TOSHIBA

ICPR2026 International Conference on Pattern Recognition LYON

ICPR2026 #692

ADGR: Adaptive Density-Guided Graph Re-ranking for Person Re-identification

Yushiro Kashimoto, Osamu Yamaguchi — Toshiba Corporation

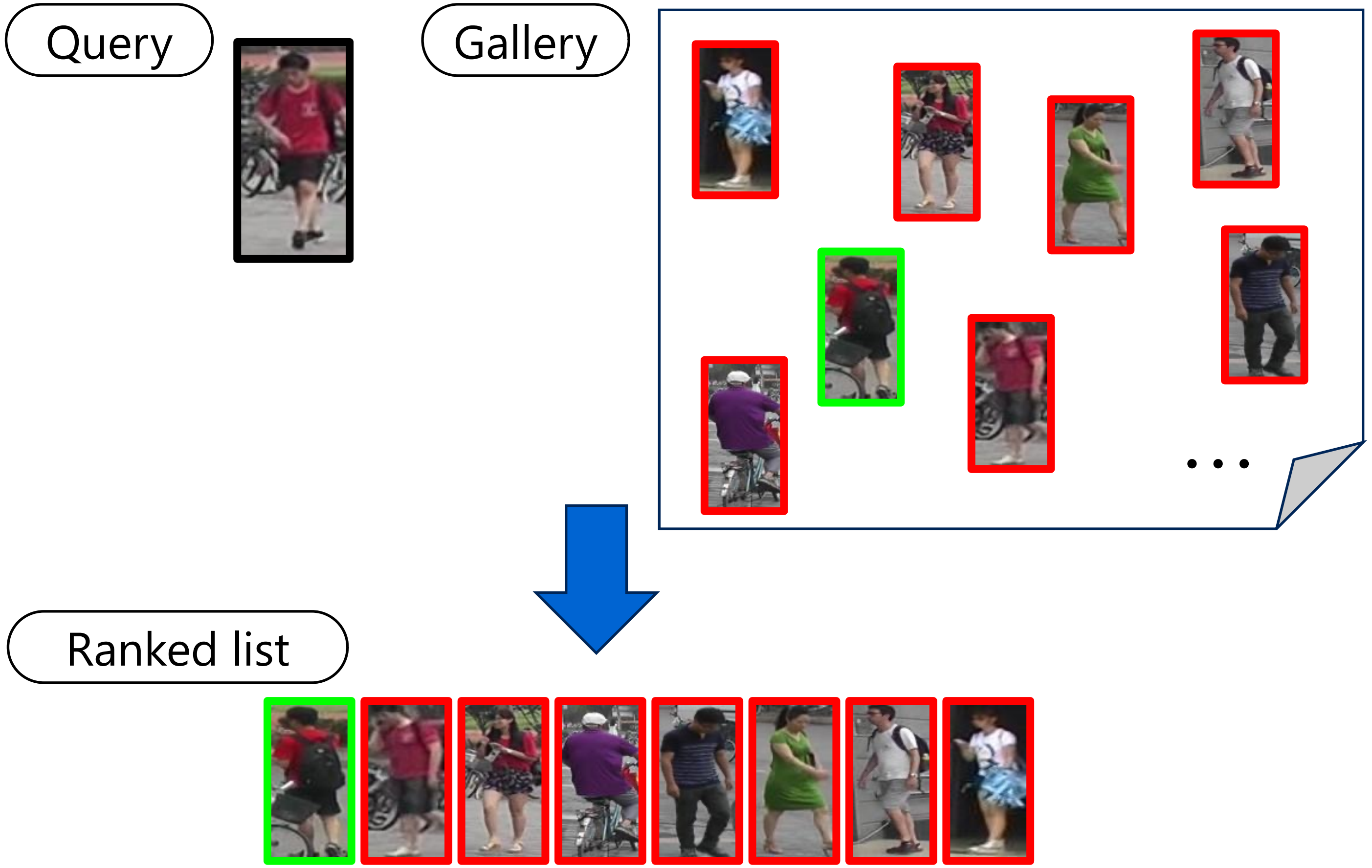
Person Re-Identification (Re-ID)

Person Re-Identification (Re-ID)

【Input】
Query image and gallery images captured by multiple cameras.

【Output】
A ranked list of gallery images according to similarity.

【Applications】
Video surveillance, multi-camera tracking, forensic search



Retrieve images of the same person captured by different cameras

Re-ranking for Person Re-ID

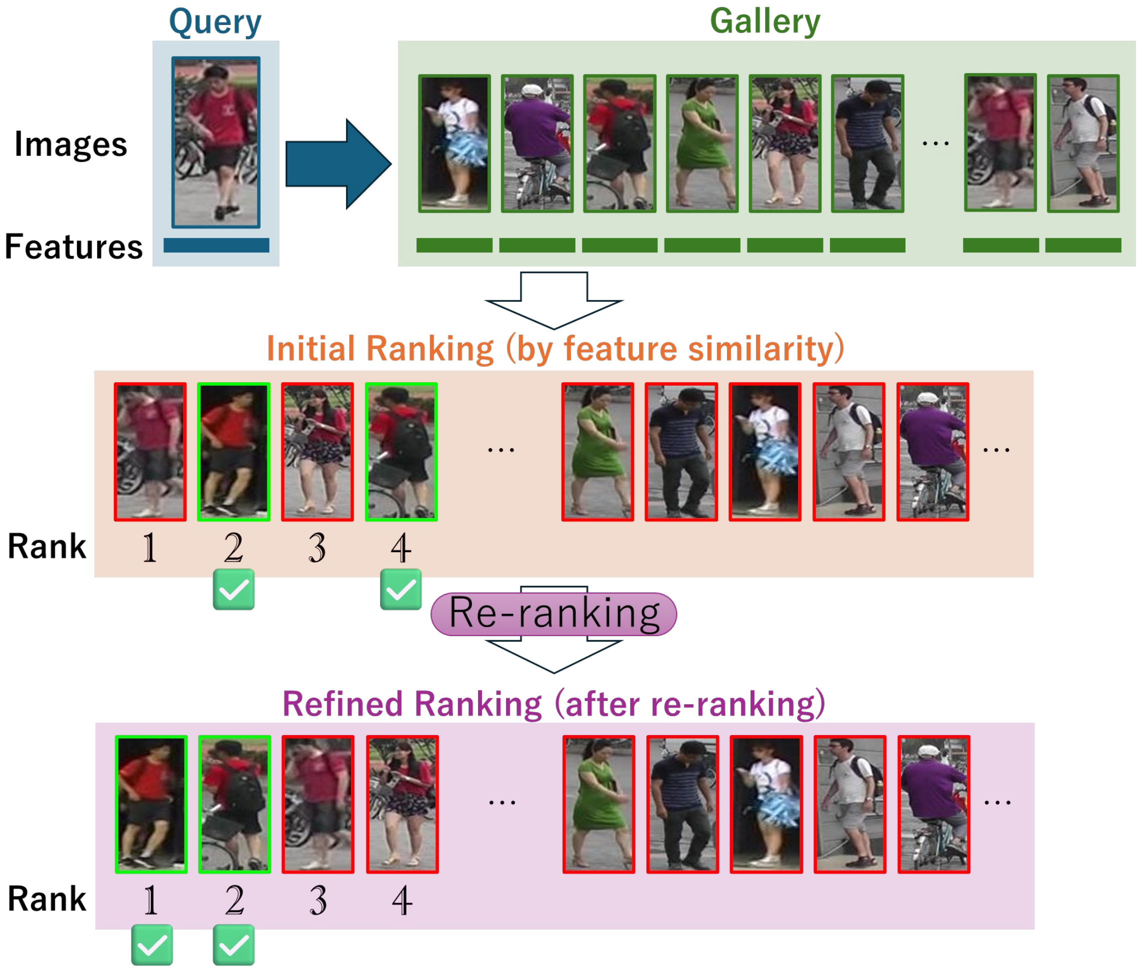
Re-ranking for Re-ID

【Goal】
Improve the ranking quality of Re-ID retrieval results.

【Input】
Initial retrieval results generated by feature-based similarity matching

【Output】
Re-ranked retrieval results reflecting contextual information among samples.

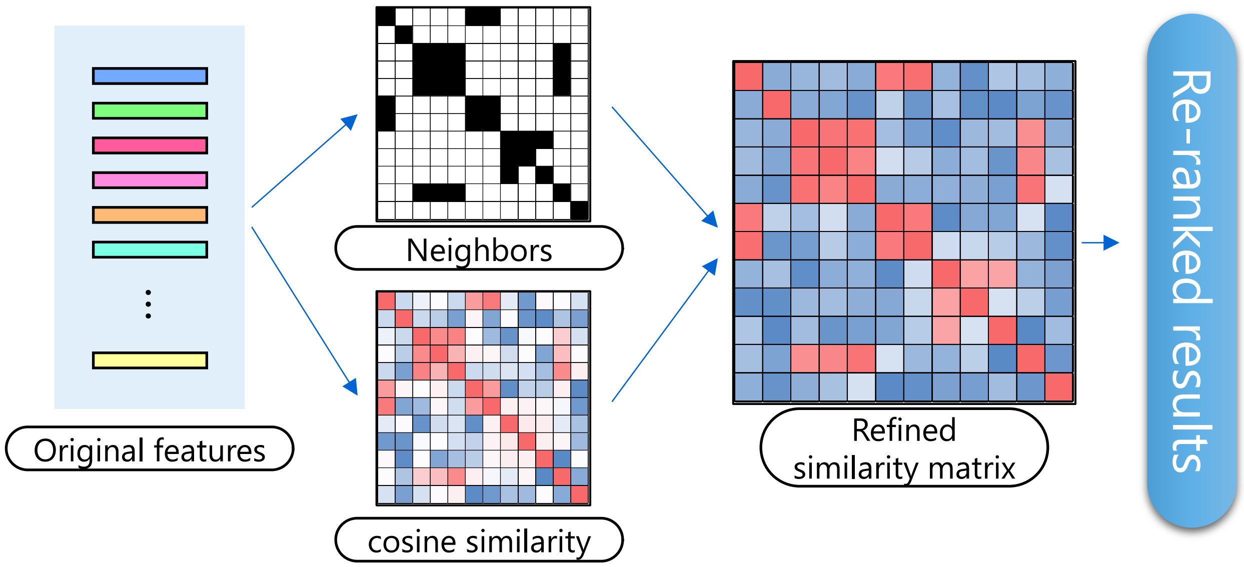
【Benefit】
Significantly improves Rank-1 and mAP performance without additional model training.



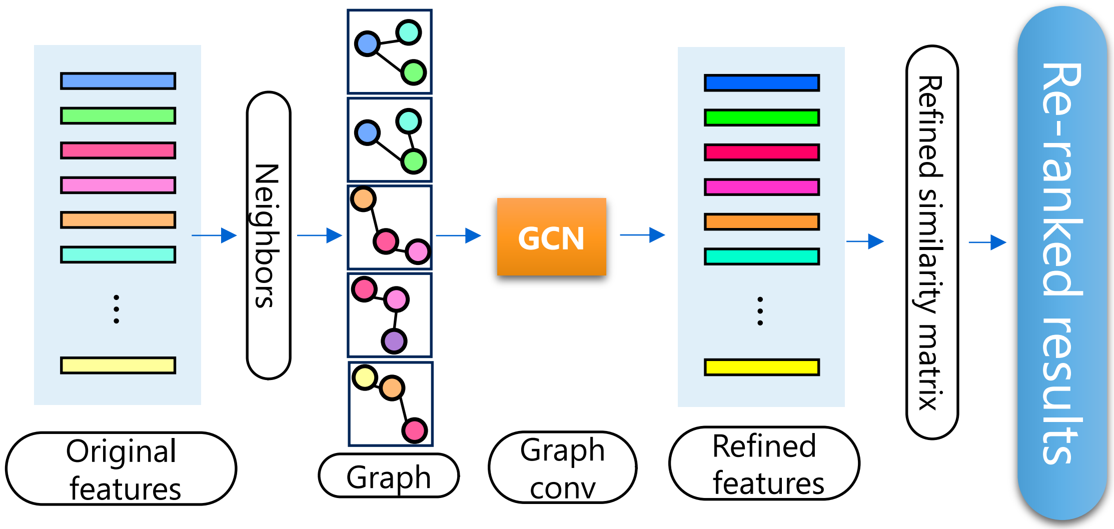
Widely used post-processing technique in Person Re-ID

Re-ranking Methods

Similarity Refinement-based Re-ranking



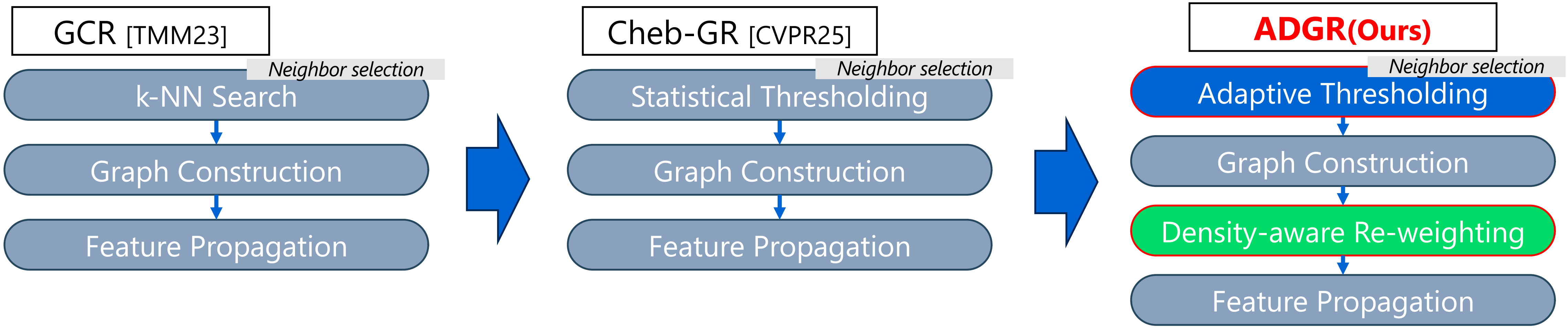
Feature Refinement-based Re-ranking



	Similarity Refinement-based Re-ranking	Feature Refinement-based Re-ranking
Refined Target	N^2 similarities (distance)	N features
Example Methods	k-reciprocal, ECN	GCR, Cheb-GR
Strength	Direct ranking correction.	Context-aware feature enhancement. Flexibility for incremental retrieval systems.
Limitation	Context is reflected only in similarity scores.	Neighbor / graph quality is critical.

Evolution of Feature Refinement-based Re-ranking

Graph based methods



😊High accuracy
😥Expensive k-NN search

😊Fast processing
😥Unstable neighbor selection

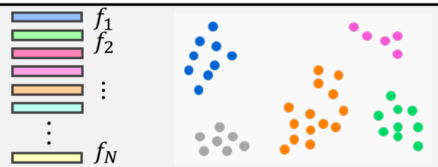
😊Fast processing
😊Stable neighbor selection

How can we achieve both fast processing and stable neighbor selection?

Overview of ADGR

1. Features

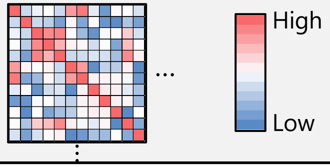
Extracted feature vectors



Input feature vectors of query and gallery images.

2. Similarity Matrix

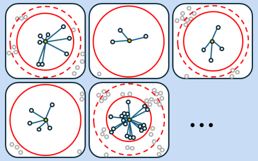
Compute pairwise similarities



Build the initial similarity matrix and obtain the initial ranking.

3. Adaptive Threshold Control (Select Neighbors)

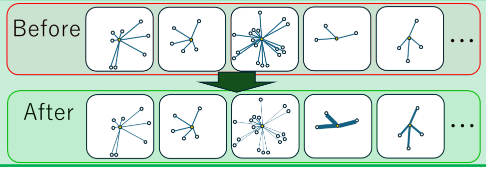
Select reliable neighbors using adaptive thresholds



Adaptively determine the number of neighbors for each sample.

4. Density-aware Similarity Re-weighting

Adjust edge weights according to local density



Re-weight similarities based on **local density** to achieve balanced propagation.

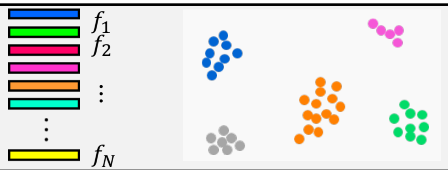
5. Graph-based Propagation

Propagate features on the adjusted graph

Propagate information through the adjusted graph to refine feature vectors.

6. Refined features

Output refined feature vectors



Obtain refined features that encode contextual information more effectively.

7. Re-ranked Results

Compute similarities with refined features and obtain final ranking

Produce the final ranking using the refined features.

Overview of ADGR

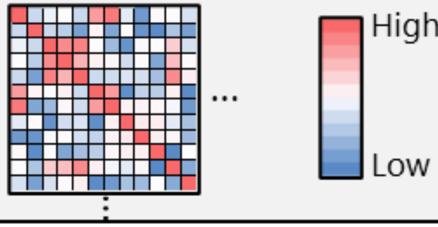
1. Features

Extracted feature vectors

Input feature vectors of query and gallery images.

2. Similarity Matrix

Compute pairwise similarities

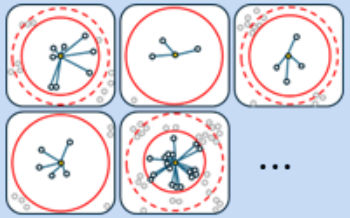


Build the initial similarity matrix and obtain the initial ranking.

3. Adaptive Threshold Control

(Select Neighbors)

Select reliable neighbors using adaptive thresholds



Adaptively determine the number of neighbors for each sample.

Key Idea 1

Adaptive Threshold Control

Stabilizes neighbor selection across different datasets and feature extractors.

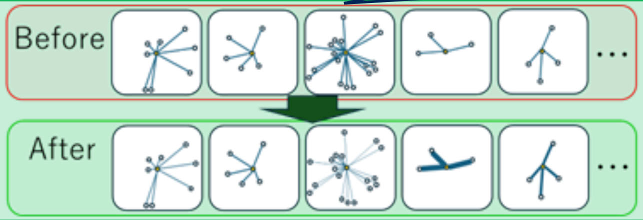
Overview of ADGR

Key Idea 2 **Density-aware Similarity Re-weighting**

Adjusts propagation weights
according to local data density.

4. Density-aware Similarity Re-weighting

Adjust edge weights according to local density



Re-weight similarities based on **local density** to achieve balanced propagation.

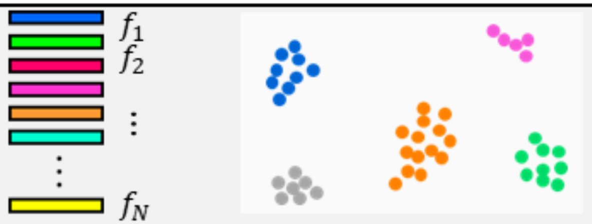
5. Graph-based Propagation

Propagate features on the adjusted graph

Propagate information through the adjusted graph to refine feature vectors.

6. Refined features

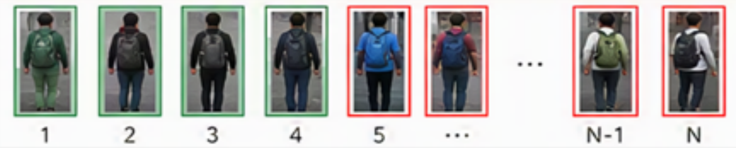
Output refined feature vectors



Obtain refined features that encode contextual information more effectively.

7. Re-ranked Results

Compute similarities with refined features and obtain final ranking

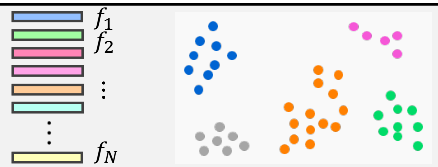


Produce the final ranking using the refined features.

Overview of ADGR

1. Features

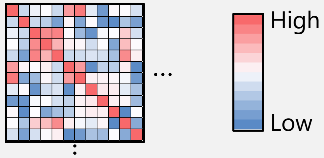
Extracted feature vectors



Input feature vectors of query and gallery images.

2. Similarity Matrix

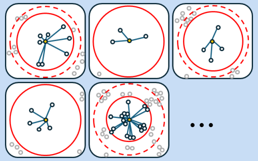
Compute pairwise similarities



Build the initial similarity matrix and obtain the initial ranking.

3. Adaptive Threshold Control (Select Neighbors)

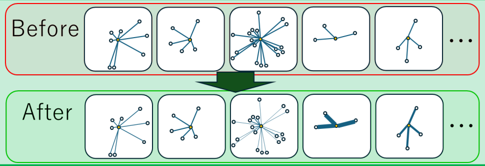
Select reliable neighbors using adaptive thresholds



Adaptively determine the number of neighbors for each sample.

4. Density-aware Similarity Re-weighting

Adjust edge weights according to local density



Re-weight similarities based on **local density** to achieve balanced propagation.

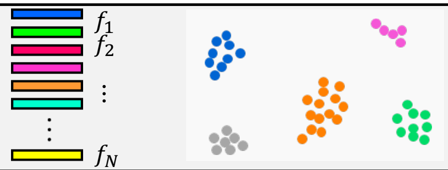
5. Graph-based Propagation

Propagate features on the adjusted graph

Propagate information through the adjusted graph to refine feature vectors.

6. Refined features

Output refined feature vectors



Obtain refined features that encode contextual information more effectively.

7. Re-ranked Results

Compute similarities with refined features and obtain final ranking

Produce the final ranking using the refined features.

Key Idea 1

Adaptive Threshold Control

Stabilizes neighbor selection across different datasets and feature extractors.

Key Idea 2

Density-aware Similarity Re-weighting

Adjusts propagation weights according to local data density.

Fast processing with stable neighbor selection and effective feature propagation

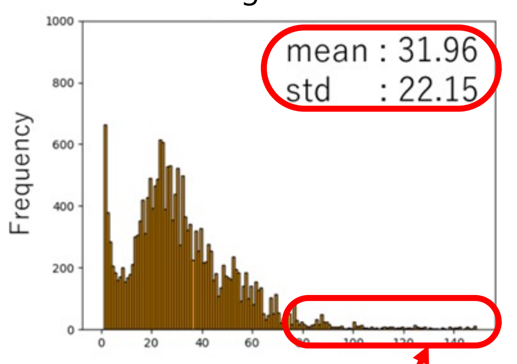
Adaptive threshold control for neighbor selection

Limitation

Fixed λ (Cheb-GR)

A fixed λ-based statistical threshold may lead to **unstable neighbor counts**.

Distribution of threshold-selected neighbor counts



Number of neighbors

Too many selected neighbors

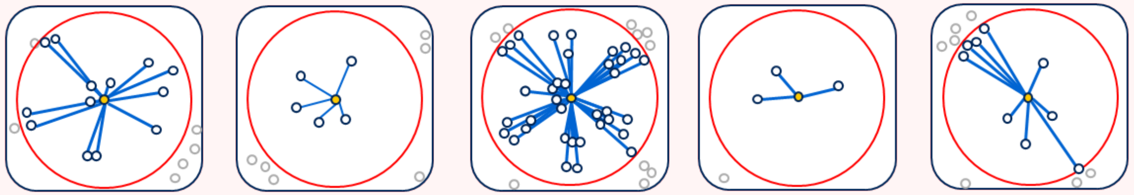
High variance in neighbor counts

Method Details

1. distance $A_{ij} = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2$

2. Initial threshold $T_i = \mu_i - \lambda\sigma_i$

Before



num: 13, 5, 36, 3, 9 …

Query sample | Selected neighbor | Non-selected neighbor | Threshold | Init threshold

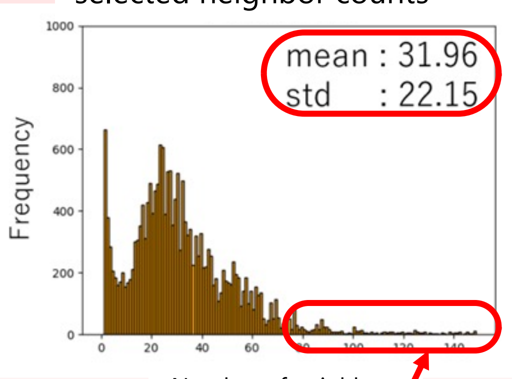
Adaptive threshold control for neighbor selection

Limitation

Fixed λ (Cheb-GR)

A fixed λ-based statistical threshold may lead to **unstable neighbor counts**.

Distribution of threshold-selected neighbor counts



Too many selected neighbors

High variance in neighbor counts

Method Details

1. distance $A_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$

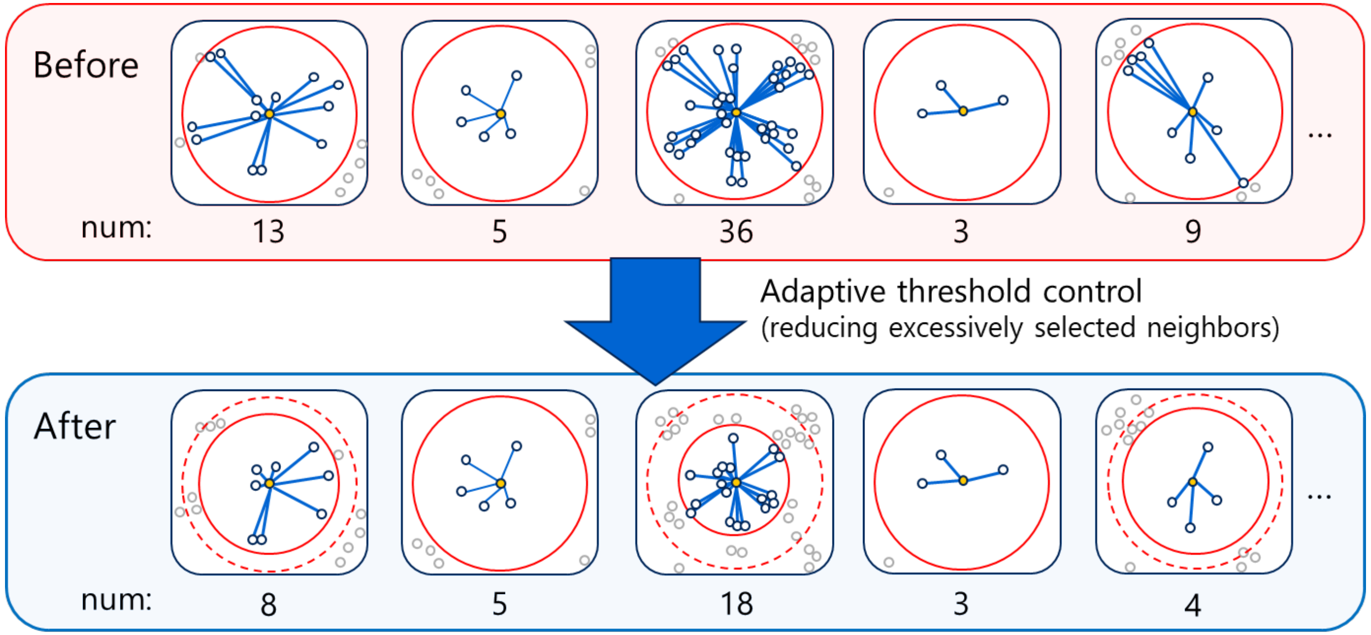
2. Initial threshold $T_i = \mu_i - \lambda\sigma_i$

3. Count ni $n_i = |\{j | A_{ij} \le T_i\}|$

4. Control lambda $\lambda_i = \begin{cases} \lambda, & n_i \le k \\ \lambda l_i, & n_i > k \end{cases}$

$l_i = \sigma(n_i - k) l_{range} + 1 - \frac{1}{2} l_{range}$

5. Adjust threshold $T_i{}' = \mu_i - \lambda_i\sigma_i$



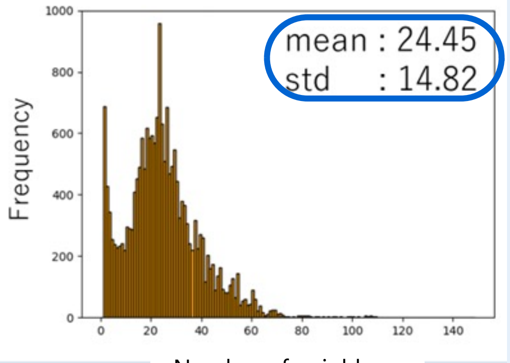
Query sample | Selected neighbor | Non-selected neighbor | Threshold | Init threshold

Our Approach

Adaptive λ_i (ADGR)

Adaptive λ_i-based threshold control **balances neighbor counts** across samples.

Distribution of threshold-selected neighbor counts

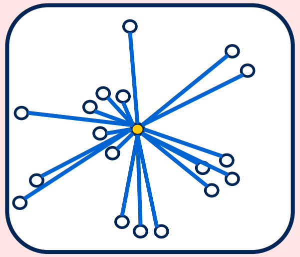


Excessive selected neighbors are suppressed.

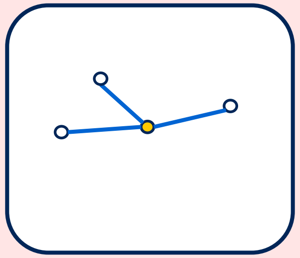
Similarity Re-weighting for propagation

Limitation

Uniform propagation ignores local density



Dense region
Too many neighbors
→ over-smoothing



Sparse region
Limited neighbors
→ weak propagation

Need density-aware propagation weights

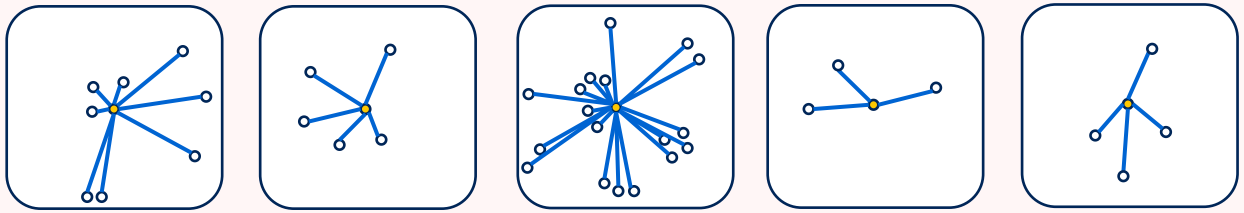
Method Details

1. Similarity $\hat{A}_{ij} = \exp(-\tilde{A}_{ij}/\gamma)$

2. Local density $D_i = \sum_j \hat{A}_{ij}$

3. Propagate $\mathbf{X}_{t+1} = \mathbf{D}^{-1/2}\hat{\mathbf{A}}\mathbf{D}^{-1/2}\mathbf{X}_t$

Before

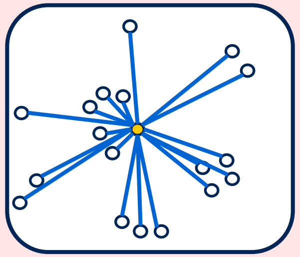


…

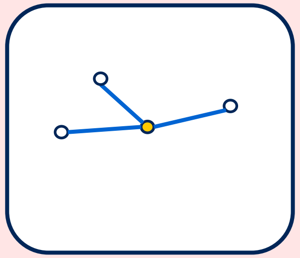
Similarity Re-weighting for propagation

Limitation

Uniform propagation ignores local density



Dense region
Too many neighbors
→ over-smoothing



Sparse region
Limited neighbors
→ weak propagation

Need density-aware propagation weights

Method Details

1. Similarity $\hat{A}_{ij} = \exp(-\tilde{A}_{ij}/\gamma)$

2. Local density $D_i = \sum_j \hat{A}_{ij}$

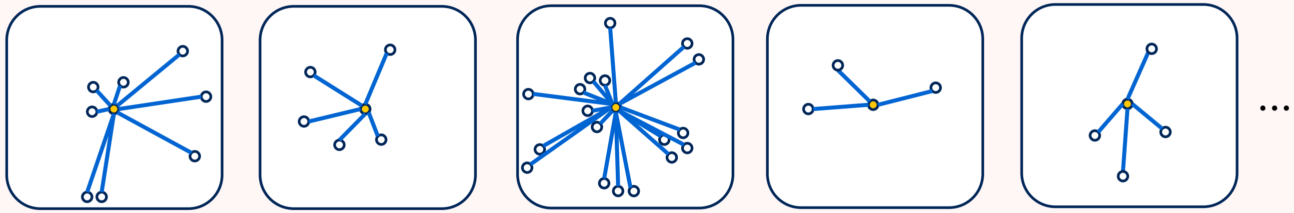
3. Scaling factor $a_i = \sigma\left(D_i - \frac{1}{N}\sum D_i\right) a_{range} + 1 - \frac{1}{2} a_{range}$

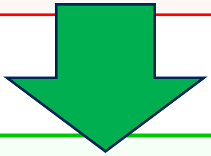
4. Re-weight $\hat{A}'_{ij} = \hat{A}_{ij}{}^{a_i}$

5. Propagate $\mathbf{X}_{t+1} = \mathbf{D}^{-1/2}\hat{\mathbf{A}}'\mathbf{D}^{-1/2}\mathbf{X}_t$

$0 < \hat{A}_{ij} < 1$: $a_i > 1 \rightarrow$ **weaker**; $a_i < 1 \rightarrow$ **stronger**

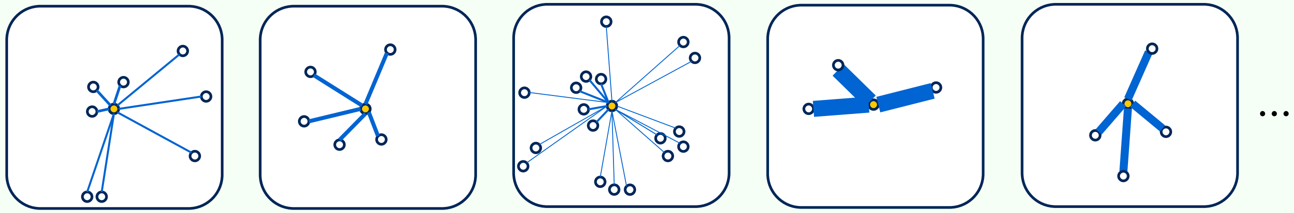
Before





Density-aware similarity scaling
(stronger in sparse regions,
weaker in dense regions)

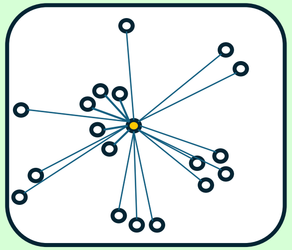
After



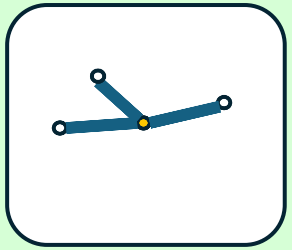
Query sample Selected neighbor Edge ()

Our Approach

Balanced feature propagation

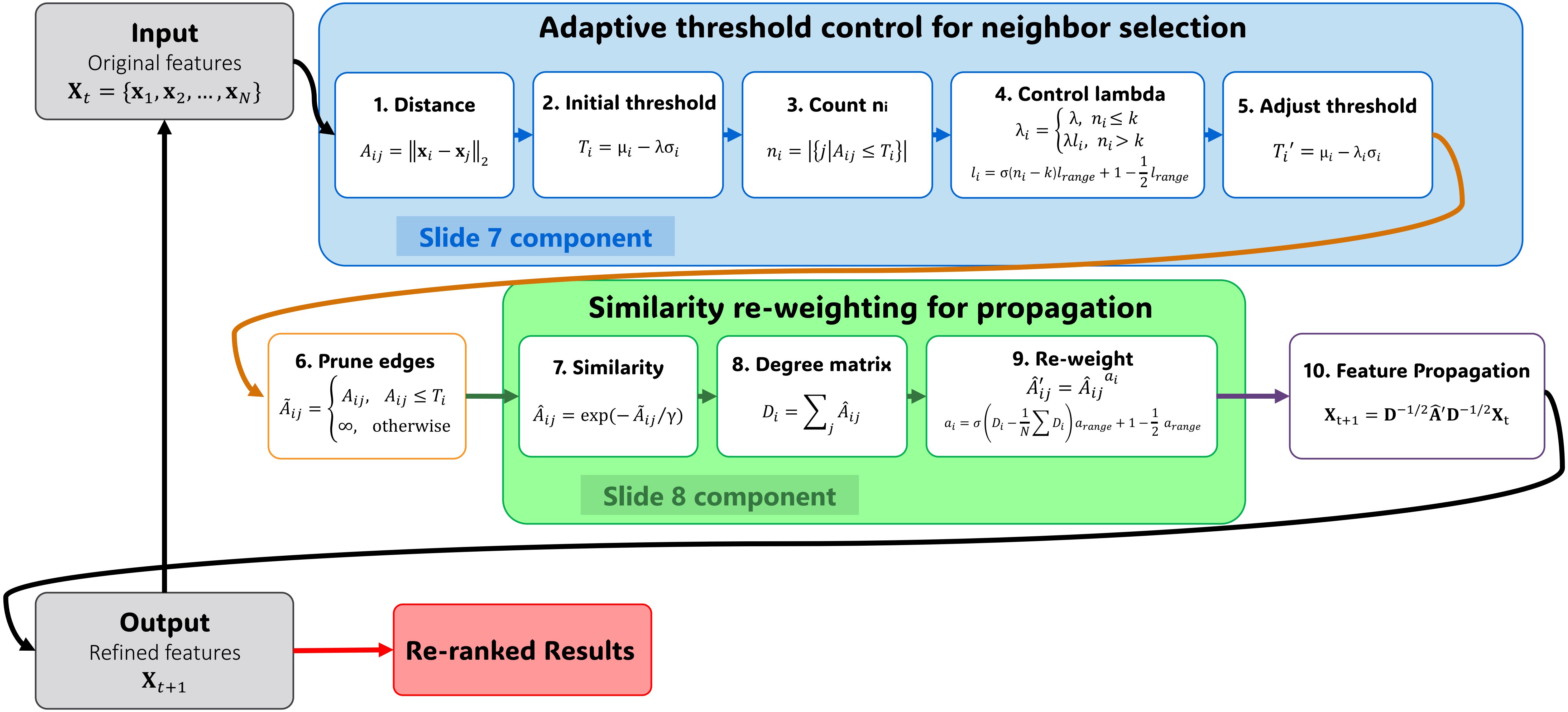


Dense region
weaker weights
reduce excessive influence



Sparse region
Stronger weights
use reliable neighbors

Overall Algorithm Flow



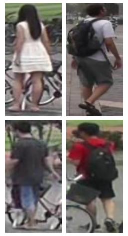
Experimental Setup

The experimental setup is designed based on Cheb-GR for fair comparison.

Datasets

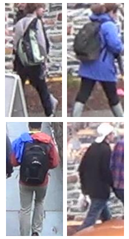
Market-1501 (M)
Zheng, L et al. [Scalable person re identification: A benchmark.] (2015)

query : 3,368
gallery : 19,732
ID : 750



DukeMTMC-reID (D)
Ristani, E et al. [Performance measures and a data set for multi-target, multi-camera tracking.] (2016)

query : 2,228
gallery : 17,661
ID : 702

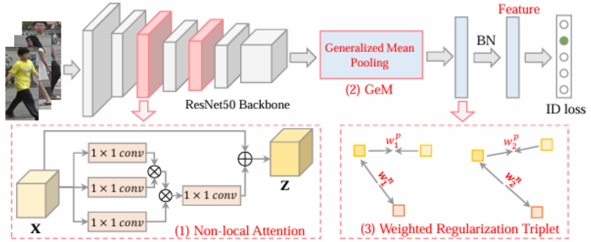


CUHK03 (C)
Li, W et al. [Deep filter pairing neural network for person re-identification.] (2014)

query : 1,400
gallery : 5,332
ID : 700

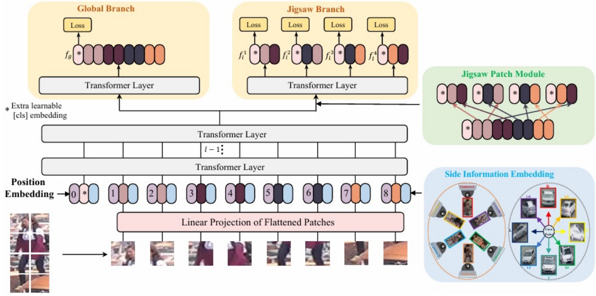
Feature Extractors

AGW
Ye, M et al. [Deep learning for person re-identification: A survey and outlook.] (2021)



TransReID
He, S et al. [TransReID: Transformer based object re-identification.] (2021)

Transformer-based backbone



Compared Methods

Feature aggregation

QE Chum, O et al. [Automatic query expansion with a generative feature model for object retrieval.] (2007)

Similarity refinement

KR Zhong, Z et al. [Re-ranking person re-identification with k reciprocal encoding.] (2017)

ECN Sarfraz, M.S et al. [A pose-sensitive embed ding for person re-identification with expanded cross neighborhood re-ranking.] (2018)

Feature transformation / propagation

LBR Luo, C et al. [Spectral feature transformation for person re-identification.] (2019)

GCR Zhang, Y et al. [Graph convolution based efficient re-ranking for visual retrieval.] (2023)

Cheb-GR Yang, J et al. [Cheb-GR: Rethinking K-nearest Neighbor Search in Re-ranking for Person Re-identification.] (2025)

Evaluation Metrics

- **CMC(Rank-1)**
- **mAP**
- **Processing time**

GPU:NVIDIA A100,
CPU:Intel(R) Xeon(R) Gold 5215 CPU @ 2.50GHz

Implementation Details

$\boldsymbol{\lambda}$: 4.2~6.0
- AGW
  λ = 6.0/5.3/6.0
  for M/D/C
- TransReID
  λ = 5.3/4.3/4.2
  for M/D/C

$\boldsymbol{\gamma}$: 0.2

$\boldsymbol{k}$: 15

$\boldsymbol{T}$: 3

follows the **Cheb-GR** settings

l_{range} : 0.3
For adaptive threshold control

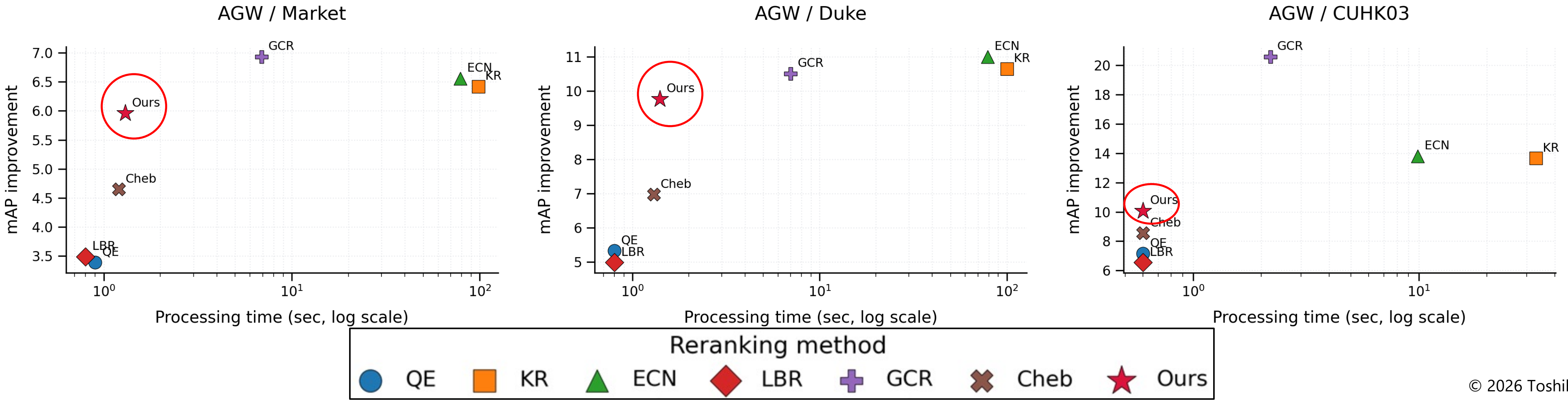
a_{range} : 0.8
Similarity re-weighting

Comparison with Existing Methods on AGW

ADGR achieves **a better accuracy-speed balance** than existing re-ranking methods.

Base	Method	Venue	Market mAP (%)	Market Rank1 (%)	Market Time (s)	Duke mAP (%)	Duke Rank1 (%)	Duke Time (s)	CUHK03 mAP (%)	CUHK03 Rank1 (%)	CUHK03 Time (s)
AGW	-	TPAMI21	88.29	95.31	-	79.60	89.27	-	62.16	64.43	-
	QE	ICCV07	91.68	95.55	0.9	84.92	91.25	0.8	69.32	65.36	0.6
	KR	CVPR17	94.71	95.93	98.4	90.25	91.74	100.4	75.81	71.93	33.0
	ECN	CVPR18	94.85	96.26	78.9	90.60	92.55	79.2	75.97	72.79	9.9
	LBR	ICCV19	91.78	95.96	0.8	84.58	91.70	0.7	68.71	67.64	0.6
	GCR	TMM23	95.22	96.79	6.9	90.11	92.73	7.0	82.73	82.29	2.2
	Cheb	CVPR25	92.94	95.84	1.2	86.57	91.83	1.3	70.71	68.86	0.6
	Ours	-	94.25	96.05	1.3	89.37	91.88	1.4	72.23	70.43	0.6

AGW: Processing time VS mAP improvement

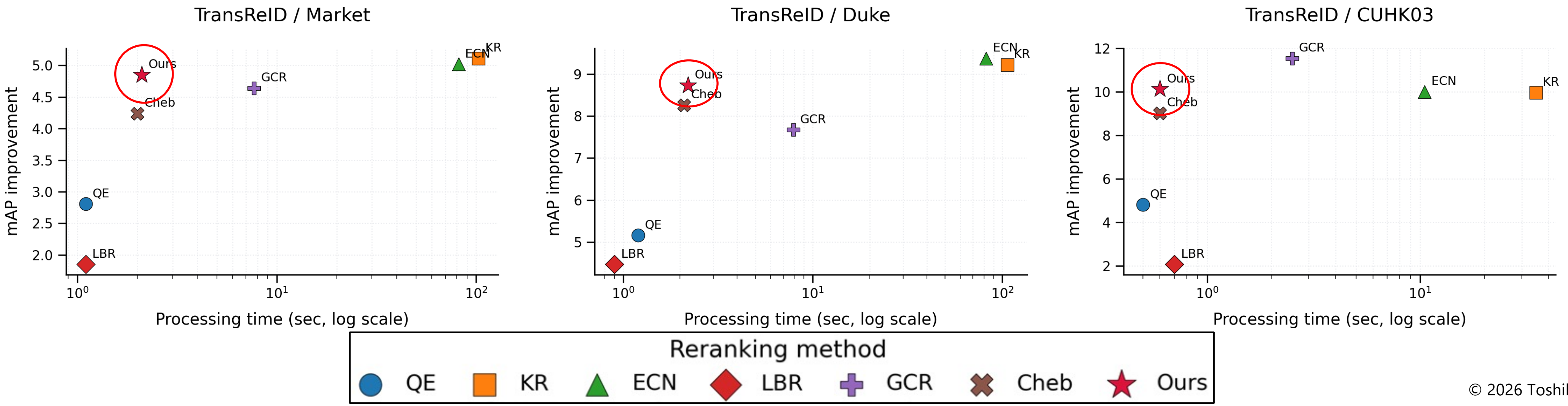


Comparison with Existing Methods on TransReID

ADGR achieves **a better accuracy-speed balance** than existing re-ranking methods.

Base	Method	Venue	Market mAP (%)	Market Rank1 (%)	Market Time (s)	Duke mAP (%)	Duke Rank1 (%)	Duke Time (s)	CUHK03 mAP (%)	CUHK03 Rank1 (%)	CUHK03 Time (s)
TransReID	-	ICCV21	88.99	95.22	-	82.11	90.71	-	78.08	79.64	-
	QE	ICCV07	91.80	95.40	1.1	87.27	92.73	1.2	82.89	81.29	0.5
	KR	CVPR17	94.10	95.58	102.8	91.33	92.68	106.3	88.04	86.36	35.0
	ECN	CVPR18	94.01	95.61	81.6	91.48	93.13	82.3	88.08	86.86	10.5
	LBR	ICCV19	90.84	94.27	1.1	86.58	92.37	0.9	80.14	79.21	0.7
	GCR	TMM23	93.63	95.67	7.7	89.78	92.24	7.9	89.61	89.43	2.5
	Cheb	CVPR25	93.23	95.22	2.0	90.37	92.73	2.1	87.10	85.57	0.6
	Ours	-	93.84	95.75	2.1	90.84	92.46	2.2	88.20	86.21	0.6

TransReID: Processing time VS mAP improvement



Effectiveness of adaptive threshold control and similarity re-weighting

Both components contribute to performance improvement, and their combination achieves the best overall results.

λ_i: Adaptive threshold control

For adaptive threshold control

a_i: Similarity re-weighting

Similarity re-weighting

w/o λ_i, w/o a_i : Neither adjustment is applied (**Cheb-GR**).

w/ λ_i, w/o a_i : Only adaptive threshold control is applied.

w/o λ_i, w/ a_i : Only similarity re-weighting is applied.

w/ λ_i, w/ a_i : Both are applied (**ADGR**).

Base	λ_i	a_i	Market mAP (%)	Market Rank1 (%)	Duke mAP (%)	Duke Rank1 (%)	CUHK03 mAP (%)	CUHK03 Rank1 (%)
AGW			92.94	95.84	86.57	91.83	70.71	68.86
	✓		93.01	**96.05**	86.67	**91.97**	70.80	69.00
		✓	94.06	95.96	88.92	91.56	72.17	70.36
	✓	✓	**94.25**	**96.05**	**89.37**	91.88	**72.23**	**70.43**

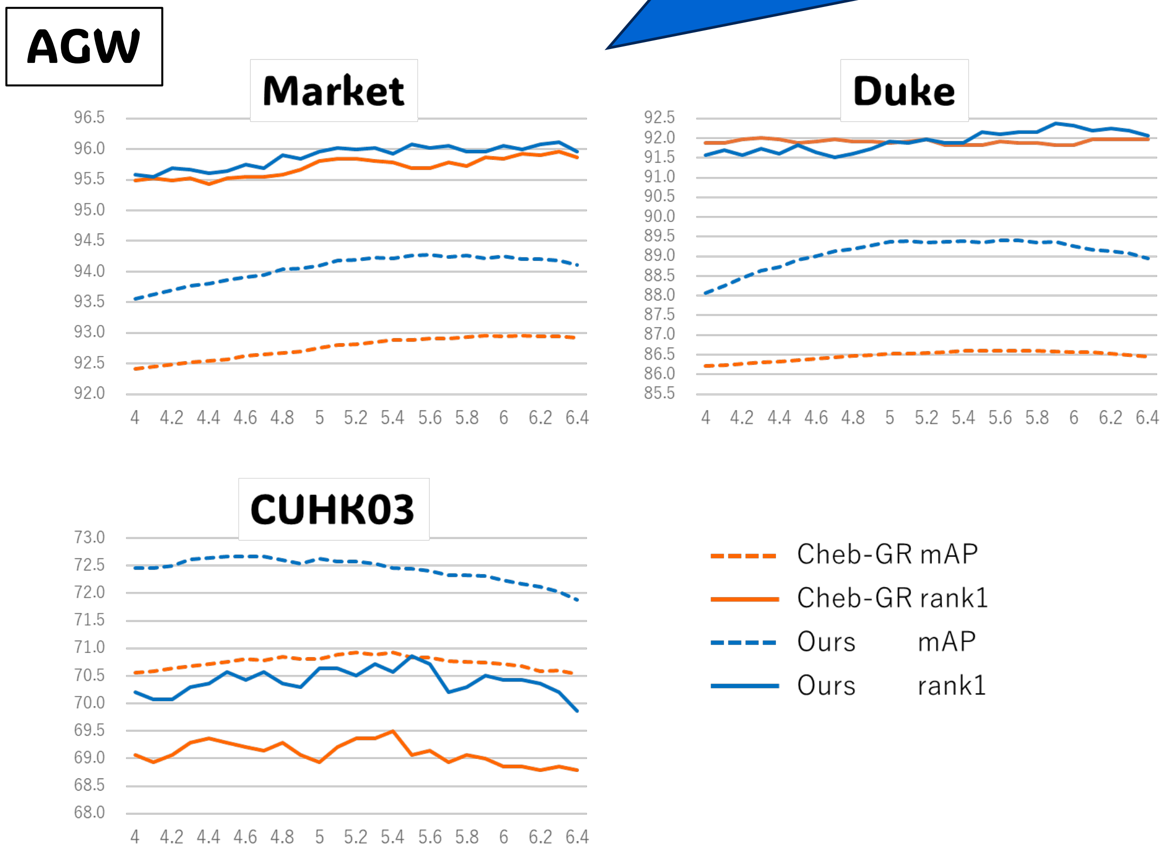
- Both components contribute to **performance improvement**.
- Similarity re-weighting has **a strong impact on mAP**.
- Full ADGR achieves the best overall results.

Parameter Analysis

ADGR maintains strong performance over practical parameter ranges.

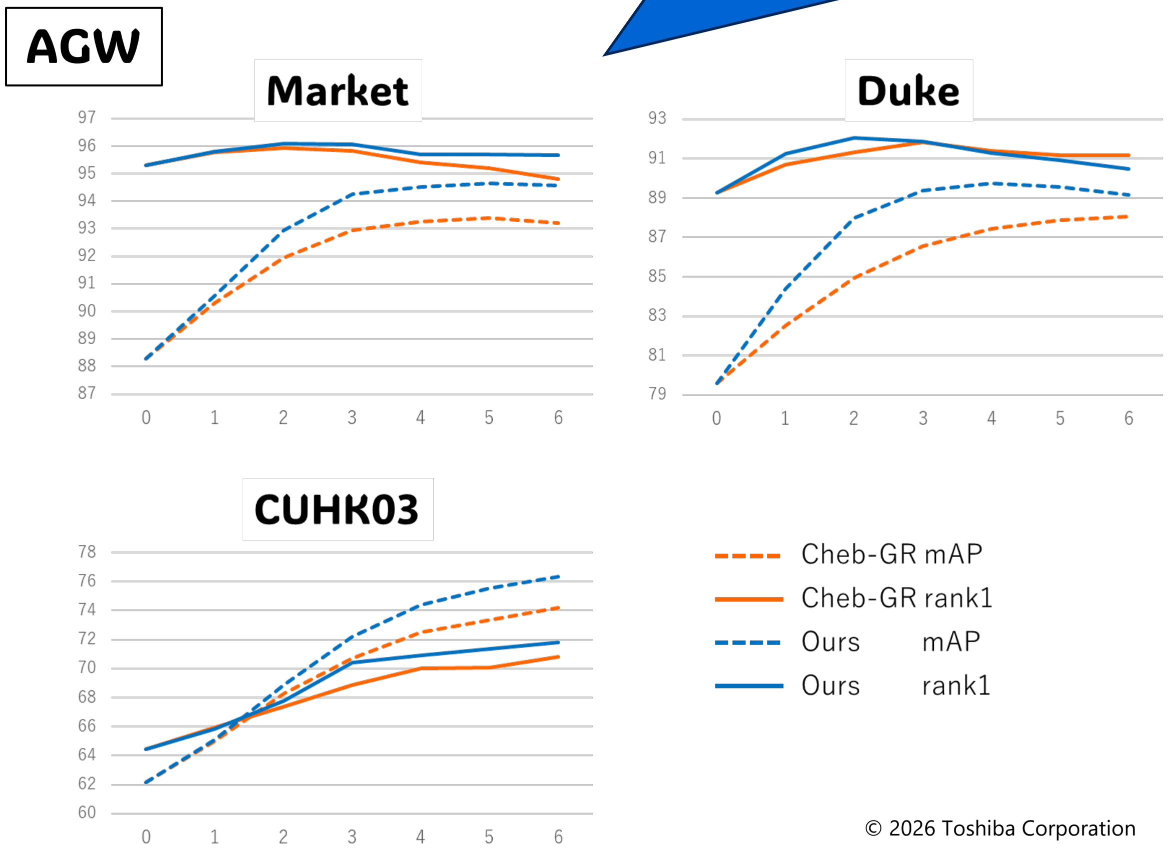
Sensitivity to λ λ controls the **initial threshold**.

ADGR generally maintains strong mAP performance across practical λ values.



Sensitivity to T **T** controls the number of **feature propagation iterations**.

ADGR achieves strong performance with only a **few propagation steps**.



Conclusion

ADGR enables fast, stable, and accurate graph-based re-ranking.

Key Contributions

1. **ADGR Framework**
   - **Adaptive threshold control**
     Stabilizes neighbor selection without k-NN sorting.
   - **Density-aware similarity re-weighting**
     Balances feature propagation according to local density.

2. **Extensive Validation**
   Validated on multiple feature extraction models and benchmark datasets, demonstrating effectiveness and robustness.

3. **Accuracy and Efficiency**
   Achieves higher accuracy than conventional statistical methods while reducing computational cost compared with k-NN-based approaches.

TOSHIBA

Thank you for your attention.

Author's email address :
yushiro.kashimoto.h87@mail.toshiba

Thank you for your attention.

Author's email address :
yushiro.kashimoto.h87@mail.toshiba

Thank you for your attention.

Author's email address :
yushiro.kashimoto.h87@mail.toshiba

TOSHIBA

Thank you for your attention.

Author's email address :
yushiro.kashimoto.h87@mail.toshiba