AN ABSTRACT OF THE THESIS OF

Alrik Firl for the degree of Master of Science in Computer Science presented on July 18, 2018.

Title: Video Analysis: Techniques for Semi-Supervised Video Object Instance Segmentation and Tracking-by-Detection in the Wild

Abstract approved: ______________________________

Fuxin Li

This thesis consists of two major components. The first part is concerned with video object instance segmentation (VOS), which is the task of assigning per-pixel labels per-frame of a video sequence to indicate foreground object instance membership, given the first frame ground truth mask. VOS has myriad applications, from video post-processing to action recognition, and is an active area of research. A novel end-to-end trainable, online algorithm utilizing a bilinear LSTM to learn long-term appearance models is presented. The bilinear LSTM is used to guide the learned CNN features, integrating temporal information and building more discriminative appearance features for specific objects during inference. The second part of this thesis examines computer vision's potential applications for performing automated ecological inference for endemic flatfish populations. Specifically, it looks at the construction of a visual tracking dataset, NHFish, consisting of underwater beam trawl videos collected along the Newport Hydrographic Line of Oregon coast benthos and the application of automated methods for video analysis of the beam trawl videos.

Video Analysis: Techniques for Semi-Supervised Video Object Instance Segmentation and Tracking-by-Detection in the Wild

by

Alrik Firl

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented July 18, 2018
Commencement June 2019

Master of Science thesis of Alrik Firl presented on July 18, 2018.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

Alrik Firl, Author

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Fuxin Li for his support and guidance, my family for their encouragement, and my fiancée Victoria for proofreading, emotional support, and being a decent human being. I would also like to thank the OSU NSF NRT program for funding support.

CONTRIBUTION OF AUTHORS

Fellow OSU NSF NRT project team members Samantha Newton and Katlyn Haven helped label data and provided invaluable interdisciplinary insights for the beam trawl video data project.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF APPENDIX FIGURES

Chapter 1: Introduction

1.1 Historical Perspective

Computer vision (CV) got its start in the 1960s as a subtask of the nascent field of artificial intelligence (AI). Although its exact origins are contested, one of the first formulations [54] of modern CV came from Lawrence Robert's 1963 thesis on extracting 3D information from 2D perspective views [113]. Work in AI predated CV, with notable advancements in the 1950s such as the coining of the phrase "Artificial Intelligence" in 1955 by John McCarthy [89] and the development of the perceptron in 1957 by Frank Rosenblatt [115]. AI and CV have come a long way since their inception, and modern approaches to e.g. image recognition have since matched or exceeded human capabilities (see [24] for an algorithm that has exceeded human performance on traffic sign recognition). Much of CV's recent advancements have been fueled by the advent of deep learning (DL). Although relatively recent, DL was largely theorized in the 1980s and 1990s, e.g. Kunihiko Fukushima's 'Neurocognitron' [37] had many of the conceptual underpinnings of modern deep convolutional networks, and Lenet-5 [74] in 1998 was a successful implementation of a multi-layer convolutional neural network (CNN) trained with backpropogation. Similarly, Long Short-Term Memory (LSTM) networks were introduced in 1997, and remain as one of the most successful recurrent neural network (RNN) architectures in the present [90]. Despite the prolific work done in the 1980s and 1990s on artificial neural networks, deep neural networks only became the preeminent paradigm starting in 2012 with AlexNet [67] and its impressive performance on ImageNet [32].

1.2 Deep Learning

The rise of DL was likely due to a confluence of necessity and hardware improvements. Datasets grew to sizes that proved difficult for contemporary approaches, and Graphical Processing Units (GPU), originally developed for computer graphics applications, became sufficiently affordable, powerful, and easily programmable to be used for DL workloads [44]. Although GPUs had been available for many years prior, they were closely tied to graphical applications, and frameworks such as OpenGL used fixed-function pipelines that made general purpose computation difficult. Compute shaders were introduced to enable more general purpose programs not directly related to graphics and/or rendering (but still operating within the graphics pipeline), and ushered in the era of general purpose GPU computing (GPGPU). Compute Unified Device Architecture (CUDA) was introduced by NVIDIA in 2006 [94] to facilitate GPGPU, and was entirely independent of graphical applications. This proved to be a critical development for DL, as DL workloads map efficiently to GPU hardware. More specifically, they have high arithmetic complexity and relatively regular memory access patterns, and thus are well suited to utilizing single instruction, multiple thread (SIMT) parallelism. Due to the large dataset sizes, massive number of trainable parameters, and intensive processing inherent to DL, the high throughput that GPUs offer [1] are necessary for efficient DL application development. All modern deep learning frameworks have CUDA and/or OpenCL backends.

Deep networks require large offline datasets to train, which necessitates the creation of labeled datasets. Depending on the task at hand, these datasets can simultaneously be invaluable for innovation and prohibitively expensive to create. We detail the creation of a visual tracking dataset in chapter 3. DL algorithm design revolves around defining networks, consisting of cascades of non-linear functions that compute representations of data. Intuitively, each successive layer in the network learns more semantically detailed representations; early (shallow) layers learn simple features, such as color features and edges, and later (deeper) layers learn higher-order information [157]. The exact feature extraction functions are defined by the network architecture, which consist of differentiable processing routines with learnable parameters, referred to as weights. Network parameters are learned at training time to minimize a loss function with respect to the desired ground truth results. Although there are different branches of DL (e.g. supervised versus unsupervised, generative versus discriminative, etc.), supervised methods have enjoyed greater successes than unsupervised, and we specifically focus on supervised disriminative approaches in this work. DL is thought to better scale to high-dimensional data than past approaches, and has relaxed local constancy and smoothness assumptions which models complicated real-world data more closely [44]. As such, DL

[1]The NVIDIA TITAN X, the current NVIDIA flagship GPU, has a reported 11 TFLOPS

is an important tool in CV (and AI in general), and will remain so for the foreseeable future.

1.3 Computer Vision

CV itself is a broad and multi-faceted field, with applications ranging from self-driving cars to robotics to image style transfer [39], and draws inspiration from diverse fields such as mathematics, electrical engineering, neuroscience, and psychology. As a discipline, the broad goal of CV is to extract information from images, whether this be 3D pose estimation from images, or semantic instance segments in video sequences. In many ways, CV can be formulated as an inverse of computer graphics: while graphics takes information about a scene and generates an image, CV takes an image and infers information about the scene it depicts. While CV is an impossibly broad field, this thesis is specifically concerned with a sub-component of CV, namely object segmentation and tracking.

Although CV and machine learning (ML) have similar roots, CV does not necessarily employ learning techniques. There are rich subfields and operations in CV that are closer to signal processing than machine learning, such as estimation or filtering. In the 1990s and first decade of the 2000s, the predominant CV approach was to have computer vision specialists hand-engineer features for specific tasks, and learn a classifier, such as a decision forest or SVM, to operate on the features. Hand-engineered features have mostly been subsumed by learned deep features, with DL methods being dominant in many CV tasks (e.g. classification and detection). However, there are still some tasks, such as visual tracking, that have seen slower adoption of learned features, or for which hand-engineered features still outperform learned ones (such as the deformable parts model [35] for pedestrian detection [73]). In all, the trend seems to be that learned features will replace hand-engineered ones, and the work presented in this thesis will focus entirely on the DL aspects of CV.

1.3.1 Applied Computer Vision

Although much CV work has been done in the theoretical domain and evaluated on fixed, artificial datasets, CV holds promise for improving efficiency and providing deeper insights in a wide variety of problem domains. Data from unconstrained environments is generally more challenging than synthetic datasets, which historically has limited the utility of CV 'in the wild'. As algorithmic accuracy has improved, applied CV has become more feasible, and tangible commercial efforts have become increasingly prominent. The work detailed in chapter 3 was done as part of an interdisciplinary team to facilitate the use of CV for fisheries science and management. We present a new and challenging visual tracking dataset of beamtrawl videos for surveying flatfish off of the Oregon coast, laying the groundwork for closer integration between CV practitioners and scienctists. Datasets are an important catalyst for CV innovation, and there is an acute need for larger, more diverse datasets across disparate tasks. Concurrently, there is a strong need for applying CV innovations to the relevant societal and environmental problems in the current day. We believe the application of CV and ML 'in the wild' will become increasingly common and important in the coming years, and the work presented in chapter 3 of this thesis is potentially an important step into this domain.

1.4 Video Analysis

Historically, CV datasets emphasized still images over video sequences. This is likely a reflection of the historical availability of camera hardware and the computational limitations of algorithms, processing, and storage hardware available at the time. However, the proliferation of accessible video capture hardware (e.g. cellphones) have accelerated the rate of video creation, GPGPU computing is now ubiquitous, and video processing is becoming increasingly important. While videos are simply collections of images, they introduce an important new dimension to the data not found in still images: time. Concretely, 2D images have spatial information, 2.5D (e.g. Kinect [159] [46]) and 3D (e.g. LIDAR) images have spatial and depth information, and videos have temporal information. In keeping, 2D videos are three dimensional; 2.5D and 3D videos are four dimensional. As a side-effect, videos require significantly more storage space and computing power to process than still frames. However, they also encode much more information, and are a more general data modality (i.e. an image is a video at a singular point in time). In this thesis, we address two video analysis tasks: semi-supervised video instance segmentation in chapter 2, and visual tracking-by-detection in chapter 3.

Since adjacent frames in a video are captured sequentially, the frames' contents generally exhibit spatial and temporal continuity. That is, assuming a standard video frame rate (such as 24 or 30 frames per second (FPS)), objects move through a video in a consistent manner with smooth spatial and temporal displacements. In the case of the beam trawl videos (presented in detail in chapter 3), frames are captured at 24 FPS, meaning there are roughly 42.67 ms between each frame. Although some objects can move very quickly, not many get far in a 43 ms span. As such, due to the high capture rate of video frames, objects in the scene move regularly in video, with (approximately) piecewise linear motion. This allows using information inferred in frame t to inform processing in the neighboring frames within a temporal window $t \pm N$. In theory, faster moving scene elements have smaller N, while slowly varying ones can afford larger N. This semantic dependence makes integrating temporal information challenging. In addition, by re-using inferred information, errors can cascade to later frames. However, simply treating each frame as being independent in a video is an incorrect assumption – at best, it is inefficient (since treating frames as being independent discards useful information); at worst, it limits the information that can be gleaned from video (maintaining a consistent identity through time is important for accurate scene characterization, e.g. counting the number of distinct objects in the video sequence). Both the VOS network presented in chapter 2 and the video tracking-by-detection dataset in chapter 3 operate over videos and explicitly integrate temporal information.

Chapter 2: ReGuide: Guidance Windows for Long-term Appearance Models in Video Object Instance Segmentation

2.1 Introduction

Segmentation is the process of assigning labels to pixels in a scene indicating logical groupings. It is a fundamental CV problem, playing a central role in myriad tasks, such as medical image analysis, action recognition, video summarization, and video editing. Segmentation is also very challenging: similar to classification, segmentation requires parsing a scene into groupings; unlike classification, segmentation operates on the per-pixel level, and thus has very stringent localization requirements. Image segmentation has been a longstanding CV task, while video segmentation has seen increasing interest in recent years. Although image and video segmentation are very similar, video segmentation introduces a temporal dimension to the problem. The temporal aspect of video segmentation introduces new challenges, such as increased data sizes, object appearance shift, object interactions, partial or full occlusion, etc. It also presents new opportunities, such as being able to leverage motion cues and share information among temporally localized frames. Although image segmentation algorithms can be applied as-is to video sequences (as a sequence of per-frame image segmentations), this implicitly treats each video frame as being independent, which is generally false. Temporal information provides important information, and many state of the art video segmentation algorithms have some form of temporal information built into them, e.g. [62], [79], [72], [122], [52], [59], [104], [143], [69], [155], [57], [153], [20], [152], [51]. Intuitively, video sequences are locally continuous, i.e. adjacent frames are slowly-varying and movement between frames is smooth and slow [87]. This allows re-using information inferred from one frame to the next, and is important for tracking and motion estimation techniques.

Segmentation can be roughly subdivided into semantic and object (non-semantic) segmentation. Semantic segmentation involves assigning a semantic class label to segments, and has seen considerable interest in recent years. In contrast, object segmentation does not require any ontological parsing, and only needs to separate foreground from background elements in the scene. While this seems like a simpler problem (i.e. it is semantic segmentation without requiring classification), object segmentation generally requires segmenting a broader set of objects, and is a less constrained problem. Specifically, semantic segmentation datasets have a finite number of semantic categories: PASCAL VOC [34] has 20 classes, Cityscapes [25] has 30, and MS COCO [82] has 80. As such, semantic segmentation algorithms only have to learn the appearance models for a restricted set of objects, and anything else is classified as background. In contrast, object segmentation datasets have no such categories and are unconstrained in the set of objects that can be labeled as foreground. As such, a successful object segmentation algorithm must be flexible, capable of segmenting a broader class of objects, and have learned a concept of *objectness*, which semantic segmentation approaches do not necessarily require.

Semi-supervised segmentation refers to a particular formulation of the segmentation problem, in which the ground truth mask is provided for the 1^{st} frame of the sequence. This serves to inform the algorithm of the foreground object(s) to segment, and more importantly, reformulates the problem into one of mask propagation and refinement. More specifically, unlike unsupervised segmentation, where the algorithm is required to discover what the foreground versus background objects are (which requires developing a notion of saliency and/or generic concepts of groupings, e.g. Gestalt laws), the foreground objects are provided. Thus, the problem becomes one of propagating information through time and re-identifying and accurately localizing the given foreground object(s) in subsequent frames. In this sense, although still predicting a mask output, semi-supervised segmentation differs significantly from other forms of segmentation.

Instance segmentation is a generalization of binary segmentation, in which the foreground objects have distinct labels which must be maintained throughout the sequence. Rather than viewing foreground objects as a homogeneous entity and performing binary foreground versus background segmentation, each foreground object has a unique ID, which corresponds to the label value in the mask predictions. This introduces more stringent constraints of temporal continuity, since specific object identities must be maintained through time. Instance segmentation has seen increasing interest in recent years, and this work we focus specifically on video object instance segmentation.

2.1.1 Motivation

The motivating idea behind the guidance window construct stems from the observation that current state of the art VOS approaches do not use long-term appearance models. Many successful contemporary approaches only integrate temporal information through the prior frame's mask and/or optical flow, but do not build appearance models over the span of multiple frames. This seems like a serious shortcoming when working with video data and successfully constructing and integrating long term appearance model information should prove valuable in building robust VOS algorithms.

2.1.2 The Guidance Window

Pre-DL approaches constructed long-term appearance models, but would generally use hand-crafted feature extractors, such as SIFT [85] or HOG [29]. Learned deep features have largely proven superior to hand-crafted features in many CV tasks, although success thus far in tracking has been mixed [49]. This is likely due to the fact that deeper features are semantically richer, but have coarser spatial localization accuracy, which is important for tracking and segmentation tasks. Ideally, a video segmentation and/or tracking approach would utilize deep features while simultaneously making use of long-term appearance information. In our formulation, we aim to update the approach used in [77] and [151], which used a multi-output regularized linear regression formulation for visual tracking. We use this as the mathematical underpinnings of our network, and approximate it using the deep convolutional and recurrent network described in section 2.3.

The formulation is as follows:

$$\begin{aligned} y &= Xw \\ w &= (X^T X)^{-1} X^T y \\ Xw &= X(X^T X)^{-1} X^T y \end{aligned}$$

The variables are defined as:

- w: the output guidance construct; if using a guidance *vector*, w is a `[d x 1]` vector.

If using a guidance *window*, w is a matrix of dimension `[d x k]`

- X: the image features; this is a matrix of dimension `[n x d]` extracted from a CNN layer. In our formulation, X is a smaller sub-block of the total feature, namely a `[13 x 13 x 512]` feature slice.

- y: the instance-dependent information; namely, X is instance agnostic, and contains feature(s) for the entire frame. y needs to have the instance-specific information. The dimensionality of y is dependent on the dimension of w – if w is `[d x 1]`, then y is `[n x 1]`. If w is `[d x k]`, then y has to be `[n x k]`.

To compute $(X^T X)^{-1}$, we take a low-rank approximation as $(X^T X)^{-1} \approx \sum_{i=1}^{b}(k_i k_i^T)$, where b is the dimensionality of the approximation, and k_i is a `[d x 1]` dimensional vector.

$$\begin{aligned} Xw &= \sum_{i=1}^{b}(X k_i k_i^T X^T y) \\ &= \sum_{i=1}^{b} [X k_i] \left[k_i^T X^T y\right] \\ &= \sum_{i=1}^{b} c_i [X k_i] \end{aligned}$$

Using this grouping, we define $\left[k_i^T X^T y\right]$ as the target-dependent component c_i, and $[Xk_i]$ as the instance-independent feature f_i, for $i \in [1, \ldots, b]$. More specifically, the f_i feature is the same across all instances, since it is devoid of any instance-specific information. Thus, we can compute it once per feature block X and re-use it for all instances. In contrast, c_i involves y, which encodes information specific to a particular instance. The other insight of note is that c_i is a scalar value, so for a given instance we will have a b-dimensional vector c. f_i is `[n x 1]`, so f is a `[n x b]` matrix for a given feature block X.

In this work, we present a novel, online, end-to-end trainable method for explicitly integrating long-term temporal information by utilizing a bilinear LSTM. This builds richer appearance models over a sequence and provides a method for online adaptation to object appearance shift. Our guidance window is a flexible and extensible network component that can be integrated into a wide variety of architectures, and can be used in tandem with other temporal refinement methods. Experimental results show great promise for harnessing the structure of video data, and improving temporal features in video analysis tasks. Experiments presented in section 2.5 show significant improvement over the baseline architecture, validating the utility of the guidance window.

2.2 Related Works

VOS has long been an active area of research, and has enjoyed considerable interest in the past few years. Prior to DL, there were multiple distinct approaches to the problem which integrated temporal information. Among unsupervised methods, clustering methods were prominent, which would track feature points across multiple frames, then cluster them based on similarity metrics, e.g. [12], [95], [96], [76] and [97]. Other approaches defined spatio-temporal superpixels and optimize over graphical models built from 3D video volumes, such as [137], [55], and [6]. Both supervised and unsupervised segment proposal-based approaches have been proposed, such as [141], [75], [68], [158], [77], [99], and [151]. Broadly, these approaches generate pools of overlapping segments per frame, and use various heuristics (e.g. appearance similarity, boundary coherence, motion, contour cues [11], etc.) to rank and merge proposals into temporally consistent segmentation volumes. The proposals represent hypotheses regarding what in the scene is plausibly a foreground object, and the selection process amounts to ranking the different hypotheses. More recently, CNN-based approaches have become dominant for VOS, likely due to the superior appearance features provided by deep networks. Despite the widespread usage of long-term temporal information in pre-DL methods, temporal information is difficult to integrate with CNN-based methods, and contemporary methods make comparatively limited use of the temporal dimension in videos.

Modern semi-supervised VOS datasets started from Segtrack [137], and have grown to include YoutubeObjects [107], SegTrackv2 [77], VSB100 [38], and more recently, DAVIS-2016 [103]. DAVIS-2017 [106] expands on DAVIS-2016, adding more sequences and instance segmentation annotations. The DAVIS Challenges in 2017 [106] and 2018 [18] have helped to distill the field and provide an objective benchmark to compare alternative VOS methods with. Early approaches to DAVIS made limited use of temporal information; OSVOS [17] was an initial successful algorithm that has inspired many subsequent ones, and formulated video object segmentation as a per-frame image segmentation problem. However, its performance has since been surpassed by methods that use temporal information. MaskTrack [104] was another early method with many similarities to OSVOS, except it used the previous frame's predicted mask as a network input for the next frame. This effectively reformulated the segmentation problem as a mask propagation and refinement task. This has become a common approach for state-of-the-art VOS algorithms, with numerous methods (including the top-placed DAVIS-2017 Challenge entries) making use of it, e.g. [62], [79], [72], [122], [52], [59]. The recently concluded DAVIS-2018 challenge continued the trend: [7] proposed an offline, graphical model with a Markov Random Field incorporating unary, appearance, and temporal energy terms; however, they simply use optical flow computed between the $K = 2$ proceeding and following frames for the temporal pixel-energy term. Since optical flow relies on smooth movement for accurate pixel associations, it is unable to make use of longer temporal connections (i.e. $K > 2$), and may struggle with video sequences that exhibit fast motion or which have low frame rates. Many methods incorporate temporal and motion information by passing the previous frame's mask prediction and/or optical flow as an input for the current frame. This provides very short-term temporal information, utilizing only the immediately adjacent few frames. We believe that making better use of longer historical information would lead to more robust features and superior tracking and segmentation performance.

There have been recent attempts at integrating long-term temporal information for VOS. MaskRNN [52] uses the previous frame prediction as input, but runs backpropagation-through-time over seven frames, thereby incorporating longer dependencies within a video sequence. [122] uses an offline spatio-temporal segment proposal tracking system that runs a backtracking re-tracker that improves its segment appearance model over the entire video. [79] uses an offline, iterative re-identification module to re-track dropped or ID-switched segments. In both cases, the long-term appearance information is added via a separate, offline module that requires the entire sequence to be processed first. In contrast, our method can be run entirely online, with past temporal information integrated directly into the feature extraction. This improves runtime performance, widens the algorithm's potential application domains, and learns better appearance models. However, our approach is complementary to these offline processing techniques, and could leverage post-hoc re-tracking and re-identification methods.

Online methods that use long-term temporal information have also recently been proposed. [58] uses all of the sequence's past frames and corresponding mask predictions in bilateral space to sample features from. However, the algorithm is only evaluated on DAVIS-2016, it is unclear how the algorithm would generalize to instance segmentation, and performance has since been eclipsed by other methods. Multiple methods have integrated tracking-by-detection approaches to enforce temporal consistency: [21] and [124] both utilize object tracking, with [21] first tracking instance parts, then segmenting and merging them, and [124] using bounding box tracks as a means to enforce spatio-temporal regularity of segments. Although both have favorable runtime performance, their accuracy suffers, likely due to the difficulty of predicting an accurate bounding box. Namely, it is common for the bounding box approach to cut off protruding parts of objects (e.g. arms, legs, or other irregularly shaped parts of). Alternately, bounding boxes can grow too large and incorporate the background; in the general case, bounding boxes are poor approximations of articulated or irregularly shaped objects, and even accurate bounding boxes can incorporate majority background pixels. In either case, the object appearance model degrades and segmentation accuracy suffers.

Building on recent advances in semantic segmentation, there have been efforts to leverage semantic segmentation methods for VOS. This has enjoyed some degree of success on DAVIS, as many of the sequences feature objects that overlap with categories in common semantic segmentation datasets e.g. MS COCO and PASCAL VOC. [16] uses a conditional pixel-wise classifier, and uses MNC [28], a semantic instance segmentation method, to define a semantic prior. Many of the successful approaches for the DAVIS-2017 Challenge used semantic segmentation algorithms to generate segment proposals, such as [122] and [62]. However, there are sequences in DAVIS that are not readily covered by semantic categories in existing semantic segmentation datasets (e.g. sequences `lock`, `aerobatics`, or `car-race` in DAVIS-2017's test-dev split), which limits the direct applicability of semantic segmentation methods.

More recently, meta-learning has emerged as a promising avenue for efficient VOS. [155] learns a spatial and visual modulator, which apply a learned affine transformation to adjust intermediate feature layers to better fit the target instance based on the initial frame ground truth. The visual modulator is a CNN that produces parameter scaling factors, and the spatial modulator produces parameter bias terms. [69] improves on [155] by introducing convLSTMs [125] on the last three feature extraction layers, thereby introducing spatio-temporal features. The spatial and visual modulator formulation remains the same as in [155]. Our work bears resemblance to the meta-learning approach, in that we are learning a layer (the bilinear LSTM), which is then used to manipulate the computed features to better focus on a particular object. However, our approach explicitly does so with a modular, online bilinear LSTM layer (i.e. it can be composed with any feature extraction network architecture and can be dropped into a wide variety of architectures). Our proposed network architecture differs significantly, and although both network's goals are similar, our approaches are very different.

A common approach in semi-supervised VOS is to run extensive fine-tuning at the start of inference; this is only possible in the semi-supervised case, where the ground truth segmentation is provided for the 1^{st} frame. OSVOS, MaskTrack, and LucidTracker all make use of this, performing a wide array of augmentations on the 1^{st} frame image and ground truth mask to simulate future frames. They then run many iterations of fine-tuning on the augmented data, hoping to learn a better fit to sequence's specific objects. In effect, they fine-tune a different model for each sequence in the dataset. This inference-time training is more valuable for the network than training offline on different video sequences, since the data is in-domain. However, this method is computationally expensive and is susceptible to overfitting to the initial frame, thus being unable to adapt to non-rigid appearance deformations in subsequent frames. This can be especially problematic when the object to segment undergoes significant appearance change during the sequence, whether due to viewpoint shift, object motion, or partial occlusion. See fig. 2.1 for an example of the magnitude of appearance shift objects can undergo in a sequence. In contrast, methods such as [143], [69] and [155] use online adaptation to fit the instance appearances at test time, avoiding the per-sequence runtime overhead and better accommodating non-rigid instance deformations. However, the online adaptation and inference-time fine-tuning are not mutually exclusive, and some report better performance doing both [69]. We believe that extensive reliance on run-time fine-tuning is not a robust solution to VOS: it is exclusive to the semi-supervised formulation, overfits to the initial object appearance, and requires training a new model for each sequence being evaluated, which is both inelegant and computationally expensive. It also is only effective for sequences where the object appearance does not undergo dramatic defor-

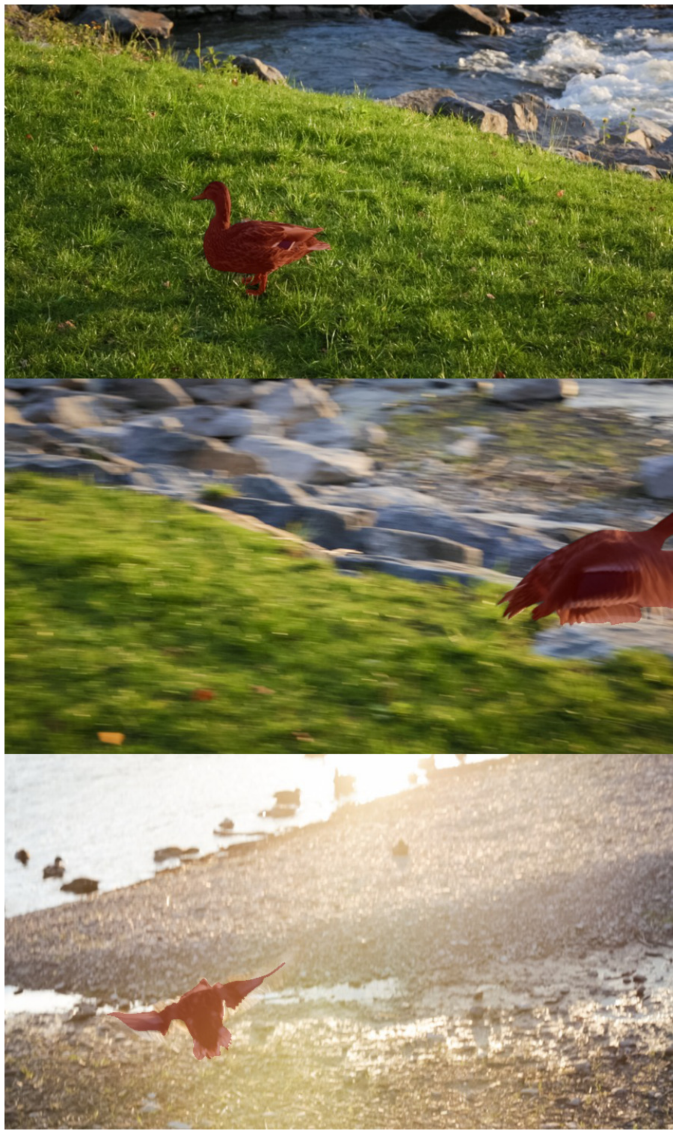


Figure 2.1: 1st, 47th, and 69th (last) frame of DAVIS sequence `mallard-fly` with ground truth mask overlays demonstrating the high degree of appearance change that an object can undergo throughout a sequence, and the perils of overfitting to the first frame

mations through time. This is because the 1^{st} frame augmentations cannot accurately model the full range of object affordances, and is mostly limited to rigid transformations. This is a fundamental limitation, as with just the first frame and mask, it is impossible to know how objects can move in the scene without further semantic information about the objects. For example, augmentations would not be able to capture the appearance shift that the duck in the first frame of fig. 2.1 undergoes, as without intimate knowledge of duck physiology, there is no information regarding what parts of the duck can move or how they move. In turn, having semantic information of scene elements is problematic for VOS, since not all objects to be segmented are common semantic categories. We explicitly design the ReGuide network to *not* rely on online fine-tuning; although it can make use of online fine-tuning with augmentations in the style of LucidTracker, we primarily rely on the guide's ability to naturally adapt online to the instance appearance shifts through the memory component of the guide layer. None of the presented results in this thesis use fine-tuning.

2.3 Guide Window

The main architectural components of the system are explained in the following sections. All components are trained end-to-end. It is important to note that our approach is geared towards simplicity, clarity, and validating the utility of the guide window, rather than maximizing segmentation performance. As such, there are no post-processing steps or extensive engineering tricks for improving performance, such as using segmentation network ensembles, multi-crop inference, or extensive hyper-parameter search. The overall network architecture is shown in fig. 2.2.

1. Initial features are computed with DeepLabv2 features [19] with a ResNet-101 backbone (we don't use the DeepLabv2 fully-connected CRFs or spatial pyramid pooling); we also have a set of lateral connections to each residual block (up to the final average pooling layer) similar to the ones in Feature Pyramid Networks (FPN) [80]. Further details are in section 2.3.1.

2. Guidance Window: This construct encompasses the guide component. It has three main elements:

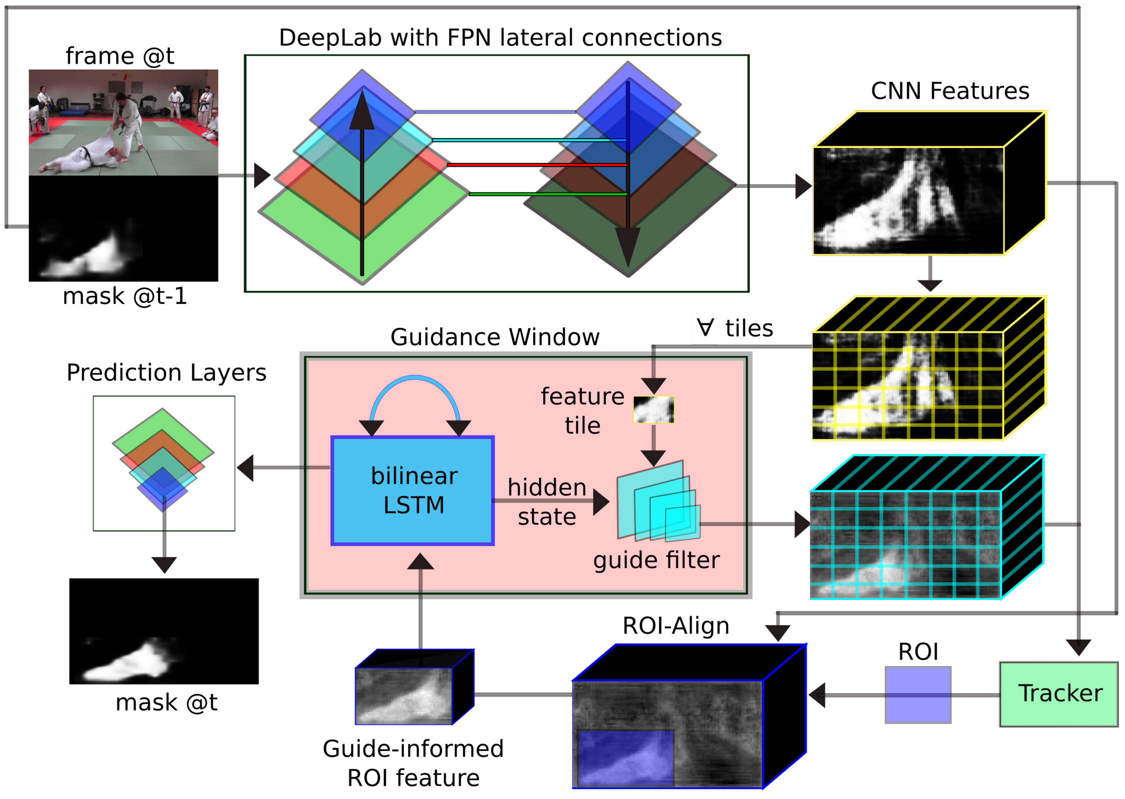


Figure 2.2: The architecture diagram and data flow for the ReGuide network. For each frame t, we run this sequence through the guidance network for each instance i. We thus generate a predicted mask per instance per frame, and use the predicted masks as temporal inputs for the network in subsequent frames. The bilinear LSTM is shown in more detail in fig. 2.4, and the tiled guide filtering in fig. 2.3. Best viewed in color.

- A pair of 1x1 convolutions to fix the depth-dimension of X and y features to be passed to the bilinear LSTM.
- The guide bilinear LSTM. Further details about the bilinear LSTM are given in section 2.3.2.1.
- The guidance filter, whose weights are taken to be the bilinear LSTM's last hidden state vector.

   The bilinear LSTM learns object-specific appearance information as its hidden state. The guide filter uses the previous frame's bilinear LSTM hidden state to compute the guide-informed CNN features. More details are in section 2.3.2.

3. Mask Predictor: A sequence of dilated (atrous) convolution layers for computing the output mask, similar to DeepLabv2. The predictor receives features from the guide components to generate the output mask (at the ROI dimensionality). More details are in section 2.3.3
4. Instance Tracker: In the present architecture, this is an extremely simple tracker. We expect a more sophisticated tracker integrating occlusion detection would be beneficial for performance. More details are in section 2.3.4.

The network input for a given frame t consists of the image (I_t), previous mask (M_{t-1}), and masked image (the image pixels at non-zero locations of the mask). Given these inputs, a DeepLabv2 ResNet-101 feature extraction network with FPN-style lateral connections computes the initial, whole frame features. These are then tiled into blocks for the guide to filter chunk-by-chunk, which then are re-composed into the guide-informed CNN features. The guide-informed feature tensor is used to compute a ROI for the instance in the tracker, which defines a bounding box for the instance in the guide-informed feature map. A normalized feature tile is then extracted using roi-align located at the tracker's ROI, which is finally used to update the guide's bilinear LSTM. The resulting filtered feature block is passed through the mask prediction layers, mapped back to image space, and written to an output frame. This overall system information flow is depicted in fig. 3.5, and the individual architecture components are explained in more detail in the following sections.

Although our algorithm performs instance segmentation, it formulates multi-instance sequences as a collection of single-instance sequences, and segments each instance independently. Then, the individual instances are merged in a post-processing step and mapped to their appropriate instance labels. This allows for the guide to better adapt online to individual instance appearances, and disentangle the visual elements of the scene.

2.3.1 Feature Extraction

We use DeepLabv2 with a ResNet-101 backbone to extract initial convolutional features. DeepLabv2 uses atrous convolutions to preserve spatial locality information, and serves in our network as a means to compute rich features over salient objects in the scene. It is important to note that the extracted features are instance-agnostic, and the DeepLabv2 network can be thought of more as a generic segmentation subnetwork. In addition, we added an FPN-style set of lateral connections to residual blocks 2 through 5, building a top-down feature map incorporating the multi-scale CNN feature maps. This is motivated by the duality inherent to deep features for localization tasks; while the later residual blocks produce semantically richer features, segmentation and tracking tasks need to preserve the spatial locality present in earlier feature maps. Also, features suitable for tracking (e.g. color) are often found in earlier feature layers [157]. As such, it is important to facilitate the preservation of certain shallow features, while taking advantage of the semantically richer deep features; the lateral connections assist in the flow of information from earlier feature maps in the CNN. The DeepLabv2 features are computed over the entire (full-resolution) frame, and thus contain features for all instances in the sequence. The guide window focuses attention on the specific instance to be segmented in the sequence.

2.3.2 Guide Window

The guide processes fixed-sized feature blocks from the CNN features, and interacts with the CNN features in two stages. In the first stage, the full-resolution CNN features are subdivided into overlapping tiles based on the guide feature size, and convolved sequentially with the guide filter. See fig. 2.3 for details. The rationale is that the guide

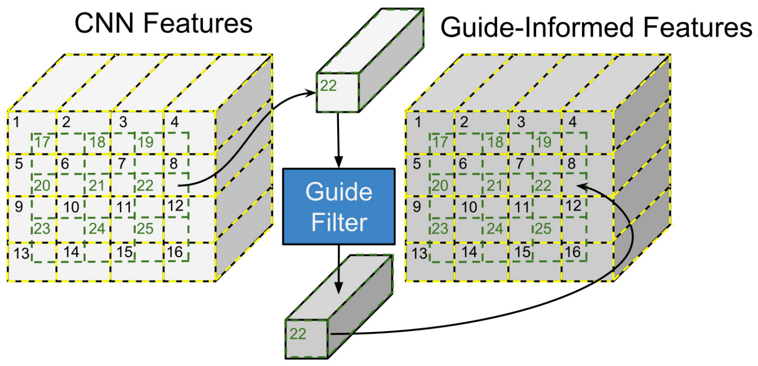


Figure 2.3: Diagram of the feature tiling; the CNN features are divided into overlapping tiles of fixed dimensions, such that every element in the feature tensor is covered at least once. Then, each tile is convolved sequentially with the guide filter, and summed into the guide-informed feature tensor at the original feature coordinate locations. The guide-informed features are then divided-elementwise by the number of tiles the element is covered by. In this figure, the tile number is in the upper-left corner of its corresponding tile.

has learned an accurate appearance model for the target object, and will have higher responses on region where the instance is present and lower responses on non-instance pixels. In contrast, the CNN features are largely agnostic of the particular instance to be segmented, so this step focuses the features on the target instance. However, we do not want to update the LSTM hidden state, since we are processing the entire scene, much of which will not contain the instance. To avoid polluting the LSTM state, we convolve the tiles with the stored guide filter. The guide-informed tiles are summed into a full-resolution feature tensor at each tile's origin coordinates, and the entire feature map is averaged based on the number of times each pixel was processed by the guide. The resultant feature map is used in conjunction with the previous mask prediction by the tracker (section 2.3.4) to compute a region-of-interest (ROI) for the instance.

2.3.2.1 Bilinear LSTM

The bilinear LSTM is an important component in the guide layer. We use the bilinear LSTM formulation from [63], which proposed the bilinear LSTM as a gating network in the Multiple Hypothesis Tacking (MHT) domain. Specifically, the bilinear LSTM is used to learn the long-term appearance model features of objects, which are used for track hypothesis ranking. While traditional LSTMs have had limited success in learning sequential appearances models, the bilinear LSTM improves upon vanilla LSTMs by using a multiplicative relationship between input and memory. Concretely, the bilinear LSTM uses the Hadamard product to integrate information from the input vector and hidden state, while conventional LSTMs simply sum them. This formulates the integration as a form of gating between the input and internal state, and changes the LSTM from first order to second order [150] [45]. This allows more expressive interactions between the input and state vectors: in vanilla (additive) LSTMs, if one information source is large and the other small, the small one will have no effect. In bilinear LSTMs, the product operation ensures both information sources have significant magnitudes of effect. This also formulates the bilinear LSTM memory as a component of a least-squares regressor, as described in section 2.1.2. While [63] used bilinear LSTMs for multi-object tracking appearance modeling, we use it for object instance segmentation.

The bilinear LSTM update equations are given by:

$$\begin{aligned}
i &= \sigma(W_{ii}x + W_{hi}h^{'} + b_i)\\
f &= \sigma(W_{if}x + W_{hf}h^{'} + b_f)\\
g &= tanh(W_{ig}x + W_{hg}h^{'} + b_g)\\
o &= \sigma(W_{io}x + W_{ho}h^{'} + b_o)\\
c &= f * c^{'} + i * g\\
h &= o * tanh(c)
\end{aligned}$$

where x is the input, $h^{'}$ is the previous hidden state, $c^{'}$ is the previous cell state, W_{zi} is the input weight matrix, W_{hz} is the hidden state weight matrix, and b_z the bias for each information source. In this notation, z corresponds to the different gating and activation functions, i.e. $z \in \{i, f, g, o\}$ for i being input, f forget, g cell, and o output. h is the next hidden state, c is the next cell state. The bilinear LSTM also retains spatial information. Concretely, although it still uses a $1D$ hidden state, we simply flatten the $2D$ input, then re-shaped the resultant hidden state back to $2D$. Since the bilinear LSTM is trained and the reshaping operation is deterministic, the bilinear LSTM implicitly learns the reshaping protocol. Intuitively, the guide filter can be viewed as learning a template of the object appearance; by using a multiplicative relationship between memory and input, the filter vectors can be viewed as different appearance templates, which explains how the bilinear LSTM is able to adapt to appearance changes in the video sequence.

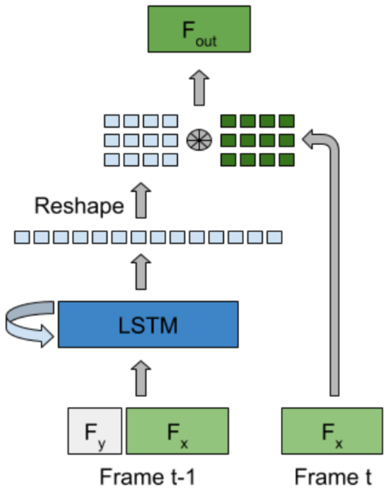


Figure 2.4: Bilinear LSTM module diagram. The inputs are the N past feature tensors F_X and F_y, which are used to update the LSTM hidden state, and store the LSTM output as a convolutional filter. In the next frame, the image features F_X are convolved with the filter, yielding improved, instance-specific features.

2.3.2.2 Guide ROI Features

After we get an instance ROI from the tracker, we apply the ROI to the original CNN feature map to extract a fixed-size feature block. We use roi-align for this step, as described in [47]. This is an important step, as the guide operates on fixed-size feature windows, but the object sizes are variable. Recomputing the features based on the image-space tracker ROI would be computationally inefficient, and roi-align has been shown to avoid the spatial quantization errors present in other ROI pooling methods.

Conceptually, the roi-align'ed feature block is X in the mathematical formulation described in section 2.1.2. Due to computational constraints, it is first convolved with a 1x1 convolution to reduce the feature depth of X; otherwise, the guide consumes too much memory (further details are provided in section 2.4.3). The fused X feature is passed into the bilinear LSTM module as input. This updates the LSTM's hidden state (i.e. the guide's instance appearance model). By using the ROI feature block rather than the whole frame, we derive two benefits:

1. An accurate ROI provides the LSTM with features consisting of minimal non-instance scene elements, thereby ensuring that the online appearance model adaptation is accurate for the target instance.
2. Despite objects being of different scales, the feature sizes are uniform. Thus, the network handles large scale variations, preserves object details, and increases segmentation performance on small objects.

After the forward pass through the LSTM, its hidden state is stored as a 1x1 convolutional filter and operates as the guide filter thereafter. This filter is then applied as a 1x1 convolution to the frame's roi-align pooled X feature to generate an updated, guide-informed feature block for the current frame. By using the bilinear LSTM hidden state as the convolutional filter and jointly training the features and LSTM, the guide can learn to model the long-term appearance information of specific objects. This has the benefit of being able to adapt over the course of the video sequence to object appearance shifts and integrate richer sources of information about the instance over longer time spans. The differences between the different features' focuses are shown in fig. 2.5.

2.3.3 Mask Prediction

The prediction layers consist of a sequence of dilated convolutions, as in DeepLabv2. The output predicted mask is generated from the guide-informed features, corresponding to the feature-space region of interest from the tracker. To map the predicted mask back to image-space, the mask is bilinearly reshaped to be of the same dimension as the input ROI in feature-space, and is written back to a full-resolution frame of zeros at the tracker's ROI in image-space.

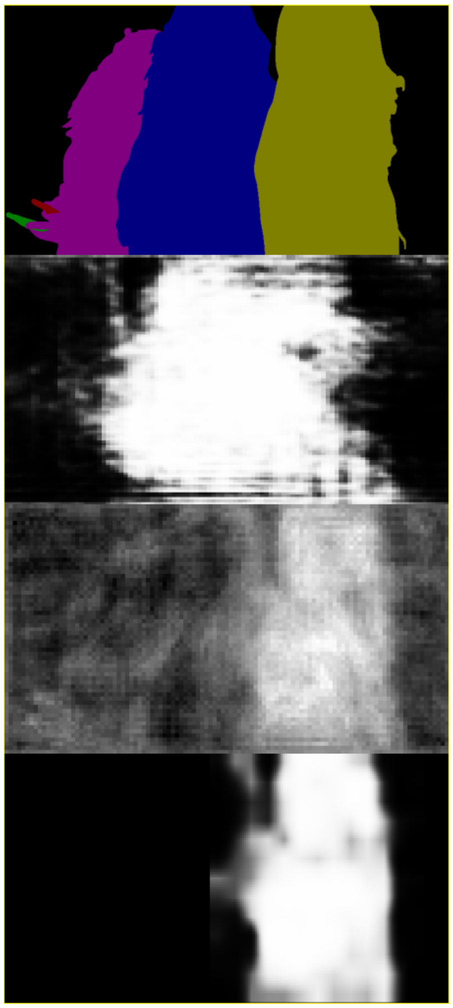


Figure 2.5: Different mask predictions based on the features used; the 1st image is the ground truth (features correspond to instance #3, the rightmost yellow one), the 2nd is from DeepLabv2 (non-guide-informed) features, the 3rd one is the tiled guide-informed features, and the 4th is the guide-ROI features. Note the progression from generic segmentation to specific object in the resultant mask predictions.

2.3.4 Bounding Box Tracker

Despite the benefits of using cropped ROI inputs for the guide (i.e. better effective resolution, smaller memory footprint, and less visual distractors), it also introduces an important dependency on having accurate ROIs. We have found that the approaches of [62] and [102] of using the optical flow-warped previous mask is insufficient for computing an accurate bounding box in the current frame. Ideally, a bounding box will be tight, including no extraneous scene elements, but also encompassing the entire object. An inaccurate bounding box will provide the LSTM with unrepresentative information about the instance, and the segment appearance model will degenerate. These errors can cascade, as an inaccurate appearance model will lead to worse predictions, which in turn lead to less accurate ROIs in the following frames. Bounding box errors fall under two categories:

1. The predicted ROI truncates part(s) of the object: the guide will then update its object appearance model to similarly exclude the truncated regions, and will be less likely to include them as part of the object in subsequent frames, even if the bounding box encompasses them again.

2. The predicted ROI is overly large: the guide features may pollute the object's long-term appearance model, which can lead to ID switches, segments for distinct but spatially adjacent instances fusing, or conversely dropping the segment entirely.

Therefore, the tracker must fully contain the object of interest, while simultaneously excluding non-object scene elements as much as possible. However, between the two bounding box errors, the former is more detrimental than the latter, as the guide is entirely deprived of the information, and recovering an accurate appearance model is more challenging in this case. The latter is more of an issue when there are visually similar, but distinct instances in the scene (such as the DAVIS sequences `camel`, `gold-fish`, or `salsa`).

In the present architecture, the tracker performs warping on the previous frame mask prediction with the current frame's inverse optical flow frame (if provided). Specifically, the mask warping is done using the inverse optical flow frame from t to $t-1$. The optical flow is computed using DCFlow [154]. This acts as a simple, albeit noisy, motion model. Frame appearance features are not integrated, but doing so would likely improve bounding box accuracy. Then, the warped mask's elements above a given threshold are tallied, using the median coordinate location as the bounding box center, and defining the bounding box dimensions using standard deviations along both spatial dimensions.

We expect that a more sophisticated tracker would be beneficial for computing tighter bounding boxes; the present tracker was chosen for the sake of simplicity, and to ensure the instance would not be truncated.

2.4 Implementation

We implemented the network in pytorch, and trained the model with a combination of offline training and online semi-supervised fine-tuning. Although the online fine-tuning is not required, we found that it improves mean intersection-over-union (mIoU) performance, as it informs the guide which scene elements to emphasize. However, it also increases the inference times significantly. Since the goal of this work is not necessarily to maximize performance, but rather to validate the guidance window concept, we elected to forgo inference-time fine-tuning in experiments. Experimental results are in section 2.5. We use two GTX 1080 Ti GPUs during training and inference. Offline training on DAVIS-2017 train & val splits for 60 epochs takes $\sim$ 55 hours, inference on test-dev without test-time fine-tuning takes $\sim$ 0.5 hours. With test-time fine-tuning, inference takes over an hour per sequence.

2.4.1 Offline Training

We train the network using DAVIS-2017's train and val splits (with 60 and 30 sequences respectively) for 60 epochs. Since we split each sequence per instance, this yields 205 effective sequences. Since we are training the same model using different sequences, it learns to be a general object detector, relying on the previous frame's predicted mask to indicate the desired foreground element to segment. We use a learning rate of 10^{-5} with a learning rate decay update step of $lr = lr * (1 - e/e_{max})^{0.9}$, where e is the current epoch, e_{max} is the maximum number of epochs, and lr is the learning rate. We use a weight decay of 0.0005 and use SGD with momentum 0.9. Since the network makes use of the previous frame's mask prediction, during training we randomly use a portion of ground truth masks and predicted masks as input. Specifically, we start training using 75% ground truth and 25% prediction, and reduce the proportion of ground truth until it hits 0% in the last epoch. We perform affine and elastic deformations when using the ground truth mask, and dilate the mask to remove finer details to preclude overfitting. Using mask predictions rather than ground truth has multiple benefits: it acts as a regularizer for the network, prevents over-reliance on simply refining the previous frame mask, better simulates conditions during inference, and ameliorates overfitting. In addition, to better handle occlusion and instances exiting the scene, we append N extra frames to the end of each training sequence from a different sequence, with all-zero (i.e. background) ground truth. This helps inhibit spurious segments in the frame in the case of occlusion, which can lead to ID-switches and/or pollute the guide LSTM hidden state. We use multiple loss sources:

- Weighted binary cross-entropy loss over the full-resolution tiled guide-informed features. This fine-tunes the DeepLabv2 feature extraction, and gives the guide and mask prediction layers a broader field of view during training.
- Weighted binary cross-entropy loss over the final mask ROI prediction. This helps to fine-tune the bilinear LSTM to focus on the given object.

We also use loss weighting to further penalize predictions made in cases where the object has exited the scene, or is fully occluded. Also, we assign higher loss weight to false negatives on small objects, to combat the class imbalance small objects suffer from. Experimentally, we have found that the initial few training iterations benefit from using very low learning rates (e.g. $\sim 10^{-7}$) to prevent the guide from diverging. Once the guide training has stabilized, we reload the model and begin training with a higher learning rate.

2.4.2 Online Fine-Tuning

Although online fine-tuning based on the first frame runs the risk of overfitting to the initial frame and generalizing to the rest of the sequence poorly, it can function as a system-wide prior for informing the system which objects emphasize. While the offline-trained network learns general object features, the test-time fine-tuning helps to better fit the current sequence. In turn, this assists the guide in learning better features for the specific instance to be segmented. To run the online training, we use the extensive augmentation algorithm from LucidTracker, generating 250 individual smoothly-varying triplet sequences per video consisting of the image, ground truth mask, and forward and inverse optical flows. Although we refer the reader to [62] for further details, we use foreground affine transformation parameters of $\pm 30°$ rotation, $\pm 15\%$ scale, and $\pm 10\%$ translation and background parameters of $\pm 10°$ rotation, $\pm 15\%$ scale. The foreground transformations simulate potential object movement and appearance variation in the scene, and background transformations simulate camera motion. We also use non-rigid foreground transformations, approximating elastic deformations by generating a grid of random numbers, filtering it with a 2D Gaussian filter (with $\alpha = 30$ and $\sigma = 5$), and applying the smoothed coordinate offsets to the affine-transformed image, mask, and optical flow frames.

2.4.3 Dimensionality

Ideally, the guide filter would be as spatially large and deep as possible. However, we are limited by LSTM memory consumption constraints. As such, the guide filter is a 1x1 convolution with a depth of 20. The input feature patches (X) have a fixed spatial resolution of [13 x 13] in feature-space, which maps roughly to [101 x 101] in image space. The features computed from DeepLabv2 with a ResNet-101 would ordinarily have a depth of 2048, but our top-down FPN-style refined feature map has a depth of 256 (as is done in [80]), and we use that as the network's CNN features. However, a feature depth of 256 is still too large for the bilinear LSTM; as such, we fuse the X feature depth channels with a 1x1 convolution to 146 (reduced from 256). Functionally, since the guide-filter has 20 output channels, the LSTM takes an input of dimension [13 x 13 x 146], and outputs (via its hidden state) 146 filters of dimension [1 x 1 x 20]. Having greater input and output channels would allow more expressive appearance model representations, and having higher filter spatial dimensions would give wider receptive fields. However, with the given parameters, the bilinear LSTM accounts for more than 10GB of GPU memory during training. As such, using larger filters is computationally intractable. The chosen parameters are the maximum they can be on current hardware, and represent a good trade-off between input feature depth, output feature depth, and filter kernel dimensions.

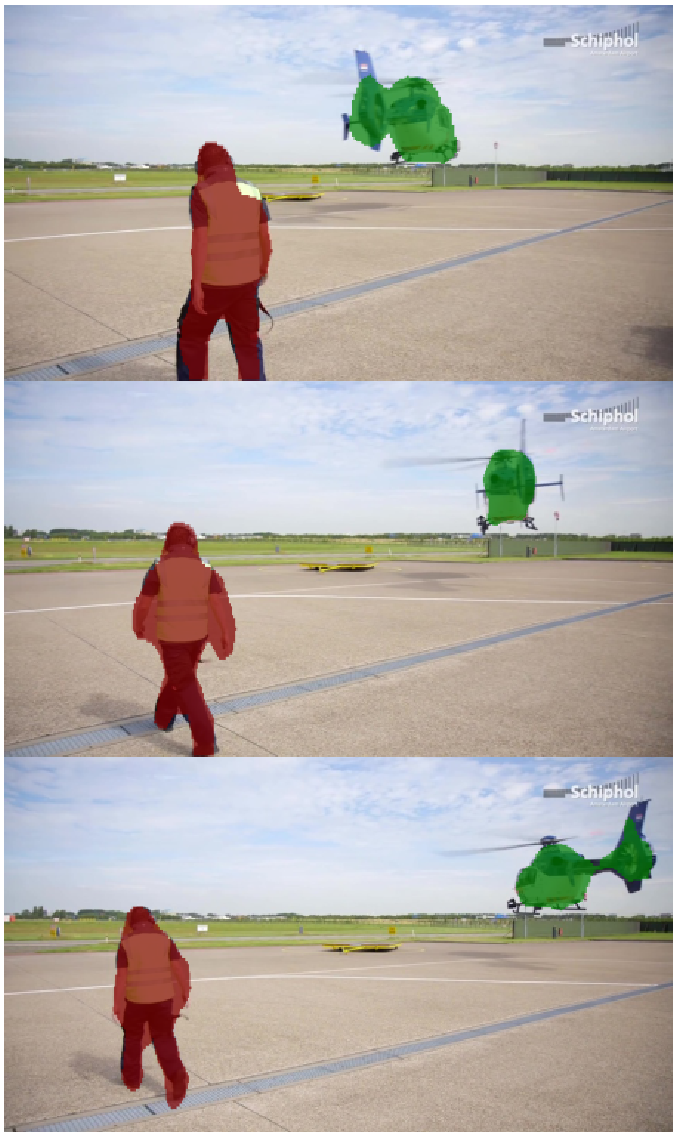


Figure 2.6: Example network results on DAVIS-2017 test-dev sequence **helicopter** with predicted masks overlaid from start, middle, and end of sequence. Note how the segments persist despite appearance shift in the helicopter.

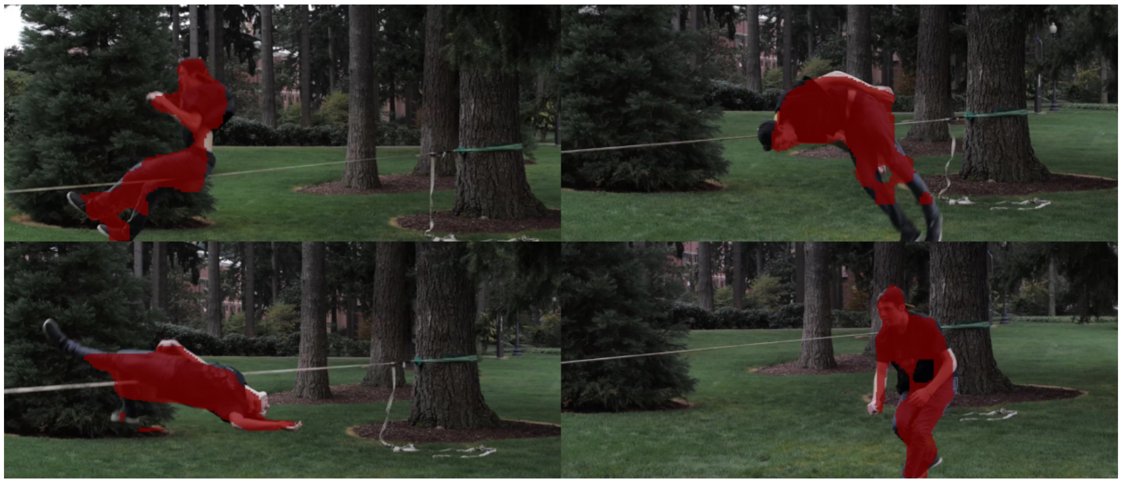


Figure 2.7: Example network results on DAVIS-2017 test-dev sequence **slackline** with predicted masks overlaid from regularly sampled points in the sequence (frames 1, 20, 40, and 59). Note how the network is able to accommodate dramatic appearance shift in object appearance.

2.5 Experiment Results

2.5.1 Baseline Architecture

The baseline architecture is identical to the ReGuide architecture, except without the Guidance Window component. This serves as a clear means to examine the impact of the Guidance Window. Based on the experiment results shown in table 2.1 and table 2.3, the guide component has a large and significant positive impact on segmentation performance across all measurement metrics. The baseline architecture diagram is shown in fig. 2.8

2.5.2 DAVIS

DAVIS-2016 is a modern dataset consisting of 50 high-resolution video sequences with densely annotated ground truth segmentations. DAVIS-2017 expanded on DAVIS-2016, subsuming the sequences from DAVIS-2016 and adding another 100 sequences, and introducing instance annotations. DAVIS-2017 has a total of 10459 frames and 376 total different instance objects, and sequences are subdivided into 4 splits: 60 `train`, 30 `val`,

	Global $\mathcal{J}$ & $\mathcal{F}$	Region $\mathcal{J}$			Boundary $\mathcal{F}$		
Method	Mean ↑	Mean ↑	Recall ↑	Decay ↓	Mean ↑	Recall ↑	Decay ↓
ReGuide	57.30	61.29	75.27	6.62	53.30	56.85	6.35
Baseline	34.10	36.91	27.02	9.25	31.28	13.91	10.32

Table 2.1: Results on the DAVIS-2016 validation dataset. Note that these results are fully online, without any runtime fine-tuning, and without any engineering or post-hoc refinement performance tricks (e.g. ensembling, CRFs, etc). The baseline is the same DeepLabv2 base feature network and prediction layers, but without the guide component. The baseline architecture is described in section 2.5.1. $\mathcal{J}$ corresponds to region similarity (mIoU), and $\mathcal{F}$ is a measure of contour accuracy.

30 `test-dev`, and 30 `test-challenge`. The `test-dev` and `test-challenge` ground truth annotations are not publicly available (other than the 1st frame), and performance is evaluated on an independent evaluation server. Since the `test-challenge` evaluation server is only available during the associated workshop challenge, we primarily evaluate our method using `test-dev`.

Method	DAVIS-2016 $\mathcal{J}$	Fine-Tune	Optical Flow	CRF
ReGuide	61.3	✗	✗	✗
Baseline	36.9	✗	✗	✗
OFL [138]	68.0	-	✓	✓
BVS [88]	60.0	-	✗	✗
ConvGRU [136]	70.1	✗	✓	✗
VPN [58]	70.2	✗	✗	✗
MaskTrack-B [102]	63.2	✗	✗	✗
SFL-B [22]	67.4	✗	✓	✗
OSVOS-B [17]	52.5	✗	✗	✗
OSNM (2^{nd})[155]	74.0	✗	✗	✗
ReConvNet [22]	78.0	✗	✗	✗
OnAVOS [143]	86.1	✓	✗	✓
OSVOS-S [87]	85.6	✓	✗	✗
PLM [156]	70.0	✓	✗	✗
MaskTrack [102]	63.2	✓	✗	✗
SFL [22]	74.8	✓	✓	✗

Table 2.2: Comparison against the state of the art on the DAVIS-2016 validation dataset. Performance numbers used collected from published works. Note that some networks use additional refinement steps and/or information sources, e.g. OSVOS-S [87] uses semantic segmentation proposals, OSVOS [17] uses contour boundary snapping, etc. $\mathcal{J}$ corresponds to region similarity (mIoU).

	Global $\mathcal{J}$ & $\mathcal{F}$	Region $\mathcal{J}$			Boundary $\mathcal{F}$		
Method	Mean	Mean	Recall	Decay	Mean	Recall	Decay
ReGuide	23.0	20.0	17.6	18.6	26.0	17.7	17.8
Baseline	14.5	10.5	5.8	11.1	18.6	6.2	12.5

Table 2.3: Results on the DAVIS-2017 test-dev dataset, as given by the DAVIS evaluation server. The baseline architecture is described in section 2.5.1. $\mathcal{J}$ corresponds to region similarity (mIoU), and $\mathcal{F}$ is a measure of contour accuracy.

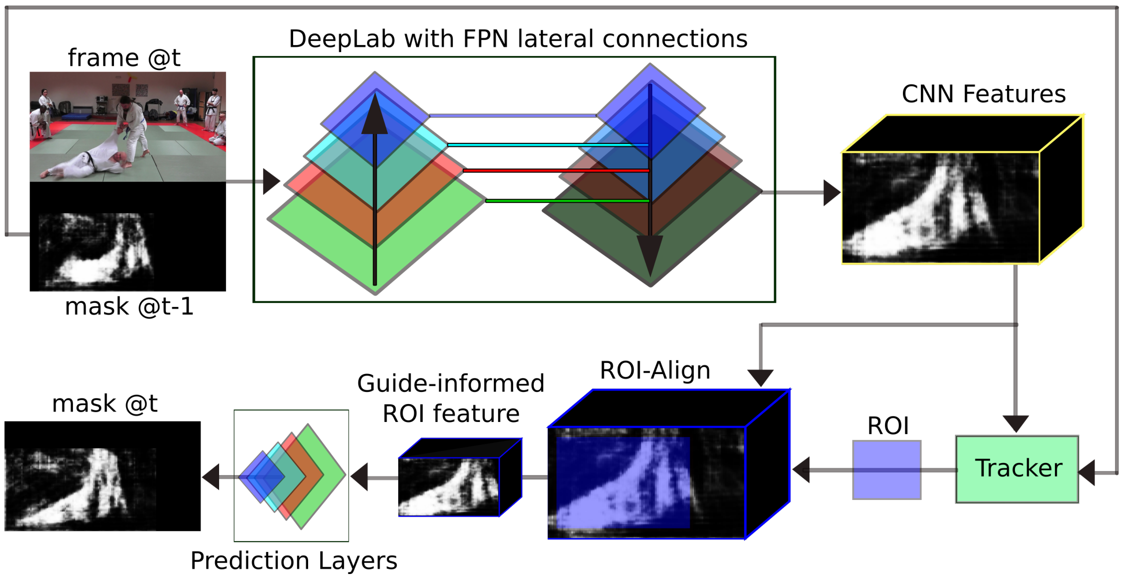


Figure 2.8: The baseline architecture diagram used for validating the Guidance Window component. This is simply the ReGuide architecture, sans the guide-related components.

2.6 Discussion

The guide window learns the long-term appearance models for a particular instance in its hidden state; the target instance is provided by the previous frame mask, which allows the guide to learn appearance models online for generic objects, without requiring online fine-tuning. Functionally, convolving the input features with the bilinear LSTM memory functions like template matching. Specifically, the different guide filters contain different appearance models of the object (e.g. its different appearances through time), and via the multiplicative relationship with the inputs, generates features with a higher response on instance pixels. Similarly, this filtering suppresses image regions that do not belong to the instance. Since the guide memory is updated per-frame over the entire sequence, it can build appearance models integrating information over longer time scales. Conceptually, the DeepLabv2 ResNet-101 feature network extracts more general features over the entire frame, the guide filter focuses the filters on the instance, and the ROI-extracted feature tile is used to update the bilinear LSTM memory with the instance information from that time step. Experiment results show that the guide component is effective, and the bilinear LSTM and associated guide filter successfully modulates learned features to encode long-term appearance information. In doing so, we resolve the two problems identified with current state of the art VOS networks, and create a modular and extensible network layer to perform online adaptation via network modulation:

1. *Limited temporal usage*: The guide layer learns object appearance information over the course of the entire sequence, integrating new information into its hidden state memory. Applying the guide filters to the CNN features injects this long-term appearance information into the network features.

2. *Over-reliance on online fine-tuning*: While numerous state of the art VOS algorithms effectively learn a model for each sequence they're applied on, our guide component operates as a general object tracking machine, and uses the previous mask information to inform what to track, rather than requiring expensive augmentations and in-situ training at run-time.

2.6.1 Limitations

The guide window is not without its limitations; as alluded to in section 2.4.3, the memory consumption of the bilinear LSTM increases commensurately with the guide filter dimensions, which limits the depth and receptive field of the guide filter.

The guide appearance model is sensitive to pollution by non-instance scene elements, and (too) easily merges multiple segment instances. Specifically, each segment often grows to encompass multiple instances when two or more instances occlude each other. This leads to incorrect segment IDs; see fig. 2.9 for an example. This is one of the main reasons for the dramatic decrease in performance between DAVIS-2016 validation and DAVIS-2017 `test-dev` sequences. Performance on `test-dev` sequences that do not exhibit multi-instance co-occlusion are much better, such as the `helicopter` and `slackline` sequence. Results with mask overlays on `helicopter` are presented in fig. 2.6 and `slackline` in fig. 2.7. We believe using a more sophisticated tracker with occlusion detection would greatly ameliorate this shortcoming, as the segment growing problem occurs due to the guide window updating its appearance model to encompass the occluding scene elements as well.

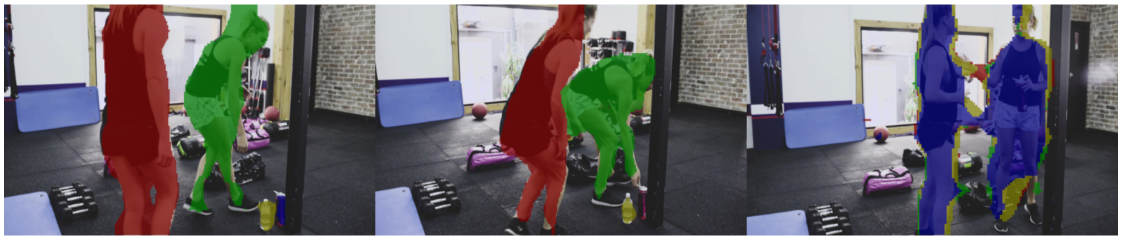


Figure 2.9: Sequence of frames showing the instance region-growing problem: each segment instance grows to encompass all instances, leading to incorrect (and conflicting) instance IDs. However, all foreground objects are still covered.

2.7 Conclusion

We present ReGuide, a novel, online, end-to-end trainable recurrent convolutional network for semi-supervised VOS. Our method integrates long-term appearance information and motion cues through a bilinear LSTM that adapts online to a particular instance.

Our design combines a convolutional network, whose features are computed over the whole frame, and an LSTM, which acts as a guide for the CNN features. In turn, the guide-informed features are used by a bounding-box tracking system to impose spatial and motion smoothness constraints, which are used to update the guide hidden state with normalized regions of the scene centered on the instance. Our system is online, and can use of (but does not require) test-time fine-tuning using the extensive data augmentation utilizing the ground truth 1^{st} frame mask. The proposed guide module is easily integrated into existing convolutional systems, and shows great promise for building discriminative, long-term features suitable for video analysis. Performance on DAVIS-2016 shows competitive results, especially in light of the lack of network elements in ReGuide such as post-hoc refinement, re-identification modules, or optical flow, or run-time fine-tuning, as is commonly found in state of the art methods. Significant improvements over the baseline architecture demonstrates the efficacy of the guide window and its capacity to learn long-term appearance models.

Chapter 3: NHFish: Tracking-by-Detection Dataset of Oregon Beam Trawl Video Data

3.1 Introduction

Research traditionally takes place in disciplinary silos, with insular terminology, methodology, and techniques. Fertile research opportunities arise at the boundaries between disciplines; a notable example is the discovery of the DNA helix structure, which required integrating "genetics, biochemistry, chemistry, physical chemistry, and X-ray crystallography," [1] and gave rise to an entirely new interdisciplinary subject, molecular biology. Even half a century later, DNA research continues to push interdisciplinary boundaries, with the Genome Project involving "scientists in a variety of disciplines, such as biology, chemistry, genetics, physics, mathematics, and computer science." [101] In general, difficult real-world problems, such as climate change [71], span multiple disciplines and require unprecedented and sweeping action across vast swaths of society. Especially pernicious and pervasive problems have been referred to as 'wicked problems', and may not even have definitive solutions or problem statements [112]. Solutions to such problems necessarily integrate multiple disciplines working closely in tandem; in addition to enabling solutions to such problems, interdisciplinary work can produce more creative, integrative, and robust solutions. Many important disciplines arose out of close collaborations between disparate fields, such as computer science or molecular biology. Interdisciplinary work is an important mode of inquiry, and is critical to solving the large-scale wicked problems looming in the future.

As CV has matured, its applicability and utility to other fields has become more apparent and important. AI and ML techniques already have widespread use in protein folding [60], high energy particle physics (such as the TrackML Particle Tracking Challenge from CERN [117] [5] [116]), and new, emerging fields such as self-driving cars, robotics, [142] and cybersecurity [15]. However, CV research is often done on synthetic datasets without concrete, tangible applications and top-ranked conferences in CV (e.g.

ICCV [1], ECCV [2] and CVPR [3]), all have a strong disposition towards pure CV work. In general, the community seem to value basic research over applied and/or interdisciplinary work. Although basic research is of critical importance, applied research appears to be comparatively neglected. This is not a problem unique to CV, and is recognized as a barrier to interdisciplinary work in scientific and medical fields, where students are often actively encouraged to specialize in narrow disciplines [40].

Although CV is intrinsically interesting, it is important to address real-world problems with it. Increased automation is projected to engender high degrees of social and economic displacement and disruption [9] [14]. However, automation also holds promise to enhance productivity and open new and transformative solutions to myriad large-scale problems (such as in law [129] or medicine [3]). It is important to use AI for ethical means to address problems of societal relevance. Applications that enhance the state of human knowledge and/or aid in solving societal and ecological problems should be encouraged. In this work, we address the problem of automated video analysis for underwater beamtrawl videos of Oregon coast benthos by proposing a new visual tracking dataset, Newport Hydrographic Groundfish, or **NHFish**. The goal of automated processing is to extract higher-order information which can subsequently be used to make ecological inferences correlating environmental conditions to observed groundfish behavior. The foundation of such analysis centers around temporal data association (tracking-by-detection): given a set of fish detections for each frame, the goal is to generate a set of fish tracks by assigning a common label to each detection through time, thereby forming a temporally consistent sequence with a common identity. This enables characterizing fish behavior, as well as localized fish population counts, which can be used for better stock assessments and scientific understanding of fish behavior and ecosystem conditions. However, this sort of automated underwater video analysis of partially-buried and visually non-distinctive groundfish in artificially turbid waters is a challenging task, and the beamtrawl videos are more difficult than comparable state of the art video tracking datasets. However, if successful algorithms prove to be more accurate than current stock assessment methods, then they will increase the spatial and

[1] International Conference on Computer Vision, https://ieeexplore.ieee.org/servlet/opac?punumber=1000149
[2] European Conference on Computer Vision, https://link.springer.com/conference/eccv
[3] Computer Vision and Pattern Recognition, https://ieeexplore.ieee.org/servlet/opac?punumber=1000147

timescale resolutions of fisheries data, improve fisheries ecosystem understanding and stock assessments, and reduce fishery scientists' data gathering workload. Conversely, as a challenging visual tracking dataset, CV work performed on NHFish may help improve the state of the art in visual tracking methods.

3.1.1 NH-Line Beam Trawl Videos

The NHFish dataset consists of 15 underwater beam trawl video sequences, collected off of the Oregon Coast along the Newport Hydrographic Line (NH-line) [4]. Traditional beam trawls are conducted using a chain and a net; the chain disturbs the benthos, and the net catches a subset of them for subsequent analysis. In our case, a top-down facing camera, spotlight, and laser range-finder are also attached to the beam trawl rig to capture visual data. Each video sequence is of a trawl conducted between two set locations (referred to as a 'tow'). At the end of a tow, the net's contents are analyzed for various parameters, such as the number of each fish population type, size, age, weight, etc. Although the fisheries science community generally just uses the net catch data and not video data, this approach is not without its flaws:

- Since the tow is only inspected at the end of a run, there is no information regarding where along the tow any given fish was caught. As such, it has very coarse spatial resolution, and complicates attempts at correlating environmental condition and fish populations.
- The sampling technique may not collect a representative sample, as it will only count the fish that were caught; this may mean that net catch tows have a bias towards counting either very young or very old fish, or ones in poor health. Conversely, healthy fish or more agile species may evade the net, and thus remain uncounted.
- The trawls involve fish mortality (i.e. caught fish are frozen and dissected); although this allows more in-depth data collection, it is invasive and disruptive for the affected environment.

[4]https://catalog.data.gov/dataset/newport-hydrographic-line-newport-hydrograhic-line

- The uncertainty in catch accuracy is not quantifiable; there is no means to quantify uncertainty regarding how representative or accurate of a sampling it is.
- The sampling is very labor-intensive and expensive, and requires fisheries scientists on a boat at sea to conduct and sift through the net catch.

In contrast, the beam trawl video-based technique addresses some of the shortcomings of traditional sampling methods, and offers potential for future innovations unavailable to existing methods. Specifically, the video-based approach can have fine-grained spatial resolution, allowing for very accurate correlation of fish behavior and environmental conditions, and is posed to benefit from future technical improvements from automated analysis techniques. Benefits of the video analysis method include:

- Analysis of fish behavior: net trawls do not provide any insight into fish behavior or actions in the water. This is an important information source for developing a richer understanding of fisheries ecosystems and behavior dynamics.
- Fine-grained spatial resolution: analysis can correlate boat GPS with video data time-stamps to get accurate location information (e.g. longitude / latitude) for any given fish in the video.
- Fine-grained temporal resolution: analysis can provide real-time information regarding fish populations in an area. It is conceivable that given sufficient video-coverage and sophisticated processing techniques, real-time automated analysis of ocean conditions and fish levels could be computed, enabling more accurate and proactive fisheries understanding and management.
- Unbiased sampling: since no fish are excluded (although visually non-distinctive fish are challenging to detect and track, this is something that will likely see sustained and consistent future improvements). Thus, even fish that escape the trawl net can be counted.
- Quantifiable uncertainty: by using labeled datasets of the beam trawl videos, algorithmic performance can be computed, giving a quantifiable measure of uncertainty regarding automated analysis results.

Relying solely on video sampling does have a few shortcomings compared to traditional methods. However, we believe many of these shortcomings are surmountable by improving CV techniques, specifically issues one through four below. Our hope is that by generating the NHFish dataset and making it available to the broader community, we will encourage further work on these tasks.

1. Inferring exact fish dimensions can be challenging and noisy, and fish subspecies can be difficult depending on the available view(s) of the fish.

2. Performing accurate detection and tracking is currently challenging.

3. Population surveys done by visual tracking can potentially double-count fish that exit and re-enter the scene (it is ambiguous whether any given fish is a new, unique fish or a repeat fish). This is partially ameliorated by the fact that fish generally escape away from the trawl chain.

4. Juvenile (i.e. very small) fish are challenging to detect and track; the video analysis approach is presently infeasible on fish < 10 cm in length.

5. Video analysis does not allow for inspecting the fish in detail (i.e. cannot dissect, and thus cannot infer diet, health, etc).

6. Video quality suffers in certain months when visibility is poor (e.g. winter storms can cause turbid conditions, some days in the spring or summer have algal blooms). In these cases, the videos are mostly occluded by marine debris and are non-informative. See fig. 3.1 for an example of a discarded sequence on account of poor visibility.

As such, we do not currently propose video beam trawl data to be a wholesale replacement for net trawl data, but rather act as a complimentary source of information offering faster turnaround times, broader sampling ranges, shorter temporal timescales, and a means to infer certain information unavailable to net catch. This will help provide more accurate and unbiased fish counts and assist in quantifying both the number of fish and the uncertainty regarding the net catches. As analysis approaches and video capture technologies improve, video-based beam trawl sampling may largely supplant net-based ones, and net catches will be infrequently performed, i.e. when certain information such

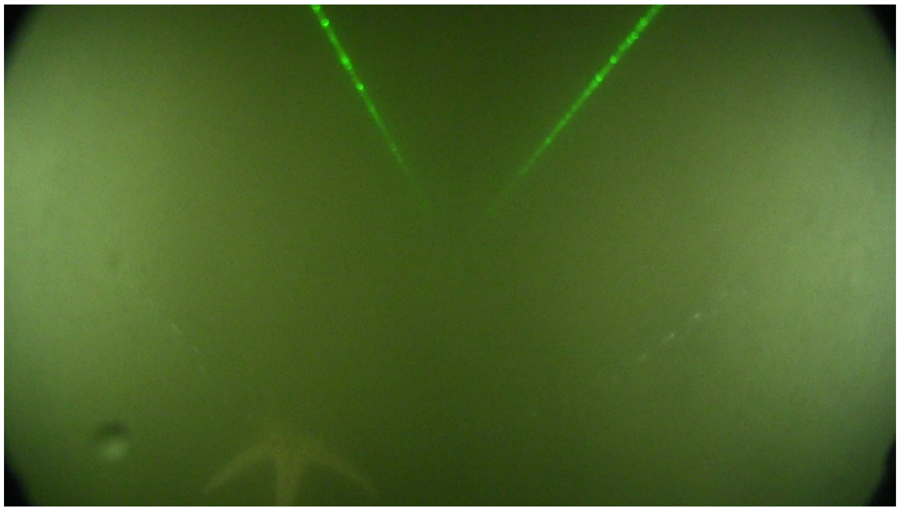


Figure 3.1: A frame from a trawl on July 11th, 2014 – the water for this trawl was too turbid to be used for analysis

as fish diet or tissue samples are required. However, this will likely be a gradual process, and is contingent on sufficiently powerful analytic techniques being developed. Although the beam trawl video analysis method is not without its flaws, it is well-posed to benefit from future technological improvements, such as camera hardware and video analysis techniques, and could enable transformative changes in fisheries science and management. The NHFish dataset will encourage further work from the CV community on problems pertinent to accurate underwater fisheries analysis, thereby improving automated video processing techniques available to fisheries science.

3.1.2 Machine Learning and Ecological Inference

Visual tracking is defined by [128] as "the analysis of video sequences for the purpose of establishing the location of the target over a sequence of frames (time)." Visual tracking is an essential component in video analysis, as it is a primary method for integrating temporal information; algorithms devoid of tracking treat each frame as being independent. Tracking specifically links detections through time into a construct called a 'track', consisting of detections that belong to a single object through a temporally sequential collection of frames. There are multiple formulations of tracking; one of the most common is tracking-by-detection, where one or more detection algorithms provide noisy bounding boxes of potential salient objects in the scene, and the tracker builds tracks from the proposed detections. This is the formulation we use for the beam trawl video analysis problem, and we present NHFish, our challenging beam trawl video dataset as a tracking-by-detection dataset in a similar vein to the MOT dataset [91][73]. The beam trawl videos fit the multi-object tracking paradigm, but feature some new and interesting challenges not seen in contemporary tracking datasets. For example:

- The data has many visual distractors, some of which are shown in fig. 3.11.
- The target objects (flatfish) are naturally camouflaged in benthic environments, as seen in fig. 3.13.
- There can be non-linear camera and object motion, out-of-plane rotation, significant scale variation, and unexpected visual occlusions (such as water droplets on the camera lens, as in fig. 3.7).
- Flatfish can exhibit significant appearance deformation fig. 3.6,
- Flatfish have relatively homogeneous object appearances and can interact with each other, as in fig. 3.10.

As such, motion cues and temporal continuity become important for building accurate tracks, and appearance models may be less useful. In addition, unlike most tracking datasets, this one does not feature humans as a detection target, reducing the utility of person-specific methods such as Deformable Parts Model (DPM) [35] and potentially motivating novel solutions. More details regarding the dataset are presented in section 3.4.

Despite the apparent difficulty of the beam trawl dataset, there is much benefit to be gleaned from automating its analysis. Past attempts at using video beam trawl data for scientific research and ecological inquiry have been limited by the high degree of manual effort by scientists required to analyze the videos. To date, there has been one published paper [133] using the NH-Line beam trawl data, despite it having been collected for over two decades. This is because performing manual analysis requires measuring characteristics of the fish from the video frames by hand, sifting through the video frame-by-frame. In addition, this sort of activity can only be done by skilled marine ecologists, so the labor burden is especially onerous. Being able to perform video analysis tasks without manual work by scientists would greatly expedite the extraction of information, and would open the door to expanded video collection and analysis. The effects this could have on marine science is not to be underestimated; in the words of Shafait et al. [123], automating fisheries video data would "transform marine science."

We hope that by making the video dataset available, fisheries science and management will benefit by having greater attention from the CV community. Although there is considerable interest in performing automated video analysis in unconstrained underwater environments, the state of the art methods employed in ecology often lag behind those proposed in the CV community. By having a dataset that straddles both, we hope that new CV approaches to the challenging beam trawl video data will be proposed and the integration of pure CV and applied ecological work will become closer.

3.1.2.1 Datasets as Boundary Objects

Datasets can act as catalysts for interdisciplinary work, lowering the barriers to working on specific tasks. On a more philosophical level, they can also serve as boundary objects [147] for interdisciplinary projects. A boundary object is a construct that collaborators from diverse disciplines can understand, and which often serve as the linchpins of collaborative projects. Different disciplines often have insular conventions, with terminologies and methodologies that are often foreign to people outside of that field. Boundary objects are important for spanning these differences and creating common ground for collaborators. Datasets make for natural boundary objects, as they are more tangible, and are often the basis for interdisciplinary collaboration to begin with. This is especially true in combining fisheries science and CV – the NHFish dataset served as a boundary object for our interdisciplinary OSU NSF NRT team, and will likely continue to do so in future collaborative work between fisheries science and CV. We believe that leveraging datasets to encourage interdisciplinary work is an area that will see increasing attention in the future, especially when AI-related fields are involved.

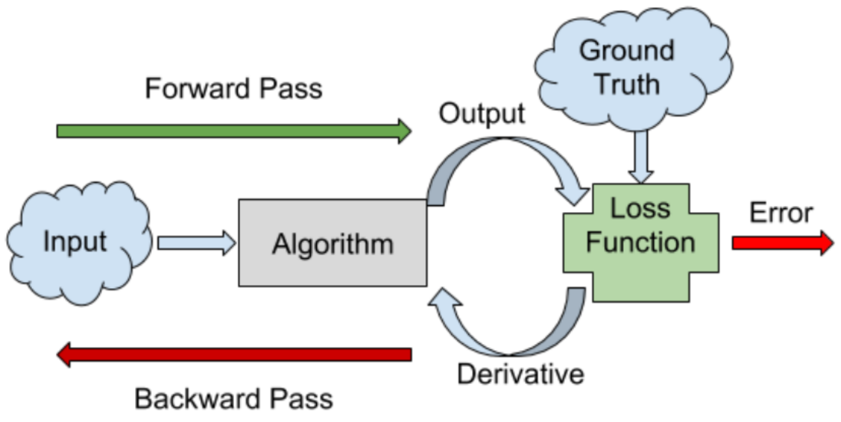


Figure 3.2: Supervised machine learning algorithms have two passes; the forward pass, where the algorithm predicts an output based on an input, and the backward pass, which computes how wrong the forward pass prediction was (via a loss function and ground truth). The algorithm weights are then adjusted to make the algorithm output predictions in the future that will minimize the loss function.

3.2 Datasets and Machine Learning

Data lies at the heart of ML, as it is the information that the machine is meant to learn from. However, in the supervised learning paradigm, the raw data itself is insufficient; there must also be the desired result of any processing. This serves to both define the task to be performed, and provides the implicit relationships between the input and output data samples. While the network's parameters are directly updated via gradient descent, which is a function of the loss, the loss is a function of the ground truth and the network predictions (see fig. 3.2). Thus, the dataset's ground truth is the most important part of any supervised algorithm's training.

On a more philosophical level, datasets are a means to distill a specific task to be performed. Any number of tasks can be performed on a piece of data; within the infinite set of potential tasks, the dataset is what defines what the desired outcome should be. For example, an image can be used as input to myriad processing routines; common tasks include classification (stating what is in the scene), detection (localizing elements within the scene), segmentation (pixel-wise localization), among others. The dataset labels determine the type of task to be performed; classification would consist of categories per image, detection would have zero or more bounding boxes, segmentation would have pixel-wise mask images. Given a different set of labels, we can define a different task on the same input data – the ground truth labels determine both which parts of the data is important, and what tasks the algorithm learns to perform.

On a more fundamental level, datasets are required for ML algorithms to learn. The "No Free Lunch" theorem (NFLT) (for supervised learning) states that no learning algorithm outperforms any other, when evaluated over all possible problems [149] (there are corresponding no free lunch theorems for other domains, notably for optimization). The NFLT implies that if there is no prior information or assumptions made about the data to evaluate an algorithm with, then no algorithm is better than any other; performing well on one task just means performing poorly on a different one. Practically, this means that algorithms need to impose assumptions regarding the data distributions involved to be successful on any given task. Often this is done by assuming structured data to be more common (i.e. data is likely to have lower Kolmogorov complexity) [70]. In general, stronger assumptions are both more informative, and more restrictive. Any such assumption must be made with a specific dataset in mind, and using an inaccurate assumption will hamper an algorithm's capabilities to generalize to the data. In addition, since the dataset specifies what the desired result is for every input (the 'ground truth'), the dataset is directly tied to how well the algorithm performs. If the dataset provides incorrect or noisy labels to the algorithm, then the algorithm will learn the incorrect model for the task at hand and will likely generalize poorly. A more insidious problem is when datasets consist of biased data. While algorithms themselves have no inherent biases (and this is often considered a benefit over humans — humans have egos and biases that inhibit objectivity, e.g. there are more favorable parole decisions given by judges early in the day and after lunch [30]), they can inherit biases that have been encoded in the datasets they are trained on. For example, the Microsoft Common Objects in Context dataset (MS COCO) was found by [160] to contain significant gender bias in visual semantic role labeling; even worse, they also found that training algorithms on such datasets can further amplify biases found in datasets. As ML algorithms have entered the commercial sphere, there has been increasing awareness about implicit biases enshrined in datasets. There have been highly visible mistakes in the past few years, such as Google's racist face recognition [26] in 2015 and Uber's self-driving car death

[144] in 2018. There are also concerns regarding less visible or regulated algorithmic decision making, such as automated credit decisions (e.g. FICO scores), or COMPAS (a proprietary predictive software application used to decide which prisoners are eligible for early release) that may have built-in biases, but which are not available to be audited or examined due to intellectual property constraints. This raises ethical concerns around proprietary and/or automated algorithms that make decisions with minimal oversight. It also highlights the profound importance of datasets in the creation of valid models.

Throughout the history of ML, datasets have propelled innovation. Recent advancements in DL have been fueled (in part) by increased dataset sizes [135]. Using even larger data in tandem with higher complexity models holds promise to continue improving representation learning. Specifically, the recent work by [135] found that such transfer learning features improved performance (on vision tasks) logarithmically with respect to the training data volume. As such, it is likely to remain beneficial for performance to continue increasing dataset sizes and expanding model complexity commensurately.

3.2.1 Technical Impacts of Datasets

The process of supervised ML broadly proceeds in three main stages: training, validation, and evaluation. Note that this is a generalization; in practice, more sophistication methods may be used, such as k-folds cross validation.

- **Training**: This stage is where the algorithm learns from data. The implementer defines an algorithm, which consists of learnable weights. These weights are updated during this stage to minimize a defined cost function, often by some variant of backpropagation and gradient descent. In practice, training is an iterative process of computing high-dimensional representations of the data in feature space, fitting decision boundaries to the feature-space samples, and adjusting the learned curve-fitting based on the algorithm loss function (which in turn, is dependent on the dataset's ground truth). In the case of binary classification, the training step would involve adjusting the model weights of the feature extraction layers, and jointly learning a decision boundary (e.g. a manifold in feature space) that separates the positive and negative samples in a way that minimizes the loss (e.g. binary cross-entropy).

- **Validation**: While the training stage provides a measure of performance on the training set, this is not necessarily a good measure of overall performance. Namely, since the model has access to the ground truth, it may end up overfitting to the training data, and thus generalize poorly. The validation stage is a way to get a measure of the model's performance on data outside of the training set. This, in turn, will reveal overfitting, especially if hyperparameters are being adjusted.

- **Evaluation**: While the validation set is meant to give an objective view of generalizability of the learned algorithm, it is often used for hyperparameter validation, which after multiple such iterations, can end up causing the algorithm to also overfit to the validation set. The true testing set is one that should be unseen during algorithm development and training. Functionally, we take the learned feature extraction model and learned decision boundary, and apply it to the new data samples; the assumption is that if the model has fit the training data accurately, and the training and testing data are from the same generating distribution, that the model will perform well on the testing data as well. Conversely, if the training and testing data diverge too much or the model unfit or overfit to the training data, then it may suffer poor performance on the testing data.

3.2.1.1 Overfitting and Underfitting

Algorithms may not generalize well for the task they have been trained on: this can broadly be attributed to the model either overfitting or underfitting the data. The characteristic trait for overfitting is that the model performs well during training, but suffers poor performance during validation and/or evaluation. In contrast, if the model underfits the data, then it will perform poorly during all the stages. The primary goal in ML is not just to train a model, but to train a model that generalizes well, i.e. that performs well on previously unseen data. Overfitting occurs when the model complexity is much higher than that of the data. This allows the model to simply memorize the training data, thus performing very well for only that specific set of data. Functionally, this usually is due to the algorithm learning both the data and the random noise in the training dataset, rather than the true underlying data distribution, as shown in fig. 3.3. As such, the model learns arbitrary, non-informative relationships within the training

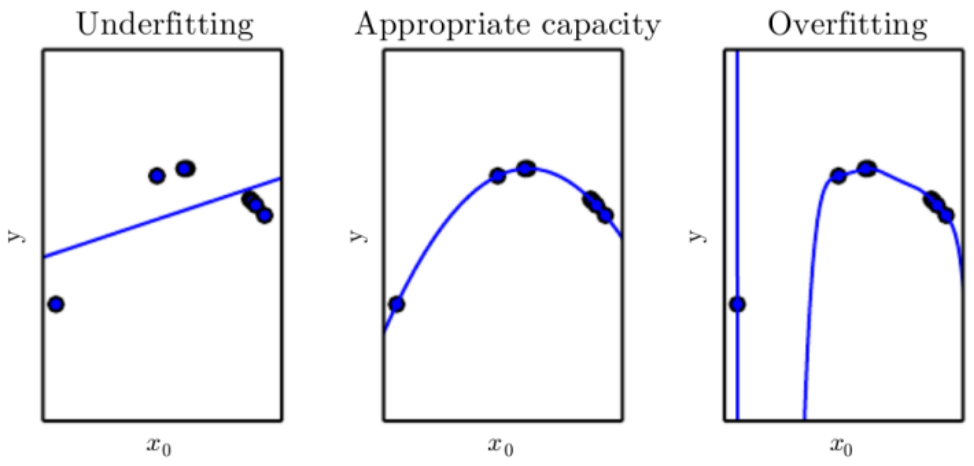


Figure 3.3: Diagram of underfitting, appropriate fitting, and overfitting of a quadratic function of one variable. Figure from [44].

data. Consequently, it invariably performs poorly on anything other than the training set. Underfitting is the converse of overfitting; underfitting occurs when the model is not sufficiently complex to capture the relationships in the data, and is thus unable to adequately model it. For example, trying to fit a high-degree non-linear function with a linear model will always result in poor predictions. No amount of additional training data will help with this fig. 3.3.

On a fundamental level, ML algorithms are tailored to the data they will be evaluated on. Algorithms that are entirely agnostic of their target data will perform poorly based on the NFLT, so it is important to create specific datasets and write algorithm architectures suitable for the target data. If the algorithm model is too complex relative to the data, then it runs the risk of overfitting; if the model is not sufficiently expressive for the data, then it will underfit. While these intuitively make sense, there is also a mathematical framework for expressing the mechanisms behind under- and overfitting. Vapnik-Chervonenkis (VC) dimension is a measure of model complexity, and can be used for defining bounds on the generalization error of an algorithm. Specifically, if H is the space of functions learnable by a statistical classification model, then the VC dimension of H define loose quantitative bounds characterizing the bias (underfitting) versus variance (overfitting) behavior of H. VC dimension bounds are distribution independent, while Rademacher complexity allows computing tighter, distribution-dependent bounds. Namely, it defines a the capability of a learner to fit random data, which can be used to derive an upper bound on the learners generalization error based its training error [162]. However, explicitly calculating these bounds are generally impractical, so in practice more empirical methods are used (specifically, comparing the observed training and testing error). However, VC dimension and Rademacher complexity are important in that they offer a degree of mathematical rigor to the intuition behind the bias-variance tradeoff.

3.2.2 Institutional Impacts of Datasets

In addition to being key components in the technical performance of algorithms, datasets also have deeper, more profound impacts on shaping the types of problems that the CV and ML communities focus on. There is a fundamental interplay between datasets and technological innovation and focus. Once a dataset is established, it becomes a benchmark that algorithms are evaluated against, and concentrates research efforts onto the problem(s) posed by the datasets. One of the most successful examples is ImageNet, which arguably launched the DL era and presided over multiple substantial gains in image classification accuracy and DL innovation. Datasets also offer an opportunity for establishing standardized benchmarks for comparing new methods to. There are often leaderboards [5] [6] [7] associated with each dataset that offer "objective measures of performance and therefore are important guides for research." [82]

While datasets are of fundamental importance to ML, they can be time- and resource-intensive to create. At its inception in 2009, ImageNet consisted of 3.2 million labeled images [31]. As of 2014, it had over 14 million images [119]. The JFT-300M dataset used in [135] consists of over 300 million (weakly annotated) images, and calls for larger datasets still. The cost of creating large, modern datasets can be prohibitive, and is contingent on the task defined by the dataset. Labeling classification tasks are comparatively simple, while video segmentation is notoriously difficult. For example, Cityscapes is a modern semantic segmentation dataset with high-quality pixel-wise instance-level

[5]DAVIS: https://competitions.codalab.org/competitions/16526#results
[6]MOT: https://motchallenge.net/results/MOT17
[7]MS COCO: http://cocodataset.org/#detection-leaderboard

semantic labels. In creating the dataset, "[a]nnotation and quality control required more than 1.5 h on average for a single image." [25] Cityscape has 5,000 high quality frames and 20,000 frames with coarser annotations. Similarly, MS COCO [82] is a modern object instance detection and segmentation dataset, consisting of roughly 2.5 million images. The authors report that when using Amazon Mechanical Turk, the image labeling process took roughly 85,000 worker hours to complete. As such, it can be prohibitively expensive to create datasets of sufficient scale for modern DL methods. In the DL era, large labeled datasets are crucial for training deep models and achieving satisfactory performance. Problems that lack suitable datasets have a higher opportunity cost to work on, and inevitably, see less attention from the CV community. While the presence of a dataset can promote research in its domain, academic and/or limited-budget institutions may not have the resources to create new datasets; as such, research efforts on topics lacking suitable datasets are inhibited. In keeping, if a suitable dataset does exist for a task (and made freely available), then the barrier to working on that task is lowered and it will receive more attention. Thus, the landscape of datasets shapes the work being done.

Datasets do not exist in a vacuum, and are generally created around problems that entities find relevant. Often this comes from a company (e.g. Cityscapes was funded in part by Daimler's autonomous driving Research and Development unit) for a specific task important to their business interests, and there are numerous large proprietary datasets. However, datasets are not strictly created for commercial purposes; many notable datasets such as ImageNet and PASCAL VOC originated from academic institutions, as they filled dataset gaps in areas of significant interest to the CV community. Thus, there is a cyclical relationship fig. 3.4 between the interests of the community and the datasets in existence: problems that researchers and/or institutions are interested in will inevitably have datasets created about them, and people will work on problems that have pre-existing datasets. This has the effect of lowering the barrier to working on problems with datasets already, and raising the bar for working on problems that have no such datasets. However, there are still many problems for which no large-scale dataset exists. More concretely, our project is concerned (in part) with performing tracking-by-detection; however, we are examining Oregon nearshore benthos, and the beam trawl environment is unique. As such, although our project falls within a common CV task, algorithms trained on existing tracking datasets will not necessarily generalize well to

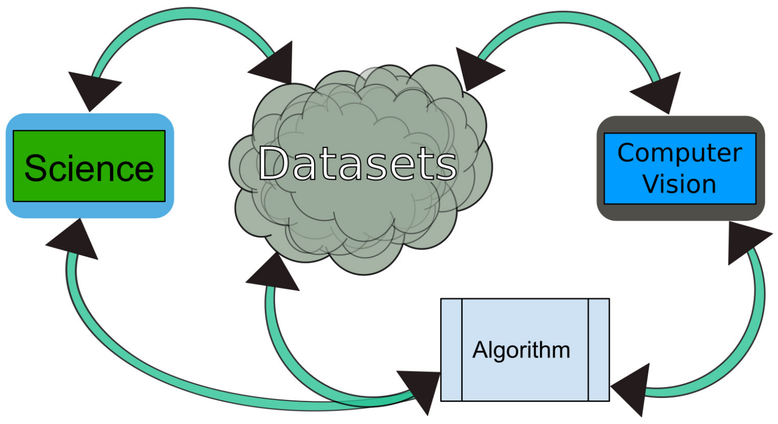


Figure 3.4: Tasks with one or more datasets have a lower barrier to working on them; tasks without datasets have a higher one. Problems that are important usually have datasets created for them, while datasets that are unimportant do not. Creating a dataset for a problem can encourage further research on the task. The sciences can encourage collaboration by creating datasets relevant to their problem domains.

the beam trawl videos (due to how divergent it is from existing datasets). Hence, we built our own dataset, NHFish. Although this was a significant undertaking, we hope it will prove beneficial in the future by encouraging others in the CV community to work with the beam trawl video data.

Our labeled NHFish dataset is the latest in a long line of video tracking datasets. Video tracking is one of the fundamental tasks in video analysis pipelines, as it allows for forming temporally consistent sequence information between frames in a video. Most recently, the VOT [66] and MOT [91] [73] datasets have taken the mantle as the state of the art single and multi-object video tracking (single camera) datasets, respectively. In addition, any video segmentation dataset can also be used as a tracking dataset, but since the labeling burden for segmentation is higher, segmentation datasets are generally smaller. In addition, there are multi-camera tracking datasets, such as PETS [100], which generally focus more on the video surveillance applications. Despite the array of other visual tracking datasets, the beam trawl video data poses some unique challenges. Namely, all of the data is captured underwater, with specific scene elements common to all beam trawl videos. More specifically, the lighting is supplied by a spotlight mounted on the camera rig, making the center regions bright and the four corners of the frame comparatively dark. Likewise, shadows are cast in a consistent manner, and the camera is a relatively consistent distance from the ocean floor. There are two laser beams (mounted on the beam trawl rig) projected into the middle of the frame for range-finding. The Oregon benthic habitats sampled are all sandy-bottomed, with colors largely in hues of brown and green. Also, the target objects for tracking are flatfish, which can both burrow into the sand, and have evolved to be well camouflaged against the ocean floor. This makes appearance features less informative and motion cues more important. Similarly, there are frequent and disruptive visual distractors, since the sampling method relies on the trawl chain chain to disturb the fish in order to induce motion. These factors make our video collection distinctive and challenging in the video tracking dataset landscape.

3.3 Related Works

3.3.1 Applied Computer Vision

CV has a long history of applied work in the natural sciences. [145] provides a comprehensive survey of CV for plant species identification, [108] gives an overview of CV for medical imaging, and [65] describes CV use in natural disaster warning and detection systems. There are multiple science-oriented CV tool boxes, such as WildBox [8] for animal counting and identification, TMARKER [120] for cell counting and staining analysis, and BioTracker [92] and trackdem [13] for visual tracking of animal populations, among others. In general, CV can reduce manual workloads and increase the scope, power, and insight of natural science inquiry when repetitive visual tasks are required. Visual tracking is especially important for many ecological tasks, as it enables making ecological inferences, such as population counting, individual behaviors, and group dynamics. There are numerous specific works of applied visual tracking, such as honey bees in controlled environments [64] and in the wild [10], elephant detection and tracking [126] [27], and most pertinent to this project, analyzing fish in underwater conditions. For example, [127] used pre-trained CNNs for fish classification on underwater images from Australia and [50] used GMMs for detection, SVMs for classification, and Kalman filters for tracking. [114] used a 3D optical stereo matching approach for fish segmentation. [23] used a deformable parts kernel matching tracker for underwater fish tracking. In addition, there is interest from the regulatory agencies, such as NOAA, regarding applications of automated recognition systems in fishery services [148].

Despite strong interest in underwater fish analysis methods, the primary existing dataset is the SeaCLEF subset of the LifeCLEF dataset [93]. However, it mostly is concerned with classification of tropical fish in less turbid waters, and is a poor fit for our specific task. The Fish4Knowledge project [36] started in 2010, and examined detection and tracking in a large corpus of underwater tropical coral reef fish. Unfortunately, the project is now defunct, the cameras were stationary, and the tropical underwater environment differed significantly from the beam trawl dataset. In keeping, despite the corpus of work on underwater fish video analysis, none of the existing approaches are suitable to our NH-line beam trawl video data. [50]'s reliance on background subtraction is only feasible for stationary cameras, [114] requires a custom multi-camera arrangement, and was performed on data from a controlled (indoor pool) environment. [23] heavily relies on appearance features, which may be of limited utility for Oregon benthos. The closest approach is detailed in [127], but transfer learning is heavily reliant on testing and training datasets being sufficiently close; none of the existing datasets are remotely close to the beam trawl video, so transfer learning performance would suffer. [123] and [56] are two recent works dealing with video detection and tracking in unconstrained underwater environments. [123] performs species recognition in video sequences, but does not address detection and tracking, and uses ImageCLEF for training and evaluation. [56] uses a two-stage graph-based tracking formulation over learned CNN features, calculating tracklets in a shortest-path problem formulation, then groups tracklets with an affinity metric based on the L2-distance of their CNN features. In all, existing work on underwater fish video analysis is promising, but taken by itself, is insufficient for our dataset.

3.3.2 Detection

Recent works in detection and tracking have significantly improved upon the past state of the art methods. PASCAL VOC [34] and MS COCO [82] are datasets with image detection annotations, and have helped drive the field of object detection forward in recent years. Broadly, modern convolutional object detection algorithms can be divided into two groups, one-stage and two-stage detectors. We provide a brief survey of modern detection algorithms to motivate our choices of detection proposal methods for the presented NHFish dataset.

Some of the most successful and popular detection algorithms in recent years have stemmed from R-CNN [42]. Proposed in 2014, it has inspired numerous subsequent methods, and the lineage of R-CNN, Fast R-CNN [41], Faster R-CNN [111], and Mask R-CNN [47] have all been highly successful as two-stage detection algorithms. R-CNN uses selective search [139] to propose $\sim 2K$ differently-sized bounding boxes on each frame, resizing each ROI to a common resolution and running them through a CNN sequentially. Afterward, each region is classified as containing an object or not. Although state of the art in 2014, it was also extremely slow, due to passing each ROI through the CNN individually. Fast R-CNN improved upon R-CNN by computing features over the entire frame once, then proposing bounding boxes in feature space using selective search. Each feature ROI is then pooled to a common resolution, and two outputs are generated: one from a softmax classification layer deciding whether the given box contains an object, and the other from a bounding box regressor to adjust the bounding box coordinates to accurately localize the object in image space. Faster R-CNN builds on insights from Fast R-CNN and replaces selective search with a Region Proposal Network (RPN) to propose bounding boxes over the image CNN features of varying aspect ratios and scales. See fig. 3.5 for the network architecture diagram. Faster R-CNN proved to be extremely successful and highly influential; as of November 2016, "half of the submissions to the COCO object detection server [...] are reported to be based on the Faster R-CNN system in some way." [53] We use Faster R-CNN as one of the detection proposal methods in this work. Mask R-CNN uses Faster R-CNN, but adds another output head to perform segmentation (pixel-wise labeling) rather than just bounding boxes, and adds a more accurate feature pooling method than the ones Fast R-CNN and Faster R-CNN used. These additions pushed Mask R-CNN to be among the most accurate methods for detection (and instance semantic segmentation) on COCO through 2017, and remains highly competitive currently. Unfortunately, Mask R-CNN requires segmentation annotations for offline training, and segmentation masks are expensive and laborious to create. The NHFish video dataset does not have segmentation masks, so evaluating Mask R-CNN as a detection proposal method was not feasible.

Although two-stage detection methods have enjoyed considerable success in recent years, there have also been considerable effort in developing 1-stage detectors. 1-stage detectors are often faster, albeit at the cost of lower performance on average. Overfeat [121]

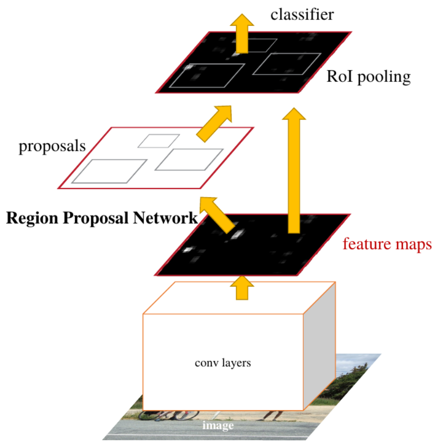


Figure 3.5: Faster R-CNN architecture diagram; we use Faster R-CNN with a ResNet-101 feature extractor as one of the detection proposal networks in the beam trawl video dataset. Figure from [111]

was an early (2013) DL approach for detection, using a CNN to perform classification, localization, and detection using multiscale sliding windows. The predicted bounding boxes are accumulated to build detection confidence measures. YOLO [110] was another impressive 1-stage detector, and formulated detection as a regression task from image space to bounding boxes, enjoying reasonable performance at real-time speeds. SSD [84] improved on YOLO's runtime and accuracy performance by defining a fixed-size set of bounding boxes, predicting likelihoods of each box containing an object, and refining probable boxes for more accurate localizations, all at multiple scales. RetinaNet [81] formulates a novel loss function that rectifies the foreground versus background pixel imbalance (i.e. on average, the majority of pixels in an image belong to scene elements not of interest). The authors couple this loss function (termed 'focal loss') with a relatively simple detection network architecture and achieve state of the art runtime and accuracy performance. Due to its impressive performance, RetinaNet is the other detection proposal method used in this work.

3.4 NHFish Dataset

The NHFish dataset consists of 15 video sequences, with 19,899 annotated bounding boxes and nearly $218K$ high-resolution frames. The beam trawl video data exhibits an array of challenging factors; we discuss them in the following section. To perform the analysis, we borrow some of the nomenclature from the DAVIS dataset. Specifically, DAVIS is a modern state of the art video object segmentation dataset, and consists of 50 high-resolution video sequences with densely annotated ground truth segmentations. For comparison, the DAVIS-2016 [105] dataset has a compendium of challenging factors in video data; the beam trawl data exhibits nearly all of them (except *Heterogeneus Object* (HO) — rather, the general lack of color variation in the beam trawl videos is actually one of the primary challenges). For reference, a (lightly edited) list of the challenging attributes from DAVIS are duplicated here:

- (**BC**) Background Clutter. The back- and foreground regions around the object boundaries have similar colors (χ^2 over histograms). See e.g. fig. 3.13.
- (**DEF**) Deformation. Object undergoes complex, non-rigid deformations. See fig. 3.6 and fig. 3.9.
- (**MB**) Motion Blur. Object has fuzzy boundaries due to fast motion.
- (**FM**) Fast-Motion. The average, per-frame object motion, computed as centroids Euclidean distance, is larger than $\tau_{fm} = 20$ pixels.
- (**LR**) Low Resolution. The ratio between the average object bounding-box area and the image area is smaller than $t_{lr} = 0.1$.
- (**OCC**) Occlusion. Object becomes partially or fully occluded. See fig. 3.13.
- (**OV**) Out-of-view. Object is partially clipped by the image boundaries. Fish generally escape to the borders of the frame (away from the trawl chain), so most fish in the dataset exhibit OV at some point.
- (**SV**) Scale-Variation. The area ratio among any pair of bounding-boxes enclosing the target object is smaller than $\tau_{sv} = 0.5$.

- (**AC**) Appearance Change. Noticeable appearance variation, due to illumination changes and relative camera-object rotation. See fig. 3.6.
- (**EA**) Edge Ambiguity. Unreliable edge detection. The average ground truth edge probability (using [33]) is smaller than $\tau_e = 0.5$.
- (**CS**) Camera-Shake. Footage displays non-negligible vibrations.
- (**HO**) Heterogeneus Object. Object regions have distinct colors; some visual distractors are colorful compared to flatfish. See fig. 3.11.
- (**IO**) Interacting Objects. The target object is an ensemble of multiple, spatially-connected objects. See fig. 3.10.
- (**DB**) Dynamic Background. Background regions move or deform; in our case, the dust cloud from the trawl chain is ever-present. fig. 3.19 and fig. 3.8.
- (**SC**) Shape Complexity. The object has complex boundaries such as thin parts and holes.

In addition to exhibiting the above difficulties, the beam trawl dataset introduces some new challenges of its own. Namely, the video sequences are often characterized by a lack of color variation: most flora and fauna along the ocean floor is a shade of brown, green, or grey (see fig. 3.8). In addition, we are primarily surveying flatfish, which have evolved natural camouflage with respect to the ocean floor, and are not colorful or visually distinctive, unlike tropical fish. As such, appearance-based features, which are typically the most discriminative, are of limited use. In addition, there are some other problems that do not show up in DAVIS:

- Non-linear and out-of-plane camera and object motion. Although related to **CS**, this concerns camera motion, as well as depth motion (i.e. the camera rig bobs up and down as the boat moves, potentially inducing significant scale variation).
- Environmental camera lens occlusion / obscuring. In some video samples, water enters the camera housing, creating water droplets on the camera lens and distorts parts of the frame. See fig. 3.7.

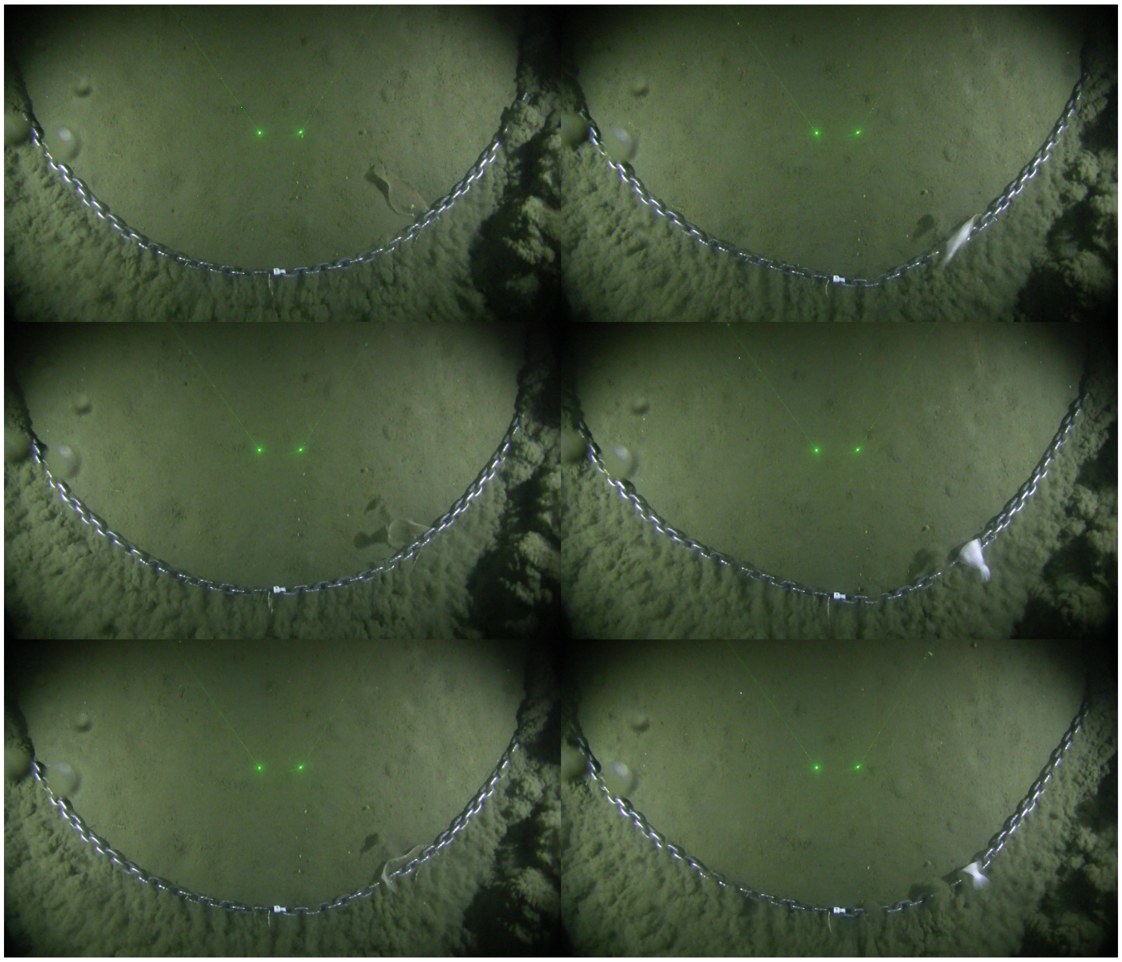


Figure 3.6: Six consecutive frames ($\sim \frac{1}{4}$ second) showing the significant appearance shift flatfish can undergo (frames in order of top-to-bottom, left-to-right), in this case due to the fish being run over by the trawl chain and flipping over.

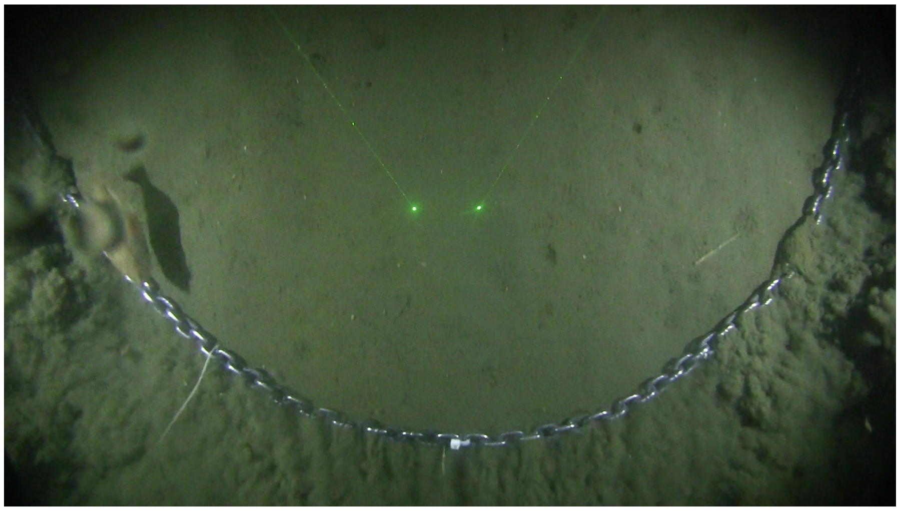


Figure 3.7: Example of a water droplet on the camera lens housing obscuring the left part of the frame around the trawl chain region; note the swimming flatfish partially occluded / distorted by the droplet.

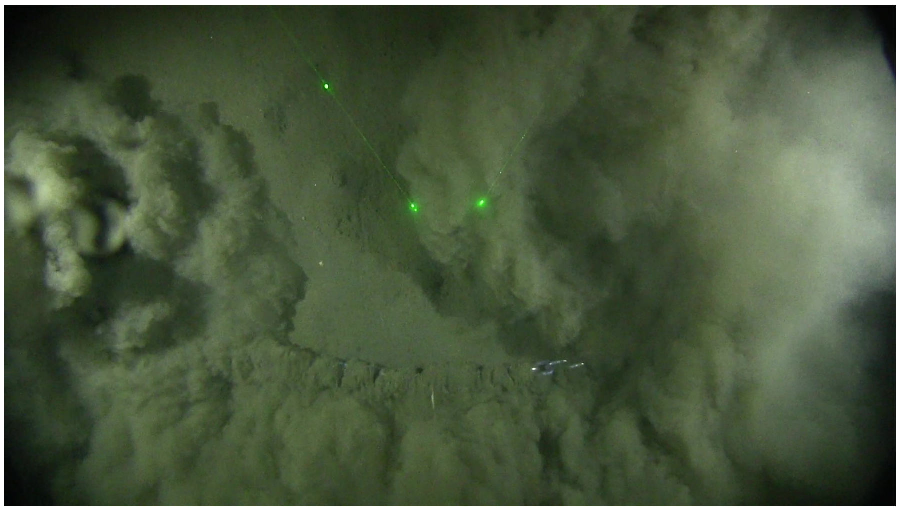


Figure 3.8: Example frame exhibiting how prominent the dust cloud can be. Depending on the boat speed, elevation above the ocean floor, and type of sandy bottom, the dust cloud will be more or less obtrusive. In this case, the boat slowed down, so the dust cloud overtook the trawl chain and obscured the majority of the scene.

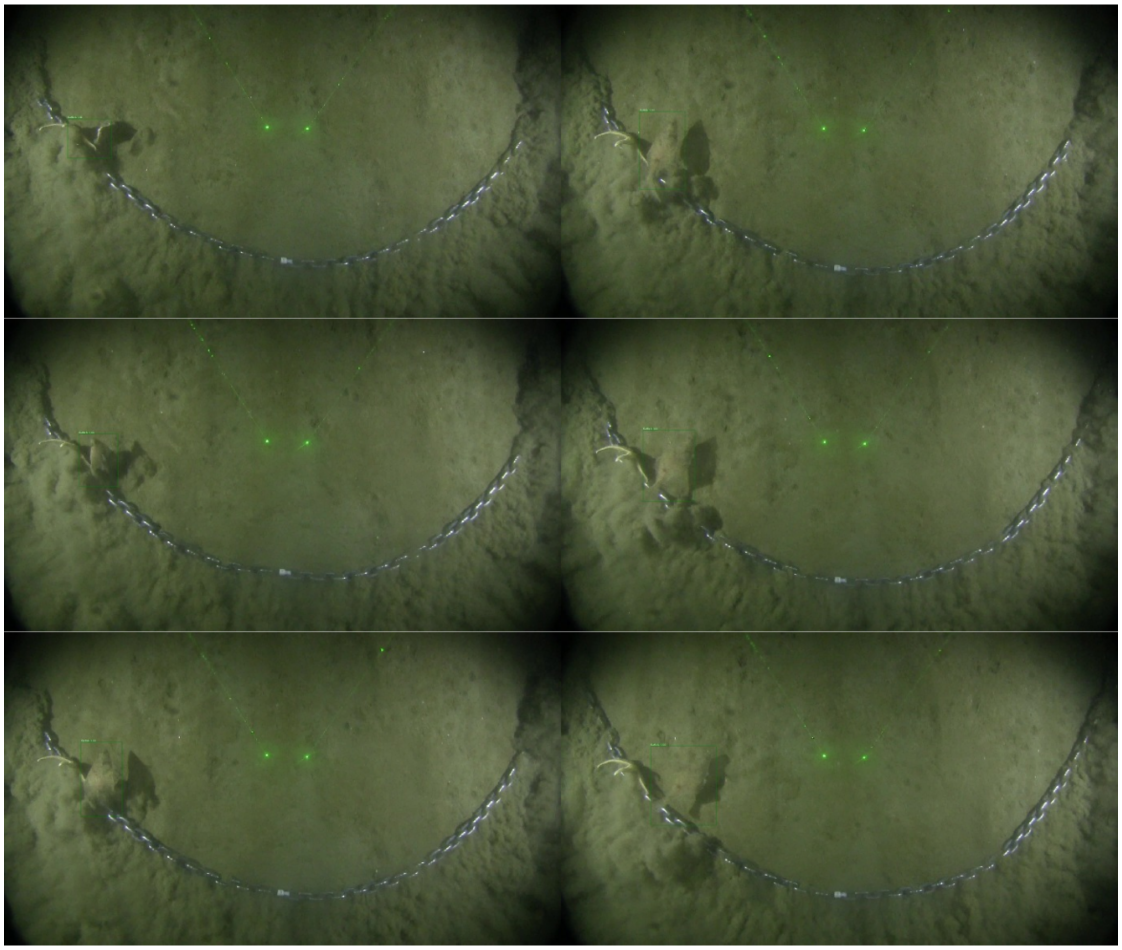


Figure 3.9: Sequence of RetinaNet detection proposals for a fish emerging from being run over by the trawl chain (as well as an unfortunate starfish caught on the trawl chain). Sequence proceeds top-down, left-to-right.

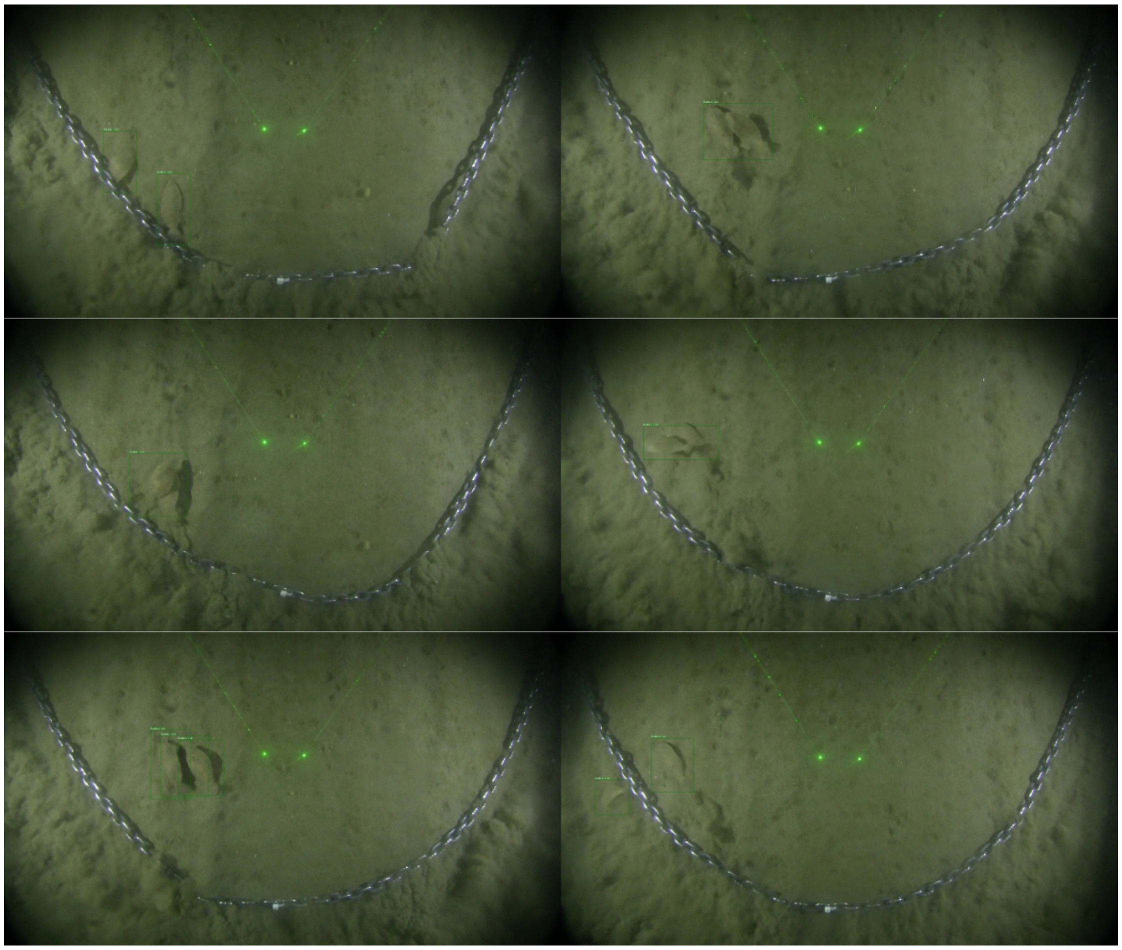


Figure 3.10: RetinaNet detection proposals of a sequence of frames showing two fish occluding each other, which leads to bounding boxes merging and diverging. This is a challenging case for a tracker to maintain consistent track IDs. Sequence proceeds top-down, left to right

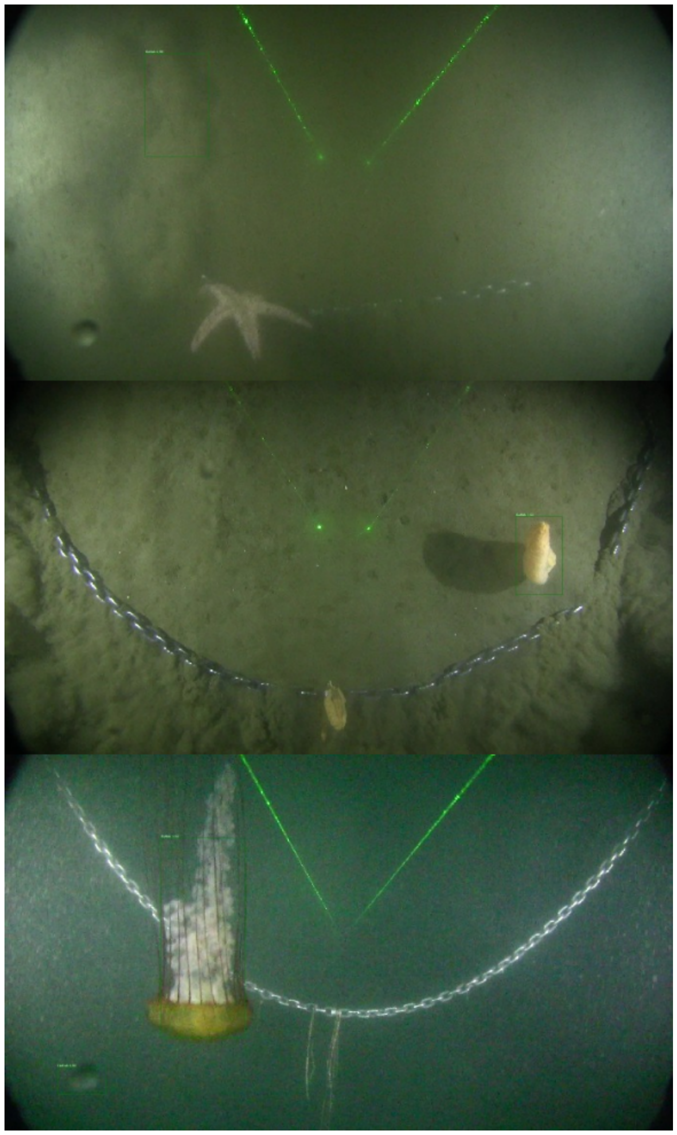


Figure 3.11: Images showing false positive detections due to some of the challenging conditions found in NHFish. The top frame depicts a false positive on the dust cloud in turbid conditions, the middle frame is a false positive on account of visual distractors (specifically, two sea pens). The bottom frame is a false positives on a jellyfish, as well as on a water droplet on the camera lens housing in the lower left corner.

Even within the various categories shared with DAVIS, the beam trawl video data is significantly more difficult than what is present in DAVIS (most notably, *BC*, *OCC*, *OV*, *AC*, and *DB*), which is the current state of the art video object segmentation dataset. While segmentation would allow semantically richer ecological inferences, in light of the difficulties posed by the beam trawl video data, we elected to opt for a slightly simpler CV task, of performing detection and tracking. Detection and tracking allows inferring information about fish trajectories and coarse behavior information, as well as computing population counts. These tasks are all both valuable and, if performed by manually by a human, work-intensive undertakings.

The NHFish dataset is relatively unique in the multi-object tracking space. Most efforts in tracking center around detecting and tracking humans, such as MOT [91] [73], Cityscapes [25], and PETS [100]. This is partially due to most practical applications of video tracking dealing primarily with human-centric activities, e.g. self-driving cars and surveillance applications. Hence, there has been considerable interest in detection and tracking of humans, as well as tertiary semantic categories, such as cars and bicycles. However, these are all a far cry from flatfish in turbid waters. Similarly, contemporary MOT datasets consist of many foreground objects in cluttered scenes; in contrast, the beam trawl video has relatively few objects in the scene concurrently, and can have long stretches of video in which there are no flatfish at all. However, there are many persistent visual distractors (notably, the dust cloud produced by the trawl chain), which can often resemble salient objects. As such, there is a strong class imbalance in the dataset, with relatively few true positives, and many potential (and convincing) false positives. In addition, flatfish are naturally camouflaged, and often have a predilection for partially burying themselves in the sand, as seen in fig. 3.13. This makes detection a very challenging task, as the target objects are, by evolution, visually non-distinctive at best, and not visible at worst. This is also why the trawl chain is of importance: the chain induces motion in the fish, and fish motion plays a large role in being able to identify them in the scene. By the same token, the trawl chain is the cause of the dust cloud, which is the primary visual distractor. In addition, an unfortunate side effect is that there is a strong prior for the fish becoming visible at the border of the dust cloud. This is because the fish will often be partially buried in the sand until it makes contact with the chain, and only begin swimming at that point. As such, fish are probabilistically more likely to appear in or around the visual distractors in the scene, which further complicates accurately detecting and tracking them (see fig. 3.13).

3.4.1 Dataset Technical Aspects

Although our main contribution is the labeled NHFish video dataset, we also present the preliminary results of applying the detection proposal methods to the dataset in section 3.6, in order to assess the feasibility of current methods in the beam trawl domain. Since we believe the dataset would primarily be of interest to the CV community as a tracking dataset, we formulate it from a tracking-by-detection perspective. We draw inspiration from the MOT dataset [91] [73], the current primary multi-object tracking dataset, in pre-computing detection proposals and including them as part of the dataset. MOT consists of 14 sequences, for a total of 33,705 frames and 3,993 distinct tracks, and supplies detection proposals from three detection methods. We adopt a tracking-by-detection paradigm for analyzing the beam trawl video, using two standard state of the art detectors, Faster R-CNN [111] and RetinaNet [81]. We use the official open source implementations provide at the Facebook AI Research Detectron repository[8] [43]. The base network models are trained on MS COCO detection, and fine-tuned on the NHFish dataset. The feature extraction backbone networks are ResNet-101 and ResNet-50 [48] for Faster R-CNN and RetinaNet, respectively. Detection proposals are extracted for all of the labeled video frames, and stored to disk as part of the dataset. We format the beam trawl dataset into the COCO JSON format, as it has become a defacto standard for detection tasks and integrates easily with many existing codebases. The filesystem layout is presented in fig. 3.14.

3.4.2 Datasets and Applied Works

We believe that the NHFish dataset is an important step in bringing together CV research and difficult problems of relevance 'in the wild'. More concretely, there are myriad social and environmental problems that could potentially benefit from CV. Already there are many successful examples of applied CV, and there is intense interest in leveraging CV for wider applications. Conversely, CV requires large, challenging, and varied datasets. As CV moves into arenas with tangible impacts, it could benefit from noisier data that better

[8]https://github.com/facebookresearch/Detectron

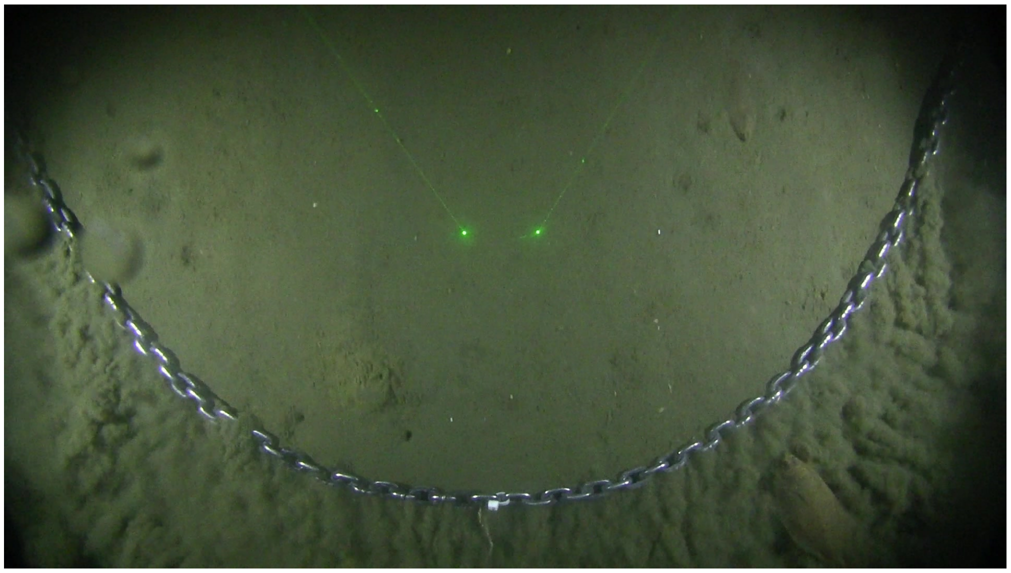


Figure 3.12: Two fish in the same frame, one stationary in the sand (upper right corner), and one lurking and partially occluded in the dust cloud (lower right corner).

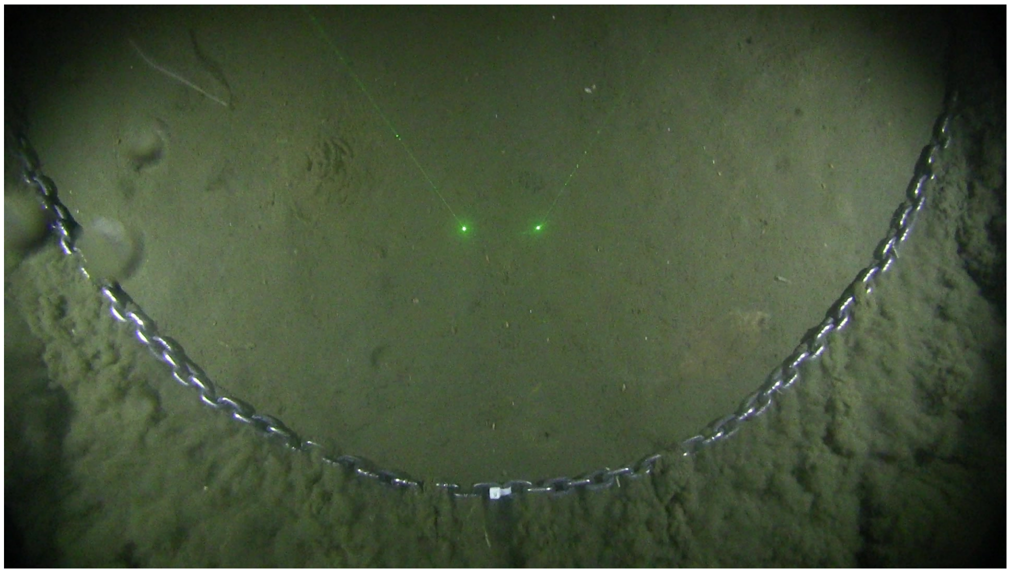
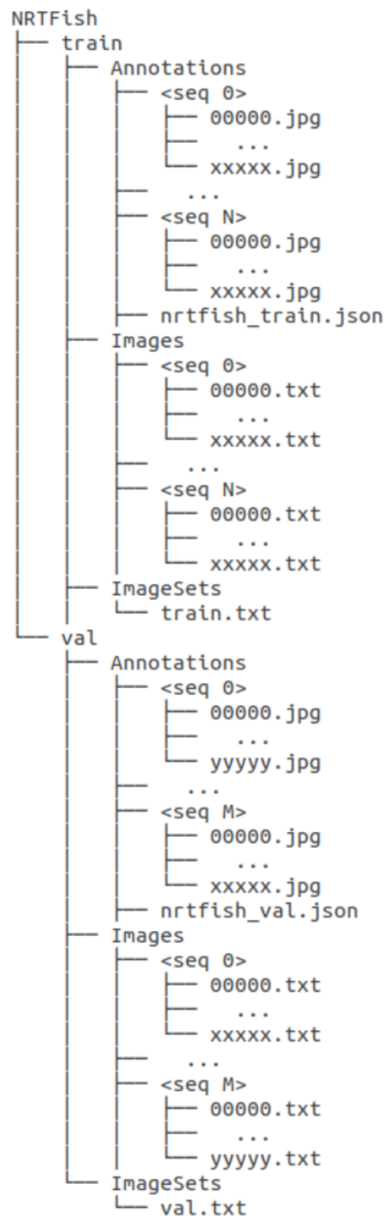


Figure 3.13: Another example of a flatfish partially buried in the sand (on the right side, just above the trawl chain). Note how well the fish blend into the background.

reflects complex, real-world conditions rather than curated datasets. Such datasets would force algorithms to be robust to noise and generalize better to unexpected scenarios found in real data (such as occasional occlusion by a stray crab pot (fig. 3.15) or trawl net, as found sometimes in NHFish). In addition, performance often improves significantly on tasks that have datasets: initial state of the art results on DAVIS-2016 improved from a mIoU of 66.5% by [88] in 2016 to 86.1% by [143] in 2017. Similarly, the 2010 ImageNet ILSVRC winner had a 71.8% top-5 classification accuracy [83], while the 2015 winner had a 96.43% top-5 classification accuracy [48]. Both datasets presided over significant gains in performance for their respective tasks. The presence of a dataset for a specific task can serve to focus the community and drive improvements in the state of the art. As such, it is plausible that creating new and challenging datasets for real-world, applied tasks would both:

1. Encourage research by the CV community on that task.
2. Improve algorithmic performance on that task and encourage novel solutions.

Given that CV can automate repetitive visual tasks, using CV increases the feasible scale of data analysis and scientific research (scaling both up and out). Therefore, it would be beneficial to have tighter integration between CV and the sciences. Since datasets are one of the important drivers of CV research and efforts, using representative data samples from the natural sciences to build datasets can be an effective and mutually beneficial way to couple CV practitioners and scientists. Concretely, CV researchers will have more data available to them, and scientists will have more powerful analytics and processing power at their disposal. Our NHFish dataset is an important work in this regard. Although there have been datasets created from the natural sciences before (e.g. the iNaturalist dataset [140] is a recent example, albeit for image classification and detection only), NHFish is a modern, large-scale, challenging high-resolution video tracking dataset created to address a significant social, environmental, and economic problem. We believe this is a pragmatic but mutually beneficial framework for facilitating future interdisciplinary work between CV specialists and the broader scientific community.



```
NRTFish
├── train
│   ├── Annotations
│   │   ├── <seq 0>
│   │   │   ├── 00000.jpg
│   │   │   ├──   ...
│   │   │   └── xxxxx.jpg
│   │   ├──   ...
│   │   ├── <seq N>
│   │   │   ├── 00000.jpg
│   │   │   ├──   ...
│   │   │   └── xxxxx.jpg
│   │   ├── nrtfish_train.json
│   ├── Images
│   │   ├── <seq 0>
│   │   │   ├── 00000.txt
│   │   │   ├──   ...
│   │   │   └── xxxxx.txt
│   │   ├──   ...
│   │   ├── <seq N>
│   │   │   ├── 00000.txt
│   │   │   ├──   ...
│   │   │   └── xxxxx.txt
│   ├── ImageSets
│   │   └── train.txt
└── val
    ├── Annotations
    │   ├── <seq 0>
    │   │   ├── 00000.jpg
    │   │   ├──   ...
    │   │   └── yyyyy.jpg
    │   ├──   ...
    │   ├── <seq M>
    │   │   ├── 00000.jpg
    │   │   ├──   ...
    │   │   └── xxxxx.jpg
    │   ├── nrtfish_val.json
    ├── Images
    │   ├── <seq 0>
    │   │   ├── 00000.txt
    │   │   ├──   ...
    │   │   └── xxxxx.txt
    │   ├──   ...
    │   ├── <seq M>
    │   │   ├── 00000.txt
    │   │   ├──   ...
    │   │   └── yyyyy.txt
    └── ImageSets
        └── val.txt
```

Figure 3.14: Filesystem layout of NHFish dataset. <seq #> denotes the sequence name, and frames with no annotations have no corresponding annotation file in the Annotations directory.

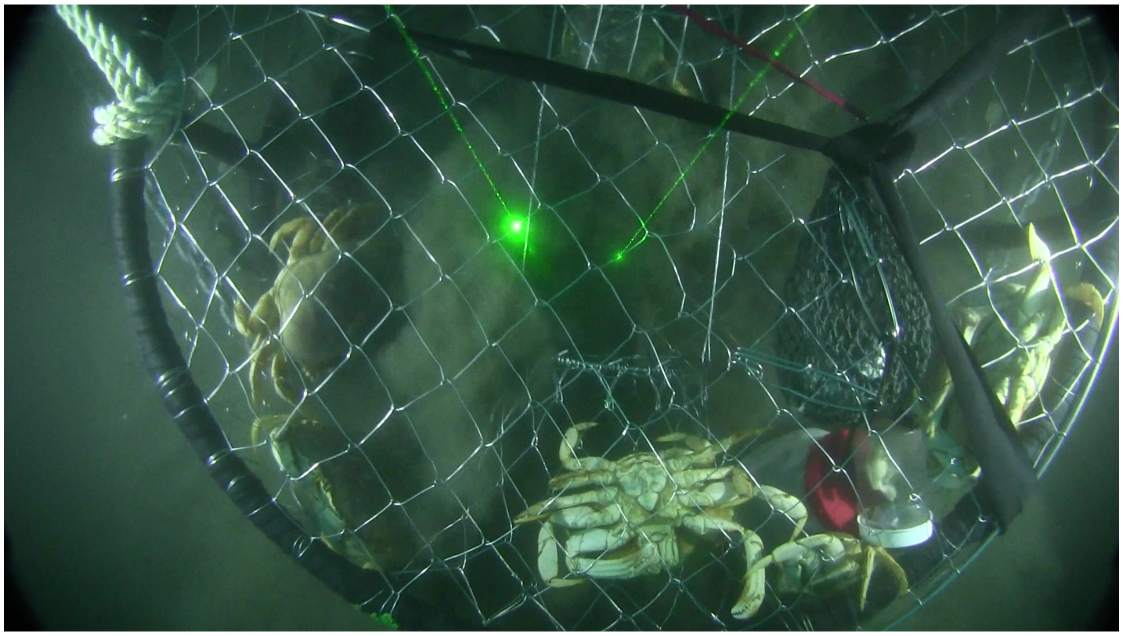


Figure 3.15: Example frame depicting the the sort of unexpected visual distractors that can arise in real-world data. In this case, the trawl chain caught an errant crab pot, which obscured the scene for the remainder of the trawl video.

3.5 Potential Ecological Inferences in Fisheries Science

CV in fisheries science has historically been deployed in largely controlled environments for recognizing dead fish, e.g. [134]. More recent work has attempted to tackle the problem of analyzing live fish in uncontrolled environments, e.g. [123], which is a much harder problem. In the context of CV, live fish analysis tasks generally involve detection and tracking, as in [123], which used a particle filter for tracking, or [146], which used keypoint descriptor tracking for movement pattern analysis. Some works use classification, done on a per-frame basis. [118] used a SVM classifier with deformable template matching, and [109] used sparse low-rank matrix decomposition for foreground extraction, computed deep features over the foreground, and classified fish species per-frame in video with a SVM. [131] performed both classification and tracking, using shape and texture features. [130] proposed a system for detection, tracking, and counting fish in unconstrained underwater video. Although highly related to our task, their work is over a decade old, evaluated solely on tropical fish as part of the now-defunct EcoGrid project, and used CamShift [4] as the tracking algorithm of choice, which has long since been surpassed by other multi-object tracking algorithms.

CV tasks can be broadly grouped into four categories: classification, detection, tracking, and segmentation. Detection and tracking are more challenging than classification, as they involve performing localization (for detection) and integrating temporal information (for tracking), both of which are absent in classification. However, detection and tracking can reveal more information regarding fish behavior, and are important for properly utilizing video data. Segmentation has even more stringent localization requirements than detection, as it involves per-pixel labeling rather than bounding boxes. Each of these fundamental CV tasks facilitate different ecological inferences. For a (non-exhaustive) list:

- Classification: Identifying the presence of specific fish species in an image, identifying habitats.
- Detection: Localizing different fish in an image, counting the number of fish. Prerequisite for performing tracking.
- Tracking: Counting the number of fish in a video sequence, analyzing fish trajectories or response times.

- Segmentation: Analyzing specific fish behaviors, detailed fish sizes and potential interactions.

Tracking is a cornerstone operation for video data processing, as it unifies per-frame information along the temporal axis. Namely, detection and classification tasks are performed on a per-frame basis, but only provide information for that given frame; each frame is implicitly treated as being independent. As such, inferring temporal continuity between frames is not possible. In contrast, tracking associates information computed between frames within a temporal window. As such, without utilizing tracking, the types of information that can be gleaned from video data is limited. For example, computing accurate counts of the number of fish in a video requires both detection and tracking. Detection provides the number of fish in any given frame, but to compute an accurate count through the entire video, it must be possible to discern new fish from old. Concretely, frame t can have 1 fish, and frame $t+1$ can also have 1 fish, but if the fish from frame t exits the scene, and a new fish enter the scene in $t+1$, then the sequence contains 2 distinct fish. If simply performing per-frame detection, it would appear that the sequence only has 1 fish. Differentiating between new and old fish can only be done if the frames are not assumed to be independent, which in turn, necessitates tracking. There are also more insidious issues, such as a fish exiting the field of view, then re-entering at a later date. Identifying these occurrences would require re-identification logic; although person re-identification is a common task (e.g. CUHK03 [78] and Market-1501 [161] are both datasets for person re-identification), doing so on flatfish is more difficult. Flatfish lack of visual distinctiveness, can be within the field of view for a short period of time (some fish in the beam trawl videos are in the scene < 5 frames (i.e. 200ms)), and often are only seen from limited angles and exhibit significant appearance variation based on viewpoint. We expect successful approaches to the NHFish dataset to have to integrate some type of re-identification module to compute accurate population counts.

3.5.1 Fishery Stock Assessment

Fisheries science makes heavy usage of predictive models, and sound fisheries management is heavily reliant on accurate population dynamics predictions [98]. However, counting the number of fish in a given area is a difficult, expensive, and error-prone undertaking. Some species of fish can be elusive, others are naturally camouflaged, and all move around (albeit some more than others). There are annual survey trawls performed along the west coast of the continental US undertaken by NOAA [61] every May through October, and are an essential resource for establishing annual fisheries management policy. Notably, such surveys are mandated by United States law (The *Magnuson-Stevens Fisheries Conservation & Management Act* [2]) for setting fishing quotas. Specifically, the law calls for establishing fishing quotas that admit the "maximum sustainable yield," as determined by "best available science." In the intervening years since its passage in 1976 (and subsequent reauthorizations), this mandate has evolved to require frequent fishery stock assessments, as fish population models heavily rely on population estimates to predict fish population dynamics. Although surveys are frequently performed using net catch data, automated video analysis would greatly reduce fishery scientist workloads and streamline stock assessments. Two of the main impediments to adopting video techniques are the lack of automated analysis techniques and the high amount of labor required to process videos manually [123]. CV has to potential to bridge this gap, and our work and resultant dataset are contributions in this arena.

3.5.1.1 Application: Counting Fish

Although there is a wealth of potential questions to be asked of the beam trawl video (as discussed in detail in section 3.5), we believe the task of fish counting is a good target application for this work. The reason for that is twofold:

1. Computing accurate fish counts is of vital importance to fisheries science and management, as it is tantamount to performing stock assessments.

2. The technical bedrock laid for performing fish counting is easily transferable to a wide array of analysis techniques pertinent to many other salient ecological questions.

3.5.2 Evaluation Metrics

For the application of counting fish, we borrow metrics used for crowd counting [86] and propose to evaluate algorithmic counting accuracy via mean absolute error:

$$\epsilon = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

where N is the number of sequences, y_i is the algorithmic prediction of the number of fish for sequence i, and $\hat{y}_i$ is the ground truth number of fish for sequence i.

Since we have frame level bounding box annotations with unique object identifiers, the dataset annotations allow for broader applications than just fish counting. As alluded to previously, we believe this dataset will be of interest to the multi-object tracking domain, and accurate tracking algorithms can enable deeper insights into fish behavior and ecosystem dynamics. Hence, we also propose to use standard multi-object tracking metrics, as used in the MOT dataset [73] [91]. Specifically, the two metrics of Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP), as defined by [132]:

$$MOTA = 1 - \frac{\sum_i^N (FN_i + FP_i + IDSW_i}{\sum_i^N GT_i}$$

$$MOTP = \frac{\sum_{i,t} d_{i,t}}{\sum_i c_i}$$

where for MOTA N is the number of frames, FN_i is the number of false negatives, FP_i is the number of false positives, and $IDSW_i$ is the number of track ID switches in a given frame i. For MOTP, $d_{i,t}$ is the distance measure of target t from its ground truth annotation in frame i, and c_i is the number of predicted bounding box matches with the ground truth in frame i. $d_{i,t}$ is typically the intersection-over-union (IoU), and we follow suite, thresholding matching bounding boxes at 50% IoU. Intuitively, MOTP is a measure of how well a tracker can localize object positions, while MOTA is a combination of three error metrics, which expresses how well the tracker can estimate the number of objects and maintain consistent track trajectories.

3.6 Results and Discussion

There are a total of 19,901 annotated bounding boxes and nearly $218K$ frames across the train and val splits of the NHFish dataset. Specifically, each split has:

- Training Set: 9361 annotations, across nearly 143K frames
- Validation Set: 10540 annotations, across nearly 75000 frames.

There are varying weather and ocean conditions across the sequences, and the data is drawn from trawls performed in different months in 2013 and 2014 along the NH Line in Oregon. This provides a representative sample of the data produced by the beam trawls. Example frames are given in fig. 3.19. Statistics regarding dataset ground truth are given in table 3.1.

Bounding Box Metric	**Value**
Mean Area	39929 px
Median Area	35534 px
Mode Area	32760 px
Median Height	171 px
Median Width	206 px
Height Std.	75.46 px
Width Std.	74.31 px
Skew, Kurtosis	4.62, 38.11

Table 3.1: Descriptive statistics about ground truth bounding box annotations across the train and validation dataset splits.

3.6.0.1 Detection Proposals

The detection proposals from RetinaNet and Faster R-CNN are included in the dataset; since proposals were generated from fine-tuned networks, we performed two training and evaluation cycles for the detector networks, starting from their base COCO-detection trained models:

1. Train on the NHFish training set, generate proposals for the validation set

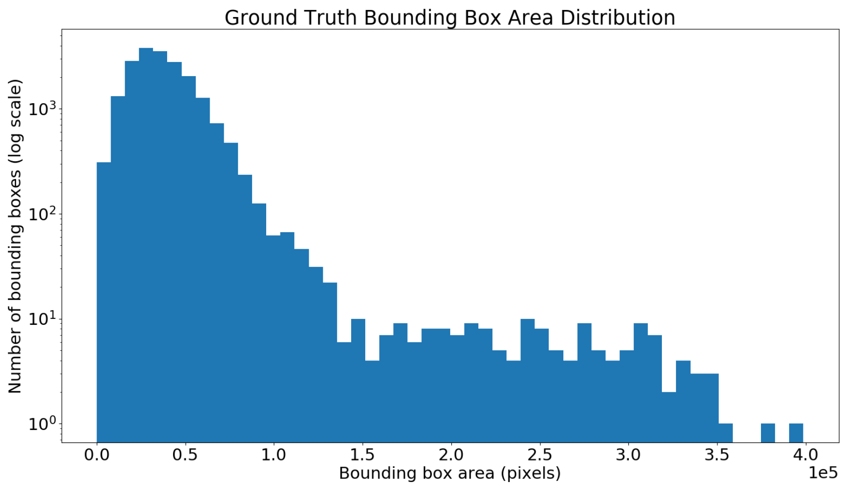


Figure 3.16: Ground truth bounding box size histogram, in terms of pixels using 50 bins. Note that the majority are comparatively small, with the smallest bounding box at 30 pixels.

2. Train on the NHFish validation set, generate proposals for the training set

In this way, we generate detection proposals for all frames in the dataset. In total, there are 70,588 bounding box proposals that have confidence scores > 0.5, and 457,104 proposals total without thresholding generated from RetinaNet. There are 60,726 Faster R-CNN proposals, 47,409 of which have confidence scores > 0.5. The detection proposal performance, as given by the Detectron project's [43] utilities, are given in table 3.2 and table 3.3.

The evaluation metrics in table 3.2 and table 3.3 are used by COCO for evaluating detection performance. The evaluation descriptions from the COCO website [9] are replicated below:

- Average Precision (AP):
    - *AP*: AP is averaged over 10 discrete thresholds (specifically, $IoU = [.50 : .05 : .95]$) rather than a single one. This provides progressively higher AP for

[9]From COCO Detection: `http://cocodataset.org/#detection-eval`

RetinaNet	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
train-val	0.2506	0.4932	0.2215	0.0	0.0022	0.2546
val-train	0.30095	0.5584	0.2930	0.0	0.0091	0.3098

Table 3.2: RetinaNet ResNet-50 detection performance on the two detection proposal generation runs; `train-val` means the RetinaNet model was trained on the NHFish training set, and the proposals were generated on the NHFish validation set. `val-train` is the inverse (training on validation, proposals on train). This provides the opportunity for detection proposals in all frames in the dataset.

Faster R-CNN	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
train-val	0.2793	0.5116	0.2747	0.0	0.0080	0.2838
val-train	0.2909	0.5233	0.2930	0.0	0.0203	0.2991

Table 3.3: Faster R-CNN ResNet-101 detection performance on the two detection proposal generation runs; `train-val` means the Faster R-CNN model was trained on the NHFish training set, and the proposals were generated on the NHFish validation set. `val-train` is the inverse (training on validation, proposals on train). This provides the opportunity for detection proposals in all frames in the dataset.

bounding boxes with more accurate object localizations.

- AP_{50}: AP with a threshold of IoU=.50 for determining bounding box matching. Concretely, if two bounding boxes have < 0.5 IoU, they do not match; otherwise they do.
- AP_{75}: AP with a threshold of IoU=.75 for determining bounding box matching. This is a more stringent localization requirement than AP_{50}.

- AP Across Scales:
  - AP_s: AP for small objects: area < 322
  - AP_m: AP for medium objects: $322 <$ area < 962
  - AP_l: AP for large objects: area > 962

RetinaNet and Faster R-CNN are both successful state of the art object detection networks. Their relatively poor performance on NHFish, as shown in table 3.2 and table 3.3, is indicative of how challenging the dataset can be for existing methods. This motivates the creation of inventive and effective automated analysis techniques that can accommodate NHFish's difficult factors. Additionally, in light of the detection network results, successful tracking approaches on NHFish will have to be robust to imperfect detection proposals. Depending on the confidence threshold used for filtering which detection proposals, there may be too few true positives, or too many false positives. In all cases, there may be poorly localized bounding boxes. This sort of noise in the detection proposals adds an additional layer of difficulty for tracking methods on NHFish.

Proposal Bounding Box Metric	**RetinaNet ResNet-50**	**Faster R-CNN ResNet-101**
Mean Area (pixels)	36486	35195
Median Area (pixels)	29439	30294
Mode Area (pixels)	2877	1890
Median Height (pixels)	178	176
Median Width (pixels)	158	160
Height Std. (pixels)	77.91	61.76
Width Std. (pixels)	64.24	79.89
Skew, Kurtosis	2.34, 13.03	2.81, 25.12

Table 3.4: Descriptive statistics about the set of RetinaNet and Faster R-CNN bounding box proposals across the train and validation dataset splits.

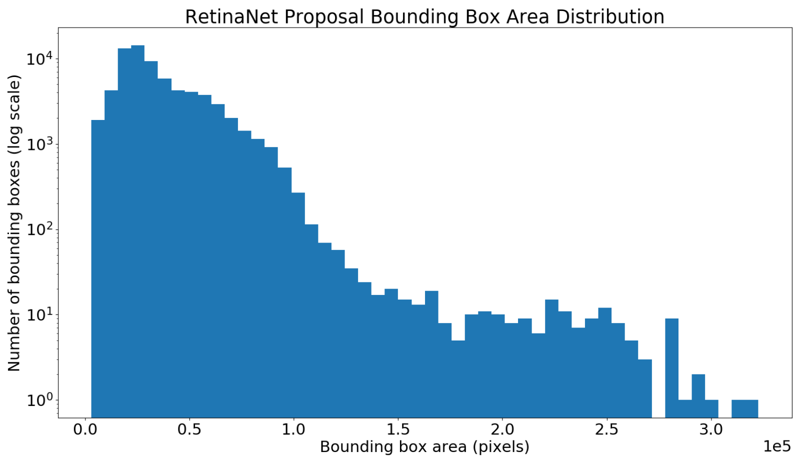


Figure 3.17: RetinaNet bounding box proposal size histogram, in terms of pixels using 50 bins. Note that the detection's area approximates the ground truth's distribution relatively closely, as seen in fig. 3.16

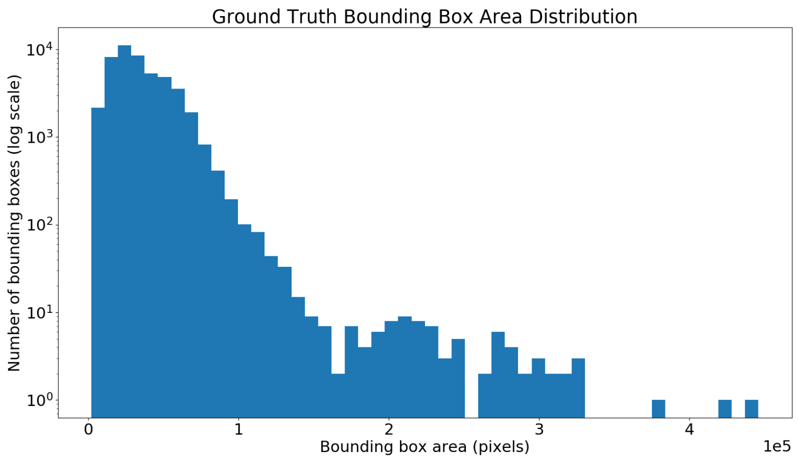


Figure 3.18: Faster R-CNN bounding box proposal size histogram, in terms of pixels using 50 bins. Note that the detection's area approximates the ground truth's distribution relatively closely, as seen in fig. 3.16, but seems more skewed towards smaller proposals than RetinaNet fig. 3.17.

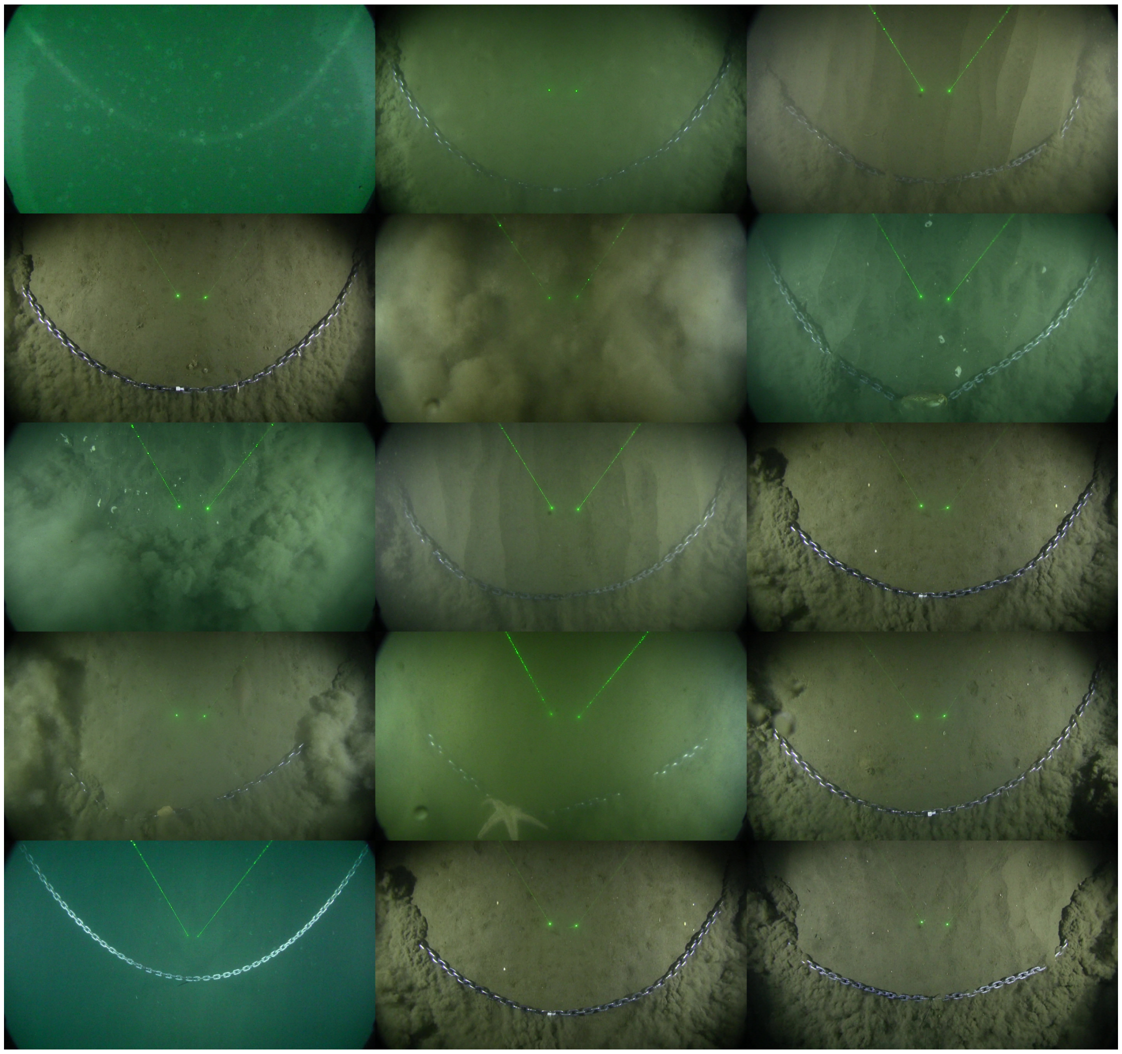


Figure 3.19: Example frames from different video sequences in the NHFish dataset; each frame is the 1000th or 10000th frame of a different sequence.

3.7 Conclusion

We present NHFish, a novel and challenging tracking-by-detection dataset of beam trawl video from the NH-Line with bounding box annotations. We believe this is a significant step towards fostering greater interdisciplinary collaboration between fisheries science and CV, and will encourage future work beneficial to both fields. The NHFish dataset has great potential for improving scientific understanding of Oregon benthos, and streamlining stock assessments, which play a critical role in fisheries management. Likewise, successful tracking and video analysis algorithms will have to integrate motion cues and be robust to myriad distractors and noise sources, which poses an interesting opportunity for innovation in CV. On a more philosophical level, we believe that datasets can operate as boundary objects for interdisciplinary projects, and are an effective and general means for catalyzing greater interdisciplinary collaborative efforts that will see increasing attention in coming years.

Chapter 4: Conclusion

In this thesis, we present two video analysis methods, one for performing semi-supervised VOS (ReGuide), and the other an application of tracking-by-detection on a newly proposed, high-resolution underwater visual tracking dataset of Oregon benthos, NHFish. In both cases, the driving component was to better utilize long-term temporal information in videos, and elevate video processing above simply frame-by-frame image processing. In the case of ReGuide, we use a bilinear LSTM to learn long-term appearance models of target objects, and apply it as a convolutional filter to modulate learned CNN features of the scene. This enables the network to adapt online to the appearance shift specific objects undergo over a video sequence. The ReGuide network is online and end-to-end trainable, and the guide network component can be integrated into a variety of network architectures. It is complementary to inference-time fine-tuning, as is common in semi-supervised VOS, and can be used in conjunction with other post-hoc refinement methods, including those involving offline temporal refinement. Initial experiments on DAVIS show promising results with favorable performance compared to baseline methods.

NHFish is a new video tracking dataset of beam trawl data along the Newport Hydrographic Line, specifically looking at endemic flatfish populations. It consists of roughly $218K$ frames, with 19,899 bounding box annotations between the training and validation set. It offers a unique set of challenges in a video tracking dataset, such as visually non-distinct (naturally camouflaged) target objects, frequent and invasive visual distractors, and non-linear object and camera motion, all in an underwater environment. Successfully integrating motion cues is an especially important aspect for successfully analyzing the data, and we believe it will be of interest to CV practitioners. We present a set of detection proposals from state of the art detection methods, and offer initial results for fish stock assessments from the videos. On the ecological side, the beam trawl data has the potential to greatly expedite the process of fishery stock assessments and perform automated ecological inferences from video data. This could prove valuable for gaining deeper insights for fisheries science into fish behavior, ecology and lifecycles, and for fisheries management in creating economically and ecologically viable management plans. We believe this dataset will foster greater integration between CV and marine science, and contribute towards the use of AI techniques for societal and environmental good.

Bibliography

[1] The francis crick papers: The discovery of the double helix, 1951-1953. *Profiles.nlm.nih.gov*, 2018.

[2] AN ACT. Magnuson-stevens fishery conservation and management act. *Public Law*, 94:265, 1996.

[3] M. N. Ahmed, A. S. Toor, K. O'Neil, and D. Friedland. Cognitive computing and the future of health care cognitive computing and the future of healthcare: The cognitive power of ibm watson has the potential to transform global personalized medicine. *IEEE Pulse*, 8(3):4–9, May 2017.

[4] John G Allen, Richard YD Xu, and Jesse S Jin. Object tracking using camshift algorithm and multiple quantized feature spaces. In *Proceedings of the Pan-Sydney area workshop on Visual information processing*, pages 3–7. Australian Computer Society, Inc., 2004.

[5] Sabrina Amrouche, Nils Braun, Paolo Calafiura, Steven Farrell, Jochen Gemmler, Cécile Germain, Vladimir Vava Gligorov, Tobias Golling, Heather Gray, Isabelle Guyon, et al. Track reconstruction at lhc as a collaborative data challenge use case with ramp. In *EPJ Web of Conferences*, volume 150, page 00015. EDP Sciences, 2017.

[6] Vijay Badrinarayanan, Ignas Budvytis, and Roberto Cipolla. Mixture of trees probabilistic graphical model for video segmentation. *International journal of computer vision*, 110(1):14–29, 2014.

[7] Linchao Bao, Baoyuan Wu, and Wei Liu. Cnn in mrf: Video object segmentation via inference in a cnn-based higher-order spatio-temporal mrf. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5977–5986, 2018.

[8] Tanya Y Berger-Wolf, Daniel I Rubenstein, Charles V Stewart, Jason A Holmberg, Jason Parham, Sreejith Menon, Jonathan Crall, Jon Van Oast, Emre Kiciman, and Lucas Joppa. Wildbook: Crowdsourcing, computer vision, and data science for conservation. *arXiv preprint arXiv:1710.08880*, 2017.

[9] Joël Blit, Samantha St Amand, and Joanna Wajda. Automation and the future of work: Scenarios and policy options. 2018.

[10] Franziska Boenisch, Benjamin Rosemann, Benjamin Wild, Fernando Wario, David Dormagen, and Tim Landgraf. Tracking all members of a honey bee colony over their lifetime. *arXiv preprint arXiv:1802.03192*, 2018.

[11] William Brendel and Sinisa Todorovic. Video object segmentation by tracking regions. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 833–840. IEEE, 2009.

[12] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European conference on computer vision*, pages 282–295. Springer, 2010.

[13] Marjolein Bruijning, Marco D Visser, Caspar A Hallmann, and Eelke Jongejans. trackdem: Automated particle tracking to obtain population counts and size distributions from videos in r. *Methods in Ecology and Evolution*, 9(4):965–973, 2018.

[14] Erik Brynjolfsson and Tom Mitchell. What can machine learning do? workforce implications. *Science*, 358(6370):1530–1534, 2017.

[15] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials*, 18(2):1153–1176, Secondquarter 2016.

[16] Sergi Caelles, Yuhua Chen, Jordi Pont-Tuset, and Luc Van Gool. Semantically-guided video object segmentation. *arXiv preprint arXiv:1704.01926*, 2017.

[17] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *CVPR 2017*. IEEE, 2017.

[18] Sergi Caelles, Alberto Montes, Kevis-Kokitsi Maninis, Yuhua Chen, Luc Van Gool, Federico Perazzi, and Jordi Pont-Tuset. The 2018 davis challenge on video object segmentation. *arXiv:1803.00557*, 2018.

[19] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs (2016). *arXiv preprint arXiv:1606.00915*, 2016.

[20] Jingchun Cheng, Sifei Liu, Yi-Hsuan Tsai, Wei-Chih Hung, Shalini De Mello, Jinwei Gu, Jan Kautz, Shengjin Wang, and Ming-Hsuan Yang. Learning to segment instances in videos with spatial propagation network. *arXiv preprint arXiv:1709.04609*, 2017.

[21] Jingchun Cheng, Yi-Hsuan Tsai, Wei-Chih Hung, Shengjin Wang, and Ming-Hsuan Yang. Fast and accurate online video object segmentation via tracking parts. *arXiv preprint arXiv:1806.02323*, 2018.

[22] Jingchun Cheng, Yi-Hsuan Tsai, Shengjin Wang, and Ming-Hsuan Yang. Segflow: Joint learning for video object segmentation and optical flow. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 686–695. IEEE, 2017.

[23] Meng-Che Chuang, Jenq-Neng Hwang, Jian-Hui Ye, Shih-Chia Huang, and Kresimir Williams. Underwater fish tracking for moving cameras based on deformable multiple kernels. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(9):2467–2477, 2017.

[24] Dan CireşAn, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural networks*, 32:333–338, 2012.

[25] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[26] Sophie Curtis. Google photos labels black people as 'gorillas'. *The Telegraph*, Jul 2015.

[27] Ranga Dabarera and Ranga Rodrigo. Vision based elephant recognition for management and conservation. In *Information and Automation for Sustainability (ICIAFs), 2010 5th International Conference on*, pages 163–166. IEEE, 2010.

[28] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

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

[30] Shai Danziger, Jonathan Levav, and Liora Avnaim-Pesso. Extraneous factors in judicial decisions. *Proceedings of the National Academy of Sciences*, 108(17):6889–6892, 2011.

[31] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[32] 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, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[33] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1841–1848, 2013.

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

[35] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[36] Robert B Fisher, Yun-Heh Chen-Burger, Daniela Giordano, Lynda Hardman, Fang-Pang Lin, et al. *Fish4Knowledge: collecting and analyzing massive coral reef fish video data*, volume 104. Springer, 2016.

[37] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[38] Fabio Galasso, Naveen Shankar Nagaraja, Tatiana Jimenez Cardenas, Thomas Brox, and Bernt Schiele. A unified video segmentation benchmark: Annotation, metrics and analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3527–3534, 2013.

[39] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[40] Simone V Gill, Misha Vessali, Jacob A Pratt, Samantha Watts, Janey S Pratt, Preeti Raghavan, and Jeremy M DeSilva. The importance of interdisciplinary research training and community dissemination. *Clinical and translational science*, 8(5):611–614, 2015.

[41] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[42] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[43] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018.

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[45] M. W. Goudreau, C. L. Giles, S. T. Chakradhar, and D. Chen. First-order versus second-order single-layer recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(3):511–513, May 1994.

[46] J. Han, L. Shao, D. Xu, and J. Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics*, 43(5):1318–1334, Oct 2013.

[47] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[49] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[50] Ekram Hossain, SM Shaiful Alam, Amin Ahsan Ali, and M Ashraful Amin. Fish activity tracking and species identification in underwater video. In *Informatics, Electronics and Vision (ICIEV), 2016 5th International Conference on*, pages 62–66. IEEE, 2016.

[51] Ping Hu, Gang Wang, Xiangfei Kong, Jason Kuen, and Yap-Peng Tan. Motion-guided cascaded refinement network for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1400–1409, 2018.

[52] Yuan-Ting Hu, Jia-Bin Huang, and Alexander Schwing. Maskrnn: Instance level video object segmentation. In *Advances in Neural Information Processing Systems*, pages 324–333, 2017.

[53] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*, volume 4, 2017.

[54] T Huang. Computer vision: Evolution and promise. 1996.

[55] Yuchi Huang, Qingshan Liu, and Dimitris Metaxas.] video object segmentation by hypergraph cut. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1738–1745. IEEE, 2009.

[56] Jonas Jäger, Viviane Wolff, Klaus Fricke-Neuderth, Oliver Mothes, and Joachim Denzler. Visual fish tracking: Combining a two-stage graph approach with cnn-features. In *OCEANS 2017-Aberdeen*, pages 1–6. IEEE, 2017.

[57] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. *arXiv preprint arXiv:1701.05384*, 2(3):6, 2017.

[58] Varun Jampani, Raghudeep Gadde, and Peter V Gehler. Video propagation networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, 2017.

[59] Won-Dong Jang and Chang-Su Kim. Online video object segmentation via convolutional trident network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5849–5858, 2017.

[60] Taeho Jo, Jie Hou, Jesse Eickholt, and Jianlin Cheng. Improving protein fold recognition by deep learning networks. *Scientific reports*, 5:17573, 2015.

[61] Aimee A Keller, John Robert Wallace, and Richard Donald Methot. The northwest fisheries science center's west coast groundfish bottom trawl survey: history, design, and description. 2017.

[62] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. Lucid data dreaming for object tracking. *arXiv preprint arXiv:1703.09554*, 2017.

[63] Chanho Kim, Fuxin Li, and James M Rehg. Multi-object tracking with neural gating using bilinear lstms. In *European conference on computer vision*. Springer, 2018.

[64] Toshifumi Kimura, Mizue Ohashi, Karl Crailsheim, Thomas Schmickl, Ryuichi Okada, Gerald Radspieler, and Hidetoshi Ikeno. Development of a new method to track multiple honey bees with complex behaviors on a flat laboratory arena. *PloS one*, 9(1):e84656, 2014.

[65] ByoungChul Ko and Sooyeong Kwak. Survey of computer vision-based natural disaster warning systems. *Optical Engineering*, 51(7):070901, 2012.

[66] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin. A novel performance evaluation methodology for single-target trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, Nov 2016.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[68] LJ Latecki and Tianyang Ma. Maximum weight cliques with mutex constraints for video object segmentation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 670–677. IEEE, 2012.

[69] Francesco Lattari, Marco Ciccone, Matteo Matteucci, Jonathan Masci, and Francesco Visin. Reconvnet: Video object segmentation with spatio-temporal features modulation. *arXiv preprint arXiv:1806.05510*, 2018.

[70] Tor Lattimore and Marcus Hutter. No free lunch versus occam's razor in supervised learning. *CoRR*, abs/1111.3846, 2011.

[71] Richard J Lazarus. Super wicked problems and climate change: Restraining the present to liberate the future. *Cornell L. Rev.*, 94:1153, 2008.

[72] T.-N. Le, K.-T. Nguyen, M.-H. Nguyen-Phan, T.-V. Ton, T.-A. Nguyen (2), X.-S. Trinh, Q.-H. Dinh, V.-T. Nguyen, A.-D. Duong, A. Sugimoto, T. V. Nguyen, and M.-T. Tran. Instance re-identification flow for video object segmentation. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017.

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

[74] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[75] Yong Jae Lee, Jaechul Kim, and Kristen Grauman. Key-segments for video object segmentation. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1995–2002. IEEE, 2011.

[76] José Lezama, Karteek Alahari, Josef Sivic, and Ivan Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3369–3376. IEEE, 2011.

[77] Fuxin Li, Taeyoung Kim, Ahmad Humayun, David Tsai, and James M Rehg. Video segmentation by tracking many figure-ground segments. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2192–2199. IEEE, 2013.

[78] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *CVPR*, 2014.

[79] X. Li, Y. Qi, Z. Wang, K. Chen, Z. Liu, J. Shi, P. Luo, C. Change Loy, and X. Tang. Video object segmentation with re-identification. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017.

[80] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017.

[81] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[82] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[83] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: Fast feature extraction and svm training. In *CVPR 2011*, pages 1689–1696, June 2011.

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

[85] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[86] Chen Change Loy, Ke Chen, Shaogang Gong, and Tao Xiang. Crowd counting and profiling: Methodology and evaluation. In *Modeling, Simulation and Visual Analysis of Crowds*, pages 347–382. Springer, 2013.

[87] Kevis-Kokitsi Maninis, Sergi Caelles, Yuhua Chen, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. Video object segmentation without temporal information. *arXiv preprint arXiv:1709.06031*, 2017.

[88] Nicolas Märki, Federico Perazzi, Oliver Wang, and Alexander Sorkine-Hornung. Bilateral space video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 743–751, 2016.

[89] John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12, 2006.

[90] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

[91] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[92] Hauke Jürgen Mönck, Andreas Jörg, Tobias von Falkenhausen, Julian Tanke, Benjamin Wild, David Dormagen, Jonas Piotrowski, Claudia Winklmayr, David Bierbach, and Tim Landgraf. Biotracker: An open-source computer vision framework for visual animal tracking. *arXiv preprint arXiv:1803.07985*, 2018.

[93] Henning Müller, Paul Clough, Thomas Deselaers, Barbara Caputo, and Image CLEF. Experimental evaluation in visual information retrieval. *The Information Retrieval Series*, 32:1–554, 2010.

[94] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. In *ACM SIGGRAPH 2008 classes*, page 16. ACM, 2008.

[95] Peter Ochs and Thomas Brox. Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. 2011.

[96] Peter Ochs and Thomas Brox. Higher order motion models and spectral clustering. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 614–621. IEEE, 2012.

[97] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1187–1200, 2014.

[98] Julian D Olden, Donald A Jackson, and Pedro R Peres-Neto. Predictive models of fish species distributions: a note on proper validation and chance predictions. *Transactions of the American Fisheries Society*, 131(2):329–336, 2002.

[99] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1777–1784, 2013.

[100] L. Patino, T. Nawaz, T. Cane, and J. Ferryman. Pets 2017: Dataset and challenge. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2126–2132, July 2017.

[101] Terry C Pellmar, Leon Eisenberg, et al. The potential of interdisciplinary research to solve problems in the brain, behavioral, and clinical sciences. 2000.

[102] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A.Sorkine-Hornung. Learning video object segmentation from static images. In *Computer Vision and Pattern Recognition*, 2017.

[103] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.

[104] Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. Learning video object segmentation from static images. In *Computer Vision and Pattern Recognition*, 2017.

[105] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732, 2016.

[106] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017.

[107] Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid, and Vittorio Ferrari. Learning object class detectors from weakly annotated video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3282–3289. IEEE, 2012.

[108] Thierry Pun, Guido Gerig, and Osman Ratib. Image analysis and computer vision in medicine. *Computerized Medical Imaging and Graphics*, 18(2):85–96, 1994.

[109] Hongwei Qin, Xiu Li, Jian Liang, Yigang Peng, and Changshui Zhang. Deepfish: Accurate underwater live fish recognition with a deep architecture. *Neurocomputing*, 187:49–58, 2016.

[110] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[111] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[112] Horst WJ Rittel and Melvin M Webber. Dilemmas in a general theory of planning. *Policy sciences*, 4(2):155–169, 1973.

[113] Lawrence G Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.

[114] Alvaro Rodriguez, Angel J Rico-Diaz, Juan R Rabuñal, Jeronimo Puertas, and Luis Pena. Fish monitoring and sizing using computer vision. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 419–428. Springer, 2015.

[115] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[116] David Rousseau. Trackml: Tracking machine learning challenge. 2016.

[117] David Rousseau, Sabrina Amrouche, Paolo Calafiura, Steven Farrell, Cecile Germain, Vladimir Gligorov, Tobias Golling, Heather Gray, Isabelle Guyon, Mikhail Hushchyn, et al. Wcci 2018 trackml particle tracking challenge. 2018.

[118] Andrew Rova, Greg Mori, and Lawrence M Dill. One fish, two fish, butterfish, trumpeter: Recognizing fish in underwater video. In *MVA*, pages 404–407, 2007.

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

[120] Peter J Schüffler, Thomas J Fuchs, Cheng Soon Ong, Peter J Wild, Niels J Rupp, and Joachim M Buhmann. Tmarker: A free software toolkit for histopathological cell counting and staining estimation. *Journal of pathology informatics*, 4(Suppl), 2013.

[121] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[122] A. Shaban, A. Firl, A. Humayun, J. Yuan, X. Wang, P. Lei, N. Dhanda, B. Boots, J. M. Rehg, and F. Li. Multiple-instance video segmentation with sequence-specific object proposals. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2017.

[123] Faisal Shafait, Ajmal Mian, Mark Shortis, Bernard Ghanem, Phil F Culverhouse, Duane Edgington, Danelle Cline, Mehdi Ravanbakhsh, James Seager, and Euan S Harvey. Fish identification from videos captured in uncontrolled underwater environments. *ICES Journal of Marine Science*, 73(10):2737–2746, 2016.

[124] Gilad Sharir, Eddie Smolyansky, and Itamar Friedman. Video object segmentation using tracked object proposals. *arXiv preprint arXiv:1707.06545*, 2017.

[125] X Shi, Z Chen, H Wang, DY Yeung, WK Wong, and WC Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. arxiv, 2015. *arXiv preprint arXiv:1506.04214*.

[126] Pushkar Shukla, Isha Dua, Balasubramanian Raman, and Ankush Mittal. A computer vision framework for detecting and preventing human-elephant collisions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2883–2890, 2017.

[127] Shoaib Ahmed Siddiqui, Ahmad Salman, Muhammad Imran Malik, Faisal Shafait, Ajmal Mian, Mark R Shortis, Euan S Harvey, and Handling editor: Howard Browman. Automatic fish species classification in underwater videos: exploiting pre-trained deep neural network models to compensate for limited labelled data. *ICES Journal of Marine Science*, 75(1):374–389, 2017.

[128] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):1, 2013.

[129] Julie Sobowale. How artificial intelligence is transforming the legal profession. *ABA Journal.[online]. Available at: www. abajournal. com/magazine/article/how_artificial_intelligence_is_transforming_the_legal_profession/. Stonier*, 2016.

[130] Concetto Spampinato, Yun-Heh Chen-Burger, Gayathri Nadarajan, and Robert B Fisher. Detecting, tracking and counting fish in low quality unconstrained underwater videos. *VISAPP (2)*, 2008(514-519):1, 2008.

[131] Concetto Spampinato, Daniela Giordano, Roberto Di Salvo, Yun-Heh Jessica Chen-Burger, Robert Bob Fisher, and Gayathri Nadarajan. Automatic fish classification for underwater species behavior understanding. In *Proceedings of the*

first ACM international workshop on Analysis and retrieval of tracked events and motion in imagery streams, pages 45–50. ACM, 2010.

[132] Rainer Stiefelhagen, Keni Bernardin, Rachel Bowers, John Garofolo, Djamel Mostefa, and Padmanabhan Soundararajan. The clear 2006 evaluation. In *International evaluation workshop on classification of events, activities and relationships*, pages 1–44. Springer, 2006.

[133] Amy Stinton, Lorenzo Ciannelli, Douglas C Reese, and W Waldo Wakefield. Using in situ video analysis to assess juvenile flatfish behavior along the oregon central coast. *California Cooperative Oceanic Fisheries Investigations Reports*, 55:158–168, 2014.

[134] Norval James Colin Strachan, Paul Nesvadba, and Alastair R Allen. Fish species recognition by shape analysis of images. *Pattern Recognition*, 23(5):539–544, 1990.

[135] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 843–852. IEEE, 2017.

[136] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. Learning video object segmentation with visual memory. *arXiv preprint arXiv:1704.05737*, 3, 2017.

[137] David Tsai, Matthew Flagg, and James M.Rehg. Motion coherent tracking with multi-label mrf optimization. *BMVC*, 2010.

[138] Yi-Hsuan Tsai, Ming-Hsuan Yang, and Michael J Black. Video segmentation via object flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3899–3908, 2016.

[139] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[140] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. 2018.

[141] Amelio Vazquez-Reina, Shai Avidan, Hanspeter Pfister, and Eric Miller. Multiple hypothesis video segmentation from superpixel flows. In *European conference on Computer vision*, pages 268–281. Springer, 2010.

[142] David Vernon. *Machine Vision: Automated Visual Inspection and Robot Vision.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[143] Paul Voigtlaender and Bastian Leibe. Online adaptation of convolutional neural networks for video object segmentation. *arXiv preprint arXiv:1706.09364*, 2017.

[144] Daisuke Wakabayashi. Self-driving uber car kills pedestrian in arizona, where robots roam. *The New York Times*, Mar 2018.

[145] Jana Wäldchen and Patrick Mäder. Plant species identification using computer vision techniques: A systematic literature review. *Archives of Computational Methods in Engineering*, 25(2):507–543, 2018.

[146] Nancy Xin Ru Wang, Sarika Cullis-Suzuki, and Alexandra Branzan Albu. Automated analysis of wild fish behavior in a natural habitat. In *Proceedings of the 2nd International Workshop on Environmental Multimedia Retrieval*, pages 21–26. ACM, 2015.

[147] Etienne Wenger. *Communities of practice: Learning, meaning, and identity.* Cambridge university press, 1999.

[148] Kresimir Williams, Chris Rooper, and John Harms. Report of the national marine fisheries service automated image processing workshop. *NOAA Technical Memorandum NMFS-F/SPO-121*, 2012.

[149] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Apr 1997.

[150] Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan R Salakhutdinov. On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 2856–2864, 2016.

[151] Zhengyang Wu, Fuxin Li, Rahul Sukthankar, and James M Rehg. Robust video segment proposals with painless occlusion handling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4194–4203, 2015.

[152] Seoung Wug Oh, Joon-Young Lee, Kalyan Sunkavalli, and Seon Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7376–7385, 2018.

[153] Huaxin Xiao, Jiashi Feng, Guosheng Lin, Yu Liu, and Maojun Zhang. Monet: Deep motion exploitation for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1140–1148, 2018.

[154] Jia Xu, René Ranftl, and Vladlen Koltun. Accurate Optical Flow via Direct Cost Volume Processing. In *CVPR*, 2017.

[155] Linjie Yang, Yanran Wang, Xuehan Xiong, Jianchao Yang, and Aggelos K Katsaggelos. Efficient video object segmentation via network modulation. *algorithms*, 29:15, 2018.

[156] Jae Shin Yoon, Francois Rameau, Junsik Kim, Seokju Lee, Seunghak Shin, and In So Kweon. Pixel-level matching for video object segmentation using convolutional neural networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2186–2195. IEEE, 2017.

[157] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[158] Dong Zhang, Omar Javed, and Mubarak Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 628–635, 2013.

[159] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–10, Feb 2012.

[160] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *arXiv preprint arXiv:1707.09457*, 2017.

[161] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Computer Vision, IEEE International Conference on*, 2015.

[162] Xiaojin Zhu, Bryan R Gibson, and Timothy T Rogers. Human rademacher complexity. In *Advances in Neural Information Processing Systems*, pages 2322–2330, 2009.

APPENDICES

Appendix A: NRT FishLabeler

As part of the OSU NSF NRT team, we manually labeled a selection of beam trawl videos to define the ground truth tracks for the NHFish dataset. As part of that effort, a labeling application (NRT FishLabeler) was developed; the driving goal was to have an application that was simple to set up on a variety of operating systems, which could step through videos frame-by-frame, and which supported a variety of per-frame metadata annotation modalities (i.e. pixel-wise segmentation labels, geometric shapes, textual information, etc). This is the document distributed among the team detailing the steps for setup, usage, and labeling heuristics. The FishLabeler software is open source, and both the docker images and source code have been made available.

A.1 Introduction

FishLabeler is a piece of software written to expedite the video labeling process. Specifically, it allows stepping through the video on a frame-by-frame basis, annotating the frames with pixel-wise segmentation labels or bounding boxes, and adding free-text annotations for frames. The resultant metadata (i.e. bounding box lists, segmentation masks, and free-text data) are written out to disk. To simplify distribution, the application is dockerized, and so in theory is relatively portable – the main challenge is getting the X-forwarding to work, since this is a GUI-based application. To date, it works on Linux, Windows, and Mac systems. You can find the docker image at the docker hub repository [1] and the source code at the git repository [2] (under the *dockerized* branch).

A.2 Before You Run

Check if your docker image is up to date: in the terminal, run `docker pull alrikai/nrt` to pull the latest image from the docker hub repository. You can see which docker images

[1]https://hub.docker.com/r/alrikai/nrt/
[2]https://bitbucket.org/alrikai/fishlabeler

are on your system by running `docker images`. If docker is not running, then begin running it (if it's not open, then you should get an error message when running any docker commands).

For example, running `docker images` gives:

```
alrik@kai:~$ docker images
REPOSITORY      TAG        IMAGE ID         CREATED         SIZE
alrikai/nrt     latest     a63170ce013c     9 days ago      1.33GB
```

Old docker images will clutter your system over time; old docker images can be deleted by running (on the command line):

```
docker system prune
```

A.3 Linux (Tested on Ubuntu 17.10 and 16.04)

A.3.1 Installation Requirements and Setup

1. Install Docker CE – follow instructions on the page
2. Follow instructions on running docker as non-root user. This is important for sharing the X11 socket.

A.3.2 Running Docker Image

1. Make sure your docker image is up-to-date section A.2
2. Run the docker image from the command line:

   ```
    docker run -ti --rm -u $(id -u) -v <src data dir>:/data \
   -v <dst data dir>:/outdata -e DISPLAY=$DISPLAY \
   -v /tmp/.X11-unix:/tmp/.X11-unix alrikai/nrt bash
   ```

   Note that the $-u$ *name* option sets the username within the docker container. On one system, I have it set to NRTFish (since this is the username that is set in the Dockerfile), but it seems to work with the current user (*$(id -u)*) as well. e.g.

```
 docker run -ti --rm -u $(id -u) -v /home/alrik/Data:/data \
-v /home/alrik/Data:/outdata -e DISPLAY=$DISPLAY \
-v /tmp/.X11-unix:/tmp/.X11-unix alrikai/nrt bash
```

3. Run commands as detailed in section A.6.

A.4 Windows

A.4.1 Installation Requirements and Setup

1. Install Docker (if Win10 Pro, install Docker CE, otherwise install Docker Toolbox).
2. Install XMimg using default options.
3. Open Docker 'Settings', select 'C' (or whatever drive has the video data) as a shared drive.

A.4.2 Running Docker Image

1. Make sure your docker image is up-to-date section A.2
2. Run XLaunch if XMimg is not running, note the display number (defaults to 0), and select the 'No Access Control' checkbox under the 'Additional Parameters' dialog. Leave everything else as defaults. Or, you can save the profile to disk to expedite the process in the future.
3. Open the XMimg log in the toolbar (right click, "view log", if open), note the XMimg IP address (the one next to *newAddress*).
4. Open `Powershell` if Docker CE, or `docker quickstart terminal` if Docker Toolbox – see below for commands depending on if you're using Docker toolbox or Docker CE.
5. Start docker if it's not running.

A.4.3 Docker CE

1.
   ```
   docker run -ti --rm -u NRTfish -v <src data dir>:/data \
   -v <dst data dir>:/outdata \
   -e DISPLAY=<ip addr>:<display #> alrikai/nrt bash
   ```

   The commands between < `brackets` > should be substituted with actual values (and the brackets omitted). As a concrete example,

   ```
   docker run -ti --rm -u NRTfish -v C:/:/data -v C:/:/outdata \
   -e DISPLAY=192.168.56.1:0 alrikai/nrt bash
   ```

2. Run commands as detailed in section A.6.

A.4.4 Docker Toolbox

1.
   ```
   docker run -ti --rm -u NRTfish -v "<src data dir>":/data \
   -v "<dst data dir>":/outdata \
   -e DISPLAY=<ip addr>:<display #> alrikai/nrt bash
   ```

   ... where the parts between <brackets> should be substituted with actual values (and the brackets omitted). As an example,

   ```
   docker run -ti --rm -u NRTfish \
   -v "/c/Users/Katlyn/Documents/NRT/NRTDATA/Source":/data \
   -v "/c/Users/Katlyn/Documents/NRT/NRTDATA/Output":/outdata \
   -e DISPLAY=10.248.234.174:0 alrikai/nrt bash
   ```

2. Run commands as detailed in section A.6.

A.5 Mac

A.5.1 Installation Requirements and Setup

1. Install homebrew (if needed)

2. Install Docker
3. Install XQuartz (or via homebrew → `brew cask install xquartz`)
4. Install socat (`brew install socat`)

A.5.2 Running Docker Image

1. Make sure your docker image is up-to-date section A.2
2. Run `socat TCP-LISTEN:6000,reuseaddr,fork UNIX-CLIENT:\"$DISPLAY\"` in a terminal, leave that running
3. Run XQuartz in a new terminal `open -a XQuartz`
4. Determine IP address for X-forwarding – `ifconfig | grep inet`
5. Run the docker image from the command line:

   ```
    docker run -ti --rm -u NRTfish -v <src data dir>:/data \
   -v <dst data dir>:/outdata -e DISPLAY=<ip addr>:<display #> \
   -v /tmp/.X11-unix:/tmp/.X11-unix alrikai/nrt bash
   ```

   Note that the $-u$ *name* option sets the username within the docker container. On one system, I have it set to NRTFish (since this is the username that is set in the Dockerfile), but it seems to work with the current user (*$(id -u)*) as well. e.g.

   ```
    docker run -ti --rm -u NRTfish \
   -v /Users/SamanthaNewton/Drive/NRTLabeling:/data \
   -v /Users/SamanthaNewton/Drive/NRTLabeling/DockerOutput:/outdata \
   -e DISPLAY=10.249.17.255:0 -v /tmp/.X11-unix:/tmp/.X11-unix alrikai/nrt bash
   ```

6. Run commands as detailed in section A.6.

A.6 Running the FishLabeler Application

In all cases, running the *docker run* command in the above sections will download (if required) and launch the docker container, build the application, and navigate to the fishlabeler directory. The next important step is to run the application – to do so, run the **LabelFish.sh** script, passing in 2 command-line arguments:

1. The path to the video to label (using the mounted path directory, i.e. starting from /data).
2. The path to the directory to write the video frames and labeling metadata (also using the mounted paths, i.e. starting from /outdata).

e.g. The following command would extract video `20130117144639.mts` and output all the resultant data (and metadata) to `/outdata/NRT/20130117144639`. Where exactly this is on your host filesystem depends on your docker run command arguments, specifically the <src data dir> and <dst data dir> arguments. If, for example, <src data dir> is `/home/alrik/Data`, then the input file would be at `/home/alrik/Data/NRT/20130117144639.mts`. Similarly, if the <dst data dir> argument is `/home/alrik/Data/NRTFish`, then the results would end up at `/home/alrik/Data/NRTFish/NRT/20130117144639`.

```
$ bash LabelFish.sh /data/NRT/20130117144639.mts /outdata/NRT/20130117144639
```

The **LabelFish.sh** script will extract the individual frames of the video and write them to the specified path as jpeg images, extract some video metadata, and then launch the labeling application. If there are already frames in the specified output directory, then the script will skip the frame extraction step. If using the script, the Fishlabeler application will pop up with the specified (i.e. newly-extracted or already-existing) video sequence selected after the frame extraction completes. In the event that the application is run manually (i.e. running the application executable directly), then there are two options:

1. No command-line arguments are provided: You should see a dialog UI pop up, and use that to navigate to the directory of frames you want to label. Once you've selected your target directory, the main window will pop up, and you can begin labeling.

2. The user passes in the target directory, which already has its frames and video metadata extracted, as is done in the **LabelFish.sh** script. Then no dialog UI will be displayed, and the user-specified video sequence will be opened. e.g. `./FishLabeler /data/NRTFishAnnotations/20130117144639`

Also, if you want to skip the frame extraction and metadata extraction step (i.e. if you've already gone through the extraction steps), then you can also just run the executable directly from the build directory.

A.7 Using the FishLabeler Application

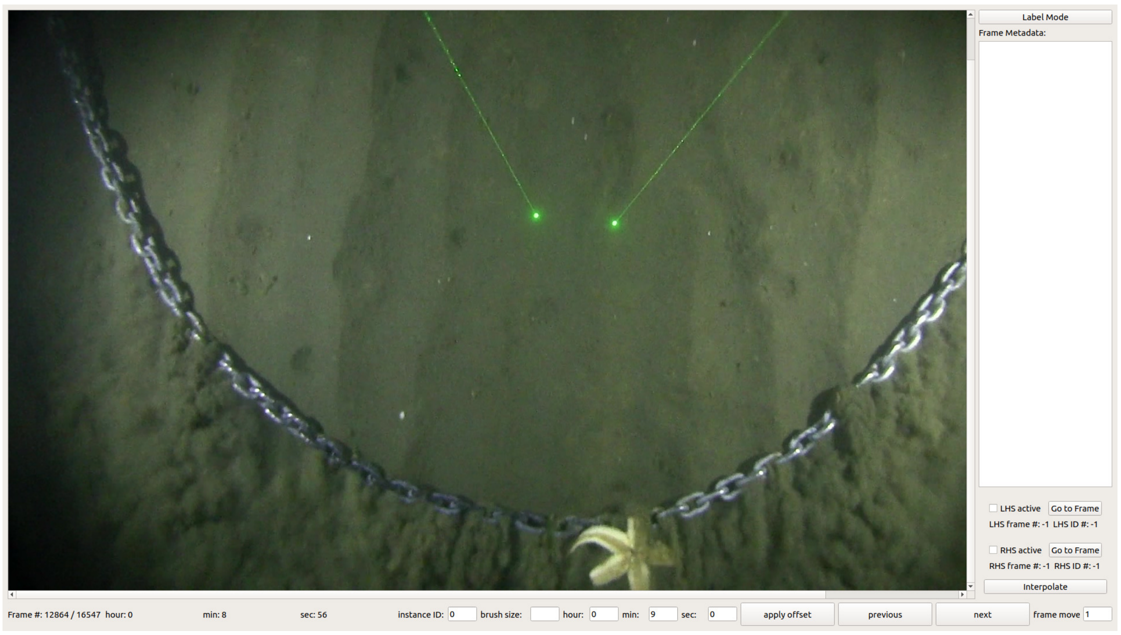
Fishlabeler is supposed to be relatively bare-bones, so it should be straightforward to use. The default window will look similar to appendix A.7. Different fish should be given different instance IDs, which correspond to different colored bounding boxes. The specific functionality provided by the Fishlabeler application is detailed in the following sections:

- Frame navigation: subsection A.7.1
- Segmentation (per-pixel labeling): subsection A.7.2
- Detection (bounding boxes): subsection A.7.3
- Detection interpolation: subsection A.7.4
- Hotkeys: subsection A.7.5

When labeling, don't bother with fish < 10 cm, as (in my experience) they are too small to be easily seen by eye. The laser range-finder is calibrated to be roughly 10 cm on the ocean floor, so this is a good heuristic to use for determining which fish to label and which to ignore.

A.7.1 Frame Navigation

The bottom panel has the frame navigation tools; the left side displays the current frame number and timestamp, the edit box for instance ID controls what ID the bounding box to be drawn is assigned, the timestamp edit fields allow jumping to a specific timestamp



LHS frame #: -1 LHS ID #: -1
RHS frame #: -1 RHS ID #: -1
Frame #: 12864 / 16547 hour: 0 min: 8 sec: 56 instance ID: 0 brush size: hour: 0 min: 9 sec: 0 frame move 1

Figure A.1: The Fishlabeler UI window of a beamtrawl video sequence frame (and starfish casualty).

(enter the timestamp, click the `apply offset` button), while `prev` and `next` buttons move through frames according to a fixed frame offset, as set in the edit field `frame move`, which defaults to 1. Corresponding hotkeys can be found in subsection A.7.5. The `brush size` edit field controls the annotation pixel size (i.e. for the bounding box border, or the number of pixels that each click corresponds to in segmentation).

A.7.2 Segmentation

Segmentation is the process of assigning labels to pixels, i.e. designating the labeled pixels as belonging to a groundfish, and the non-labeled pixels as being background (or anything not-groundfish). This differs from detection in that segmentation operates in pixel-space. As such, it is correspondingly more work-intensive to label, since one has to assign labels to every pixel of interest in the scene. However, segmentation also gives more information about the scene than detection does, so it is valuable to do. To perform segmentation in Fishlabeler, click the `Label Mode` button in the upper right until it says `Segmentation`, then clicking on the displayed frame will produce labels for the affected pixels, contingent on the current brush size and instance ID. An example is shown in appendix A.7.2. In the interest of full disclosure, segmentation annotations are extremely tedious to label, and despite having implemented the functionality for it in the application, I gave up on segmentation labeling after a few dozen frames.

A.7.3 Detection

Detection is the process of enclosing a target object within a (axis-aligned) rectangle, such that the entirety of the object is contained within the rectangle, and as few pixels that *don't* belong to the target object are included as possible. This is the weakest form of localization information, but is fast to draw and to process, and enables detection and tracking algorithms (which form the basis of many video analysis algorithms). The rectangle is commonly referred to as a `bounding box`. To draw detection annotations in Fishlabeler, ensure the annotation mode is not set to `Segmentation`, and then draw the bounding box by first clicking on the frame for the upper-left corner of the bounding box, dragging the mouse to the bottom right corner, then releasing the button. The bounding box will then be shown on the frame. Clicking on an existing bounding box outline and dragging it elsewhere will translate the bounding box according to the mouse movement rather than creating a new bounding box. An example labeled frame is shown in appendix A.7.3.

A.7.4 Interpolation

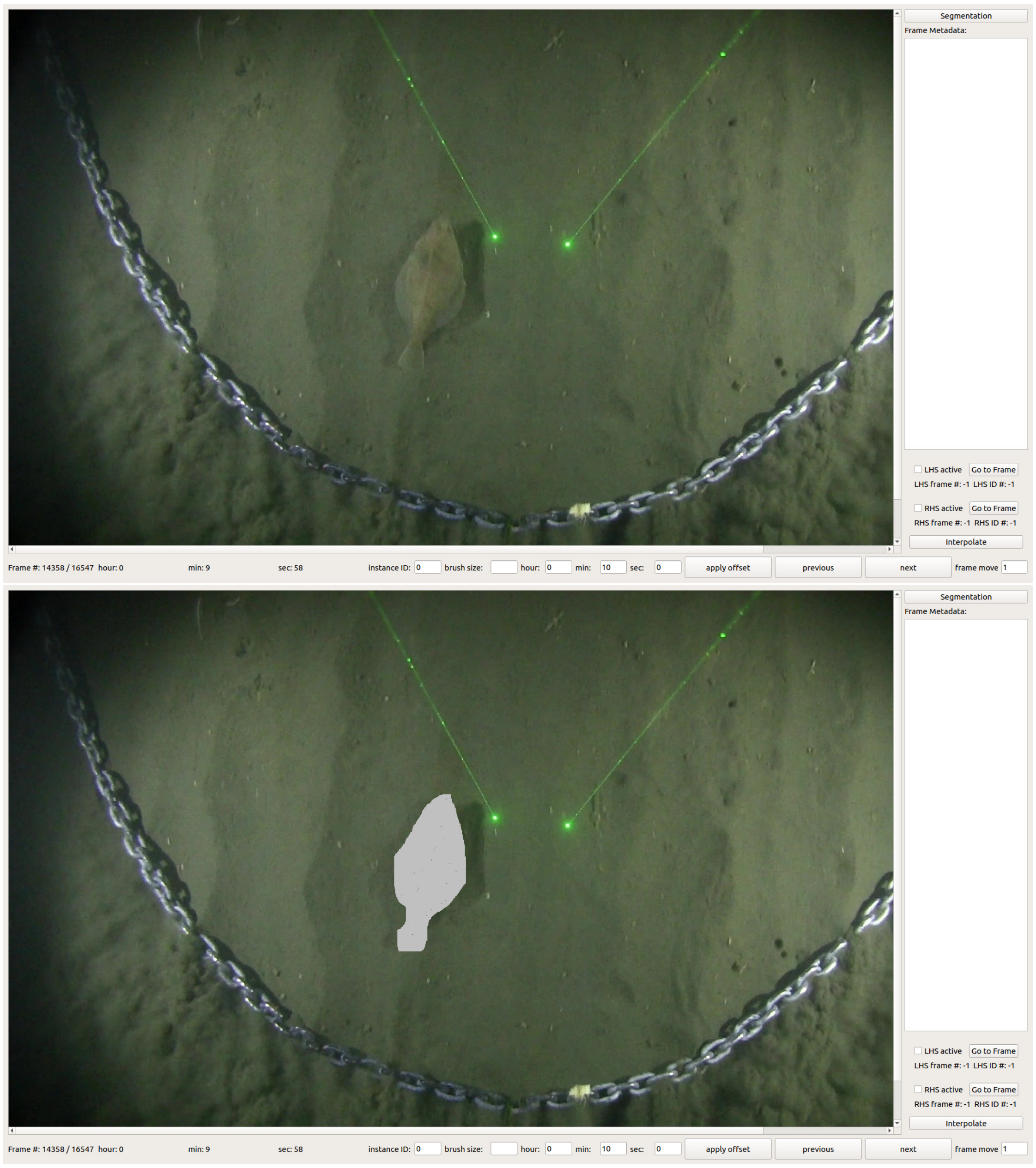
The aim of the interpolation module is to automate some of the tedious detection labels; currently, only linear interpolation is implemented, so it works best when the apparent fish motion is linear. Note that since the camera is also moving (in a non-linear fashion), linearity assumptions are limited, and the apparent fish size is contingent on where the fish is in the scene, and varies heavily based on the distance of the fish from the camera (if it swims closer to the camera, its apparent size increases). Thus, it is not a good idea to rely solely on interpolation, as it will often be inaccurate. However, it can give good results in certain situations (constant, linear motion), so it is still of use. I've found it to be best applied over relatively short time-scales, and/or when the fish is stationary and the boat is moving at a relatively constant rate.

To use the interpolation, the starting and ending bounding boxes have to be provided, and the interpolation logic will 'guess' the bounding boxes in between. To provide the starting (LHS) and ending (RHS) boxes, check the corresponding checkbox in the lower-right side of the window, then draw the box on the frame. This will register that annotation to either the LHS or RHS, based on which checkbox was chosen. Once a box has been registered, the frame index that the box is at, and the instance ID of the box will be displayed. Clicking the `Go to frame` button will change the current frame to the associated bounding boxes' frame. Once the LHS and RHS boxes have been chosen, click the `interpolate` button to compute the intervening frame's bounding boxes. Navigating to these frames should display them, and the user can adjust the boxes accordingly (either by translating the boxes by dragging them by their frame, or by undoing them (using `ctrl + z`, see subsection A.7.5 for hotkey details) and re-drawing, if necessary. The LHS and RHS instance IDs should match, since it is assumed that they refer to the same fish (and indeed, they should).

A.7.5 Hotkeys

- **n**: next frame, according to the current `frame move` value (hold the `n`-key down to move quickly)
- **p**: previous frame, according to the current `frame move` value (hold the `p`-key down to move quickly)
- **ctrl + z**: undo the last bounding box or pixel-label on the current frame. This can be used to erase existing annotations (e.g. hold down `ctrl`, then press `z` to remove detections / pixel labels)
- **ctrl + r**: redo the last bounding box or pixel-label on the current frame that was undone. This can be used to restore an undo operation; i.e. if you erroneously erased an existing annotations with `ctrl + z`, then ctrl + r will restore it. Once you navigate to a different frame, the re-do history will be lost however.
- **f**: toggle the frame display mode; either full-sized (which generally requires scrolling),

or zoomed-to-fit (no scrolling required, but the image is smaller to fit the screen).



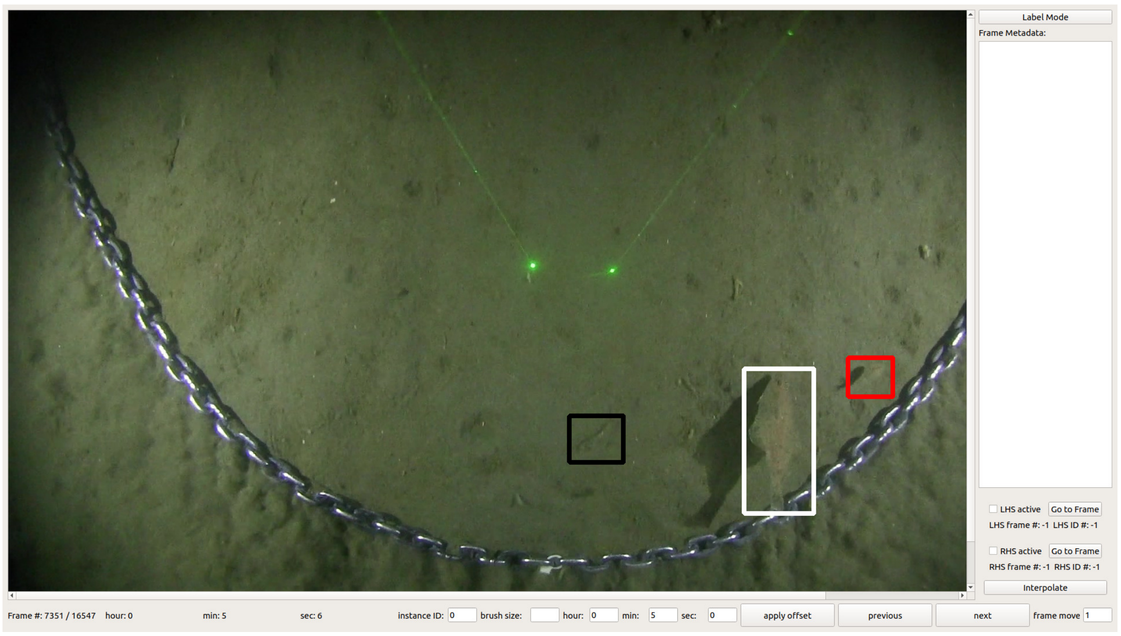
LHS frame #: -1 LHS ID #: -1
RHS frame #: -1 RHS ID #: -1

Frame #: 14358 / 16547 hour: 0 min: 9 sec: 58 instance ID: 0 brush size: hour: 0 min: 10 sec: 0 frame move 1

LHS frame #: -1 LHS ID #: -1
RHS frame #: -1 RHS ID #: -1

Frame #: 14358 / 16547 hour: 0 min: 9 sec: 58 instance ID: 0 brush size: hour: 0 min: 10 sec: 0 frame move 1

Figure A.2: Before and after screenshots of a segmentation. Note that the upper-right button is in `Segmentation` mode. The color of the segment corresponds to the instance ID (in this case, ID 0). The goal in creating a segmentation annotation is to cover the pixels belonging to the fish as accurately as possible.



LHS frame #: -1 LHS ID #: -1

RHS frame #: -1 RHS ID #: -1

Frame #: 7351 / 16547 hour: 0 min: 5 sec: 6 instance ID: 0 brush size: hour: 0 min: 5 sec: 0 frame move 1

Figure A.3: Screenshot of the Fishlabeler UI with three fish detections annotated. The different colored bounding boxes correspond to different instance IDs. The bounding boxes should cover the entirety of the fish, but be as tight as possible to the fish (i.e. not have extraneous non-fish background pixels with in the bounding box).

A.8 After Running the FishLabeler Application

The metadata results will be in the output folder based on how the docker command was run. There will be 3 folders with the labeling results, `Annotations`, `Detections`, `Metadata` for segmentation masks, detection bounding boxes, and text information respectively. These are not uploaded anywhere automagically, so (unless you really enjoy labeling), upload them somewhere before deleting anything. Currently, the defacto location is at `box.com` under the `NRT/Labeling Metadata` folder. After labeling a sequence, you can safely delete the video frames that get extracted to your system, as they can always be regenerated from the video file (and after you've uploaded the metadata, you can delete those folders too).