SceneSqueezer: Learning to Compress Scene for Camera Relocalization

Luwei Yang[1*] Rakesh Shrestha[1*] Wenbo Li[2] Shuaicheng Liu[3]
Guofeng Zhang[2] Zhaopeng Cui[2†] Ping Tan[1,4†]

[1] Simon Fraser University [2] State Key Lab of CAD & CG, Zhejiang University
[3] University of Electronic Science and Technology of China [4]Alibaba XR Lab

{luweiy, rakeshs, pingtan}@sfu.ca, {wenboli, zhangguofeng, zhpcui}@zju.edu.cn,
liushuaicheng@uestc.edu.cn

Abstract

Standard visual localization methods build a priori 3D model of a scene which is used to establish correspondences against the 2D keypoints in a query image. Storing these pre-built 3D scene models can be prohibitively expensive for large-scale environments, especially on mobile devices with limited storage and communication bandwidth. We design a novel framework that compresses a scene while still maintaining localization accuracy. The scene is compressed in three stages: first, the database frames are clustered using pairwise co-visibility information. Then, a learned point selection module prunes the points in each cluster taking into account the final pose estimation accuracy. In the final stage, the features of the selected points are further compressed using learned quantization. Query image registration is done using only the compressed scene points. To the best of our knowledge, we are the first to propose learned scene compression for visual localization. We also demonstrate the effectiveness and efficiency of our method on various outdoor datasets where it can perform accurate localization with low memory consumption.

1. Introduction

Visual localization aims to estimate the camera pose (*i.e.*, position and rotation) of an RGB query image with respect to a known 3D scene. This is a fundamental problem in 3D computer vision with various applications such as autonomous driving, augmented reality, indoor navigation, etc. Classical visual localization methods build 3D point cloud models of the scene beforehand, where each point is associated with one or more image descriptors, which are used to match against the 2D points detected in the query image. Once the 2D-3D correspondences are established, robust pose estimation [11, 19] can be applied to recover the camera pose. Retaining the pre-built 3D scene models costs expensive memory usage for large-scale environments, especially when deploying on mobile devices with limited storage. Thus, many methods have been proposed to compress 3D scene models to improve scalability while maintaining localization accuracy.

Previous 3D scene compression methods can be roughly divided into three categories. The first category [7, 39] compresses the descriptors of 3D points using quantization, inevitably sacrificing the distinctiveness of the features while offering only a limited amount of compression. The second are learning-based visual localization methods [14, 16, 49]. Without explicitly storing a 3D scene model, they directly regress the camera pose or scene coordinates of each pixel via deep neural networks, which can be considered as a special type of scene compression as the network weights encode the 3D scene information. However, these methods either have low accuracy or generalize poorly. The third and the most common category compresses the 3D models by carefully selecting a subset of 3D points based on certain handcrafted criteria which generally include spatial coverage and visual distinctiveness of the selected points [6, 23, 27]. However, these criteria fail to directly consider the impact of the compression on the final pose estimation. Manually designing rules to select points is known to be a challenging problem for both absolute and relative pose estimation [13].

Motivated by the latest hierarchical localization methods [36, 38] which lead the benchmarks for visual localization, we propose a novel hierarchical strategy for 3D scene compression. Specifically, we compress the 3D scene model in three levels. At the coarse level, we conduct database image clustering and divide the scene into multiple clusters, enabling us to compress the clusters individually. This also speeds up localization by limiting the matching

*Equal contribution, †Corresponding authors
Project Page: sfu-gruvi-3dv.github.com/s_squeezer

of the query keypoints to a smaller set of 3D points. For the middle level, we solve a QP problem [27] to select 3D points from each cluster. However, different from [27], we design a novel differentiable point selection method that enables us to learn to select 3D points based on their 2D observations and influence on the final pose estimation. To this end, we design a new multi-view observation fuser which learns to extract and aggregate features for each 3D point based on their ability to yield correct matches. With the extracted multi-view features we can learn the distinctiveness scores and pairwise proximity of the points which are used to select a subset of points via a differentiable QP solver [1]. Our experiments show that our learned point selection performs better than the heuristic methods. At the finest level, we exploit the latest differentiable quantization method [12] to further compress the feature descriptors. With these three levels of compression, our system outperforms existing compression-based localization methods.

The contributions of our paper can be summarized as follows. At first, we propose a novel hierarchical pipeline for scene compression taking inspiration from the latest visual hierarchical localization methods. Secondly, we propose a novel differentiable point selection that learns distinctiveness scores and pairwise proximity of scene points. We train it alongside the feature matching and pose estimation, which enables the point selection to be based on the pose estimation accuracy instead of heuristic criteria. Extensive experiments show that our method outperforms previous methods on various benchmark datasets.

2. Related Work

Camera Localization: Conventionally, the problem of camera pose estimation has been tackled by generating 2D-3D correspondences from handcrafted keypoint descriptors [22, 24, 39–41, 45] followed by Perspective-n-Point (PnP) algorithm [11, 19]. Despite impressive performance, these methods are limited to well textured scenes due to the handcrafted features.

Deep learning has been proposed as an alternative to the traditional methods. PoseNet [16] and its follow-up works [14,49] directly regress the absolute camera pose of a query image using CNN and LSTM. However, these methods are more akin to image retrieval than to structure-based pose estimation in terms of performance [43]. More robust learning-based methods [2–4,20,21] instead learn to regress the dense absolute scene coordinates of query images in a scene-specific manner. Despite of some success, they perform poorly on larger scenes and require costly retraining to adapt to novel scenes. Scene-agnostic regression [47,50] alleviate the generalization limitation but are costly in terms of memory as they need to store scene coordinate maps of the entire database.

Many recent methods [9,35,46] exploit deep learning to obtain better feature descriptors and matchers. HLoc [36] achieves impressive results by exploiting the SuperGlue matcher [37] along with a hierarchical, coarse-to-fine approach for pose estimation. PixelLoc [38] learns to extract dense image features to align a query image to database images while also following the hierarchical localization strategy of [37]. All of these methods still require explicit 3D model with large amounts of points to achieve good performance.

Scene Compression: K-cover is a popular technique to reduce the number of 3D scene points. Li *et al*. [23] use K-cover to find a subset of points with even spatial distribution and high visibility. Cao and Snavely [6] extend it and propose a probabilistic K-cover solution to maximize the visual distinctiveness (expressed in terms of descriptor distance) and the spatial coverage of the points. Camposeco *et al*. [5] present a K-cover based hybrid compression algorithm with two sets of points at different descriptor resolutions. Sun *et al*. [8] proposes an adaptive weighted K-cover solution to maximize the selected points' visibility. These K-cover variants are NP-hard and can only be solved approximately.

Another common formulation for scene compression is quadratic programming (QP). Chen *et al*. [44] formulate a mixed-integer QP problem to optimize visibility and spatial distribution. Dymczyk *et al*. [10] improve the scalability of [44] by partitioning the problem into co-visible subgraphs that are solved individually. Mixed-integer QP is non-convex and therefore do not guarantee optimal solution despite being memory/time intensive. Mera-Trujillo *et al*. [27] propose a convex QP formulation and design an efficient solver for their problem formulation. These methods uses handcrafted heuristics to encode the visual distinctiveness which generally need to be tuned or changed altogether based on the dataset.

A different approach to scene compression is feature quantization. Cheng *et al*. [7] propose a cascaded parallel filter to compresses descriptors to compact binary representation. Sattler *et al*. [39] compress scene features into quantized vocabularies.

Learned Point Cloud Sampling: While not always aimed for localization, point cloud subsampling is a related topic. Earlier methods use farthest point sampling which has been extended to differentiable form and used in various neural network architectures [31, 33, 51]. SampleNet [18] proposes a learned sampling method specific to the downstream tasks.

Our method aims to overcome the weaknesses of existing works while preserving their strengths. We borrow the convex QP problem formulation of [27] and implement a differentiable solution to learn to select points directly based on the pose estimation accuracy and eliminate the need of handcrafted heuristics.

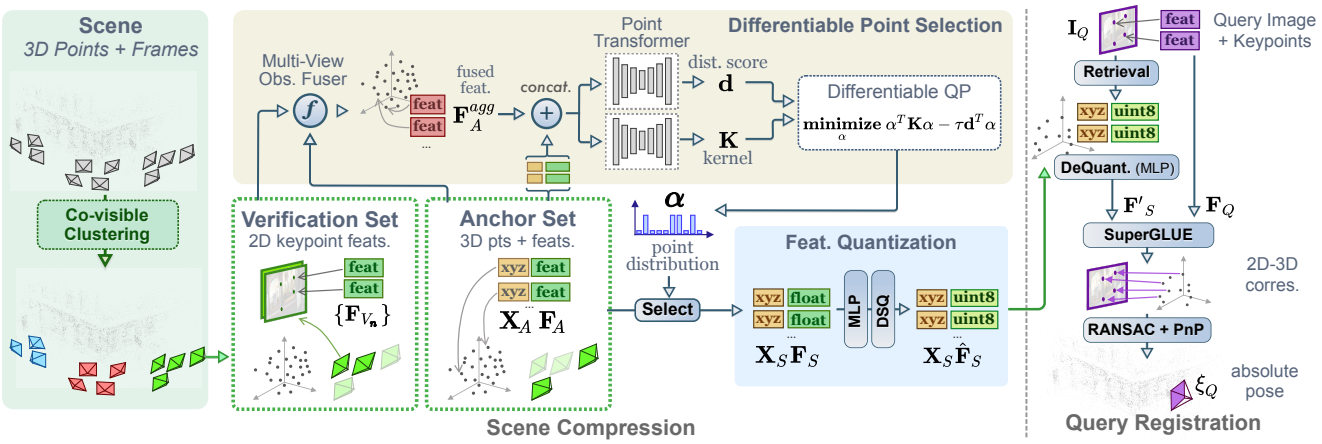

Figure 1. **Pipeline of our system**. A scene is compressed in 3 stages: co-visible frames clustering, 3D scene points selection and feature quantization. The compressed scene is then used for query frame registration.

3. Method

Our objective is to generate a compressed scene representation suitable for estimating absolute camera pose $\boldsymbol{\xi}_Q$ of a calibrated query image $\mathbf{I}_Q$ w.r.t the scene. The input is a scene database consisting of image frames $\{\mathbf{I}_R\}$ (known as reference frames) and 3D points $\{\mathbf{X}_R\}$, and their observed 2D keypoints $\mathbf{P}_R = \{(\mathbf{x}_R, \mathbf{F}_R)\}$, where $\mathbf{x}_R \in \mathbb{R}^2$ is position of a keypoint in the image plane and $\mathbf{F}_R \in \mathbb{R}^{256}$ is its corresponding SuperPoint [9] descriptor.

Figure 1 shows our scene compression pipeline. It can be divided into 3 stages: 1) Coarse-Level: **Co-visible Frames Clustering** where co-visible frames in the database are grouped using hierarchical clustering [28] and only a sparse set of frames are kept for localization (Section 3.1); 2) Middle-Level: **Differentiable Point Selection**, a point selection network for compressing point clouds from each cluster by taking into account the 2D-3D correspondences and high-level features of the 3D points (Section 3.2); 3) Fine-Level: **Feature Quantization** where the selected points' descriptors are encoded and compressed for storage such that they can be decompressed during runtime for feature matching (Section 3.3).

3.1. Co-visible Frames Clustering

Since compression of large-scale point clouds with millions of points is not feasible due to memory constraints of our point selection network, we divide the scene into multiple clusters and compress each cluster individually. Additionally, we can reduce the number of database frames required for coarse, image retrieval based localization by selecting fewer images that cover most of the observed points from each cluster. We use the hierarchical clustering [28] approach where the distance between the two frames i and j is a *co-visibility* distance $\psi(i,j)$ formulated as the log of inverse of Intersection over Union (IoU) of their observed 3D points $\{\mathbf{X}_i\}$ and $\{\mathbf{X}_j\}$:

$$\psi(i,j) = \log\left(\frac{|\{\mathbf{X}_i\} \cup \{\mathbf{X}_j\}|}{|\{\mathbf{X}_i\} \cap \{\mathbf{X}_j\}|}\right). \tag{1}$$

From each cluster we keep up to 20 frames that observe the largest number of 3D points as co-visible frames.

3.2. Learning to Select Points

We will further compress each cluster by reducing the number of 3D points while minimizing the loss of localization quality. It is well-known that accurate pose estimation not only depends on correct 2D-3D correspondences which is related to the points' distinctiveness, but also on good feature distribution. However, there is no golden standard to explicitly evaluate the distinctiveness and distribution of the points for the final pose estimation [13].

To this end, we design a differentiable point selection module that can be trained together with the final pose estimation, through which we can learn to evaluate the distinctiveness and distribution from the training data.

Specifically, given m 3D points $\mathbf{X}_A = \{\mathbf{X}_i \in \mathbb{R}^3 | \, i = 1, ..., m\}$, we estimate a distribution $\boldsymbol{\alpha} \in \mathbb{R}^m$ over the points where higher $\boldsymbol{\alpha}_i$ means higher likelihood of selecting the point $\mathbf{X}_i$. Similar to [27], we compute $\boldsymbol{\alpha}$ by optimizing the following QP problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & \boldsymbol{\alpha}^\mathsf{T}\mathbf{K}\boldsymbol{\alpha} - \tau\mathbf{d}^\mathsf{T}\boldsymbol{\alpha} \\ \text{subject to} \quad & \sum_i^m \alpha_i = 1, \\ & 0 \leq \alpha_i \leq \frac{1}{\nu m}; \quad i = 1, ..., m \end{aligned} \tag{2}$$

where $\mathbf{d} \in \mathbb{R}^m$ is the distinctiveness score vector of the 3D points, $\mathbf{K} \in \mathbb{R}^{m \times m}$ is the symmetric pairwise proximity matrix between the points, $\nu \in (0, 1]$ is the compression factor with $\nu = 1$ meaning no compression and lowering

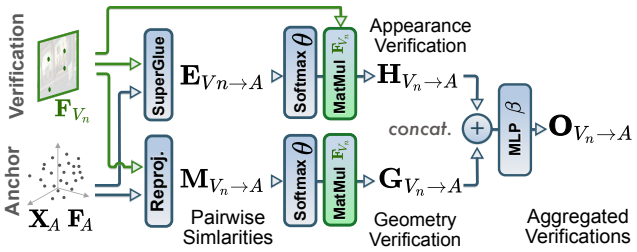

Figure 2. Anchor-Verification fusion between verification frame V_n and anchor set $\mathbf{A}$.

ν increases the compression (*i.e.* selects fewer points). τ is the scalar weight balancing the two costs. Unlike [27] which adopts heuristic rules, we propose to calculate both $\mathbf{d}$ and $\mathbf{K}$ from learned 3D point features. We therefore solve the QP problem in a differentiable way [1], enabling us to learn $\mathbf{d}$ and $\mathbf{K}$. The points corresponding to top-νm $\boldsymbol{\alpha}$ are selected and the rest discarded. We explain how we obtain $\mathbf{d}$ and $\mathbf{K}$ from the learned point features in the following sections.

3.2.1 Multi-view Observation Fuser

In order to better estimate the visual distinctiveness of 3D points, we first extract features of the points by using their 2D observations across multiple views. Instead of naively aggregating the SuperPoint features of the corresponding 2D keypoints, we consider the similarity between a 3D point and all possible 2D observations. The similarity is defined in terms of the correlation matrix between the 2D-3D pairs produced by a pre-trained SuperGlue matcher [37] along with the reprojection distance of the 3D points from the 2D keypoint positions. These respectively encode appearance and geometry similarity between 3D points and 2D observations.

Anchor-Verification Split: Exhaustively iterating over all 3D points with their paired 2D keypoints in multiple frames is expensive, we therefore split a co-visible cluster into two sets: *anchor* frames, which cover most of the 3D points, and the rest are *verification* frames. The 2D-3D pairs are then limited to 3D points from the *anchor* frames and 2D keypoints from the *verification* frames. Only the 3D points contained within the anchor frames are considered for subsequent compression.

Specifically, let $\mathbf{A} = \{A_k\}$ be the anchor frames with m_A 3D points $\{\mathbf{X}_i \in \mathbb{R}^3 | \, i = 1, ..., m_A\}$ and their features $\mathbf{F}_A \in \mathbb{R}^{m_A \times C}$. The 3D point features $\mathbf{F}_A$ are generated by aggregating the corresponding 2D keypoints' features using mean pooling. Let $\{V_n\}$ be the set of verification frames and $\mathbf{F}_{V_n} \in \mathbb{R}^{l_n \times C}$ be the features of l_n 2D keypoints in the verification frame V_n. Features of each keypoint is obtained by using SuperGlue's *Keypoint Encoder* which takes as input the SuperPoint descriptor/confidence of the keypoint along with the normalized keypoint position and outputs a $C = 256$ dimensional feature.

Anchor-Verification Fusion: We then fuse anchor and verification sets based on their appearance and geometry similarity. Figure 2 depicts the fusion of verification frame V_n with the anchor points. The appearance similarity matrix between keypoints of verification frame V_n and the anchor points is represented by $\mathbf{E}_{V_n \to A} \in \mathbb{R}^{m_A \times l_n}$. Here, we use the `score matrix` from SuperGlue before the optimal transport layer (Sinkhorn Algorithm [30]) in order to obtain continuous similarity values instead of hard assignments. Let $\mathbf{M}_{V_n \to A} \in \mathbb{R}^{m_A \times l_n}$ be the pairwise geometry similarity defined as the inverse of the reprojection distance between the 2D keypoints in V_n and 2D projections of the anchor points $\{\mathbf{X}_i\}$ on the verification frame.

The subsequent verification features $\mathbf{G}_{V_n \to A} \in \mathbb{R}^{m_A \times C}$ and $\mathbf{H}_{V_n \to A} \in \mathbb{R}^{m_A \times C}$ are the weighted sum of the verification keypoint features $\mathbf{F}_{V_n}$ based on the appearance and geometry similarity to the 3D points:

$$\begin{aligned} \mathbf{H}_{V_n \to A} &= \theta(\mathbf{E}_{V_n \to A})\mathbf{F}_{V_n} \\ \mathbf{G}_{V_n \to A} &= \theta(\mathbf{M}_{V_n \to A})\mathbf{F}_{V_n}, \end{aligned} \tag{3}$$

where θ denotes row-wise normalization using `softmax`. The concatenated $\mathbf{G}_{V_n \to A}$ and $\mathbf{H}_{V_n \to A}$ is then fed into MLP(β) to obtain aggregated verification features $\mathbf{O}_{V_n \to A}$ which account for appearance and geometric similarities.

Finally, the fused feature $\mathbf{F}_A^{agg} \in \mathbb{R}^{m_A \times C}$ combines the aggregated verification features $\{\mathbf{O}_{V_n \to A}\}$ from all the verification frames $\{V_n\}$ with the anchor features $\mathbf{F}_A$. This is done using scaled dot-product attention [48]:

$$\mathbf{F}_A^{agg} = \text{Attention}(\mathbf{O}_{V_1 \to A}, ..., \mathbf{O}_{V_n \to A}, \mathbf{F}_A), \tag{4}$$

where $\mathbf{F}_A$ acts as *Key* and $\{\mathbf{O}_{V_n \to A}\}$ as *Query* for the attention mechanism. Please refer to the supplementary document for more detail.

3.2.2 Learning Distinctiveness and Proximity

With the fused features $\mathbf{F}_A^{agg}$ of the anchor set $\mathbf{A}$, we estimate the distinctiveness $\mathbf{d}$ of the 3D points and their pairwise proximity $\mathbf{K}$, which are utilized to solve the point distribution α using Equation (2). As shown in Figure 1, the $\mathbf{d}$ and $\mathbf{K}$ are obtained using two identical Point Transformer [52] networks. We choose Point Transformer instead of other point-based architectures [32, 33] because of their superior self-attention based information aggregation mechanism and impressive performance in various 3D tasks. Please refer to our supplementary materials for more details on the network architecture. The fused verification features $\mathbf{F}_A^{agg}$ along with their anchor feature $\mathbf{F}_A$ and 3D coordinates $\mathbf{X}_A$ are fed into the two point transformers to generate the distinctiveness scores $\mathbf{d}$ and point features $\mathbf{f} \in \mathbb{R}^{64}$ of each 3D point that are used to calculate the pairwise proximity matrix $\mathbf{K}$.

The pairwise proximity $\mathbf{K}$ is conventionally set to $\mathbf{K}^{\text{spatial}}$ [27] where each entry is a (Gaussian) radial basis

function of spatial distance between pairs of points (i, j) *i.e.* $\mathbf{K}_{ij}^{\text{spatial}} = \text{RBF}(\|\mathbf{X}_i - \mathbf{X}_j\|, \sigma_{\text{RBF}})$ with

$$\text{RBF}(x, \sigma_{\text{RBF}}) = \exp\Big(-\frac{x^2}{2{\sigma_{\text{RBF}}}^2}\Big). \tag{5}$$

Instead of only using the spatial distance for $\mathbf{K}$, we add a learned matrix $\Delta\mathbf{K}$ to obtain $\mathbf{K} = \frac{1}{2}(\mathbf{K}^{\text{spatial}} + \Delta\mathbf{K})$. Let $\hat{\mathbf{f}}$ denote the normalized features of $\mathbf{f}$. We generate $\Delta\mathbf{K}$ using RBF of pairwise feature distance: $\Delta\mathbf{K}_{ij} = \text{RBF}(\|\hat{\mathbf{f}}_i - \hat{\mathbf{f}}_j\|)$ for each pair of points (i, j). This allows us to not only get a good spatial coverage, but also to potentially reduce visual redundancies and avoid pose estimation degeneracies with the selected points.

After generating $\mathbf{d}$ and $\mathbf{K}$, we optimize $\boldsymbol{\alpha}$ in Equation (2) with a differentiable QP solver [1] in order to backpropagate the error gradients to $\mathbf{K}, \mathbf{d}$ and τ during training.

3.3. Fine-Level Feature Quantization

For each selected point $\mathbf{X}_S$ we need its feature $\mathbf{F}_S$, a 256 dimensional vector of 32-bit floating-point numbers, in order to perform query frame registration. We compress these descriptors using an auto-encoder network.

Our encoder contains a 2-layered MLP that reduces the dimensions of the input descriptor $\mathbf{F}_S$ by half, and a Differentiable Soft Quantization (DSQ) [12] layer that quantizes the MLP output from 32-bit floats to 8-bit unsigned integers. Let $\hat{\mathbf{F}}_S$ denote the resulting quantized features. A decoder consisting of MLP identical to the encoder then tries to recover the original descriptor in the form of $\mathbf{F'}_S$ of same dimensions and data type as $\mathbf{F}_S$. The network is trained using L2-loss: $L_Q = \|\mathbf{F'}_S - \mathbf{F}_S\|_2$.

After the scene compression, we only store the lower dimensional quantized descriptors $\hat{\mathbf{F}}_S$ generated by the encoder network, resulting in an eight-fold reduction in memory. During runtime localization the decoder network outputs $\mathbf{F'}_S$ which can be used for feature matching.

3.4. Training Losses

We use two losses L_d and L_ξ to train our network to produce appropriate $\mathbf{d}$, $\mathbf{K}$ and subsequently $\boldsymbol{\alpha}$. L_d encourages the network to assigned high $\mathbf{d}$ values to points that are easier to track across multiple frames and is implemented as follows:

$$L_d = \sum_{j \in \mathbf{A}} \frac{1}{2{\sigma_j}^2} \sum_{(j,i), i \in Q} \|\pi_Q(\boldsymbol{\xi}_{gt}, \mathbf{X}_j) - \mathbf{x}_i\| + \sum_{j \in \mathbf{A}} \frac{1}{2}\log({\sigma_j}^2), \tag{6}$$

where σ_j represents the uncertainty of the 3D point $\mathbf{X}_j$ in the anchor set $\mathbf{A}$, Q is a query frame with ground truth pose $\boldsymbol{\xi}_{gt}$, $\mathbf{x}_i \in \mathbb{R}^2$ is the 2D image coordinates of the keypoint

Figure 3. **Visualization of the learned distinctiveness scores d**. Our system learns to assign high scores to strong features that can be accurately matched with the query images

in query frame corresponding to the point $\mathbf{X}_j$. π_Q is the projection function mapping a 3D points in world coordinate frame to 2D image coordinates given the camera pose and intrinsics. The (j, i) correspondences between reference and query points are predicted by a pre-trained SuperGlue matcher.

The above loss function takes inspiration from Bayesian deep learning [15]. The first term is the weighted errors of the predicted 2D-3D matches (j, i) while the second term serves as regularizer to prevent all the points from degenerating to high uncertainty. Finally, the distinctiveness score of the point $\mathbf{X}_j$ is set to $\mathbf{d}_j = \frac{1}{2{\sigma_j}^2}$ meaning lower uncertainty in matching the 3D point with its 2D keypoint leads to higher distinctiveness score.

Figure 3 shows the learned distinctiveness scores $\mathbf{d}$ for 2 example images from the *Cambridge Landmarks* dataset [16]. In the figure we can see that the distinct features like corners and edges receive high scores. Since both 2D and 3D information are used for computation of $\mathbf{d}$, the scores can take into account the 3D structure along with the appearance information.

Another loss term L_ξ tries to maximize the accuracy of the camera pose estimated using the points sampled from the distribution $\boldsymbol{\alpha}$:

$$L_\xi = \sum_{h=1}^{H} \Bigg(\Big(\prod_{i=1}^{4} \boldsymbol{\alpha}_i^h \Big) \|\pi_Q(\boldsymbol{\xi}_h, \mathbf{X}^R) - \pi_Q(\boldsymbol{\xi}_{gt}, \mathbf{X}^R)\| + \sum_{i=1}^{4} \boldsymbol{\alpha}_i^h \|\mathbf{x}_i^Q - \pi_Q(\boldsymbol{\xi}_{gt}, \mathbf{X}_i^R)\| \Bigg), \tag{7}$$

where we randomly sample $h = 1, ..., H$ hypotheses, with each hypothesis h consisting of 4 points with distinctiveness scores $\{\boldsymbol{\alpha}_i^h \mid i = 1, ..., 4\}$ randomly selected from the set of 3D points $\mathbf{X}^R \in \mathbb{R}^{k \times 3}$. $\mathbf{x}_i^Q \in \mathbb{R}^{k \times 2}$ represents the 2D keypoint positions in query frame Q corresponding to $\mathbf{X}^R$ obtained from the SuperGlue matcher. $\boldsymbol{\xi}_h$ is the estimated pose of the hypothesis h using 4 points solution of PnP [19] and $\boldsymbol{\xi}_{gt}$ is the ground truth pose of the query frame.

The first term inside the summation in Equation (7) updates the $\boldsymbol{\alpha}_i$ of all 4 points in a hypothesis based on the pose error. Since a higher pose error does not necessarily mean all 4 points led to incorrect matches, we add the second term

which updates the exact $\boldsymbol{\alpha}_i$ based on the reprojection error of predicted 2D-3D correspondence between $\mathbf{x}_i^Q$ and $\mathbf{X}_i^R$. By doing this we can encourage the point selection to make sure that any combination of 4 selected points will lead to good localization while also learning the distinctiveness of individual points.

3.5. Runtime Localization

After our network is trained, we compress the point clouds of each cluster using a custom implementation of the Sequential Minimal Optimization (SMO) based quadratic programming solver proposed in [27]. This allows us to compress large point clouds efficiently using the predicted $\mathbf{d}$, $\mathbf{K}$ and τ. We also compress the AP-GeM [34] global descriptor of the frame with the most observed 3D points from each co-visible cluster. The same technique of Section 3.3 is used to quantize the 2048 dimensional, floating-point AP-GeM descriptors to 1024 8-bit unsigned integer descriptors. Only the compressed point clouds and quantized features are kept for localization.

For runtime query frame registration, we use a strategy similar to HLoc [36]. First, image retrieval [34] is used to find nearest co-visible clusters similar to the query frame $\mathbf{I}_Q$. We then register the query frame against each of the compressed point cloud clusters corresponding to the image retrieval results using PnP+RANSAC [19]. The final camera pose ξ_Q is obtained by running PnP+RANSAC again on all the 2D-3D correspondences from the clusters that led to successful registration. More details about the global descriptor quantization and localization process can be found in the supplementary document.

4. Experiments

Datasets: We use the training set of the *RobotCar Seasons* [26] dataset to train our network and evaluate the trained network on the *Cambridge Landmarks* [16] and *Aachen Day-Night* [42] datasets. All of the datasets are city-scale capturing small/medium (*Cambridge Landmarks*) to large (*Aachen Day-Night*, *RobotCar Seasons*) scale scenes. Although SIFT [25] based 3D reconstructions are available for the datasets, since our network is based on SuperPoint features, we perform re-triangulation of the scene using the ground truth camera poses in a similar fashion to [36].

Training Dataset: The *RobotCar Seasons* dataset consists of 5.7K training images under various weather conditions such as `dawn`, `dusk`, `night` etc. It also contains an additional set of 26K images called `overcast-reference` which are used to generate the 3D scene point cloud and the co-visible clusters (Subsection 3.1). There are in total 4.1K co-visible clusters, 20% of which are used as validation set.

In one training iteration we take a set of co-visible frames from which 30% frames are selected as *anchor* frames and another 30% as *verification* frames. Frames within 10 meters and 45° of the cluster center are used as query frames from the remaining 40%.

Testing Dataset: Following [5, 27] we select 4 scenes from the *Cambridge Landmarks* dataset, viz., `Shop Facade`, `King's College`, `Old Hospital` and `St.Mary Church` for evaluation, which cover small to medium-scale scenarios. We also use the *Aachen Day-Night* dataset to evaluate our method on large-scale scene and more challenging nighttime conditions. Their training sets are used as reference frames and the test sets are used for evaluation. No fine-tuning is performed on our network using these datasets.

Similar to [36], we select top-15 image retrieval matches of a query frame for *Cambridge Landmarks* dataset and top-50 for the larger *Aachen Day-Night* to find the potential clusters where the query image could belong to and then perform finer pose estimation (Subsection 3.5).

Implementation Details: Our network is implemented in PyTorch [29]. The training is performed on NVIDIA A5000 GPU with 24 GB memory. We use Adam optimizer [17] with an initial learning rate 5×10^{-4} and a batch size of 1.

Due to memory constraints we do not optimize the weights of the SuperGlue network and limit the size of the point cloud input to the QP solver to 1000 by random selection during training. Although the QP input is truncated, we use all the points for calculating the losses in Equations 6 and 7. The RBF bandwidth (σ_{RBF}) for both spatial and feature distance kernels are fixed to 1.0 while the learned weight τ is initialized with 0.5. The compression ratio ν is fixed to 1.5% during training while during testing it is 1.5% as long as the number of compressed points in the cluster is at least 100. Otherwise, an adaptive compression ratio is applied such that the number of points is 100 in order to be able to accurately localize query images belonging to those clusters. Please refer to our supplementary document for further details.

4.1. Quantitative Comparisons

Cambridge Landmarks dataset: We perform quantitative comparison against various localization methods in Table 1. QP+RootSIFT [27], Hybrid [5], KP [23], KCP [6] are compression-based methods. We also include uncompressed methods for references in our comparison. DSAC++ [3] and PoseNet(Geo.) [14] are scene-specific methods that implicitly encode a scene in their deep network weights. The scene-agnostic SANet [50] on the other hand stores the RGBD images of the entire training set. Active Search (AS) [41] and HLoc [41] are sparse feature-based methods where a scene is stored as uncompressed 3D point cloud along with their 2D keypoint descriptors. Our method and HLoc perform hierarchical, coarse-to-fine localization while the rest do not.

Along with the median translation error in meters and median rotation error in degrees, we also report the size

Method	Shop Facade Size (MB)	Shop Facade Trans. Err (m)	Shop Facade Rot. Err (°)	Old Hospital Size (MB)	Old Hospital Trans. Err (m)	Old Hospital Rot. Err (°)	King's College Size (MB)	King's College Trans. Err (m)	King's College Rot. Err (°)	St. Mary's Church Size (MB)	St. Mary's Church Trans. Err (m)	St. Mary's Church Rot. Err (°)
Compression-based methods												
Ours	**0.13**	**0.11**	**0.38**	**0.53**	**0.37**	**0.53**	**0.3**	**0.27**	**0.38**	**0.95**	**0.15**	**0.37**
QP+RootSIFT [27]	0.41	0.72	1.4	1.1	0.9	2.17	2.2	1.53	1.09	3.3	0.56	0.89
Hybrid [5]	0.16	0.19	0.54	0.62	0.75	1.01	1.01	0.81	0.59	1.34	0.5	0.49
KC [23]	0.85	0.51	0.87	6	1.35	1.06	3.1	1.48	1.23	18	0.46	0.69
KCP [6]	1.30	0.44	0.8	8.2	1.19	1	5.9	0.99	0.86	24	0.4	0.61
Uncompressed methods												
PoseNet(Geo.) [14]	50	0.88	3.78	50	3.2	3.29	50	0.88	1.04	50	1.57	3.32
AS [41]	38.7	0.12	0.41	140	0.52	1.12	275	0.57	0.7	359	0.22	0.62
SANet [50]	27	0.1	0.47	105	0.32	0.53	143	0.32	0.54	174	0.16	0.57
DSAC++ [3]	207	0.06	0.3	207	0.2	0.3	207	0.18	0.3	207	0.13	0.4
HLoc [36]	316	0.04	0.2	1335	0.15	0.3	1877	0.12	0.2	2009	0.07	0.21

Table 1. **Cambridge Landmarks** quantitative comparison. We report the size of the scene representation in MB, the median translation error in meters(m) and median rotation error in degrees(°) for each method.

Method	Size (MB)	Aachen Day 0.25m, 2° / 0.5m, 5° / 5m, 10°	Aachen Night 0.25m, 2° / 0.5m, 5° / 5m, 10°
Compression-based methods			
Ours	31	75.5 / **89.7** / **96.2**	**50.0** / **67.3** / **78.6**
Cascaded [7]	140	**76.7** / 88.6 / 95.8	33.7 / 48.0 / 62.2
QP+R.Sift* [27]	31	62.6 / 76.3 / 84.7	16.3 / 18.4 / 24.5
Uncompressed methods			
PixLoc(E2E) [38]	2189	61.7 / 67.6 / 74.8	46.9 / 53.1 / 64.3
ESAC(50) [4]	1315	42.6 / 59.6 / 75.5	6.1 / 10.2 / 18.4
AS [41]	750	57.3 / 83.7 / 96.6	28.6 / 37.8 / 51.0
HLoc [36]	7828	89.6 / 95.4 / 98.8	86.7 / 93.9 / 100.0

Table 2. **Aachen Day-Night** quantitative comparison. We report the size of the scene representation in MB and percentage(%) of query frames successfully localized within the given translation and rotation errors in meters and degrees respectively.

of scene representation in MB stored by each method. Our method outperforms all the existing compression-based methods while also keeping a lower memory footprint.

As for the uncompressed methods, our method outperforms Active Search (AS) [41] and is also on par with SANet [50]. Although DSAC++ [3] and HLoc [36] yield better localization performance, they come with considerably high cost in memory compared with our method.

Aachen Day-Night dataset: We compare our method against existing methods on the *Aachen Day-Night* dataset in Table 2. The method "Cascaded" [7] compresses the descriptor directly without reducing the number of 3D points. Since [27] has no source code available, we use our implementation of their method but keep our image retrieval module (denoted by QP+RootSift*). We include the uncompressed method ESAC(50) [4] which extends DSAC++ [3] along with the end-to-end method PixLoc(E2E) [38].

Our method performs better than or on par with all the other methods, for all the thresholds with HLoc being a notable exception. However, the higher performance of HLoc comes at a steep memory cost. The compression-based method "Cascaded" [7] performs similar to ours for the daytime images but our method significantly outperforms theirs for the nighttime images while only using 20% of their memory.

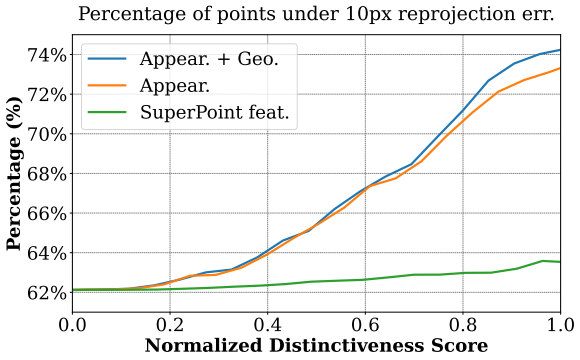

Figure 4. **Performance of Differentiable Point Selection with different inputs.** 3D points are selected based on a threshold on the distinctiveness scores **d** (shown in `x`-axis) and the percentage of selected points that have less than 10 pixel re-projection error against their 2D correspondences are reported in `y`-axis.

4.2. Ablation Study

Effectiveness of using verification set: In this experiment we use the 4 scenes of *Cambridge Landmarks* dataset and select three different types of input to feed to our Differentiable Point Selection (DPS) module and examine how it affects the distinctiveness score **d**: 1) the anchor and verification frame features are aggregated using appearance and geometric similarities (Equation (3)), denoted by `Appear.+Geo.`; 2) using only the appearance similarity (referred to as `Appear.`) for aggregating the anchor and verification features; 3) SuperPoint features of the anchor frames are directly used without fusing with verification frames, denoted by `SuperPoint feat.` Figure 4 shows the predicted distinctiveness score (normalized) of the 3D scene points against the percentage of correct 2D-3D matches in the query frames. Here we select the subset of scene points with distinctiveness score higher than a given threshold (`x`-axis). After reprojecting those 3D points to the query image planes using ground-truth camera poses, we then evaluate the percentage of the points where the re-

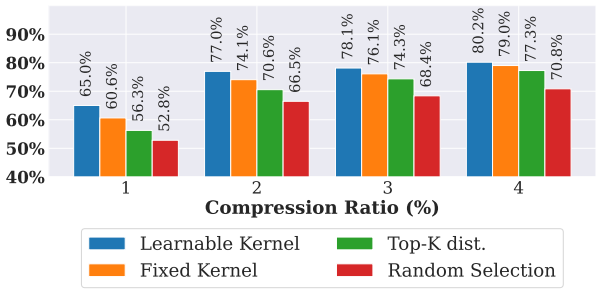

Figure 5. **Performance of our system when using learned kernel vs fixed spatial kernel.** We report the localization recall under the translation and rotation error thresholds (0.5m, 5°). The random selection and naive selection of the top distinct points (`Top-k dist.`) baselines are also included in the comparison.

projection distance against their matched query keypoints is under 10 pixels (`y`-axis).

Learned vs Fixed Kernel: We evaluate the localization performance when using fixed spatial distance kernel $\mathbf{K}^{\text{spatial}}$ instead of the learned kernel $\mathbf{K}$ when solving Equation (2) to find the selected point. For comparison, we also show the accuracy when we randomly select the points and when we directly choose the points with highest distinctiveness score $\mathbf{d}$ (`Top-k dist.`). We select the `Kings College` scene for our experiment and the results are shown in Figure 5. The gap between random selection and other methods is large indicating that our learned distinctiveness score $\mathbf{d}$ correspond to the ability to correctly match the 3D points with query image keypoints. Using $\mathbf{K}^{\text{spatial}}$ along with $\mathbf{d}$ performs better than `Top-k dist.` and the learned $\mathbf{K}$ performs the best, especially at higher compression (corresponding to lower compression ratio). The difference in accuracy between learned kernel, fixed kernel and top distinctiveness score gets lower when compression ratio is increased. This is because a larger number of points will be used for localization which increases the number of correct 2D-3D matches and the PnP with RANSAC [19] scheme becomes more robust against outliers.

Figure 6 shows the points selection on two frame in a co-visible cluster using the fixed and learned kernels respectively. Both use a compression ratio of 1.5%. The fixed kernel only uses the 3D position information while the learned kernel considers 2D position and feature information as well. The learned kernel therefore leads to better feature distribution, in the image coordinates and not just the 3D coordinates. This can explain why the learned kernel performs better than the fixed spatial kernel in Figure 5.

Compression Time: In Table 3 we report the time required for scene compression by our system (compression ratio 1.5%), QP+R.Sift [27], KC [23] and KCP [6] on the `St.Marys Church` scene from the *Cambridge Landmarks* dataset. For QP+R.Sift, KC and KCP, we present the compression time reported by [27]. Our compression time is similar to KC and lower than KCP, but it is significantly

Figure 6. **Visualization of the points selected using fixed and learned kernels.** The selected 3D points are reprojected to the image plane and shown as Green dots.

Our	QP+R.Sift [27]	KC [23]	KCP [6]
484	18	328	798

Table 3. Compression time (seconds) of different methods.

higher than QP+R.Sift since our computation of $\mathbf{d}$ and $\mathbf{K}$ is more involved than QP+R.Sift.

5. Limitations

There are several limitations of our system: 1) Our co-visible clusters consist of maximum 20 frames, 30% of which are *anchor* frames. Although we select the frames with the most observed points, some points are still ignored. A smarter strategy that takes both efficiency and wider coverage into account could be used for improvement; 2) Due to the limited capacity of the existing point transformer, we can only process one cluster at a time. This may cause unnecessary costs for processing overlapping clusters; 3) Multiple retrieval candidates are required to get an accurate registration of a query image. With larger scenes such as *Aachen Day-Night* [42], a larger number of candidates are needed to tackle ambiguous retrievals. This can be too expensive for real-time applications, especially for mobile platforms where the computational resources are limited.

6. Conclusion

We present a novel method for compressing a scene while retaining visual localization accuracy. Our scene compression is done in three stages: co-visible frames clustering, learned point selection and learned feature quantization. Unlike existing methods [5, 6, 23, 27], we do not use handcrafted heuristics but learn to select a subset of scene points that can maintain pose estimation accuracy. Experiments on various datasets corroborate our method's ability to perform accurate localization with low memory footprint.

Acknowledgement. This work is supported by the Canada NSERC Discovery Project (611664), the National Natural Science Foundation of China (No. 61872067, No. 62102356), and the Zhejiang Lab (2021PE0AC01).

References

[1] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. In *NeurIPS*, 2019. 2, 4, 5

[2] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Ijcv ransac for camera localization. In *CVPR*, 2017. 2

[3] Eric Brachmann and Carsten Rother. Learning less is more-6d camera localization via 3d surface regression. In *CVPR*, 2018. 2, 6, 7

[4] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *ICCV*, 2019. 2, 7

[5] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid scene compression for visual localization. In *CVPR*, 2019. 2, 6, 7, 8

[6] Song Cao and Noah Snavely. Minimal scene descriptions from structure from motion models. In *CVPR*, 2014. 1, 2, 6, 7, 8

[7] Wentao Cheng, Weisi Lin, Kan Chen, and Xinfeng Zhang. Cascaded parallel filtering for memory-efficient image-based localization. In *ICCV*, 2019. 1, 2, 7

[8] Wentao Cheng, Weisi Lin, Xinfeng Zhang, Michael Goesele, and Ming-Ting Sun. A data-driven point cloud simplification framework for city-scale image-based localization. *TIP*, 26(1):262–275, 2016. 2

[9] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, 2018. 2, 3

[10] Marcin Dymczyk, Simon Lynen, Michael Bosse, and Roland Siegwart. Keep it brief: Scalable creation of compressed localization maps. In *IROS*, 2015. 2

[11] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *T-PAMI*, 25(8):930–943, 2003. 1, 2

[12] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *CVPR*, 2019. 2, 5

[13] You-Yi Jau, Rui Zhu, Hao Su, and Manmohan Chandraker. Deep keypoint-based camera pose estimation with geometric constraints. In *IROS*, 2020. 1, 3

[14] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *CVPR*, 2017. 1, 2, 6, 7

[15] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, 2017. 5

[16] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015. 1, 2, 5, 6

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[18] Itai Lang, Asaf Manor, and Shai Avidan. Samplenet: Differentiable point cloud sampling. In *CVPR*, 2020. 2

[19] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *IJCV*, 81(2):155–166, 2009. 1, 2, 5, 6, 8

[20] Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical scene coordinate classification and regression for visual localization. In *CVPR*, 2020. 2

[21] Xiaotian Li, Juha Ylioinas, and Juho Kannala. Full-frame scene coordinate regression for image-based localization. *arXiv preprint arXiv:1802.03237*, 2018. 2

[22] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, 2012. 2

[23] Yunpeng Li, Noah Snavely, and Daniel P Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*, 2010. 1, 2, 6, 7, 8

[24] Hyon Lim, Sudipta N Sinha, Michael F Cohen, and Matthew Uyttendaele. Real-time image-based 6-dof localization in large-scale environments. In *CVPR*, 2012. 2

[25] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 6

[26] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017. 6

[27] Marcela Mera-Trujillo, Benjamin Smith, and Victor Fragoso. Efficient scene compression for visual-based localization. In *3DV*, 2020. 1, 2, 3, 4, 6, 7, 8

[28] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011. 3

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NeurIPSW*, 2017. 6

[30] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019. 4

[31] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019. 2

[32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 4

[33] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 2, 4

[34] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *ICCV*, 2019. 6

[35] Jerome Revaud, Philippe Weinzaepfel, César Roberto de Souza, and Martin Humenberger. R2D2: repeatable and reliable detector and descriptor. In *NeurIPS*, 2019. 2

[36] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 1, 2, 6, 7

[37] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature

matching with graph neural networks. In *CVPR*, 2020. 2, 4

[38] Paul-Edouard Sarlin, Ajaykumar Unagar, Måns Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, and Torsten Sattler. Back to the Feature: Learning robust camera localization from pixels to pose. In *CVPR*, 2021. 1, 2, 7

[39] Torsten Sattler, Michal Havlena, Filip Radenovic, Konrad Schindler, and Marc Pollefeys. Hyperpoints and fine vocabularies for large-scale location recognition. In *ICCV*, 2015. 1, 2

[40] Torsten Sattler, Michal Havlena, Konrad Schindler, and Marc Pollefeys. Large-scale location recognition and the geometric burstiness problem. In *CVPR*, 2016. 2

[41] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *T-PAMI*, 39(9):1744–1756, 2016. 2, 6, 7

[42] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image retrieval for image-based localization revisited. In *BMVC*, 2012. 6, 8

[43] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *CVPR*, 2019. 2

[44] Hyun Soo Park, Yu Wang, Eriko Nurvitadhi, James C Hoe, Yaser Sheikh, and Mei Chen. 3d point cloud reduction using mixed-integer quadratic programming. In *CVPRW*, 2013. 2

[45] Linus Svarm, Olof Enqvist, Magnus Oskarsson, and Fredrik Kahl. Accurate localization and pose estimation for large 3d models. In *CVPR*, 2014. 2

[46] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018. 2

[47] Shitao Tang, Chengzhou Tang, Rui Huang, Siyu Zhu, and Ping Tan. Learning camera localization via dense scene matching. In *CVPR*, 2021. 2

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

[49] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for structured feature correlation. In *ICCV*, 2017. 1, 2

[50] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. Sanet: Scene agnostic network for camera localization. In *ICCV*, 2019. 2, 6, 7

[51] Kangxue Yin, Zhiqin Chen, Hui Huang, Daniel Cohen-Or, and Hao Zhang. Logan: Unpaired shape transform in latent overcomplete space. *TOG*, 38(6):1–13, 2019. 2

[52] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 4