Arya Goel, Jordan White, David Zoro

06/062023

EE 443

Title: Multi-Object Tracking Challenge Report

Abstract

Multi abstract tracking is a technique used to track multiple objects in a video or image sequence. It involves tracking objects by extracting abstract features from them, such as shape, texture, or color. These abstract features provide a higher-level representation of the objects, enabling more robust and accurate tracking. By utilizing the embedded features, the tracking algorithm can more accurately follow the object's movement throughout the video or image sequence. We also changed the scaling method in the processing file by using MinMaxScalar. By altering the scaling method, the tracking algorithm can adapt to varying object sizes more effectively, enhancing the overall tracking performance. In the context of tracking, using a feature matrix in the Intersection over Union (IoU) tracking file involves creating a matrix that represents the relationship between the features of the tracked objects and their respective IoU values.

Introduction

The problem to be solved in the Multi-Object Tracking Challenge involves accurately tracking multiple objects across a series of frames in a video sequence. Given a video input, the goal is to detect and track objects of interest as they move and interact with each other over time. This problem is particularly challenging due to several factors, such as occlusions, scale variations, motion blur, and complex object interactions.

The first aspect of the problem is object detection, where the algorithm needs to identify and localize objects in each frame of the video. This can be done using various techniques, such as deep learning-based object detection algorithms like Faster R-CNN or YOLO. Once the objects are detected, the tracking algorithm needs to associate the correct identities to the objects across different frames, ensuring consistent tracking of each object over time.

One of the main challenges in multi-object tracking is dealing with occlusions, where objects may be partially or completely hidden by other objects in the scene. The tracking algorithm should be able to handle occlusions by predicting the object's position even when it is not visible, and correctly re-establishing the track once the object re-enters the camera's field of view.

Another important aspect is re-identification, which involves correctly associating a tracked object in one part of the video with the same object that reappears later in a different part of the video. This requires robust and accurate matching of object features or appearances to ensure consistent tracking across different camera views or when objects temporarily leave and re-enter the scene.

Finally, to ensure the tracking results are reproducible, the challenge requires algorithms to be able to process videos in real-time or near real-time. This means the tracking algorithms should be efficient and capable of handling large amounts of data while maintaining accurate and reliable tracking performance.

In summary, the Multi-Object Tracking Challenge addresses the problem of detecting and tracking multiple objects across video sequences. It encompasses tasks such as object detection, object association across frames, re-identification of objects, and ensuring the tracking results are reproducible.

Data Description

For the validation datasets (71, 72, 73, and 74), the videos had the same scene and the same number of people throughout. The only variation among these videos was the camera angle used to capture the scene. Each of these validation videos lasted for a duration of 10 minutes.

In these videos, the number of people remained constant, and whenever a person exited the scene, they were given a new identification box when they re-entered. This means that each person was assigned a unique identifier whenever they appeared in the video, even if they had been previously tracked. This approach allowed for consistent referencing and tracking of individuals across different frames.

The availability of ground truth for these validation videos was clear, as there were only five people present in the entire video sequence. This knowledge helped in accurately evaluating and comparing the performance of different tracking algorithms. With a known ground truth of the number of people, their identities, and their movements, it became easier to assess the accuracy of the tracking algorithms and measure their ability to correctly detect, track, and re-identify individuals in varying camera angles.

Regarding the testing data, there was a single video that was processed using the tracking code. Similar to the validation/training videos, this testing video also lasted for 10 minutes. The scene in the testing video closely resembled the validation/training videos, as it featured the same individuals and their movements. However, the key difference was the camera angle, which was positioned towards the southbound wall of the room.

Due to the fixed number of people (five) in the testing video, their movements caused them to enter and exit the frame throughout the duration of the video. As with the validation videos, whenever a person reappeared in the scene, they were given a new identification box for reference. This ensured consistent tracking and identification of individuals, despite their temporary absence from the camera's view.

Methodology

- Object Detection: The run method of the tracker class is responsible for tracking the data. Additionally, the run method iterates over each frame. The run method filters over each frame and stores them in cur_frame detection and cur_frame features.
- Re-Identification: The re-identification is done using a cost matrix and the Hungarian algorithm. Furthermore, we altered the ratio of the cost matrix to find an optimal solution via manual testing. The cost matrix is composed of the IoU cost matrix and the feature cost matrix. We used an optimal search strategy to find the right value. The algorithm returns the indices of the matched pairs. The code iterates over the matched tracklet detection pairs and updates the existing tracklets. The detections that are not matched are considered as unmatched directions. The tracklets that are not present in a frame are removed. All the tracklets throughout every frame are stored.
- Multi-Object Tracking: We implemented multi-object tracking by altering the cost matrix. Additionally, we implemented multi-object tracking by using embedded features and altering the scaling method. The changes we made considerably improved our accuracy. Furthermore, we altered the ratio of the cost matrix to find an optimal solution. Beyond that, we altered hyper-parameters such as the threshold value.

Implementation

In the main file, editing is performed using confidence scores. A confidence score is a numerical value that represents the algorithm's confidence or certainty in the correctness of a particular tracking result. By incorporating confidence scores, the main file can evaluate and filter out

low-scoring tracking results. This filtering process helps to improve the overall accuracy and reliability of the tracking system by discarding unreliable or uncertain object tracks based on their confidence scores.

In the context of tracking, using a feature matrix in the Intersection over Union (IoU) tracking file involves creating a matrix that represents the relationship between the features of the tracked objects and their respective IoU values. IoU is a commonly used metric to measure the overlap between two bounding boxes or regions. By incorporating a feature matrix, the tracking algorithm can leverage the correlation between object features and IoU values to make more informed decisions during tracking, such as estimating object association or predicting future object locations.

```
else:
    iou_cost_matrix = np.zeros((len(self.cur_tracklets),len(cur_frame_detection)))

    for i in range(len(self.cur_tracklets)):
        for j in range(len(cur_frame_detection)):
            iou_cost_matrix[i][j] = 1 - calculate_iou(self.cur_tracklets[i].cur_box, cur_frame_detection[j][3:7])

    if len(cur_frame_features) > 0:
        det_features = np.asarray([cur_frame_features[j] for j in range(len(cur_frame_detection))], dtype=np.float64)
        track_features = np.asarray([track.get_avg_features() for track in self.cur_tracklets], dtype=np.float64)

        # Ensure both feature matrices are 2-dimensional
        if len(det_features.shape) == 1:
            det_features = det_features.reshape(1, -1)
        if len(track_features.shape) == 1:
            track_features = track_features.reshape(1, -1)

        feature_cost_matrix = np.maximum(0.0, cdist(track_features, det_features, 'cosine'))

        # Fuse cost matrices
        cost_matrix = 0.325 * iou_cost_matrix + 0.675 * feature_cost_matrix

        row_inds,col_inds = linear_sum_assignment(cost_matrix)

        matches = min(len(row_inds),len(col_inds))
```

Tracking using embedded features refers to the use of embedded features within an object for tracking purposes. Embedded features are specific characteristics or properties that are inherent to an object, such as edges, corners, or keypoints. These properties allow the tracking algorithm to more successfully track the motion of the objects throughout the sequence, which enhances tracking performance.

To alter the scaling method for the processing file means modifying the technique used to adjust the size of the objects being tracked. We used the Robust Scaling preprocessing technique used to transform numerical data to a common scale while minimizing the influence of outliers. Scaling plays a crucial role in tracking, as objects may appear at different scales due to perspective changes or distance from the camera. By altering the scaling method, the tracking algorithm can adapt to varying object sizes more effectively, enhancing the overall tracking performance.

```python
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler

class postprocess:
    def __init__(self, number_of_people, cluster_method, scale_method):
        self.n = number_of_people
        if cluster_method == 'kmeans':
            self.cluster_method = KMeans(n_clusters=self.n, random_state=0)
        else:
            raise NotImplementedError

        if scale_method == 'standard':
            self.scaler = StandardScaler()
        elif scale_method == 'minmax':
            self.scaler = MinMaxScaler(feature_range=(0, 1))
        elif scale_method == 'robust':
            self.scaler = RobustScaler()
        else:
            raise NotImplementedError

    def run(self, features):
        print('Start Clustering')

        # Scale or normalize the features
        scaled_features = self.scaler.fit_transform(features)

        # Fit the clustering algorithm
        self.cluster_method.fit(scaled_features)

        print('Finish Clustering')

        return self.cluster_method.labels_
```

Evaluation

In our study, we used an extensive range of evaluation metrics to comprehensively assess the effectiveness and accuracy of our multiple object tracking system.

One of the main metrics employed was the IDF1 score. This score is the harmonic mean of Identification Precision (IDP) and Identification Recall (IDR), offering a balanced measure of our system's performance in terms of both identification and temporal consistency of tracked objects. IDP quantifies how many of the computed detections were correctly identified, whereas IDR measures the proportion of actual detections that were correctly identified by our system. These metrics together provide a clear picture of our system's performance in terms of precision and recall.

Moreover, the evaluation also counted the instances of ID True Positives, ID False Positives, and ID False Negatives. These raw numbers denote the system's ability to correctly match objects, incorrectly match objects, and miss detections, respectively, giving a granular insight into its performance across all frames.

Metrics such as Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP) were instrumental in our evaluation process. MOTA accounts for the total number of errors, including false positives, false negatives, and identity switches relative to the number of ground truth objects. This provides a measure of the overall accuracy of our tracking system. On the other hand, MOTP measures the alignment between the predicted and ground truth bounding boxes, thereby assessing the spatial precision of our object localization.

Finally, our evaluation process took into consideration Recall, Precision, ID Switches, and lifespan related metrics like Mostly Tracked, Partially Tracked, and Mostly Lost. These metrics represent the system's ability to correctly identify tracked objects, predict positive cases accurately, handle identity switches during tracking, and effectively track objects across different stages of their lifespan. Collectively, these metrics enabled a thorough and multi-faceted evaluation of our tracking system, ensuring its performance is evaluated holistically.

Table 1: Evaluation Metrics

	IDF1	IDP	IDR	idtp	idfp	idfn	MOTA	MOTP	Rcll	Prcn	IDs	MT	PT	ML
c071	95.81	98.2	93.54	58423	1072	4038	91.87	10.48	93.58	98.25	27	4	1	0
c072	96.75	97.81	95.72	48286	1081	2161	95.28	11.73	96.61	98.72	38	5	0	0
c073	95.7	96.05	95.36	51273	2111	2493	91.74	12.03	95.62	96.3	109	5	0	0
c074	85.22	84.17	86.29	27059	5088	4300	89.96	10.71	96.27	93.91	17	4	0	0
average	93.37	94.05	92.73	46260	2338	3248	92.21	11.238	95.52	96.795	47.75	4.5	0.25	0
max	96.75	98.2	95.72	58423	5088	4300	95.28	12.03	96.61	98.72	109	5	1	0
min	85.22	84.17	86.29	27059	1072	2161	89.96	10.48	93.58	93.91	17	4	0	0

The table presents Multiple Object Tracking (MOT) metrics for four validation sets (c071, c072, c073, c074). The IDF1, IDP, and IDR scores, which measure the system's precision and recall, average around 93%, indicating good performance in identifying and tracking objects. The MOTA score, considering false positives, false negatives, and identity switches, averages at 92.2125%, suggesting effective tracking despite some errors. However, the MOTP score, measuring tracking precision, averages at 11.2375%, indicating room for improvement. The Recall (Rcll) and Precision (Prcn) scores are high, averaging at 95.52% and 96.795% respectively, showing the system's effectiveness in identifying and accurately tracking objects. The number of identity switches (IDs) varies significantly, with c073 having the most at 109 and c071 the least at 27. The MT, PT, and ML scores show effective tracking with few objects being

lost. Notably, the c074 validation set was a lower outlier in the results, suggesting it may have been a more challenging scenario or an anomaly. The average IDF1 score (without taking into account the c074 results) was 96.1%.

Conclusion

In conclusion, we used embedded feature tracking and scaling methods to find an optimal solution to the task at hand. Through additional iterations and research, we found an appropriate scaler. Min/Max scaler is good for normalizing data. Min/Max scaler is especially good for normalizing data with set bounds such as RGB values. A robust scaler is good for standardizing data on a normal distribution. Via observation and analysis, we used a robust scaler to complete the object-tracking task. Furthermore, we altered the ratio between the IoU cost-matrix and the feature cost matrix to enhance performance. Future directions of the work include finding additional ways to improve performance beyond feature extraction. Moreover, another direction includes finding different methods that can improve the IDF1 score.

References

[1]“StandardScaler, MinMaxScaler and RobustScaler techniques - ML,” *GeeksforGeeks*, Jul. 15, 2020. https://www.geeksforgeeks.org/standardscaler-minmaxscaler-and-robustscaler-techniques-ml/

[2]“MinMaxScaler vs StandardScaler - Python Examples,” *Data Analytics*, Jul. 27, 2020. https://vitalflux.com/minmaxscaler-standardscaler-python-examples/

[3]E. Hofesmann, “IoU a better detection evaluation metric,” *Medium*, Mar. 01, 2021. https://towardsdatascience.com/iou-a-better-detection-evaluation-metric-45a511185be1

[4]D. Yuan, X. Shu, N. Fan, X. Chang, Q. Liu, and Z. He, “Accurate bounding-box regression with distance-IoU loss for visual tracking,” *Journal of Visual Communication and Image Representation*, vol. 83, p. 103428, Feb. 2022, doi: https://doi.org/10.1016/j.jvcir.2021.103428.

[5]Z. Chen, G. Qiu, H. Zhang, B. Sheng, and P. Li, “FIOU Tracker: An Improved Algorithm of IOU Tracker in Video with a Lot of Background Inferences,” pp. 145–156, Oct. 2020, doi: https://doi.org/10.1007/978-3-030-61864-3_13.