Simple but Effective Triplet-Based Compression Strategies for Compact Visual Localization

-

Supplementary Material

Torsten Sattler[1] Zuzana Kukelova[2]
[1] Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague
[2] Visual Recognition Group, Faculty of Electrical Engineering, Czech Technical University in Prague

This supplementary material contains the following: Sec. A provides details on the Sharing-Based Triplet Selection (STS) and Fine-Grained Point Selection (FPS) variants briefly described in Sec. 3 of the main paper. Sec. A also provides an analysis of the computational complexities of the different strategies. In addition, the section presents an experiment that studies the pose accuracy that can be achieved by the 3D points selected by the TTS approach for images taken from different viewpoints than the database images. Sec. B describes the two localization systems used in the experimental evaluation of the main paper in more detail (as promised in Sec. 4 of the main paper). Sec. C provides ablation studies for the STS and FPS variants (as promised in Sec. 4.1 of the main paper), shows figures discussed in the main paper but not shown there, and presents ablation studies for combining our approaches with descriptor compression schemes. Sec. D reports localization times achieved when using the TTS strategy.

A. Additional Variants

In this section, we provide more details on the additional variants (STS and FPS) of our method that were briefly mentioned in Sec. 3.2 of the main paper. For reference, we include the pseudocode for our main Trivial Triplet Sampling (TTS) method in Alg. 1.

A.1. Sharing-Based Triplet Selection (STS)

As shown in Sec. 4 of the main paper, the TTS approach performs well despite its rather trivial nature. In particular, it offers comparable or better pose accuracy at similar memory requirements compared to more complex compression schemes. However, even with PROSAC-style sampling, we notice that the triplets selected for the individual database images are rather disjoint. As a result, the TTS strategy is unable to select very small subsets of points. This motivated the design of the STS variant, which was briefly described in the main paper. The following expands upon the description in the main paper.

Algorithm 1 Trivial Triplet Sampling (TTS)

input: Desired number N of points per image, number n_{triplets} of triplets to be estimated per database image, orientation error threshold α, position error threshold τ

```
selectedPoints = ∅
for each image do
    select n_triplets triplets of 2D-3D correspondences
    goodTriplets = ∅
    for each triplet t = {p1, p2, p3} do
        estimate camera poses {(R_i, c_i)} using a P3P solver
        for each estimated pose (R_i, c_i) do
            // Measure orientation error
            orient_err = arccos((trace(R_sfm^-1 R_i) - 1)/2)
            if orient_err < α then
                goodTriplets = goodTriplets ∪ {t}
    sort goodTriplets based on position errors Δ_i = ||c_i - c_sfm||
    Δ_t1 = position error of first triplet t1 in goodTriplets
    tmpPoints = ∅
    for triplet t = {p1, p2, p3} in goodTriplets do
        if Δ_t > τ · Δ_t1 then
            break
        selectedPoints = selectedPoints ∪ {p1, p2, p3}
        tmpPoints = tmpPoints ∪ {p1, p2, p3}
        if |tmpPoints| ≥ N then
            break
return: selectedPoints
```

Given that each database image has hundreds to thousands of 2D-3D matches, it is unlikely to sample triplets containing exactly the same 3D points in two database images. Still, a triplet sampled in one image can lead to an accurate pose estimate in another image. Our **Sharing-Based Triplet Selection** (**STS**) approach thus extends the list *goodTriplets* for a database image `img` with good triplets selected for other database images.

Rather than selecting triplets purely based on pose qual-

Algorithm 2 Sharing-Based Triplet Selection (STS)

input: Desired number N of points per image, number n_{triplets} of triplets to be estimated per database image, orientation error threshold α, position error threshold τ

```
selectedPoints = ∅
for each image img do
    compute the set goodTriplets(img) as in Alg. 1
for each image img do
    // Extend the set of good triplets for other images
    for each t = {p1, p2, p3} ∈ goodTriplets(img) do
        determine all other images observing p1, p2, and p3
        add t to goodTriplets(img') for each such image img'
for each image img do
    sort goodTriplets(img) based on position errors
    filter goodTriplets(img) using α and τ
for each triplet t do
    score(t) = |{img | t ∈ goodTriplets(img)}|
for each image img do
    re-order goodTriplets(img) based on decreasing score(t)
    tmpPoints = ∅
    for triplet t = {p1, p2, p3} in goodTriplets(img) do
        selectedPoints = selectedPoints ∪ {p1, p2, p3}
        tmpPoints = tmpPoints ∪ {p1, p2, p3}
        if |tmpPoints| ≥ N then
            break
return: selectedPoints
```

ity, STS preferably selects triplets that lead to good poses[1] in as many database images as possible. Compared to TTS, STS selects fewer 3D points for a given value for N, at the cost of reduced pose accuracy.

Alg. 2 shows pseudocode for the STS variant. First, we compute a set of good candidate triplets for each database image. This is done using the same mechanism as in TTS (for convenience, Alg. 1 replicates the pseudocode for TTS from the main paper): For each database image, we sample n_{triplets} of 2D-3D matches and estimate camera poses for each triplet using a P3P solver. We filter out triplets for which all estimated poses have a rotation error equal to or larger than α compared to the ground truth pose from the Structure-from-Motion model. We then sort all remaining triplets based on their position errors and filter out triplets for which the position error is τ times larger than the position error of the triplet with the smallest position error. For each triplet t remaining after filtering, we determine the set of all other database images that observe all three points p_1, p_2, and p_3 in the triplet $t = \{p_1, p_2, p_3\}$. For each such image `img`, we obtain the 2D-3D matches corresponding to the points p_1, p_2, and p_3 and estimate camera poses using a P3P solver. If one pose passes the orientation filter (*cf*. Alg. 1), we add t to the list of candidate triplets for `img`. For each image, we sort its list of candidate triplets based on increasing position errors and remove triplets that do not pass the distance check (*cf*. Alg. 1).

STS select triplets that lead to good poses in as many database images as possible to reduce the number of 3D points that are selected. Thus, in the next step, STS computes a score for each triplet that corresponds to the number of such images. For each database image, we then re-order its triplets in descending order of these scores. From that sorted list, we then select triplets and their corresponding points until N points have been selected for a database image, or until we have ran out of candidates.

Discussion. Compared to the TTS strategy, the STS approach does not add any additional parameters. However, the parameter τ becomes more important: As for the TTS approach, τ controls the accuracy of the camera poses estimated from the triplets. Increasing τ increases the number of images for which a triplet t generates a good pose. Thus, increasing τ makes stronger compression possible. However, increasing τ also increases the chance of less accurate localization results: Since the STS approach does not strictly select triplets based on pose accuracy, relaxing the definition of good poses increases the chance that points leading to less accurate poses are selected. Still, as we will show in Sec. C, there is a reasonably large range of values for τ that lead to good results, *i.e*., the choice of τ is not too critical.

A.2. Fine-Grained Point Selection

For each database image, the TTS and STS approaches select the top-ranked triplets from sorted lists: For TTS, triplets are ranked based on their position errors. For STS, triplets are ranked based on the number of database images for which they produce accurate poses. From the perspective of selecting as few 3D points as possible overall, taking the top-ranked triplets is sub-optimal: Consider a simple example consisting of triplets $t_1 = \{p_1, p_2, p_3\}$, $t_2 = \{p_4, p_5, p_6\}$, and $t_3 = \{p_1, p_2, p_4\}$, given in this order, and $N = 4$. TTS and STS will select triplets t_1 and t_2, thus selecting 6 distinct points. In contrast, selecting triplets t_1 and t_3 leads to only 4 distinct points.

We thus propose another triplet selection scheme (**Fine-Grained Point Selection** (**FPS**)), which considers individual points and can readily replace the original selection strategy in TTS and STS. FPS prefers to select triplets containing 3D points that have already been selected: Given a sorted list of triplets for a database image, FPS selects the top-ranked triplet in that list containing three points that have already been selected, *i.e*., points in the list *selectedPoints* in Alg. 1 or Alg. 2. If no such triplet exists, FPS selects the first triplet in the sorted list that contains two points that have already been selected. If no such triplet exists, FPS selects the first triplet in the sorted list that contains one point that has already been selected. If no such triplet exists, FPS selects the first triplet in the sorted list. The se-

[1] Defined as passing the checks based on the thresholds α and τ on the rotation errors and relative position errors of the corresponding poses.

lected triplet is then removed from the list. This process terminates once N distinct points have been selected for the database image, or if there is no more triplet to consider.

The **TTS+FPS** and **STS+FPS** variants (using TTS respectively STS to generate the ordered lists of candidate triplets) produce smaller sets of selected points for a given N at the cost of pose accuracy.

Discussion. Again, the FPS strategy does not introduce any new parameters. As with the STS approach, the choice of τ impacts pose accuracy and the compression rate: A larger value for τ makes it possible to select fewer points, at the risk of selecting points that lead to less accurate pose estimates. Again, we observe that there is a sensible range of choices for τ that lead to good results.

A.3. Computational Complexity

The following analyzes of the computational complexities of the different strategies proposed in this work.

The number of 3D points observed in a database image, and thus the number of unique triplets that can be selected for that image, is bounded by the number of local features extracted from the image (typically a constant, *e.g.*, at most 4096). Thus, TTS processes a constant number of triplets and selects at most a constant number of 3D points per image. As a result, the run-time per image is constant and independent of the total number of database images or the scene scale, yielding an overall complexity of $\Theta(M)$ for M database images. Any structure compression method must process all P 3D points at least once, implying a lower bound of $\Omega(P)$. Since the number of 3D points contributed per image is bounded by a constant, $\mathcal{O}(P) = \mathcal{O}(M)$. Therefore, TTS achieves an asymptotically optimal run-time complexity and scales well to large scenes.

TTS treats each database image independently of the others. In contrast, STS exchanges triplets between images, leading to a worst-case complexity of $\mathcal{O}(M^2 \log M)$, as each database image might exchange triplets with each other database image. In practice, the computational complexity it is often closer to linear as each database image typically only has visual overlap with a small number of other database images.

Given a list of triplets, the FPS scheme traverses this list until a pre-defined number of triplets / 3D points is selected. This number is in practice a constant and FPS does not change the length of the list. Thus, using the FPS scheme does not affect the asymptotic complexity of either the TTS or STS strategy.

A.4. Impact of Viewpoint Changes on Pose Accuracy

The pose accuracy that can be achieved for query images depends on many factors, *e.g.*, scene depth, camera distance and viewing angle, point noise, feature robustness, and database–query overlap. Since the query poses are unknown, it is unclear which factors dominate in advance, making comprehensive ablation studies difficult. These complex dependencies also appear for out-of-distribution query images. Together with a lack of suitable datasets, this likely explains why prior work does not study such scenarios in depth. Similarly, extensive ablations are beyond the scope of this work.

Nevertheless, we include an experiment on the Aachen dataset to simulate an out-of-distribution setting. For simulation, we group pairs of database images by the angle between their optical axes (*e.g.*, 5–10°, 10–20°, *etc.*). One image in each pair is treated as a query image, and localization is performed using TTS. We report the percentage of images localized within 50cm and 5° per group in Tab. A. The points selected by TTS enable robust localization under a wide range of viewpoint changes. The drop in performance for smaller N largely comes from a lack of sufficient matches in the overlapping regions. In practice, using multiple database images alleviates this issue.

B. Details on the Localization Pipelines

This section provides more details on the localization pipelines used in Sec. 4 of the main paper.

Direct Localization (DLoc). DLoc follows the direct matching paradigm [9]. It establishes 2D-3D correspondences by directly comparing the descriptors of the features extracted from a query image with the descriptors associated with the 3D points, without an intermediate image retrieval step.

Each selected 3D point is represented via a single feature descriptor, corresponding to the mean of the descriptors of all features that were used to triangulate the 3D point. This mean is computed from the full SfM model.

Given a query image and its features, we find the k nearest neighbors for each query feature using exhaustive matching (implemented on the GPU using the Faiss library [3]). For a query feature f with descriptor $\mathbf{d}_f$, let $(\mathbf{p}_i, \mathbf{d}_i)$ be the i-th nearest neighbor, consisting of a 3D point position $\mathbf{p}_i$ and the corresponding descriptor $\mathbf{d}_i$. Assuming that the neighbors are sorted based on ascending descriptor distances, we establish a 2D-3D correspondence $f \leftrightarrow \mathbf{p}_1$ if $\|\mathbf{d}_f - \mathbf{d}_1\|_2 < 0.9 \cdot \|\mathbf{d}_f - \mathbf{d}_j\|_2$. The j-th neighbor is selected as

$$j = \text{argmin}_{l=2,\cdots,k} \|\mathbf{p}_1 - \mathbf{p}_l\| \quad \text{s.t. } \|\mathbf{p}_1 - \mathbf{p}_j\| \geq 10\text{cm} \ ,$$

i.e., we select the 3D point among the k-nearest neighbors that is closest to $\mathbf{p}_1$ in 3D space but at least 10cm away. This strategy is a variation of the commonly used SIFT ratio test [7]. Rather than using the 2nd nearest neighbor in descriptor space, we select the neighbor with the closest 3D

	0–5°	5–10°	10–20°	20–30°	30–40°	40–50°	50–60°	60–70°
TTS ($N = 10$)	68.9	66.4	63.9	60.5	54.2	48.0	43.9	39.9
TTS ($N = 20$)	87.5	85.7	84.3	83.4	79.9	75.7	73.2	69.3
TTS ($N = 30$)	93.6	92.6	91.7	91.3	89.4	87.7	85.7	83.6

Table A. **Simulating the pose accuracy obtained with 3D points selected by TTS for out-of-distribution images.** We select pairs of database images from the Aachen dataset. For each pair, we record the angle between the two optical axes and group pairs based on these angles. For each pair, we treat one image as the query image and localize it using the 3D points selected by the TTS approach from the second image. We report the percentage of query images localized within 50cm and 5° per group.

point to the 3D point of the nearest neighbor. Intuitively, this variant is able to better handle global ambiguities, *i.e.*, similar structures found in different parts of the model, under the assumption that descriptors are locally unique.[2] We observe a small boost for $k > 2$ for relatively easy and small scenes such as Cambridge Landmarks and 7Scenes and a significant boost for more complex scenes such as Aachen v1.0. We use $k = 8$ for Cambridge and 7Scenes, and $k = 2048$ for Aachen.[3]

We sort the resulting set of 2D-3D matches in ascending order of the ratio test values and estimate the pose by applying a P3P solver [8] inside RANSAC with local optimization [5] and PROSAC sampling [2]. We run RANSAC for at most 100k iterations and use the implementation provided by the PoseLib [4] library.

The memory consumption of the DLoc pipeline is

$$\mathrm{m}_{\text{DLoc}} = \mathrm{m}_{\text{over}} + \mathrm{n}_{\text{points}} \left(\mathrm{m}_{\text{desc}} + \mathrm{m}_{\text{point pos}}\right) \text{ bytes} \ , \quad (1)$$

where $\mathrm{n}_{\text{points}}$ is the number of 3D points, $\mathrm{m}_{\text{point pos}}$ is the memory required to store the 3D position of a 3D point ($3 \cdot 4 = 12$ bytes in our implementation), m_{desc} is the memory required to store a descriptor, and m_{over} accounts for overheads, *e.g.*, to store a PCA-based projection matrix or the codebooks required for product quantization.

Retrieval-based Localization (RLoc). RLoc uses a standard hierarchical localization pipeline: First, image retrieval is used to identify the top-20 ranked database images[4] for each query image. RLoc uses 256-dimensional EigenPlaces descriptors [1] for retrieval. Each descriptor entry is quantized to 4 bits using a Scalar quantizer from the Faiss library. Again, each selected 3D point is represented via a single feature descriptor, corresponding to the mean of the descriptors of all features that were used to triangulate the 3D point. This mean is computed from the full SfM model.

For each of the top-ranked database images, we match the features extracted from the query image against the descriptors of the 3D points visible in the image using the learning-based LightGlue [6] matcher (to this end, we project the 3D points into the database image). We gather the resulting 2D-3D matches from all top-ranked images and run a single RANSAC with local optimization for pose estimation. Again, we run RANSAC for 100k iterations and use the RANSAC implementation from the PoseLib library.

For a set of 3D points selected by a structure compression scheme, we keep for each database image `img` references to the selected points that are visible in that image, *i.e.*, the set of selected points that were triangulated using features from `img`.

The memory consumption of the RLoc pipeline is

$$\begin{aligned} \mathrm{m}_{\text{DLoc}} \quad = \quad & \mathrm{m}_{\text{over}} + \mathrm{n}_{\text{points}} \left(\mathrm{m}_{\text{desc}} + \mathrm{m}_{\text{point pos}}\right) \\ & + \sum_j \left(28 + \mathrm{m}_{\text{r-desc}} + \mathrm{m}_{\text{ptidx}}(j)\right) \text{ bytes} \ . \end{aligned} \quad (2)$$

As for DLoc, $\mathrm{n}_{\text{points}} \left(\mathrm{m}_{\text{desc}} + \mathrm{m}_{\text{point pos}}\right)$ accounts for the memory required to store the descriptors and point positions. The sum iterates over all database images. For each image, it sums up the memory needed to store its camera pose (stored using 7 floating point values, corresponding to the camera position and its orientation stored as a quaternion, resulting in 28 bytes of storage), the memory $\mathrm{m}_{\text{r-desc}}$ needed to store the retrieval descriptor for that image, and the memory $\mathrm{m}_{\text{ptidx}}(j)$ required to store references[5] to the points visible in that database image. Note that RLoc uses strictly more memory than DLoc for the same set of selected 3D points.

C. Experimental Evaluation

In this section, we provide an ablation study of the parameters of STS and FPS, extended versions of some of the figures shown in the paper, additional figures promised in the main paper, and an ablation study on the impact of descriptor compression on the performance of TTS.

C.1. Ablation of STS and FPS

We evaluate the impact of the parameters τ and α. To this end, we vary the parameter N controlling the num-

[2] This is a valid assumption given that during the SfM process, features are matched between images, which often filters out locally ambiguous features.

[3] $k = 2048$ is the maximum number of nearest neighbors that can be returned by the GPU nearest neighbor search of Faiss.

[4] Retrieving more images improves localization accuracy at the cost of higher run-times. We found that retrieving the top-20 images is a good compromise.

[5] We use 16 bits if there are less than 2^{16} selected 3D points and 32 bits otherwise.

ber of points selected per image. We measure the percentage of query images localized within 10cm and 1° of the ground truth among all query images from the 5 Cambridge scenes.[6] For a given value for N, we report the memory (in MB) needed to store all 5 scenes.

Impact of parameter τ. As discussed in Sec. 3.1 of the main paper, the parameter τ has no significant impact on the TTS strategy, as long as it is set large enough, and thus enough triplets are available for selection. Motivated by the observation that the poses with the smallest position error have errors around 1cm, we thus set $\tau = 10$ per default, allowing us to select triplets with tens of centimeters of position error.

In contrast to the TTS approach, the parameter τ impacts the STS method. Fig. A (middle) evaluates this impact of τ. As can be seen, larger values for τ enable stronger compression, at the cost of reduced pose accuracy: Larger values for τ enable the STS approach to select triplets that are visible in many database images, but lead to less accurate poses. Using $\tau = 5$ or $\tau = 10$ leads to similar results, and any values in between can be expected to work as well, showing that choosing a suitable value for τ is not too hard.

Impact of the FPS strategy for point selection. Figs. A (left) and (right) show how using the FPS strategy, instead of the simple triplet selection from Algs. 1 and 2, impacts the TTS and STS approaches. As discussed in Sec. A.2, the choice of τ can affect both TTS+FPS and STS+FPS. Hence, we also show the impact of τ on both variants.

The FPS strategy was designed to allow our approaches to select smaller subsets of 3D points. As can be seen from the figures, the FPS strategy achieves this goal (even though the reduction in memory can be small). Larger values for τ enable stronger compression, as evident from the curves starting further to the left. This is due to more triplets being available for selection. However, setting τ too large also (significantly) impacts pose accuracy. However, both $\tau = 5$ and $\tau = 10$ lead to memory reductions while offering higher pose accuracy at similar memory footprints.

Based on the results presented above, we use $\tau = 10$ for all further experiments and for all of our approaches (TTS, TTS+FPS, STS, STS+FPS).

Ablation on the number of selected points for a given N. The second column of Tab. F shows the number of 3D points selected by the different variants for the Aachen Day-Night v1.0 dataset. As can be seen, for a given value of N, STS selects fewer points than TTS, while using FPS further reduces the number of selected points.

[6]Note that in the main paper, we showed the average percentage of query images localized within the error threshold, averaged over the 5 scenes.

C.2. Extended Figures

Next, we provide extended versions of the figures presented in the main paper.

Extended version of Fig. 2 in the main paper. Fig. B extends Fig. 2 from the main paper by including plots for the other pose error thresholds for the Aachen Day-Night v1.0 dataset (as promised in the caption of Fig. 2 and as discussed in Sec. 4.2).

As noted in the main paper, all of our variants clearly offer a better pose accuracy-memory tradeoff than all baselines for Cambridge and 7Scenes. For Aachen Day, our strategies offer the best performance for the fine threshold (25cm, 2°), while all strategies perform similarly well for the medium (50cm, 5°) and coarse (5m, 10°) thresholds. For Aachen Night, our approaches again outperform the baselines for the fine threshold. For the medium and coarse threshold, our methods offer the best pose accuracy-memory tradeoff for model sizes exceeding 5MB. For smaller models, the simple *Track Length* baseline offers a (slightly) better tradeoff. In practice, this is not much of an issue: In order to further reduce the memory footprint, one typically also compresses the descriptors. For higher compression rates, this often reduces pose accuracy. Thus, it can be better to start from a significantly more accurate-but slightly less compact representation and combine it with descriptor compression.

Extended version of Fig. 3 in the main paper. Fig. C provides an extended version of Fig. 3 from the main paper by, in addition to results averaged over all three pose error thresholds, showing results for each individual threshold. The first row repeats the plots shown in the main paper. The second and third row show results for individual thresholds for the Aachen Day (second row) and Aachen Night (third row) queries, using the DLoc pipeline. The fourth and fifth row show results for individual thresholds for the Aachen Day (fourth row) and Aachen Night (fifth row) queries, using the RLoc pipeline.

As can be seen, using TTS improves performance for the fine and medium pose thresholds ((25cm, 2°) respectively (50cm, 5°)) compared to *Max. Feats.*, independently of which features are used. For the coarse threshold (10m, 10°), TTS performs similar or better than *Max. Feats.* Clearly, the results validate the conclusions drawn in the main paper.

Extended versions of Fig. 4 in the main paper. Fig. 4 in the main paper showed that the TTS strategy can readily be combined with the *Max. Feats.* scheme. To this end, we showed results for aliked features, reporting averages over the three pose error thresholds of the Aachen dataset. Fig. D extends Fig. 3 from the main paper by showing results for the individual thresholds. In the first row, we again show the averaged results for reference.

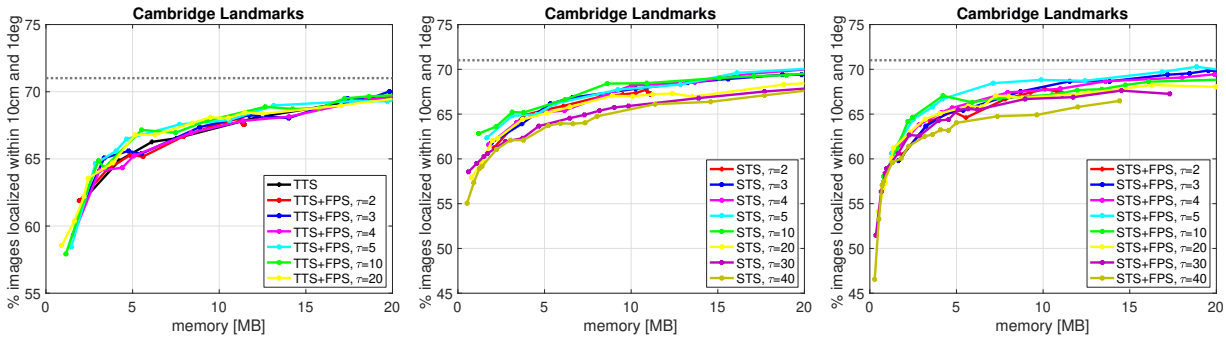

Figure A. **The impact of the parameter** τ the on pose accuracy and memory consumption of the TTS and STS approaches, **with and without the FPS strategy**. Larger values for τ allow for higher compression rates and thus lower memory footprints, at the cost of reduced performance. Both $\tau = 5$ and $\tau = 10$ lead to good results for all three tested approaches (TTS+FPS, STS and STS+FPS). For these values, using the FPS strategy enables stronger compression, while offering a higher pose accuracy at similar memory requirements. The dashed horizontal gray line denotes the pose accuracy achieved when using all 3D points per scene, *i.e.*, without structure compression. Camera poses are computed using RootSIFT features and the DLoc pipeline.

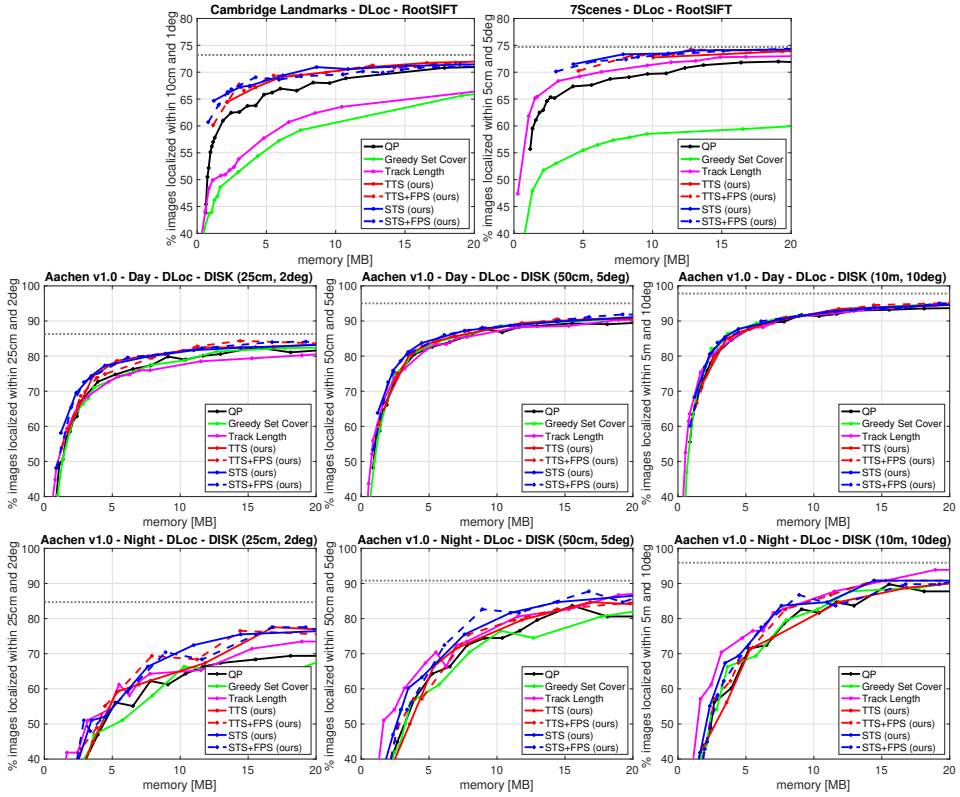

Figure B. **Comparison with structure compression approaches.** For Cambridge and 7Scenes, all of our strategies (TTS, TTS+FPS, STS, STS+FPS) clearly offer a better memory-pose accuracy trade-off. For the fine pose threshold (25cm, 2°) of Aachen, our approaches outperform the baselines. For the medium threshold (50cm, 5°), all strategies perform similarly well for Aachen Day. For Aachen Night, our approaches offer the best pose accuracy-memory tradeoff from about 5MB onwards. For smaller memory footprints, the simple *Track Length* baseline performs slightly better. We make similar observations as for the medium threshold also for the coarse threshold (5m, 10°). The horizontal line denotes the pose accuracy achieved by DLoc without compression.

Figures E and F show corresponding plots obtained with DISK respectively SuperPoint features. We observe the same behavior, *i.e.*, applying our TTS approach on top of the *Max. Feats.* scheme improves performance if 128 or more features are extracted per database image.

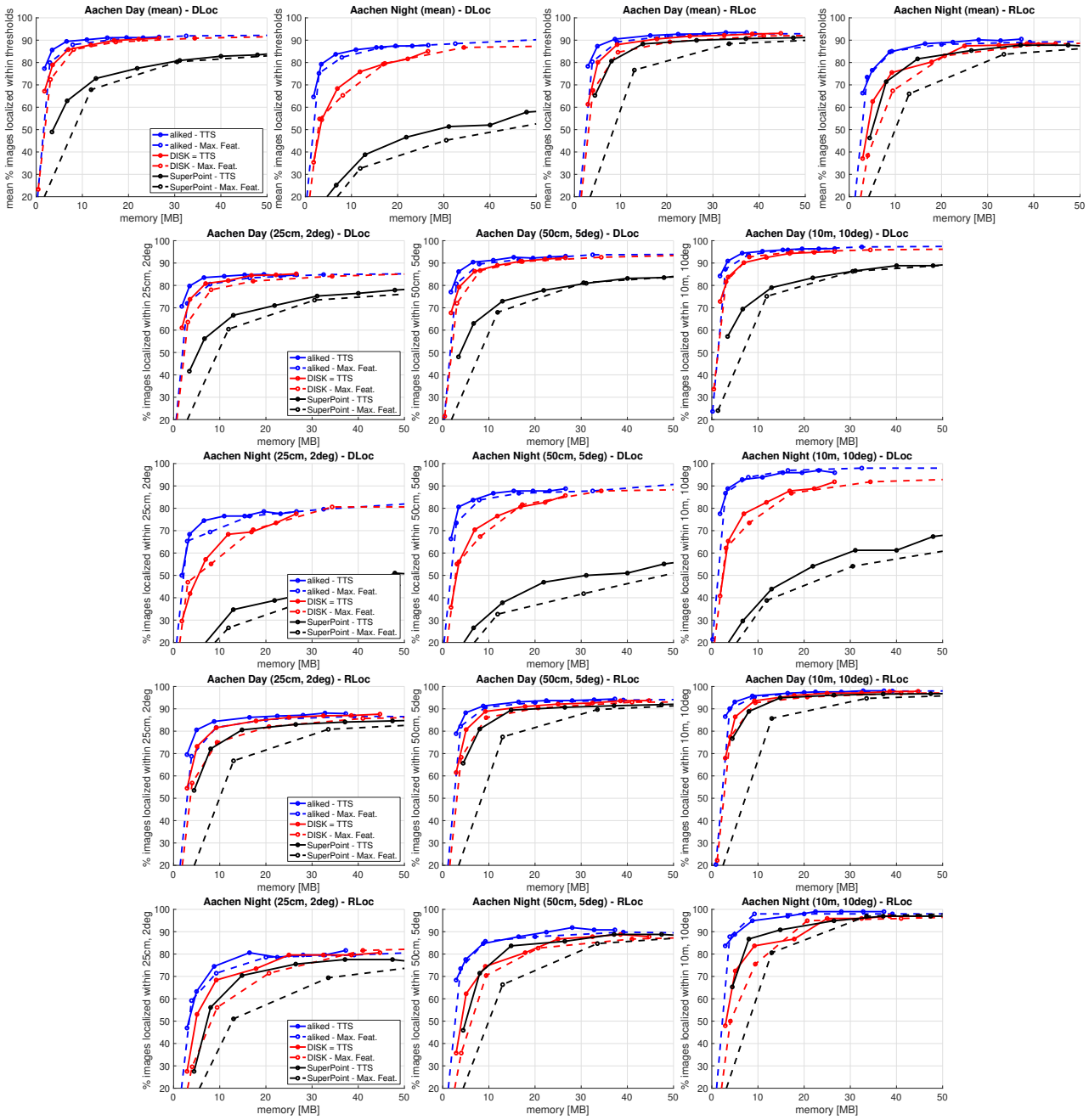

Figure C. **Comparison with extracting a maximum number of features per database image (*Max. Feats.*).** We report the mean recall over all three pose error thresholds, as well as results for the individual thresholds, for Aachen. Our TTS approach performs similar or better than the *Max. Feats.* strategy [10], for all three local features and both localization pipelines. The drop in pose accuracy observed for SuperPoint inside the DLoc pipeline is not due to TTS as the good performance inside RLoc shows. Rather, SuperPoint descriptors are not globally unique enough for direct matching.

C.3. Using Descriptor Compression

In this part, we study the impact of descriptor compression on our structure compression approaches.

Ablation study for descriptor compression via PCA. For the Cambridge Landmarks and 7Scenes datasets, we compress descriptors by projecting them into a lower-dimensional space via PCA, followed by quantizing each descriptor entry using 4 bits (via a Scalar quantizer from the Faiss library). We found this simple choice for compressing descriptors to already work well for both datasets.

In Tables 1 and 2 of the main paper, we project the de-

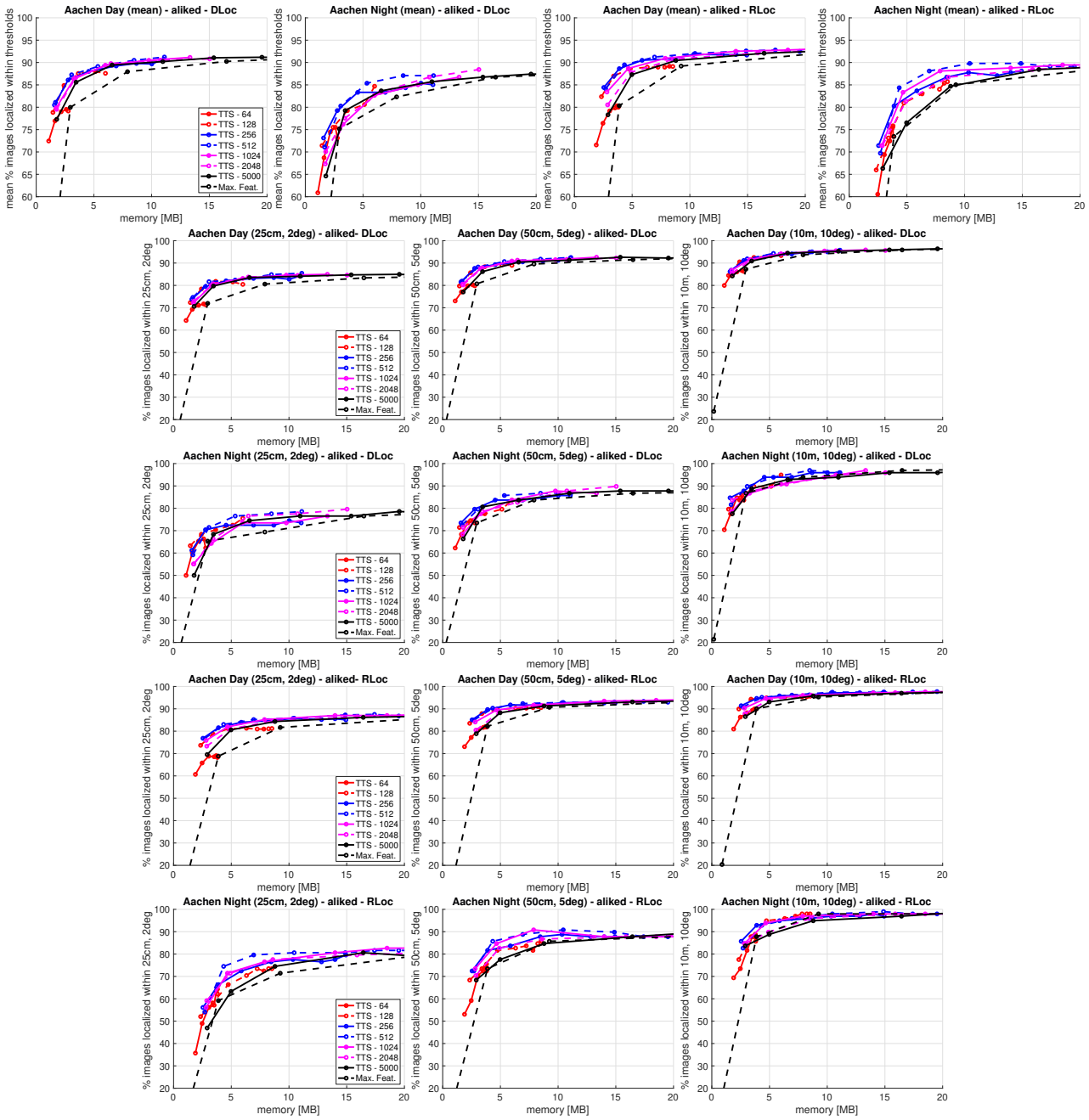

Figure D. **Combining TTS with the *Max. Feats.* strategy using aliked features.** We report the mean recall over all three pose error thresholds, as well as results for the individual thresholds, when extracting at most 64, 128, 256, 512, 1024, 2048, or 5000 features per database image. Applying our triplet-based approach (TTS) on top of *Max. Feats.* further improves performance. The black line (TTS - 5000) corresponds to the lines "aliked - TTS" (blue full lines) in Fig. C.

scriptors into a 16-dimensional space. Tab. B justifies this choice by evaluating the memory-pose accuracy tradeoff for varying numbers of dimensions and varying values for N. Results are obtained using RootSIFT, TTS, and the DLoc pipeline.

Looking at the results for all scenes (last column), using a 16-dimensional space leads to more accurate poses compared to an 8-dimensional space. Using a higher-dimensional space slightly improves pose accuracy, at the cost of a higher memory consumption. Based on the results, we thus used a 16-dimensional space for the experiments in the paper.

For completeness, Tab. C shows results for all four of our variants using a projection into a 16-dimensional space.

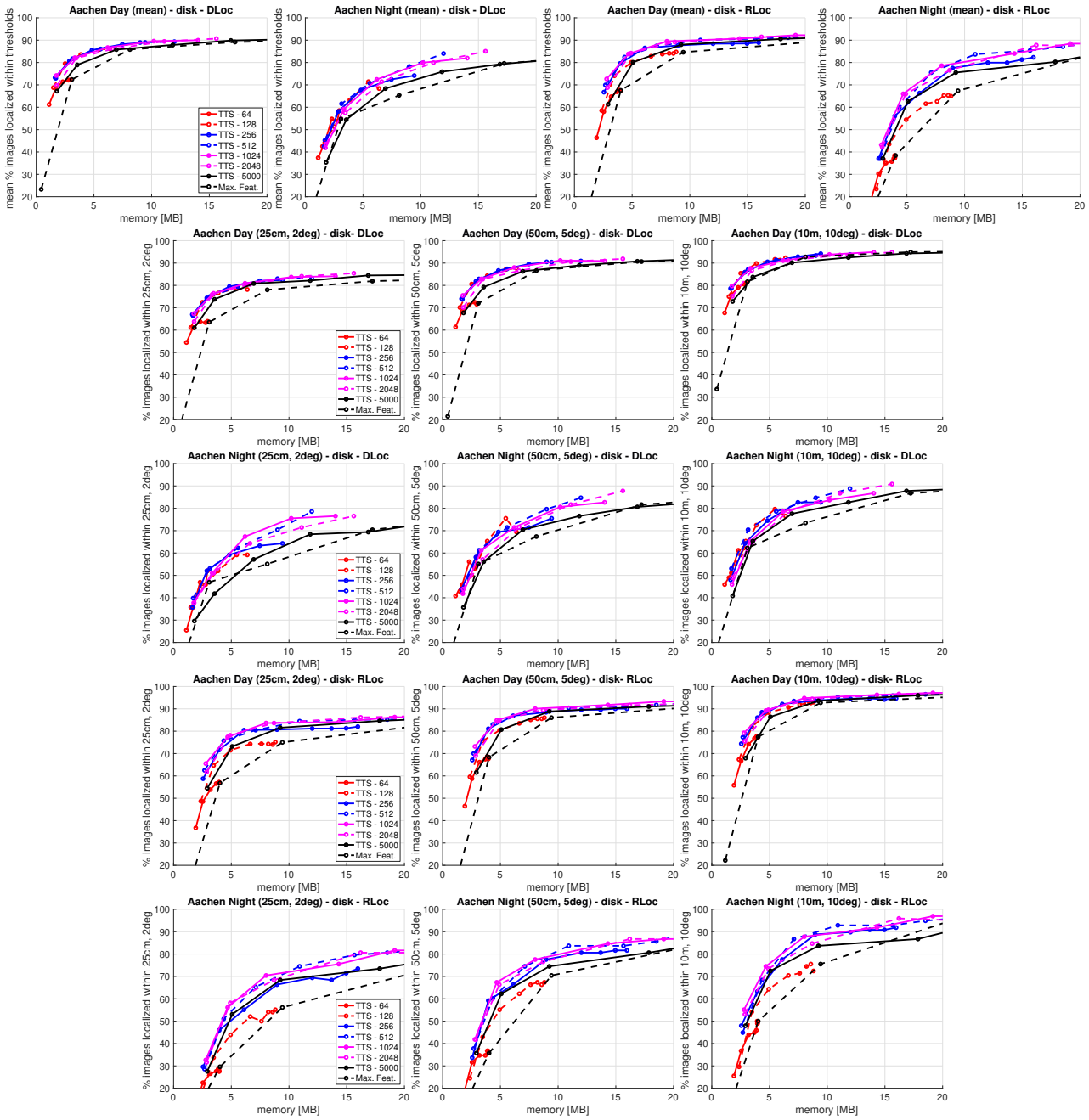

Figure E. **Combining TTS with the *Max. Feats.* strategy using DISK features.** We report the mean recall over all three pose error thresholds, as well as results for the individual thresholds, when extracting at most 64, 128, 256, 512, 1024, 2048, or 5000 features per database image. Applying our triplet-based approach (TTS) on top of *Max. Feats.* further improves performance. The black line (TTS - 5000) corresponds to the lines "DISK - TTS" (red full lines) in Fig. C.

Again, for a given value of N, we observe that STS and FPS reduce memory requirements compared to TTS at the cost of localization performance. This table extends the results of our methods shown in Tab. 2 of the main paper by adding results for varying N for each of our methods.

Tab. D extends the results reported for our approaches in Tab. 1 of the main paper. We report results for all four of our variants. Note that due to Cambridge Landmarks being an easy dataset, we only use $N = 1$ as this setting already provides state-of-the-art results (see Tab. 1 in the main paper).

Ablation study for product quantization settings. Tab. E provides an ablation study for different settings for the product quantizer used in Sec. 4 of the main paper for experi-

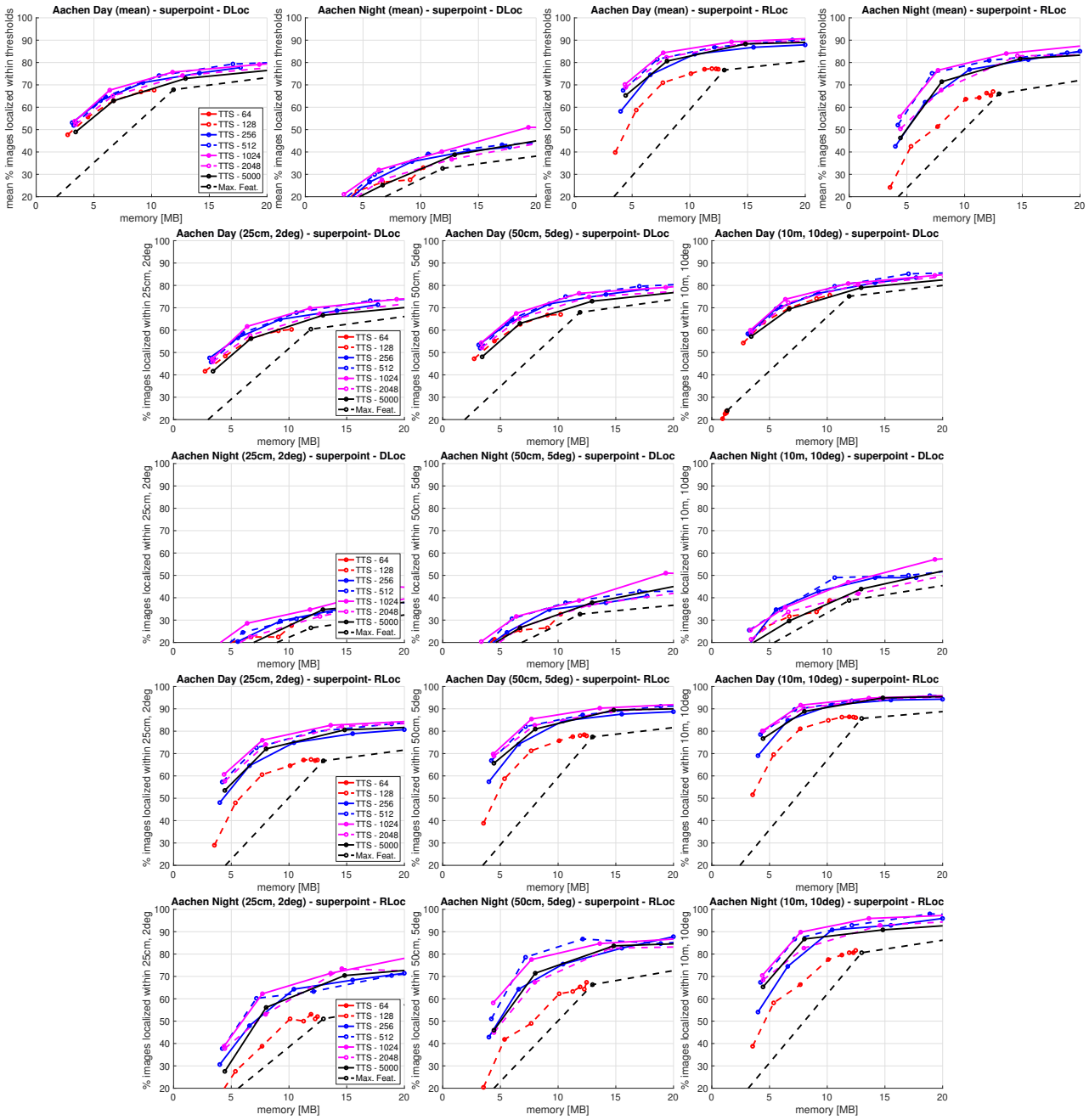

Figure F. **Combining TTS with the *Max. Feats.* strategy using SuperPoint features.** We report the mean recall over all three pose error thresholds, as well as results for the individual thresholds, when extracting at most 64, 128, 256, 512, 1024, 2048, or 5000 features per database image. Applying our triplet-based approach (TTS) on top of *Max. Feats.* further improves performance. The black line (TTS - 5000) corresponds to the lines "SuperPoint - TTS" (black full lines) in Fig. C.

ments on the Aachen Day-Night v1.0 dataset. As can be seen, splitting aliked descriptors into 32 parts and representing each by 10 bits (PQ32x10) typically results in the best localization performance for both localization pipelines, even though it also results in the highest memory requirements. This motivated us to use the PQ32x10 setting for Tab. 3 in the main paper. Note that this setting uses 40 bytes per descriptor. In contrast, MAD-DR [10] uses 8 or 16 bytes. Still, as Tab. 4 in the main paper shows, our approach offers a better trade-off between memory consumption and camera pose accuracy. This results clearly shows the importance of structure compression.

For completeness, Tab. F shows results for all four of our variants using the PQ32x10 setting. Again, we observe

PCA	N	Chess		Fire		Heads		Office		Pumpkin		Kitchen		Stairs		All		
		MB	%	MB	%	MB	%	MB	%	MB	%	MB	%	MB	%	MB	cm / °	%
8	1	0.18	84.2	0.05	92.0	0.10	84.8	0.26	63.8	0.18	39.5	0.32	39.1	0.09	43.7	1.18	4.0 / 1.29	63.9
	5	0.33	85.2	0.08	95.7	0.18	86.6	0.47	66.5	0.32	41.0	0.60	43.0	0.16	56.1	2.14	3.7 / 1.17	67.7
	6	0.34	85.0	0.09	95.9	0.18	87.2	0.49	67.2	0.34	39.9	0.63	41.9	0.17	56.7	2.24	3.7 / 1.17	67.7
	10	0.58	86.2	0.14	97.4	0.32	89.2	0.81	66.5	0.55	42.5	1.10	43.6	0.26	60.8	3.76	3.7 / 1.14	69.5
16	1	0.23	85.7	0.06	96.4	0.12	90.6	0.33	69.7	0.23	43.3	0.41	46.7	0.11	62.8	1.49	3.6 / 1.14	70.8
	5	0.42	87.0	0.10	98.3	0.22	90.9	0.59	71.7	0.40	42.6	0.76	48.1	0.20	68.1	2.69	3.3 / 1.07	72.4
	6	0.43	86.8	0.11	99.3	0.23	90.8	0.62	71.9	0.43	43.0	0.79	48.3	0.21	71.2	2.82	3.3 / 1.08	73.0
	10	0.73	87.1	0.18	99.7	0.41	91.5	1.01	71.0	0.69	43.8	1.37	48.9	0.33	71.5	4.72	3.4 / 1.08	73.4
32	1	0.33	85.2	0.09	97.4	0.18	90.7	0.47	71.5	0.32	42.8	0.57	48.3	0.16	68.0	2.12	3.6 / 1.11	72.0
	5	0.59	86.5	0.15	99.2	0.32	90.5	0.83	73.3	0.56	43.2	1.06	48.6	0.28	73.5	3.80	3.3 / 1.07	73.5
	6	0.61	86.9	0.16	99.4	0.32	90.2	0.87	73.1	0.61	43.6	1.11	49.2	0.30	73.2	3.99	3.3 / 1.06	73.7
	10	1.03	87.0	0.26	99.9	0.57	91.1	1.42	72.5	0.97	44.6	1.93	50.1	0.47	75.2	6.64	3.3 / 1.07	74.3

Table B. **Ablation study on the impact of PCA dimensionality reduction on the 7Scenes dataset.** We report the memory requirements (in MB) and the percentage of images localized within 5cm, 5° of the ground truth using TTS and the DLoc pipeline. As a summary, we sum up the memory requirements over all scenes and show the mean percentage of successfully localized images. We vary the number of descriptor dimensions (PCA) obtained via PCA, as well as the parameter N controlling the number of selected points. Each descriptor entry is stored using 4 bits (using a Scalar quantizer from the Faiss library).

	Chess		Fire		Heads		Office		Pumpkin		Kitchen		Stairs		All		
	MB	%	MB	%	MB	%	MB	%	MB	%	MB	%	MB	%	MB	cm / °	%
TTS, $N = 1$ (**ours**)	0.23	85.7	0.06	96.4	0.12	90.6	0.33	69.7	0.23	43.3	0.41	46.7	0.11	62.8	1.49	4 / 1.14	70.8
TTS, $N = 5$ (**ours**)	0.42	87.0	0.10	98.3	0.22	90.9	0.59	71.7	0.40	42.6	0.76	48.1	0.20	68.1	2.69	3 / 1.07	72.4
TTS, $N = 6$ (**ours**)	0.43	86.8	0.11	99.3	0.23	90.8	0.62	71.9	0.43	43.0	0.79	48.3	0.21	71.2	2.82	3 / 1.08	73.0
TTS, $N = 10$ (**ours**)	0.73	87.1	0.18	99.7	0.41	91.5	1.01	71.0	0.69	43.8	1.37	48.9	0.33	71.5	4.72	3 / 1.08	73.4
STS, $N = 1$ (**ours**)	0.10	85.0	0.03	97.0	0.05	92.9	0.14	68.0	0.11	43.7	0.20	45.0	0.05	53.6	0.67	4 / 1.17	69.3
STS, $N = 5$ (**ours**)	0.17	84.0	0.04	98.8	0.08	93.5	0.23	69.4	0.18	42.9	0.35	46.3	0.08	66.4	1.12	3 / 1.11	71.6
STS, $N = 6$ (**ours**)	0.17	84.7	0.04	99.3	0.08	93.2	0.24	69.6	0.19	42.3	0.37	46.8	0.09	65.9	1.18	3 / 1.10	71.7
STS, $N = 10$ (**ours**)	0.28	85.8	0.06	98.6	0.13	91.1	0.38	70.9	0.31	44.0	0.62	48.7	0.13	67.9	1.91	3 / 1.09	72.4
TTS+FPS, $N = 1$ (**ours**)	0.10	84.5	0.03	93.1	0.06	89.8	0.14	69.3	0.11	43.5	0.22	43.0	0.06	51.7	0.72	4 / 1.17	67.9
TTS+FPS, $N = 5$ (**ours**)	0.17	85.5	0.04	97.0	0.10	90.8	0.23	72.4	0.17	44.0	0.36	46.7	0.00	0.0	1.07	3 / 0.96	62.3
TTS+FPS, $N = 6$ (**ours**)	0.18	86.1	0.05	99.2	0.10	91.7	0.25	70.5	0.19	41.5	0.40	47.4	0.10	62.6	1.27	3 / 1.12	71.3
TTS+FPS, $N = 10$ (**ours**)	0.26	86.5	0.07	98.5	0.16	92.5	0.36	71.9	0.28	43.5	0.61	48.4	0.14	67.0	1.88	3 / 1.08	72.6
STS+FPS, $N = 1$ (**ours**)	0.07	82.2	0.02	92.2	0.04	91.1	0.10	67.3	0.08	41.5	0.15	43.1	0.04	50.6	0.49	4 / 1.20	66.9
STS+FPS, $N = 5$ (**ours**)	0.11	85.4	0.03	93.2	0.06	91.0	0.16	70.1	0.12	41.9	0.25	43.5	0.06	57.5	0.80	4 / 1.14	68.9
STS+FPS, $N = 6$ (**ours**)	0.12	84.6	0.04	95.9	0.06	92.2	0.17	70.0	0.14	42.4	0.28	45.8	0.07	61.4	0.89	3 / 1.13	70.3
STS+FPS, $N = 10$ (**ours**)	0.19	85.0	0.05	96.5	0.10	91.5	0.26	70.9	0.21	41.3	0.44	48.4	0.10	64.0	1.34	3 / 1.11	71.1

Table C. **Comparison of all of our variants on the 7Scenes dataset.** We report the memory requirements (in MB) and the percentage of images localized within 5cm, 5° of the ground truth. As a summary, we sum up the memory requirements over all scenes and show the mean percentage of successfully localized images. We also report the mean of the median position (in cm) and orientation (in °) errors. Descriptors are compressed by projecting them into a 16-dimensional space via PCA, followed by quantizing each entry using 4 bits. For each of our variants, we show its memory-pose accuracy tradeoff by showing results for different values of N. Results are obtained with RootSIFT and the DLoc pipeline.

	Great Court				King's College				Old Hospital				Shop Facade				St. Mary's Church			
method	[MB]	[cm]	[°]	[%]	[MB]	[cm]	[°]	[%]	[MB]	[cm]	[°]	[%]	[MB]	[cm]	[°]	[%]	[MB]	[cm]	[°]	[%]
TTS, $N = 1$ (**ours**)	0.10	12	0.06	46.6	0.08	9	0.12	58.0	0.06	13	0.23	44.5	0.02	3	0.12	87.4	0.09	4	0.14	80.2
STS, $N = 1$ (**ours**)	0.06	10	0.05	50.0	0.04	10	0.12	46.1	0.04	13	0.23	42.3	0.02	3	0.12	86.4	0.06	5	0.15	82.3
TTS+FPS, $N = 1$ (**ours**)	0.06	12	0.06	43.8	0.04	10	0.14	46.9	0.04	16	0.26	39.6	0.02	4	0.16	80.6	0.05	5	0.17	74.7
STS+FPS, $N = 1$ (**ours**)	0.04	12	0.06	43.5	0.03	11	0.13	44.3	0.03	14	0.26	40.1	0.01	4	0.14	80.6	0.04	6	0.18	73.4

Table D. **Comparison of all of our variants on the Cambridge Landmarks dataset.** We report the memory requirements (in MB), the median positon (in cm) and orientation (in °) errors, and the percentage of images localized within 10cm, 1° of the ground truth. We report results for RootSIFT and the DLoc pipeline. Descriptors are compressed by projecting them into a 16-dimensional space, followed by quantizing each descriptor entry to 4 bits using a Scalar quantizer from the Faiss library. All of our variants take radial distortion in the database and query images into account.

that STS and FPS reduce memory requirements compared to TTS at the cost of localization performance. This table extends the results of our methods shown in Tab. 3 of the main paper.

D. Localization Times

Finally, we report localization times for the DLoc pipeline when using the TTS strategy for structure compression. All times were measured on a desktop PC with an Intel Core

localizer	quantizer	N	mem. [MB]	% images localized within 25cm,2° / 50cm,5° / 5m,10° day	night
DLoc	PQ32x4	1	0.338	71.24 / 77.43 / 81.67	47.96 / 56.12 / 66.33
	PQ32x8	1	0.591	73.3 / 81.07 / 85.32	57.14 / 68.37 / 76.53
	PQ16x10	1	0.644	73.18 / 80.46 / 85.44	56.12 / 67.35 / 73.47
	PQ32x10	1	0.883	74.39 / 82.04 / 86.04	62.24 / 70.41 / 79.59
	PQ32x4	5	0.617	77.79 / 84.1 / 89.08	61.22 / 70.41 / 76.53
	PQ16x10	5	0.963	79.0 / 85.8 / 89.68	68.37 / 77.55 / 84.69
	PQ32x8	5	1.029	80.58 / 87.99 / 91.63	69.39 / 78.57 / 87.76
	PQ32x10	5	1.401	79.85 / 86.65 / 90.66	70.41 / 79.59 / 90.82
	PQ32x4	10	1.072	79.85 / 87.99 / 91.75	67.35 / 77.55 / 86.73
	PQ16x10	10	1.483	82.52 / 89.32 / 92.72	72.45 / 82.65 / 89.8
	PQ32x8	10	1.744	82.16 / 90.29 / 93.57	75.51 / 89.8 / 94.9
	PQ32x10	10	2.245	83.13 / 90.29 / 93.57	73.47 / 82.65 / 90.82
	PQ32x4	20	1.699	81.92 / 89.68 / 92.84	70.41 / 81.63 / 89.8
	PQ16x10	20	2.199	82.89 / 90.17 / 94.42	73.47 / 86.73 / 93.88
	PQ32x8	20	2.729	83.5 / 91.26 / 94.42	74.49 / 86.73 / 96.94
	PQ32x10	20	3.410	84.83 / 91.38 / 94.42	74.49 / 85.71 / 93.88
RLoc	PQ32x4	1	1.369	71.24 / 80.1 / 85.19	46.94 / 62.24 / 71.43
	PQ32x8	1	1.622	76.7 / 84.95 / 90.78	54.08 / 68.37 / 80.61
	PQ16x10	1	1.675	75.61 / 83.37 / 88.96	50.0 / 63.27 / 79.59
	PQ32x10	1	1.914	76.21 / 84.71 / 91.26	50.0 / 63.27 / 79.59
	PQ32x4	5	1.885	79.98 / 87.86 / 92.84	61.22 / 70.41 / 87.76
	PQ16x10	5	2.230	82.04 / 89.44 / 94.05	67.35 / 83.67 / 92.86
	PQ32x8	5	2.297	82.77 / 90.17 / 94.42	71.43 / 79.59 / 90.82
	PQ32x10	5	2.668	83.01 / 90.41 / 94.9	77.55 / 83.67 / 92.86
	PQ32x4	10	2.670	84.1 / 90.78 / 95.51	73.47 / 81.63 / 95.92
	PQ16x10	10	3.080	84.47 / 91.26 / 95.51	76.53 / 87.76 / 94.9
	PQ32x8	10	3.341	85.07 / 92.11 / 95.87	76.53 / 87.76 / 95.92
	PQ32x10	10	3.843	85.07 / 92.23 / 95.87	80.61 / 87.76 / 95.92
	PQ32x4	20	3.668	85.32 / 92.23 / 96.6	72.45 / 84.69 / 96.94
	PQ16x10	20	4.168	86.04 / 92.23 / 96.84	77.55 / 89.8 / 97.96
	PQ32x8	20	4.698	86.65 / 92.6 / 97.09	77.55 / 89.8 / 96.94
	PQ32x10	20	5.379	86.53 / 92.84 / 97.33	79.59 / 88.78 / 96.94

Table E. **Ablation study on the impact of different descriptor compression schemes on the performance of TTS on the Aachen Day-Night v1.0 dataset**. We vary the parameter N as well as the compression schemes and report results for both DLoc and RLoc using aliked features. PQMxK denotes using product quantization for descriptor quantization, where the original descriptor is split into M parts and each part is stored using K bits. We apply TTS on top of the *Max. Feats.* scheme, detecing 512 descriptors per database image.

i5-10600 CPU running at 3.30GHz, 48GB RAM, and an NVidia GeForce RTX 3060 GPU with 12GB of memory. The code was running on Ubuntu 20.04.6 inside WSL.

Tab. G extends the results of our approach from Tab. 1 in the main paper by reporting localization times (including descriptor matching and pose estimation, but excluding feature extraction) measured when varying the maximum number of RANSAC iterations and the ratio test threshold. Note that results can vary compared to Tab. 1 in the main paper due to the randomness in RANSAC as we ran all variants again to allow for a fair comparison of timings.[7]

Similarly, Tab. H extend Tab. 3 in the main paper by adding run-time results. Note that for the Aachen dataset, we report the average localization time over all images and do not break down localization times per condition.

[7]The Results reported in the main paper were obtained by running experiments in parallel, which affects timings.

References

[1] Gabriele Berton, Gabriele Trivigno, Barbara Caputo, and Carlo Masone. Eigenplaces: Training viewpoint robust models for visual place recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 4

[2] Ondrej Chum and Jiri Matas. Matching with PROSAC - Progressive Sampling Consensus. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 4

[3] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library. *arXiv:2401.08281*, 2024. 3

[4] Viktor Larsson and contributors. PoseLib - Minimal Solvers for Camera Pose Estimation, 2020. 4

[5] Karel Lebeda, Juan E. Sala Matas, and Ondřej Chum. Fixing the Locally Optimized RANSAC. In *BMVC*, 2012. 4

[6] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *ICCV*, 2023. 4

[7] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 2004. 3

[8] Mikael Persson and Klas Nordberg. Lambda twist: An accurate fast robust perspective three point (p3p) solver. In *ECCV*, 2018. 4

[9] T. Sattler, B. Leibe, and L. Kobbelt. Fast Image-Based Localization using Direct 2D-to-3D Matching. In *ICCV*, 2011. 3

[10] Qiang Wang. MAD-DR: Map Compression for Visual Localization with Matchness Aware Descriptor Dimension Reduction. In *ECCV*, 2024. 7, 10

[11] Xiaoming Zhao, Xingming Wu, Jinyu Miao, Weihai Chen, Peter C. Y. Chen, and Zhengguo Li. Alike: Accurate and lightweight keypoint detection and descriptor extraction. *IEEE Transactions on Multimedia*, 2022. 14

[12] Xiaoming Zhao, Xingming Wu, Weihai Chen, Peter C. Y. Chen, Qingsong Xu, and Zhengguo Li. Aliked: A lighter keypoint and descriptor extraction network via deformable transformation. *IEEE Transactions on Instrumentation & Measurement*, 72:1–16, 2023. 14

method	#points	mem. [MB]	% images localized within 25cm,2° / 50cm,5° / 5m,10° day	night
TTS (512), $N = 10$, RLoc (**ours**)	38,139	3.8	85.1 / 92.2 / 95.9	80.6 / 87.8 / 95.9
TTS (512), $N = 5$, RLoc (**ours**)	21,904	2.7	83.0 / 90.4 / 94.9	77.6 / 83.7 / 92.9
TTS (512), $N = 1$, RLoc (**ours**)	11,934	1.9	76.2 / 84.7 / 91.3	50.0 / 63.3 / 79.6
TTS (512), $N = 10$, DLoc (**ours**)	38,139	2.2	83.1 / 90.3 / 93.6	73.5 82.7 / 90.8
TTS (512), $N = 5$, DLoc (**ours**)	21,904	1.4	79.9 / 86.7 / 90.7	70.4 / 79.6 90.8
TTS (512), $N = 1$, DLoc (**ours**)	11,934	0.9	74.4 / 82.0 / 86.0	62.2 / 70.4 / 79.6
STS (512), $N = 10$, RLoc (**ours**)	23,582	2.9	84.0 / 91.1 / 95.6	79.6 / 86.7 / 98.0
STS (512), $N = 5$, RLoc (**ours**)	13,734	2.1	81.4 / 89.4 / 94.0	70.4 / 82.7 / 91.8
STS (512), $N = 1$, RLoc (**ours**)	7,848	1.6	72.9 / 84.1 / 89.3	58.2 / 72.5 / 83.7
STS (512), $N = 10$, DLoc (**ours**)	23,582	1.5	81.0 / 88.5 / 92.5	66.3 / 77.5 / 89.8
STS (512), $N = 5$, DLoc (**ours**)	13,734	1.0	76.6 / 85.1 / 89.6	64.3 / 76.5 / 85.7
STS (512), $N = 1$, DLoc (**ours**)	7,848	0.7	69.9 / 78.2 / 83.7	58.2 / 71.4 / 78.6
TTS+FPS (512), $N = 10$, RLoc (**ours**)	17,570	2.3	81.9 / 90.2 / 95.3	68.4 / 81.6 / 91.8
TTS+FPS (512), $N = 5$, RLoc (**ours**)	10,346	1.8	75.8 / 85.6 / 91.1	54.1 / 71.4 / 83.7
TTS+FPS (512), $N = 1$, RLoc (**ours**)	6,237	1.5	65.4 / 76.5 / 83.9	37.8 / 54.1 / 66.3
TTS+FPS (512), $N = 10$, DLoc (**ours**)	17,570	1.2	77.7 / 85.2 / 89.9	69.4 / 78.6 / 88.8
TTS+FPS (512), $N = 5$, DLoc (**ours**)	10,346	0.8	73.4 / 81.8 / 86.5	63.3 / 70.4 / 80.6
TTS+FPS (512), $N = 1$, DLoc (**ours**)	6,237	0.8	73.4 / 81.8 / 86.5	63.3 / 70.4 / 80.6
STS+FPS (512), $N = 10$, RLoc (**ours**)	15,064	2.1	82.0 / 89.7 / 94.9	67.3 / 82.7 / 93.9
STS+FPS (512), $N = 5$, RLoc (**ours**)	8,543	1.7	76.3 / 84.6 / 90.7	51.0 / 63.3 / 78.6
STS+FPS (512), $N = 1$, RLoc (**ours**)	4,976	1.4	63.6 / 74.5 / 82.2	42.9 / 54.1 / 62.2
STS+FPS (512), $N = 10$, DLoc (**ours**)	15,064	1.0	78.6 / 86.8 / 90.2	61.2 / 72.5 / 84.7
STS+FPS (512), $N = 5$, DLoc (**ours**)	8,543	0.7	72.2 / 79.2 / 84.7	58.2 / 66.3 / 76.5
STS+FPS (512), $N = 1$, DLoc (**ours**)	4,976	0.5	58.9 / 67.3 / 72.5	40.8 / 50.0 / 58.2

Table F. **Comparison of all of our variants on the Aachen Day-Night v1.0 dataset**. We use aliked features and product quantization of descriptors into 32 parts, each quantized using 10 bits. (512) indicates extracting at most 512 features per database image, *i.e.*, using the *Max. Feats.* strategy with at most 512 features per database image. We also report the number of 3D points (#points) selected by the different schemes.

	Great Court					King's College					Old Hospital					Shop Facade					St. Mary's Church				
method	[MB]	[cm]	[°]	[%]	[s]	[MB]	[cm]	[°]	[%]	[s]	[MB]	[cm]	[°]	[%]	[s]	[MB]	[cm]	[°]	[%]	[s]	[MB]	[cm]	[°]	[%]	[s]
TTS, 0.8, 1k	0.1	13	0.06	42.6	0.022	0.08	8	0.11	58.3	0.032	0.06	15	0.28	39.56	0.035	0.02	3	0.11	86.4	0.016	0.09	4	0.15	78.7	0.038
TTS, 0.8, 10k	0.1	13	0.06	43.8	0.052	0.08	8	0.11	58.6	0.062	0.06	15	0.27	40.1	0.112	0.02	3	0.11	86.4	0.072	0.09	4	0.15	79.4	0.089
TTS, 0.8, 100k	0.1	12	0.06	45.3	0.127	0.08	9	0.11	58.0	0.074	0.06	13	0.26	42.3	0.610	0.02	3	0.11	87.4	0.552	0.09	4	0.14	79.8	0.337
TTS, 0.9, 1k	0.1	13	0.06	44.9	0.030	0.08	9	0.12	58.0	0.045	0.06	15	0.25	42.3	0.053	0.02	3	0.12	86.4	0.024	0.09	4	0.14	79.3	0.048
TTS, 0.9, 10k	0.1	12	0.06	45.0	0.103	0.08	9	0.12	57.7	0.157	0.06	14	0.25	43.4	0.215	0.02	3	0.12	86.4	0.131	0.09	4	0.14	79.6	0.164
TTS, 0.9, 100k	0.1	12	0.06	46.7	0.550	0.08	9	0.12	58.6	0.548	0.06	13	0.23	44.5	1.488	0.02	3	0.12	87.4	1.074	0.09	4	0.14	80.2	1.117

Table G. **Localization times for the DLoc pipeline on the Cambridge Landmarks dataset for TTS-based structure compression.** We set $N = 1$ and compress descriptors via PCA to 16 dimensions, representing each descriptor entry using 4 bits. We report the memory requirements (in MB), the median positon (in cm) and orientation (in °) errors, the percentage of images localized within 10cm, 1° of the ground truth, and the average localization times (in seconds, including 2D-3D matching and pose estimation via RANSAC, but excluding feature extraction). We take radial distortion into account. We compare results obtained with different ratio test thresholds (0.8 and 0.9) and different maximum numbers of RANSAC iterations (1k, 10k, 100k).

method	mem. [MB]	time [s]	% images localized within 25cm,2° / 50cm,5° / 5m,10° day	night
TTS, $N = 10$, 0.8, 1k	2.2	2.537	81.6 / 90.1 / 92.8	69.4 / 81.6 / 87.8
TTS, $N = 10$, 0.8, 10k	2.2	2.604	82.0 / 90.4 / 93.5	72.5 / 86.7 / 91.8
TTS, $N = 10$, 0.8, 100k	2.2	2.660	82.7 / 91.3 / 94.4	74.5 / 87.8 / 94.9
TTS, $N = 10$, 0.9, 1k	2.2	2.609	82.0 / 89.6 / 93.0	71.4 / 82.7 / 86.73
TTS, $N = 10$, 0.9, 10k	2.2	2.676	82.0 / 89.4 / 93.0	72.5 / 82.7 / 87.8
TTS, $N = 10$, 0.9, 100k	2.2	3.055	83.1 / 90.4 / 93.8	73.5 / 83.7 / 89.8
TTS, $N = 5$, 0.8, 1k	1.4	1.695	79.4 / 86.2 / 90.5	64.3 / 73.5 / 85.7
TTS, $N = 5$, 0.8, 10k	1.4	1.721	80.0 / 87.0 / 90.8	64.3 / 73.5 / 86.7
TTS, $N = 5$, 0.8, 100k	1.4	1.838	81.7 / 88.2 / 93.1	67.4 / 77.6 / 91.8
TTS, $N = 5$, 0.9, 1k	1.4	1.666	79.7 / 86.3 / 89.7	67.4 / 74.5 / 87.8
TTS, $N = 5$, 0.9, 10k	1.4	1.745	79.6 / 86.2 / 89.8	69.4 / 76.5 / 88.8
TTS, $N = 5$, 0.9, 100k	1.4	2.353	80.2 / 87.0 / 91.1	69.4 / 78.6 / 90.8
TTS, $N = 1$, 0.8, 1k	0.9	1.106	74.5 / 82.2 / 87.0	59.2 / 69.4 / 77.6
TTS, $N = 1$, 0.8, 10k	0.9	1.145	74.6 / 82.7 / 87.6	61.2 / 70.4 / 80.6
TTS, $N = 1$, 0.8, 100k	0.9	1.447	74.5 / 82.5 / 87.9	61.2 / 71.4 / 80.6
TTS, $N = 1$, 0.9, 1k	0.9	1.110	74.3 / 82.2 / 86.4	61.2 / 68.4 / 77.6
TTS, $N = 1$, 0.9, 10k	0.9	1.186	74.0 / 81.8 / 86.2	62.2 / 69.4 / 76.5
TTS, $N = 1$, 0.9, 100k	0.9	1.986	74.6 / 82.3 / 86.8	61.2 / 71.4 / 80.6

Table H. **Localization times for the DLoc pipeline on the Aachen Day-Night dataset for TTS-based structure compression.** We compress descriptors product quantization of descriptors into 32 parts, each quantized using 10 bits. For the database images, we extract at most 512 features per database images. We use aliked features [11, 12] for 2D-3D matching. We report the memory requirements (in MB), the average localization times (in seconds, including 2D-3D matching and pose estimation via RANSAC, but excluding feature extraction), and the percentage of images localized within given thresholds. We compare results obtained with different ratio test thresholds (0.8 and 0.9) and different maximum numbers of RANSAC iterations (1k, 10k, 100k).