M-JPDA: Multi-Object Tracking using Metric Learning and Joint Probabilistic Data Association

Farbod T. Motlagh

Supervisors: Dr. Hamid Rezatofighi and Prof. Ian Reid

Advanced Topics in Computer Science

June 2018

Contents

1 Introduction and Motivation

Object tracking is classified as a mid-level task in computer vision and is considered an essential component of high-level tasks such as pose estimation, action recognition, human computer interaction, and virtual/augmented reality [16]. These practical applications have motivated the research community to overcome the challenges associated with object tracking. In multi-target tracking the goal is to locate objects of interests in a video sequence and subsequently infer useful information such as trajectory and identity of the detected objects throughout the video. In addition to the problems that single-object tracking algorithms attempt to solve which revolve around finding robust appearance and motion models, MOT algorithms have two challenging tasks to address:

1. Initialization and termination of tracks, i.e. track management.

2. Maintaining the identity of the objects of interest throughout the video sequence which could be extremely difficult in real-world situations due to severe occlusion and abrupt appearance changes.

Figure 1: Example of multi-target tracking output from Rezatofighi et al..

Although multi-object tracking algorithms have progressed significantly in the past decade, there is still room for improvement. A recent review of state-of-the-art MOT (multi-object tracking) methods [14] observes that the majority of top performing methods are focused on building robust appearance-based similarity measurement systems, whereas methods with less satisfactory performance do not utilise appearance-based affinity models. Hence, the authors suggest that building strong affinity models should be the focus of future work in MOT. Additionally, [16] acknowledges video adaptation and adaptive appearance models as a future direction in MOT. This indicates that adaptability is another area of growth in MOT which worth exploring.

In this project, we explore the potential of adaptive deep-learned similarity metrics for improving the performance of multi-object tracking. Specifically, we aim to leverage recent advances in meta-learning and metric-learning to develop adaptive appearance similarity costs [7, 18] in order to distinguish between relevant and irrelevant observations with respect

to a specific target. We believe that adaptive meta-learned models can outperform metric-learning methods which act as static feature extractors. However, harnessing the power of meta-learning for multi-target tracking is a challenging task. In addition to this, we aim to develop a stand-alone multi-object tracking library in C++ which would enable straightforward and effortless experimentation with MOT algorithms. We acknowledge that developing a comprehensive multi-object tracking library is a difficult task, therefore our focus would be to implement a modularized version of only one well-established tracking-by-detection algorithm. Adding different MOT methods and algorithms to the library will be regarded as future work. We will be developing an end-to-end tracking-by-detection system using JPDA revisited algorithm [22]. This will be implemented in C++ as a decoupled module to be used in both practical applications and research experiments. We will be aiming for an optimised implementation of the algorithm.

In summary, the main goal of the project is to explore the effectiveness of meta-learning and metric-learning methods for improving the performance of a classical multi-target tracking algorithm i.e. joint probabilistic data association (JPDA). A secondary outcome of the project would be a C++ MOT library, MOTracker which serves as an end-to-end tracking application to be used for research and practical purposes. To test our proposed methods, we will be comparing our results to other state-of-the-art multi-object tracking algorithms on MOT challenge [17] datasets. The stretch goal is to be placed in top 10 on the MOT challenge leader-board. [1]

2 Related Work

There has been great advances in both multi-object tracking [14] and meta-learning [7, 18] in recent years, however, the effectiveness of meta-learning methods on MOT systems is yet to be explored. [5] proposes a novel method for real-time tracking of a single target using siamese matching networks and meta-learning. In [19], the authors utilise meta-learning for fast adaptation of visual features to achieve highly accurate tracking of a single object. Both methods show promising results and have improved the state-of-the-art which indicates that meta-learning methods have the potential to help solve other object tracking problems such as multi-object tracking.

On the other hand, applying metric-learning on tracking has been greatly explored in the past few years. [13] uses siamese matching networks to acquire high-level visual features

[1] https://motchallenge.net/results/MOT17/

for data association. By feeding the matching framework to a simple tracking method they have surpassed other complex methods which mostly utilize only motion information. [11] has introduced a new method for learning a metric space and utilizing the similarity measure for Bayesian-based tracking algorithm. Our method is somewhat similar to [11], however, we leverage recent advances in deep-learning to build a more accurate high-level similarity measure and we use JPDA for data association which is a more principled way of matching measurements to targets.

3 Background

3.1 Multiple Object Tracking

Multi-object tracking in the context of computer vision is the task of locating an unknown and unfixed number of objects of interest while maintaining their unique identities and finding their individual trajectories given an input video. An object of interest can be defined as any physical object that interacts with the outside world such as pedestrians on the street, soccer players on the field, a group of birds in the sky, and the like. In this work we focus on pedestrian tracking for 2 main reasons. 1) Most of the literature on multi-object tracking is focused on pedestrian tracking which makes fair comparison of methods in the research community straightforward. 2) Pedestrians are non-rigid objects with ever changing appearance which makes them an ideal object of interests for MOT.

MOT algorithms are usually categorized based on 2 main criteria [16].

- *Initialization method.* There are 2 types of MOT methods based on how their targets are initialized: 1) Detection based tracking, objects of interest are first detected then tracked. 2) Detection free, targets are initialized manually in the beginning of the video and then tracked.

- *Processing mode.* Similarly MOT algorithms can be categorized into two groups depending on how they process the input video: 1) On-line, the algorithm is only dependent on the current and previous frames. 2) Off-line, tracking algorithm consumes batches of video frames and outputs the tracking results which means it cannot be used in real-time scenarios.

In this work, we focus on on-line detection-based multi-target tracking. The problem can be formulated as a multi-variable estimation problem. Given a series of observations at time t, $\mathbf{O_t} = o_{1_t}, o_{2_t}, ...o_{i_t}$ and target states, $\mathbf{S_t} = s_{1_t}, s_{2_t}, ...s_{i_t}$, the goal is to find $P(\mathbf{S_{1:t}}|\mathbf{O_{1:t}})$.

3.2 Bayesian Filtering

A probabilistic formulation of $P(\mathbf{S_{1:t}}|\mathbf{O_{1:t}})$ can be derived by applying Bayesian estimation method [3]. The solution is often described as a recursive 2-step procedure:

$$\text{Predict the successor state:} P(\mathbf{S_t}|\mathbf{O_{1:t-1}}) = P(\mathbf{S_t}|\mathbf{S_{t-1}})P(\mathbf{S_{t-1}}|\mathbf{O_{1:t-1}}) \tag{1}$$

$$\text{Update using the new observation:} P(\mathbf{S_t}|\mathbf{O_{1:t}}) \propto P(\mathbf{O_t}|\mathbf{S_t})P(\mathbf{S_t}|\mathbf{O_{1:t-1}}) \tag{2}$$

$P(\mathbf{S_t}|\mathbf{S_{t-1}})$ is regarded as the *Dynamical or Process model* and $P(\mathbf{O_t}|\mathbf{S_t})$ is the *Observation or measurement model*. Finding the exact solution to Bayesian formulation is often impractical, therefore we seek an approximation of the solution or impose limiting assumptions to make it possible to the optimal solution. Kalman filter and particle filter algorithms are 2 well-known methods for finding an approximation to recursive Bayesian estimation problem.

Specifically, Kalman filter assumes that the process and measurement models can be described as Gaussian distributions, hence a complete solution can be found analytically ([3], Chapter 3). The general idea of Kalman is that if f $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$ is Gaussian $\mathcal{N}(\mathbf{x}_{k-1}, \mu_{k-1}, \Sigma_{k-1})$, it can be proved that $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ is also Gaussian, when certain assumptions hold ([3], Chapter 3). Here, we have x_t as the target state and z_t as a measurement at time t. (1) and (2) can be written as:

Prediction:

$$\mu_{k|k-1} = F_k\mu_{k-1|k-1} \tag{3}$$

$$\Sigma_{k|k-1} = F\Sigma_{k-1|k-1}F^T + Q_k \tag{4}$$

Update:

$$\mu_{k|k} = \mu_{k|k-1} + K_k(\mathbf{z}_k - H_k\mu_{k|k-1}) \tag{5}$$

$$\Sigma_k = (I - K_kH)\Sigma_{k|k-1} \tag{6}$$

$$K_k = \Sigma_{k|k-1}H^T(H\Sigma_{k|k-1}H^T + R)^{-1} \tag{7}$$

Where F and Q are process model's transition and covariance matrices respectively and

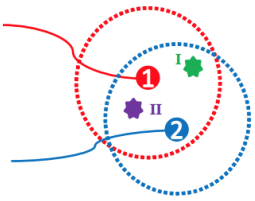
In most real-world problems, a single process model is unable to capture the dynamical behavior of a complex system, therefore different methods have been proposed to incorporate

multiple models to describe a system. The interacting multiple model algorithm is one of the well-known methods for fusing multiple motion models to better capture the dynamical behavior of a system [8]. To make the process of predicting and updating tracks more accurate we utilise the IMM method. In summary, IMM assigns a running probability to each dynamical model and fuses them during the update step.

A key step in the development of multi-target tracking systems is data association. Data association refers to the problem of assigning the observations or detections to the correct target. There are different methods for performing data association, in this project we utilise JPDA algorithm which offers a probabilistic solution to the problem of data association.

3.3 JPDA Revisited

JPDA is a powerful data association algorithm which incorporates the statistical properties of tracking error and clutter to associate detections to candidate targets. Unlike its predecessor, Probabilistic data association (PDA), JPDA does not make the assumption that each detection belongs to one target. The algorithm considers all the PDF (Probability Density Function) properties of detections in the order to associate weighted detections to each target ([3], Chapter 5). Figure 2 shows the combination of measurement to target associations.



Θ	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7
1	0	0	0	I	I	II	II
2	0	I	II	0	II	0	I
$p(\theta)$	$p_0^1 p_0^2$	$p_0^1 p_{\text{I}}^2$	$p_{\text{II}}^1 p_0^2$	$p_{\text{I}}^1 p_0^2$	$p_{\text{I}}^1 p_{\text{II}}^2$	$p_{\text{II}}^1 p_0^2$	$p_{\text{II}}^1 p_{\text{I}}^2$

Figure 2: Measurement to target association Rezatofighi et al..

JPDA solves the problem of data association by assigning a score to each target-measurement pair Eq 8.

$$q_j^i = \sum_{\theta \in \Theta_j^i} p(\theta) \tag{8}$$

When we only consider the dynamical properties of the targets $p(\theta_j^i)$ (probability of assigning target i to measurement j) is :

$$p(\theta_j^i) \propto \begin{cases} (1 - p_D)\beta & \text{if no detection} \\ p_D . \mathcal{N}(z_t^j, x_t^i, \Sigma_s) \end{cases} \tag{9}$$

Where p_D is the probability of a detection (measurement), β is the probability of detection clutter, and Σ_s is the innovation matrix from Kalman filter.

As the number of targets and measurements increases solving q_i^j becomes intractable (number of assignments grows exponentially $O(N!M!)$. Therefore, JPDA in its normal form has exponential time complexity which makes it impractical to use when there are a high number of targets. [22] proposes an efficient way to approximate JPDA's solution without hurting its performance significantly. This work aims to utilize [22] to produce an optimised on-line multi-target tracking library with high accuracy. In summary, [22] only takes into account the m highest probabilities ($p(\theta)$) and ignores the other ones.

JPDA uses a similarity cost (i.e. $p(\theta)$) to associate the observations to their corresponding targets. This similarity measurement can be of any nature e.g. dynamical properties or appearance cues. More relevant and accurate affinity cues yield more accurate tracking results. Appearance-based metrics often offer more relevant information about the target and the observation, however, it is difficult to harness this information. Metric learning is a well-known method to discriminate based on appearance cues between observations to yield better data association.

3.4 Metric Learning

Metric learning in the context of computer vision refers to building models that are able to discriminate between images based on their content. The goal in metric learning is to project the input data-points onto a space where semantically related data-points are close to each other whereas data-points with different semantic content are placed far away from one and other. In other words, it is an attempt to map images onto a space where similar images are close to each other and images with different content are far apart. A number of powerful deep-learning based methods have been introduced recently which offer great ways for building high-level visual metric spaces. Siamese networks [12], triplet loss [10], and prototypical networks [24] are among the superior methods for deep-metric learning.

In this work we build a metric space using a convolution neural network trained with the triplet loss function. Specifically, we utilise the method proposed by [9] for Person Re-Identification to improve data association by matching target and measurement features. Triplet loss proposed by [23] works by pulling semantically similar examples while pushing dissimilar examples away. To achieve this, in each training iteration 3 images are chosen: an anchor a, a positive example (anchor match) p, and a negative example (anchor mismatch) n (Figure 3). If the label for training example i is y_i, then $y_a = y_p \neq y_n$. The loss function

is thus as follows:

$$\mathcal{L}(\phi) = \sum_{a,p,n} [m + D(a,p) - D(a,n)] \tag{10}$$

Where ϕ is the network weights, D is the Euclidean distance between 2 feature embeddings and m the margin. [9] extends the triplet loss by focusing on hard negatives and hard positives, which helps the network to 'understand' the concept of semantic similarity more efficiently. The final similarity measure shows promising result for the task of Person Re-Identification i.e. matching the same person in different conditions.

Figure 3: Examples of anchors (blue) with positive (a,c) and negative (b,d) images

Although metric learning can be considered as powerful similarity metrics, however, it does not provide flexibility and does not adapt well when encountering new situations. In target tracking, adaptation is essential because the appearance of targets are ever changing due to illumination and pose variation. Therefore exploring more adaptive methods such as on-line and meta learning is worthwhile.

3.5 Meta Learning

In meta-learning, the goal is to train a model on a variety of tasks so that the final model is able to learn to perform well on unseen tasks quickly and efficiently. For example, in image classification, we aim to find models that can learn to classify images of unseen classes using a small number of training samples and iterations. This is also referred to as few-shot learning.

[7] proposes a novel meta-learning technique (MAML) which is model-agnostic and different to usual meta-learning methods in nature. The key idea underlying this method is to find an initial set of parameters using samples from the distribution of tasks such that

the model has maximal performance on a new task after the parameters have been updated through a few gradient steps computed with a small amount of data from that new task.

Unlike prior meta-learning methods that learn an update function or learning rule [1], MAML does not make any assumption about the network architecture and does not enforce any constraints, hence it can be applied to any type of deep neural networks. The method can also be used with a variety of loss functions and optimization methods. This opens up an opportunity to incorporate meta-learning in tracking algorithms without the need to use complicated neural network architectures and optimisation methods. Following MAML, [18] proposes a simple yet effective meta-learning algorithm called Reptile which is similar to MAML in nature.

Formally, MAML and Reptile aim to find an initial set of parameters θ so that for a random task t with the corresponding loss L_t, the model will yield low loss after a small number of updates k.

$$\arg\min_{\theta} \mathbb{E}_t[L_t(Update_t^k(\theta))]$$

$Update_t^k$ is the parameter update operation. In few-shot learning, $Update$ operation is SGD (or any other gradient-based optimizer). MAML [7] and Reptile [18] tackle the above-mentioned problem in the same manner, however, MAML requires training-test split and calculating second or higher order derivatives which makes it harder to implement. On the other hand, Reptile offers a surprisingly simple algorithm which is similar to joint-training but solves the meta-learning problem. By applying multiple gradient decent on a single task instead of one (when performing joint learning) . In our experiments we use Reptile for meta-learning as it is simple to implement, less computationally expensive to train and test, and has shown state-of-the-art performance on few-shot learning tasks.

Meta-learning procedure described in [18] has three main stages. We will be referring to these stages throughout the document:

1. Meta-training: in this stage an initialization is (meta-)learned given the training tasks.

2. Eval-training: the network is fine-tuned on a specific task (usually a task that was not seen in training) by applying a small number of updates for the task using a small number of training samples.

3. Eval-testing: finally, the network is used to predict new unseen sample for the fine-tuned task.

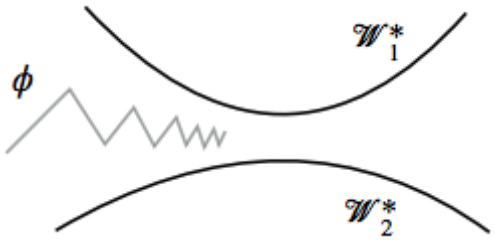


Figure 4: Visualization of meta-learning an initialization from Nichol et al.. If W_1 and W_2 are optimal parameters for solving task 1 and 2. Reptile attempts to find an initial set of parameters ϕ which lies in between the optimal W_1 and W_2

4 Methodology

Firstly, we put forward a method to incorporate visual features acquired from a deep-learned metric model into JPDA. It is expected that visual features would greatly improve the accuracy of data association step. Secondly, we explore meta-learning methods and propose some preliminary ideas for applying meta-learning to multi-target tracking.

4.1 Metric Learning and JPDA

We use a CNN trained with a variation of Triplet loss proposed in [9] to extract embedding features for each detection. We, then, use the embeddings for data association. Specifically, Eq. 9 is modified to incorporate appearance-based matching.

$$p(\theta_j^i) \propto p_{motion} \cdot p_{appearance} \tag{11}$$

We can substitute p_{motion} with Eq. 9 and $p_{appearance}$ to be the negative distance between the target and measurement embeddings.

$$p(\theta_j^i) \propto \begin{cases} (1 - p_D).\beta.\alpha. - D_0 & \text{if no detection} \\ p_D. \exp(\mathcal{N}(z_t^j, x_t^i, \Sigma_s)). \exp(-\alpha D(f_\phi(x_t^i), f_\phi(z_t^j))) \end{cases} \tag{12}$$

Where D is the Euclidean distance function, D_0 is the minimum threshold for a mismatch, α is the regularizer, and f_ϕ is the embedding function i.e. CNN with ϕ as its parameters.

Since appearance cues are more reliable than motion, assigning more weight to $p_{appearance}$ could yield better results.

4.2 Meta-Learning

Applying few-shot learning to multi-target tracking in its normal form for image classification can be challenging. In classification the number of classes (identities) are fixed and known, whereas in multi-target tracking the number of objects are unknown and unfixed. Therefore, a metric space to be used as visual similarity measure is deemed to be essential. We propose 2 methods for utilising meta-learning in data-association:

1. Using Reptile to meta-train for an embedding feature. MAML ([7]) claims that during the meta-training the network acquires a generalized representation for learning a task. Thus, we hypothesize that using the embedding layer of this generalized initialization should provide a reasonable embedding space for matching 2 semantically similar input data points. However, this hypothesis could be flawed (as the experiment shows) as the generalized representation is for the task of **learning** the task and not the task itself.

2. A better approach is to use meta-training step to acquire a parameter initialisation for quick and efficient learning of new metric space. In other words, we use meta-learning for quick metric-learning. To test this method, we use prototypical networks loss and meta-train with reptile. This would provide us a with an initialization that can learn a metric space quickly for any n number of classes.

5 Experiments and Results

5.1 MOT Challenge

MOT Challenge [17] is one of the well-established MOT benchmarks in the field of tracking. The benchmark has been greatly explored and used in the research community. In MOT challenge, the task is to track pedestrians in crowded scenes with severe occlusion and abrupt appearance changes. There are multiple video sequences each with different characteristic. The dataset for MOT16 challenge is divided to train and test sets. The training set includes 7 videos with a total of 5,316 frames, 517 tracks (identities) and 110,407 ground truth boxes. Similarly, the test has 7 videos with a total of 5,919 frames, 759 tracks, and 182,326 bounding boxes. MOT17 has the same train and test set, however, the ground truth boxes are calibrated and also 3 different detections are provided.

MOT challenge uses CLEAR MOT metrics proposed by Bernardin and Stiefelhagen in 2008 [2], the final score is MOTA which is a combination of a number of metrics:

- **FP**: False positives is the total number of occurrences where the track indicates the existence of an object but there are no object in the frame.
- **FN**: False negative is when an object in a frame has not been detected.
- **ID Switch (IDs)**: Identity switches is the number of times an object is assigned a new ID in its track.
- **Multiple object tracking accuracy (MOTA)**: A combination of FP, FN, and IDs.

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDs_t)}{\sum_t G_t}$$

  Where t and G_t are the frame number and number of objects present at time t respectively.

- **Multiple object tracking precision (MOTP)**: Measures the alignment of the predicted tracks and the ground truth. Heavily reliant on the detection quality and not the tracking performance.

$$MOTP = 1 - \frac{\sum_t (d_{t,i})}{\sum_t c_t}$$

  Where t and G_t are the frame number and number of objects present at time t respectively.

- **False alarm per frame (FAF)**: Average of false positive per frame.
- **Mostly tracked targets (MT)**: When a ground-truth target is assigned the same identity for at least 80% of its lifetime.
- **Mostly lost target (ML)**: When a ground-truth target is assigned the same identity for at most 20% of its lifetime.
- **Fragmentation (Frag)**: the number of times an object is lost in a frame but then re-detected later in the video, thus fragmenting the track.
- **Hz**: The speed of a tracker (frames per second).

We, first, evaluated our method on the training set of MOT17 and submitted our tracking results for the test set of MOT16 (since MOT16 allows for private detectors).

5.1.1 Detector

Detection is a crucial component in MOT algorithm. Inaccurate and noisy measurement can greatly change the accuracy of the tracker. To prove this, we run our experiments with 2 different detectors. The first detector is an off-the-shelf YOLOv3 [21] trained on MSCOCO dataset [15].

We trained a specialized pedestrian detector (1 class) using YOLOv3 and 3 datasets: MOT17Det training set [17], Pascal VOC challenge [6] (person class), and MSCOCO [15] (person class). Table 1 shows the evaluation results on the training set of MOT17Det for the off-the-shelf YOLOv3 detector and pedestrian detector (YOLOv3-MOT) trained for our tracker, it is clear that the specialized detector outperforms the off-the-shelf detector. Table 2 shows the results of evaluation on the test data set MOT17Det. It can be seen that our custom YOLOv3 does better than MOT17 FRCNN and DPM, but is less accurate than SDP trained. The evaluation result for YOLOv3 is available at [2].

Detector	AP ↑	FAR ↓	GT	TP ↑	FP↓	FN ↓	Recall	Precision
YOLOv3-COCO	0.61	1.75	66,393	44,834	**9,318**	21,559	67.5	82.8
YOLOv3-MOT	**0.77**	**2.84**	66,393	**58,077**	15,079	**8,316**	**87.5**	**79.4**

Table 1: Comparing YOLOv3 detectors on MOT17Det, train dataset. AP=Average precision

Detector	AP ↑	FAR ↓	TP ↑	FP↓	FN ↓	Recall	Precision
MOT17-SDP	**0.81**	**1.3**	**95,699**	**7,599**	**18,865**	**83.5**	**92.6**
YOLOv3-MOT	0.77	3.7	94,146	22,047	20,418	82.2	81.0
MOT17-FRCNN	0.72	1.7	88,601	10,81	25963	77.3	89.8
MOT17-DPM	0.61	7.1	78,007	42,308	36,557	68.1	64.8

Table 2: Comparing YOLOv3 with MOT17 default detectors on the test set.

Figure 5 also shows precision over recall plot for YOLOv3 COCO and MOT. Generating a figure for test was not possible as the evaluation code is hidden and only accessible through MOT server.

[2] https://motchallenge.net/detector/yolov3

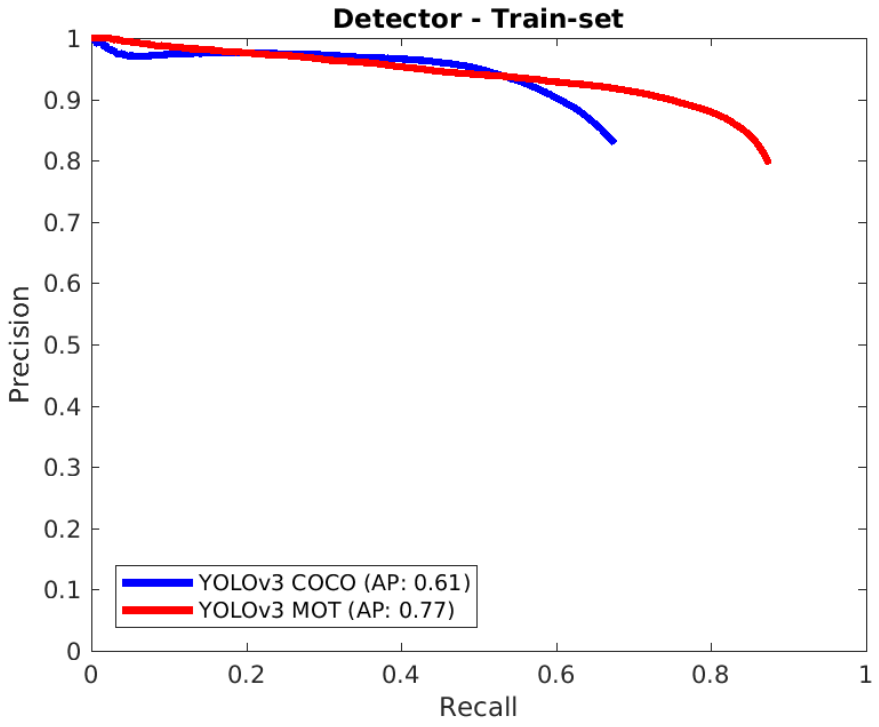


Figure 5: Recall Precision plot for YOLOv3 on MOT17Det train set.

5.1.2 With/Without Appearance

We evaluated our tracker with multiple detectors and data association methods to investigate the effectiveness of the proposed metric learning method for JPDA. Four different experiments were performed:

- Ex1. Detector: YOLOv3 - COCO (off-the-shelf), JPDA: Only motion model (M).
- Ex2. Detector: YOLOv3 - COCO (off-the-shelf), JPDA: Motion and appearance models (MA).
- Ex3. Detector: YOLOv3 - MOT, JPDA: Only motion model (M).
- Ex4. Detector: YOLOv3 - MOT, JPDA: Motion and appearance models (MA).

Table 3 shows MOT evaluation for the above-mentioned experiments. It is clear that using appearance cues improve the performance of the tracker significantly. The detector also plays an important role, as it was mentioned earlier.

The best tracker parameters were searched for individually (M and MA) and the highest MOTA was scored for each. Figure 6 shows examples of tracking visualization.

	MOTA	MOTP	MOTAL	Recall	Prcn	FAR	GT	MT	PT	ML	FP	FN	IDs	Frag
Ex1. M	30.0	74.2	31.2	46.9	74.9	3.29	530	75	250	205	17,477	59,238	1,344	1,863
Ex2. MA	34.6	74.9	35.4	45.3	82.1	2.07	530	59	234	237	11,002	61,024	847	1,374
Ex3. M	36.0	76.0	38.4	**60.3**	73.3	4.61	530	105	293	132	24,500	44,215	2,629	2,974
Ex4. MA	**44.1**	**76.6**	**45.4**	58.9	**81.4**	2.83	530	99	262	169	15,025	45,852	1,411	2,047

Table 3: Comparing YOLOv3 with MOT17 default detectors on the test set.

Figure 6: Tracker visualization example.

5.1.3 Test Dataset MOT16

We also submitted the fourth set-up (YOLOv3-MOT with motion and appearance model) to MOT16 evaluation server. Table 4 shows the results of our tracker compared with 3 other trackers. Although the performance of our tracker is acceptable, comparing to other private trackers in MOT16 leader-board it yields a poor results. This could be due to multiple reasons: 1- Our algorithm is near-online and real-time. 2- Currently, our method uses a heuristic track management. A principled way for track management is expected to yield better results. 3- A more extensive hyper parameter search is deemed to be essential for state-of-the-art results.

Our tracker comes 52nd out of 83rd trackers in MOT16 Leader board.

	MOTA	MT	ML	FP	FN	IDs	Frag
HT_SJTUZTE (No. 1)	71.3	46.5	19.5	9,238	42,521	617	743
NOSVM	45.4	15.7	42.7	7,091	91,941	608	1,666
JPDA_MA (52nd)	43.3	15.4	37.4	24,241	77,291	1,759	2,387
DeepS	43.0	15.3	41.8	9,808	93,287	876	865

Table 4: Comparing JPDA_MA (our tracker) with other trackers in MOT16 leader board.

5.2 Few-shot learning on Mini-Imagenet

To better understand meta-learning methods and for future experimentation, Reptile was implemented [3] using PyTorch in Python. To test the implementation, the Mini-Imagenet experiment from [18] was replicated using the exact same experiment set-up (hyper-parameters and dataset). Mini-Imagenet is a subset of Imagenet dataset (ILSVRC2012) proposed by [20] for testing the performance of few-shot learning methods. The dataset consists of 64 image classes for training, 12 classes for validation, and 24 classes for testing. Our implementation achieved $\approx \mathbf{66}\%$ accuracy on the testing dataset for 5-way 5-shot image classification which is close to the reported results in [18].

Prototypical networks [24] has also shown promising results for few-shot learning tasks. The method falls into the metric-learning category as it learns a similarity metric for classifying images using the training data. During the test time, a prototype is created for each class by taking the mean of embeddings for all the eval-training samples (support set). To classify an image, we simply find the distances between the prototypes and the embedding for a given test image and return the class of the prototype with the smallest distance. This simple method yields state-of-the-art results on few-shot learning only by learning a metric space from training data (no training during the test time). The reported accuracy for 5-way 5-shot image classification on Mini-Imagenet is $\approx \mathbf{68}\%$

We also tried combining Reptile and Prototypical networks as an attempt to build an adaptive metric space i.e. meta-metric-learning. The idea of combining meta-learning with metric-learning has been explored before using other meta-learning and metric-learning methods [4]. We were able to achieve $\approx \mathbf{60}\%$ for 5-way 5-shot image classification on Mini-Imagenet without any hyper-parameter optimisation. This method worth exploring further, however, there is one limitation that must be considered. During eval-training, since we are fine-tuning our metric space, at least 2 training samples are needed per class, One acts as a prototype and the other would serve as training data (to calculate loss). This implies that one-shot learning is impossible with the meta-metric method, however, the method might be suitable for tracking as there are more than one training samples available in Eval-training stage.

Going forward, building adaptive metric spaces using meta-learning seems to be a reasonable method for multi-target tracking. Extensive experiments and redefinition of the method is required to get sensible results

[3] https://github.com/farbodtm/reptile-pytorch

6 MOTracker Library

As part of this project we have developed a C++ tracking library for on-line and off-line video processing. In this section we present a summary of the design. The goal was to implement a modularized and optimised version of [22] to be used for research experiments and practical applications. However, other tracking methods and algorithms can be implemented and added as modules to the library in future. Our tracking methodology is classified as an online probabilistic detection-based tracking. We will be utlising IMM, JPDA, metric and meta learning methods.

6.1 Design

The tracking procedure returns the state of targets in a video feed by applying the detection and tracking algorithms on video frames. JPDA is implemented in a recursive and on-line manner which means it does not require to process all the frames at once to return the result of tracking. IMM, interactive multiple model algorithm is used for state estimation. The algorithm uses multiple dynamic models to accurately estimate the dynamical properties of targets at each given time. At each given frame, the targets and their current state can be requested. A high-level pseudo-code of the overall procedure is as follows:

```
while HasNextFrame():
        targets = []
        frame = GetNextFrame()
        detections = GetDetections(frame)

        foreach target in targets:
                target.predict()  # IMM prediction
                target_measurement_cost.append(GetCosts(target,
                                                        measurements))

        target_measurement_prob = JPDAF.apply(target_measurement_cost)

        foreach target in targets:
                target.update(target_measurement_prob)
```

OpenCV is used for frame matrices and *Blaze* library was used for performing linear algebra and storing matrices.

Proposed library is consists of following components:

1. Video feed: the main feed which supplies each frame for tracking, this could be coming from a camera feed or a video file. (abstract)
2. Detector: detects objects to be tracked in each frame. This can happen in each frame or every few frames for fast performance. (abstract)
3. Target: each detected object is turned to a target via track management. Targets are the main entities in the library describing the motion and appearance of each tracked object.
4. Tracker: the main component of the library which holds the targets and applies IMM, JPDA, and meta learning at each frame. The tracker is the detailed later in the section.
5. Visualizer: Mainly for debugging purposes, draws the targets, detections, and tracks on the frame.

6.1.1 Video Feed

This is an abstract class which the tracker uses to request for video frames from a source. The source must provide a `NextFrame()` method which returns OpenCV mat. `VideoFeed` and `CameraFeed` are implemented which read frames from a video file and camera respectively.

6.1.2 Detector

Abstract class to return detection results from a detector. This can be any detector as long as it implements `GetDetections()` method which should accept an OpenCV matrix (from the feed) and returns a vector of `Detections`.

6.1.3 Target

Represents a target in the scene and encapsulate the properties of a target. Includes a history of states from the birth of the target to its death. When a target is terminated it does not get deleted it only gets ignored.

6.1.4 Tracker

The main tracker object which carries out the heavy work. Tracker processes each frame by applying the detector and performing IMM and JPDA. Track managements happens after estimation where targets with no detections associated with them for a certain period of time would be terminated and detections without targets become new targets.

Track parameters are encapsulated by `TrackerParams` struct where all the dynamical models, IMM and JPDA parameters and etc. is defined.

6.1.5 Visualizer

Given a tracker and a series of options visualizes the tracking results on the frame. Colors and style of tracking visualization would be flexible.

7 Conclusion

In this project we explored the effectiveness of using appearance cues in data association using JPDA and Meta-Learning methods. As it was anticipated, visual features improve the performance of our tracker significantly. In addition to this, we also briefly investigated the potential application of meta-learning in multi-target tracking. Although the preliminary results of our meta-metric-learning method looks promising, extensive experimentation is required.

In addition to the experiments, we developed a C++ tracking library for future research and practical applications. The library is in beta version at the moment, however, we are planning to continue the development for a production-ready version.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. *CoRR*, abs/1606.04474, 2016. URL http://arxiv.org/abs/1606.04474.

[2] Keni Bernardin and Rainer Stiefelhagen. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309, December 2008. ISSN 1687-5281. doi: 10.1155/2008/246309. URL https://link.springer.com/article/10.1155/2008/246309.

[3] S. S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. D. Artech House, 1999.

[4] Yu Cheng, Mo Yu, Xiaoxiao Guo, and Bowen Zhou. Few-Shot Learning with Meta Metric Learners. page 5.

[5] Janghoon Choi, Junseok Kwon, and Kyoung Mu Lee. Deep meta learning for real-time visual tracking based on target-specific feature space. *CoRR*, abs/1712.09153, 2017. URL http://arxiv.org/abs/1712.09153.

[6] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The <Emphasis Type="SmallCaps">Pascal</Emphasis> Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. ISSN 0920-5691, 1573-1405. doi: 10.1007/s11263-009-0275-4. URL https://link.springer.com/article/10.1007/s11263-009-0275-4.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

[8] Anthony F Genovese. The Interacting Multiple Model Algorithm for Accurate State Estimation of Maneuvering Targets. *Johns Hopkins APL Technical Digest*, 22(4):10, 2001.

[9] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In Defense of the Triplet Loss for Person Re-Identification. *arXiv:1703.07737 [cs]*, March 2017. URL http://arxiv.org/abs/1703.07737. arXiv: 1703.07737.

[10] Elad Hoffer and Nir Ailon. Deep metric learning using Triplet network. *arXiv:1412.6622 [cs, stat]*, December 2014. URL http://arxiv.org/abs/1412.6622. arXiv: 1412.6622.

[11] J. Hu, J. Lu, and Y. P. Tan. Deep Metric Learning for Visual Tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(11):2056–2068, November 2016. ISSN 1051-8215. doi: 10.1109/TCSVT.2015.2477936.

[12] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese Neural Networks for One-shot Image Recognition. page 8.

[13] Laura Leal-Taixe, Cristian Canton-Ferrer, and Konrad Schindler. Learning by Tracking: Siamese CNN for Robust Target Association. pages 418–425. IEEE, June 2016. ISBN 978-1-5090-1437-8. doi: 10.1109/CVPRW.2016.59. URL http://ieeexplore.ieee.org/document/7789549/.

[14] Laura Leal-Taixé, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, and Stefan Roth. Tracking the Trackers: An Analysis of the State of the Art in Multiple Object Tracking. *arXiv:1704.02781 [cs]*, April 2017. URL http://arxiv.org/abs/1704.02781. arXiv: 1704.02781.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv:1405.0312 [cs]*, May 2014. URL http://arxiv.org/abs/1405.0312. arXiv: 1405.0312.

[16] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, Xiaowei Zhao, and Tae-Kyun Kim. Multiple Object Tracking: A Literature Review. September 2014. URL http://arxiv.org/abs/1409.7618.

[17] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, March 2016. URL http://arxiv.org/abs/1603.00831. arXiv: 1603.00831.

[18] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. 2018. URL https://arxiv.org/abs/1803.02999.

[19] Eunbyung Park and Alexander C. Berg. Meta-tracker: Fast and robust online adaptation for visual object trackers. *CoRR*, abs/1801.03049, 2018. URL http://arxiv.org/abs/1801.03049.

[20] Sachin Ravi and Hugo Larochelle. Optimization as a Model for Few-Shot Learning. November 2016. URL https://openreview.net/forum?id=rJY0-Kcll.

[21] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv:1804.02767 [cs]*, April 2018. URL http://arxiv.org/abs/1804.02767. arXiv: 1804.02767.

[22] S. Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint probabilistic data association revisited. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 3047–3055, Washington, DC, USA, 2015. IEEE Computer Society. ISBN 978-1-4673-8391-2. doi: 10.1109/ICCV.2015.349. URL http://dx.doi.org/10.1109/ICCV.2015.349.

[23] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, June 2015. doi: 10.1109/CVPR.2015.7298682.

[24] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical Networks for Few-shot Learning. March 2017. URL https://arxiv.org/abs/1703.05175.