

Ultimate Vision: A System to Autonomously Track An Ultimate Frisbee in Video Frames

Mallika Parulekar*
Stanford

Yash Kankariya*
Stanford

Abstract

Tracking a fast-moving Ultimate Frisbee in real-world game footage presents unique challenges due to the frisbee’s small size, rapid motion, and frequent occlusions. We introduce Ultimate Vision, a hybrid tracking system designed to localize the frisbee over multiple frames using a combination of zero-shot detection models and a custom-trained ResNet with a regression head. Starting from a single annotated frame, our pipeline auto-regressively tracks the frisbee by applying dynamic region-of-interest cropping around the predicted location. We integrate Grounding DINO and the Segment Anything Model (SAM) for segmentation, and improve robustness by training a modified ResNet-18 (CoordResNet18) to estimate the frisbee center within cropped patches. Through extensive experiments, we demonstrate that smaller crops significantly boost segmentation accuracy but suffer from tracking drift if the frisbee leaves the frame. To address this, we implement fallback strategies—resorting to full-frame detection or multi-stage cropping—to recover from failures. Our system outperforms baseline methods across challenging scenarios in Ultimate gameplay footage.

GitHub link for code

1. Introduction

Ultimate Frisbee is a fast-paced, non-contact team sport that has grown rapidly in popularity across collegiate, club, and professional levels worldwide. In Ultimate, two teams compete to pass a flying disc (“frisbee”) from player to player, advancing it toward an opponent’s end zone to score points. Because possession changes only when the disc is dropped or intercepted, tracking the exact trajectory of the disc in video footage is crucial for performance analysis, strategy development, and highlight generation. Traditional video annotation methods—relying on manual frame-by-frame labeling—are laborious, error-prone, and infeasible

at scale for tournaments or extended gameplay.

Autonomous tracking of an Ultimate Frisbee presents a unique set of challenges. First, the disc often subtends only a few pixels in standard broadcast or handheld camera resolutions, making it difficult to distinguish from background noise. Second, the disc’s bright white or neon color may blend with player jerseys, boundary lines, or on-field markings, leading to ambiguous detections. Finally, the frisbee’s rapid accelerations and abrupt changes in direction can produce motion blur, further complicating reliable detection.

In this work, we introduce Ultimate Tracker, a frisbee system to track the frisbee over multiple frames. Our approach begins by scraping diverse Ultimate Frisbee footage from YouTube and manually annotating each frame to build a high-quality training dataset. We then introduce a preprocessing step: using the disc’s detected initial position to define a tight crop around the frisbee, dramatically reducing search space and improving detection confidence. During tracking, this crop is dynamically updated via spatial heuristics. We then run our perception pipeline of GroundingDino and SAM to detect the exact location of the frisbee within the crop. Furthermore, we train a ResNet to predict the location of the frisbee within the crop, so that we can further tighten the crop and improve segmentation accuracy.

2. Problem Statement

Our system takes as input a sequence of RGB images—extracted frames from a short game segment—and outputs, for every frame, the pixel coordinates of the disc’s center. All frames are stored as individual image files, and both ground-truth annotations and model predictions are recorded in a simple CSV file listing each frame’s filename alongside its x and y center positions. To measure performance, we compare predicted and true centers by their Euclidean distance in pixels and report the fraction of frames whose error falls below a small tolerance (10 pixels).

3. Related Works

Object tracking in sports videos presents unique challenges due to the fast motion, frequent occlusions, and small size of key objects such as balls or discs. Prior work has addressed these challenges through a combination of detection, temporal modeling, and physics-based constraints.

Small Object Tracking in Sports. Zhang et al. [1] proposed a golf ball tracking system that leverages a CNN-based detector combined with a Kalman filter. The detector first localizes the ball in a small crop, and the Kalman filter then predicts its position in the next frame, enabling real-time performance and maintaining focus on the object despite fast motion. Similar principles have been applied in basketball and soccer tracking systems, where player and ball positions are estimated using learned motion priors and spatial context [2, 3]. These systems motivate our decision to center a dynamic crop on the frisbee and update it over time using predictions from a learned model.

Backbone Networks for Localization. ResNet [4] is widely used as a backbone for both detection and regression tasks due to its strong feature representation and residual connections, which facilitate stable training. In our work, we adapt ResNet-18 for center-point regression by extending it to accept five-channel inputs (RGB + X + Y coordinate maps), which provide explicit spatial information. Prior work has shown that including positional encodings or coordinate channels improves localization accuracy, as seen in [5] and keypoint estimation [6]. This motivated our decision to inject coordinate information directly into the network input, but also highlighted a limitation: while ResNet is effective at coarse localization, its performance can degrade when the object is extremely small or partially occluded—conditions common in our frisbee tracking task. This is why it was important to know the initial position of the frisbee to increase the odds of correctly keeping track.

Zero-shot Detection and Segmentation. Recent foundation models have introduced powerful capabilities for open-vocabulary and zero-shot vision tasks. Grounding DINO [7] extends the DINO object detection architecture by incorporating language grounding, allowing detection to be conditioned on arbitrary text prompts. Segment Anything Model (SAM) [8] enables high-quality object segmentation given minimal prompts (e.g., a point or bounding box) and generalizes well to unseen object categories. The combination of Grounding DINO and SAM has been applied in interactive annotation tools and vision-language segmentation pipelines, and serves as a foundation for our two-stage frisbee localization system: DINO generates a bounding box from a textual prompt, and SAM produces a fine-grained segmentation mask from that box.

Motion Blur in Object Tracking. Motion blur presents a significant challenge in visual object tracking, particularly in dynamic environments such as sports. When objects

move rapidly across frames, they often appear blurred, making detection and tracking difficult. This blurring can lead to missed detections or inaccurate bounding boxes, especially in scenarios with high-speed motion or low frame rates [9]. This is clearly related to frisbee as it results in a small high speed moving object thus making it hard to track using normal methods warranting novelty in this area. TrackNet [10] addresses a similar problem by training a network specifically designed to handle the fast motion of tennis balls, suggesting that specialized architectures or motion-aware models are essential in such scenarios. We attempt to mitigate this issue with no explicit method but implicitly solving it via the cropping the image making it easier to identify.

Frisbee-specific Challenges. Unlike balls, frisbees are flat, often rotate rapidly, and may be occluded by players or motion-blurred during throws. Despite these difficulties, their disc-like geometry can still be reliably identified by powerful visual encoders when the crop is centered effectively. Our hybrid pipeline—combining smart cropping, autoregressor with DINO-SAM, and a ResNet cropper - builds on these insights to create a robust tracking system without requiring frame-by-frame human supervision.

4. Dataset

4.1. Data Source

Our primary source of high-quality Ultimate Frisbee videos is Ultiworld, which routinely publishes match highlights and full-game broadcasts on YouTube. By following links from Ultiworld event pages, we compile a collection of passes from a short video segment, containing multiple different camera viewpoints (one per pass). We collect a total of 260 frames across these passes.

4.2. Video Scraping and Segmentation

Video URLs are programmatically retrieved from Ultiworld event pages using the YoutubeDL, an open-source command-line program to download YouTube videos. Then, we manually label the video in data segments for each individual frisbee pass, where each segment has a start and end timestamp and save each of these values in a csv.

We then write a custom Python script to read each segment’s start and end timestamps, and output frames at a fixed rate (5 frames per second) of the video between those timestamps to generate our input data frames and save each segment in a separate folder. For example, we could have the start timestamp of a segment be 0:20 and the end be 0:22, and this would create a folder called 0 : 20 – 0 : 22, which would contain 10 frames within it for the corresponding part of the video.

4.3. Data Annotation

To supervise our model, we manually annotate each video frame with the location of the frisbee. This is done by displaying each frame to a human annotator and capturing a single click at the center of the frisbee. The (x, y) coordinates of this click are recorded and stored in a CSV file alongside the frame filename. Our annotation interface supports skipping ambiguous frames (e.g., if the frisbee is occluded or not visible), and automatically prevents re-annotation of previously labeled frames.

The annotated CSV uses the format:

```
frame,x_center,y_center
0_15_0_17/frame_00001.jpg,432,212
...
```

Each line represents the annotated center of the frisbee for a single frame. This annotation format enables precise center-point localization while remaining lightweight compared to full bounding box or segmentation annotations.

4.4. Data Augmentation for Training

To enhance generalization and increase dataset size, we apply targeted data augmentation centered around our frisbee annotations. During dataset construction, we use the annotated center point to extract a square crop (default size: 250×250 pixels) around the frisbee. To introduce variability, we apply random translations to the crop center, displacing it within a controlled range from the original location.

This augmentation strategy simulates natural jitter and positional noise while preserving the relative location of the frisbee in the crop. The model is then trained to predict the offset of the true center from the crop center, which remains fixed at the center of the image.

Concretely, for each image:

- A random offset is sampled in both x and y directions (up to ± 105 pixels for a 250-pixel crop).
- The crop is padded if it extends beyond the image borders.
- The label is computed as the pixel distance from the crop center to the actual frisbee center.

This results in a wide variety of local appearances and positions, helping the model generalize to different viewpoints, occlusions, and camera motions without requiring extensive manual labeling. This increased our dataset by 10x.

5. Methods

Our goal is to track the frisbee through across a pass in static-camera shot, across multiple frames. Provided an initial point by an annotator, we perform cropping, followed

optionally by ResNet localization, after which we apply Grounding DINO and SAM to locate the mask of the frisbee. We continue this process auto-regressively to track the frisbee across frames, re-cropping the image for each frame based on the detected frisbee location in the previous frame. You can see an example of a prediction on a single frame in Figure 1.

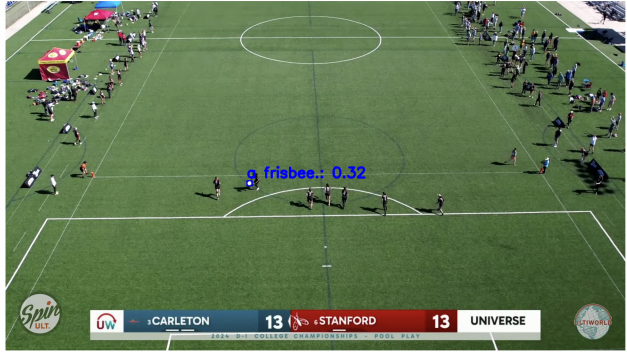


Figure 1. Example SAM segmentation of the frisbee in a single frame

5.1. Problem Setting

We assume each clip contains exactly one frisbee pass and that the camera remains fixed (no panning, tilting, or zooming). This simplifies the scene dynamics, so that all motion arises from player and disc movement rather than camera motion.

5.2. Initialization

An annotator provides the frisbee’s center point in the very first frame. This one manual click seeds our tracker’s state and acts as the initial position for the next frame.

5.3. Region-of-Interest Cropping

To reduce search space and improve detection on a tiny, fast-moving disc, we crop each incoming frame to a small window around the last known position. We will evaluate based on different cropping dimensions.

Within these smaller windows, the frisbee occupies a larger fraction of pixels, creating our hypothesis that it is easier to detect even under motion blur and among distractor objects.

5.4. Detection Backbone: Grounding DINO and SAM

Once we have the cropped image, we apply Grounding DINO and Meta’s Segment Anything Model (SAM) in order to identify the frisbee.

Grounding DINO is an object detection model that combines grounding (i.e., linking text descriptions to regions in

an image) with DINO, a strong detector based on transformers. It allows users to detect objects based on open-ended text prompts, enabling flexible, zero-shot object detection. Meta’s Segment Anything (SAM2) is a general-purpose image segmentation model that can segment any object in an image given a prompt like a point, bounding box, or mask. It is trained on a massive dataset and excels at high-quality segmentation across a wide variety of objects and scenes without requiring fine-tuning.

The combination of Grounding DINO and SAM work perfectly for our use-case, because Grounding DINO alone can only provide a bounding box, while SAM requires a bounding-box or point to condition its segmenting. Thus, we feed the label ”a frisbee.” to Grounding DINO, which outputs a bounding box. We then feed it into SAM and it outputs a mask of the frisbee.

5.5. CoordResNet18 Localizer

ResNet (Residual Network) is a widely used convolutional neural network architecture that utilizes skip connections to help mitigate vanishing gradient issues in deep networks. ResNet-18 is a lightweight variant with 18 layers and offers a good balance between performance and efficiency, making it suitable for real-time or low-latency inference tasks.

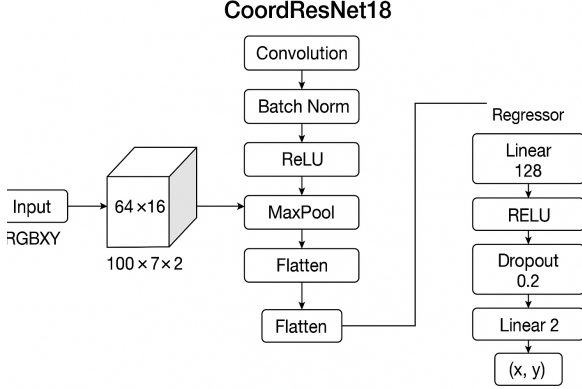


Figure 2. CoordResNet18 Architecture

We develop a modified architecture of ResNet-18, called *CoordResNet18*, to predict the center of the frisbee from cropped image patches. We input a similar 500 by 500 pixel crop that we passed into the previous DINO/SAM method. The key modification is that, instead of taking only the RGB image as input, our model takes a 5-channel input: the RGB image concatenated with two coordinate channels (X and Y), each normalized to $[-1, 1]$. These coordinate maps explicitly provide spatial information to the model, improving its ability to reason about relative position within the crop.

The model consists of three main components:

1. The original ResNet-18 backbone, modified to accept 5-channel inputs by initializing the first convolutional layer with pretrained RGB weights and zero-initialized weights for the coordinate channels.
2. A global average pooling layer that compresses the spatial feature map to a 512-dimensional vector.
3. A regression head consisting of two linear layers with a ReLU activation and dropout, producing a 2D output corresponding to the predicted center coordinates (normalized to $[0, 1]$).

During training, we minimize the Smooth L1 Loss (Huber loss) between the predicted and ground-truth center coordinates. Input crops are augmented via random translations (see previous section), and the model is trained to predict the offset-adjusted center of the frisbee. We convert the ground-truth pixel coordinates into normalized coordinates for training, and revert predictions to pixel space during evaluation.

During training, we compute the mean μ and standard deviation σ of the Euclidean prediction error on the validation set. Then, at test time, we define a new square crop of size:

$$\text{Crop Size} = (\mu + 3\sigma)$$

centered around the predicted frisbee position.

By Chebyshev’s inequality, for any random variable X with mean μ and standard deviation σ , the probability that X deviates from μ by more than $k\sigma$ is bounded by:

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}$$

Setting $k = 3$, we obtain:

$$P(|X - \mu| < 3\sigma) \geq 1 - \frac{1}{9} = \frac{8}{9} \approx 88.9\%$$

Thus, our crop size is chosen to contain at least 88.9% of the prediction errors, increasing the likelihood that the frisbee remains within the crop region. This refined crop is then passed to the Grounding DINO and SAM pipeline for more accurate identification.

5.6. Baseline: Grounding DINO and SAM

Our primary baseline is a naïve frame-by-frame pipeline: apply SAM + Grounded-DINO on the full frame (no cropping, no motion prior), select the top frisbee proposal, and record its center. We will compare this method to our approaches using different crop windows as well as the ResNet localizer

5.7. Adding Baseline fallback to both methods

We also predicted that sometimes the frisbee will leave the cropped frame so we will experiment on adding a heuristic where if the frisbee isn't detected on the crop, we run our dino-sam pipeline on the entire image.

5.8. Occlusions

Sometimes we see frames where there are occlusions (i.e. the frisbee is not visible from the camera's viewpoint) for example in Figure 3 . To account for this case, if a frisbee is not detected in the frame, we assume that the frisbee was in its last known location. The reason we can make this assumption is because in ultimate frisbee, the player throwing the frisbee is not allowed to move. Therefore, we know that in subsequent frames, we can look in the same region, until the frisbee becomes visible.



Figure 3. Example of an occlusion, where the frisbee is not visible in the image. The true location of the frisbee is in the hand of dark shirt team player, and is occluded by the hat of a spectator.

6. Experiments and Analysis

6.1. ResNet FineTuning

We train the model on the augmented dataset of 2600 images and validated on 400 images while running for 20 epochs using the Adam optimizer with a learning rate of $1e-4$. We monitor validation loss and save the model with the best performance.

GT=(114.0,92.4) Pred=(124.0,96.5)



Figure 4. CoordResNet18 with 250 crop frisbee prediction

As can be seen in 5, the prediction does well, with typically low Euclidean error (right skewed distribution) and if we recrop the image around the ResNet, there is a higher chance our DINO-SAM model will be able to correctly identify the bounded box for the frisbee. As seen in Table 1, most of the error of the ResNet comes within a small range of the mean with some outliers.

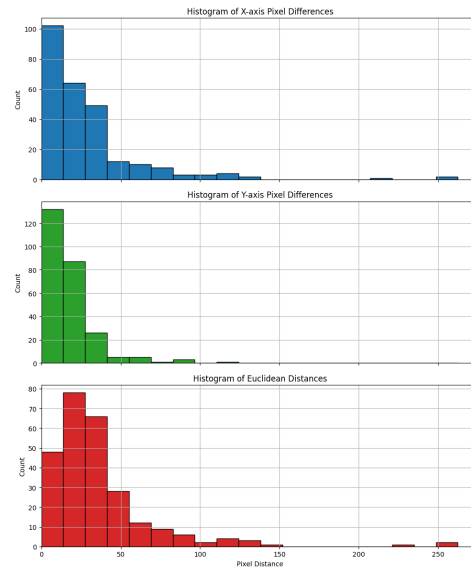


Figure 5. CoordResNet Euclidean Error for crop size of 500

Mathematically, according to Chebyshev's inequality, a minimum of 88% of the values fall within $3 * std$ of the mean. Using this we can create our new crop region from the predicted point, that would still contain the frisbee with high probability . Below is the table that represents the mean, std, and new crop size of each CoordResNet18 trained on the different crop sizes.

Crop Size	Mean Error	STD	New Crop Size
250	19.39	20.11	80
500	36.50	34.73	150
750	72.30	52.85	230

Table 1. Error statistics and adjusted crop sizes for different initial crop sizes (Units in pixels)

6.2. Crop Only Experiments

We ran a total of 7 experiments:

1. **Baseline:** DINO-SAM pipeline on uncropped image

The next 2 experiments were run on crop sizes of 250, 500, and 750 respectively:

2. **Cropped:** Run DINO-SAM pipeline on cropped image centered around last known prediction
3. **CoordResNet18:** Run our custom pretrained ResNet18 model to identify frisbee center and then recrop to mean error + $3 \cdot \text{std}$ centered on prediction. Then feed this into our DINO-SAM pipeline.

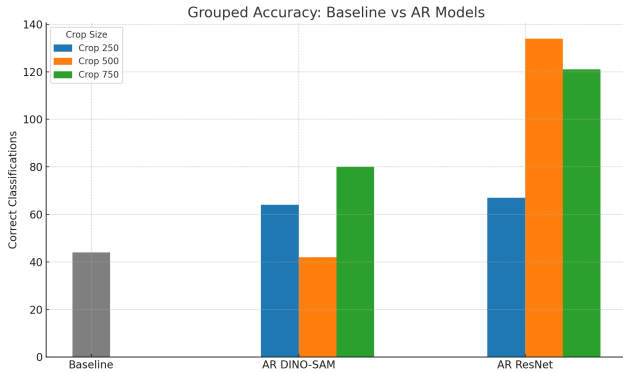


Figure 6. Crop Only Experiment Results

As can be seen, introducing the cropped image does result in either the same results or increased improvement. At a crop size of 250, DINO-SAM clearly can identify the frisbee more often than the baseline as long as the frisbee is in the frame. This is due to the fact we are processing a smaller image. However, we see a considerable decrease in performance when going to a crop size of 500 as we lose the benefit of the smaller crop size for easier identification. Moreover, even though the frisbee is identified better in some instances compared to the baseline, there is a chance the frisbee goes out of the cropped frame resulting in a compounding error as it can never find it again. This is why the 750 crop does better as there is a smaller chance the frisbee leaves the frame without identification even if we can't identify it in between a set of frames. In general, we can see there is a considerable balance between crop sizes

as a smaller crop results in better identification but higher compounding errors if the frisbee has left the frame. This will also be further explored in our Fall-Back experiments.

We can see these results further augmented by using our ResNet model. In general, the ResNet to DINO-SAM pipeline does outperform its corresponding counterparts as the ResNet takes the bigger crop size, identifies a region and re-crops it. This takes the benefit that we saw earlier where DINO-SAM performs well on small cropped sizes by cropping it to said smaller images, and mitigates the error we saw if it misses the frisbee in one instance causing it to be out of the cropped frame in later instances.

6.3. Fall-Back Experiments

We ran the same experiments as above except with the fallback where if the pipeline failed to detect the frisbee, then we would result to our baseline model which runs the DINO-SAM pipeline on the entire image.

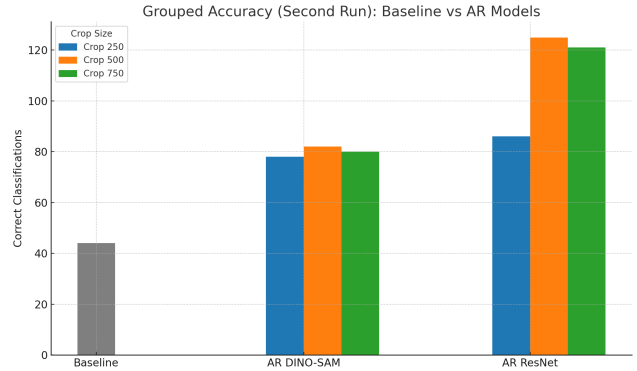


Figure 7. Fall-Back Experiments

As can be seen, we see a massive increase on both crop size 250 and 500 for the DINO-SAM pipeline. This further reinforces our analysis from before that the smaller crop sizes are able to identify more but there are parts where the frisbee ends up outside the cropped frame thus having a larger view of the image is better. We see minimal improvement on 750 as it is already close to the original image size so it identifies at least the same amount as the baseline.

Even for ResNet we see slight improvements as there are times where it does in fact go outside the initial cropped box due to missing a previous frame. That's where the use of running DINO-SAM on the entire image is more useful to identify the frisbee if we lose track of it. The main reason why we might not see considerable increase is due to miss-classification of the frisbee when we run the baseline model if ResNet is unable to detect the frisbee. This is because if we missclassify, it compounds into our next frame as the new cropped box is centered around our false classification. This was clearly seen during occluded moments where the frisbee was not visible but the DINO-SAM pipeline on

the entire image identified a frisbee on the ground instead, throwing the autoregressive predictions off entirely.

7. Conclusion

Initially, DINO-SAM does not do a great job on identifying a frisbee on a large image. However, via our experiments, we can clearly see it does better when given a smaller cropped image. This is where we were met with the limitation of the frisbee going out of frame. To combat the issue, two approaches were taken. The first is to simply sacrifice the benefit and increase crop size. The next was to fine tune our own ResNet model to identify the approximate region of the frisbee from a larger crop size and then feed in a smaller cropped version into the DINO-SAM pipeline to get the accurate identification. To further address these limitations of it losing track of the frisbee, we also added a fall back that ran DINO-SAM on the entire image as a worst case scenario if the frisbee couldn't be identified.

8. Future Works

The next steps to try out would be to create a more dynamic cropping mechanism. As was seen, 750 crop size didn't benefit from the fall-back mechanism so maybe utilizing that as the fall back crop size instead of the entire image would be more useful. Moreover, instead of just doing one ResNet region approximation, we could start off at a larger crop size say 750, and then crop down to 250 and use our ResNet for crop size of 250 to further identify the region. A final improvement would also be to add a momentum vector to the crop box where we take the calculate speed / direction of the frisbee from the last 2 frames to predict where it would end up and use that to set up our initial crop box instead of centering it around the previous frisbee. Simply put, we would experiment with different ways of setting up our crop to increase the odds the frisbee didn't leave the box.

References

- [1] Zhang, Tao, et al. “Efficient Golf Ball Detection and Tracking Based on Convolutional Neural Networks and Kalman Filter.” *arXiv*, 15 Dec. 2020. arXiv:2012.09393. <https://arxiv.org/abs/2012.09393>
- [2] Cioppa, Anthony, et al. “A Context-Aware Loss Function for Action Spotting in Soccer Videos.” *arXiv*, 3 Dec. 2019. arXiv:1912.01326. <https://arxiv.org/abs/1912.01326>
- [3] Arbués-Sangüesa, Adrià, Coloma Ballester, and Gloria Haro. “Single-Camera Basketball Tracker through Pose and Semantic Feature Fusion.” *arXiv*, 5 June 2019. arXiv:1906.02042. <https://arxiv.org/abs/1906.02042>
- [4] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition.” *arXiv*, 10 Dec. 2015. arXiv:1512.03385. <https://arxiv.org/abs/1512.03385>
- [5] Liu, Rosanne, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. “An Intriguing Failing of Convolutional Neural Networks and the CoordConv Solution.” *arXiv*, 9 July 2018. arXiv:1807.03247. <https://arxiv.org/abs/1807.03247>
- [6] Jin, Ruibing, Min Wu, Keyu Wu, Kaizhou Gao, Zhenghua Chen, and Xiaoli Li. “Position Encoding Based Convolutional Neural Networks for Machine Remaining Useful Life Prediction.” *2022 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2022, pp. 1–8. <https://ieeexplore.ieee.org/document/9849459>
- [7] Liu, Shilong, et al. “Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection.” *arXiv*, 9 Mar. 2023. arXiv:2303.05499. <https://arxiv.org/abs/2303.05499>
- [8] Kirillov, Alexander, et al. “Segment Anything.” *arXiv*, 5 Apr. 2023. arXiv:2304.02643. <https://arxiv.org/abs/2304.02643>
- [9] Guo, Qing, et al. “Effects of Blur and Deblurring to Visual Object Tracking.” *arXiv*, 21 Aug. 2019. arXiv:1908.07904. <https://arxiv.org/abs/1908.07904>
- [10] Huang, Yu-Chuan, et al. “TrackNet: A Deep Learning Network for Tracking High-Speed and Tiny Objects in Sports Applications.” *arXiv*, 10 July 2019. arXiv:1907.03698. <https://arxiv.org/abs/1907.03698>