2020 International Conference on 3D Vision (3DV) | 978-1-7281-8128-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/3DV50981.2020.00111

# Efficient Scene Compression for Visual-based Localization

Marcela Mera-Trujillo
West Virginia University
mameratrujillo@mix.wvu.edu

Benjamin Smith
West Virginia University
bbsmith1@mix.wvu.edu

Victor Fragoso
Microsoft
victor.fragoso@microsoft.com

## Abstract

*Estimating the pose of a camera with respect to a 3D reconstruction or scene representation is a crucial step for many mixed reality and robotics applications. Given the vast amount of available data nowadays, many applications constrain storage and/or bandwidth to work efficiently. To satisfy these constraints, many applications compress a scene representation by reducing its number of 3D points. While state-of-the-art methods use $K$-cover-based algorithms to compress a scene, they are slow and hard to tune. To enhance speed and facilitate parameter tuning, this work introduces a novel approach that compresses a scene representation by means of a constrained quadratic program (QP). Because this QP resembles a one-class support vector machine, we derive a variant of the sequential minimal optimization to solve it. Our approach uses the points corresponding to the support vectors as the subset of points to represent a scene. We also present an efficient initialization method that allows our method to converge quickly. Our experiments on publicly available datasets show that our approach compresses a scene representation quickly while delivering accurate pose estimates.*

## 1. Introduction

Estimating the camera pose (*i.e.*, position and orientation) is a crucial step for applications in self-driving cars [16, 17, 28], robotics [12, 20, 26], and mixed reality [14, 31, 33, 43, 48]. This is because these applications use camera poses to understand how the camera is positioned and oriented with respect to an environment.

While many 3D computer vision systems successfully localize themselves in an environment, they struggle to scale well when the environment becomes very large [38]. Their struggle has various reasons. First, the memory and/or disk space requirements needed to store and represent the environment can be substantial. This is because the common scene representation can contain a collection of images, 3D points, and 2D image features with their respective feature descriptors (*e.g.*, SIFT [30]). Second, most of these systems use pose estimators that require longer time

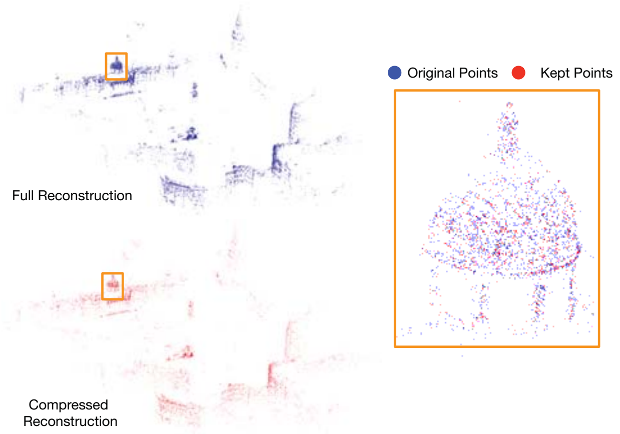

Figure 1. Unlike $K$-cover-based scene compression techniques, we formulate the scene representation compression problem with a quadratic program (QP). The goal of the QP is to keep 3D points that have good spatial coverage and visual distinctiveness. Inspired by one-class support vector machines (SVMs), we derive an efficient sequential-minimal-optimization (SMO) QP solver for scene compression. Similar to SVMs that use the support vectors to define its decision function, our method keeps the 3D points that are marked as the support vectors and discard the remaining ones.

to operate when the representation of the scene is large. Although there exist efforts that increase efficiency of pose estimation (*e.g.*, [5, 7, 44, 45]), they still struggle when the scene representation is large.

To improve the scalability of these computer vision systems, we aim to compress scene representations. In particular, we focus on compressing point clouds computed via structure-from-motion (SfM) pipelines. This is because SfM point clouds are the most common scene representation for visual-based localization.

The reader must recall that an SfM point cloud has a collection of 3D points describing the geometry of a scene. Every point in this representation (typically) has a set of 2D image features and their respective feature descriptors [29, 36]. The goal of this work is to carefully select a subset of 3D points from an SfM point cloud such that the selection provides enough information to accurately estimate a camera pose. Consequently, compressing an SfM point cloud reduces the storage footprint because it prunes

Authorized licensed use limited to: West Virginia University. Downloaded on December 26,2021 at 21:47:51 UTC from IEEE Xplore. Restrictions apply.

"unnecessary" points. This is useful especially for mobile agents (*e.g.*, mobile devices, robots, etc.) that have limited storage and need to self-localize in an environment.

Most state-of-the-art point cloud compression methods [6, 8, 29, 41] use the $K$-cover-based methodology. In simple terms, the $K$-cover-based methodology aims to find a minimum subset of 3D points such that each database image sees at least $K$ points in the subset. While this approach effectively compresses an SfM point cloud, finding the right parameters to reduce the size of an SfM point cloud by a certain factor is not a trivial task.

We present an optimization problem for compressing an SfM point cloud that is convex and resembles the one-class support vector machine (SVM) [40]; see Fig. 1 for an illustration. Thanks to this resemblance, we derive an efficient solver based on the sequential minimal optimization (SMO) [34], which is an efficient and scalable optimization technique to train SVMs. The proposed approach aims to select points that cover sufficiently the surface to represent but at the same time present a visual distinctiveness. From the SVM perspective, the support vectors [10, 39] correspond to the 3D-point selection that best represent a scene. Moreover, our proposed approach has parameters with intuitive meanings, which makes the parameter tuning simpler than that of $K$-cover-based methods. Our experiments on existing large-scale image-based localization datasets (*e.g.*, Dubrovnik [36] and Cambridge Landmarks dataset [24]) show that our approach compresses an SfM point cloud efficiently while yielding accurate pose estimates.

In sum, we present the following contributions: **1)** a QP convex formulation for compressing SfM point clouds with easy to tune parameters; **2)** an effective initialization method for the QP compression problem; and **3)** an efficient SMO-based constrained QP solver for compression.

## 2. Related Work

Reducing the number of points in an SfM point cloud has been addressed by means of $K$-cover-inspired algorithms, mixed-integer quadratic program (QP) optimization methods, and deep-learning-based approaches. This section covers work that falls under these three different approaches.

### 2.1. $K$-cover-based Approaches

Li *et al.* [29] presented a $K$-cover-inspired algorithm to select a minimum subset of 3D points such that each database image sees at least $K$ points in the subset. While this approach works well for reducing an SfM point cloud size, computing the subset of 3D points is a challenging combinatorial problem. Li *et al.* [29] computes this minimum subset by incrementally building it. Their method uses a gain function that allows the algorithm to select points that contribute to the construction of the subset.

While the $K$-cover-inspired algorithm [29] reduces an SfM point cloud size, it does not include information about the visual distinctiveness of each of the selected points. This aspect is important for image-based localization since visual features (*e.g.*, SIFT [30]) are crucial to establish 2D-to-3D correspondences which are the input for any pose estimator. To address this issue, Cao and Snavely [8] extended the $K$-cover algorithm [29] by considering the coverage and the visual distinctiveness of the points. The coverage aspect imposes the constraint that the points in the subset are highly visible, *i.e.*, that a new camera observing the scene has a high probability of seeing most of the points in the subset. Moreover, their extension includes a visual distinctiveness term that favors the selection of points that are easy to visually identify.

Camposeco *et al.* [6] presented a hybrid scene compression method that computes two sets of 3D points: the first set of points is small and contains raw descriptors while the second set is larger and contains quantized descriptors. The first set yields high-quality 2D-3D correspondences while the second set allows the hybrid compression method to verify hypotheses within a RANSAC loop. This hybrid method thus reduces the memory or storage footprint by decreasing the number of 3D points to keep and quantizing descriptors. At its core of this hybrid compression method is a variant of the $K$-cover algorithm and quantization methods all integrated with RANSAC. Although the $K$-cover-based methods effectively compress an SfM point cloud, it is hard to find the parameter $K$ such that the resultant compressed SfM point cloud reduces its size by a certain factor. In contrast to these methods, our approach has parameters that are easy to set given their intuitive meaning

### 2.2. Mixed-integer-QP-based Approaches

An alternative approach to the $K$-cover-based algorithms is a formulation using mixed-integer programming. Park *et al.* [41] introduced a constrained quadratic program (QP) formulation mimicking the $K$-cover problem. This problem aims to compute a binary vector. The $i$-th entry of this vector is set to 1 when the $i$-th point is kept, and it is set 0 otherwise. Unfortunately, solving the formulated constrained QP is not scalable and requires specialized mixed-integer solvers. This method struggles to scale due to the $n \times n$ matrix that encodes pairwise relationships among the points; $n$ is the number of points. Clearly, for large-scale datasets $n$ is large and the scalability of this method depends of the used solver. Dymczyk *et al.* [13] build on Park *et al.* [41] work and scale it by dividing the problem into sub-problems. Although our proposed formulation also uses a constrained QP formulation, we present an efficient solver that scales well. This is because our QP formulation shares the structure of a one-class SVM [40] and can be solved efficiently using a variant of the sequential minimal optimization [34] (SMO) method that we present in Sec. 3.

Authorized licensed use limited to: West Virginia University. Downloaded on December 26,2021 at 21:47:51 UTC from IEEE Xplore.  Restrictions apply.

## 2.3. Deep-learning-based Approaches

Several deep-learning-based approaches [18, 23, 24] aim to address image-based localization or pose estimation. These methods can be considered as compression approaches because the learned weights of the neural network encode the parameters of a scene. Kendall *et al*. presented PoseNet [24], a convolutional neural network (CNN) that estimates camera poses for relocalization. Walch *et al*. [49] combined a CNN with LSTMs [18] to estimate camera poses. While deep-learning-based approaches have shown impressive results, they still require specialized equipment (*e.g*., GPUs) to train them, struggle to generalize on unseen scenes, and are computationally expensive for mobile devices. In contrast, our proposed method does not require specialized equipment, has an explainable or interpretable compression model, and allows estimators to compute accurate poses quickly by using an SMO-based solver.

## 3. Efficient Scene Compression

Our solution aims to select points that are visually distinct and far away from each other. Visually distinct points help feature matchers produce good correspondences, and consequently, produce good pose estimates. Having some distance between pairs of points enforces the solution to cover most of the scene and benefits the pose estimator. This is because 3D points that are far from each other often produce 2D-3D correspondences that impose constraints yielding good pose estimates. To find the set of 3D points that satisfy the aforementioned constraints, we present a novel convex optimization problem that can be solved efficiently. Unlike the state of the art which builds on the $K$-cover problem, the proposed formulation aims to learn a sparse discrete probability distribution over the set of points. This learned distribution has non-zero values on the selected points and zero on the points that are discarded.

Mathematically, we aim to learn a sparse distribution $\boldsymbol{\alpha}$ over the $m$ 3D points of the input point cloud. This sparse distribution has non-zero probabilities on those points that are considered visually distinct and have a reasonable average spatial distance with respect to their closest neighbors. To formulate this problem, we need to define terms that measure the spatial distance to their neighbors and visual distinctiveness of each of the points as a function of $\boldsymbol{\alpha}$. To measure the spatial distance among pairs of points, we use the following term:

$$C = \boldsymbol{\alpha}^\intercal K \boldsymbol{\alpha}, \tag{1}$$

where the entries of the matrix $K \in \mathbb{R}^{m \times m}$ are

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right), \tag{2}$$

$\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^3$ are two point positions, and $\sigma$ is a parameter that controls when two points are considered close enough. Because the matrix $K$ is an RBF kernel [39], the spatial distance term $C$ ranges between 0 and 1. It decreases when two points are far away and increases when two points are close to each other. To fulfill the goal of keeping points that are far apart of each other, we need to minimize $C$.

The term $C$ can be interpreted as the expected spatial distance score among pairs of points, *i.e*.,

$$\mathbb{E}\left[k(\mathbf{x}_i, \mathbf{x}_j)\right] = \sum_{i,j} k(\mathbf{x}_i, \mathbf{x}_j) p(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\alpha}^\intercal K \boldsymbol{\alpha} = C \tag{3}$$

where $\mathbb{E}[\cdot]$ is the expectation operator, and $p(\mathbf{x}_i, \mathbf{x}_j) = p(\mathbf{x}_i)p(\mathbf{x}_j) = \alpha_i \alpha_j$ is the joint probability encoding the chances that the point pair $(\mathbf{x}_i, \mathbf{x}_j)$ is selected. By minimizing $C$ over $\boldsymbol{\alpha}$, we are indirectly maximizing the expected distance between the selected point pairs. Thus, by minimizing $C$ we enforce the algorithm to learn a distribution $\boldsymbol{\alpha}$ that aims to maximize the expected pairwise distance between points in the input point cloud. Since the algorithm learns a sparse distribution $\boldsymbol{\alpha}$, the algorithm thus learns to select only a handful of 3D points from the point cloud.

To keep the most visually distinctive or "easy to match" 3D points, we use the following term:

$$D = \mathbf{d}^\intercal \boldsymbol{\alpha}, \tag{4}$$

where $\mathbf{d} \in \mathbb{R}^m$ is a vector holding a visual distinctiveness score for every point in the input point cloud. Because our goal is to keep the most visually distinctive points, we need to maximize this term. Examples of visual distinctiveness scores are the average of the matching scores (*e.g*., descriptor distances) of every 3D point or the amount of 2D image features associated to a 3D point.

The term $D$ also can be interpreted as the expected visual distinctiveness score of the selected points, *i.e*.,

$$\mathbb{E}\left[d_i\right] = \sum_i d_i p(\mathbf{x}_i) = \sum_i d_i \alpha_i = \mathbf{d}^\intercal \boldsymbol{\alpha} = D, \tag{5}$$

where $d_i$ is the visual distinctiveness score for the $i$-th point, and $\alpha_i = p(\mathbf{x}_i)$ is the probability of selecting the $i$-th point.

As stated above, we want to minimize $C$ (*i.e*., maximize the expected spatial distance among pairs of points) and maximize $D$ (*i.e*., keep the most visually distinctive points). To this end, we use the following cost function:

$$J = C - \tau D = \boldsymbol{\alpha}^\intercal K \boldsymbol{\alpha} - \tau \mathbf{d}^\intercal \boldsymbol{\alpha}, \tag{6}$$

where $\tau$ is a scalar that controls the trade-off between spatial distance term $C$ and the visual distinctiveness $D$. By minimizing $J$ over $\boldsymbol{\alpha}$, we minimize $C$ and maximize $D$. When visual distinctiveness is more important, then $\tau$ must be high. On the other hand, when the spatial distance term is more important, then $\tau$ must be near zero.

Authorized licensed use limited to: West Virginia University. Downloaded on December 26,2021 at 21:47:51 UTC from IEEE Xplore. Restrictions apply.

To learn the sparse distribution $\boldsymbol{\alpha}$ that minimizes $J$, we solve the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{minimize}} \quad & \boldsymbol{\alpha}^{\intercal}\mathbf{K}\boldsymbol{\alpha} - \tau\mathbf{d}^{\intercal}\boldsymbol{\alpha} \\ \text{subject to} \quad & \sum_{i}^{m}\alpha_i = 1, \\ & 0 \leq \alpha_i \leq \frac{1}{\nu m}; \quad i = 1, \dots, m, \end{aligned} \tag{7}$$

where $\nu \in (0, 1]$ is the compression factor, a scalar that controls the sparsity of the distribution $\boldsymbol{\alpha}$. This parameter thus controls the compression rate. This is because when $\nu = 1$, then $\boldsymbol{\alpha}$ becomes a uniform distribution, which is equivalent to no compression. On the other hand, when $\nu < 1$, then we allow the algorithm to put more mass on a few points. In this case, this is equivalent to select only a few points which reduces the size of a point cloud.

The problem shown in Eq. (7) is a quadratic program (QP). As such, our compression problem is convex. This is because the RBF kernel matrix $K$ is positive-semi-definite matrix [39], and the problem has linear equality and inequality constraints. Any convex solver (*e.g.*, Newton-like solvers [3]) can be used to find $\boldsymbol{\alpha}$. Unfortunately, these methods do not scale well when the number of unknown variables is large. Nevertheless, we can exploit the intimate relationship that this QP problem has with one-class SVMs to derive an efficient solver.

## 3.1. Relation with One-class SVMs

The proposed problem in Eq. (7) has a direct relationship to one-class classifiers. We can obtain the exact one-class SVM dual formulation [40] by setting $\tau = 0$. With this setting, we omit the linear term $D$ and only keep the spatial distance term $C$. This reveals that the proposed approach with this setting reduces a point cloud by keeping the support vectors. Recall that the support vectors have a corresponding non-zero entry in $\boldsymbol{\alpha}$, and are the points that allow an SVM to define a decision boundary for recognition.

While this relationship provides insight as to how the proposed algorithm operates, it also enables an opportunity to derive an efficient solver. This is because efficient and scalable solvers that train an SVM exist, *e.g.*, the sequential-minimal optimization (SMO) [34]. Unfortunately, we cannot directly use the one-class-SVM SMO solver for the proposed problem. There are two reasons that limit the SMO solver for our proposed problem. The first one is that the original one-class-SVM SMO solver does not consider a linear term; in our case the distinctiveness term $D$. The second reason is that the SMO uses the SVM decision rule to determine efficient variable updates. Our proposed problem is not a classification one. As such, our problem does not have a decision rule. Nevertheless, as we discuss in the next section, it is still possible to derive an SMO-like solver that can efficiently solve the proposed problem.

## 3.2. Efficient SMO-like Solver

Inspired by the SMO solver, we aim to formulate the simplest sub-problem that we can sequentially solve at a time. By solving these sub-problems sequentially, we can find the solution for the proposed problem. As shown by Platt [34], the simplest problem that we can solve in an SVM involves a quadratic program with only two variables. The advantage of solving a QP problem with two variables is that we can solve it analytically. Consequently, we avoid expensive matrix operations (*e.g.*, matrix inversions and multiplications) which are fundamental operations in Newton-based optimization methods.

To obtain the simplest QP problem with two variables, we need to algebraically manipulate Eq. (6). Recall that the goal is to obtain a cost function $J'$ that focuses on only two variables: $\alpha_i$ and $\alpha_j$ which are the $i$-th and $j$-th entries of $\boldsymbol{\alpha}$, respectively, and $i \neq j$. After algebraic manipulations, we obtain the following $J'$:

$$\begin{aligned} J'(i,j) = {} & \alpha_i^2 + 2\alpha_i\alpha_j K_{ij} + \alpha_j^2 \\ & + 2\alpha_i \sum_{l \neq i, l \neq j} \alpha_j K_{il} + 2\alpha_j \sum_{l \neq i, l \neq j} \alpha_l K_{jl} \\ & - \tau d_i \alpha_i - \tau d_j \alpha_j + g\left(\{\alpha_t : t \neq i, j\}\right), \end{aligned} \tag{8}$$

where $g(\cdot)$ is a function including all the remaining entries $\{\alpha_t : t \neq i, j\}$ in $\boldsymbol{\alpha}$. For the full derivation of $J'$, we refer the reader to the supplemental material.

The original problem shown in Eq. (7) aims to learn a probability distribution $\boldsymbol{\alpha}$. Since $J'$ focuses on only two variables, we need to update the equality constraints when we only optimize for the two entries $\alpha_i$ and $\alpha_j$. To do this, we need to ensure that the sum of all the entries in $\boldsymbol{\alpha}$ equals to one. At the same time we still need to ensure the inequality constraints shown in Eq. (7). After considering these aspects, we obtain the following the problem:

$$\begin{aligned} \underset{\alpha_i, \alpha_j}{\text{minimize}} \quad & J'(i,j) \\ \text{subject to} \quad & \alpha_i + \alpha_j = \Delta, \\ & 0 \leq \alpha_i, \alpha_j \leq \frac{1}{\nu m}, \end{aligned} \tag{9}$$

where $\Delta$ is the joint probability mass between $\alpha_i$ and $\alpha_j$. This means that we can minimize $J'$ as long as we maintain the probability mass $\Delta$ between $\alpha_i$ and $\alpha_j$ constant. Similar to the SMO, this constraint imposes a solution over a line $\alpha_i + \alpha_j = \Delta$ and keeps a valid sum: $\sum_i \alpha_i = 1$. This is because the solver assumes that the starting probability distribution $\boldsymbol{\alpha}$ is feasible, *i.e.*, it sums up to one and satisfies the inequality constraints.

Authorized licensed use limited to: West Virginia University. Downloaded on December 26,2021 at 21:47:51 UTC from IEEE Xplore. Restrictions apply.

Similar to the SMO algorithm, the proposed algorithm needs to iteratively select a pair of variables $\alpha_i$ and $\alpha_j$, and solve the problem shown in Eq. (9). To solve this simplified problem analytically, we leverage the equality constraint to set $\alpha_j = \Delta - \alpha_i$. Via substitution, we simplify the cost function $J'(\alpha_i)$:

$$\begin{aligned} J'(\alpha_i) &= \alpha_i^2 + 2\alpha_1 (\Delta - \alpha_i) K_{ij} + (\Delta - \alpha_i)^2 \\ &\quad + 2\alpha_i \sum_{l\neq i, l\neq j} \alpha_l K_{il} \\ &\quad + 2(\Delta - \alpha_i) \sum_{l\neq i, l\neq j} \alpha_l K_{jl} \\ &\quad - \tau (d_i \alpha_i + d_j (\Delta - \alpha_i)) + g(\{\alpha_t : t \neq i, j\}). \end{aligned} \tag{10}$$

Given that $J'(\alpha_i)$ is a function of a single variable, we can obtain the optimal $\alpha_i^\star$ analytically by solving $\frac{\partial J'}{\partial \alpha_i} = 0$. This analytical solution is:

$$\alpha_i^\dagger = \frac{1}{2}\left(\frac{T}{2(1 - K_{ij}) + \Delta}\right), \tag{11}$$

where

$$T = \tau(d_i - d_j) - 2\sum_{l\neq i, l\neq j} \alpha_l K_{il} + 2\sum_{l\neq i, l\neq j} \alpha_l K_{jl}. \tag{12}$$

The solution $\alpha_i^\dagger$ does not consider the inequality constraints shown in Eq. 7. To enforce these inequality constraints, we box the solution $\alpha_i^\dagger$, *i.e.*,

$$\alpha_i^\star = \max\left(0, \min\left(\min\left(\frac{1}{\nu m}, \Delta\right), \alpha^\dagger\right)\right). \tag{13}$$

The $\min(\cdot)$ operations above ensures that the upper bounds are satisfied. On the other hand, the $\max(\cdot)$ operation makes sure the lower bound is enforced. Finally, we compute $\alpha_j = \Delta - \alpha_i^\star$.

## 3.3. Algorithm

To find the probability distribution $\boldsymbol{\alpha}$ and identify the 3D points to keep, we need to solve the QP problem shown in Eq. (7). The solution of this QP problem requires selecting a sequence of pairs of points and solving a simpler QP problem for each pair (see Eq. (9)). The solution of each of the simpler QP problems can be computed via Eq. (11) and Eq. (13). Our proposed SMO-solver is guaranteed to converge as it can be seen as a special case of the generalized SMO algorithm [21]. Algorithm 1 summarizes the SMO-based point-cloud compression procedure.

**Initialization.** Given the set of $m$ 3D points and their corresponding visual distinctiveness scores $\mathcal{M} = \{(\mathbf{x}_i, d_i)\}_{i=1}^m$, the efficient scene compression algorithm first initializes

**Algorithm 1:** Efficient Scene Compression

```
   Input      : Set of $m$ 3D points and their corresponding
                distinctiveness scores $\mathcal{M} = \{(\mathbf{x}_i, d_i)\}_{i=1}^m$.
   Output     : Probability distribution $\boldsymbol{\alpha} \in \mathbb{R}^m$.
   Parameters: Compression factor $\nu$
                Trade-off factor $\tau$
                RBF kernel bandwidth $\sigma$
1  // Initialize with a feasible probability distribution.
2  $\boldsymbol{\alpha}$ = InitializeProbability($\mathcal{M}, \nu$)
3  repeat
4      $i, j$ = SelectPair($m$)
5      $\Delta = \alpha_i + \alpha_j$
6      // Compute Eq. (13).
7      $\alpha_i^\star$ = ComputeProbability($i, j, \Delta, \nu, m, \sigma, \mathcal{M}$)
8      // Update probability pair.
9      $\alpha_i = \alpha_i^\star$
10     $\alpha_j = \Delta - \alpha_i$
11 until Convergence;
```

$\boldsymbol{\alpha}$ with a feasible probability distribution satisfying the inequality constraints from Eq. (7) (see step 2 in Algorithm 1). To satisfy the inequality constraints, the algorithm first sets $\boldsymbol{\alpha} = \mathbf{0}$. Subsequently, it ranks the $m$ 3D points based on the visual distinctiveness score vector $\mathbf{d}$. Then, the algorithm selects the $n = \lceil \nu m \rceil$ most visually distinctive points by using their corresponding scores in $\mathbf{d}$, and sets their corresponding probability entries in $\boldsymbol{\alpha}$ to $\frac{1}{\nu m}$. When the sum of the probabilities exceeds 1, then the algorithm identifies one of the selected $n$ points and updates its probability entry such that the sum of the entries in $\boldsymbol{\alpha}$ equals 1. Using the $n$ most visually distinctive points to initialize $\boldsymbol{\alpha}$ helps the algorithm search for the solution from a set of points that are likely to match more easily and quickly. This initialization method is inspired on SVM initialization methods [35].

**Pair selection.** Given the initialized $\boldsymbol{\alpha}$, the algorithm starts solving the sequence of simpler QP problems (steps 2 - 11). To construct a simple QP problem shown in Eq. (9), the algorithm first selects a pair of points $(i, j)$. The simplest selection process is a random pick of two different points. A more sophisticated process selects a pair of points such that at least one of their corresponding probabilities is non-zero. This sophisticated process likely will decrease the number of iterations since a pair with zero probabilities does not produce an update on $\boldsymbol{\alpha}$. However, this process requires tracking the non-zero probabilities which increases the computational complexity.

**Updating pair probabilities.** Given a pair $(i, j)$, the algorithm first computes the probability mass $\Delta$ that needs to be redistributed (step 5). Subsequently, the algorithm computes the probability $\alpha_i^\star$ for the $i$-th point using Eq. (13) and assigns it to $\alpha_i$ in step 9. Finally, the algorithm computes the probability $\alpha_j$ for the $j$-th point in step 10.

Similar to modern SVM solvers [9, 19], the implementation of the efficient scene compression summarized in Algorithm 1 uses a cache to alleviate the storage and compu-

Authorized licensed use limited to: West Virginia University. Downloaded on December 26,2021 at 21:47:51 UTC from IEEE Xplore. Restrictions apply.

tational cost of the RBF kernel function. Using this cache avoids storing the kernel matrix $K$ which can become prohibitively expensive as it requires $\mathcal{O}(m^2)$ storage cost.

The theoretical stopping criterion of the algorithm is based on the Karush-Kuhn-Tucker (KKT) [3] conditions of a QP. Unfortunately, evaluating the KKT conditions given a large set of $m$ points also requires a large storage footprint. Consequently, using this theoretical stopping criterion becomes impractical. Unlike the supervised classification or regression SVM problems that have stopping criteria based on the expected targets, the scene compression problem lacks any of these expected targets. Consequently, the stopping criteria from the publicly available SVMs do not apply for this problem. Inspired by recent large-scale optimization techniques used in deep learning [2, 15, 25], the implementation of the optimal scene compression procedure shown in Algorithm 1 runs the loop in steps 3 to 11 for a fixed number of iterations.

## 4. Experiments

The goals of the experiments in this section are to measure 1) the compression times; 2) storage footprint of a compressed scene; 3) the localization rate; and 4) the accuracy of the estimated poses using the compressed scenes.

**Datasets.** We used Kings College, Old Hospital, Shop Facade, and St. Mary's Church datasets from the Cambridge Landmark dataset collection [24], and Dubrovnik [29]. These datasets present different imaging conditions, a wide range of number of images, and large number of 3D points in the reconstructions. The Cambridge Landmarks datasets [24] contain a set of query images for which visual-based localization systems need to estimate their camera poses, a reference SfM reconstruction, and their corresponding set of images and features. For the Dubrovnik [29] dataset, we used the protocol presented by Li *et al.* [29] where a few images and points are removed from the full reconstructions and then used as query images. To compute the localization accuracy, we compare the estimated poses and those of the original reconstruction.

**Implementation Details.** We implemented our compression procedure[1] and an image-based localization system in C++ using the Theia SfM library [46]. We used Root-SIFT [1] descriptors, FLANN [32] as an approximate nearest-neighbor matcher, and P3P [27] as the camera pose estimator. We build a FLANN index for every dataset to better represent their Root-SIFT distribution [11]. We ran our algorithm for up to 4096 iterations because our initialization process returns a good feasible solution. We used a radial-basis-function (RBF) to map low average descriptor distances to higher values. This is needed because our approach assumes that higher visual distinctiveness scores correspond to more distinctive features.

[1]Code:https://github.com/mameratrujillo/Efficient_Scene_Compression

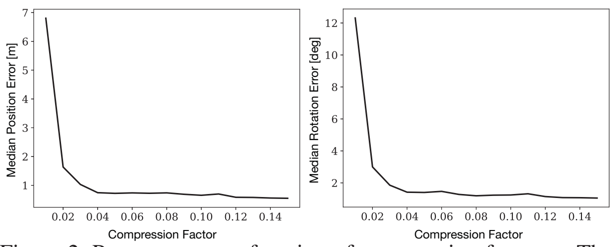

Figure 2. Pose errors as a function of compression factor $\nu$. The larger the compression factor $\nu$, the smaller the position and rotation errors (left and right, respectively).

**Visual distinctiveness scores.** The experiments use three different scores: **1)** the average descriptor distance from the pair-wise image matches coming from the SfM tracks; we use an exponential mapping to assign a high score for lower descriptor distance and a higher one for a larger distance; **2)** the fraction of cameras that see a 3D point normalized by the total number of cameras; and **3)** the fraction of cameras that see a 3D point normalized by the maximum number of cameras observing a 3D point in the input reconstruction. See the supp. material for more details on these scores.

### 4.1. Ablation Studies

The experiments in this section show the effect on localization performance of the compression factor $\nu$, spatial-distance-and-visual-distinctiveness trade-off parameter $\tau$, and the RBF kernel bandwidth $\sigma$. We use the average descriptor distance visual distinctiveness score.

**The compression factor impacts the localization performance.** We measured the position and rotation errors of a localized camera as a function of the compression factor $\nu$. For this experiment, we used the Shop Facade [24] dataset. In Fig. 2 we can see that lower compression factors $\nu < 0.05$ return higher errors. This is because the compressed scene contains fewer 3D points and this makes the image-based localizer struggle to operate optimally. However, the errors reduce considerably when we use $\nu > 0.05$.

**The spatial-distance-and-visual-distinctiveness trade-off $\tau$ tends to impact more the accuracy than the RBF kernel bandwidth $\sigma$.** For this experiment we set $\nu = 0.05$ and use the Shop Facade [24] dataset. We vary $\tau$ and $\sigma$ in the ranges $[0, 2]$ and $[1, 10]$, respectively. We set $\sigma = 1$ when we vary $\tau$, and set $\tau = 1$ when we vary $\sigma$. Fig. 3(a) and Fig. 3(b) show the localization performance as a function of $\sigma$ and $\tau$, respectively. Because this experiment computes position and rotation errors as a function of RANSAC estimates, we observe that the median position and rotation errors tend to vary around a common value for both $\sigma$ and $\tau$ (horizontal gray line). However, we observe that the std. deviations of the errors (gray shaded area) as a function of $\sigma$ and $\tau$ tend to increase as we increase $\sigma$ and $\tau$. We fit a line to these std. deviations and compare their slopes to measure their rate of growth (shown in the top left of each plot). The slopes of the position errors are comparable for both $\sigma$ and

Authorized licensed use limited to: West Virginia University. Downloaded on December 26,2021 at 21:47:51 UTC from IEEE Xplore. Restrictions apply.

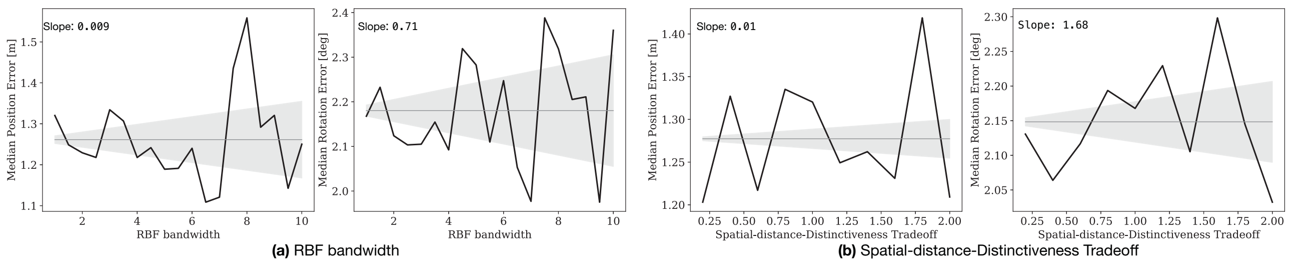
Figure 3. **(a)** Median position and rotation errors as a function of RBF kernel bandwidth $\sigma$ and **(b)** spatial-distance-visual-distinctiveness trade-off $\tau$. The errors vary around a common value (horizontal gray line) due to RANSAC, and observe that the variation increases as a function of $\tau$ and $\sigma$ (shaded gray area). The slopes of the lines enclosing the variation are larger for those of $\tau$ than that of $\sigma$, especially for rotation errors. Therefore, $\tau$ induces a larger uncertainty and thus impacting more the localization performance than that of $\sigma$.

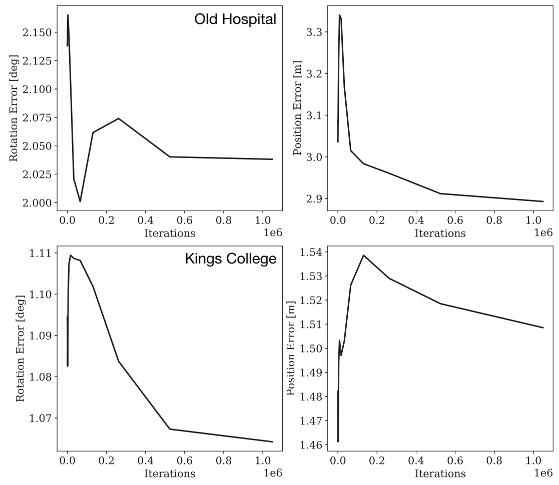
Figure 4. Rotation and position errors as a function of the number of iterations on Old Hospital (top row) and King's Collegue (bottom row). The initialization method gives a good starting solution and the optimization modestly improves pose accuracy.

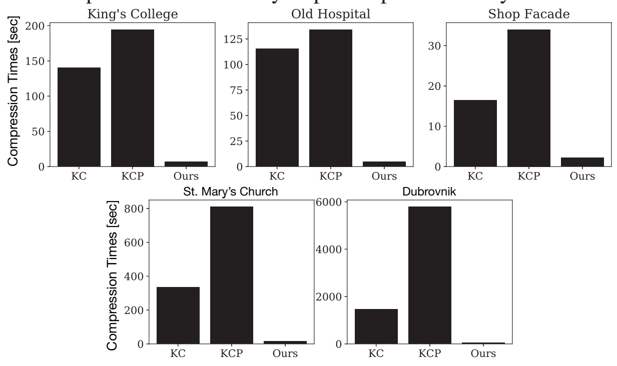
Figure 5. Compression times in seconds of KC [29], KCP [8], and our approach on Cambridge Landmarks dataset [24]. KC and KCP require a significant amount of time compared to our approach.

$\tau$. However, the slope of rotation error when varying $\tau$ is greater than that of $\sigma$ by about $2\times$. This suggests that $\tau$ affects more rotation performance than $\sigma$.

## 4.2. Compression Times and Registration Accuracy

The goals of this experiment are twofold: 1) measure the time a compression algorithm takes to reduce the size of an SfM point cloud; and 2) evaluate the localization performance using these compressed scenes.

**Our method is on average 20x faster than the k-cover-based methods.** We used the k-cover (KC) [29] and probabilistic k-cover (KCP) [8] methods as baselines. We used publicly available implementations of these methods[2]. The experiment did not include the hybrid-compression method by Camposeco *et al.* [6] since there is not a publicly available implementation. We ran these experiments on a machine with 32GB of RAM and an Intel i7 with 6 cores. Fig. 5 shows the compression times in seconds. This speed-up is due to the good initialization point described in Sec. 3.3 which mainly computes a feasible initial point using the average descriptor distance visual distinctiveness score. The time our method requires depends linearly on the number of iterations as the Sec. 3.3 states.

To measure the localization performance, we compute the rotation and position errors w.r.t. to the reference SfM reconstruction provided by the Cambridge Landmarks datasets [24] and Dubrovnik dataset [29]. We also measured the sizes of the reconstructions using file-system utilities. We considered that an image was successfully registered when it had at least 12 inliers. The baselines for this experiment include state-of-the-art-feature-based localization methods [37, 42, 47, 50], deep-learning-based ones [22, 23, 24, 49], and state-of-the-art scene-compression algorithms [5, 8, 29]. Table 1 shows the results of this experiment on the Cambridge Landmarks datasets.

**Localization accuracy marginally improves with a large number of iterations due to our good initialization procedure.** Fig.4 shows the progression of the rotation and position errors for no more than a million iterations on Old Hospital and King's College. The rotation errors have improvements that are less than one degree while the position errors have improvements in the order of centimeters.

**Our method reduces the reconstruction sizes without sacrificing localization accuracy.** Most experiments achieved 100% registration rate of all query images with the exception of the Hybrid compression [6] that achieved 98% registration rate and Active search [37] that achieved

[2]https://github.com/caosong/minimal_scene

Authorized licensed use limited to: West Virginia University. Downloaded on December 26,2021 at 21:47:51 UTC from IEEE Xplore. Restrictions apply.

Table 1. Localization performance and compression results on the Cambridge Landmarks datasets [24]. The Table presents the size in MBs of the compressed reconstructions (MB Used columns) and position and orientation errors (Errors columns). The results of the bottom 9 rows come from Camposeco *et al.* [6] while the results of the top seven rows were computed using Theia library. Our method using different visual distinctiveness scores (shown in bold) achieves a comparable localization accuracy.

| **Method** | **Kings College** | | **Old Hospital** | | **Shop Facade** | | **St. Mary's Church** | |
|---|---|---|---|---|---|---|---|---|
| | MB Used | Errors $[m, \circ]$ | MB Used | Errors $[m, \circ]$ | MB Used | Errors $[m, \circ]$ | MB Used | Errors $[m, \circ]$ |
| **Avg. descriptor distance (initial soln.)** | 2.2 | 1.48, 1.08 | 1.1 | 2.97, 2.14 | 0.41 | 0.75, 1.48 | 3.3 | 0.60, 0.89 |
| **Avg. descriptor distance** | 2.2 | 1.53, 1.09 | 1.1 | 0.90, 2.17 | 0.41 | 0.72, 1.40 | 3.3 | 0.56, 0.89 |
| **Num. cameras per point** | 2.2 | 0.94, 0.57 | 1.1 | 1.33, 0.99 | 0.41 | 0.31, 0.56 | 3.3 | 0.43, 0.64 |
| **Max. num. cameras per point** | 2.2 | 0.92, 0.59 | 1.1 | 1.24, 0.96 | 0.41 | 0.31, 0.59 | 3.3 | 0.46, 0.65 |
| KC [29] | 3.1 | 1.48, 1.23 | 6.0 | 1.35, 1.06 | 0.85 | 0.51, 0.87 | 18 | 0.46, 0.69 |
| KCP [8] | 5.9 | 0.99, 0.86 | 8.2 | 1.19, 1.00 | 1.3 | 0.44, 0.80 | 24 | 0.40, 0.61 |
| No compression | 98 | 0.57, 0.50 | 34 | 0.96, 0.79 | 11 | 0.23, 0.43 | 131 | 0.28, 0.45 |
| Hybrid comp. [6] | 1.01 | 0.81, 0.59 | 0.62 | 0.75, 1.01 | 0.16 | 0.19, 0.54 | 1.34 | 0.50, 0.49 |
| DenseVLAD [47] | 10.06 | 2.80, 5.72 | 13.98 | 4.01, 7.13 | 3.61 | 1.11, 7.61 | 23.23 | 2.31, 8.00 |
| PoseNet [24] | 50 | 1.92, 5.40 | 50 | 2.31, 5.38 | 50 | 1.46, 8.08 | 50 | 2.65, 8.48 |
| Bayes PoseNet [22] | 50 | 1.74, 4.06 | 50 | 2.57, 5.14 | 50 | 1.25, 7.54 | 50 | 2.11, 8.38 |
| LSTM PoseNet [49] | $\approx 50$ | 0.99, 3.65 | $\approx 50$ | 1.51, 4.29 | $\approx 50$ | 1.18, 7.44 | $\approx 50$ | 1.52, 6.68 |
| $\sigma^2$ PoseNet [23] | $\approx 50$ | 0.99, 1.06 | $\approx 50$ | 2.17, 2.94 | $\approx 50$ | 1.05, 3.97 | $\approx 50$ | 1.49, 3.43 |
| Geom. PoseNet [23] | $\approx 50$ | 0.88, 1.04 | $\approx 50$ | 3.20, 3.29 | $\approx 50$ | 0.88, 3.78 | $\approx 50$ | 1.57, 3.32 |
| DSAC++ [4] | 207 | 0.18, 0.30 | 207 | 0.20, 0.30 | 207 | 0.06, 0.30 | 207 | 0.13, 0.40 |
| Active Search [37] | 275 | 0.57, 0.70 | 140 | 0.52, 1.12 | 38.7 | 0.12, 0.41 | 359 | 0.22, 0.62 |

Table 2. Localization and compression performance on the Dubrovnik [29] dataset. The Table presents the same format as in Table 1. Our approach achieves comparable localization accuracy.

| | **Ours** | KC [29] | KCP [8] | No comp. |
|---|---|---|---|---|
| MB Used | 22 | 51 | 65 | 318 |
| Errors $[m, \circ]$ | 1.79, 0.56 | 1.99, 0.60 | 1.82, 0.55 | 1.31, 0.44 |

99% registration rate on Old Hospital dataset. Table 1 shows the sizes of the compressed reconstructions (MB Used columns) and the position and rotation errors (Errors columns). The last nine rows of Table 1 come from Camposeco *et al.* [6] and use different feature quantization methods, pose estimator algorithms, and a different structure-from-motion library. Thus, the sizes in MB of the compressed reconstructions and pose errors of the last nine rows are not strictly comparable. The first four rows of the Table show the results of our method with different visual distinctiveness scores: 1) initial feasible solution using the avg. descriptor distance score; 2) our method refining the result of 1); 3) fraction of number of cameras per point; 4) and fraction of number of cameras per point normalized with the max. number of cameras seen a point in the reconstruction. The fifth and sixth rows show the k-cover (KC) [29], and probabilistic k-cover (KCP) [8], respectively. The seventh row shows the localization results using a non-compressed scene. All methods aim to compress at 5% using Theia library. For KC and KCP, we computed the parameter $K$ that produced a compression close to 5% but not less. We can observe that our method achieves a similar localization accuracy compared to the accuracy of the baselines in all the datasets while reducing its storage footprint. Table 2 presents the compression sizes and localization performance on the Dubrovnik dataset; all the compression methods kept around 10% of the 3D points of the full SfM point cloud. For this experiment, our method used the avg. descriptor distance visual distinctiveness score. The last column in Table 2 shows the size of the original point cloud and localization performance without compression. We see that our method successfully reduces the size of a large-scale point cloud without sacrificing localization performance. Moreover, our method is efficient and simpler to implement than KC, KCP, and [6]. However, the good performance of [6] is due to its modified RANSAC algorithm, its weighted set cover problem, and the two sets of points as described in Sec. 2. Unlike [6], our method only requires computing a visual distinctiveness score for each point, minimize the proposed cost function via Algorithm 1, and use classical RANSAC and pose estimators.

## 5. Conclusions

We presented a simple and efficient method that compresses SfM point clouds and keeps the image-based localization accurate. Unlike the K-cover-based methods [5, 8, 29], our method operates by solving a convex quadratic program (QP) that aims to keep 3D points that present a good visual distinctiveness and that maintain a sufficient distance of each other. Our QP formulation resembles the one of one-class support vector machines (SVMs) [40]. Given the resemblance with the SVMs, our method keeps the points labeled as support vectors and derived an efficient and easy-to-implement QP solver based on the sequential minimal optimization algorithm [34]. Our experiments on small- and medium-scale datasets showed that our method operates 20× faster than [8, 29] thanks to our computed initial point, and reduces the size of a point cloud by a factor of 35x without sacrificing localization performance.

**Acknowledgements:** This work was partially funded by NSF grant IIS-1657179.

Authorized licensed use limited to: West Virginia University. Downloaded on December 26,2021 at 21:47:51 UTC from IEEE Xplore. Restrictions apply.

## References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2012. 6

[2] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proc. of Intl. Conf. on Computational Statistics*. Springer, 2010. 6

[3] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 4, 6

[4] E. Brachmann and C. Rother. Learning less is more-6d camera localization via 3d surface regression. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018. 8

[5] F. Camposeco, A. Cohen, M. Pollefeys, and T. Sattler. Hybrid camera pose estimation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018. 1, 7, 8

[6] F. Camposeco, A. Cohen, M. Pollefeys, and T. Sattler. Hybrid scene compression for visual localization. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019. 2, 7, 8

[7] F. Camposeco, T. Sattler, and M. Pollefeys. Minimal solvers for generalized pose and scale estimation from two rays and one point. In *Proc. of the European Conf. on Computer Vision*, 2016. 1

[8] S. Cao and N. Snavely. Minimal scene descriptions from structure from motion models. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2014. 2, 7, 8

[9] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011. 5

[10] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 2

[11] Q. Cui, V. Fragoso, C. Sweeney, and P. Sen. Graphmatch: Efficient large-scale graph construction for structure from motion. In *Proc. of the Intl. Conf. on 3D Vision*, 2017. 6

[12] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007. 1

[13] M. Dymczyk, S. Lynen, M. Bosse, and R. Siegwart. Keep it brief: Scalable creation of compressed localization maps. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2015. 2

[14] V. Fragoso, S. Gauglitz, S. Zamora, J. Kleban, and M. Turk. Translatar: A mobile augmented reality translator. In *Proc. of the IEEE Workshop on Applications of Computer Vision (WACV)*, 2011. 1

[15] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016. 6

[16] C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, P. Furgale, T. Sattler, and M. Pollefeys. 3d visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. *Image and Vision Computing*, 68:14–27, 2017. 1

[17] G. Hee Lee, F. Faundorfer, and M. Pollefeys. Motion estimation for self-driving cars with a generalized camera. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2013. 1

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 3

[19] T. Joachims. Making large-scale svm learning practical. Technical report, Technical report, SFB 475: Komplexitätsreduktion in Multivariaten , 1998. 5

[20] M. Kaess and F. Dellaert. Probabilistic structure matching for visual slam with a multi-camera rig. *Computer Vision and Image Understanding*, 114(2):286–296, 2010. 1

[21] S. S. Keerthi and E. G. Gilbert. Convergence of a generalized smo algorithm for svm classifier design. *Machine Learning*, 46(1-3):351–360, 2002. 5

[22] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *Proc. of the IEEE International Conf. on Robotics and Automation*, 2016. 7, 8

[23] A. Kendall, R. Cipolla, et al. Geometric loss functions for camera pose regression with deep learning. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017. 3, 7, 8

[24] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proc. of the IEEE Intl. Conf. on Computer Vision*, 2015. 2, 3, 6, 7, 8

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of the Intl. Conf. on Learning Representations*, 2015. 6

[26] L. Kneip, P. Furgale, and R. Siegwart. Using multi-camera systems in robotics: Efficient solutions to the npnp problem. In *Proc. of the IEEE Intl. Conference on Robotics and Automation*, 2013. 1

[27] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2011. 6

[28] G. H. Lee, F. Fraundorfer, and M. Pollefeys. Structureless pose-graph loop-closure with a multi-camera system on a self-driving car. In *Proc. of the IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2013. 1

[29] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *Proc. of the European Conf. on Computer Vision*, 2010. 1, 2, 6, 7, 8

[30] D. G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the IEEE Intl. Conf. on Computer Vision*, 1999. 1, 2

[31] S. Middelberg, T. Sattler, O. Untzelmann, and L. Kobbelt. Scalable 6-dof localization on mobile devices. In *Proc. of the European Conf. on Computer Vision*, 2014. 1

[32] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014. 6

[33] M. Petter, V. Fragoso, M. Turk, and C. Baur. Automatic text detection for mobile augmented reality translation. In *Proc. of the IEEE Intl. Conf. on Computer Vision Workshops*. IEEE. 1

[34] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998. 2, 4, 8

[35] J. C. Platt, J. Shawe-Taylor, A. J. Smola, R. C. Williamson, et al. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001. 5

[36] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *Proc. of the IEEE Intl. Conf. on Computer Vision*, 2011. 1, 2

[37] T. Sattler, B. Leibe, and L. Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1744–1756, 2016. 7, 8

[38] T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, and T. Pajdla. Are large-scale 3d models really necessary for accurate visual localization? In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017. 1

[39] B. Schölkopf, A. J. Smola, et al. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 2, 3, 4

[40] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt. Support vector method for novelty detection. In *Proc. of the Conf. on Advances in Neural Information Processing Systems*, 2000. 2, 4, 8

[41] H. Soo Park, Y. Wang, E. Nurvitadhi, J. C. Hoe, Y. Sheikh, and M. Chen. 3d point cloud reduction using mixed-integer quadratic programming. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*, 2013. 2

[42] L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1455–1461, 2016. 7

Authorized licensed use limited to: West Virginia University. Downloaded on December 26,2021 at 21:47:51 UTC from IEEE Xplore. Restrictions apply.

[43] C. Sweeney, J. Flynn, B. Nuernberger, M. Turk, and T. Höllerer. Efficient computation of absolute pose for gravity-aware augmented reality. In *Proc. of the IEEE Intl. Symposium on Mixed and Augmented Reality*, 2015. 1

[44] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. gdls: A scalable solution to the generalized pose and scale problem. In *Proc. of the European Conf. on Computer Vision*, 2014. 1

[45] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. Large scale sfm with the distributed camera model. In *Proc. of the IEEE Intl. Conf. on 3D Vision*, 2016. 1

[46] C. Sweeney, T. Hollerer, and M. Turk. Theia: A fast and scalable structure-from-motion library. In *Proc. of the Intl. Conf. on Multimedia*, 2015. 6

[47] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2015. 7, 8

[48] M. Turk and V. Fragoso. Computer vision for mobile augmented reality. In *Mobile Cloud Visual Media Computing*, pages 3–42. Springer, 2015. 1

[49] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. In *Proc. of the IEEE Intl. Conf. on Computer Vision*, 2017. 3, 7, 8

[50] W. Zhang and J. Kosecka. Image based localization in urban environments. In *3DPVT*, volume 6, pages 33–40. Citeseer, 2006. 7

Authorized licensed use limited to: West Virginia University. Downloaded on December 26,2021 at 21:47:51 UTC from IEEE Xplore. Restrictions apply.