MLVOT Report

Introduction

In the latest iteration of our algorithm, a neural network has been integrated to enhance the detection and tracking capabilities. This neural network efficiently computes embeddings from the image patches obtained from both the tracks and the detections within the specified file. The utilization of these embeddings allows us to establish a similarity score, employing the cosine similarity metric, providing valuable insights into the resemblance between the current bounding box and previous tracking data.

Enhancing Detection Relevance

Expanding on our approach, we further augment our evaluation by incorporating Intersection over Union (IOU) calculations, providing a definitive score that helps prioritize and identify the most relevant detections. This combination of IOU and image similarity metrics equips us with a comprehensive understanding of the context, refining our ability to distinguish relevant objects in a given frame.

Step-by-Step Process

To break down the methodology:

1. **Getting Detections:** Acquire information from the current frame, identifying objects from the dataset
2. **Fetching Tracks:** Retrieve tracks that contain information from the previous frame
3. **IOU Computation:** Calculate the IOU for each detection in relation to each track, quantifying the spatial overlap and assessing the relevance of detections.
4. **Image Similarity Computation:** Leverage the neural network to compute image similarity scores for each detection corresponding to each track, contributing to a holistic evaluation.
5. **Score Aggregation:** Sum the IOU and image similarity scores, providing a unified metric for each detection-track pair.
6. **Hungarian Algorithm:** Employ the Hungarian algorithm to efficiently determine the maximum score for each track, facilitating optimal track assignments.
7. **Track Management:** Identify and manage correct track assignments, ensuring the removal of outdated tracks and the incorporation of new ones based on the computed scores.

Performance

Without the neural network model, it is possible to gain alot of performance by avoiding for loops when necessary.

The biggest leap is when we have to compute the gain matrix for the hungarian algorithm.

In a for loop, for each track we have to compute the IOU for each detection.

This takes a long time especially if we compute on the entire detections array.

It can be improved like so:

```python
def __get_iou(self, box1):
    """
    Returns a function that calcultes the iou for a given bounding box
    """

    def filter_iou(detection):
        iou = self.__calculate_iou(box1, detection[2:6])
        return iou if iou > self.sigma_iou else 0

    return filter_iou

def track_detection(self, frame_number: int) -> dict:
    """
    Update tracks using iou threshold
    """
    detections = self.detections[self.detections[:, 0] == frame_number]
    matched_detections = set()
    for _, track in self.tracks.items():
        ious = np.apply_along_axis(self.__get_iou(track["bbox"]), 1, detections)
        best_index = ious.argmax()
        if ious.max() == 0 or best_index in matched_detections:
            track["last_updated"] = -1
            continue
        track["bbox"] = detections[best_index][2:6]
        track["last_updated"] = frame_number
        matched_detections.add(best_index)
```

Here we compute the IOU for each detections done in the frame by mapping directly onto the array.

This allows us to reach upwards of 40 to 50 fps whereas we can hardly reach 30 fps with a normal for loop.

I couldn't measure the accuracy using TrackEval since the code was not compatible with my setup (code deprecated and I couldn't revert back to an older version).

However, we do notice that it is hard to maintain the same id on the video feed since the IOU values on the detection can below the threshold which causes the program to create new tracks from scratch.

Looking at the detections file, we notice that the confidence score on certain observations can reach below 10% which could explain those difficulties I had making sure the ID was unique and consistent: it couldn't since it had no tracks to follow.

Decreasing the threshold to 0.3 seems to have helped but going any further would hurt accuracy, especially on overlapping objects (unique IDs would often teleport).

The neural network has helped in that regard resulting in a FPS count in the single digits.

A MOT neural network optimized for video feed would have helped but would render the rest of the code pointless.

Potential Enhancements

Continuing our pursuit of optimization, we propose the introduction of a coefficient, gamma, which can be adjusted to fine-tune the influence of IOU in the overall score. This adaptive approach allows for a dynamic assessment, emphasizing the significance of spatial overlap in the scoring mechanism based on the specific requirements of the application. We could also multiply

the IOU by the confidence field from the det dataset (scaled from 0 to 1).

Additionally, integrating YOLOv2 (You Only Look Once) into our framework presents a promising avenue for immediate and substantial improvements. YOLO, known for its real-time object detection capabilities, could enhance the accuracy and efficiency of our system, providing more accurate and timely results.

Conclusion

In conclusion, our iterative approach to refining the MLVOT algorithm demonstrates a commitment to enhancing object detection and tracking in video analysis. By combining state-of-the-art techniques such as neural network-based embeddings, IOU computations, and potential integrations with advanced models like YOLO, we aim to continually elevate the performance and reliability of our system in real-world scenarios.

We couldn't benchmark our accuracy but the video feed has shown accurate tracking at the cost of unique ID tracking.