple interacting objects. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[119] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling. In *ICRA*, 2014.

[120] Laura Leal-Taixé, Gerard Pons-Moll, and Bodo Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 120–127. IEEE, 2011.

[121] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. MOTchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv:1504.01942*, 2015.

[122] K.C. Lee, J. Ho, M.H. Yang, and D. Kriegman. Visual tracking and recognition using probabilistic appearance manifolds. *CVIU*, 99(3):303–331, 2005.

[123] L Adrián León, Ana C Tenorio, and Eduardo F Morales. Human interaction for effective reinforcement learning. In *European Conf. Mach. Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2013)*, 2013.

[124] Annan Li, Min Lin, Yi Wu, Ming-Hsuan Yang, and Shuicheng Yan. NUS-PRO: A new visual tracking challenge. 2015.

[125] Cheng Li and Kris M. Kitani. Pixel-Level Hand Detection in Ego-centric Videos. *Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[126] Cheng Li and Kris M. Kitani. Model recommendation with virtual probes for egocentric hand detection. In *International Conference on Computer Vision*, 2013.

[127] Cheng Li and Kris M. Kitani. Pixel-level hand detection in ego-centric videos. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[128] Cheng Li and Kris M. Kitani. Model recommendation with virtual probes for egocentric hand detection. In *International Conference on Computer Vision*, 2013.

[129] Hanxi Li, Yi Li, Fatih Porikli, et al. Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. *British Machine Vision Conference*, 1(2):3, 2014.

[130] Peiyi Li, Haibin Ling, Xi Li, and Chunyuan Liao. 3d hand pose estimation using randomized decision forest with segmentation index points. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 819–827, 2015.

[131] Yuan Li, Chang Huang, and Ramakant Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. *Computer Vision and Pattern Recognition (CVPR)*, pages 2953–2960, 2009.

[132] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

[133] B. Liu, J. Huang, L. Yang, and C. Kulikowsk. Robust tracking using local sparse appearance model and k-selection. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011.

[134] Jia Liu, Fangxiaoyu Feng, Yuzuko C. Nakamura, and Nancy S. Pollard. A taxonomy of everyday grasps in action. In *14th IEEE-RAS International Conf. on Humanoid Robots, Humanoids 2014*.

[135] Chao Ma, Jia-Bin Huang, Xiaokang Yang, and Ming-Hsuan Yang. Hierarchical convolutional features for visual tracking. *International Conference on Computer Vision*, pages 3074–3082, 2015.

[136] Chao Ma, Xiaokang Yang, Chongyang Zhang, and Ming-Hsuan Yang. Long-term correlation tracking. *Computer Vision and Pattern Recognition (CVPR)*, pages 5388–5396, 2015.

[137] Santiago Manen, Radu Timofte, Dengxin Dai, and Luc Van Gool. Leveraging single for multi-target tracking using a novel trajectory overlap affinity measure. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9. IEEE, 2016.

[138] Steve Mann, Jason Huang, Ryan Janzen, Raymond Lo, Valmiki Rampersad, Alexander Chen, and Taqveer Doha. Blind navigation with a wearable range camera and vibrotactile helmet. In *ACM International Conf. on Multimedia*, MM '11, 2011.

[139] Manuel Jesús Marín-Jiménez, Andrew Zisserman, Marcin Eichner, and Vittorio Ferrari. Detecting people looking at each other in videos. *International Journal of Computer Vision*, 106(3):282–296, 2014.

[140] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2004.

[141] Stefan Mathe, Aleksis Pirinen, and Cristian Sminchisescu. Reinforcement learning for visual object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2894–2902, 2016.

[142] L. Matthews, T. Ishikawa, and S. Baker. The template update problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2004.

[143] X. Mei and H. Ling. Robust visual tracking and vehicle classification via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(11): 2259–2272, 2011.

[144] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics based 3D skeletal hand tracking. *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games - I3D '13*, 2013.

[145] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, March 2016. URL `http://arxiv.org/abs/1603.00831`. arXiv: 1603.00831.

[146] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[147] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. *NIPS*, pages 2204–2212, 2014.

[148] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015.

[149] Zhenyao Mo and Ulrich Neumann. Real-time hand pose recognition using low-resolution depth images. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1499–1505. IEEE, 2006.

[150] Tomas Moller. A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2: 25–30, 1997.

[151] Andrew W Moore, Andy J Connolly, Chris Genovese, Alex Gray, Larry Grone, Nick Kanidoris II, Robert C Nichol, Jeff Schneider, Alex S Szalay, Istvan Szapudi, et al. Fast algorithms and efficient statistics: N-point correlation functions. In *Mining the Sky*. Springer, 2001.

[152] Marius Muja and David G. Lowe. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, November 2014. ISSN 0162-8828.

[153] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[154] Hyeonseob Nam and Bohyung Han. Learning Multi Domain Convolutional Neural Networks for Visual Tracking. *arXiv:1510.07945*, 2015.

[155] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands Deep in Deep Learning for Hand Pose Estimation. *Computer Vision Winter Workshop (CVWW)*, 2015.

[156] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Training a feedback loop for hand pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3316–3324, 2015.

[157] Markus Oberweger, Gernot Riegler, Paul Wohlhart, and Vincent Lepetit. Efficiently creating 3d training data for fine hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4957–4965, 2016.

[158] Eshed Ohn-Bar and Mohan Manubhai Trivedi. Hand Gesture Recognition in Real Time for Automotive Interfaces: A Multimodal Vision-Based Approach and Evaluations. *Intelligent Transportation Systems, IEEE Transactions on*, 2014. ISSN 1524-9050.

[159] Eshed Ohn-Bar and Mohan Manubhai Trivedi. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE transactions on intelligent transportation systems*, 15(6):2368–2377, 2014.

[160] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using kinect. In *British Machine Vision Conference (BMVC)*, 2011.

[161] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. Tracking the Articulated Motion of Two Strongly Interacting Hands. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[162] K. Okuma, A. Taleghani, N. Freitas, J.J. Little, and D.G. Lowe. A boosted particle filter: Multitarget detection and tracking. *ECCV*, pages 28–39, 2004.

[163] Lars Otten. LaTeX template for thesis and dissertation documents at UC Irvine. `https://github.com/lotten/uci-thesis-latex/`, 2012.

[164] Lucas Paletta, Gerald Fritz, and Christin Seifert. Q-learning of sequential attention for visual object recognition from informative local descriptors. *ICML*, pages 649–656, 2005.

[165] Stephen Palmer, Eleanor Rosch, and Paul Chase. Canonical perspective and the perception of objects. *Attention and performance*, 1(4), 1981.

[166] Yu Pang and Haibin Ling. Finding the Best from the Second Bests - Inhibiting Subjective Bias in Evaluation of Visual Tracking Algorithms. *International Conference on Computer Vision (ICCV)*, December 2013.

[167] D.W. Park, J. Kwon, and K.M. Lee. Robust visual tracking using autoregressive hidden markov model. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1964–1971. IEEE, 2012.

[168] Federico Pernici and Alberto Del Bimbo. Object tracking by oversampling local features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(12): 2538–2551, 2014.

[169] Tu-Hoa Pham, Abderrahmane Kheddar, Ammar Qammaz, and Antonis A. Argyros. Towards force sensing from vision: Observing hand-object interactions to infer manipulation forces. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[170] A. Pieropan, G. Salvi, K. Pauwels, and H. Kjellstrom. Audio-visual classification and detection of human manipulation actions. *In International Conference on Intelligent Robots and Systems (IROS)*, 2014.

[171] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[172] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1201–1208. IEEE, 2011.

[173] Gerard Pons-Moll, Andreas Baak, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bodo Rosenhahn. Multisensor-fusion for 3d full-body human motion capture. In *Computer Vision and Pattern Recognition (CVPR)*, pages 663–670, 2010.

[174] Prashan Premaratne, Quang Nguyen, and Malin Premaratne. *Human computer interaction using hand gestures*. Springer, 2010.

[175] PrimeSense. Nite2 middleware, 2013. Version 2.2.

[176] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[177] Donald Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, 1979.

[178] Xiaofeng Ren, Charless C Fowlkes, and Jitendra Malik. Figure/ground assignment in natural images. In *Computer Vision–ECCV 2006*, pages 614–627. Springer, 2006.

[179] Zhou Ren, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011.

[180] Grégory Rogez, Maryam Khademi, James Supancic, José María Martínez Montiel, and Deva Ramanan. 3d hand pose detection in egocentric RGB-D images. In *ECCV Workshop on Consumer Depth Camera For Computer Vision*, 2014.

[181] Gregory Rogez, Maryam Khademi, J.S Supancic, J.M.M. Montiel, and Deva Ramanan. 3d hand pose detection in egocentric rgbd images. In *ECCV Workshop on Consuper Depth Camera for Vision (CDC4V)*, pages 1–11, 2014.

[182] Grégory Rogez, James Steven Supancic III, and Deva Ramanan. First-person pose recognition using egocentric workspaces. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[183] Grégory Rogez, James Supancic, III, and Deva Ramanan. First-person pose recognition using egocentric workspaces. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[184] Grégory Rogez, James S Supancic, and Deva Ramanan. Understanding everyday hands in action from RGB-D images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3889–3897, 2015.

[185] Grgory Rogez, Maryam Khademi, James Supancic, III, J. M. M. Montiel, and Deva Ramanan. 3D hand pose detection in egocentric RGB-D images. *CDC4CV Workshop, European Conference on Computer Vision (ECCV)*, 2014.

[186] J. Romero, H. Kjellstrom, and D. Kragic. Hands in action: real-time 3D reconstruction of hands in interaction with objects. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 458–463. doi: 10.1109/ROBOT.2010.5509753.

[187] Javier Romero, Hedvig Kjellstr, and Danica Kragic. Monocular Real-Time 3D Articulated Hand Pose Estimation. *Humanoid Robots, International Conference on*, 2009.

[188] Javier Romero, Thomas Feix, H Kjellstrom, and Danica Kragic. Spatio-temporal modeling of grasping actions. In *IROS*, 2010.

[189] Javier Romero, Hedvig Kjellstrom, Carl Henrik Ek, and Danica Kragic. Non-parametric hand pose estimation with object context. *Image and Vision Computing*, 2013.

[190] D.A. Ross, J. Lim, R.S. Lin, and M.H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1), 2008.

[191] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified modeling language reference manual, the.* Pearson Higher Education, 2004.

[192] Olga Russakovsky, Jia Deng, Zhiheng Huang, Alexander C Berg, and Li Fei-Fei. Detecting avocados to zucchinis: what have we done, and where are we going? In *International Conference on Computer Vision (ICCV)*. IEEE, 2013.

[193] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[194] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach.* Prentice Hall, 2003.

[195] Zafer Sahinoglu, Sinan Gezici, and Ismail Guvenc. Ultra-wideband positioning systems. *Cambridge, New York*, 2008.

[196] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *Computer Vision and Pattern Recognition (CVPR)*, pages 723–730. IEEE, 2010.

[197] Ashutosh Saxena, Justin Driemeyer, Justin Kearns, and Andrew Y Ng. Robotic grasping of novel objects. In *NIPS*, 2006.

[198] Daniel Scharstein. A Taxonomy and Evaluation of Dense Two-Frame Stereo. *International journal of computer vision*, 2002.

[199] Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *International Conference on Computer Vision (ICCV)*. IEEE, 2003.

[200] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, Daniel Freedman, Pushmeet Kohli, Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Computer-Human Interaction, ACM Conference on*, April 2015.

[201] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. 35(12):2821–2840. ISSN 0162-8828. doi: 10.1109/TPAMI.2012.241.

[202] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 2013.

[203] Guang Shu, Afshin Dehghan, Omar Oreifej, Emily Hand, and Mubarak Shah. Part-based multiple-person tracking with partial occlusion handling. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1815–1821. IEEE, 2012.

[204] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[205] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(7):1442–1468, 2014.

[206] Andres Solis Montero, Jochen Lang, and Robert Laganiere. Scalable kernel correlation filter with sparse feature integration. *ICCV Workshops*, December 2015.

[207] Shuran Song and Jianxiong Xiao. Tracking revisited using RGBD camera: Unified benchmark and baselines. *International Conference on Computer Vision*, pages 233–240, 2013.

[208] Shuran Song and Jianxiong Xiao. Sliding Shapes for 3D Object Detection in Depth Images. *European Conference on Computer Vision (ECCV)*, 2014.

[209] Shuran Song and Jianxiong Xiao. Sliding shapes for 3D object detection in RGB-D images. In *European Conference on Computer Vision*, 2014.

[210] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. Interactive Markerless Articulated Hand Motion Tracking Using RGB and Depth Data. *International Conference on Computer Vision (ICCV)*, 2013.

[211] Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. Fast and robust hand tracking using detection-guided optimization. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[212] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2006.

[213] Bjorn Stenger, Arasanathan Thayananthan, Philip H S Torr, and Roberto Cipolla. Model-based hand tracking using a hierarchical Bayesian filter. *Pattern Analysis and Machine Intelligence, IEEE transactions on*, September 2006. ISSN 0162-8828.

[214] William C Stokoe. Sign language structure: An outline of the visual communication systems of the american deaf. *Journal of deaf studies and deaf education*, 2005.

[215] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. Cascaded hand pose regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 824–832, 2015.

[216] James Supancic and Deva Ramanan. Self-paced learning for long-term tracking. *Computer Vision and Pattern Recognition (CVPR)*, pages 2379–2386, 2013.

[217] James Supancic and Deva Ramanan. Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning. In *International Conference on Computer Vision*, 2017.

[218] James Supancic, Gregory Rogez, Yi Yang, Jamie Shotton, and Deva Ramanan. Depth-based hand pose estimation: data, methods, and challenges. In *International Conference on Computer Vision*, 2015.

[219] James Supancic, Peter Carr, and Deva Ramanan. Why can't we all just get along? effective identity-aware multi-object tracking from sensored annotation? In *International Conference on Computer Vision*, 2017.

[220] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. Latent regression forest: Structured estimation of 3D articulated hand posture. *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[221] Danhang Tang, Jonathan Taylor, Pushmeet Kohli, Cem Keskin, Tae-Kyun Kim, and Jamie Shotton. Opening the black box: Hierarchical sampling optimization for estimating human hand pose. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3325–3333, 2015.

[222] Shuai Tang, Xiaoyu Wang, Xutao Lv, Tony X Han, James Keller, Zhihai He, Marjorie Skubic, and Shihong Lao. Histogram of oriented normal vectors for object recognition with a depth sensor. In *ACCV 2012*. 2013.

[223] Siyu Tang, Mykhaylo Andriluka, and Bernt Schiele. Detection and tracking of occluded people. *International Journal of Computer Vision*, 110(1):58–69, 2014.

[224] Siyu Tang, Bjoern Andres, Miykhaylo Andriluka, and Bernt Schiele. Subgraph decomposition for multi-target tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5033–5041, 2015.

[225] Jonathan Taylor, Richard Stebbing, Varun Ramakrishna, Cem Keskin, Jamie Shotton, Shahram Izadi, Aaron Hertzmann, and Andrew Fitzgibbon. User-specific hand modeling from monocular depth sequences. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014.

[226] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, et al. Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. *ACM Transactions on Graphics (TOG)*, 35(4):143, 2016.

[227] Andrea Lockerd Thomaz, Guy Hoffman, and Cynthia Breazeal. Real-time interactive reinforcement learning for robots. In *AAAI 2005 workshop on human comprehensible machine learning*, 2005.

[228] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *Graphics, ACM Transactions on*, August 2014.

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

[230] Dimitrios Tzionas and Juergen Gall. A comparison of directional distances for hand pose estimation. In Joachim Weickert, Matthias Hein, and Bernt Schiele, editors, *Pattern Recognition*, number 8142 in Lecture Notes in Computer Science. Springer Berlin Heidelberg, January 2013.

[231] Dimitrios Tzionas, Abhilash Srikantha, Pablo Aponte, and Juergen Gall. Capturing hand motion with an RGB-D sensor, fusing a generative model with salient points. In *German Conference on Pattern Recognition (GCPR)*, Lecture Notes in Computer Science. Springer, September 2014.

[232] V. Vapnik. *The nature of statistical learning theory.* springer, 1999.

[233] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon.* Moscow, Russia, 2003.

[234] Tomas Vojir, Jana Noskova, and Jiri Matas. Robust scale-adaptive mean-shift for tracking. *Image Analysis*, pages 652–663, 2013.

[235] Carl Vondrick and Deva Ramanan. Video annotation and tracking with active learning. In *NIPS*, pages 28–36, 2011.

[236] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1):184–204, 2013.

[237] Marin Šarić. Libhand: A library for hand articulation, 2011. Version 0.9.

[238] Tuan-Hung Vu, Anton Osokin, and Ivan Laptev. Context-aware cnns for person head detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2893–2901, 2015.

[239] Chengde Wan, Angela Yao, and Luc Van Gool. Hand pose estimation from local surface normals. In *European Conference on Computer Vision*, pages 554–569. Springer, 2016.

[240] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual Tracking with Fully Convolutional Networks. *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[241] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. STCT: Sequentially training convolutional networks for visual tracking. *CVPR*, 2016.

[242] Naiyan Wang, Jianping Shi, Dit-Yan Yeung, and Jiaya Jia. Understanding and diagnosing visual tracking systems. *International Conference on Computer Vision*, pages 3101–3109, 2015.

[243] Robert Wang and Jovan Popovic. Real-time hand-tracking with a color glove. *ACM Trans on Graphics*, 2009.

[244] Aaron Wetzler, Ron Slossberg, and Ron Kimmel. Rule of thumb: Deep derotation for improved fingertip detection. In *British Machine Vision Conference (BMVC)*. BMVA Press, September 2015.

[245] Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

[246] Y. Wu and T.S. Huang. Color tracking by transductive learning. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 133–138. IEEE, 2000.

[247] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. *Computer Vision and Pattern Recognition (CVPR)*, pages 2411–2418, 2013.

[248] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[249] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. *International Conference on Computer Vision*, pages 4705–4713, 2015.

[250] Tong Xiao, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learning deep feature representations with domain guided dropout for person re-identification. *arXiv preprint arXiv:1604.07528*, 2016.

[251] Chi Xu and Li Cheng. Efficient Hand Pose Estimation from a Single Depth Image. *International Conference on Computer Vision (ICCV)*, December 2013.

[252] Chi Xu and Li Cheng. Efficient hand pose estimation from a single depth image. In *International Conference on Computer Vision*, 2013.

[253] Yi Yang and Deva Ramanan. Articulated pose estimation with flexible mixtures-of-parts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2013.

[254] Mao Ye, Xianwang Wang, Ruigang Yang, Liu Ren, and Marc Pollefeys. Accurate 3d pose estimation from a single depth image. In *International Conference on Computer Vision*, pages 731–738, 2011.

[255] Qi Ye, Shanxin Yuan, and Tae-Kyun Kim. Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation. In *European Conference on Computer Vision*, pages 346–361. Springer, 2016.

[256] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm Computing Surveys (CSUR)*, 38(4):13, 2006.

[257] Shoou-I Yu, Yi Yang, and Alexander Hauptmann. Harry potter's marauder's map: Localizing and tracking multiple persons-of-interest by nonnegative discretization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3714–3720, 2013.

[258] Shoou-I Yu, Deyu Meng, Wangmeng Zuo, and Alexander Hauptmann. The solution path algorithm for identity-aware multi-object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3871–3879, 2016.

[259] Shoou-I Yu, Yi Yang, Xuanchong Li, and Alexander G Hauptmann. Long-term identity-aware multi-person tracking for surveillance video summarization. *arXiv preprint arXiv:1604.07468*, 2016.

[260] Y. Zha, Y. Yang, and D. Bi. Graph-based transductive learning for robust visual tracking. *Pattern Recognition*, 43(1):187–196, 2010.

[261] Xiangxin Zhu, Carl Vondrick, Deva Ramanan, and Charless Fowlkes. Do we need more training data or better models for object detection?. In *British Machine Vision Conference (BMVC)*, 2012.