Beyond Tie Points: Satellite Image Block Adjustment based on Dense Feature Consistency

Supplementary Material

In this supplementary material, we provide the following information to support the main paper:

1. Sec. A provides additional details of pretraining feature extractor, including the purpose of proxy task training, 3D coordinate and confidence GT generation, decoder architecture, positive/negative sample generation, and training strategy.
2. Sec. B details the adjustment parameter refinement, including time cost and loss convergence.
3. Sec. C provides comprehensive details regarding the Manual Check Points (MCPs), encompassing the acquisition methodology, spatial distribution overview, and residual statistics.
4. Sec. D provides additional experiments and discussions.

A. Pretraining Feature Extractor

Why Use Proxy Task Training. This approach is motivated by the fact that our adjustment parameter solver (Sec. 3.3) involves thousands of iterations rather than a straightforward feed-forward pass. Consequently, the corresponding computational graph is dynamic and dependent on the convergence state, rendering the process essentially non-differentiable. Forcing an end-to-end approach would result in highly unstable gradient signals, thereby impeding the effective training of the feature extractor. Furthermore, Planar Block Adjustment (PBA) optimization requires the simultaneous processing of all images within a region. In an end-to-end setting, backpropagating through a computational graph that encompasses thousands of iterations over the entire dataset would be computationally prohibitive in terms of both time and memory. To address these challenges, we decouple the optimization problem. Specifically, we pre-train the feature extractor via a tailored proxy task, priming the feature space for subsequent geometric optimization. Subsequently, the feature extractor is frozen, and global parameter optimization is performed exclusively through iterative solving.

We adopt geographic coordinate regression as the specific proxy task for training the feature extractor. Our adjustment methodology hinges on the principle of feature consistency. This is predicated on the assumption that pixels across multiple images corresponding to the same physical location (i.e., homologous points) yield feature representations that are not only highly similar but ideally the most similar to each other. Consequently, the alignment of feature maps intrinsically implies the alignment of the images themselves. To satisfy this assumption without resorting to end-to-end training, we leverage the geographic coordinate regression task. In this framework, extracted features are mapped to their corresponding geographic coordinates via a simple Multilayer Perceptron (MLP). This imposes two critical constraints on the feature extractor: (1) to accurately regress the coordinates of each pixel, the extracted features must exhibit strong spatial discriminability; and (2) to regress homologous points from different images to the same coordinate, their corresponding features must be highly consistent. These requirements align perfectly with the core assumptions underpinning our adjustment framework.

Dataset Details. We train the feature extractor for the geographic coordinate regression task using a dataset comprising 221 sets of multiview imagery. Derived from satellite sources including GF-3, GF-7, SV-2, and WV-2, this dataset encompasses diverse terrain types such as forests, mountains, urban areas, water bodies, and farmlands. Within each group, images are pre-processed to ensure uniform dimensions and pixel-wise alignment. Furthermore, each dataset entry includes ground truth annotations for 3D coordinates and confidence maps. We conduct training across all groups concurrently; in each epoch, the model iterates through the entire collection of data groups. During the traversal of each group, we randomly select a pair of images. Subsequently, we extract cropping windows from both images that share an overlapping region but vary in randomized position, scale, and orientation. These crops are then resampled to a resolution of 1024×1024. Leveraging this random cropping strategy ensures that the training samples vary with every iteration. This effectively simulates training on a dataset exceeding one million image pairs, thereby substantially enhancing the model's performance.

3D Coordinates GT Generation. As indicated by Eq. 1 in the main paper, deriving the 2D geographic coordinates of a pixel via Rational Polynomial Coefficients (RPC) requires an input elevation value; conversely, obtaining this elevation value necessitates sampling elevation data using those very 2D coordinates. To resolve this cyclic dependency, we employ an iterative refinement strategy to generate pixel-wise 3D coordinate ground truth across the entire image. Prior to this iterative process, we perform high-precision geometric correction on the imagery utilizing Ground Control Points (GCPs). To initialize the 3D coordinate computation, the elevation of each pixel is set to

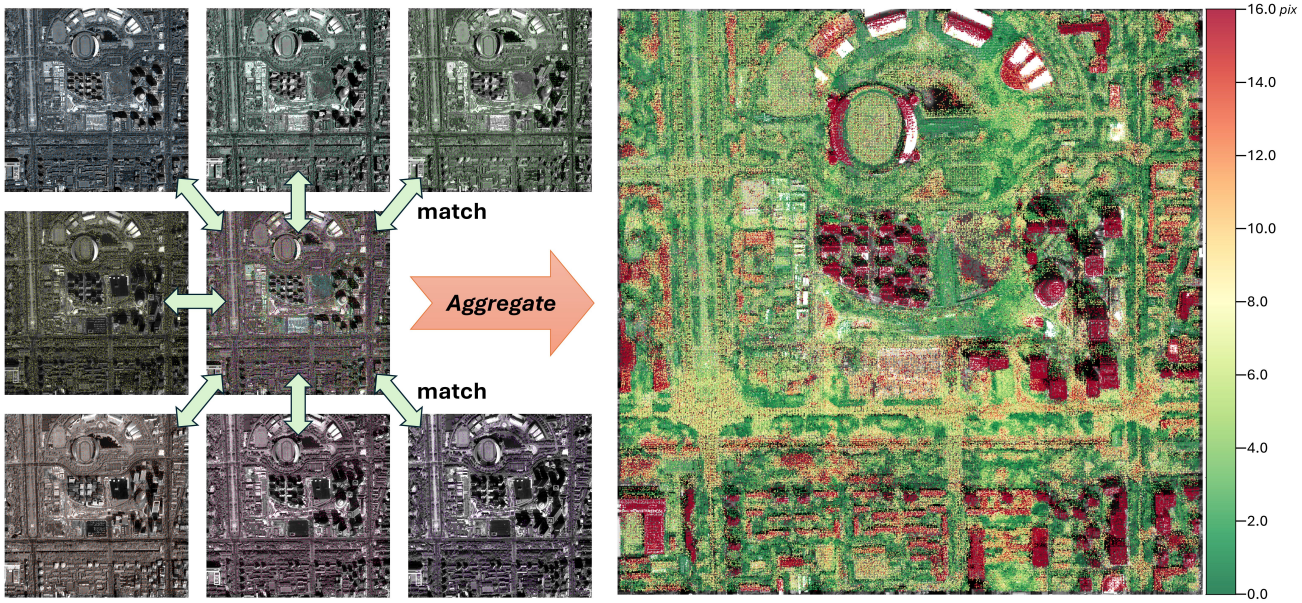

Figure 1. **Parallax Map Generation.** To generate a comprehensive parallax map for each image in a group, parallax is first computed via exhaustive pairwise matching against all other images and then aggregated.

the average elevation of the covered area. In each subsequent iteration, we calculate 2D geographic coordinates using RPC based on the current elevation estimates, and then sample the elevation data at these coordinates to update the elevation values. The iterative process terminates when the convergence criterion is met: specifically, when fewer than 1% of pixels exhibit an elevation change exceeding 0.1 m. This methodology yields precise, pixel-wise ground truth for 3D coordinates.

Confidence GT Generation. We utilize the parallax of ground features as the reference basis for training confidence estimation. As detailed before, each training data group is pre-processed to ensure identical spatial extent and pixel-wise alignment. In this aligned setup, ground points with minimal parallax appear well-registered, whereas those with significant parallax exhibit substantial coordinate discrepancies across different views. Leveraging this property, we employ a stereo matching algorithm to generate a parallax map for each image. The matching algorithm serves a dual purpose: it identifies features with high geometric parallax and detects unstable, variable regions that are inherently difficult to match, such as clouds and water bodies. Within each group, we perform pairwise matching between every image and all other images using the CasP [2] algorithm. For each matched pair, the Euclidean distance between the pixel coordinates in the two images is calculated to represent the parallax value. If a single point yields varying parallax values across different pairwise comparisons, the median value is adopted. The calculation process of the parallax map is shown in Fig. 1.

Since the feature extractor involves a $16\times$ downsampling of the input images, we apply a corresponding downsampling operation to the parallax maps. Specifically, we apply a sliding window with a kernel size of 16×16 and a stride of 16. Within each window, the median of all computed parallax values is calculated. The absence of valid parallax measurements within a window indicates the presence of unstable or intractable features. In such cases, the window is assigned the maximum possible parallax value. Finally, the median value of the aggregated parallax map is mapped to a confidence score of 0.5. Fig. 2 shows the examples of confidence ground truth and the equation below provides the exact transformation from parallax to the final confidence ground truth,

$$C_{gt} = \max\left(0, \min\left(1, \frac{2 \cdot \text{median}(P) - \min(P) - P}{2 \cdot (\text{median}(P) - \min(P))}\right)\right) \tag{1}$$

where C_{gt} represents the confidence ground truth, P represents the calculated parallax, $min(\cdot)$, $max(\cdot)$ and $median(\cdot)$ represent the minimum, maximum and median value of the given sequence respectively.

Decoder Architecture. The decoder employs a streamlined MLP architecture. Specifically, it is composed of a hidden processing stage and a final output stage. The hidden stage consists of two stacked residual blocks featuring skip connections [5]. Each block comprises a sequence of three 1×1 convolutional layers with channel dimensions of $[512, 1024, 512]$. The output stage incorporates two ded-

icated heads, one for geographic coordinates and one for elevation, collectively generating the final 3D coordinate predictions.

Positive and Negative Sample Generation. During the random cropping process for a pair of training images, we explicitly calculate the spatial transformation matrix describing the geometric relationship between the two crops. Given that the underlying source imagery is pixel-aligned, we utilize this transformation matrix to map pixel coordinates from one crop to the other, thereby identifying precise pixel-wise correspondences within the overlapping regions. Leveraging these correspondences, we sample the respective feature maps to construct positive pairs for contrastive training. Conversely, to generate negative pairs, we apply random offsets to these correspondences, intentionally inducing misalignment between the feature points.

Training Strategy. The model is trained for a total of 5000 epochs. We employ a customized learning rate scheduling strategy throughout the training process. Initially, we implement a warm-up phase of 500 epochs, during which the learning rates for the encoder and decoder are linearly increased to their respective peak values of 1×10^{-4} and 1×10^{-3}. Following this, the learning rates are maintained at these peak levels for a subsequent 1500 epochs. For the final 3000 epochs, a cosine annealing schedule is applied to gradually decay the learning rates to zero.

B. Refining Adjustment Parameters

Time Cost. Tab. 1 presents the computation time required for parameter adjustment across the various datasets. Notably, during the first hierarchical level, the early stopping mechanism was triggered at the 645th and 574th iterations for the Beijing-b and Guangzhou-b datasets, respectively. For the Guangzhou-b dataset, the first hierarchical level is partitioned into only 3 grid cells due to spatial extent constraints, whereas all other datasets are initialized with 8 grid cells. Attributable to our quadtree-based coarse-to-fine optimization strategy, the computation time exhibits an approximate $4\times$ increase as the hierarchy progresses from lower to higher levels. Additionally, the computation time scales quadratically with the number of images included in the adjustment.

Loss Convergence. Fig. 3 illustrates the loss convergence curves observed during the parameter adjustment process for each dataset. The loss values are scaled by a factor of 1×10^4. Across all hierarchical levels, the loss demonstrates a steady and consistent descent, indicating stable convergence.

Table 1. **Computation Time for Refining Adjustment Parameters** across different datasets.

Datasets	Time (s)		
	Level-1	Level-2	Level-3
Beijing-a	223.36	917.28	/
Beijing-b	146.03†	913.03	3671.04
Guangzhou-a	36.04	146.25	/
Guangzhou-b*	18.73†	69.62	209.89
San Jose-a	36.05	143.59	/
San Jose-b	34.91	147.18	561.38

† Early stopped.
* Initialized with 3 grid cells, 8 grid cells for other datasets.

C. Manual Check Points

We acquire Manual Check Points (MCPs) through a rigorous manual annotation process. Initially, we employ ArcGIS Pro to pinpoint MCPs on multiview imagery that has undergone initial geometric correction. Throughout this process, we adhere to three primary criteria: (1) Points must be located on ground features that exhibit negligible parallax; (2) The selected locations must possess distinct visual features across all images to ensure precise correspondence; and (3) The points should be uniformly distributed throughout the overlapping regions. Subsequently, we fine-tune the position of each MCP on every image to ensure precise alignment with the homologous ground features. Fig. 9 depicts the spatial distribution of MCPs for each dataset. Fig. 5 illustrates the locations of a single set of MCPs across multiview imagery, while Figs. 6, 7, and 8 provide overviews of the MCPs within each dataset . Finally, Tab. 2 presents the residual statistics for the MCPs in each dataset.

Table 2. Residual statistics of MCPs across various datasets.

Datasets	Residual (m)			
	Mean	Median	@1m	@3m
Beijing	0.59	0.52	86.56	100
Guangzhou	0.78	0.62	75.00	100
San Jose	0.32	0.30	100	100

D. Additional Experiments

D.1. Adjustment on Real-World Dataset

As mentioned in Sec. 4.1, we construct the evaluation benchmarks by injecting artificial perturbations into datasets that have undergone high-precision geometric registration. These artificial perturbations primarily serve to controllably standardize the initialization error magnitudes across diverse datasets, thereby enabling a systematic evaluation of the adjustment methods in terms of precision and robustness, rather than replacing real-world dataset. Since

the perturbations are strictly applied to the RPCs while preserving the original image content, they can be considered a first-order error approximation that faithfully simulates the vast majority of real-world RPC inaccuracies. Furthermore, we provide supplementary comparative results utilizing the raw RPCs (without perturbations) and report the comprehensive runtime of each baseline under identical hardware and a unified implementation. The results presented in Tab. 3 demonstrate that the model's adjustment performance on real-world data aligns consistently with that on the simulated datasets.

Table 3. **Performance of PBA on real-world datasets.**

Methods	Beijing		Guangzhou		San Jose	
	med (m)	time (s)	med (m)	time (s)	med (m)	time (s)
SIFT [6]	6.28	1317.45	4.84	196.18	0.99	208.03
SP [3]+SG [7]	7.46	**600.78**	10.13	**76.26**	0.79	**135.15**
LoFTR [9]	5.73	617.96	4.37	79.86	0.85	148.01
ASF [1]	6.63	1605.82	5.33	226.14	0.85	392.04
RoMa [4]	7.54	2919.40	11.80	448.11	0.78	613.02
ours	**2.15**	1105.17	**1.31**	181.75	**0.52**	180.61

D.2. Feature Consistency and Discriminability

In Sec. 4.5, we validated and qualitatively visualized the consistency and discriminability of the dense features across diverse viewpoints, regions, and geometric transformations. Here, we provide a supplementary quantitative evaluation by measuring the feature similarity separability between corresponding and non-corresponding points in Tab. 4. For comparison, we also report this metric using features extracted solely by the frozen DINOv3 [8] backbone.

Table 4. Separability of feature similarities for matched vs. non-matched points measured by AUC in cross-domain scenarios.

Domains	Ours	DINOv3 [8]
Illumination	0.9798	0.6149
Season	0.9908	0.8225
Sensor	0.9717	0.9454
Viewpoint	0.9988	0.9795
Geometry	0.9995	0.8631

D.3. Corner Cases

As a deep learning-based alternative to classical geometric algorithms, our approach inevitably encounters failures in certain corner cases. Nevertheless, owing to the inherent robustness of our model demonstrated in Sec. D.2, the proposed approach successfully overcomes the majority of these challenging scenarios. The remaining failure cases are primarily attributed to extreme temporal variations, severe textureless regions, or repetitive patterns. These scenarios typically lack distinct correspondence cues, thereby introducing intrinsic feature ambiguity. Fig. 4 illustrates the performance of our method across several such corner cases.

References

[1] Hongkai Chen, Zixin Luo, Lei Zhou, Yurun Tian, Mingmin Zhen, Tian Fang, David Mckinnon, Yanghai Tsin, and Long Quan. Aspanformer: Detector-free image matching with adaptive span transformer. In *European conference on computer vision*, pages 20–36. Springer, 2022. 4

[2] Peiqi Chen, Lei Yu, Yi Wan, Yingying Pei, Xinyi Liu, Yongxiang Yao, Yingying Zhang, Lixiang Ru, Liheng Zhong, et al. Casp: Improving semi-dense feature matching pipeline leveraging cascaded correspondence priors for guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 28063–28072, 2025. 2

[3] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018. 4

[4] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. Roma: Robust dense feature matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19790–19800, 2024. 4

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[6] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 4

[7] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 4

[8] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025. 4

[9] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021. 4

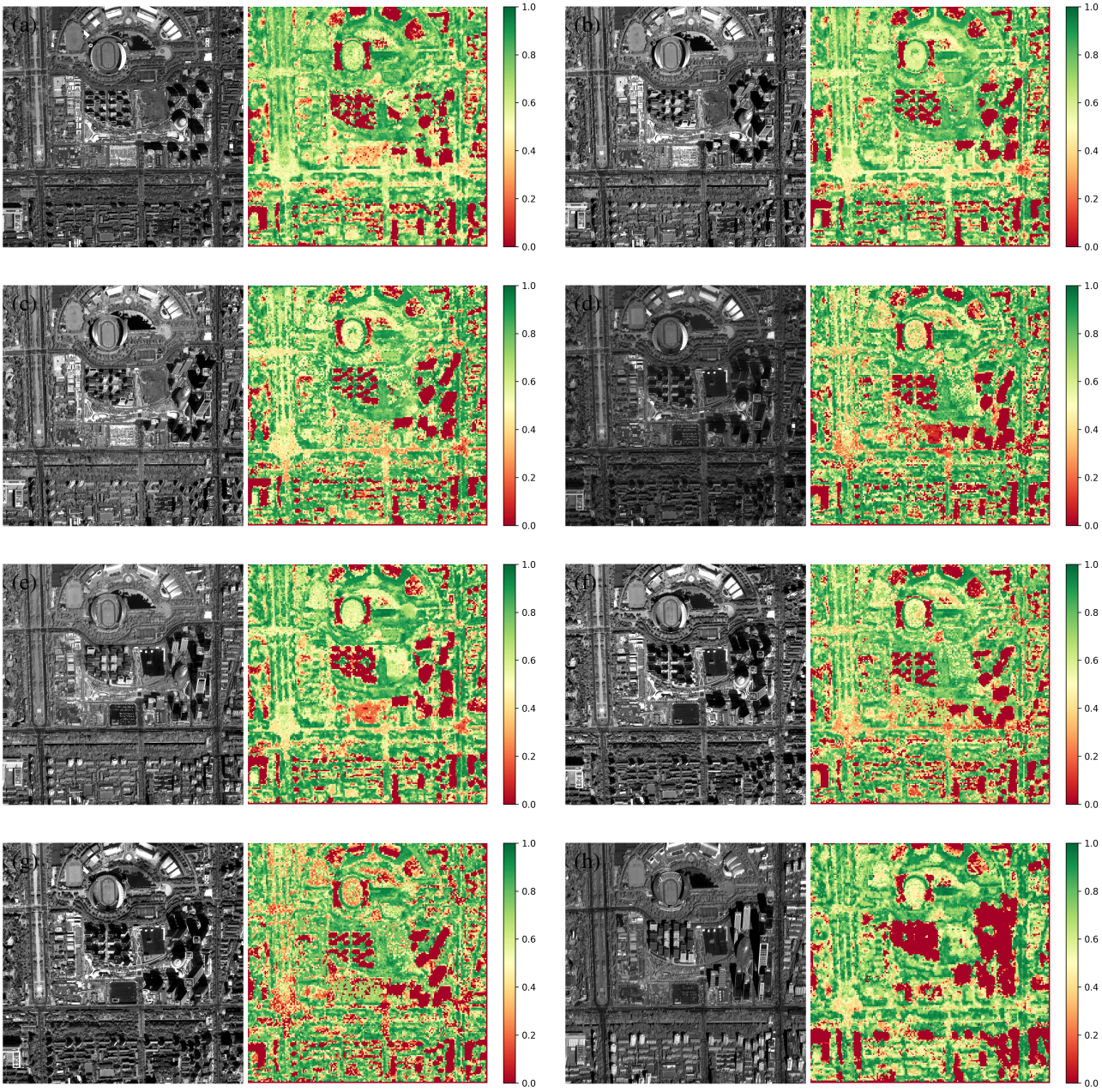

Figure 2. Examples of confidence ground truth derived from parallax maps across varying viewpoints.

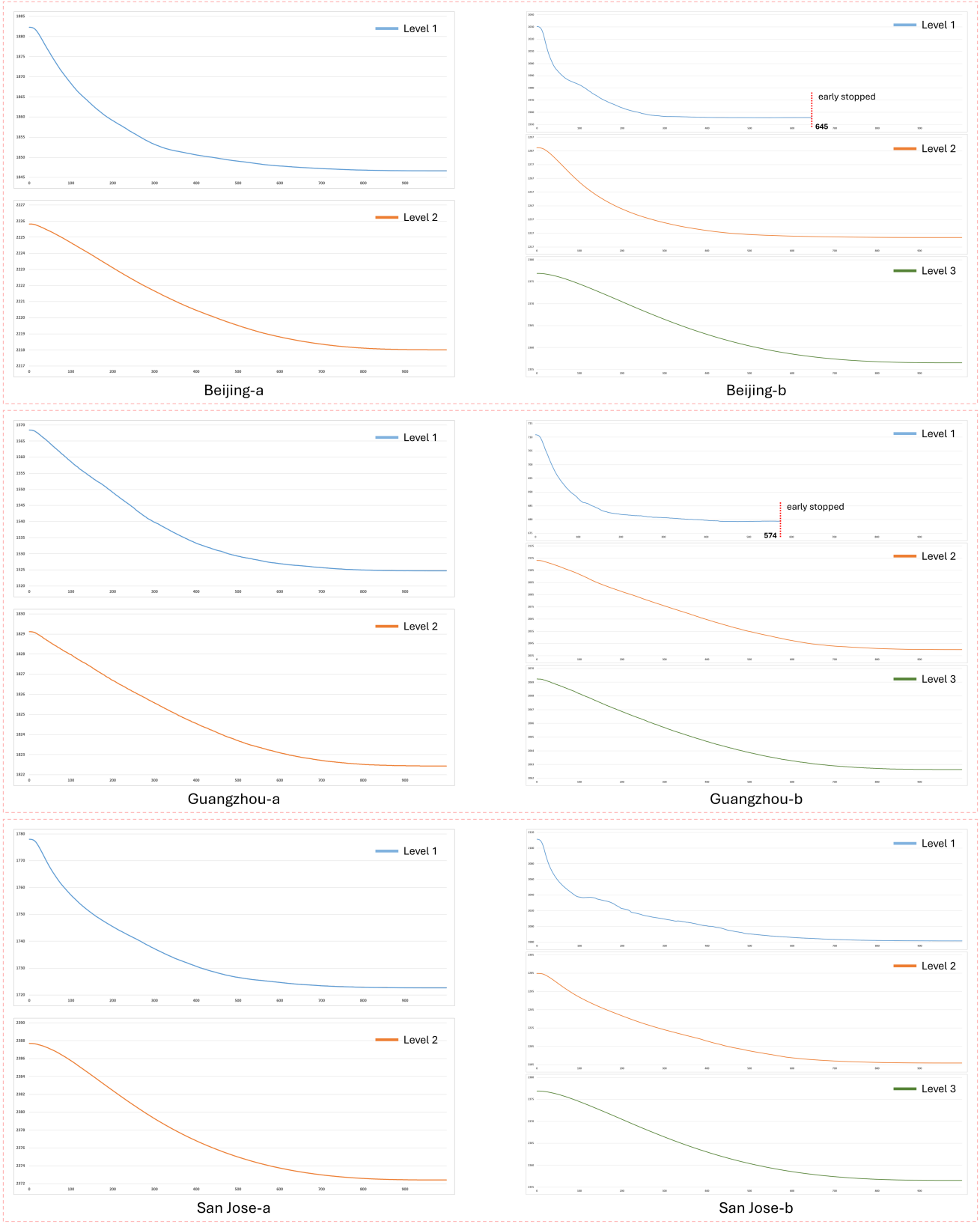

Figure 3. Loss convergence curves during adjustment parameter optimization across various datasets.

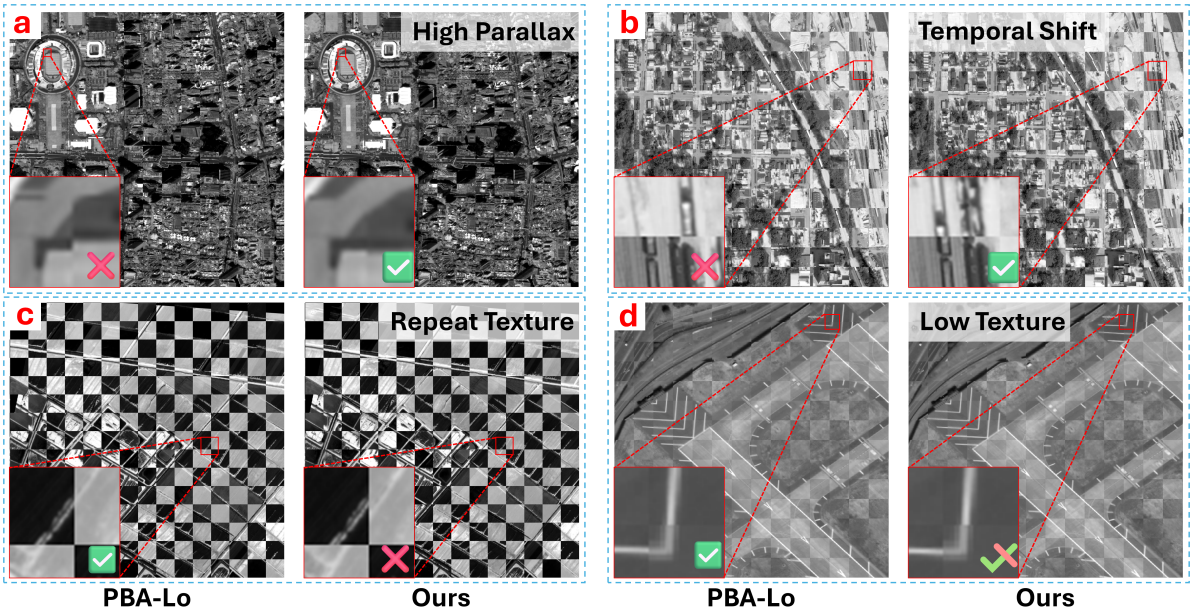

Figure 4. Performance of our method and PBA-Lo in corner cases.

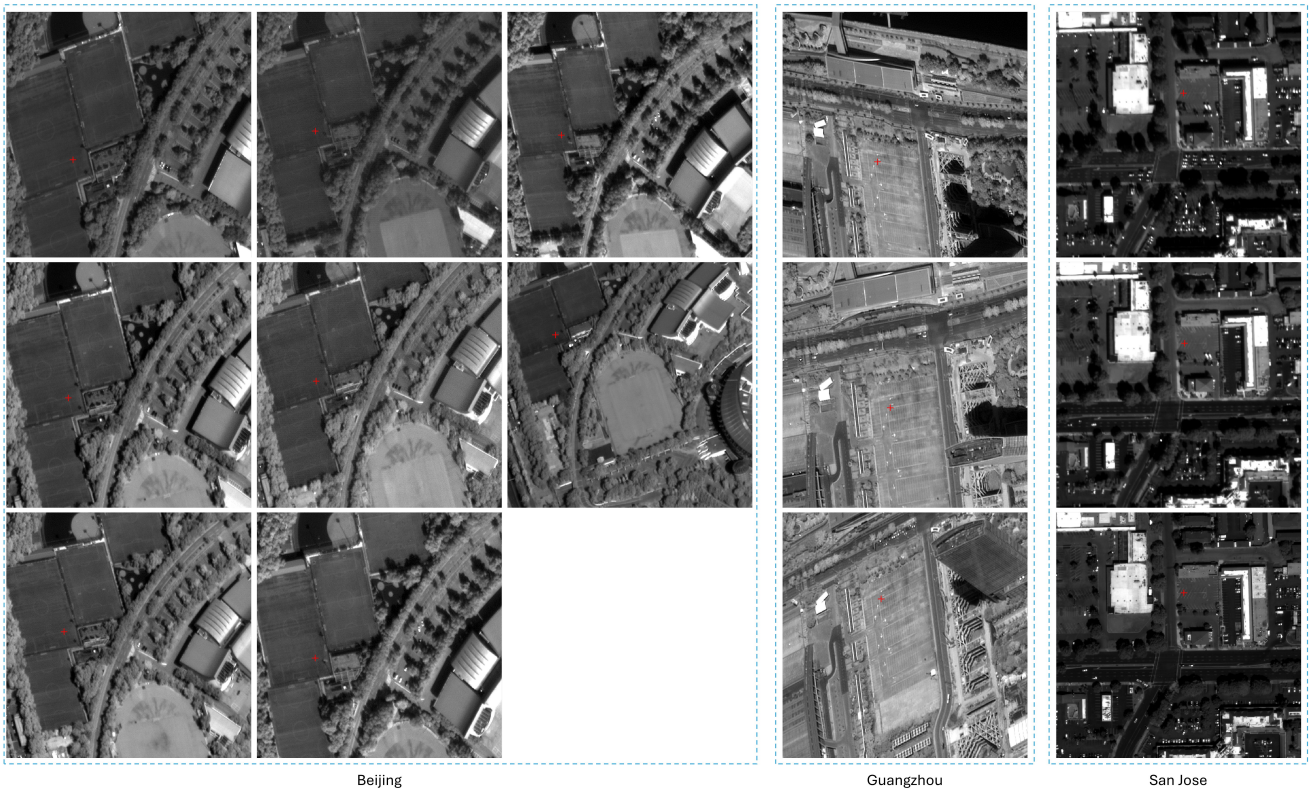

Figure 5. Examples of homologous MCPs across multiview imagery from various datasets.

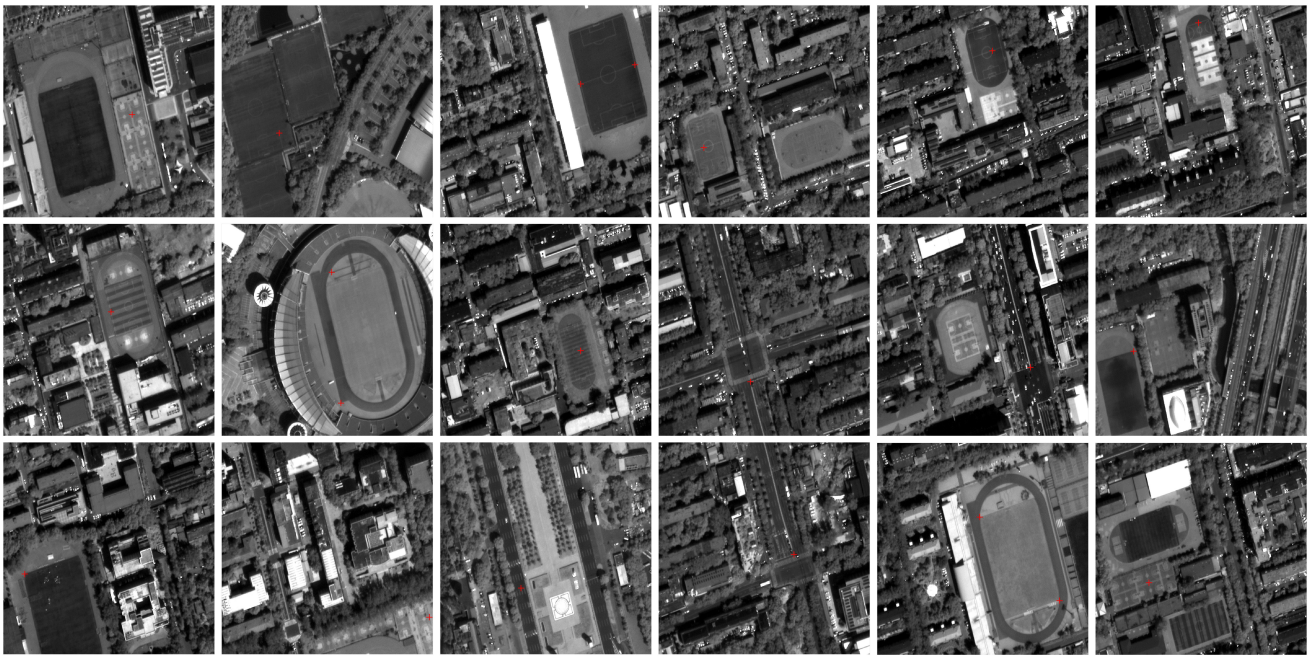

Figure 6. Overview of MCPs in Beijing Dataset.

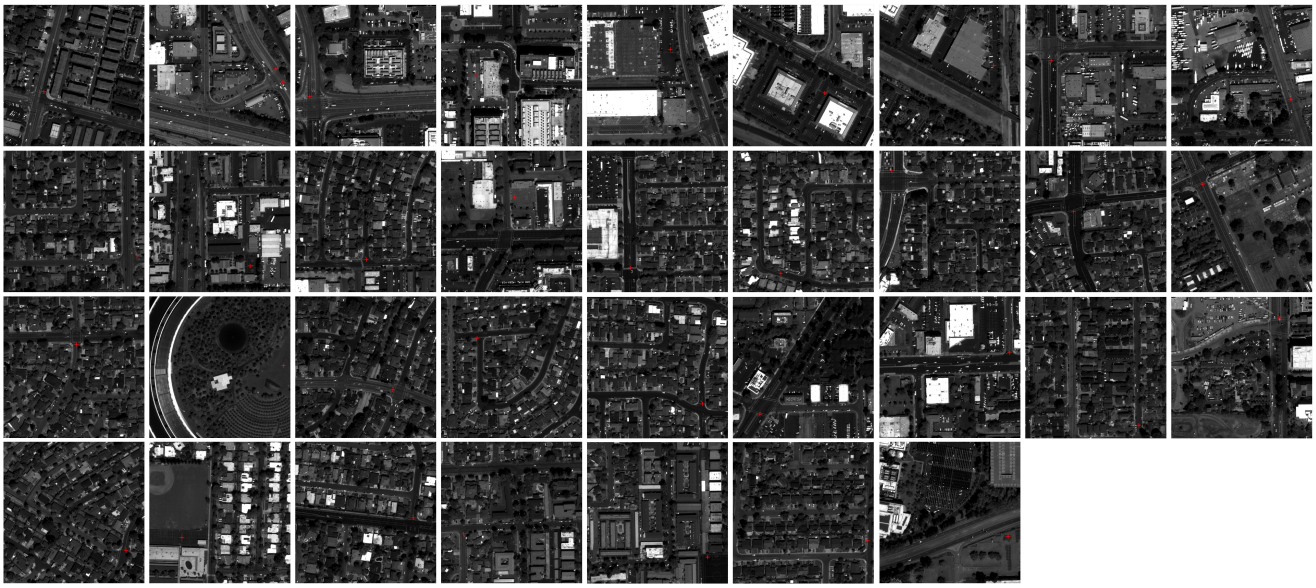

Figure 7. Overview of MCPs in San Jose Dataset.

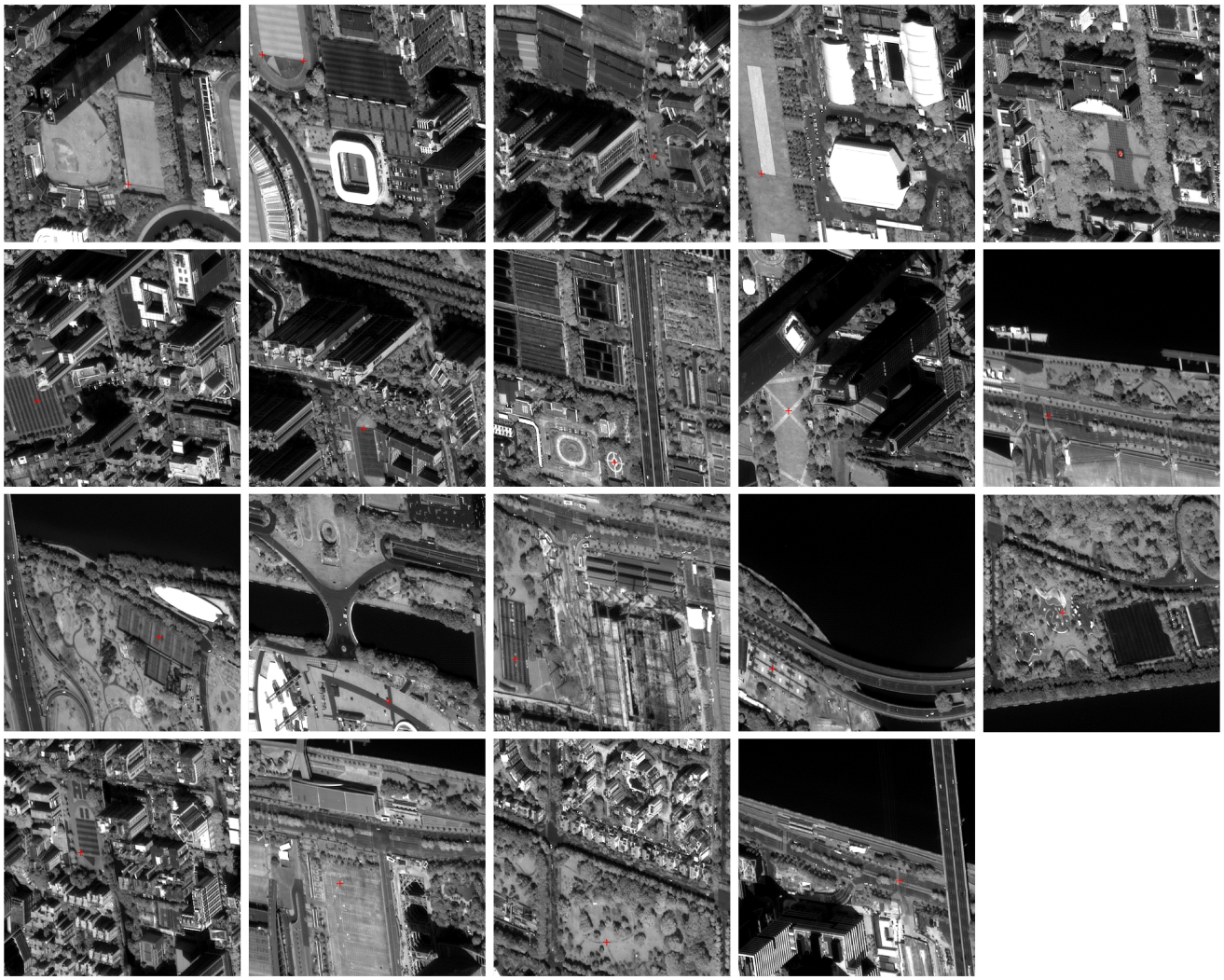

Figure 8. Overview of MCPs in Guangzhou Dataset.

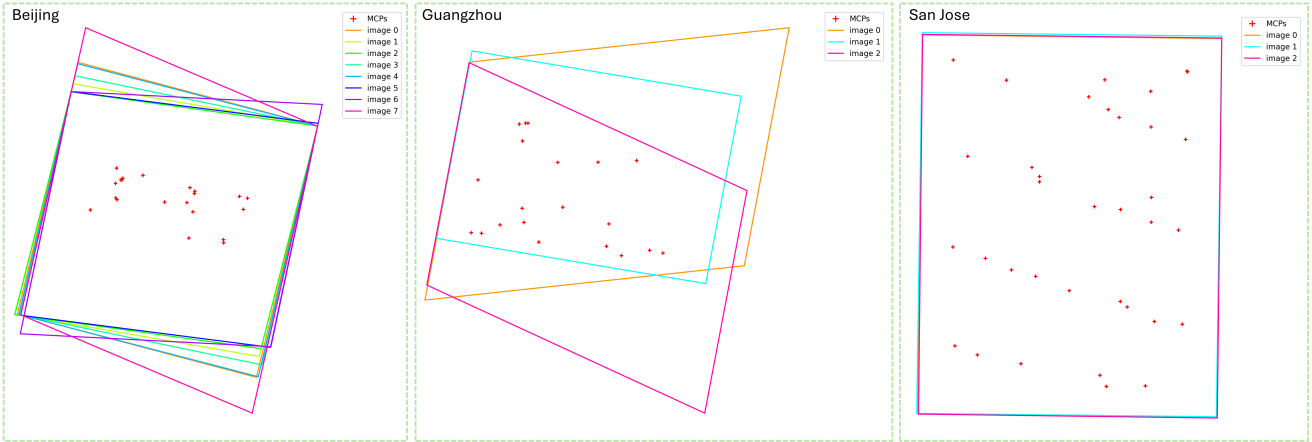

Figure 9. Spatial distribution of MCPs across various datasets.