NATIONAL UNIVERSITY OF HO CHI MINH CITY
UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY
SPECIALIZED FIELD IN COMPUTER VISION
& COGNITIVE CYBERNETICS DEPARTMENT
SUBJECT : COMPUTER VISION

PROJECT SEMINAR

REPORT

This document describes the content of the project for the subject Computer Vision

Instructors

Phạm Minh Hoàng
Nguyễn Trọng Việt
Võ Hoài Việt

Students

Phan Trí Nguyên
Trần Hoàng Kim

Ho Chi Minh City, April 28, 2023

INDEX

1. INFORMATION

1.1 STUDENTS INFORMATION

Student ID	Name	Student email	Class	Group
20127578	Phan Trí Nguyên	20127578@student.hcmus.edu.vn	20TGMT01	03
19127039	Trần Hoàng Kim	19127039@student.hcmus.edu.vn	20TGMT01	03

1.2 PROJECT INFORMATION

No.	Project Name	Programming Environment
10	Object Tracking	Softwares: Visual Studio Code, Google Collab, Github, Zoom. Programming Languages: Python Topic: 2nd project seminar report: Problem statement, Related Works, Methodology, Experiments and Conclusion.

2. OVERVIEW

2.1 TOPIC MOTIVATION

2.1.1 Application motivation

In this modern age, Object Tracking has a wide range of applicational significances in various fields, including the applicational significance of object tracking is that it enables real-time monitoring, analysis, and control of various processes and systems.

Object Tracking can be used in surveillance to detect and track objects or people, in traffic monitoring to manage traffic flow and prevent accidents, in sports analysis to analyze player movements and performance, in robotics to enable robots to navigate and interact with their environment, and in augmented reality to create immersive experiences. Object tracking can also be used for behavior analysis, such as studying animal or human behavior patterns.

2.1.2 Scientific motivation

Lying on image processing, the topic of Object tracking is a process of locating and following a specific object or region of interest (ROI) in a video stream or a sequence of images.

Object Tracking is the automatic identification and tracking of objects in videos or image sequences, with applications in computer vision, machine learning, robotics, and augmented reality. It can be used for surveillance, traffic monitoring, sports analysis, and studying behavior patterns.

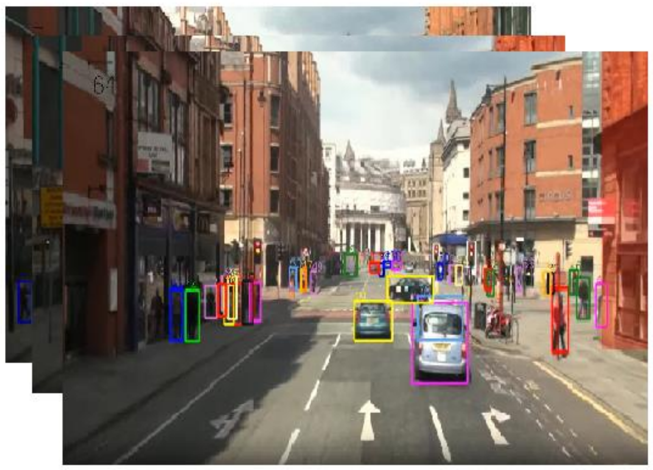
2.2 INTRODUCTION

2.2.1 Problem statement

2.2.1.1 Basic understanding of Input and Output of the system

Input of the system **[1,2]**: a sequence of frames from a video contains one or more objects.

Output of the system **[1,2]**: Trackless which includes the object’s frame, bounding boxes, identification over time.

Input of the system	Output of the system
	

2.2.1.2 Multiple Object Tracking (MOT)

Multiple Object Tracking (MOT) [2] is the process of simultaneously tracking multiple objects in a video sequence or a set of images. MOT is a challenging problem in computer vision and has many practical applications, including traffic monitoring, crowd analysis, and robotics.

The goal of MOT is to assign a unique identity to each object and estimate their locations and trajectories over time. MOT algorithms typically use a combination of object detection, data association, and motion estimation techniques to track multiple objects.

2.2.2 Challenges

2.2.2.1 Problem Limitation

Ensuring that the ID of each object remains unchanged across frames.

When the object is occluded or disappears after a few frames, the system still needs to ensure that the correct ID is assigned when the object reappears.

Issues related to processing speed to ensure real-time and high applicability.

2.2.2.2 Realtime Object Tracking

The method called "real-time" needs to ensure that the output speed is faster or at least equal to the speed of input.

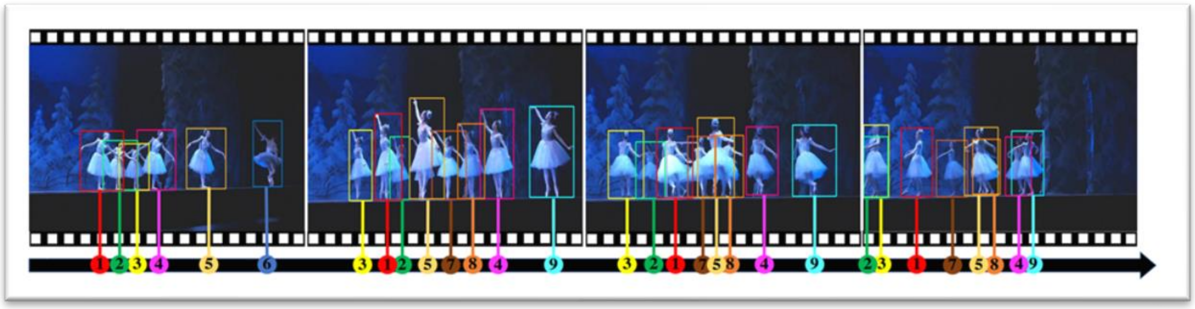
In reality, if processing each frame only causes a delay of 1 second compared to its average speed, this processing can still be acceptable as real-time. However, even when accepting a delay, ensuring real-time is still always a challenging issue.

Currently, the latest research is still seeking methods fast enough to aim for real-time processing.

2.2.2.3 New Challenges

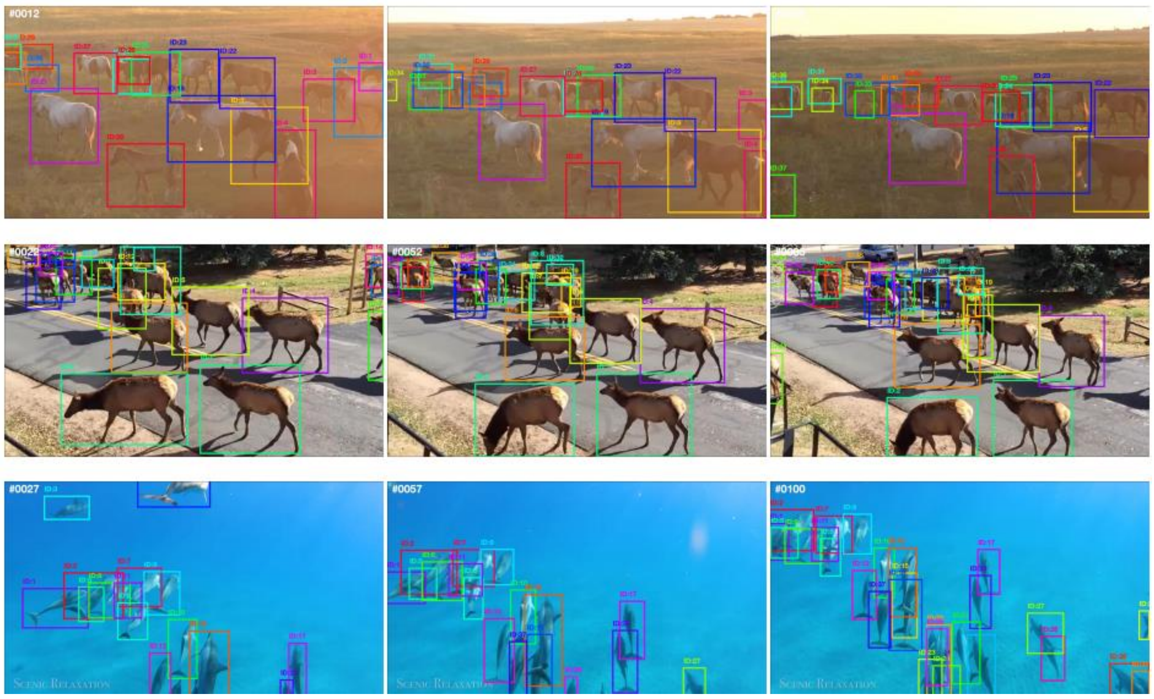
Large-scale data for tracking, where objects have a similar appearance and diverse motion.

The numbers below show their identifications which experience frequent relative position switches and occlusions.



DanceTrack: Multi-Object Tracking in Uniform Appearance and Diverse Motion [5]

Kind of animals have a **high similarity appearance** which easily causes **switching identities**.



AnimalTrack: A Benchmark for Multi-Animal Tracking in the Wild **[6]**

2.3 CONTRIBUTIONS

2.3.1 Online Tracking

Using only the current frame and the immediately preceding frame for tracking.
May reduce the accuracy of the algorithm, but it reflects the practical online nature of the problem.

2.3.2 Offline Tracking

Typically use the entire video frames, thus achieving much higher accuracy compared to Online Tracking.

2.3.3 Detection based Tracking (2-stage)

Focusing on the close relationship between Object Detection and Object Tracking
Using the results of detection to track the object across frames.

2.3.4 Summary of Multi-Object Tracking

Tracking by Detection (2-stage):
Step 1: Detector for object detection: YOLO
Step 2: Object association among frames:
Appearance matching
Motion analysis
Detection Free Tracking (1-stage): Detection and tracking steps are simultaneously produced in a single network.

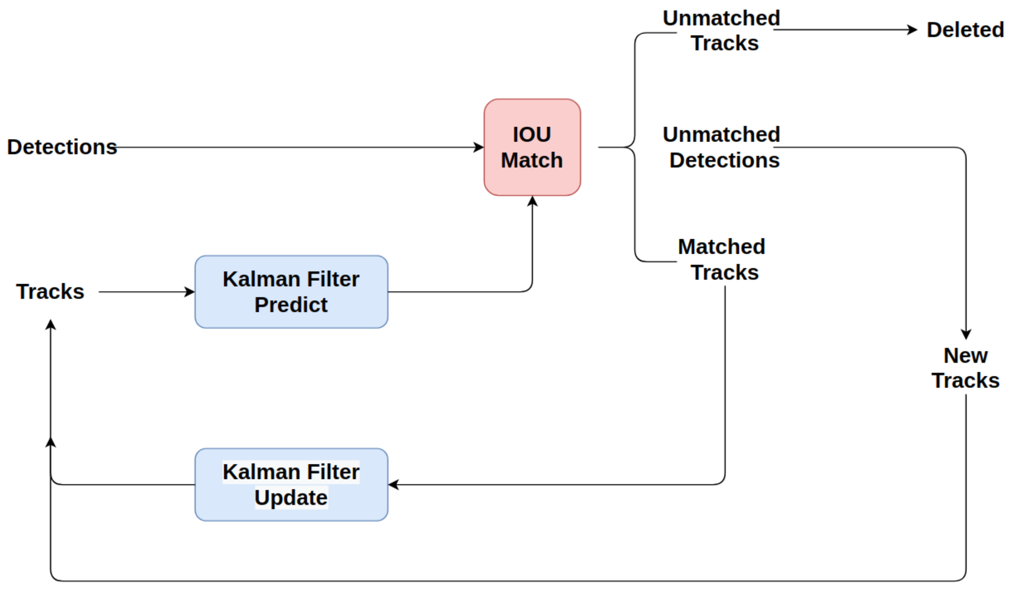
3. RELATED RESEARCH WORKS

3.1 WORKSPACES RELATED TO THE TOPIC

3.1.1 Two-Stage MOT (Tracking-by-detection)

3.1.1.1 Sort

SORT - Simple Online Realtime Object Tracking **[7]**, first introduced in 2016 and updated with version 2 in 2017, proposes a solution for object tracking while addressing both multiple object tracking and real-time object tracking issues.



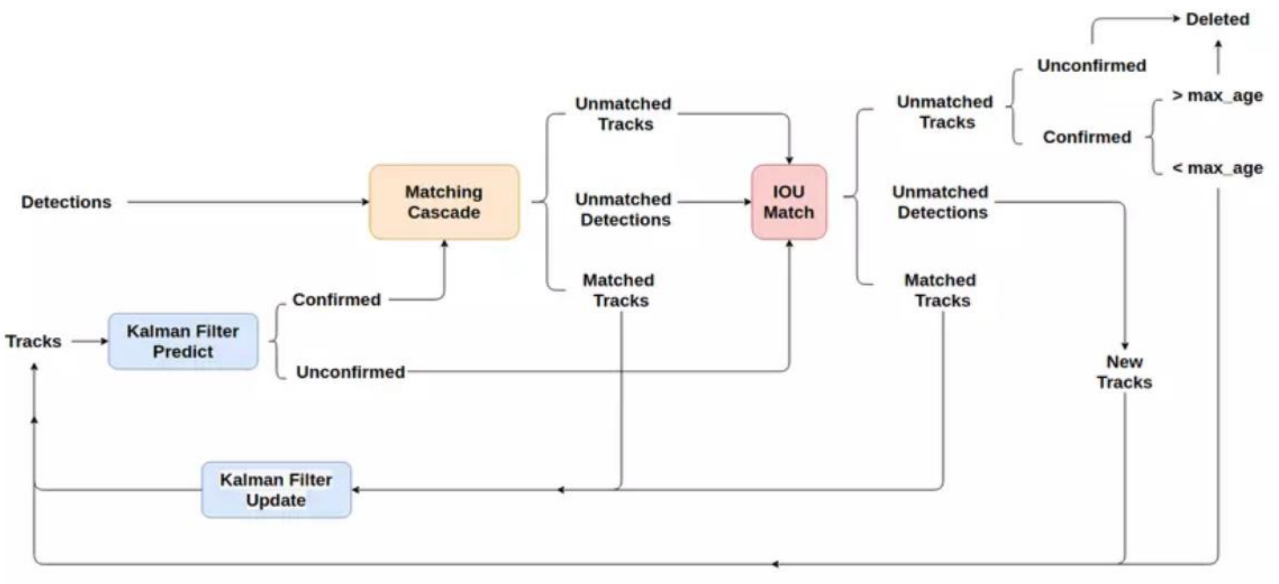
SORT uses the Kalman filter [15] to predict the position of tracks in the next frame based on the past frames.

And then matching detections (from object detection) and tracks by Hungarian matching algorithm (matching IoU).

3.1.1.2 DeepSort

DeepSort (SORT with a Deep Association Metric) [8] is an extension of the SORT algorithm that introduces a deep association metric to improve the tracking performance by matching object features.

DeepSort addresses the limitations of SORT in handling occlusions and identity switches by using deep features for association and re-identification of tracked objects.



DeepSort also introduces a re-identification module that helps to maintain the identity of tracked objects when they are occluded or leave the camera's field of view. The re-identification module uses a Siamese network to compare the appearance of the currently tracked object with the appearance of objects in previous frames and re-identify them if they reappear.

3.1.1.3 Weakness

This shows the weaknesses of two-state methods. These method are completely dependent on object detection.

Slow speed is a problem in processing real-time tracking. So we will utilize a one-stage method in this seminar to improve these weaknesses.

Although one-stage method has a significant impact on practical application through its real-time capability, this method is complicated.

3.1.2 One Stage MOT (End-to-end)

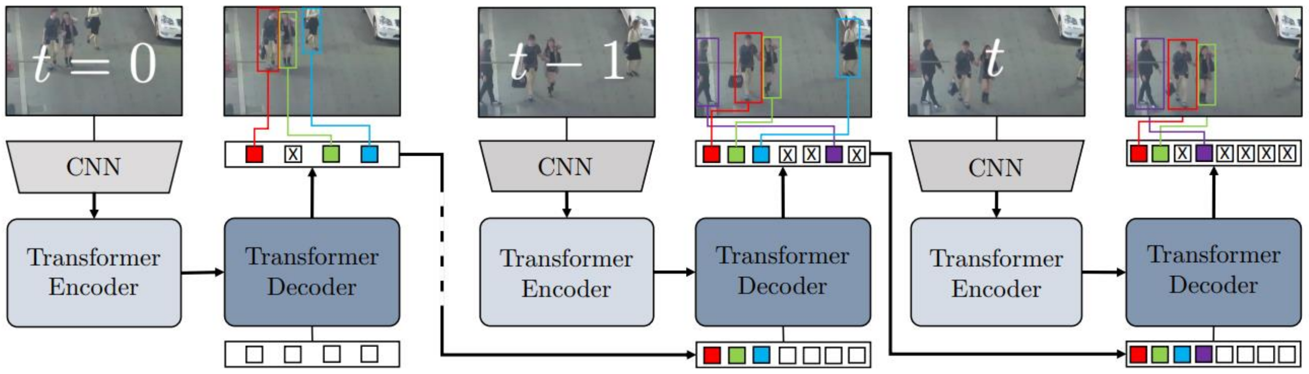
Joint detection and tracking methods are classified as one-stage MOT, in which the detection and tracking steps are simultaneously produced in a **single network**.

In this category, object detection can be modeled within a single network with **re-ID feature extraction** or **motion features**

3.1.2.1 TrackFormer: Multi-Object Tracking with Transformers

TrackFormer utilizes Transformer with two main tasks:

- Transform object queries (white squares) to new tracks queries or background (like object detection task)
- Transform track queries (color squares) from previous frames to tracks in current frame (like matching task)



TrackFormer [10] casts multi-object tracking as a set prediction problem performing joint detection and tracking-by-attention. The architecture consists of a CNN for image feature extraction, a Transformer [11] encoder for image feature encoding and a Transformer decoder which applies self- and encoder-decoder attention to produce output embeddings with bounding box and class information. At frame t = 0, the decoder transforms Nobject object queries (white) to output embeddings either initializing new autoregressive track queries or predicting the background class (crossed). On subsequent frames, the decoder processes the joint set of Nobject + Ntrack queries to follow or remove (blue) existing tracks as well as initialize new tracks (purple).

3.1.2.2 Quasi-Dense Similarity Learning for Multiple Object Tracking

Training pipeline: The training process creates a powerful re-ID network by quasi-dense similarity learning which learns the feature embedding space that can associate identical objects and distinguish different objects for online multiple object tracking.

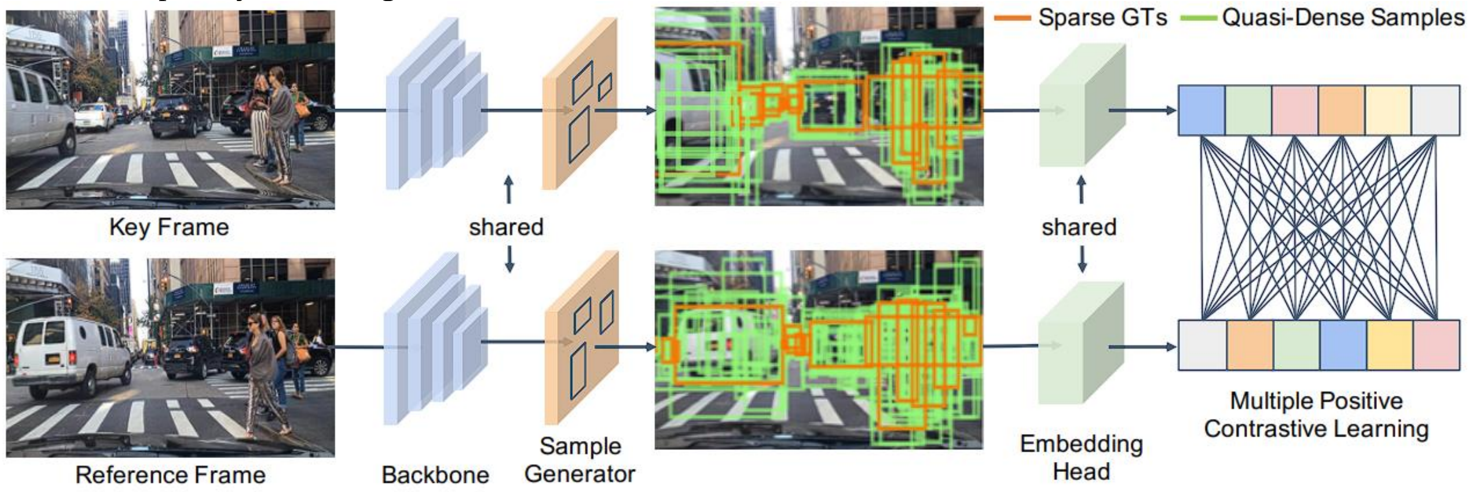


Figure testing pipeline [12]: apply dense matching between quasi-dense samples on the pair of images and optimize the network with multiple positive contrastive learning.

Testing pipeline: utilizes re-ID trained network to match proposal objects in the current/key frame and past/reference frames.

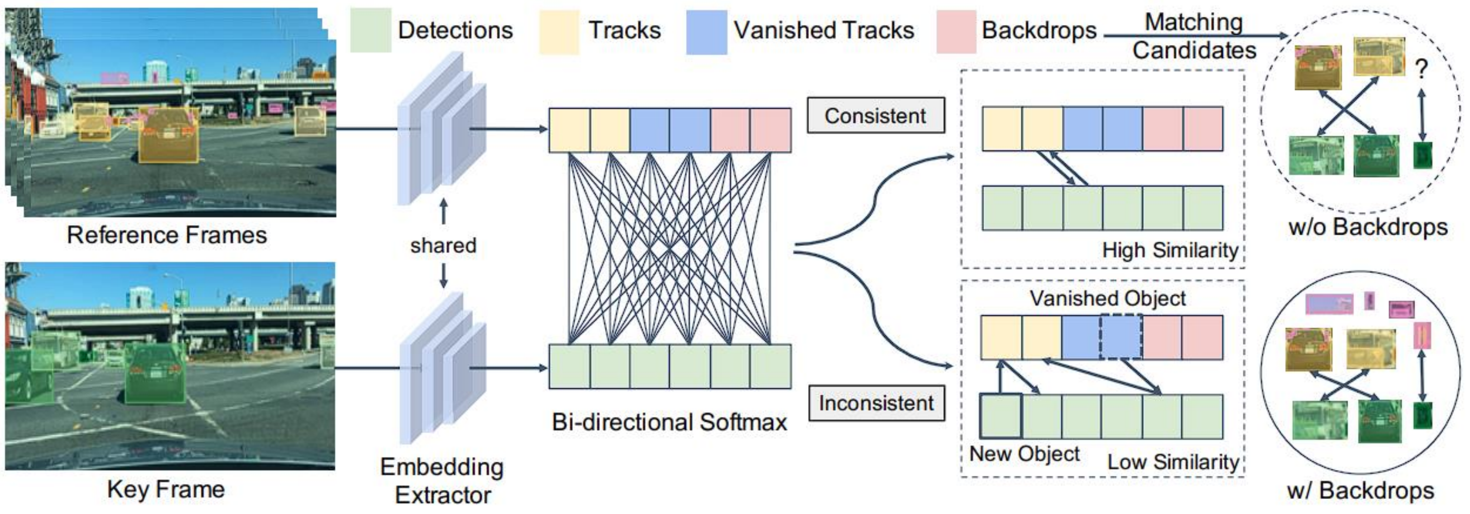


Figure testing pipeline [12]: Maintain the matching candidates and use bi-softmax to measure the instance similarity so that we can associate objects with a simple nearest neighbors search in the feature space.

3.2 SOLUTION COMPARISON

Solution Name	Principle	Methodology				Performance	
		Detection	Estimation Model	Data Association	Creation and Deletion of Track Identities	Dataset	Metrics
SORT[7]-Simple Online Realtime Object Tracking	Including a constant-velocity motion model, a data association model, a deep association metric, a **Kalman filter**, and non-maximum suppression, enable it to achieve real-time performance while maintaining high tracking accuracy.	Utilising the Faster Region CNN (**FrRCNN**) detection framework.	The motion model used to propagate a target's identity into the next frame: $x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T$ Using a **Kalman filter** framework to solve the velocity components.	The assignment cost matrix is then computed as the intersection-over-union (**IOU**) distance	For creating trackers, we consider any detection with an overlap less than **IOU**$_{\min}$ to signify the existence of an untracked object.	**MOT** benchmark database	MOTA MOTP FAF MT,ML FP,FN IDsw Frag
		Track Handling and State Estimation	Assignment Problem	Matching Cascade	Deep Appearance Descriptor		
DeepSort [8]-SORT with a Deep Association Metric	The **DeepSORT** algorithm improves upon the **SORT** algorithm by incorporating a deep association metric and feature extraction enhancements.	The track handling and **Kalman filtering** framework.	Use the (squared) **Mahalanobis** distance between predicted **Kalman states** and newly arrived measurements:	Compute cost matrix $C = [c_{i,j}]$ using: $c_{i,j} = \lambda d(i,j) + (1 - \lambda) d(i,j)$ Compute gate matrix $B = [b_{i,j}]$ using:	Employ a **CNN** that has been trained on a large-scale person **re-identification dataset**. A wide residual network with two convolutional	MOT16 challenge dataset	MOTA MOTP MT ML ID,FM FP,FN Runti--me

Solution Name	Principle	Methodology				Performance	
			$d(i,j) = (d_j - y_i)^T {S_i}^{-1} (d_j - y_i)$	$b_{i,j} = \prod_{m=1}^{2} b_{i,j}^{(m)}$	layers followed by six residual blocks.		
		Track initialization	Track queries	Track query re-identification	TrackFormer training	Dataset	Metrics
TrackFor--mer [10]	Detecting objects in each frame of a video sequence, transformer network to extract features from the detected objects, matching objects across frames using a bipartite matching algorithm and trained end-to-end.	New objects appearing in the scene are detected by a fixed number of N_{object} output embedding	Casting multi-object tracking as a set prediction problem performing joint detection and **tracking-by-attention**.	Searching for the injective minimum cost mapping $\hat{\sigma}$ in the following assignment problem $:\hat{\sigma} = \text{argmin} \sum_{k_i \in Kobject}$	The C_{box} term penalizes bounding box differences by a combination of ℓ_1 distance and generalized intersection over union (IoU) cost C_{iou}, $C_{box} = \lambda_{\ell 1} \|\|b_i - b_{\sigma(i)}\|\|$	MOT17 MOT20 MOTS20 challenge dataset	MOTA IDF1 MT,ML FP,FN ID Sw.
		Object detection	Quasi-dense similarity learning	Object association		Dataset	Metrics
Quasi-Dense Similarity Learning [12]	Building upon the SORT tracker by integrating a novel similarity learning module to better handle occlusions, false alarms, and missed detections. The similarity learning module is based on a quasi-dense sampling strategy.	Taking **Faster R-CNN** with **Feature Pyramid Network** (FPN).	Using the region proposals generated by **Region Proposal Network** (RPN) to learn the instance similarity with quasi-dense matching.	Main inference strategy is **bi-directional matching** in the embedding space. Intra-class duplicate removal by **None Maximum Suppression** (NMS).		MOT16 MOT17 BDD100K Waymo TAO	MOTA IDF1 MOTP MT,ML FP,FN IDs

Solution Name	Advantages	Disadvantages
SORT [7]-Simple Online Realtime Object Tracking	SORT is designed for real-time applications, and it can process video frames at high frame rates with low computational overhead	It struggles to maintain high accuracy in crowded scenes where multiple objects are in close proximity to each other.
	Easy to implement, as it uses a straightforward pipeline that consists of object detection, data association, and Kalman filtering	The SORT algorithm is designed for short-term tracking, and it may fail to track objects over long periods.
	Despite its simplicity, the SORT algorithm achieves high accuracy in object tracking tasks.	It relies on the quality of object detections, and it may suffer from tracking errors if the detections are inaccurate or incomplete.
DeepSort [8]-SORT with a Deep Association Metric	DeepSORT achieves high accuracy more than SORT in object tracking tasks by combining a deep neural network with a classical tracking algorithm	DeepSORT requires significant computational resources due to the use of a deep neural network for feature extraction.
	It is robust to occlusions and partial object detections, as it uses appearance information to associate objects across frames	It still relies on the quality of object detections, and it may suffer from tracking errors if the detections are inaccurate or incomplete which is the same cons as the SORT method.
	DeepSORT can track objects over long periods, as it incorporates temporal information and memory of past detections.	DeepSORT is more complex to implement than simpler tracking algorithms like SORT.
TrackFormer [10]	Achieves state-of-the-art results on various datasets for multi-object tracking tasks.	This method requiries large amounts of data to train effectively.
	Allowing for parallelization and efficient computation, making it suitable for real-time tracking applications.	Sensitive to the choice of hyperparameters, as the reason for time-consuming.
	Able to handle occlusions and partial visibility of objects.	Limited to a fixed number of objects.
	Simplified the training process and allows for better optimization by end-to-end training.	
Quasi-Dense Similarity Learning [12]	This method achieves state-of-the-art performance on multiple benchmark datasets for multiple object tracking	It is currently still limited to tracking a single object category.
	Be able to handle challenging scenarios such as occlusion, motion blur, and illumination changes.	Highly dependent on the accuracy of the object detection module.
	It is computationally efficient and can operate in real-time scenarios.	May not be able to maintain tracking continuity over a long time horizon.
	Easily adapted to different object tracking scenarios due to its simple design.	The QDTrack method uses a similarity metric that may not be easily interpretable.

4. METHODOLOGY

4.1 OBJECT DETECTION

In the first step, the methodology for Object Tracking can be easily coupled with most existing detectors with end-to-end training.

It takes Faster R-CNN [13] with Feature Pyramid Network (FPN) [14] as an example [12]. First of all, Faster R-CNN is a two-stage detector that uses Region Proposal Network (RPN) to generate Region of Interests (RoIs). Then it localizes and classifies the regions to obtain semantic labels and locations.

Based on Faster R-CNN, FPN exploits lateral connections to build the top-down feature pyramid and tackles the scale-variance problem. The entire network is optimized with a multi-task loss function:

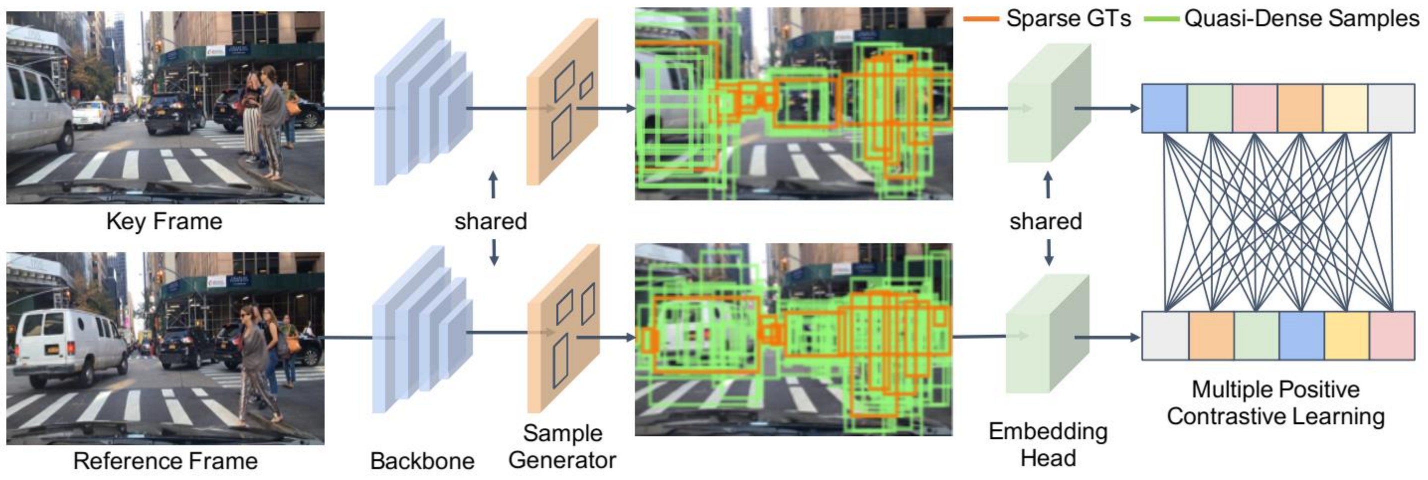
$$\mathcal{L}_{det} = \underbrace{\mathcal{L}_{rpn}}_{\text{RPN loss}} + \lambda_1 \underbrace{\mathcal{L}_{cls}}_{\text{Classification Loss}} + \lambda_2 \underbrace{\mathcal{L}_{reg}}_{\text{Regression Loss}} \quad (1)$$

The loss weights λ_1 and λ_2 are set to 1.0 by default.

- Input: Single image
- Output: Region proposals which include bounding boxes and feature maps (by Roi Align)

4.2 QUASI-DENSE SIMILARITY LEARNING

The region proposals generated by RPN to learn the instance similarity with quasi-dense matching.



As shown in the training pipeline above, given a key image I1 for training, we randomly select a reference image I2 from its temporal neighborhood. The neighbor distance is constrained by an interval k, where $k \in [-3; 3]$.

By using RPN to generate RoIs from the two images and RoI Align [15] to obtain their feature maps from different levels in FPN according to their scales. Finally, extra lightweight embedding head is added to extract features for each RoI.

RoI is defined as positive to an object if they have an IoU higher than α1, or negative if they have an IoU lower than α2. α1 and α2 are 0.7 and 0.3 in our experiments.

Assume there are V samples on the key frame as training samples and K samples on the reference frame as contrastive targets. For each training sample, we can use the non-parametric softmax with cross-entropy to optimize the feature embeddings:

$$\mathcal{L}_{\text{embed}} = -\log \frac{\exp(\text{v.k}^+)}{\exp(\text{v.k}^+) + \sum_{\text{k}^-} \exp(\text{v.k}^-)} \quad \textbf{(a)}$$

Each training sample on the key frame has more than one positive targets on the reference frame, so equation **(a)** can be extended as:

$$\mathcal{L}_{embed} = \log\,[1 + \sum_{k^+}\sum_{k^-} \exp(v.k^- - v.k^+)] \quad \textbf{(2)}$$

Moreover, with an auxiliary loss:

$$\mathcal{L}_{aux} = (\frac{v.k}{||v||.||k||} - c)^2 \quad \textbf{(3)}$$

The aim for this auxiliary loss is to constrain the logit magnitude and cosine similarity. So whole network is joint optimized under by **(1)**, **(2)** and **(3)**:

$$\mathcal{L} = \mathcal{L}_{det} + \gamma_1\mathcal{L}_{embed} + \gamma_2\mathcal{L}_{aux}$$

where γ_1 and γ_2 are set to 0.25 and 1.0 by default [12].

- Input: Key frame, reference frame.
- Output: each frame has own region proposals which include bounding boxes and feature embeddings (feature maps of RoIs are extracted by embedding head).

4.3 OBJECT ASSOCIATION

4.3.1 Bi-directional softmax

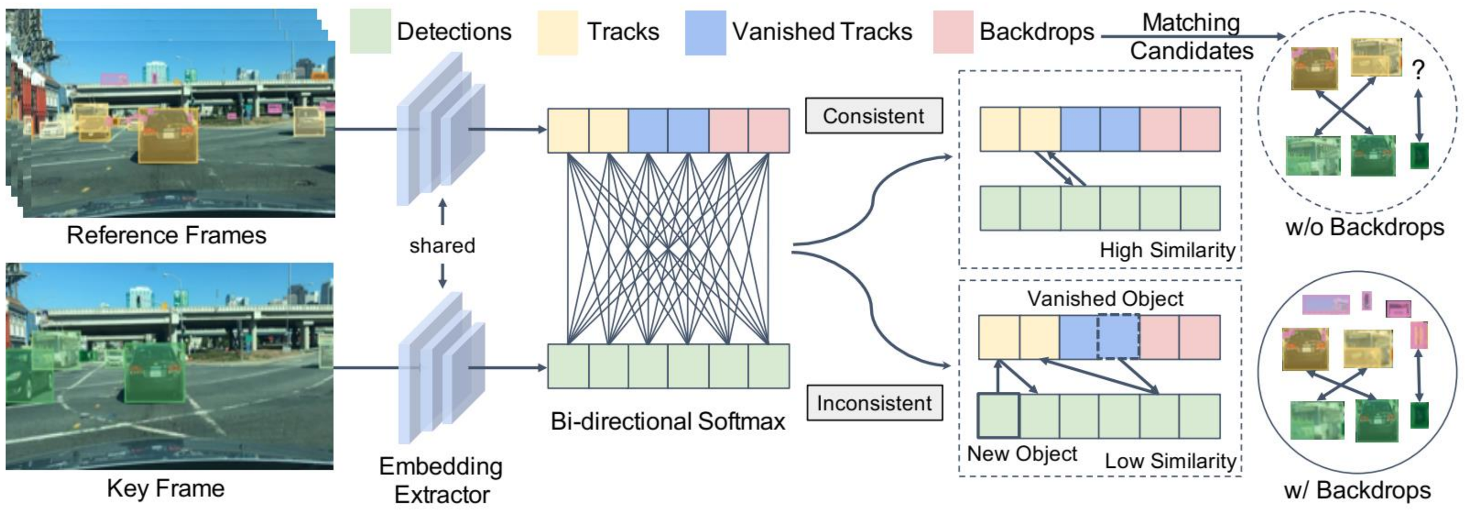


Figure 3 [12]: The testing pipeline of our method. We maintain the matching candidates and use bi-softmax to measure the instance similarity so that we can associate objects with a simple nearest neighbour search in the feature space.

Supposing there are **N detected objects** in frame t with feature embeddings n, and **M matching candidates** from the past x frames with feature embeddings m, the similarity f between the objects and matching candidates is obtained by bi-directional softmax (bi-softmax):

$$f(i,j) = \left[\frac{\exp(n_i . m_j)}{\sum_{k=0}^{M-1} \exp(n_i . m_k)} + \frac{\exp(n_i . m_j)}{\sum_{k=0}^{N-1} \exp(n_k . m_j)}\right] / 2$$

The instance similarity f can directly associate objects with a simple nearest neighbor search.

4.3.2 Considering cases

No target cases:

- Objects without a target in the feature space should not be matched to any candidates.
- The bi-softmax is proposed as a solution to this problem, as it can handle objects that cannot obtain bi-directional consistency.
- Backdrops, which are unmatched objects that are kept during matching as they can be useful regions that future objects are likely to match. By keeping backdrops, the number of false positives in object tracking can be reduced.

Multi-targets cases:

- Object association where state-of-the-art detectors only use Non-Maximum Suppression (NMS) for intra-class duplicate removal, which can lead to objects at the same locations having different categories.
- This can boost object recall and contribute to a high mean Average Precision (mAP) but creates duplicate feature embeddings. In order to address this, inter-class duplicate removal is done by NMS using different IoU thresholds depending on the detection confidence level.
- A higher IoU threshold is used for objects with high detection confidence, and a lower IoU threshold is used for objects with low detection confidence.

- Input: Key frame, reference frames.
- Output: Track identification for each object (after matching successful embedding feature by “bi-softmax”) in key frame.

5. EXPERIMENTS

5.1 DATASET

5.1.1 MOT Challenge

5.1.1.1 Design Principle

In this topic, our team performed experiments on two MOT benchmarks: MOT16 and MOT17 **[16]**. The datasets contain videos captured from real-world scenarios, such as pedestrian areas, public events, and shopping malls. This ensures that the datasets are representative of the challenges that MOT algorithms face in real-world situations.

The datasets contain multiple challenges, including occlusions, crowded scenes, and varying lighting conditions. This ensures that the datasets can be used to evaluate the robustness of MOT algorithms.

5.1.1.2 Data Collection

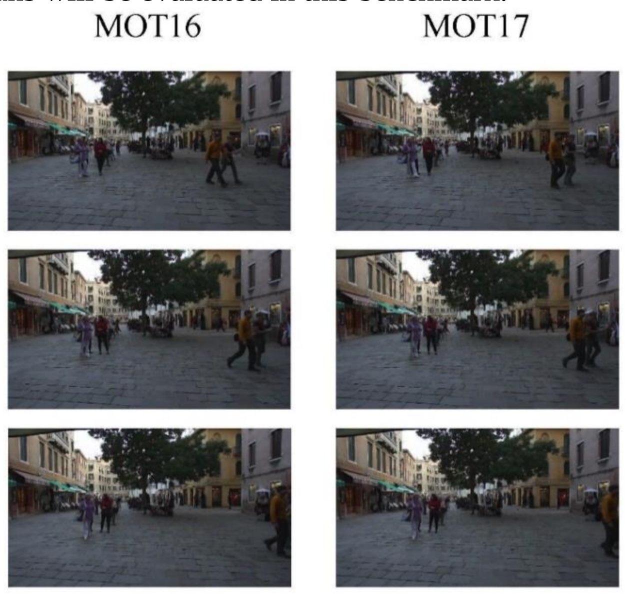
Sequences are very different from each other; we can classify them according to:

- Moving or static camera – the camera can be held by a person, placed on a stroller or on a car, or can be positioned fixed in the scene.
- Viewpoint – the camera can overlook the scene from a high position, a medium position (at pedestrian's height), or a low position.
- Conditions – the weather conditions in which the sequence was taken are reported in order to obtain an estimate of the illumination conditions of the scene.

5.1.1.3 Dataset Split

The dataset contains 7 videos (5,316 images) for training and 7 videos (5,919 images) for testing with the video frame rate being approximately between 14 and 30 frames per second (FPS).

In addition, only pedestrians will be evaluated in this benchmark.



An overview of the MOT16 and MOT17 dataset

5.1.2 DanceTrack

5.1.2.1 Design Principle

DanceTrack **[5]** focuses on the scenarios where objects have similar or even the same appearance and diverse motion patterns, including frequent crossover, occlusion and body deformation.

This dataset also focuses on situations where multiple objects are moving in a "relatively" large range, where the occluded areas are dynamically changing, and they are even in crossover.

5.1.2.2 Data Collection

To achieve the design goals described above, we collected videos including mostly group dancing from the Internet. As shown in Figure 2, the dancers usually wear very similar or even the same clothes. They make a large-range motion, diverse gestures and frequent crossover.

The videos used in this dataset from different search engines with keywords like "classical dance", "street dance", "pop dance", "large group dance" and "sports".

5.1.2.3 Dataset Split

DanceTrack consists of 100 videos by default using 40 videos as training set, 25 as validation set and 35 as testing set while still keeping the distribution of subsets close in terms of average length, average bounding box number, scenes and the motion diversity with the video frame rate for each video is roughly 20 frame per second.

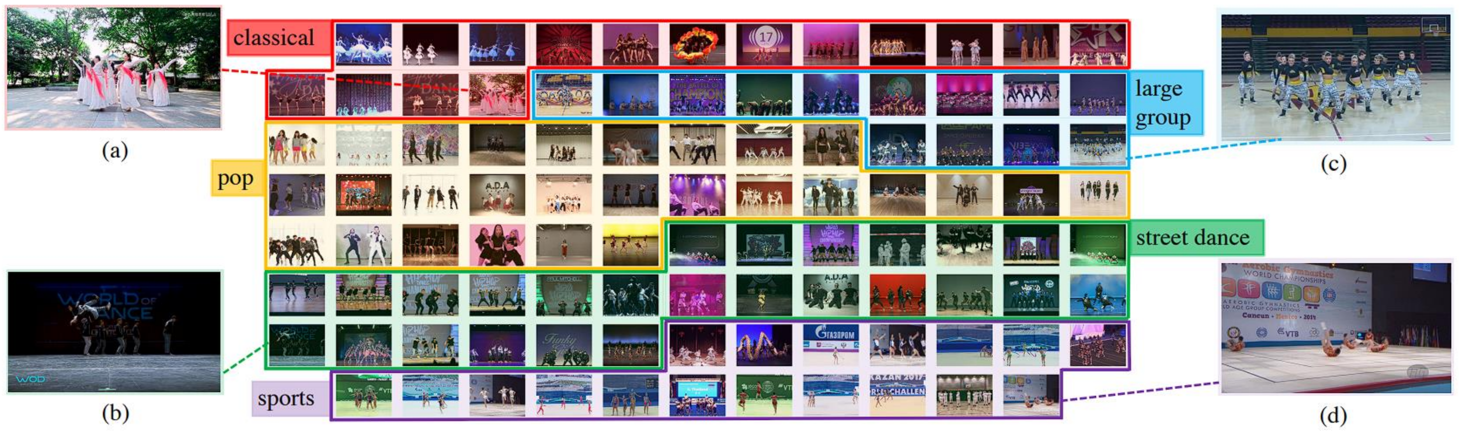


Figure 2 **[5]** – Sampled scenes from DanceTrack dataset. DanceTrack contains multiple genres of dance, including classical dance, street dance, pop dance, large group dance and sports. The scenes in DanceTrack are diverse: (a) outdoor scenes; (b) low-lighting and distant camera scenes; (c) large group of dancing people; (d) gymnastics scene where the motion is usually even more diverse, and people have more aggressive deformation.

5.1.3 AnimalTrack

5.1.3.1 Design Principle

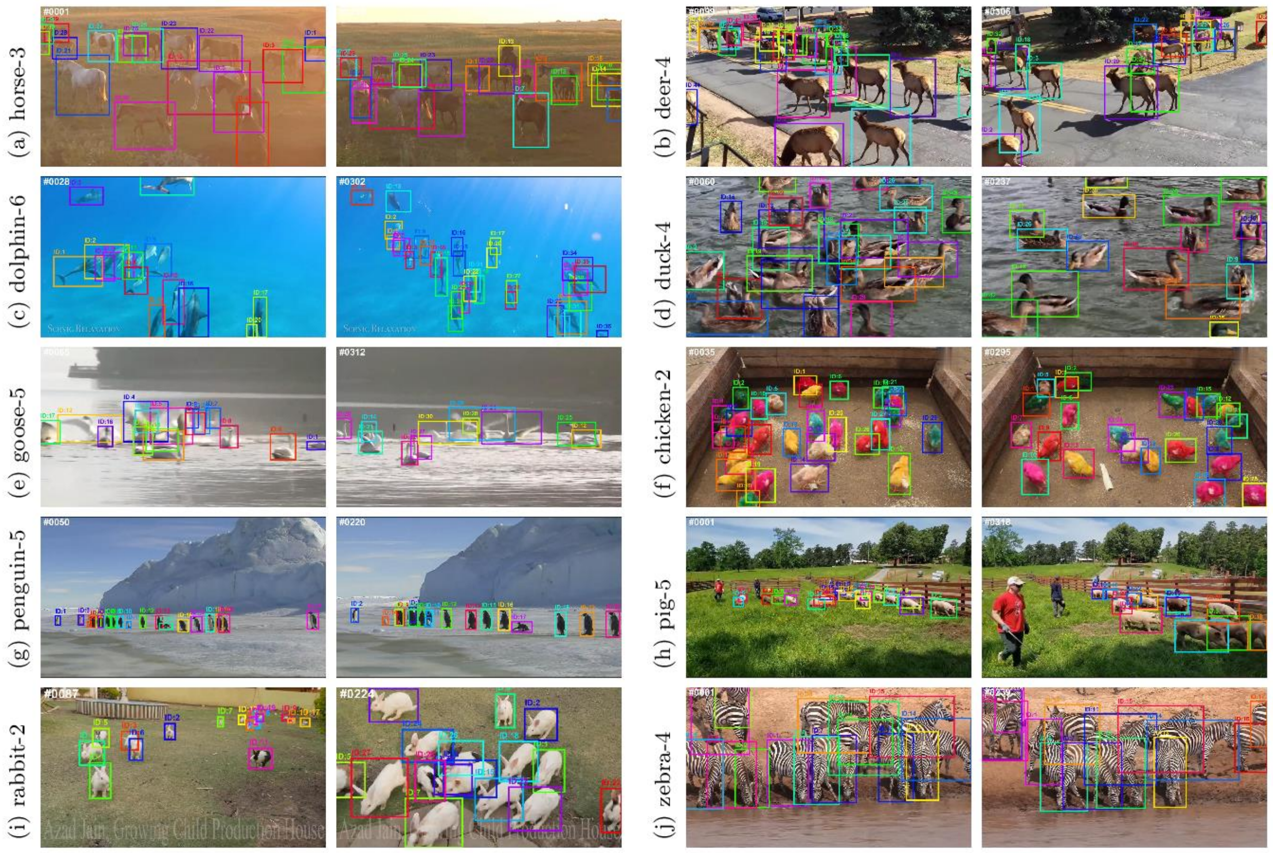
AnimalTrack **[6]** expects to provide the community with a new dedicated platform for studying MOT on animals. In particular, in the deep learning era, it aims at both training and evaluation for deep trackers.

5.1.3.2 Data Collection

AnimalTrack focuses on dense multi-animal tracking. We start benchmark construction by selecting 10 common animal categories that are generally dense and crowded in the wild. These categories are Chicken, Deer, Dolphin, Duck, Goose, Horse, Penguin, Pig, Rabbit, and Zebra, perching in very different environments.

5.1.3.3 Dataset Split

AnimalTrack consists of 58 video sequences. By utilizing 32 videos for the training sets and the rest 26 videos for testing sets; and the video frame rate is at 30 frames per second level.



5.2 SOURCE CODE TEAM DEVELOPED

5.2.1 Pseudocodes

The entire operation of the project code are as the following pseudocodes:

```
Class Embedding_Extractor:
    Function init(config):
        self.backbone = create _backbone(config.backbone_name)
        self.rpn = create_region_proposal_network(config.rpn)
        self.box_head = create_convolution2D(config.box_head)
        self.embedding_head = create_convolution2D(config.embed_head)
    Function forward(single_frame):
        visual_features = self.backbone(single_frame)
        RoIs_features = self.rpn(self.visual_features)
        RoIs_boxes = self.box_head(self.visual_features)
        RoIs_features = self.embedding_head(RoIs_features)
        Return {“boxes”: RoIs_boxes, “features”: RoIs_features}
```

```
Class QDTrack():
    Function init(config):
        self.extractor = Embedding_Extractor(config)
        self.tracklets = []
        # select similarity function (”Bi-directional softmax” or “cosine")
        self.similariy_function = select_similarity_function(default=”Bi-directional softmax”)
    Function forward(single_frame):
        RoIs = self.extractor(single_frame)
        RoIs_features = RoIs[“features”]
        Tracklets_features = self.tracklets[“features”]
        Similarity_scores = self.similariy_function(RoIs_features, Tracklets_features)
        Assigned_RoIs = Assign_ID(RoIs, self.tracklets, Similariy_scores, config.threshold)
        Update Assigned_RoIs to self.tracklets
        Return self.tracklets
```

```
Main_function(config, video):
    model = QDTrack(config)
    track_result = [] # [frame_id, track_id, x_left, y_left, width, height, 1, 1, 1]

    For each frame in sequence of video:
        Result = model(frame)
        Record_and_show_result(track_result, Result)
    Save track_result to file
```

5.2.2 Implementation detailed

In this report, our team used ResNet-50 **[12]** as the backbone.

By selecting 128 RoIs from the key frame as training samples, and 256 RoIs from the reference frame with a positive-negative ratio of 1.0 as contrastive targets.

Using IoU-balanced sampling to sample RoIs and *4conv1fc* head with group normalization to extract feature embeddings, and the channel number of embedding features is set to 256.

At first, the models are trained with a total batch size of 16 and an initial learning rate of 0.02 for 12 epochs, then it is decreased the learning rate by 0.1 after 8 and 11 epochs.

Here, the backdrops are only kept for one frame. The objects can be associated only when they are classified as the same category.

Finally, for dataset setting, the DanceTrack is randomly resized and cropped the longer side of the images to 1088 and it is not changed the aspect ratio at the training and inference time.

5.3 DEMONSTRATION THE PROGRAM

The DanceTrack dataset is a video dataset of multiple people dancing in a crowded environment, where occlusions and interactions among the dancers are common.

In this project, our team would take up to 25 videos in validate set with the length of each video from 80 seconds to 160 seconds. The dataset provides ground truth annotations for the positions and identities of the dancers in each frame.

Our team has proposed the dataset of DanceTrack by the following links: bit.ly/3LiivfS

Our team has proposed a demonstration of how to run a source code which applys Quasi-dense Similarity Learning for Multiple Object Tracking method on DanceTrack dataset as the following link: bit.ly/3HpPgXf

5.4 MAIN RESULTS

5.4.1 Metrics description

The Quasi-dense Track outperforms all existing methods on aforementioned benchmarks without bells and whistles. The performance are evaluated with the official metrics.

MOTA [17, 18]: Multiple Object Tracking Accuracy. This metric measures the overall accuracy of both the tracker and detection. It deals with both tracker output and detection output.

MOTP [17, 18]: short for Multiple Object Tracking Precision. This metric measures the accuracy of localization of detection boxes. It deals with only detection output from the model and not doing anything with tracker output.

To compute the MOT values, a standard approach to find the correspondence between Ground Truth (GT) and Prediction are compulsory.

By using Intersection over Union (IOU), a metric to evaluate the object detector's accuracy. It is calculated between two bounding boxes' (Ground truth & Predicted bounding boxes) overlap area with union area.

$$\mathrm{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

MOTA Calculation:

Step 1: The correspondence will be made between Ground truth and predicted objects to ensure that those are correct matches.

Step 2: If the Ground Truth object has NO MATCH in detection output, then the count of MISS(FN_t) will be incremented by 1.

Step 3: If the object in detection output has NO MATCH in the real world (Ground Truth), it will be considered false detection. In that case, the count of FALSE POSITIVES(FP_t) will be incremented by 1.

Step 4: As mentioned earlier, these errors will be calculated for a series of frames. In the end, we calculate MOTA with the following formula.

$$\mathrm{MOTA} = 1 - \frac{\sum_t \mathrm{FN}_t + \mathrm{FP}_t + \mathrm{IDS}_t}{\sum_t \mathrm{GT}_t}$$

where FN is the number of false negatives, FP is the number of false positives, IDS is the number of identity switches, and GT is the total number of ground truth objects. A higher value of MOTA indicates better tracking performance.

In the meantime, MOTP will be calculated with the following formula:

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t}$$

where d_t – Distance between the localization of objects in the ground truth and the detection output, and c_t – total matches made between ground truth and the detection output.

HOTA [18] (Higher Order Tracking Accuracy) is designed to:

- Providing a single score for tracker evaluation which fairly combines all different aspects of tracking evaluation.
- Evaluating long-term higher-order tracking association.
- Decomposing into submetrics which allow analysis of the different components of tracker's performance.

Measuring Association:
For a given TP, *c*, the set of TPAs is the set of TPs which have both the same gtID and the same prID as *c*:

$$\begin{aligned}&\mathrm{TPA}(c) = \{k\},\\ &k \in k\{\mathrm{TP} \mid \mathrm{prID}() = \mathrm{prID}(c) \land \mathrm{gtID}(k) = \mathrm{gtID}(c)\}\end{aligned} \quad \textbf{(4)}$$

For a given TP, *c*, the set of FNAs is the set of gtDets with the same gtID as *c*, but that were either assigned a different prID as *c*, or no prID if they were missed:

$$\begin{aligned}&\mathrm{FNA}(c) = \{k\},\\ &k \in \{\mathrm{TP} \mid \mathrm{prID}(k) \neq \mathrm{prID}(c) \land \mathrm{gtID}(k) = \mathrm{gtID}(c)\}\\ &\quad \cup \{\mathrm{FN} \mid \mathrm{gtID}(k) = \mathrm{gtID}(c)\}\end{aligned} \quad \textbf{(5)}$$

Finally, for a given TP, *c*, the set of FPAs is the set of prDets with the same prID as *c*, but that were either assigned a different gtID as *c*, or no gtID if they did not actually correspond to an object:

$$\begin{aligned}&\mathrm{FPA}(c) = \{k\},\\ &k \in \{\mathrm{TP} \mid \mathrm{prID}(k) = \mathrm{prID}(c) \land \mathrm{gtID}(k) \neq \mathrm{gtID}(c)\\ &\quad \cup \{\mathrm{FP} \mid \mathrm{prID}(k) = \mathrm{prID}(c)\}\end{aligned} \quad \textbf{(6)}$$

By define the HOTA_α score for a particular localisation threshold α for a scoring function from **(4)**, **(5)** and **(6)** equations:

$$\mathcal{A}(c) = \frac{|\mathrm{TPA}(c)|}{|\mathrm{TPA}(c)| + |\mathrm{FNA}(c)| + |\mathrm{FPA}(c)|} \quad \textbf{(7)}$$

$$HOTA_{\alpha} = \sqrt{\frac{\sum_{c\in\{TP\}} \mathcal{A}(c)}{|TP| + |FN| + |FP|}}$$

Integrating over Localisation Thresholds: the final HOTA score is the integral (area under the curve) of the HOTA score across the valid range of α values between 0 and 1 from equation (7):

$$HOTA = \int_0^1 HOTA_{\alpha} \approx \frac{1}{19} \sum_{\alpha\in\{0.05,0.1,\ldots,0.9,0.95\}} HOTA_{\alpha}$$

IDF1 [17, 18] (Identity F1 score): This metric measures the identity tracking accuracy of a method by computing the F1 score of the intersection over union (IoU) between ground truth and predicted bounding boxes. It is defined as:

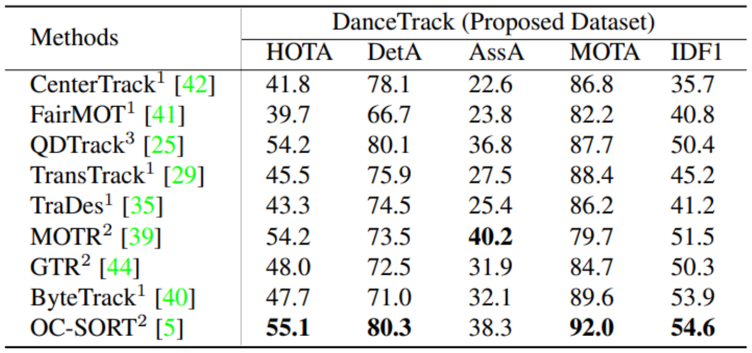
$$\mathrm{IDF1} = \frac{|\mathrm{IDTP}|}{|\mathrm{IDTP}| + 0.5\,|\mathrm{IDFN}| + 0.5\,|\mathrm{IDFP}|}$$

where IDTP is the number of true positives, IDFP is the number of false positives, and IDFN is the number of false negatives. A higher value of IDF1 indicates better identity tracking performance.

MT [17] (Mostly Tracked): It measures the percentage of ground truth trajectories that are tracked for at least 80% of their duration. A higher value of MT indicates better tracking performance.

ML [17] (Mostly Lost): ML measures the percentage of ground truth trajectories that are tracked for less than 20% of their duration. A lower value of ML indicates better tracking performance.

5.4.2 Results comparison



Methods	DanceTrack (Proposed Dataset)				
	HOTA	DetA	AssA	MOTA	IDF1
CenterTrack[1] [42]	41.8	78.1	22.6	86.8	35.7
FairMOT[1] [41]	39.7	66.7	23.8	82.2	40.8
QDTrack[3] [25]	54.2	80.1	36.8	87.7	50.4
TransTrack[1] [29]	45.5	75.9	27.5	88.4	45.2
TraDes[1] [35]	43.3	74.5	25.4	86.2	41.2
MOTR[2] [39]	54.2	73.5	**40.2**	79.7	51.5
GTR[2] [44]	48.0	72.5	31.9	84.7	50.3
ByteTrack[1] [40]	47.7	71.0	32.1	89.6	53.9
OC-SORT[2] [5]	**55.1**	**80.3**	38.3	**92.0**	**54.6**

Table 1: DanceTrack [5]: Multi-Object Tracking in Uniform Appearance and Diverse Motion

Table 1 proposes the tracking performance of investigated algorithms on the DanceTrack test set. The result comparison shows the evidently increased difficulty of performing multi-object tracking on the the DanceTrack dataset. This project has benchmarked the current state-of-the-art multi-object tracking algorithms on DanceTrack. The evaluation is in the “private” setting where the algorithm performs both detection and association. The methods inherit the settings on the DanceTrack training set. The Quasi-Dense Tracking Method proposed a significant performance gap at the level of 54.2 in the HOTA tracking quality measured. This suggests that detection is not the bottleneck to having good tracking performance on DanceTrack and further highlights the drop in association performance for Quasi-Dense Tracking to the others.

Tracker	HOTA	MOTA	IDF1	IDP	IDR	MT	PT	ML↓	FP↓	FN↓	IDs↓	FM↓
SORT [6]	42.8%	55.6%	49.2%	58.5%	42.4%	333	470	301	19,099	86,257	2,530	3,730
IOUTrack [7]	41.6%	55.7%	45.7%	51.9%	40.7%	**388**	454	**262**	25,206	**77,847**	4,639	5,259
DeepSORT [54]	32.8%	41.4%	35.2%	49.7%	27.2%	213	452	439	14,131	124,747	3,503	4,527
JDE [52]	26.8%	27.3%	31.0%	51.0%	22.0%	106	414	584	17,887	155,623	3,187	5,031
FairMOT [62]	30.6%	29.0%	38.8%	62.8%	28.0%	143	462	499	17,653	152,624	2,335	5,447
CenterTrack [63]	9.9%	1.6%	7.0%	8.9%	5.8%	265	423	416	32,050	117,614	89,655	7,583
CTracker [41]	13.8%	14.0%	14.7%	35.2%	9.3%	20	313	771	**13,092**	192,660	3,437	8,019
Tracktor++ [3]	44.2%	**55.2%**	51.0%	58.5%	45.1%	364	472	**268**	25,477	**81,538**	**1,976**	4,149
ByteTrack [61]	40.1%	38.5%	51.2%	**64.9%**	42.3%	310	465	329	31,591	116,587	**1,309**	**3,513**
QDTrack [39]	**47.0%**	**55.7%**	**56.3%**	**65.6%**	**49.3%**	**367**	420	317	22,696	83,057	1,970	5,656
TADAM [23]	32.5%	36.5%	37.2%	44.4%	32.0%	258	**495**	351	41,728	110,048	2,538	4,469
OMC [29]	43.0%	53.4%	50.3%	61.8%	42.4%	324	478	302	**15,910**	92,570	4,938	7,162
Trackformer [37]	31.0%	20.4%	36.5%	40.9%	32.8%	230	**491**	383	70,404	118,724	4,355	**3,725**
TransTrack [48]	**45.4%**	48.3%	**53.4%**	63.4%	**46.1%**	327	416	361	28,553	95,212	1,978	6,459

Table 2: AnimalTrack **[6]**: A Benchmark for Multi-Animal Tracking in the Wild

Table 2 comparison, it shows the overall evaluation results and comparison of different tracking algorithms on AnimalTrack. The best two results on each metric are highlighted in red and blue fonts. It extensively evaluates 14 state-of-the-art tracking algorithms.

From **Table 2**, we observe that Quasi-Dense Tracking shows the best overall result by achieving a 47.0% HOTA score and TransTrack the second best with a 45.4% HOTA score, respectively. Quasi-Dense Tracking densely samples numerous regions from images for similarity learning and thus can alleviate the problem of complex animal poses in detection to some degree, as evidenced by its best result of 55.7% on MOTA that focuses more on detection quality. This dense sampling strategy not only improves detection but also benefits later association, which is shown by its best 56.3% IDF1 score.

Dataset	Method	MOTA ↑	IDF1 ↑	MOTP ↑	MT ↑	ML ↓	FP ↓	FN ↓	IDs ↓
MOT16	TAP [57]	64.8	**73.5**	78.7	292	164	12980	50635	**571**
	CNNMTT [29]	65.2	62.2	78.4	246	162	6578	55896	946
	POI* [54]	66.1	65.1	**79.5**	258	158	**5061**	55914	3093
	TubeTK_POI* [35]	66.9	62.2	78.5	296	**122**	11544	47502	1236
	CTrackerV1 [37]	67.6	57.2	78.4	250	175	8934	48305	1897
	O QDTrack	**69.8**	67.1	79.0	**316**	150	9861	**44050**	1097
MOT17	Tracktor++v2 [2]	56.3	55.1	78.8	498	831	**8866**	235449	1987
	Lif_T* [20]	60.5	65.6	78.3	637	791	14966	206619	**1189**
	TubeTK* [35]	63.0	58.6	78.3	735	**468**	27060	177483	4137
	CTrackerV1 [37]	66.6	57.4	78.2	759	570	22284	160491	5529
	CenterTrack* [56]	67.8	64.7	78.4	816	579	18498	160332	3039
	O QDTrack	**68.7**	**66.3**	**79.0**	**957**	516	26589	**146643**	3378

Table 3: Quasi-Dense Similarity Learning for Multiple Object Tracking **[12]** Results on MOT16 and MOT17 test set with private detectors. ↑ means higher is better, ↓ means lower is

Finally, **Table 3** shows the benchmarks on the MOT16 and MOT17 datasets of Quasi-Dense Tracking with the other methods. The Quasi-Dense Tracking achieves the best MOTA of 68.7% and IDF1 of 66.3% on the MOT17. The method outperforms the state-of-the-art tracker CenterTrack by 0.9 points on MOTA and 1.6 points on IDF1 respectively.

6. CONCLUSION AND DEVELOPMENT ORIENTATIONS

6.1 CONCLUSION

This program that we implemented using QDTrack, this is a promising approach for multiple object tracking in complex scenarios. The QDTrack method utilizes a novel quasi-dense similarity learning framework to effectively capture the similarity between object proposals, which allows for robust tracking even in challenging scenarios such as crowded scenes with occlusions and complex interactions between objects.

In comparison to other state-of-the-art methods, the QDTrack method demonstrates competitive performance on benchmark datasets, achieving high scores on evaluation metrics such as MOTA, IDF1, MOTP, MT and FN.

Furthermore, the QDTrack method is able to handle complex scenarios such as long-term occlusions and track fragmentation, which are common challenges in multiple object tracking.

However, it is worth noting that the QDTrack method may not be suitable for all tracking scenarios, as it relies on the availability of object proposals and assumes relatively stable object appearance and motion patterns. In scenarios where these assumptions are not valid, other tracking methods may be more appropriate.

6.2 DEVELOPMENT ORIENTATIONS

On the method Quasi-Dense Similarity Learning for Multiple Object Tracking **[12]**, the authors used ResNet-50 as the backbone by default. However, which is quite old and not as efficient as the current latest backbones. We propose using SwinTransformer **[19]** and Attention Deformable **[20]** instead for next improvement.

7. REFERENCES

[1] Borijan Georgievski (2020) Object Detection and Tracking in 2020

[2] Sanyam (2022) Understanding Multiple Object Tracking using DeepSORT

[3] Hexin Bai, Wensheng Cheng, Peng Chu, Juehuan Liu, Kai Zhang, Haibin Ling (2021) GMOT-40: A Benchmark for Generic Multiple Object Tracking

[4] Bui Tien Tung (2020) SORT - Deep SORT : Một góc nhìn về Object Tracking (phần 1)

[5] Peize Sun, Jinkun Cao, Yi Jiang, Zehuan Yuan, Song Bai, Kris Kitani, Ping Luo (2022) DanceTrack: Multi-Object Tracking in Uniform Appearance and Diverse Motion

[6] Libo Zhang, Junyuan Gao, Zhen Xiao, Heng Fan (2022) AnimalTrack: A Benchmark for Multi-Animal Tracking in the Wild

[7] Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple Online and Realtime Tracking with a Deep Association Metric. In 2016 IEEE International Conference on Image Processing (ICIP) (pp. 3464-3468). IEEE.

[8] Wan, Z., Wang, Q., & Yang, S. (2019) Simple Online And Realtime Tracking With A Deep Association Metric

[9] Mengmi Zhang, Claire Tseng, and Gabriel Kreiman, Joint Detection and Tracking by Semantic and Dynamic Contextual Modulation in arXiv:1911.07349v3 [cs.CV] 25 Mar 2020

[10] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, Christoph Feichtenhofer (2022) TrackFormer: Multi-Object Tracking with Transformers

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017) Attention is all you need. In Adv. Neural Inform. Process. Syst

[12] Jiangmiao Pang, Linlu Qiu, Xia Li, Haofeng Chen, Qi Li, Trevor Darrell, Fisher Yu (2021) Quasi-Dense Similarity Learning for Multiple Object Tracking

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks in Advances in Neural Information Processing Systems 28 (NIPS 2015)

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie, Feature Pyramid Networks for Object Detection [19 Apr 2017]

[15] Daniel Tomer, What I Was Missing While Using The Kalman Filter For Object Tracking [Jun 14, 2022] in towardsdatascienceric website

[16] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler, MOT16: A Benchmark for Multi-Object Tracking, arXiv:1603.00831v2 [cs.CV] 3 May 2016

[17] VisAI Labs, Evaluating multiple object tracking accuracy and performance metrics in a real-time setting [April 25, 2021]

[18] Jonathon Luiten, Aljosa O˘sep˘, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, Bastian Leibe, HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking in arXiv:2009.07736v2 [cs.CV] 29 Sep 2020.

[19] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo, Swin Transformer: Hierarchical Vision Transformer using Shifted Windows 2021

[20] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, Gao Huang, Vision Transformer with Deformable Attention, in arXiv:2201.00520v3 [cs.CV] 24 May 2022