# HypeVPR: Exploring Hyperbolic Space for Perspective to Equirectangular Visual Place Recognition

Supplementary Material

## S1. Contents

This supplementary material includes the following sections:



## S2. More details on hierarchical modeling of equirectangular image

This section provides additional discussion and examples regarding the hierarchical modeling strategy introduced in Section 4.2 of the main paper.

Given query image resized to $H \times W$ and a panorama resized to $W \times 8W$, the top level corresponds to the full image. Each subsequent level halves the horizontal extent of the previous one:

$$I_d^{(\ell)} \in \mathbb{R}^{H \times \frac{W'}{2^{\ell-1}} \times C},$$

as described in Eq. (6) of the main paper and illustrated in Fig. S1.

For example, when $L = 4$, the lowest level consists of eight $W \times W$ windows, matching the query resolution. These windows can be processed by the same backbone used for query encoding, ensuring architectural consistency and balanced representational capacity across levels. According to Eq. (6) of the main paper, using $L = 5$ would nominally produce windows with width $W/2$. However, since our model shares the backbone between queries and database windows, we keep the window width fixed at $W$, introducing an overlap of $W/2$ between adjacent windows. For the last window that would partially exceed the image boundary, we adopt the strategy of [12], where the left portion of the first window is used to fill the missing region, as illustrated in Fig. S1. This results in sixteen lowest-level windows while preserving compatibility with the shared backbone.

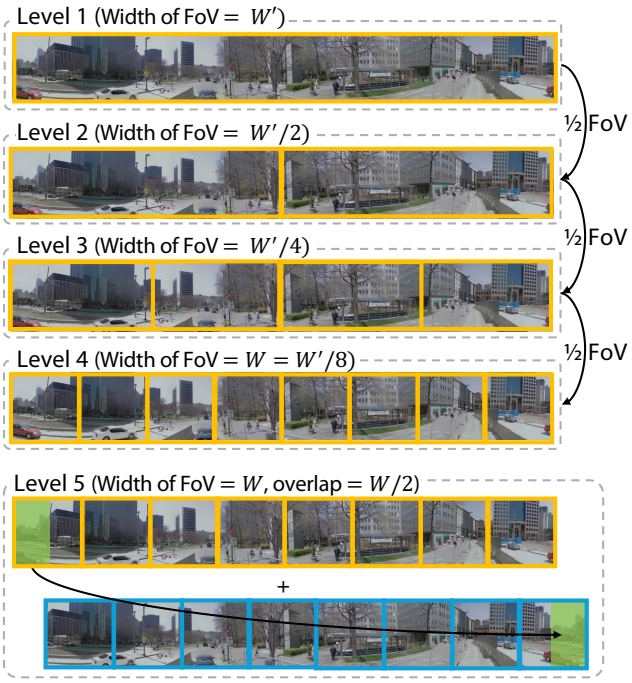


Figure S1. Hierarchical decomposition of an equirectangular image across five levels. At Levels 1–4, the horizontal FoV is halved at each level, forming a coarse-to-fine hierarchy. The final level (Level 5) uses a fixed FoV of $W \times W$ to match the query resolution, allowing all lowest-level segments to be encoded with the same backbone as the perspective query.

To handle these overlapping windows during training and to construct a valid hierarchical structure, we apply a sub-tree partitioning strategy, as described in Section S3.

## S3. Sub-tree Partitioning for Overlapped Windows

As described in Section S2, when $L = 5$, we intentionally introduce a $W/2$ overlap between adjacent windows for practical reasons, such as matching the spatial resolution of queries and database images so that both can be processed with the same backbone (Fig. S1). This overlap, however, leads to incorrect negative relationships at the same level in the $\mathcal{L}_{\text{hier}}$ component. To address this issue, we adopt a simple yet effective sub-tree partitioning strategy.

When $L = 5$, we group windows that share the same remainder when their index is divided by 2. This yields two complete, non-overlapping panoramic window sequences.

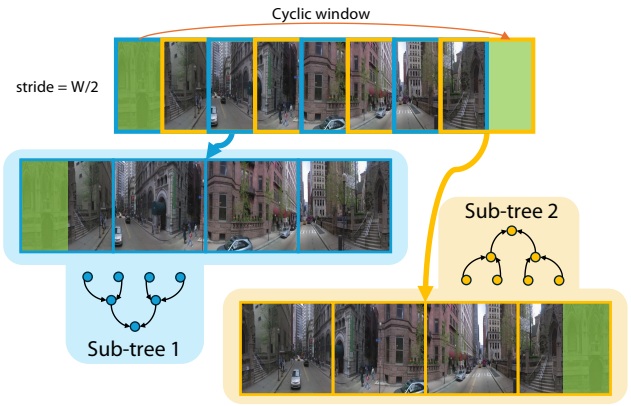


Figure S2. Sub-tree partitioning strategy used when windows have an overlap of $W/2$. Due to this overlap, windows are divided into two groups based on whether their indices are odd or even, resulting in two complete and non-overlapping panoramic sequences. Each sequence is used to construct an independent sub-tree, and together they resolve the negative-pair ambiguity caused by the overlapping windows.

For each sequence, we independently construct a sub-tree and build a hierarchical structure following the procedure described in Section 4.2 of the main paper, resulting in two binary trees with four levels each. The top-level descriptors from these two trees are then fused using the hyperbolic aggregation operator $\mathcal{A}_{\text{hyp}}$, producing the final representative descriptor for the image.

Consequently, the overall structure begins with two nodes at Level 2 and forms a binary hierarchy from that level onward. This approach overcomes the limitations imposed by fixed window sizes and retains fine-grained information from smaller windows, enabling effective use of our proposed loss functions. Figure S2 shows an example in which one sub-tree is constructed from windows with odd indices and the other from windows with even indices.

## S4. Effect of sub-tree partitioning

With overlapping 16 windows, we apply the sub-tree partitioning to handle the overlaps between windows. To demonstrate the effectiveness of this technique, we trained a model with the same setup but constructed a 5-level tree using the 16 overlapping windows in their original sequence. As shown in Tab. S1, the network without the technique performs significantly worse. This is because the network fails to learn proper representations due to the negative relationships among overlapping windows enforced by $\mathcal{L}_{\text{hier}}$. Note that the reported values correspond to the model without reranking.

| Method | R@1 | R@5 | R@10 | R@20 |
|---|---|---|---|---|
| w/o sub-tree | 3.1 | 7.5 | 10.6 | 16.0 |
| Full model | 17.9 | 36.5 | 47.0 | 59.3 |

Table S1. Effect of sub-tree partitioning.

Table S2. Ablation on each loss without the EigenPlace backbone.

| Method | R@1 | R@5 | R@10 | R@20 |
|---|---|---|---|---|
| w/o $\mathcal{L}_{\text{euc}}$ | 5.2 | 10.6 | 15.9 | 22.0 |
| w/o $\mathcal{L}_{\text{hyp}}$ | 17.9 | 36.6 | 41.7 | 53.4 |
| w/o $\mathcal{L}_{\text{hier}}$ | 21.7 | 41.3 | 50.2 | 60.1 |
| Full model | **29.4** | **51.4** | **60.6** | **70.0** |

Table S3. Ablation on spatial aggregation selection.

| Spatial aggregator | GeM | Avg | Max |
|---|---|---|---|
| R@1 | 29.4 | 13.9 | 18.4 |

## S5. Effect of each loss without the EigenPlace backbone

The loss ablation study presented in the main paper (Table 5) was conducted using a backbone initialized with weights from EigenPlaces [3]. In this section, we further evaluate the performance from scratch—as shown in Table S2—to eliminate any potential bias from the EigenPlaces weights. Consistent with the findings in the main paper, our model maintains robust performance even when using only the hierarchical structural loss without $\mathcal{L}_{\text{hyp}}$. This further validates our proposed hierarchical modeling approach. Ultimately, the best performance is achieved when all three loss functions are utilized in conjunction.

## S6. Ablation on spatial aggregator selection

To establish a more comprehensive baseline, we further compare GeM pooling [10] against standard average and max pooling prior to the hyperbolic mapping. As summarized in Table S3, GeM achieves the highest Recall@1 (R@1) by adaptively highlighting discriminative regions via its learnable pooling exponent. These results validate our selection of GeM as the optimal spatial aggregation strategy for our framework.

## S7. More details about hyperbolic manifold

### S7.1. Klein model

To perform hyperbolic aggregation, features in the Poincaré ball model must be converted to the Klein model. Although

the Klein model shares the same underlying space as the Poincaré ball [7], a given point is represented by different coordinates in each model. Let $\mathbf{x}_{\mathbb{D}}$ and $\mathbf{x}_{\mathbb{K}}$ denote the coordinates of the same point in the Poincaré and Klein models, respectively. The transformation between these coordinate systems is given by:

$$\mathbf{x}_{\mathbb{D}} = \frac{\mathbf{x}_{\mathbb{K}}}{1 + \sqrt{1 - c\|\mathbf{x}_{\mathbb{K}}\|^2}}, \tag{1}$$

$$\mathbf{x}_{\mathbb{K}} = \frac{2\mathbf{x}_{\mathbb{D}}}{1 + c\|\mathbf{x}_{\mathbb{D}}\|^2}. \tag{2}$$

Therefore, given a point in the Poincaré ball model, it can be mapped to the Klein model, enabling the computation of averages and subsequently transformed back to the Poincaré model.

### S7.2. Möbius addition

We use Möbius addition in Eq. (3), Eq. (4), and Eq. (5) of the main paper. For a pair of vectors $\mathbf{x}, \mathbf{y} \in \mathbb{D}_c^n$, the Möbius addition is defined as

$$\mathbf{x} \oplus_c \mathbf{y} := \frac{(1 + 2c\langle \mathbf{x}, \mathbf{y} \rangle + c\|\mathbf{y}\|^2)\mathbf{x} + (1 - c\|\mathbf{x}\|^2)\mathbf{y}}{1 + 2c\langle \mathbf{x}, \mathbf{y} \rangle + c^2\|\mathbf{x}\|^2\|\mathbf{y}\|^2}. \tag{3}$$

## S8. More details about experiments

### S8.1. Datasets

We evaluate our model on two perspective-to-equirectangular (P2E) datasets and one perspective-to-perspective (P2P) dataset to compare against their respective baselines.

For the P2E baselines, we use Pitts250K-P2E [12] and YQ360 [12], two datasets from prior P2E VPR research with predefined training, validation, and testing splits.

Pitts250K-P2E is generated from the original Pitts250K dataset [14]. It contains 2,940 training, 3,804 validation, and 4,140 testing queries, along with 2,466 training, 2,116 validation, and 2,158 testing database images. Each database panorama is created by stitching twelve lower-yaw images [13], and queries are also formed from twelve lower-yaw images captured at different times to introduce viewpoint and appearance variations.

YQ360 includes 182 training, 78 validation, and 250 testing query images, along with 91 training, 39 validation, and 125 testing database images. The panoramas are captured with a Panoramic Annular Lens (PAL), while the queries come from front and rear perspective cameras, creating a challenging vertical FoV mismatch.

For the P2P baselines, we use the San Francisco eXtra Large (SF-XL) dataset [2], one of the largest VPR datasets. SF-XL is constructed from Google Street View equirectangular panoramas by horizontally splitting each into twelve crops, yielding 41.2M training, 8k validation, and 2.8M testing database images, along with 8k validation queries, 1,000 test-v1 queries, and 598 test-v2 queries. Test-v1 queries are collected from Flickr, and test-v2 queries from the San Francisco Landmark dataset [4].

Because SF-XL originates from panoramas, it is also suitable for constructing a P2E benchmark. To evaluate HypeVPR in this setting, we construct an additional panoramic test database by mapping each of the 2.8M pre-crop PV images to its corresponding panorama, producing 233,820 panoramic database images for 2,805,840 PV images.

### S8.2. Evaluation metric

We use the Recall@K metric, which is convention in VPR [1, 5, 6, 9, 11, 15, 16]. It measures the proportion of queries where at least one of the top-$K$ database predictions lies within 25 meters.

Execution time refers to the wall-clock time per query required to compute distances from the query descriptor to all database descriptors and rank them, using a full pairwise distance computation without any indexing. Feature extraction is excluded, as the query feature extraction network is almost identical to that used in standard P2P pipelines and therefore provides no structural advantage in our setting; including it would not yield a meaningful comparison. All results are reported on the test set.

### S8.3. More training details

We use equal weights for the three triplet losses and apply the same margin to each. The number of queries per epoch is set to 500, and the cache refresh rate to 125, meaning the model is updated after processing every 125 queries. We also use 10 negatives per query.

For training on YQ360 [12], we follow the baseline protocol: the model is initialized with weights pre-trained on Pitts250K-P2E and then fine-tuned on YQ360.

The various HypeVPR models used for comparison with the P2E baselines were independently trained for each backbone and descriptor size configuration.

### S8.4. More details on model settings

In our main experiments, we use different values of $K'$ for each dataset. For Pitts250K-P2E [12], which contains 2,158 database images in the test split, we set $K' = 400$. For YQ360 [12], which has 125 test database images, we use $K' = 20$. For the SF-XL dataset [2], which contains 233,820 test database images, we apply $K' = 50{,}000$.

## S9. Performance differences in hierarchical retrieval based on $K'$

In adjustable hierarchical retrieval, the number of descriptors filtered by the representative feature—denoted as

| $K'$ | R@1 | R@5 | R@10 | R@20 | time (s) |
|---|---|---|---|---|---|
| 10k | 84.7 | 89.0 | 90.5 | 91.5 | 1.11 |
| 20k | 84.9 | 89.4 | 90.7 | 91.9 | 1.56 |
| 30k | 85.1 | 89.4 | 90.8 | 92.0 | 2.01 |
| 40k | 85.0 | 89.4 | 90.7 | 92.0 | 2.47 |
| 50k | 85.2 | 89.4 | 90.8 | 92.1 | 2.92 |
| 60k | 85.3 | 89.5 | 90.9 | 92.2 | 3.37 |
| 70k | 85.4 | 89.6 | 91.0 | 92.3 | 3.82 |

Table S4. Performance and retrieval time according to $K'$.

$K'$—significantly affects both retrieval accuracy and speed, along with the selected level set $\mathbb{L}$ and the corresponding weights. Using the SF-XL [2] dataset, which contains 233,820 database images, we vary $K'$ from 10k to 70k and evaluate the resulting performance. The model configuration follows HypeVPR-L as used in Tab. 3 of the main paper.

Table S4 reports the performance differences across values of $K'$. Notably, even with $K' = 10$k, the model already achieves strong retrieval accuracy. Although the performance at $K' = 30$k is slightly higher than at $K' = 40$k for reasons that are unclear, the overall trend shows consistent improvement with larger $K'$.

These results demonstrate that our model provides flexible trade-off control through multiple mechanisms, and that selecting an appropriate $K'$ offers a practical balance among speed, recall, and storage efficiency.

## S10. Performance differences in hierarchical retrieval based on score weighting

Our reranking process allows the use of different weights for scores at each hierarchical level in Eq. (20) of the main paper. We measured the performance differences by adjusting the weights applied to each level, as shown in Fig. S3 and Fig. S4. We use an HypeVPR-L model ($L = 5$) with a ConvNeXt-Small backbone [8].

What can be observed here is that combining scores from multiple levels in appropriate proportions leads to better performance compared to relying solely on $\ell = 5$ descriptors for exhaustive searching. This synergy appears to result from the combination of overall scene information provided by the wide FoV of the equirectangular image and the finer details captured by the smallest windows. In the Pitts250k-P2E dataset [12] used for this experiment, the test set achieved the highest recall@1 of 43.5% when $w_1 = 0.25$, while the validation set achieved the highest recall@1 of 48.4% when $w_1 = 0.30$.

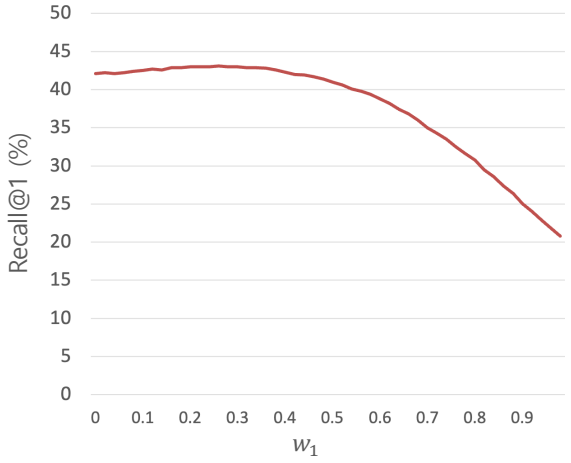


Figure S3. The recall-weight graph of our reranking process shows that the highest recall@1 performance of 43.5% is achieved when $w = 0.25$ on Pitts250k-P2E test set.

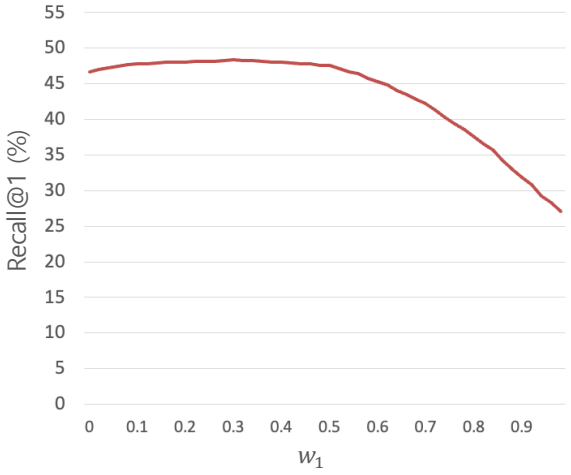


Figure S4. The recall-weight graph of our reranking process shows that the highest recall@1 performance of 48.4% is achieved when $w = 0.3$ on Pitts250k-P2E validation set.

## S11. Visualization of Hierarchical Descriptors Embedded in the Poincaré Ball

Figure S5, S6, S7 provides a joint visualization of our hierarchical descriptors in the Poincaré ball and their corresponding panoramic windows at each level. For the descriptor plot, each $\mathbf{h}_d^{(\ell)}$ is represented using only its norm and angular components for clarity. As the hierarchy level increases, the descriptors move closer to the origin, exhibiting progressively smaller hyperbolic norms. This trend reflects the semantic hierarchy of our representation: higher-level descriptors encode more abstract and global scene information, whereas lower-level descriptors, located near the boundary, capture fine-grained and localized details.

In addition, the descriptors form a clear tree-like structure across levels, where each parent descriptor lies between its associated child descriptors. This geometric organization demonstrates that the proposed hyperbolic embedding effectively preserves the intended hierarchical relationships, which is crucial for coarse-to-fine retrieval and robust matching across varying viewpoints.

To complement this embedding visualization, we also present the actual panorama image decomposed into hierarchical windows (right side of Figures). Showing the image patches alongside their corresponding descriptors helps illustrate how the hyperbolic hierarchy aligns with spatial decomposition in the panorama and clarifies the semantic role of each level.

We note that the Poincaré ball plot is a simplified 2D projection based solely on descriptor norms and angular directions. Despite this simplification, the visualization clearly reveals how our embedding space encodes semantic hierarchy and maintains coherent geometric structure.

## S12. Analyzing failure cases through Poincaré ball visualization

We analyze retrieval failures by visualizing the hierarchical descriptors in the Poincaré ball. Specifically, we examine cases where retrieval based on the representative descriptor $\mathbf{h}_d^{(1)}$ fails.

As shown in Fig. S8, unlike the examples in Figs. S5, S6, and S7, the failure case exhibits a clear outlier among the lower-level window descriptors. This outlier lies far from the main hierarchical structure formed by the other descriptors.

Such an outlier indicates that the corresponding window contains visual content that is semantically inconsistent with the rest of the panorama. When a query image happens to overlap with this outlier region, the representative descriptor $\mathbf{h}_d^{(1)}$ becomes insufficient to guide correct retrieval, causing the system to retrieve an incorrect database image.

## S13. Qualitative results

In Figures S9, S10, S11, S12, and S13, we present the qualitative results of our method. As the baseline, we used the sliding window method from [12] with a ConvNeXt-small [8] backbone and a 16-window setting. Our method was evaluated under the same conditions, using the same backbone and a HypeVPR-L setting.

We visualized the top-3 retrieved database images, where images marked in red represent incorrect retrievals, and those marked in green represent correct retrievals. For our method, which does not rely on window-level predictions, the highest $\ell = 5$ score is highlighted in yellow.

As shown in Fig. S9, the baseline method makes incorrect predictions for all top-3 retrieved images. In contrast, our method correctly predicts the top-1 image but makes incorrect predictions for the second and third images. Since the second and third images contain regions similar to the query image, it appears challenging for our method, which represents the entire image with a single descriptor, to capture all the fine details. However, through the reranking process, this limitation is overcome, leading to more accurate predictions. Fig. S10 and Fig. S11 also show that, while the initial predictions are somewhat inferior to those of the baseline method, the reranking process clearly yields better results.

Our method excels in challenging scenarios, as illustrated in Figures S12 and S13. For example, in Fig. S12, where there is a significant difference in scene distance between the query and database images, the baseline method fails to match windows correctly. In contrast, our method successfully matches the images using hierarchical features. Similarly, in Fig. S13, where both window-level matching and matching with our method's representative feature are difficult, the reranking process effectively combines coarse and fine information, enabling accurate retrieval.

## S14. Visualization of hierarchical retrieval

In Fig. S14, we visualize the scores of hierarchical features at each level and illustrate the process and results of reranking using these scores.

On the left, we observe the initially predicted top-3 retrieved results using the Hierarchy Level-1 scores. While the second and third predictions are correct, the top-1 prediction is incorrect. On the right, the scores for the top-1 and second predictions, calculated using Eq. (19) of the main paper, are visualized. Each hierarchical window's score is displayed below the corresponding window, with higher scores highlighted in deeper red.

Using the Level-1 scores ($\ell = 1$), 3.3 and 3.2, the top-2 database images were retrieved. However, the top-1 prediction is incorrect, as shown in the figure. Unlike the second image, where the scores are evenly distributed across all levels, the first image lacks accurate matches with high scores at $\ell = 4$. When the scores are recalculated using $\mathbb{L} = \{4\}$ in Eq. (20) of the main paper, the score of the first image decreases to 1.4, causing it to fall out of the top-3 predictions. Meanwhile, the score of the second image increases to 3.0, making it the top-1 prediction.

As a result, the hierarchical retrieval process produces more accurate predictions, as illustrated in the bottom-left image.

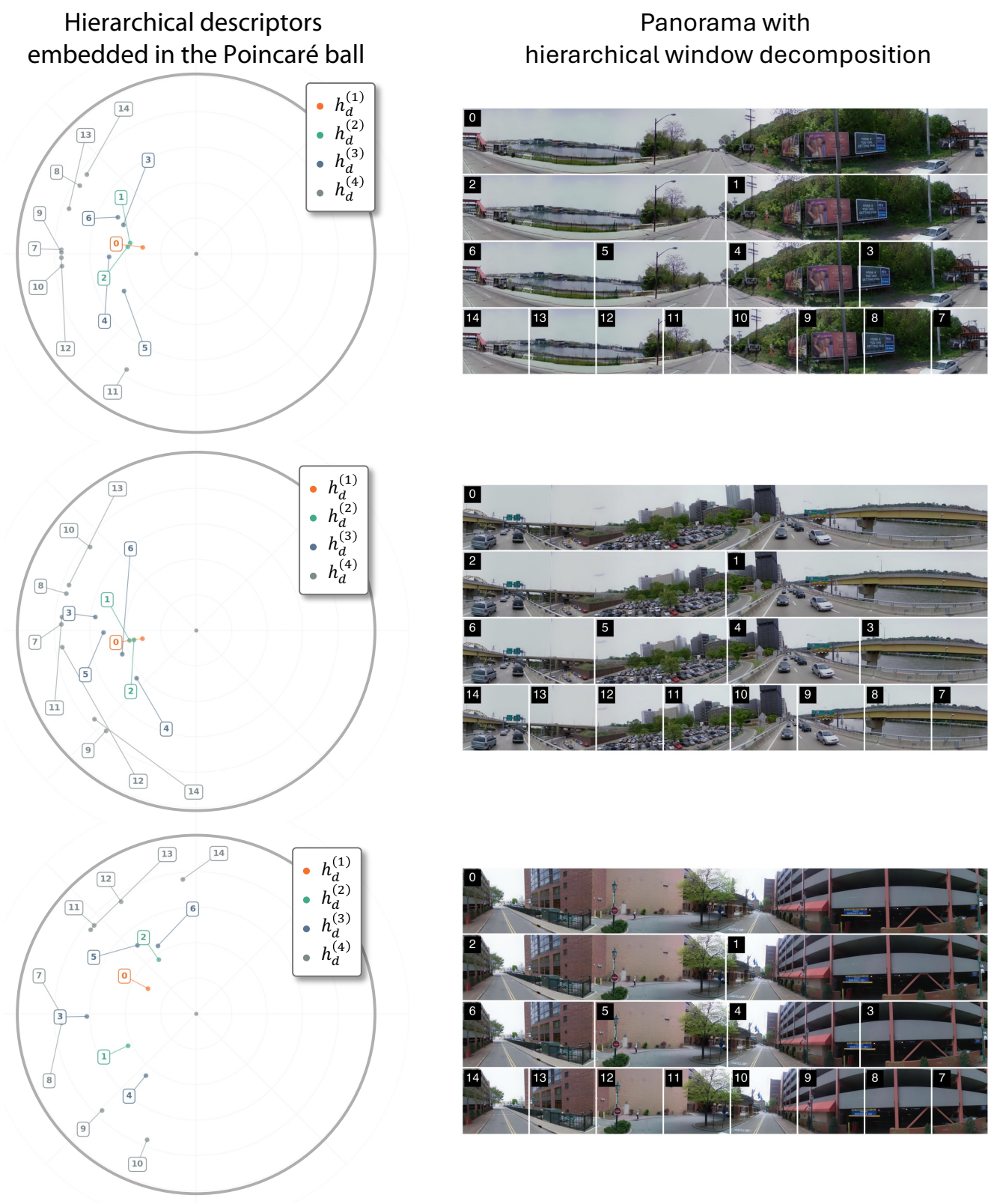


Figure S5. Visualization of hierarchical descriptors embedded in the Poincaré ball (left) and their corresponding hierarchical window decomposition on the panoramic image (right).

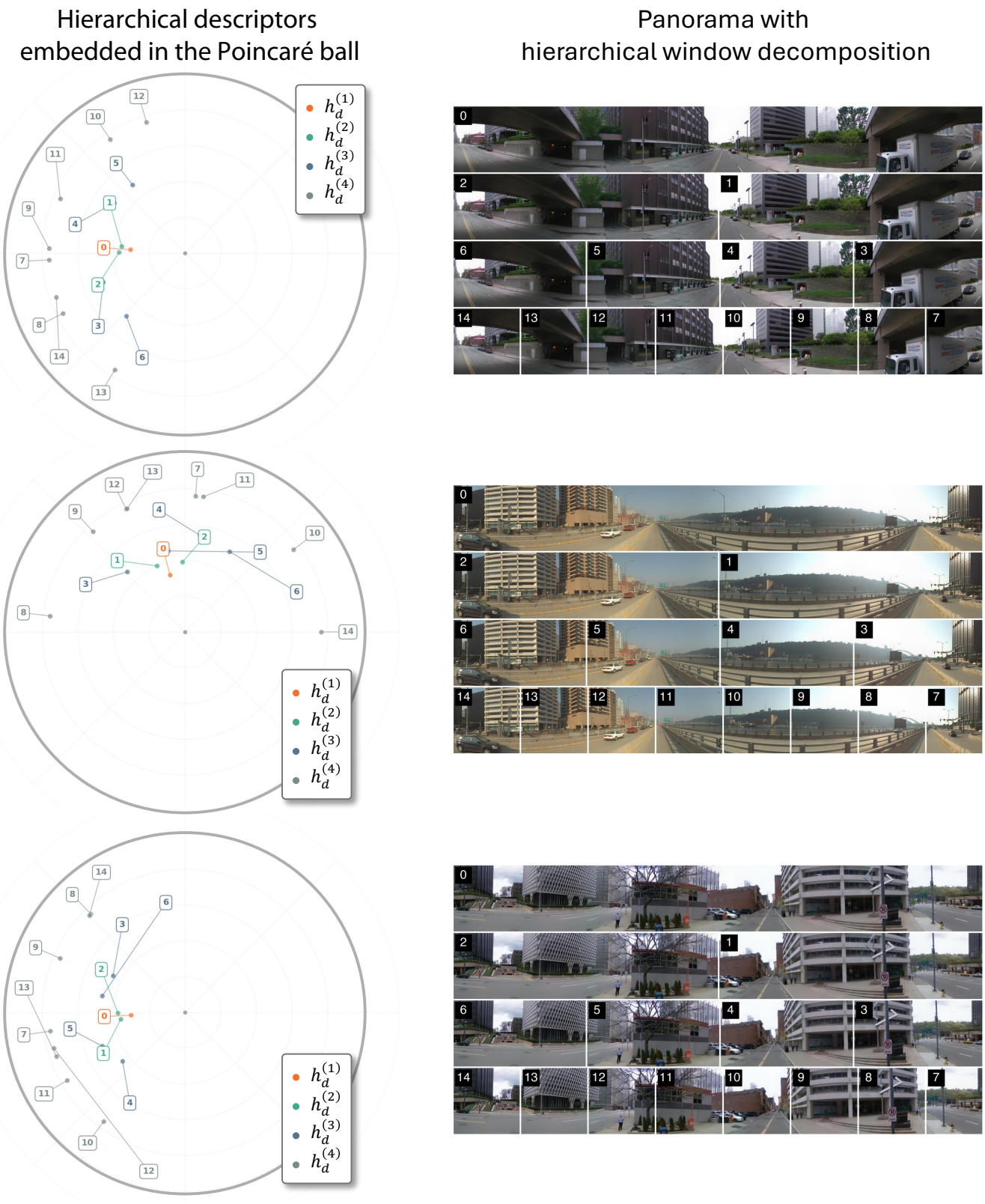


Figure S6. Visualization of hierarchical descriptors embedded in the Poincaré ball (left) and their corresponding hierarchical window decomposition on the panoramic image (right).

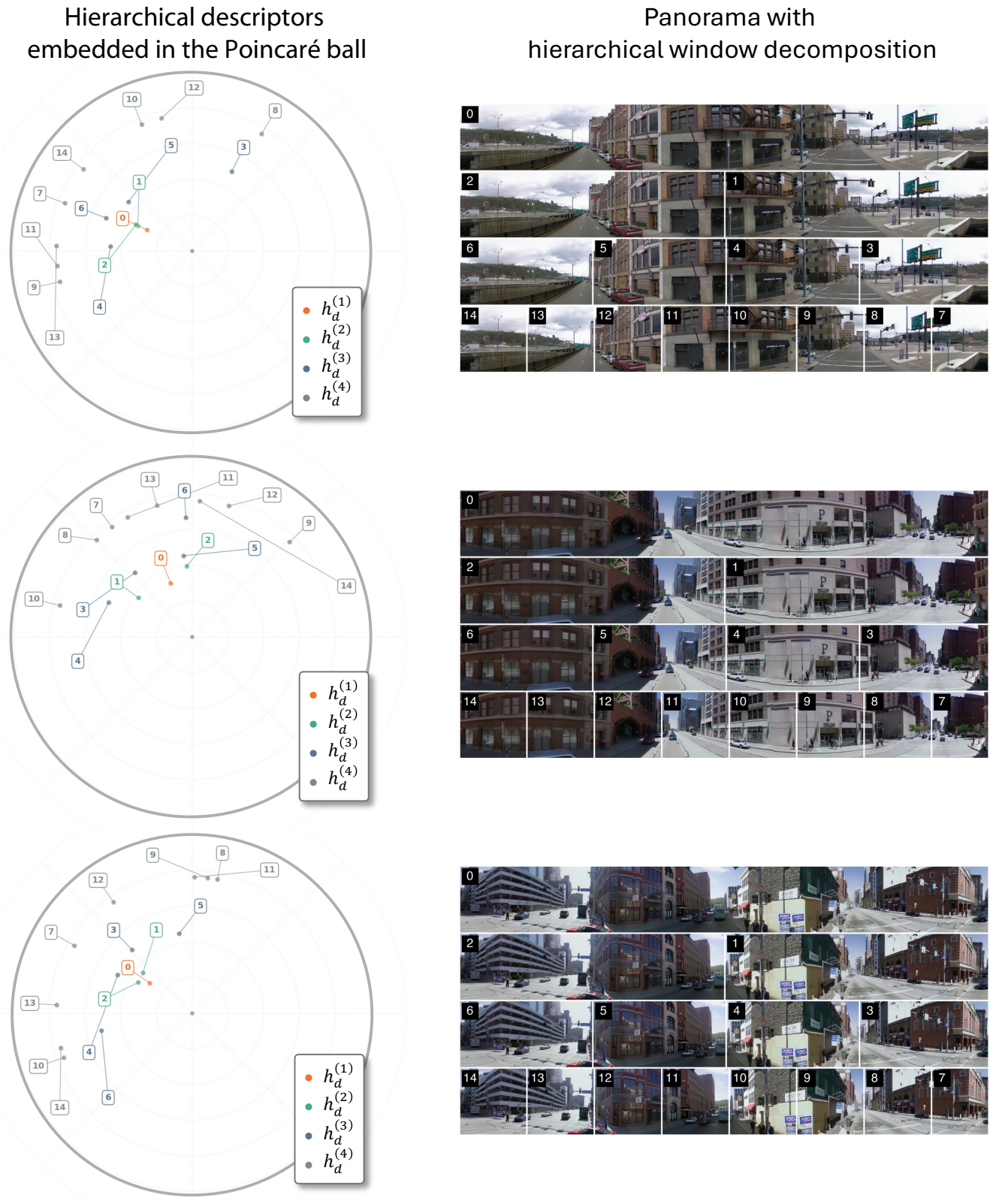


Figure S7. Visualization of hierarchical descriptors embedded in the Poincaré ball (left) and their corresponding hierarchical window decomposition on the panoramic image (right).

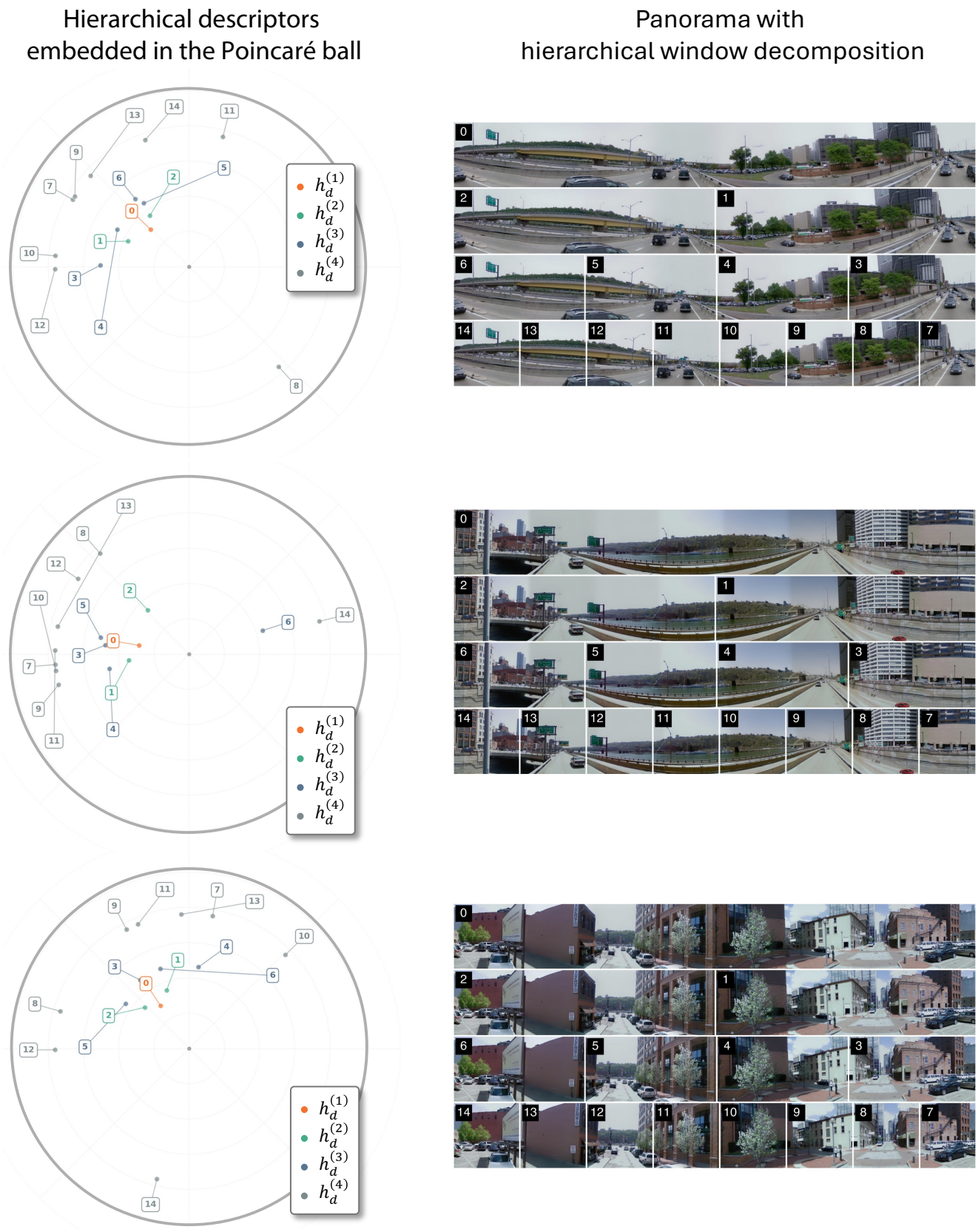


Figure S8. Failure cases where a descriptor corresponding to a specific panoramic window becomes an outlier in the Poincaré ball. The query window associated with this region consistently fails retrieval, as the outlier disrupts the hierarchical matching.


Figure S9. Qualitative results. Correct predictions are marked in green, while incorrect predictions are marked in red. In our method, the window with the highest score within the positive pair is highlighted in yellow.


Figure S10. Qualitative results. Correct predictions are marked in green, while incorrect predictions are marked in red. In our method, the window with the highest score within the positive pair is highlighted in yellow.

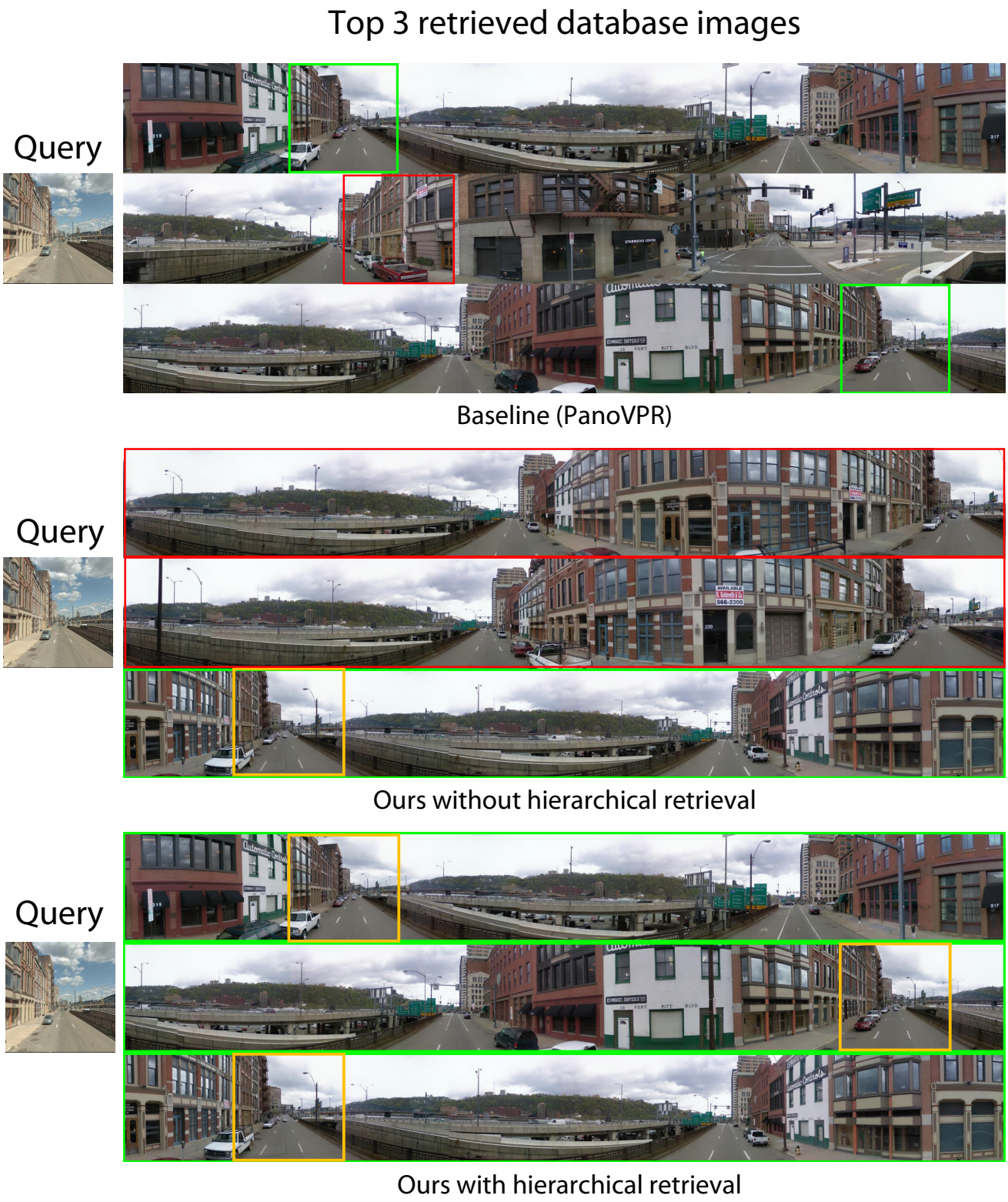


Figure S11. Qualitative results. Correct predictions are marked in green, while incorrect predictions are marked in red. In our method, the window with the highest score within the positive pair is highlighted in yellow.

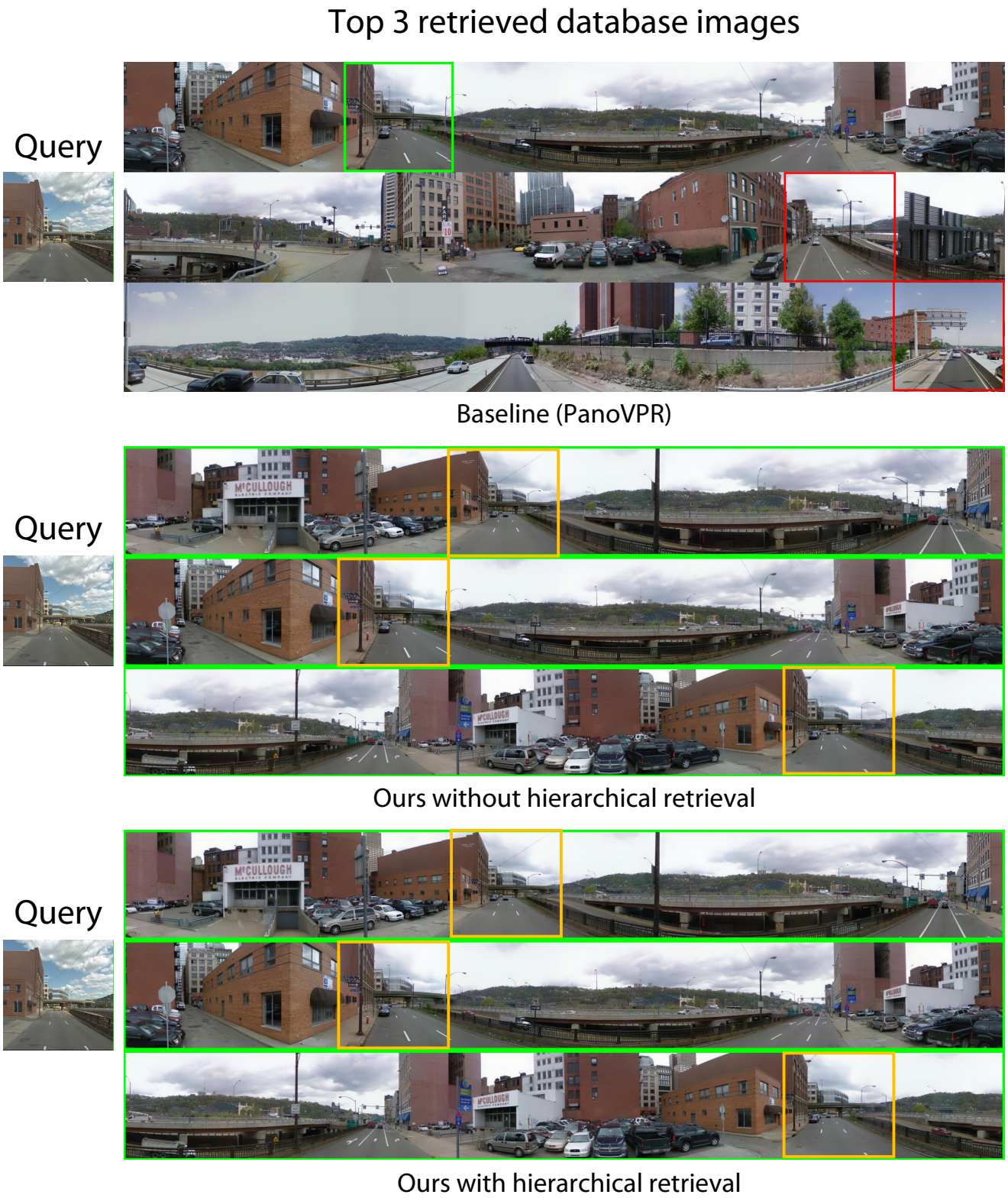


Figure S12. Qualitative results. Correct predictions are marked in green, while incorrect predictions are marked in red. In our method, the window with the highest score within the positive pair is highlighted in yellow.

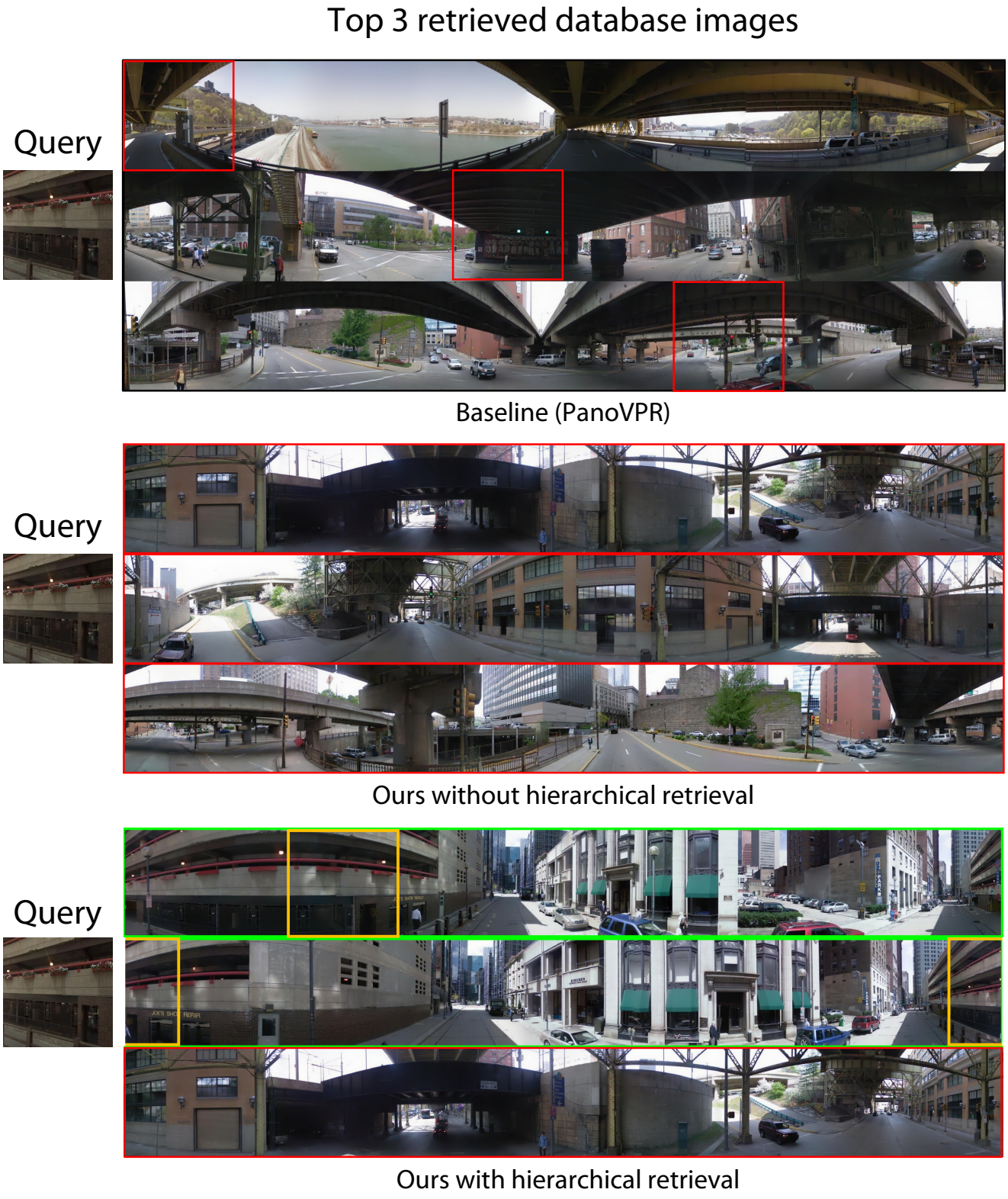


Figure S13. Qualitative results. Correct predictions are marked in green, while incorrect predictions are marked in red. In our method, the window with the highest score within the positive pair is highlighted in yellow.

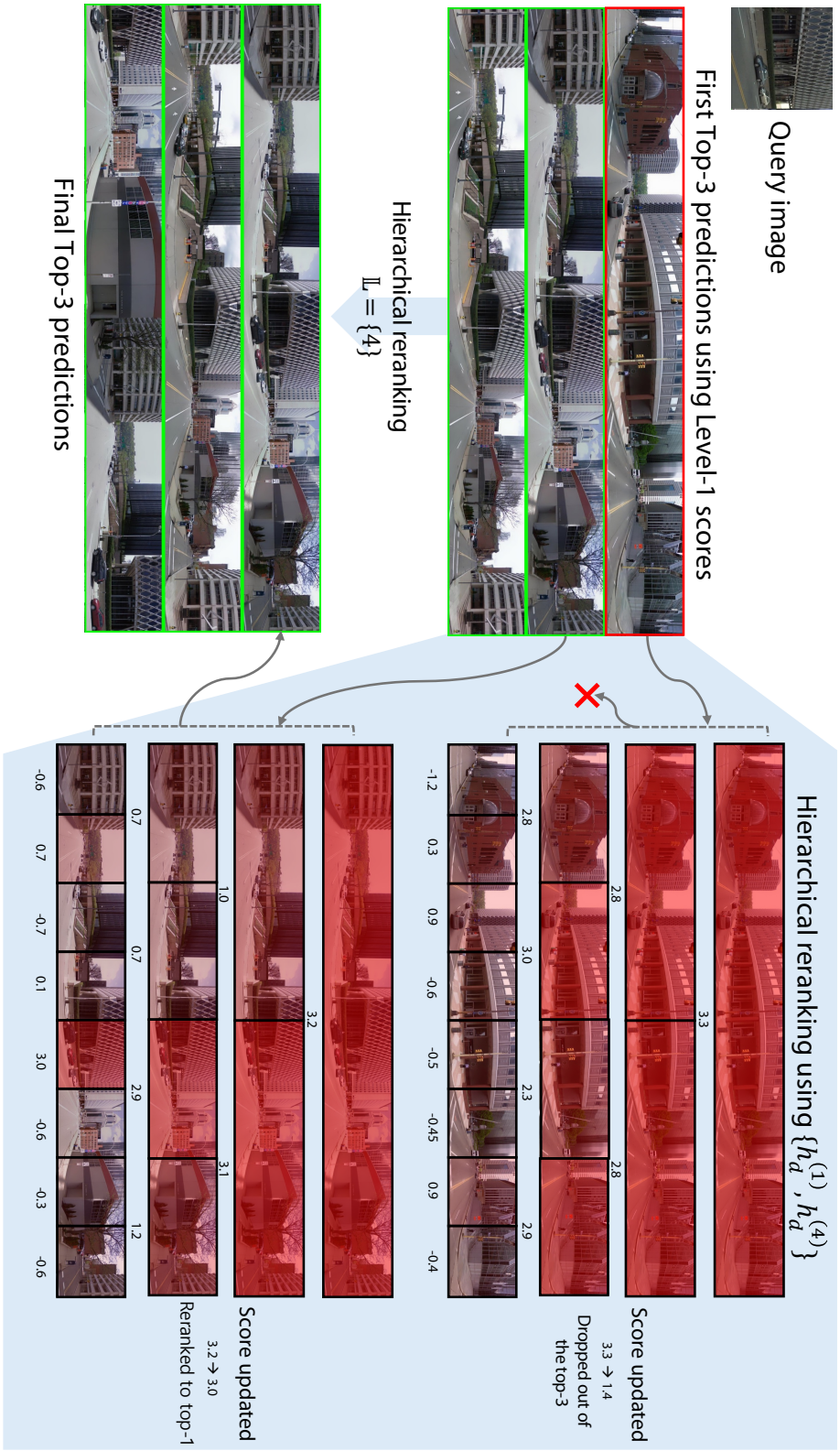


Figure S14. Visualization of results before and after reranking, along with scores at each hierarchical level. Green borders on the retrieved images indicate correct predictions, while red borders represent incorrect predictions.

## References


[1] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. Boq: A place is worth a bag of learnable queries. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 17794–17803, 2024. 3

[2] Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking visual geo-localization for large-scale applications. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4878–4888, 2022. 3, 4

[3] Gabriele Berton, Gabriele Trivigno, Barbara Caputo, and Carlo Masone. Eigenplaces: Training viewpoint robust models for visual place recognition. In *Int. Conf. Comput. Vis.*, pages 11080–11090, 2023. 2

[4] David M Chen, Georges Baatz, Kevin Köser, Sam S Tsai, Ramakrishna Vedantham, Timo Pylvänäinen, Kimmo Roimela, Xin Chen, Jeff Bach, Marc Pollefeys, et al. City-scale landmark identification on mobile devices. In *CVPR 2011*, pages 737–744. IEEE, 2011. 3

[5] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 14141–14152, 2021. 3

[6] Sergio Izquierdo and Javier Civera. Optimal transport aggregation for visual place recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 17658–17668, 2024. 3

[7] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6418–6428, 2020. 3

[8] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11976–11986, 2022. 4, 5

[9] Feng Lu, Xiangyuan Lan, Lijun Zhang, Dongmei Jiang, Yaowei Wang, and Chun Yuan. Cricavpr: Cross-image correlation-aware representation learning for visual place recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 16772–16782, 2024. 3

[10] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018. 2

[11] Yanqing Shen, Sanping Zhou, Jingwen Fu, Ruotong Wang, Shitao Chen, and Nanning Zheng. Structvpr: Distill structural knowledge with weighting samples for visual place recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11217–11226, 2023. 3

[12] Ze Shi, Hao Shi, Kailun Yang, Zhe Yin, Yining Lin, and Kaiwei Wang. Panovpr: Towards unified perspective-to-equirectangular visual place recognition via sliding windows across the panoramic view. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1333–1340. IEEE, 2023. 1, 3, 4, 5

[13] Akihiko Torii, Josef Sivic, Tomas Pajdla, and Masatoshi Okutomi. Visual place recognition with repetitive structures. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 883–890, 2013. 3

[14] A. Torii, J. Sivic, M. Okutomi, and T. Pajdla. Visual place recognition with repetitive structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015. 3

[15] Ruotong Wang, Yanqing Shen, Weiliang Zuo, Sanping Zhou, and Nanning Zheng. Transvpr: Transformer-based place recognition with multi-level attention aggregation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 13648–13657, 2022. 3

[16] Sijie Zhu, Linjie Yang, Chen Chen, Mubarak Shah, Xiaohui Shen, and Heng Wang. R2former: Unified retrieval and reranking transformer for place recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 19370–19380, 2023. 3