ProgTrack: A Multi-Object Tracking Algorithm with Progressive Matching Strategy Supplementary Material (Paper ID: 39378)

1. Details of PKF

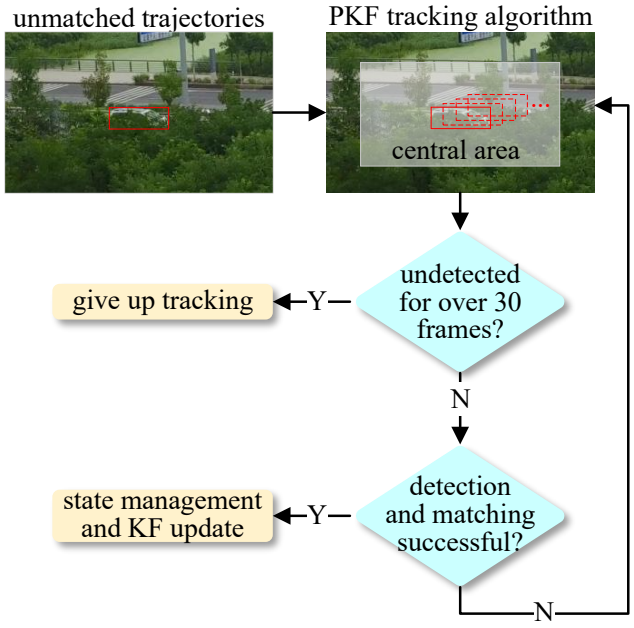

Figure 1. The flowchart of the PKF tracking algorithm.

The PKF tracking strategy process is shown in Figure 1. The strategy targets trajectories that do not match and are located in the central region. Firstly, the PKF tracking algorithm is employed to track targets that meet the above-mentioned conditions. Secondly, it checks whether the target has been undetected for more than 30 frames: if it does, the tracking is abandoned; if not, the tracking results from PKF are matched with the detection results (if any). Third, if the matching is successful, state management and KF state update are performed; if the matching fails, the PKF tracking algorithm continues to carry out the tracking. The algorithm 1 presents the detailed process of the PKF Tracking Strategy.

Algorithm 1 Pseudo-code of PKF Tracking Strategy

Input: Unmatched trajectories (tracks)
Output: Processed results of unmatched tracks
Pseudo Code:

```
for each track in unmatched_tracks do
    if track is located in central region then
        // 1. Init PKF prediction parameters
        PKF_counter ← 0
        target_detected ← false
        // 2. PKF prediction phase
        while PKF_counter < 30 and not target detected do
            // 2.1. Predict next state using KF
            track.predicted ← KF.Predict(track.current)
            // 2.2. Check detection and matching status
            if DetectAndMatch(track.predicted_state) is successful then
                // 2.2.1. Update state with detection and reset counter
                track.current_state ← KF.Update(track.predicted_state, matched_detection)
                track.status ← "NORMAL_TRACK"
                target_detected ← true
            else
                // 2.2.2. Continue with PKF
                track.current ← track.predicted
                PKF_counter ← PKF_counter +1
                track.status ← PKF_prediction
            end if
        end while
        // 3. Post-processing after prediction phase
        if not target_detected then
            // Retain predicted position for 30 frames
            track.status ← "RETAINED_POSITION"
            track.retain_counter ← 30
        end if
    end if
end for
```

2. Visualization Results of the GRNED Module

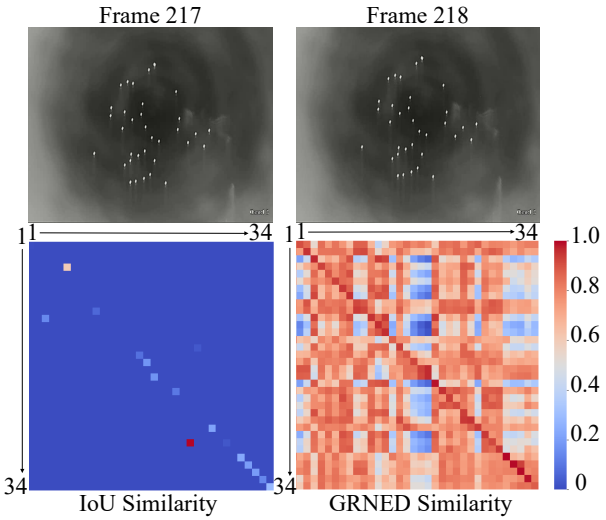

Figure 2. The visualization results of the GRNED module.

The GRNED module, as an implementation method of the GMI strategy, exhibits excellent matching performance facing lens shake and target occlusion. Figure 2 shows the visualization result of the GRNED module. The data comes from the 217th and 218th frames of the 6th video in the MultiUAV training dataset, which is seriously characterized by lens shake. The two pictures at the bottom are obtained using the IoU and GRNED matching algorithms. The value at coordinate (i,j) represents the IoU or GMI similarity between the i-th and j-the target in adjacent frames. If a value on the main diagonal is greater than all values in its row and column, the corresponding target is successfully matched.

By comparing the main diagonals of the two similarity graphs, it can be observed that when using IoU matching, less than half of the similarities on the main diagonal are greater than the values in the corresponding rows and columns. This means that less than half of the targets are successfully matched.

When using GRNED matching, the most obvious feature is that the similarity score of each coordinate is almost greater than 0, which is attributed to the feature of the global motion information based matching strategy. At the same time, it can be observed that the similarity scores of the main diagonal are almost all higher than those of the corresponding rows and columns. This indicates that almost all targets have been successfully matched.

3. Visualization of Training and Validation Results of FastReID and CE-ReID

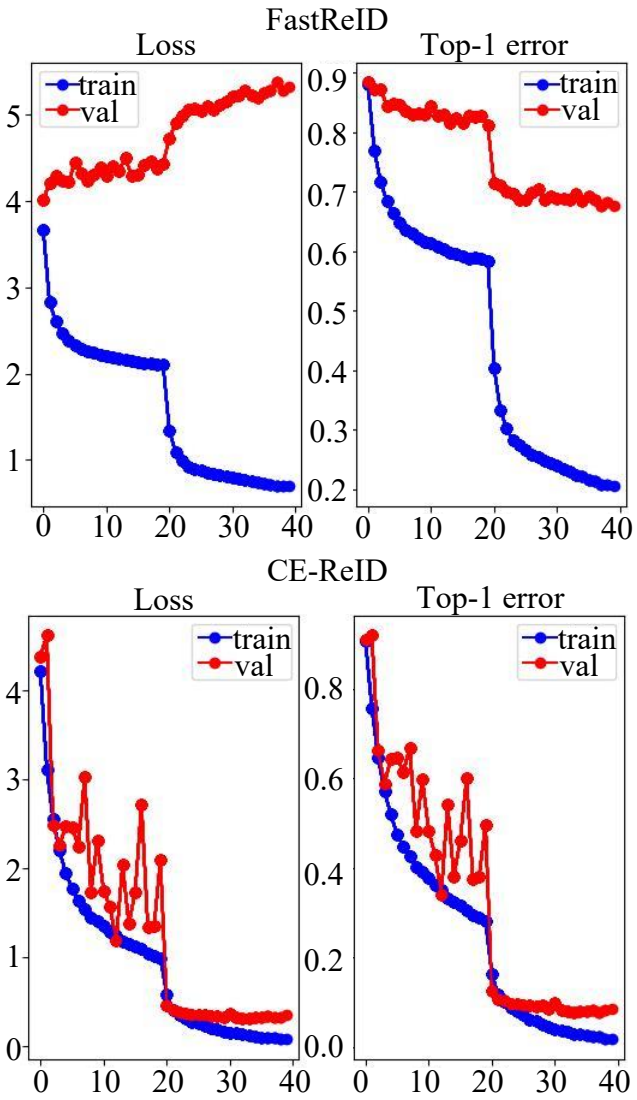

Figure 3. The Loss and Top1err curves of FastReID and CE-ReID.

As shown in Figure 3, it presents the Loss and Top1err curves of FastReID and CE-ReID during training and validation. The loss curve tracks the prediction error of a ReID module during training. Lower values mean better learning. The Top-1 error curve measures incorrect top predictions in classification. A decline indicates improved performance.

For the training stage, it can be observed that the de-

crease rate of the Loss and Top1 error of the CE-ReID module is much faster than that of the FastReID module. By the 20th epoch, the CE-ReID module has already decreased to a Loss of 1 and a Top1 error of 0.2. While at this point, the Loss and Top1 error values of the FastReID module are twice those of the CE-ReID module. This is because during the training phase, the CE-ReID module employed a combined loss to ensure the correct optimization of the model parameters. The formula for the combined loss is as follows:

$$CombinedLoss = LocalFeatureLoss + ContextEnhancedFeatureLoss \quad (1)$$

During the validation stage, for the Loss curve, the loss value of FastReID was extremely unstable and even showed an upward trend. In contrast, CE-ReID eventually dropped to a level similar to the loss value during training. For the Top-1 error curve, the Top-1 error of FastReID basically remained at around 0.7 and did not decrease further, while the Top-1 error of CE-ReID almost dropped to 0. It is demonstrated that the FastReID model has weaker generalization ability and inferior feature extraction capability compared to CE-ReID.

4. Distinguishability in Extreme Scenarios

Table 1. Strategies and Performance in Extreme Scenarios

Scenario	Strategy	Metric	Score
Severe Occlusion	CE-ReID	AUC of ROC	0.871
Full Occlusion	PKF	MOTA	0.384
Lens Jitter / Small Object	GRNED	Matching Rate	0.965

To verify the robustness of ProgTrack in extreme visual scenarios, we conducted dedicated experiments on the MultiUAV dataset, which covers the most challenging conditions in UAV-based multi-object tracking, including severe occlusion, full occlusion, lens jitter, and small-scale objects. Table 1 summarizes the specialized strategies employed by ProgTrack to address each scenario and the corresponding performance metrics.

For severe occlusion, where targets are heavily obscured by background or neighboring objects, the CE-ReID (Context Enhanced Re-ID) strategy is activated. This module leverages spatial context information to enrich target appearance features, effectively mitigating the degradation of discriminative power caused by occlusion. As a result, it achieves an AUC of ROC score of 0.871, demonstrating strong capability to distinguish targets even under heavy visual interference.

In full occlusion scenarios, where target appearances are completely unavailable, the PKF (Pure Kalman Filter) strategy takes over. It relies on precise kinematic state prediction to maintain target trajectories during occlusion, avoiding track loss or drift. This strategy yields a MOTA score of 0.384, which reflects its effectiveness in preserving tracking continuity when no visual detection is available.

For lens jitter and small-scale objects, which introduce inter-frame misalignment and weak target features respectively, the GRNED (Global Motion Information Matching) module is utilized. By modeling global motion patterns and matching based on trajectory consistency, it achieves a high matching rate of 0.965, successfully addressing the matching failures caused by camera shake and low-resolution small targets.

For comparison, we also evaluated two state-of-the-art trackers, BoT-SORT and TOPIC, on the same set of extreme scenarios. BoT-SORT achieves scores of **[0.845, 0.304, 0.847]** for the three scenarios, while TOPIC obtains **[0.851, 0.327, 0.896]**. The results clearly show that ProgTrack's modular, scenario-specific design outperforms these competitors across most metrics, confirming its superior robustness and generalization ability in complex UAV tracking environments.

5. Sensitivity Analysis of PKF Window Size

Table 2. Performance Variation with PKF Window Size

Window Size	MOTA (↑)	IDs (↓)	+MOTA	-IDs
27	39.30	22195	-	-
28	39.65	21888	+0.35	-307
29	39.92	21605	+0.27	-283
30	40.23	21286	+0.31	-319
31	40.35	21155	+0.12	-131
32	40.39	21083	+0.04	-76
33	40.40	21058	+0.01	-25

To determine the optimal window size for trajectory retention, we conducted an ablation study across window sizes ranging from 27 to 33 frames. The results, presented in Table 2, reveal a clear trade-off between tracking performance and memory consumption.

Below 30 frames: The performance scales nearly linearly with the window size. As the window size increases from 27 to 30, the MOTA improves steadily from 39.30 to 40.23, while the number of ID switches (IDs) decreases significantly from 22195 to 21286, with consistent gains of +0.27 to +0.35 in MOTA and reductions of 283 to 319 in ID switches per step. This indicates that a longer window effectively preserves more historical trajectory information, enhancing tracking continuity and accuracy.

At 30 frames: This size achieves the best balance between performance and memory overhead. It delivers a notable 0.93 MOTA improvement and 909 fewer ID switches compared to the 27-frame baseline, while avoiding excessive memory usage that would come with larger windows.

Beyond 30 frames: The performance growth follows a logarithmic trend, with diminishing returns. Increasing the window size from 30 to 33 only yields a marginal 0.17 MOTA gain (from 40.23 to 40.40) and a small reduction of 228 ID switches (from 21286 to 21058). The incremental benefits become negligible, indicating that 30 frames is the sweet spot where further increases in window size do not justify the additional memory cost.