Joint Probabilistic Matching Using m-Best Solutions

Supplemental Material

Seyed Hamid Rezatofighi[1] Anton Milan[1] Zhen Zhang[2] Qinfeng Shi[1] Anthony Dick[1] Ian Reid[1]
[1]School of Computer Science, The University of Adelaide, Australia
[2]School of Computer Science and Technology, Northwestern Polytechnical University, Xian, China
hamid.rezatofighi@adelaide.edu.au

1. Content

The supplementary material contains multiple items that complete the main submission. First, we provide more details on the Binary Tree Partition algorithm. We also show that the approximation of the joint space using m-best solutions has an upper error bound, which is exponentially decreasing by incorporating more solutions. Next, we explain in more detail how the assignment problem from Sec. 4.1 is reformulated as an integer linear program. We then provide a detailed description of the experiments of the sequential re-id task from Section 4.2. These results are also accompanied by a video. Finally, a more in-depth analysis on feature point matching results provides a better intuition on the quality and on the diversity of the m-best solutions for the given task.

2. Binary Tree Partition Algorithm

Assuming finding optimal solution X^* of the following MAP problem (a one-to-one matching problem with a general objective function $f(X)$) is possible, Binary Tree Partition (BTP) algorithm is a computationally efficient approach to calculate the next $m-1$ optimal (best) solutions. Let us consider the optimization problem

$$X^* = \underset{X \in \mathcal{X}}{\operatorname{argmin}} \quad f(X), \tag{S.1}$$

where the $\mathcal{X}$ is the set of all possible solutions of the entire matching problem as in (S.17). To illustrate the BTP method, we define the feasible sets

$$\begin{aligned} \mathcal{X}_1 &= \mathcal{X}, \\ \mathcal{X}_m &= \{\mathcal{X} \,|\forall k \in \{1, 2, \dots, m-1\}, \langle X, X_k^* \rangle < \|X_k^*\|_1 \}, \end{aligned} \tag{S.2}$$

where $\langle , \rangle$ denotes the inner product of two vectors. The constraint $\langle X, X_k^* \rangle < \|X_k^*\|_1$ essentially says that $X \neq X_k^*$. This is true because X and X_k^* are binary vectors.[1]

For arbitrary $k < m$, the following holds:

$$\begin{aligned} X_k^* &= \underset{X \in \mathcal{X}_k}{\operatorname{argmin}} f(X), \\ \mathcal{X}_{k+1} &= \mathcal{X}_k \cap \{X | \langle X_k^*, X \rangle < \|X_k^*\|_1\}. \end{aligned} \tag{S.3}$$

Assume that X_1^* and X_2^* are already known. To obtain the third-best solution, we proceed by picking an arbitrary dimension j where the first two solutions differ, *i.e.* $j \in \{i | (X_1^*)_i \neq (X_2^*)_i\}$, and split the feasible set

$$\mathcal{X}_3 = \left\{ X \in \mathcal{X} \;\middle|\; \begin{array}{l} \langle X, X_1^* \rangle < \|X_1^*\|_1, \\ \langle X, X_2^* \rangle < \|X_2^*\|_1 \end{array} \right\} \tag{S.4}$$

[1]It is also equivalent to use the equality and inequality constraint $\langle X, X_k^* \rangle \leqslant \|X_k^*\|_1 - 1$.

into two disjoint sets by assigning one of the two possible values (0 and 1) to the picked dimension j as follows:

$$\mathcal{X}_3^1=\left\{X\in\mathcal{X}\;\middle|\;\underbrace{(X)_j=(X_1^*)_j}_{\text{Assignment}},\underbrace{\begin{matrix}\langle X,X_1^*\rangle<\|X_1^*\|_1,\\ \langle X,X_2^*\rangle<\|X_2^*\|_1,\end{matrix}}_{\text{Redundant}}\right\},$$
$$\mathcal{X}_3^2=\left\{X\in\mathcal{X}\;\middle|\;\underbrace{(X)_j=(X_2^*)_j}_{\text{Assignment}},\underbrace{\begin{matrix}\langle X,X_2^*\rangle<\|X_2^*\|_1,\\ \langle X,X_1^*\rangle<\|X_1^*\|_1,\end{matrix}}_{\text{Redundant}}\right\}. \tag{S.5}$$

Note that $(X_1^*)_j\neq(X_2^*)_j$, we have that $(X)_j=(X_1^*)_j$ enforces $\langle X,X_2^*\rangle<\|X_2^*\|_1$, and $(X)_j=(X_2^*)_j$ enforces $\langle X,X_1^*\rangle<\|X_1^*\|_1$. Moreover, by the fact that $(X)_j\in\{0,1\}$ we must have that $\mathcal{X}_3^1\cup\mathcal{X}_3^2=\mathcal{X}_3$. Furthermore, the constraints $(X)_j=(X_1^*)_j$ and $(X)_j=(X_2^*)_j$ are in fact fixed assignments to that variable, which reduces the total number of unknowns.

Since we are dealing with a binary problem, *i.e.* $(X)_j\in\{0,1\}$, it follows that $\mathcal{X}_3^1\cap\mathcal{X}_3^2=\emptyset$ and $\mathcal{X}_3^1\cup\mathcal{X}_3^2=\mathcal{X}_3$. Thus let

$$X_3^1=\underset{X\in\mathcal{X}_3^1}{\operatorname{argmin}}\,f(X), \tag{S.6a}$$
$$X_3^2=\underset{X\in\mathcal{X}_3^2}{\operatorname{argmin}}\,f(X), \tag{S.6b}$$

and let $l_3=\operatorname{argmin}_{l\in\{1,2\}}X_3^l$. Then, X_3^* can be obtained via

$$X_3^*=X_3^{l_3}. \tag{S.7}$$

Now we are going to find the fourth best solution. We show that $\mathcal{X}_4$ can be obtained by splitting $\mathcal{X}_3^{l_3}$ into two parts, and copying the other $\mathcal{X}_3^l$. The process is illustrated in Figure 5.

For those $l\in\{1,2\},l\neq l_3$, we must have that $X_3^*\notin\mathcal{X}_3^l$, which means that the constraint $\langle X_3^*,X\rangle<\|X_3^*\|_1$ is redundant for them. As a result, we have $\forall l=1,2,l\neq l_3$

$$\mathcal{X}_4^l=\mathcal{X}_3^l\cap\{X|\langle X_3^*,X\rangle<\|X_3^*\|_1\}. \tag{S.8}$$

For $\mathcal{X}_3^{l_3}$, after adding a new constraint $\langle X_3^*,X\rangle<\|X_3^*\|_1$ we obtain the set

$$\mathcal{X}_3^{l_3}\cap\{X\in\mathcal{X}|\langle X_3^*,X\rangle<\|X_3^*\|_1\}.$$

We pick an arbitrary $j'\in\{i|(X_3^*)_i\neq(X_{l_3}^*)_i\}$, after which $\mathcal{X}_3^{l_3}\cap\{X\in\mathcal{X}|\langle X_3^*,X\rangle<\|X_3^*\|_1\}$ can be partitioned into two disjoint parts as in (S.9).

$$\mathcal{X}_4^{l_3}=\left\{X\in\mathcal{X}\;\middle|\;\begin{matrix}\langle X_{l_3}^*,X\rangle<\|X_{l_3}^*\|_1,\overbrace{\langle X_3^*,X\rangle<\|X_3^*\|_1}^{\text{Redundant}},\\ (X)_j=(X_{l_3}^*)_j,(X)_{j'}=(X_{l_3}^*)_{j'}\end{matrix}\right\}, \tag{S.9a}$$
$$\mathcal{X}_4^3=\left\{X\in\mathcal{X}\;\middle|\;\begin{matrix}\overbrace{\langle X_{l_3}^*,X\rangle<\|X_{l_3}^*\|_1}^{\text{Redundant}},\langle X_3^*,X\rangle<\|X_3^*\|_1,\\ (X)_j=(X_{l_3}^*)_j,(X)_{j'}=(X_3^*)_{j'}\end{matrix}\right\}, \tag{S.9b}$$

Similar to (S.5), there are redundant constraints in (S.9) and they can be safely removed without altering the problem. $\mathcal{X}_4$ is then divided into three disjoint parts. Subsequently, the optimal solution from each $\mathcal{X}_4^l,l=1,2,3$ can be obtained via

$$X_4^l=\underset{X\in\mathcal{X}_4^l}{\operatorname{argmin}}\,f(X),\text{ for }l\in\{l_3,3\}, \tag{S.10a}$$
$$X_4^l=X_3^l,\qquad\qquad\text{otherwise} \tag{S.10b}$$

and X_4^* can be obtained via

$$X_4^*=\min_{l\in\{1,2,3\}}X_4^l. \tag{S.11}$$

Algorithm 1: Binary Tree Partition Algorithm

input : $f(X), m$

output: $X_k^*, k = 1, \dots m$

$X_1^* = \underset{X \in \mathcal{X}}{\text{argmin}}\, f(X)$;

$X_2^* = \underset{X \in \mathcal{X}}{\text{argmin}}\, f(X)$ s.t. $\langle X_1^*, X \rangle < \|X_1^*\|_1$;

Select arbitrary $j \in \{i | (X_1^*)_i \neq (X_2^*)_i\}$;

$\mathcal{X}_3^1 = \{X \in \mathcal{X} | \langle X, X_1^* \rangle < \|X_1^*\|_1, (X)_j = (X_1^*)_j\}$;

$\mathcal{X}_3^2 = \{X \in \mathcal{X} | \langle X, X_2^* \rangle < \|X_2^*\|_1, (X)_j = (X_2^*)_j\}$;

$X_3^1 = \text{argmin}_{X \in \mathcal{X}_3^1} f(X)$;

$X_3^2 = \text{argmin}_{X \in \mathcal{X}_3^1} f(X)$;

for $k = 3, \dots, m$ **do**

  $l_k = \text{argmin}_l f(X_k^l), \quad X_k^* = X_k^{l_k}$;

  $\mathcal{X}_{k+1}^l = \mathcal{X}_k^l, \forall l < k, l \neq l_k$;

  Select arbitrary $j_k \in \{i | (X_{l_k}^*)_i \neq (X_k^*)_i\}$;

  $\mathcal{X}_{k+1}^{l_k} = \mathcal{X}_k^{l_k} \cap \{X | \langle X_k^*, X \rangle < \|X_k^*\|_1, X_{j_k} = (X_{l_k}^*)_{j_k}\}$;

  $\mathcal{X}_{k+1}^k = \mathcal{X}_k^k \cap \{X | \langle X_k^*, X \rangle < \|X_k^*\|_1, X_{j_k} = (X_k^*)_{j_k}\}$;

  Remove constraints $\langle X_k^*, X \rangle < \|X_k^*\|_1$ from $\mathcal{X}_{k+1}^{l_k}$;

  Remove constraints $\langle X_{l_k}^*, X \rangle < \|\langle X_{l_k}^* \|_1$ from $\mathcal{X}_{k+1}^k$;

  $X_{k+1}^l = \text{argmin}_{X \in \mathcal{X}_{k+1}^l} f(X), l \in \{l_k, k\}$;

This procedure is repeated until we find the m-best solutions. The algorithm is summarized in Algorithm 1. In Figure 5 an example run of Algorithm 1 is given. In each iteration of the algorithm, we select the optimal node, and then split the node into two sub-nodes. All the nodes construct a binary tree, so we call it a Binary Tree Partition algorithm. The exactness of Algorithm 1 is shown in the following proposition.

Proposition 1. *For arbitrary input $f(X)$ and m, if the set $\{X \in \mathcal{X}\}$ has at least m elements, then Algorithm 1 outputs m-best solutions defined in* (S.3).

Proof. Obviously, X_1^*, X_2^*, X_3^* and X_4^* obtained from Algorithm 1 are correct. Moreover, $\mathcal{X}_3^l, l \in \{1, 2\}$ can be formulated as

$$\mathcal{X}_3^l = \{X \in \mathcal{X} | \langle X, X_l^* \rangle < \|X_l^*\|_1, X_{L_3^l} = \mathbf{a}_3^l\},$$

where $L_3^l \subseteq \{1, 2, 3, \dots, M \times (N+1)\}$, $\mathbf{a}_3^l \in \{0, 1\}^{|M \times (N+1)|}$ is some deterministic assignment, and $X_{L_3^l} = [(X)_i]_{i \in L_3^l}$.

For some $k >= 4, k <= m-1$, we assume that

1. X_k^* from Algorithm 1 is correct;

2. $\forall l \in \{1, 2, \dots, k-1\}$, $\mathcal{X}_k^l$ can be formulated as

$$\mathcal{X}_k^l = \{X \in \mathcal{X} | \langle X, X_l^* \rangle < \|X_l^*\|_1, X_{L_k^l} = \mathbf{a}_k^l\}.$$

3. $\cup_{l=1}^{k-1} \mathcal{X}_k^l = \mathcal{X}_k$

We let $X_k^l = \text{argmin}_{X \in \mathcal{X}_k^l} f(X)$ and $l_k = \text{argmin}_l f(X_k^l)$. Again, we have the kth best solution $X_k^* = X_k^{l_k}$. For those $l \in \{1, 2, \dots, k-1\}, l \neq l_k$, we let

$$\mathcal{X}_{k+1}^l = \mathcal{X}_k^l \cap \{X \in \mathcal{X} | \langle X, X_k^* \rangle < \|X_k^*\|\} = \mathcal{X}_k^l. \tag{S.12}$$

For l_k, by picking an arbitrary $j_k \in \{j \in \{1,2,\ldots,n\} | (X^*_{l_k})_j \neq (X^*_k)_j\}$, we split $\mathcal{X}^{l_k}_k \cap \{X \in \mathcal{X} | \langle X, X^*_k \rangle < \|X^*_k\|\}$ into two disjoint sets as follows:

$$\mathcal{X}^{l_k}_{k+1} = \left\{ X \in \mathcal{X} \;\middle|\; \begin{array}{l} \langle X^*_{l_k}, X \rangle < \|X^*_{l_k}\|_1, \overbrace{\langle X^*_k, X \rangle < \|X^*_k\|_1}^{\text{Redundant}}, \\ X_{L^{l_k}_3} = \mathbf{a}^{l_k}_3, X_{j_k} = (X^*_{l_k})_{j_k} \end{array} \right\}, \tag{S.13a}$$

$$\mathcal{X}^{k}_{k+1} = \left\{ X \in \mathcal{X} \;\middle|\; \begin{array}{l} \overbrace{\langle X^*_{l_k}, X \rangle < \|X^*_{l_k}\|_1}^{\text{Redundant}}, \langle X^*_k, X \rangle < \|X^*_k\|_1, \\ X_{L^{l_k}_3} = \mathbf{a}^{l_k}_3, X_{j_k} = (X^*_k)_{j_k} \end{array} \right\}. \tag{S.13b}$$

Then Algorithm 1 will return:

$$\begin{aligned} X^l_{k+1} &= X^l_k, \forall l < k, l \neq l_k, \\ X^l_{k+1} &= \underset{X \in \mathcal{X}^l_{k+1}}{\text{argmin}}\ f(X), l \in \{l_k, k\}. \end{aligned} \tag{S.14}$$

Then it is easy to verify that $X_{k+1} = \text{argmin}_{X \in \{X^l_{k+1} | l \in \{1,2,\ldots,k\}\}} f(X)$ is correct, and $\mathcal{X}^l_{k+1}$ satisfies the same condition as $\mathcal{X}^l_k$. Thus, by mathematical induction the proposition holds. □

3. Upper Bound Error for The Approximation Using m-Best Solutions

We approximate the joint space for marginalization by m solutions. If the m estimated solutions (sorted or unsorted) are the m optimal solutions, we can show that the upper bound error $\mathcal{E}$ for the approximation is exponentially decreasing with respect to m for any arbitrary objective function $f(\cdot)$.

Assuming n_h is total number of solutions (permutations) in the one-to-one matching space $\mathcal{X}$, *i.e.* $n_h = |\mathcal{X}|$, and $\tilde{\mathcal{X}}^m$ is the space approximated by m-best solutions, *i.e.* $|\tilde{\mathcal{X}}^m| = m$, the approximation error has an upper bound value $\mathcal{E}$ as

$$\begin{aligned} \mathcal{E} &= \sum_{X \in \mathcal{X}} \exp\left(-f(X)\right) - \sum_{X \in \tilde{\mathcal{X}}^m} \exp\left(-f(X)\right) \\ &= \sum_{k=1}^{n_h} \exp(-f(X^*_k)) - \sum_{k=1}^{m} \exp(-f(X^*_k)) \\ &= \sum_{k=1}^{m} \exp(-f(X^*_k)) + \sum_{k=m+1}^{n_h} \exp(-f(X^*_k)) \\ &\quad - \sum_{k=1}^{m} \exp(-f(X^*_k)) \\ &= \sum_{k=m+1}^{n_h} \exp(-f(X^*_k)) \\ &\leqslant (n_h - m) \exp(-f(X^*_m)). \end{aligned} \tag{S.15}$$

4. Linear Assignment for Re-ID

Let $[\mathcal{A}] = \{1, \cdots, M\}$ be the set of M query (or probe) images and $[\mathcal{B}] = \{1, \cdots, N\}$ be the set of N gallery images. We define a binary indicator variable x^j_i such that $x^j_i = 1$ assigns the person index $i \in [\mathcal{A}]$ from the first set to the person index $j \in [\mathcal{B}]$ in the second set. Furthermore, let c^j_i be a matching cost for the assignment x^j_i. Therefore, for assigning all M persons in query set to the all N persons in gallery set, we define the solution using a binary vector

$$X = \left(x^j_i\right)_{i \in [\mathcal{A}],\ j \in [\mathcal{B}]}$$

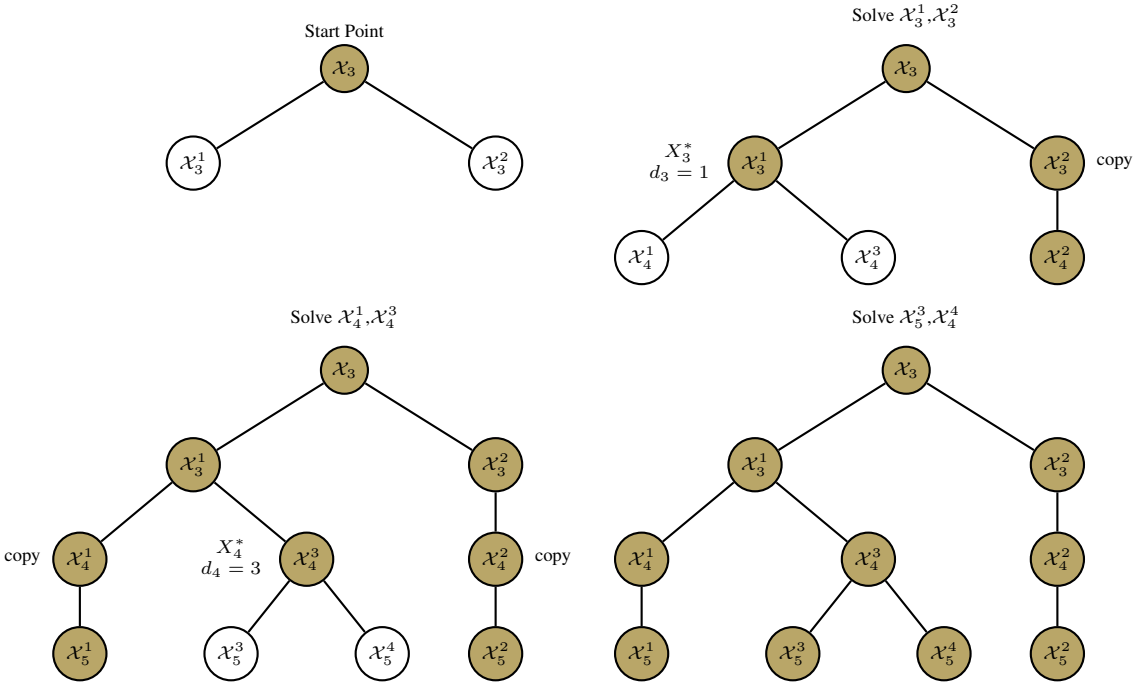

Figure 5. An example of Algorithm 1. The colored nodes are already computed, while the white ones are to be solved. In each iteration, we solve two integer programs derived from last iteration, while the other results are obtained by copying the previously computed ones. Note that by eliminating (fixing) variables, the scale of the problem becomes smaller than the original optimization problem.

and its associated cost

$$C = \left(c_i^j\right)_{i\in[\mathcal{A}],\ j\in[\mathcal{B}]}.$$

Since every individual in the query set has a unique match in the gallery set, every matching solution is defined on the one-to-one matching space $\mathcal{X}$. Therefore the marginalized cost c_i^j for matching the person index $i \in [\mathcal{A}]$ from the first set to the person index $j \in [\mathcal{B}]$ is obtained by calculating the m-best solutions of the following optimization problem:

$$X^* = \underset{X\in\mathcal{X}}{\operatorname{argmin}} \quad C^\top X, \tag{S.16}$$

Note that the above optimization problem is in fact a binary linear problem with linear constraints:

$$X^* = \underset{X}{\operatorname{argmin}} \quad C^\top X \tag{S.17}$$

$$\text{s. t.} \quad \forall i \in [\mathcal{A}], \forall j \in [\mathcal{B}] : x_i^j \in \{0,1\}, \tag{a}$$

$$\forall j \in [\mathcal{B}] : \quad \sum_{i \in [\mathcal{A}]} x_i^j \leqslant 1, \tag{b}$$

$$\forall i \in [\mathcal{A}] : \quad \sum_{j \in [\mathcal{B}]} x_i^j = 1. \tag{c}$$

The constraints ensure that every solution does not violate the one-to-one constraint (*i.e*. is located in the one-to-one matching space $\mathcal{X}$). More precisely, the inequality constraints S.17(b) guarantee that at most one person in the gallery set is associated with a person in query set and the equality constraints S.17(c) ensure that each person in the query set finds exactly one corresponding match in the gallery set.

5. Sequential Re-Identification

Here we elaborate on the sequential re-identification (SeqReID) introduced in Section 4.2 of the paper. In this problem, the aim is to successfully track visually similar objects in a video sequence by considering their appearance only and without using a motion prior. In other words, an object in successive time steps is assumed to belong to a trajectory as long as it is visually matched over the two frames. The main difficulty is that all objects look similar and, without considering motion, tracking is very challenging, even for humans (Fig. 6). Moreover, to have an acceptable tracking result, the matching results between two consecutive frames need to be almost impeccable. Our aim in introducing this problem is to further verify our marginalization idea for visual matching and to show that this idea can be incorporated into multi-target tracking frameworks to deal with cases where dynamic priors are weak; *e.g*. long occlusions or videos with low frame rate.

Competing method and datasets. In [2], a reliable multi-target tracking method (IHTLS) has been proposed which exclusively uses motion dynamics as a cue to track and distinguish targets with similar appearance. This method has been tested on several sequences including targets with very similar appearance such as human heads in crowds, flying birds and pedestrians from surveillance footage. We compare the results of our SeqReID tracker against this state-of-the-art multi-target tracking approach on videos where appearance is a weaker cue than motion. We use a part of their dataset including crowds, seagulls, TUD (crossing and campus) and we include additional challenging sequences into this evaluation. We include one of the popular PETS 2009 video sequences (S2L2), widely used for pedestrian tracking. Furthermore, we introduce three sequences for tracking players in a game of Australian Rules Football (AFL). The main difficulty in these sports sequences is that the players look very similar due to team uniform and that there is frequent crowding of players and interaction between them (Fig. 6 (Top)). The first AFL sequence is captured by a low resolution camera while the other two sequences are taken by higher resolution camera but temporally down-sampled in order to simulate low-frame-rate video.

Implementation details. Similar to [2], we use the annotated bounding box provided for every target. For two consecutive frames, we only extract two different visual features: the HSV histogram and the Histogram of Oriented Gradients (HOG) from each bounding box and calculate the Euclidean distance between them. We also use a fixed value as a cost for the '*dummy*' hypothesis $c_i^0 = 15, \forall\, i \in [\mathcal{A}]$. Then, the joint matching costs $\mathfrak{c}_i^j$ using our proposed marginalization approach for $m = 100$ are calculated. We build a trajectory by comparing the best match for the bounding boxes between these two consecutive frames. To ease the computational complexity and improve the results, we use a large and fixed gate size (350 pixels for all videos) to visually match every target with any bounding box candidate within this radius. The matching is performed frame-by-frame only and a track is terminated if the best match is assigned to the '*dummy*' term. All unmatched bounding boxes in the next frame are initiated as new trajectories.

For Dicle *et al.*'s tracker, we used their implementation script with their tuned parameters for the sequences such as Crowds, Seagulls and TUD (Crossing and Campus). Note that these parameters are tuned for each sequence separately. For the other sequences, we found the parameters that resulted in best performance, again separately for every sequence.

Performance measure. To quantitatively evaluate the tracking results, we report the commonly used multi-object tracking accuracy (MOTA) performance measures [1]. MOTA combines the errors such as false positives (FP), false negatives (FN)

Dataset (Sequence)	Method	IDs ↓	FP ↓	FN ↓	MOTA % ↑
Crowds	Dicle *et al.* [2]	23	0	2	99.88
	Our SeqReID	**19**	0	**0**	**99.91**
Seagulls	Dicle *et al.* [2]	**4**	0	3	**99.91**
	Our SeqReID	34	0	**0**	99.26
TUD (Crossing)	Dicle *et al.* [2]	**2**	0	0	**99.78**
	Our SeqReID	8	0	0	99.14
TUD (Campus)	Dicle *et al.* [2]	2	0	3	98.18
	Our SeqReID	2	0	**0**	**99.27**
PETS (S2L2)	Dicle *et al.* [2]	**35**	0	15	**99.51**
	Our SeqReID	60	0	**0**	99.42
AFL (S1)	Dicle *et al.* [2]	1	0	0	99.96
	Our SeqReID	**0**	0	0	**100.0**
AFL (S2)	Dicle *et al.* [2]	22	0	16	77.25
	Our SeqReID	**13**	0	**0**	**92.22**
AFL (S3)	Dicle *et al.* [2]	35	0	32	65.28
	Our SeqReID	**23**	0	**0**	**88.08**
Average	Dicle *et al.* [2]	**15.5**	0	8.8	92.46
	Our SeqReID	19.87	0	**0**	**97.16**

Table 1. Quantitative comparison results of our SeqReID tracker against the state-of-the-art tracker [2] for tracking visually similar targets on different video sequences.

Figure 6. The results of our SeqReID on AFL sequence.

and identity switches (IDs) into a single number. To fairly evaluate the performance of the trackers, the same evaluation script used in [2] is applied to quantify our results.

Results. The results reported in Table 1 indicate that a very simple mechanism described above can track visually similar targets as well as or even better than IHTLS in most of the sequences w.r.t. MOTA. However since we ignore the motion of the targets, our results include more ID switches compared to IHTLS. The sequences such as AFL (S2 and S3) seem more challenging for IHTLS due to the unreliable motion patterns of the targets.

Video. The accompanying video shows the tracking results for the aforementioned video datasets using our SeqReID approach.

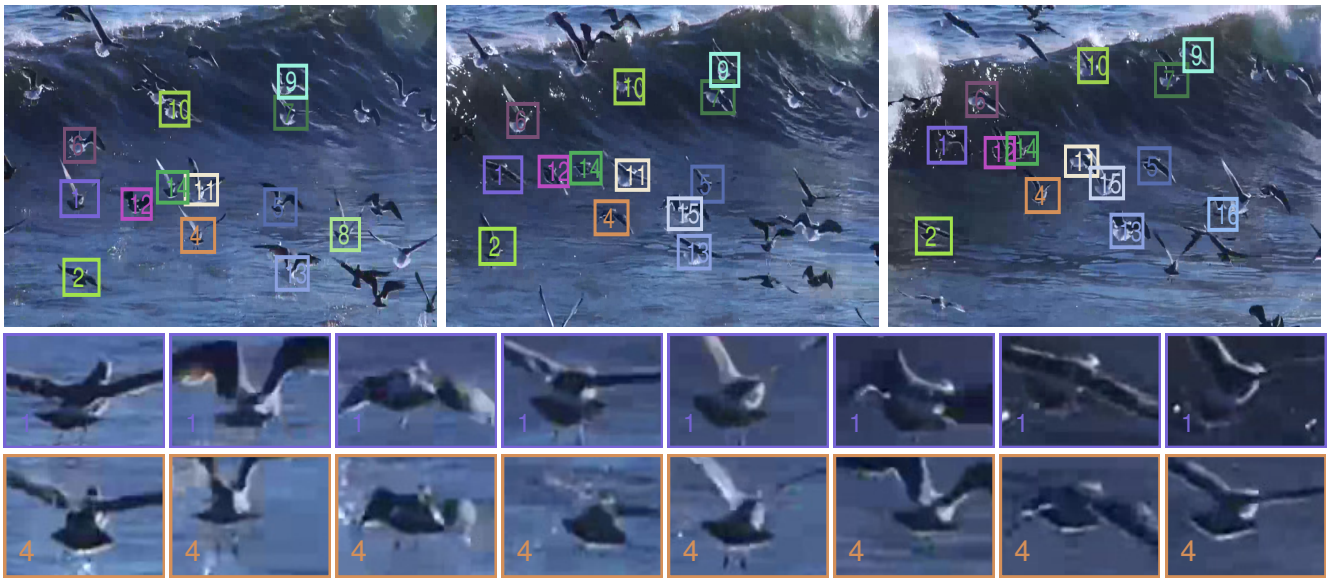

Figure 7. The results of our SeqReID on seagulls sequence.

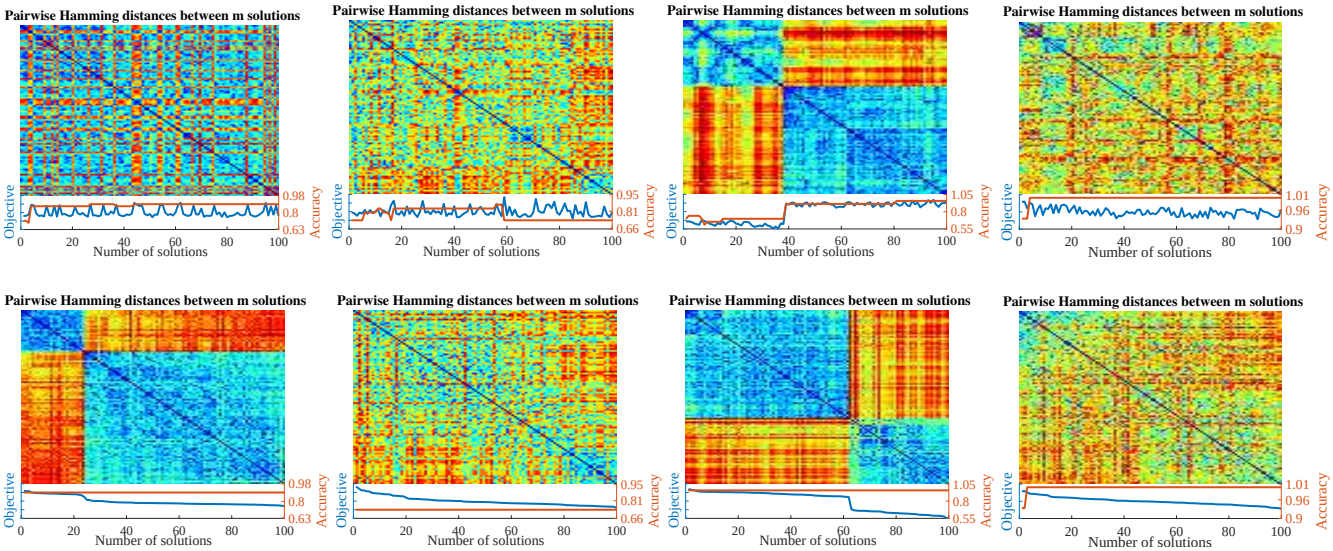

Figure 8. Pairwise Hamming distance between m unsorted (first row) and sorted (last row) solutions calculated using IPFP and BP solvers for two feature points matching examples. Objective values (blue) and matching accuracy (red) using marginalization for each solution is shown on the bottom.

6. Further Analysis on Feature Matching

As we discussed in the main text, exact marginalization over all solutions empirically shows the same or superior performance compared to the MAP problem. However, we approximate the marginalization by m solutions. In the feature matching application with quadratic objective costs, the optimality of each of the m estimated solutions cannot be guaranteed. Therefore, there exist some particular instances for which marginalization does not lead to a better matching accuracy or sometimes even degrades the performance. Nonetheless, the overall matching accuracy is still improved on average due to the effect of marginalization and/or finding a better objective value by iterating through successive solutions.

Here, we include some more examples (Fig. 8) to show the diversity of the m solutions by their pairwise Hamming distance, their objectives and the matching accuracy after marginalization over the number of solutions. Our experiments show that there is no apparent correlation between the diversity of solutions and their contribution to the joint matching probability distribution.

References

[1] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *Image and Video Processing*, 2008(1):1–10, May 2008. 6

[2] C. Dicle, O. I. Camps, and M. Sznaier. The way they move: Tracking multiple targets with similar appearance. In *ICCV*, pages 2304–2311, 2013. 6, 7