Automating Deep-Sea Video Annotation Using Machine Learning

Hanson Egbert and Lubomir Stanchev
Computer Science and Software Engineering Department
California Polytechnic State University
San Luis Obispo, CA, USA
hegbert@calpoly.edu stanchev@gmail.com

Benjamin Ruttenberg
Biological Sciences Department
California Polytechnic State University
San Luis Obispo, CA, USA
bruttenb@calpoly.edu alexandra.wolman@gmail.com

***Abstract*—As the world explores opportunities to develop offshore renewable energy capacity, there will be a growing need for pre-construction biological surveys and post-construction monitoring in the challenging marine environment. Underwater video is a powerful tool to facilitate such surveys, but the interpretation of the imagery is costly and time-consuming. Emerging technologies have improved automated analysis of underwater video, but these technologies are not yet accurate or accessible enough for widespread adoption in the scientific community or industries that might benefit from these tools. To address these challenges, we developed a website that allows us to: (1) Quickly play and annotate underwater videos, (2) Create a short tracking video for each annotation that shows how an annotated concept moves in time, (3) Verify the accuracy of existing annotations and tracking videos, (4) Create a neural network model from existing annotations, and (5) Automatically annotate unwatched videos using a model that was previously created. The website was seeded with 50 hours of high-resolution underwater videos that were generously provided by the Monterey Bay Aquarium Research Institute (MBARI). The biology students that were part of the project created more than 30,000 annotations that range over more than 20 concepts. About 3,000 of these annotations were then verified for accuracy by our marine biology experts. Using both validated and unvalidated annotations and automatically generated annotations from trackings, our software was able to count the number of *Rathbunaster californicus* (starfish) and *Strongylocentrotus fragilis* (sea urchin) with count accuracy of 97% and 99%, respectively, and F_1 score accuracy of 0.90 and 0.81, respectively.**

I. Introduction

MBARI and other deep-sea exploration organizations collect thousands of deep-sea underwater videos every day. Usually, this data is collected by Unmanned Underwater Vehicles (UUVs) that cross the ocean floors daily. Unfortunately, annotating these videos by a human is a very costly and lengthy process. For example, it took us about 600 hours of student annotations to annotate just 30 hours of videos. Therefore, the problem we are trying to solve is how to automate the annotation task. This includes creating tools for fast video annotations by humans and tools for automatic video annotations once a model has been trained. We also included tools to track the accuracy of human annotations and computer-generated annotations (via validation sets).

The huge backlog of underwater videos that are not annotated requires a new approach. One that allows marine biologists to annotate videos from anywhere using a web browser through a user-friendly interface. Once enough videos have been annotated, our approach also allows for automatic video annotation. This can be beneficial not only for deep-sea pre-construction and post-construction surveys, but also for a range of applications, such as analyzing drone videos for marine life or using stationary videos to analyze the effect of human-made artifacts, such as a desalination plants, on marine life [1].

Organizations that explore underwater marine life are struggling to annotate all their videos. The reason is that current tools (e.g., [2]) are slow, not versatile, and not much automation is possible. What makes the problem even more challenging is that a single frame may not be sufficient to identify a concept. For example, the angle of the camera or the distance to the object may make recognition hard or impossible. Moreover, additional information, such as the depth of the video or the pattern of movement may be required in order to make a correct identification. This is why our tool allows annotators to see a short video (six seconds or shorter) around the annotation point, called a *tracking video*, which includes a bounding box around the objects of interest. Moreover, our machine learning tool examines these tracking videos when identifying a concept in order to increase the accuracy of the algorithm. Another problem that we faced is that it is difficult to develop a web application that correctly identifies the frame in the video where an annotation is made. We believe that this may be related to the way the video is compressed and displayed by JavaScript. In order to fix this problem, we had to match the currently displayed frame in the web browser to the frames in the video around the annotation time in order to identify the correct frame.

There are many reasons why a comprehensive web-based deep-sea annotation tool with good automatic annotation capabilities has not been previously developed. First, this a niche area with limited funding. Second, the hardware (e.g., graphic processing units (s)) and good object detection algorithms, such as R-CNN [3], fast R-CNN [4], faster R-CNN [5], Yolo [6], and RetinaNet [7], have only recently been developed. We were lucky enough to receive a $200,000 grand from the California Energy Commission [8] and $50,000 in Amazon Web Services (AWS) credit. We used this money to develop a comprehensive website with good automatic annotation

978-1-7281-6332-1/20/$31.00 ©2020 IEEE
DOI 10.1109/ICSC.2020.00010

capabilities. Fourteen students and two faculties at California Polytechnic State University (Cal Poly) have worked on the project for about 18 months to develop the software tool. We utilized the AWS credits to deploy powerful instances with 4 GPUs and 64 virtual CPUs and state-of-the-art convolutional neural network models, such as RetinaNet.

When using our website, the user first selects the concepts of interest. They chose from a hierarchy of more than 2,000 underwater species. Next, they can select the video they want to annotate, watch it, stop it at any point and create bounding boxes around objects of interest and tag them with the appropriate concept name. Our software supports four lists of videos: “My In Progress Videos”, which keeps track of the videos that are currently annotated by the user, “Unwatched videos”, which contain no annotations, “Annotated Videos”, which have been fully annotated, and “In progress videos”, which are currently being annotated by someone. We use the *Kernelized Correlation Filter* algorithm [9] to create additional annotations from tracking the object that is being annotated. Our verification tab allows the user to verify the validity of both user-created and tracking annotations in a collection of annotations. Our reporting tab can show annotations sorted by video, concept, or user, where there are additional options to show only verified annotations or annotations that are marked as unsure. Tracking annotations are not displayed in the reporting tool. Finally, the models tab allows the user to create and train a model and use a model on an unwatched video to automatically annotate it. We use the RetinaNet [7] convolutional neural network as our annotation algorithm, where the initial weights are based on the COCO dataset [10].

In what follows, in Section II we go over related research. The main contributions of the paper are in the next three sections. In Section III, we describe the functionality of our website and the workflow of how to use it. In Section IV, we examine the technical details of how we built the website. This includes a novel algorithm for correctly assigning the category of a concept based on multiple frames of tracking the concept with a bounding box. Our experimental results are presented in Section V, while the summary and areas for future research are shown in Section VI.

II. Related Research

As [11] described, there is a trade-off between the accuracy and the speed of an object detection algorithm. One of the first highly successful algorithm to use convolutional neural networks was regional convolutional neural networks (R-CNN). It is a two-pass algorithm, where the first pass identifies about 2,000 regions of interest in the image using *selective search* [12] and the second pass transforms each region into a rectangle and then classifies it using a convolutional neural network (CNN). However, training and inference was slow. Two improvements: Fast R-CNN [4] and Faster R-CNN [5] were introduced later. Fast R-CNN speeds up the process by first using a CNN to generate a feature map. Then, the selective search algorithm works with the feature map instead of pixels from the image, which speeds up the process. Faster R-CNN eliminates the need for the selective search algorithm all together by using a convolutional neural network to select the objects of interest. An extension of Faster R-CNN is Mask R-CNN [13], which is able to segment the objects in the image. This means that instead of bounding boxes, the algorithm detects the precise curved boundary of each object inside the image.

An alternative approach to object detection is using a feed-forward network in a single pass. Such algorithms include *You Only Look Once* (YOLO) [6] and *Single Shot Detection* (SSD) [14]. The algorithms split the input image into grids and explores different bounding boxes in each grid cell. Although these approaches are very fast, the accuracy is not at good as the two-stage methods, such as Faster R-CNN.

Recently, the RetinaNet algorithm [7] was published. Although it is a one-stage convolutional neural network algorithm, it is able to achieve accuracy that is comparable with two-stage algorithms, such as Faster R-CNN. The algorithm addresses class imbalance during training by using a new focal loss function.

For our website, we experimented with Faster R-CNN, YOLO, and RetinaNet. As expected, using RetinaNet we got reasonable training times (e.g., about four hours to train the network on a single concept) and good accuracy. Faster R-CNN and YOLO were slower and the accuracy numbers were not as good.

There is great utility for our approach in the marine sciences because many marine research projects utilizes video or imagery. Some projects that use still images have begun to employ machine learning to automate the task of identification of plankton [15], megafauna, such as sharks and whales [16], [17], birds [18], and even corals [19], but few projects have been successful in applying these approaches to video. There is a wide range of marine research and monitoring projects that use videos, including measuring the size structure of fishes [20], evaluating the impacts of fishery closures on target populations [21], monitoring and evaluating human impacts in deep-sea ecosystems [22], [23], [24], surveying pelagic ecosystems [25], and tracking biodiversity [26], among many others. The videos that are generated require a great deal of time to process, which adds cost, slows data analysis, and limits the data that researchers can extract and analyze, all of which reduces the potential impact the data can have on our understanding and managing of ecosystems. While we are developing this tool for a single, specific project, the potential applications of this tool across marine science and any other discipline that collects video data are wide and varied.

III. Our website

We built our website on AWS using nodeJS and React. We used PostgreSQL as our database back-end. The location of our website is `www.deepseaannotations.com`. The website is password protected because the videos are property of MBARI and cannot be shown without their permission. The software is developed under Apache license and it can be downloaded from:

github.com/video-annotation-project. The website has six tabs: Concepts, Collections, Annotate, Report, Model, and Account, which we cover next.

A. Concepts Tab

The concepts tab allows us to select the concepts of interest. The concepts are displayed in a tree hierarchy, where there is an image associated with each concept. These hierarchy corresponds to the taxonomic hierarchy of marine-life organisms. The concept tree is initially populated from a JSON file. The user can navigate the tree or directly type the name of the concept. There is no limitation to the number of concepts that can be selected. All selected concepts are put in the user's *concept bag*.

B. Collections Tab

The Collections tab has three sub-tabs: Annotations, Concepts, and Videos. The Annotation Collection sub-tab allows the user to create a collection of annotations or add annotations to an existing collection. First, users, videos, and concepts are selected. Next, all annotations from these selections are displayed. As expected, there is an option to choose all users, all videos, or all concepts. For videos and concepts, there is also the option to select from an existing video collection or concept collection, respectively. Once the user has described the annotation collection based to the annotators, videos, and concepts, they have the option to select whether to include annotations from tracking to the collection. On average, we store about 55 tracking annotations (about three seconds of tracking video) for each user annotation. Annotation collections are used when working with models. For example, we can use an annotation collection to train a model. Similarly, when the software makes predictions on a video, the result is stored in an annotation collection.

The Concept Collection sub-tab allows the user to create custom collections of concepts. The user can only select concepts from their concept bag that is created through the Concept tab. If the user wants to add a concept that is not part of their concept bag to a collection, then the concept needs to be first added to the concept bag. Concept collections are useful when creating annotation collections.

Lastly, the Video Collection sub-tab allows the user to create collections of videos. When the user is adding a video to a collection, they are allowed to play the video and see video information. Video information includes the start/end time of the video, the start and end depth in meters, the video description, summary of the concepts that were annotated in the video, and the density of the concepts in the video (e.g., how many sea stars can be seen in the video per kilometer). We found this sub-tab useful because we had different sets of videos: for example, videos that are high quality, videos that contain the species that we are interested in, and so on.

C. Annotate Tab

The Annotate tab has two sub-tabs: Videos and Verify. The Videos sub-tab is used to annotated videos. It has the capability of playing a video at different speeds, stopping a video, and annotating objects in the video using rectangular bounding boxes. The software allows to only annotate species that are in the concept basket, but it also allows the user to quickly add new concepts to the concept basket. When an annotation is performed, the user has the option to add a comment to the annotation or mark it as uncertain so that it can be later reviewed by a different annotator. The tool keeps track of which videos are currently being annotated and which videos have already been annotated. This allows annotators to choose to work on new videos that have not been previously annotated and the website gives a warning when multiple annotators try to annotate the same video.

The Verify sub-tab is used to verify an existing collection of annotations. The user can select whether to include annotations from tracking and whether to verify tracking videos. The annotations from the collection are shown to the user one by one. The user has the option to move the bounding box, change the label of the annotated concept, or even create a new annotation. For each frame, all available annotations are displayed. This includes annotations outside the annotation collection. The reason is that we want to make sure that all frames that are used as input to a model contain all relevant annotations. Four colors are used to display the different bounding boxes – see Figure 1. Red is used to display the annotation that we are hovering over with the mouse. This includes the option to delete the annotation. Green is used for annotations that are already verified and are part of the collection. Blue is used for annotations that are outside the set of concepts for the annotation collection. Finally, orange is used for the current bounding box. The tool also contains the option to jump between an annotation and the corresponding tracking video. It is recommended that an annotation collection is verified by a human for accuracy before it is used to train a neural network model.

D. Report Tab

The report tab shows all annotations, verified only annotations, or unsure annotations sorted by video, concept, and/or annotator. The result is shown in a tree that can be expended or collapsed. Once an annotation is displayed, the user has the option to modify it, delete it, or watch the tracking video that is associated with the annotation. This tab can be used to examine the work that is done by the different annotators because it shows counts relative to the chosen sorting order. Alternatively, if the result is sorted by concept, then we can see the total number of annotations for each concept (see Figure 2). This tab is also useful as a learning tool because it can display all the annotations with trackings for each concept. The similarity between this tab and the verify tab are obvious: both tabs can be used to view and change annotations. However, the verify tab shows the annotations one at a time and its main purpose is to double-check our work. Conversely, the report tab is useful not only to examine individual annotations, but also see a summary of the annotation count by concept, annotator, or video (similar to the `cube` operator in relational databases).

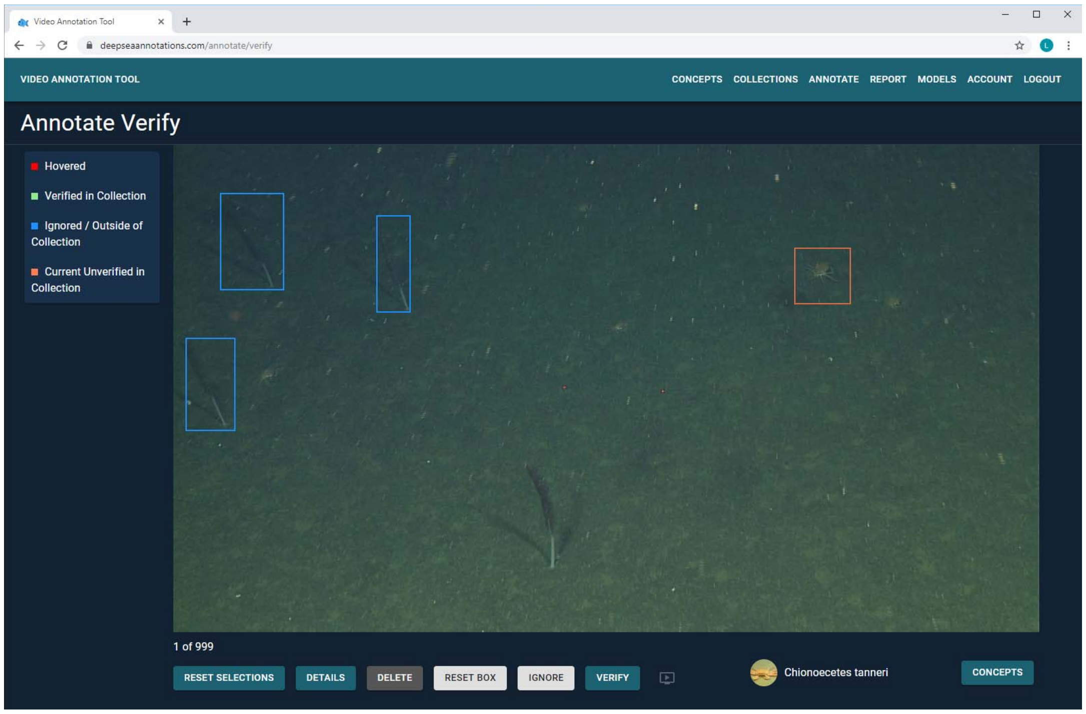


Fig. 1. The Verify sub-tab.

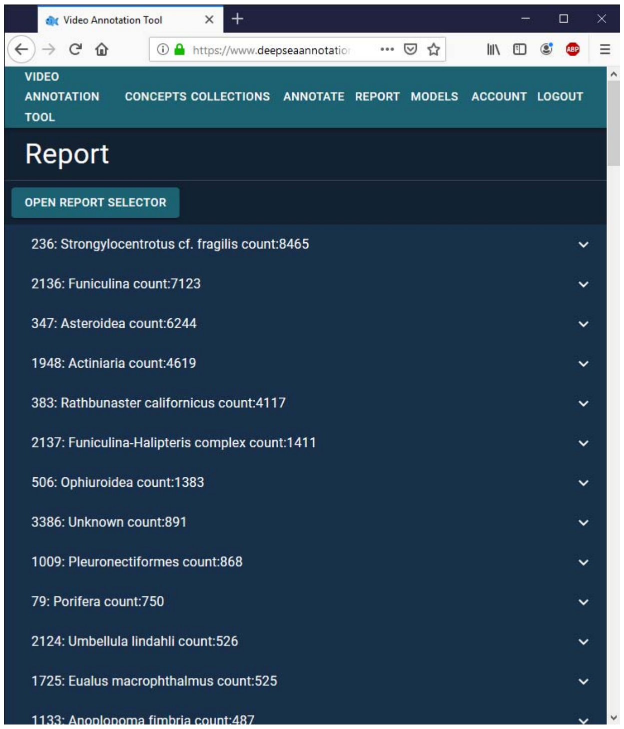
E. The Model Tab

The model tab shows all available models. There is a "+" in the top right of the tab that can be used to create a new model. If pressed, a new popup window opens where the user types the model name, the concepts that are part of the model (directly specified, or specified using a concept collection) and the verification videos. The verification videos must be videos that are fully annotated and that contains some of the model's concepts. The verification videos must be different from the videos that were used to train the model. The videos can be chosen from a list or an existing video collection can be selected. After a model is trained, the verification videos are used to verify the accuracy. For example, for each concept of the model, the following values are calculated: the number of true positives, the number of false negatives, precision, recall, F_1 score, the number of predicted concepts, the ground truth number of concepts as annotated by users, and the count accuracy. These numbers are used to access the quality of the model and guide the user if additional training data is needed.

For each model, there is a "train" button that creates a new version of the model. The initial weights are based on the COCO dataset [10]. Once the button is pressed, a new popup window appears that asks for the name of the annotation collection, the number of epochs to train, and the number of images to use. Note that the annotations that are used for training cannot be from the verification videos. Next, the button is changed to "training". If the "training" button is pressed, information about the training (e.g., current epoch for the training stage or video being annotated and percent progress for the verification stage). Once the training has finished, a new version appears under the model. The web page displays tree of versions for each model. For example, Version 2.3 is the third version that is created from the second version of the model.

Each model version has a "predict" button that allows us to use the trained model version to annotate a new video. The result of running "predict" is generating automatic annotations on the new unwatched video and a new computer-annotated video that shows the annotated concepts with bounding boxes throughout the video. There is also a "video" button for each model version that shows the videos that are generated for the specific model version. Annotated videos are generated for each of the verification videos. The annotated video shows the annotated concepts with bounding boxes from the moment they appear in the video to the moment they disappear from the video with the label of the concept and confidence that the prediction is correct. Note that for each concept appearance, only a single annotation is generated and the rest of the annotations in the annotated video are generated using the tracking algorithm [9]. More details on how the annotation videos and the automatic annotations are generated are presented in the next section.

An information button is also associated with each model version. When pressed, we can see the precision, accuracy, F_1 score, and count accuracy of each concept on the verification videos. If these numbers are good, then we can assume that our model is good. If they are not, then we have two options. First, we can verify the output annotations of the model using the



Report

236: Strongylocentrotus cf. fragilis count:8465
2136: Funiculina count:7123
347: Asteroidea count:6244
1948: Actiniaria count:4619
383: Rathbunaster californicus count:4117
2137: Funiculina-Halipteris complex count:1411
506: Ophiuroidea count:1383
3386: Unknown count:891
1009: Pleuronectiformes count:868
79: Porifera count:750
2124: Umbellula lindahli count:526
1725: Eualus macrophthalmus count:525
1133: Anoplopoma fimbria count:487

Fig. 2. The Report tab shows the top concepts relative to number of annotations.

verify sub-tab. Then, we can retrain the model with the verified annotations. In the spirit of reinforcement learning (e.g., [27], [28]), corrected annotations are given a greater weight when creating the new model version. The second option would be to create a new model version by training the existing weights using an additional annotation collection.

F. Account Tab

Lastly, the account tab has three sub-tabs: Profile, Create Users, and Users. The Profile tab allows the user to change the current password. The Create User tab allows us to create a new user, which can be an annotator or an admin. Only admin users have access to some of the functionality, such as training models. Lastly, the Users tab can be used to monitor the work of the annotators. Specifically, it can show the number of annotations for each user, concept, and time period.

G. Workflow

A rough overview of the website workflow is shown in Figure 3. We have used a double rectangle to denote the terminal state of the workflow. First, the annotators will select the concepts that they care about in their concept basket. Next, they will annotate multiple videos with the selected concepts. One or more senior annotators can then validate the annotations for accuracy and make sure that there are no mistakes or omissions in the annotated frames. The next step is to create a model with the important concepts and train it using part of the created annotations. Note that one or more videos

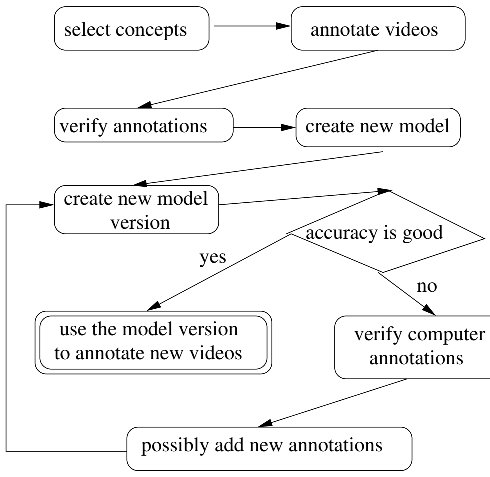


Fig. 3. Website workflow.

must be designated as verification videos and annotations from these videos should not be used for training the model. Once the first model version is produced, the user will check the accuracy against the verification videos. This accuracy can me measured as the F_1 score or the count accuracy for the different concepts. If this accuracy is satisfactory, then we have built a good model version and we can use it to automatically annotate new videos. If it is not satisfactory, then we can manually verify the computer-generated annotations and possibly add new annotations to the model. We have also found that watching the computer-generated annotated video is an efficient way to "debug" the model version and determine witch concepts have been incorrectly labeled. Providing more annotations for these concepts usually leads to improvement in accuracy.

IV. Website Internals

The heart of our website is the algorithm that creates the computer annotations. It is shown in Algorithm 1. The input to the algorithm is a trained model (i.e., a CNN with trained weights) and an unwatched video. The model is trained on a concept collection using an annotation collection. The algorithm produces a set of annotations. An annotation is characterized by the frame ID, bounding box (`x, y` pixel coordinates of the top left and bottom right corner), object ID, concept ID, and confidence. The concept ID identifies the concept in the video, while a new object ID is created for each occurrence of a concept in the video. For example, if a starfish appears in the video and then it disappears after few frames, than this is one occurrence of the concept and it is assigned a unique object ID. Our algorithm also maintains an array of current trackings. For each tracking, we store the concept name of the object that is being tracked, a unique object ID, the the bounding box for each fame, and the computer generated annotation for each 10th frame (this was chosen to make the algorithm faster). The goal of the tracking array is ensure that all annotations from the same tracking are tagged with the same concept ID.

Algorithm 1: `create_computer_annotations`

```
   Data: video, model
   Result: annotations
 1 annotations ← []
 2 trackings ← []
 3 for frame ∈ video.getFrames() do
 4     for tracking ∈ trackings do
 5         if the object in tracking is present in
             frame then
 6             tracking.addFrame(frame)
 7         end
 8         else
 9             remove tracking from trackings
10         end
11     end
12     if frame.getNumber() % 10 = 0 then
13         mAnn ←
           model.getAnnotations(frame)
14         update(mAnn,trackings,frame)
15         annotations ← annotations ∪ mAnn
16     end
17 end
18 return calibrate(annotations)
```

Algorithm 1 starts by initializing the array of annotations and trackings as empty arrays (Lines 1-2). Next, our algorithm performs a one-pass scan of all the frames in the videos (Line 3). This implies that the algorithm is linear and relatively fast. In practice, it takes about 30 minutes to automatically annotate a 15-minute video. Next, we iterate through all our current trackings (Line 4) and check if tracking extends to the current frame (Line 5). If this is the case, then we add the current frame to the tracking (Line 6). Otherwise, the tracking has ended and accordingly we remove it from the list of current trackings (Line 9). In order to make the algorithm fast, we only annotate every 10$^{\text{th}}$ frame (Line 12). The `mAnn` variable stores all the annotation for the current frame (Line 13). Line 14 updates the tracking data using these annotations, while Line 15 adds the computer generated annotations to the set of annotations. Lastly, Line 18 calibrates the annotations by picking the concept with the highest average confidence among each tracking and then tagging all the annotations along the tracking with this concept.

The `create_computer_annotations` method calls two auxiliary methods. The first one is the `update` method, which updates the tracking data. The method is called with the frame number and all the computer annotations and tracking data for the frame. We first iterate over all the annotations (Line 1) and check if there is an overlap between the bounding box of an existing tracking and a computer annotation. We consider two bounding boxes overlapping if the overlap area is more than 20%. Line 4 updates the annotation with the ID of the object that is being tracked, while Line 5 add the

Algorithm 2: `update`

```
   Data: annotations,trackings,frame
   Result: Updates trackings
 1 for annotation ∈ annotations do
 2     if annotation.getBoundingBox() overlaps
         with tracking.getBoundingBox(frame)
         for some tracking in trackings then
 3         objectID ← tracking.getObjectID()
 4         annotation.objectID ← objectID
 5         tracking.addAnnotation(annotation)
 6     end
 7     else
 8         tracking  ← new tracking starting at
           frame and bounding from annotation
 9         tracking.addAnnotation(annotation)
10         annotation.objectID ←
           tracking.objectID
11         trackings ← trackings ∪ tracking
12     end
13 end
14 for tracking ∈ trackings do
15     if tracking.getBoundingBox(frame) does
         not overlaps with some annotation in
         annotations and there has not been a match
         for the last 30 frames then
16         remove tracking from trackings
17     end
18 end
```

annotation to the tracking. Line 7 covers the case when there is an an object that is recognized by the prediction algorithm in the current frame, but there is no tracking for it. In this case, Line 8 creates a new tracking for this object. Note that this automatically generates a new object ID. Line 9 adds the annotation to the tracking. Lines 10 sets the object ID for the annotation to the ID of the object that is being tracked. Line 11 adds the tracking to the set of current trackings. Lines 14-18 cover the case when we keep tracking an object for 30 frames without the object being recognized by the prediction software. In this case, we are assuming that the object that is being tracked is no longer recognized and therefore we stop tracking it.

Lastly, the `calibrate method` reassigns the concept labels of the computer-generated annotations. In particular, the method first finds all annotations that trace an object (Lines 1-2). We then find the average confidence for each concept in the tracking, pick the concept that has the highest confidence (Line 4), and use this concept to relabel the annotations along the tracking (Line 5). For example, if along a tracking the machine-learning algorithm recognizes a concept `A` with confidence 0.2, a concept `B` with confidence 0.3 and then a concept `A` with confidence 0.5, then we will relabel all concepts as `A` because the average confidence for `A` is 0.35, while the average confidence for `B` is 0.3. Note that in our

Algorithm 3: `calibrate`

```
Data: annotations
Result: calibrated annotations
1 for objectID ∈ annotations.getObjectIDs() do
2     nAnnotations ← all annotations with objectID
3     find average confidence for each concept in nAnnotations
4     conceptID ← concept with highest average confidence
5     change the conceptID of all annotations in nAnnotations to conceptID
6 end
7 return annotations
```

database we store only the annotation in the middle for each tracking, while the the other annotations are derived using the *Kernelized Correlation Filter* tracking algorithm.

V. Experimental Evaluation

A. Training Process

Our training script runs on a AWS EC2. Specifically, we used a g3.16xlarge, which has 4 NVIDIA Tesla M60 GPUs (32 Gbs GPU memory). The model trains on information a user selects: epochs, annotation collection, and number of training images. We use a custom image generator, which feeds our model. Our generator downloads multiple images at a time. The generator checks if the image exists in our training server, and when the image is not found, it downloads it from our S3 bucket. Once there are enough images for a batch, our model starts training. While training on the first batch, the generator continues to prepare images for the next batch. Parallelizing image retrieval, checking existing images, and training immediately made the process run fast.

B. Validation

We compute a model's accuracy on a set of verification videos. Those are videos that are not used to train the model. After each training job, all verification videos are run against the model, and compared with user annotations. If a bounding box in the model overlaps with a user's box by 20% or more, then this a true positive (TP), otherwise it is a false positive (FP). If our model does not place a box that overlaps with a user's box, then this is a false negative (FN). We use these numbers to calculate precision (P), recall (R), and F1 score.

$$P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN} \quad F1 = \frac{2 \cdot P \cdot R}{P + R}$$

Finally, we take the number of human annotations (*userCount*) and compare it with the number of objects our model (*modelCount*) detected for each species.

$$count\ accuracy = 1 - \frac{|modelCount - userCount|}{max(userCount, modelCount)}$$

This gives us an idea of how well our model is doing, without the need to watch the video ourselves. For further investigation, we also generate the verification video with both human and model annotations.

C. Experiments

Our team was interested in comparing two settings for training a model:

1) User annotations only and
2) User annotations and tracking annotations.

We used 5,000 random annotations of each concept from the collection, 1280x720 images, a batch size of eight, and three epochs for each training session. A session took on average two hours to train the model. Each model was trained twice.

The current standard is to train on only human annotations. This setting requires a lot of work from biologists, but the annotations are more consistent and accurate than tracking annotations. The second setting adds tracking annotations to the set. For each user annotation, the tracking algorithm generates, on average, 55 additional annotations. So, on average, we have access to 55 times more annotations than the first setting.

D. Experimental Results

Tables I and II show the results from the two settings on a verification video. The first setting, trained on only user annotations, does very well on identifying starfish, but not on the sea urchin. After inspecting our user's annotations, the starfish frames were annotated very well. They are big, easy to capture, and do not appear in clusters. The sea urchin is the opposite. A single biologist is easily overwhelmed, and can miss them. Our tracking algorithm generates annotations on every frame, so the biologist does not need to do so. With the addition of these annotations in our collection, we were able to reach very high count accuracy on both (over 95%).

VI. Conclusion and Future Research

In this paper, we outlined our work on the Deep-Sea Annotations project. We created a website that can be used to perform human annotations, human verification of annotations, and computer annotations of deep-sea videos. We showed the usability of the website by using it to create more than 30,000 annotations and then verify about 3,000 of them. The experimental results show that our approach is promising because our algorithm was able to determine the density of both sea urchins and starfish with very small count error.

One area of future research is to allow our algorithm to classify objects in a hierarchical way. For example, if our algorithm is not sure about the type of sea pan that is displayed in a bounding box (e.g., *funiculina* vs *funiculina-Halipteris complex*), then it can just use the *funiculinidae* label, which is the name of the super concept, to classify an object.

Acknowledgment

This work has been sponsored by the California Energy Commission, Program: EPIC, Agreement Number: EPC-17-029. We would also like to acknowledge the general support

TABLE I
ACCURACY RESULTS ON VIDEO 86: TRAINED ON 5,000 USER ANNOTATIONS ONLY

Species Name	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**model count**	**user count**	**count accuracy**
Rathbunaster californicus	134	11	9	0.924	0.937	0.931	145	145	100%
Strongylocentrotus cf. fragilis	77	12	35	0.865	0.688	0.766	89	140	63.6%

TABLE II
ACCURACY RESULTS ON VIDEO 86: 5,000 USER & TRACKING ANNOTATIONS

Species Name	**TP**	**FP**	**FN**	**P**	**R**	**F1**	**model count**	**user count**	**count accuracy**
Rathbunaster californicus	126	14	13	0.900	0.906	0.903	140	145	96.6%
Strongylocentrotus cf. fragilis	109	30	21	0.784	0.838	0.810	139	140	99.3%

of Amazon Web Services (AWS), which provided $50,000 in AWS credits. This work would also not be possible without the help of the Monterey Bay Aquarium Research Institute (MBARI), which provided 50 hours of high-definition breathtaking underwater videos. Finally, we would like to thank the following Cal Poly San Luis Obispo students: Ishaan Jain, Samantha Gunzl, Jacob Territo, Kyle Maxwell, Kyaw Soe, Trace Rainbolt, Kevin Label, and Justin Cho, who wrote the website software.

REFERENCES

[1] I. S. Al-Mutaz, "Environmental impact of seawater desalination plants," *Environmental Monitoring and Assessment*, 1991.

[2] MBARI, "Video Annotation and Reference System (VARS)," *https://www.mbari.org/products/research-software/video-annotation-and-reference-system-vars/*, 2019.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *https://arxiv.org/abs/1311.2524v5*, 2013.

[4] R. Girshick, "Fast R-CNN," *https://arxiv.org/abs/1504.08083*, 2015.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *https://arxiv.org/abs/1506.01497*, 2016.

[6] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *https://arxiv.org/abs/1804.02767*, 2018.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," *https://arxiv.org/abs/1708.02002*, 2017.

[8] P. Lubomir Stanchev and C.-P. Benjamin BenjaRuttenberg, "Lowering costs of underwater biological surveys to inform offshore renewable energy," *California Energy Commission EPIC Grant EPC-17-029*, 2018.

[9] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[10] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollar, "Microsoft COCO: Common Objects in Context," *https://arxiv.org/abs/1405.0312*, 2014.

[11] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/Accuracy Trade-offs for Modern Convolutional Object Detectors," *https://arxiv.org/abs/1611.10012*, 2017.

[12] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[13] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," *https://arxiv.org/abs/1703.06870*, 2017.

[14] W. LiuEmail, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," *European Conference on Computer Vision*, pp. 21–37, 2016.

[15] J. Y. Luo, J. Irisson, B. Graham, C. Guigand, A. Sarafraz, C. Mader, and R. K. Cowen, "Automated Plankton Image Analysis using Convolutional Neural Networks," *Limnology AND Oceanography Methods*, vol. 16, no. 12, pp. 814–827, 2018.

[16] A. P. Colefaxa, P. A. Butchera, D. E. Pagendam, and B. P. Kelaher, "Reliability of Marine Faunal Detections in Drone-based Monitoring," *Ocean and Coastal Management*, vol. 174, pp. 108–115, 2019.

[17] D. Risch, T. Norris, M. Curnock, and A. Friedlaender, "Common and Antarctic Minke Whales: Conservation Status and Future Research Directions," *Frontiers in Marine Science*, vol. 6, 2019.

[18] R. A. Orben, A. B. Fleishman, A. L. Borker, W. Bridgeland, A. J. Gladics, J. Porquez, P. Sanzenbacher, S. W. Stephensen, R. Swift, M. W. McKown, and R. M. Suryan, "Comparing Imaging, Acoustics, and Radar to Monitor Leach's Storm-petrel Colonies," *PeerJ*, vol. 7, 2019.

[19] I. D. Williams, C. S. Couch, O. Beijbom, T. A. Oliver, B. Vargas-Angel, B. D. Schumacher, and R. E. Brainard, "Leveraging Automated Image Analysis Tools to Transform Our Capacity to Assess Status and Trends of Coral Reefs," *Frontiers in Marine Science*, vol. 6, 2019.

[20] T. B. Letessier, J.-B. Juhel, L. Vigliola, and J. J. Meeuwig, "Low-cost Small Action Cameras in Stereo Generates Accurate Underwater Measurements of Fish," *Journal of Experimental Marine Biology and Ecology*, no. 466, pp. 120–126, 2015.

[21] J. S. Goetze, S.D.Jupiter, T. J. Langlois, S.K.Wilson, E.S.Harvey, T.Bond, and W. Naisilisili, "Diver Operated Video Most Accurately Detects the Impacts of Fishing within Periodically Harvested Closures," *Journal of Experimental Marine Biology and Ecology*, vol. 462, pp. 74–82, 2015.

[22] F. Althaus, A. Williams, T. A. Schlacher, R. J. Kloser, M. A. Green, B. A. Barker, N. J. Bax, P. Brodie, and M. A. Schlacher-Hoenlinger, "Impacts of Bottom Trawling on Deep-coral Ecosystems of Seamounts are Long-lasting," *Marine Ecology Progress Series*, vol. 397, pp. 279–294, 2009.

[23] A. Cánovas-Molina, M. Montefalcone, G. Bavestrello, A. Cau, C. N. Bianchi, C. Morri, S. Canese, and M. Bo, "A New Ecological Index for the Status of Mesophotic Megabenthic Assemblages in the Mediterranean based on ROV Photography and Video Footage," *Continental Shelf Research*, vol. 121, pp. 13–20, 2016.

[24] V. A. I. Huvenne, B. J. Bett, D. G. Masson, T. P. L. Bas, and A. J. Wheeler, "Effectiveness of a Deep-sea Cold-water Coral Marine Protected Area, Following Eight Years of Fisheries Closure," *Biological Conservation*, vol. 200, pp. 60,69, 2016.

[25] P. J. Bouchet and J. J. Meeuwig, "Drifting Baited Stereo-videography: A Novel Sampling Tool for Surveying Pelagic Wildlife in Offshore Marine Reserves," *ECOSPHERE*, vol. 6, no. 8, 2015.

[26] A. W. J. Bicknell, B. J. Godley, E. V. Sheehan, S. C. Votier, and M. J. Witt, "Camera technology for monitoring marine biodiversity and human impact," *Frontiers in Ecology and the Environment*, vol. 14, no. 8, pp. 424–432, 2016.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *https://arxiv.org/abs/1312.5602*, 2013.

[28] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, 2018.