applied sciences

MDPI

Article

Hash Indexing-Based Image Matching for 3D Reconstruction

Mingwei Cao [1,2,*], Haiyan Jiang [1,2] and Haifeng Zhao [1,2]

[1] Anhui Provincial Key Laboratory of Multimodal Cognitive Computation, Anhui University, Hefei 230601, China
[2] School of Computer Science and Technology, Anhui University, Hefei 230601, China
* Correspondence: caomw@hfut.edu.cn

Featured Application: Our method can be used for 3D reconstruction, metaverse, virtual reality and augmented reality.

Abstract: Image matching is a basic task in three-dimensional reconstruction, which, in recent years, has attracted extensive attention in academic and industrial circles. However, when dealing with large-scale image datasets, these methods have low accuracy and slow speeds. To improve the effectiveness of modern image matching methods, this paper proposes an image matching method for 3D reconstruction. The proposed method can obtain high matching accuracy through hash index in a very short amount of time. The core of hash matching includes two parts: creating the hash table and hash index. The former is used to encode local feature descriptors into hash codes, and the latter is used to search candidates for query feature points. In addition, the proposed method is extremely robust to image scaling and transformation by using various verifications. A comprehensive experiment was carried out using several challenging datasets to evaluate the performance of hash matching. Experimental results show that the HashMatch presents excellent results compared to the state-of-the-art methods in both computational efficiency and matching accuracy.

Keywords: 3D reconstruction; image matching; structure from motion; local feature; feature matching

Citation: Cao, M.; Jiang, H.; Zhao, H. Hash Indexing-Based Image Matching for 3D Reconstruction. *Appl. Sci.* **2023**, *13*, 4518. https://doi.org/10.3390/app13074518

Academic Editors: Marek Milosz, Jacek Kęsik and Krzysztof Koszela

Received: 28 January 2023
Revised: 28 March 2023
Accepted: 31 March 2023
Published: 2 April 2023

1. Introduction

In recent years, image matching has attracted much attention from both computer vision and artificial intelligence communities and has been used in various practical applications, such as 3D reconstruction [1], augmented reality, virtual reality [2,3], visual localization, visual tracking, object detection, image classification [4,5], and medical image analysis. In the case of 3D reconstruction [6], the components of image-based 3D reconstruction mainly consist of image matching, camera pose estimation, triangulation, and bundle adjustment. Thus, image matching is one of the most important modules in image-based 3D reconstruction [7]. The goal of image matching is to generate feature correspondences across multiple views. Once feature correspondences have been generated, the structure from motion (SFM) system can be used to recover sparse point clouds and camera parameters, then the multi-view stereo and Poisson surface reconstruction can be utilized to generate dense point clouds and texture models, respectively. As a result, we can conclude that the quality of 3D reconstruction heavily depends on image matching methods.

The standard pipeline of image matching consists of feature point detection, feature descriptor computing, feature matching, and outlier removal [8]. Each component of image matching has many potential choices. In the case of feature point detection, there are scale-invariant feature transform (SIFT) [9], speeded-up robust features (SURF) [10], KAZE features [11], oriented fast and rotated brief (ORB) [12], and learned invariant feature transform (LIFT) [13]. These features can not only locate feature points but also generate feature descriptors. Moreover, there are also some deep learning-based methods, such as boosted efficient local image descriptor (BELID) [14], repeatable and reliable detector

and descriptor (R2D2) [15], joint description and detection of local features (D2Net) [16], and SuperGlue [17]. According to reports in corresponding papers, every method has a state-of-the-art performance. For outlier removal, the ratio test [9] is the simplest but most efficient method, and it uses a constant threshold to reject incorrect feature correspondences and is widely used in various computer vision tasks. Ma et al. [18] proposed an automatic approach to remove outliers by vector field learning. This method has an excellent performance even when dealing with large-scale feature collections with heavy outliers.

For feature matching, there are also many methods. For example, brute force matching (BFM) [9] is one of the most famous methods. For a given query point, BFM chose a maximal confident candidate from reference points as the corresponding point. Usually, the feature correspondences generated by BFM have some outliers, often caused by the feature descriptor's ambiguity. Thus, BFM combined with the ratio test is a good solution for image matching. Wang et al. [19] proposed the use of a graph-matching approach to generate feature correspondences in the presence of outliers. The core idea is to reject incorrect matches by using the zero-assignment constraint. Jiang et al. [20] proposed the use of spatial clustering to generate a robust image matching result. The key idea of this method is to progressively cluster the putative matches into several motion-consistent clusters, including an outlier cluster. Zhou et al. [21] proposed progressive large-scale-invariant image matching in scale space for 3D reconstruction, resulting in a desirable performance in matching precision. Han et al. [22] proposed a deep learning-based approach to image matching as they designed a unified solution for learning feature representations and learning feature comparison functions. Although image matching has made significant progress, the topic still needs more attention, especially in image-based 3D reconstruction, and there is still much room for improvement in the accuracy and time efficiency of image matching methods.

After a deep investigation into the topic of image matching, we found that hash indexing is an efficient approach to feature descriptor matching. As a result, in this paper, we propose the design of a HashMatch based on hash indexing for image matching in 3D reconstruction. The HashMatch consists of two modules. First, constructing a big hash table for located feature points via Haar wavelet feature transform [23]. Second, indexing a potential candidate by using a short hash code for the query point. The HashMatch can significantly reduce computational costs on image matching when dealing with big image datasets. According to our experiments, the HashMatch has not only proved to be extremely fast but also has the desired matching precision and is very suitable for the processing of large-scale 3D reconstruction. In summary, the main contributions of this work are summarized as follows:

- We propose the use of hash indexing to replace the original brute force matching for image matching and present a HashMatch for 3D reconstruction.
- We propose the use of the Haar wavelet feature transformation to construct a big and robust hash table for the located feature points for feature descriptor indexing.
- We have designed a parallel architecture for hash matching to speed up the image matching process and save computing time. Additionally, a systematic experiment conducted on several benchmarking datasets is provided to assess the HashMatch against the state-of-the-art methods.

The rest of this paper is organized as follows: After a brief introduction in Section 1, related work about image matching is presented in Section 2. In Section 3, we present the HashMatch in detail. In Section 4, a systematic experiment was conducted on several benchmarking datasets to prove the effectiveness of the HashMatch. Finally, the conclusion and remarking comments are presented in Section 5.

2. Related Works

In this section, we will briefly review the most relative work to our research topic, including image matching, 3D reconstruction with SFM, MVS, and surface reconstruction.

2.1. Image Matching

Image matching has obtained great progress in the past decades due to many potential applications, so various feature-matching methods have been proposed by researchers [24,25]. In a nutshell, existing methods can be roughly divided into two categories according to the types of input data (video, and images). If the input data are in video form, the KLT and variants are usually adopted to locate feature correspondences by computing the displacements between two consecutive frames [26]. The Lucas and Kanade tracker (KLT) [27] is generally considered to be the best choice for video feature tracking, especially in robotic navigation, trajectory extraction for moving objects, and visual object tracking. Zach et al. [28] presented an adaptive KLT that was implemented in parallel programming for real-time applications. Hwangbo et al. [29] proposed the use of the inertial measurement unit (IMU) to improve the stability of KLT for a moving camera. Poling et al. [30] presented a novel framework for jointly tracking a collection of feature points, which can share information between the different feature points in the scene. This method achieves stable feature tracking via subspace constraints. Zhao et al. [31] formulated the feature tracking problem as a spatio-temporal statistical learning framework, so they proposed a robust keypoint tracking method via metric learning-driven multi-task structured output optimization. Generally, KLT and its variants can produce promising results for video sequences.

Regarding the second category, there have also been many approaches published in recent years. For example, Zhang et al. [31] proposed an efficient feature-tracking method to process a nonconsecutive image sequence for large-scale SFM, in which the parallel SIFT feature and second-pass matching strategy were adopted to handle feature-tracking drift. Lin et al. [32] developed a novel method called RepMatch for reconstructing the city model, in which the Epipolar guided approach is adopted to accommodate both wide baselines and repeated structures. Additionally, the RepMatch also utilized random sample consistency (RANSAC) to handle different true or false feature correspondences. Lukos et al. [33] proposed wide-baseline image matching via projective view synthesis for two-view image matching. This method can aptly handle the limitations caused by the approximate character of affine transformations via projective transformations, and can also use the essential matrix between images to reject incorrect feature correspondences. Recently, Lowry et al. [34] proposed to make use of local geometric priors to boost the spatial verification for filtering outliers. This method also utilized the scale and orientation information from feature neighborhoods to identify which feature points are likely to be correct feature correspondences and has proved to be excellent in practice.

Recently, the deep learning technique has provided excellent performances in various computer vision tasks, such as object detection and recognition, visual tracking and localization, and depth estimation. Thus, some researchers have tried to use deep learning techniques to handle image matching, including feature point detection, descriptor computing, and feature matching. For example, the first work of deep learning-based image matching was MatchNet, proposed by Han et al. [22], in which an end-to-end neural network architecture was designed to generate feature correspondences for image pairs. Zhao et al. [35] presented a hierarchical neural network named NM-Net to generate high-precision feature correspondences via mining reliable neighbors, in which NM-Net takes the generated graph as an input, then boosts the robustness to the order of feature correspondences. Yi et al. [36] designed a deep neural network architecture for the purpose of learning to locate good feature correspondences for wide-baseline images. They train the network in an end-to-end solution to label the feature correspondences as inliers or outliers. Sarlin et al. [17] proposed a neural network named SuperGlue to match two sets of local feature points by jointly finding feature correspondences and rejecting incorrect matches. Additionally, they also presented a flexible context aggregation mechanism by making use of visual attention to enable SuperGlue to reason the underlying geometry model. As a result, SuperGlue is an excellent solution based on deep learning for image matching. Ma

et al. [37] presented a systematic survey to analyze the research state of image matching. More information about image matching can be found in Ma's work.

2.2. 3D Reconstruction

Reconstructing geometric models from image sets is a basic task in the fields of computer vision and computer graphics, which has led to the emergence of various 3D reconstruction methods. The classic pipeline of image-based 3D reconstruction consists of sparse, dense, and surface reconstruction. Each component has different solutions or methods. The goal of sparse reconstruction is to recover sparse point clouds and camera parameters from the generated feature correspondences via SFMs. For instance, Snavely et al. [38] developed a system named Bundler to recover sparse geometry from Internet image collections. Bundler possesses an excellent implementation and has the robustness to scene scales. Wu et al. [39] present a novel SFM system that reaches linear-time reconstruction via incremental way with the parallel SIFT and multi-core bundle adjustment. In addition to fast speeds, Wu's system can generate compact point clouds by re-triangulation for drifted features. Chris et al. [40] presented a large-scale SFM for outdoor scene reconstruction, in which they make use of the distributed camera model to describe image observations in terms of light rays rather than image pixels. Zhu et al. [41] developed a very large-scale global SFM to replace incremental SFMs, in which distributed motion averaging is utilized to estimate the camera pose in a parallel manner. Johannes et al. [42] revised the development progress of SFMs, then designed an excellent incremental SFM via scene graph augmentation. According to the reports in [43,44], modern SFMs can recover a world-scale model from Internet image collections in a few days.

For dense reconstruction, the patch-based multi-view stereo (PMVS) [45] is an excellent solution, which takes local feature points and edges as seeds to densify the sparse point clouds to generate dense point clouds via point expansion. Goesele et al. [46] proposed a depth-map-fusion-based MVS for generating dense point clouds. The method first computes individual depth maps using a stereo matching approach, then the depth maps are merged into a single dense model using a direct volumetric fusion. Silvano et al. [47] developed a massively parallel method for high-quality multi-view stereo matching, with the core idea being based on the PathMatch, which uses a robust photo-consistency measure to check depth errors. Kuhn et al. [48] proposed a scalable and efficient MVS method to reconstruct an accurate dense geometry model from hundreds of high-resolution images, in which all depth maps are estimated by semi-global matching (SGM) methods and a TV previously utilized to improve the quality of MVS. Inspired by the success of deep learning in pattern recognition, some researchers have tried to design neural networks for MVS reconstruction with end-to-end approaches. For example, Yao et al. [49] proposed the use of a deep learning-based approach to inference depth maps for generating dense point clouds. This resulted in the construction of a novel neural network architecture named MVSNet. Later, they modified MVSNet with recurrent neural networks (RNN), resulting in RMVSNet [50].

Once dense point clouds have been generated, surface reconstruction methods can be utilized to generate texture models. Among the existing methods, the most famous method is Poisson surface reconstruction [51], which formulates the problem of reconstructing the textured model as the Poisson equation solver. Zhou et al. [52] proposed the use of data-parallel Octrees to accelerate Poisson surface reconstruction when dealing with large-scale scenes, subsequently resulting in parallel Poisson surface reconstruction. Waechter et al. [53] presented a novel approach to generating high-quality textures for large-scale 3D reconstructions. This method can handle the complexity and scale of reconstructed models. Ummenhofer et al. [54] developed a variational method for surface reconstruction from the MVS data, which can handle a billion points in a short time. Recently, Tom et al. [55] presented a BigSUR for large-scale structured urban reconstruction, in which a binary integer program is proposed for global balances sources of error to generate semantically parsed mass models with associated façade elements. With the rapid development of deep

learning technology, some learning-based approaches have been proposed for reconstructing texture models from image collections via end-to-end ways, such as SurfaceNet [56]. We believe that more efficient methods based on deep learning will be proposed in the future.

3. The Proposed Method

To improve the quality of modern 3D reconstruction, in this paper, we present an efficient image matching method based on hashing indexing, named HashMatch. The pipeline of the HashMatch is depicted in Figure 1, where we first use the SURF feature to find local feature points and compute feature descriptors for given image pairs. Second, Haar wavelet coding is adopted to construct a compact hash table for the located feature points and descriptors. Thirdly, the hash index replaces the traditional KNN to select the corresponding feature point from the reference feature set, and then the wrong feature correspondences are caused by repeating the query process. Note that the ratio test is also integrated into the HashMatch for rejecting outliers. The procedures of the HashMatch are presented in Algorithm 1. More details can be found in the subsequent sections.

Algorithm 1 HashMatch.

Input: Image collection $I = \{I_n | n \in [1, N]\}$
Output : Feature correspondences $M_{i,j} = \{f(p,q) | p, q \in [1, M]\}$

Step 1: Use the SURF feature to detect feature points for each image in image collection I.
Step 2: Use the SURF feature to compute feature descriptors for each located feature point.
Step 3: Create a hash table for all feature descriptors:
for idx = 1 to N':
(1) Compute Haar wavelet coefficients for each descriptor via Equation (8);
(2) Calculate the values of e_{k1}, e_{k2}, e_{k3}, and e_{k4} via Equations (9)–(12);
(3) Assign values to the members of hash table *htable* via Equations (13)–(16);
Step 4: Indexing hash table for given numbers i and j:
(1) Compute the values of id_0, id_1, and id_2 according to Equation (18);
(2) Calculate the indexing table for locating feature descriptors via Equation (19);
Step 5: Measure the difference between feature descriptor d_p and d_q via L2-distance;
Step 6: Repeat Step 4~Step 5. This should result in a set of feature correspondences for each image pairs.

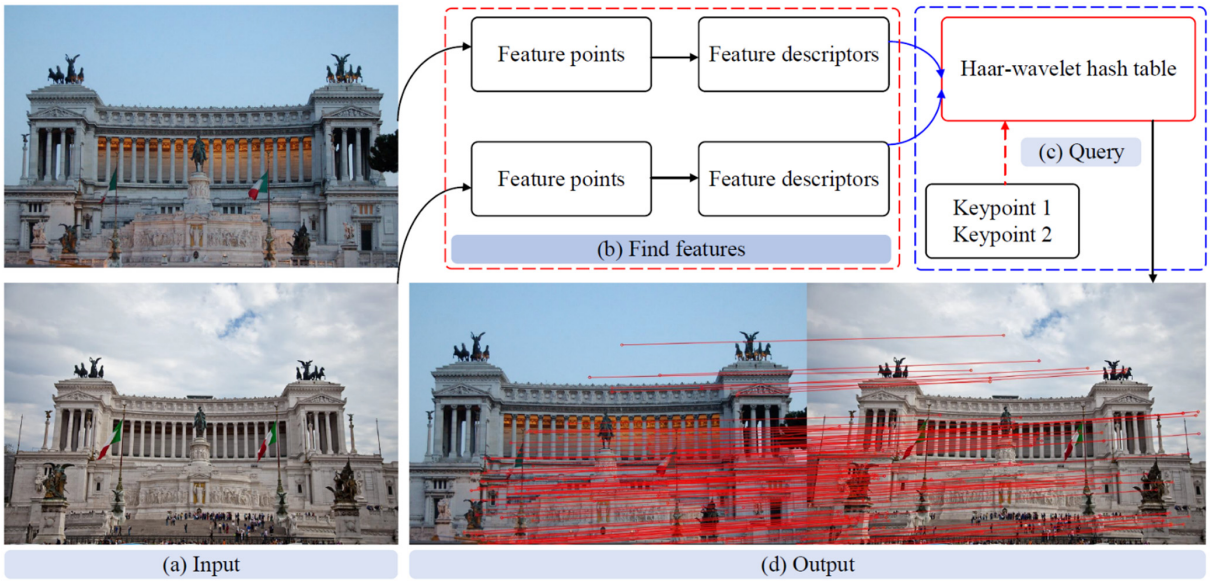

Figure 1. Flowchart of the HashMatch. The core of the proposed method consists of two parts: the first involves finding features including keypoint detection and computing feature descriptor, and the second involves matching feature descriptors by searching the created Haar wavelet hash table.

3.1. Feature Detection

Many feature point detectors can be candidates for image matching in 3D reconstruction, such as SIFT [9], ORB [12], KAZE [11], SuperPoint [57], and SuperGlue [17]. In this paper, the SURF [10] feature is adopted to find feature points and compute feature descriptors due to its fast speed and comparable matching precision to SIFT. Figure 2 presents the pipeline of the SURF detector for locating feature points from the given image. Specifically, for the input image I, the box filter is utilized to construct multi-scale spaces via different kernel sizes, and the integral image operation is also adopted to accelerate convolutional operation.

$$I_{\propto}(x,y) = \sum_{\substack{0\le x'<kernel.cols \\ 0\le y'<kernel.rows}} kernel(x',y') * I(x+x'-anchor.x, y+y'-anchor.y) \quad (1)$$

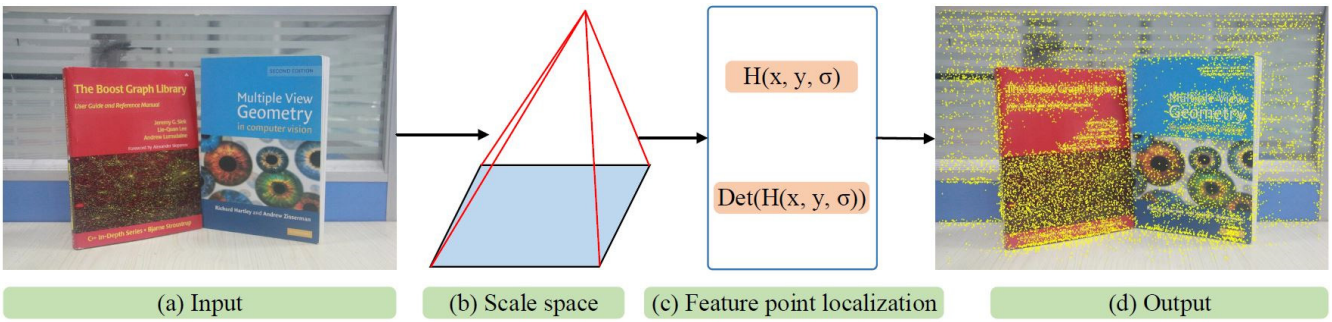

Figure 2. Flowchart of the SURF detector. The core idea of SURF consists of two steps: the first constructs a multiple-scale space for each input image, then the second locates the feature-point position.

The position of the feature point can be computed by the Hessian matrix in multi-scale spaces. The Hessian matrix $H(x,y,\sigma)$ in (x,y) at scale, σ is defined as

$$H(x,y,\sigma) = \begin{bmatrix} L_{xx}(x,y,\sigma) & L_{xy}(x,y,\sigma) \\ L_{xy}(x,y,\sigma) & L_{yy}(x,y,\sigma) \end{bmatrix} \quad (2)$$

where $L_{xx}(x,y,\sigma)$ is the convolution of the Gaussian second-order derivative $G(x,y,\sigma)$ with image I in point (x,y).

$$L_{xx}(x,y,\sigma) = \frac{\partial^2 g(x,y,\sigma)}{x^2} \otimes I(x,y,\sigma) \quad (3)$$

Notice that $L_{xy}(x,y,\sigma)$ and $L_{yy}(x,y,\sigma)$ can be calculated similarly.

To this end, we can judge whether (x,y) is a feature point according to the approximate determinant $\mathrm{Det}(H'(x,y,\sigma))$ of the Hessian matrix $H(x,y,\sigma)$.

$$\mathrm{Det}(H'(x,y,\sigma)) = L_{xx}(x,y,\sigma)L_{yy}(x,y,\sigma) - \left(0.9L_{xy}(x,y,\sigma)\right)^2 \quad (4)$$

If $\mathrm{Det}(H'(x,y,\sigma))$ is greater than all pixel values in the neighborhood of (x,y), then (x,y) is a feature point, otherwise, it is not. As depicted in Figure 2, the circles containing yellow are the located feature points.

3.2. Descriptor Extraction

The descriptor is a representation of the local feature point, some novel descriptors have been proposed in recent years. For example, Lowe's SIFT and its variants, AKAZE feature [11], local difference binary (LDB) [58], locally uniform comparison image descriptor (LUCID) [59], learned arrangements of three patch codes (LATCH) [60], learned invariant feature transform (LIFT) [13], and SURF [10]. Binary descriptors possess a fast computational efficiency but have low matching precision. Floating-point descriptors have

high matching precision but slow speed. There is an exception, namely SURF, which is a floating-point descriptor extractor, but it has comparable computational efficiency to binary descriptor extractors such as ORB and LDB. As a result, in this paper, we also use SURF to compute feature descriptors for located feature points due to it providing the best balance between distinctiveness and computational cost.

Figure 3 presents the pipeline for extracting feature descriptors for the located feature points via the SURF feature descriptor extractor. For the given feature point x $= (x, y)$ and corresponding image I, we first compute a dominant orientation θ for feature point x by computing Haar wavelet responses in the neighborhood centered at x with radius $6s$, where $s = 1.2h/9$ and h is the size of the current filter kernel. With the dominant orientation θ, we crop the patch p with the size of $20s \times 20s$ centered at x from image I, then split the patch p into 16 sub-regions along with the dominant orientation θ, as depicted in Figure 3c, where each sub-region has four-types of Haar wavelet responses, namely

$$Resp_i = \left\{ \sum_i dx, \sum_i |dx|, \sum_i dy, \sum_i |dy| \right\} \tag{5}$$

where i denotes the index of sub-regions.

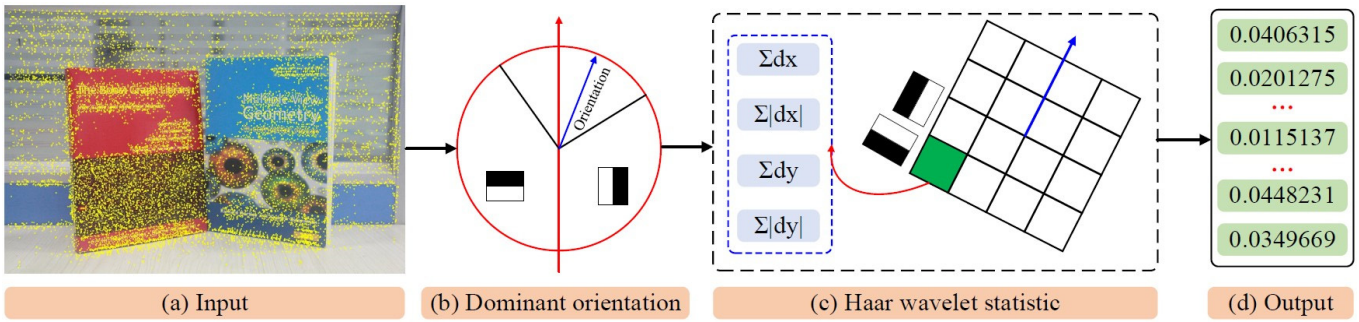

Figure 3. The pipeline of extracting feature descriptors.

To this end, we can obtain a floating-point feature vector with 64 elements.

$$D_x = \{Resp_i | i = 1, 2, \cdots, 16\} \tag{6}$$

Furthermore, we can normalize D_x by the Frobenius normal to boost its robustness to light and scale changes, and repeated texture.

3.3. Hash Matching

The goal of feature descriptor matching is to find feature correspondences for given independent feature collections. In the past decade, many approaches to feature descriptor matching have been proposed by researchers from academia. The simplest method is a k-nearest neighbor (KNN) matching with a ratio test. However, KNN is time-consuming, especially in large-scale feature collections. Johannes et al. [42] formulate feature descriptor matching as image searching for large-scale 3D reconstruction to reduce the computational cost of image matching, but matching precision may decrease in practice. Although there are some GPU-accelerated methods for image matching, they can also be improved in terms of both computational burden and matching precision. To improve the quality of image matching for boosting modern 3D reconstruction, in this paper, we propose a simple but efficient approach to feature descriptors matching via hashing indexing; thus, the proposed method is named hash indexing.

The pipeline of hash indexing is depicted in Figure 4, where feature descriptors are computed by the SURF feature [10]. For a given set of feature descriptors, the following steps should be conducted to generate feature correspondences: (1) create a hash table for all feature descriptors that included the initial hash table, feature transformation, and updated

feature bins; (2) perform hash matching, specifically searching feature correspondences using the hash value of feature descriptors. In the implementation, we can make use of the Haar wavelet to create an approximate hash table for the acceleration of feature descriptors that match further. According to the report in Section 4, the proposed method outperforms the state-of-the-art methods in both efficiency and accuracy. It is salient to note that hash matching belongs to similar matching rather than brute-force matching; thus, it has a faster speed and desired matching precision.

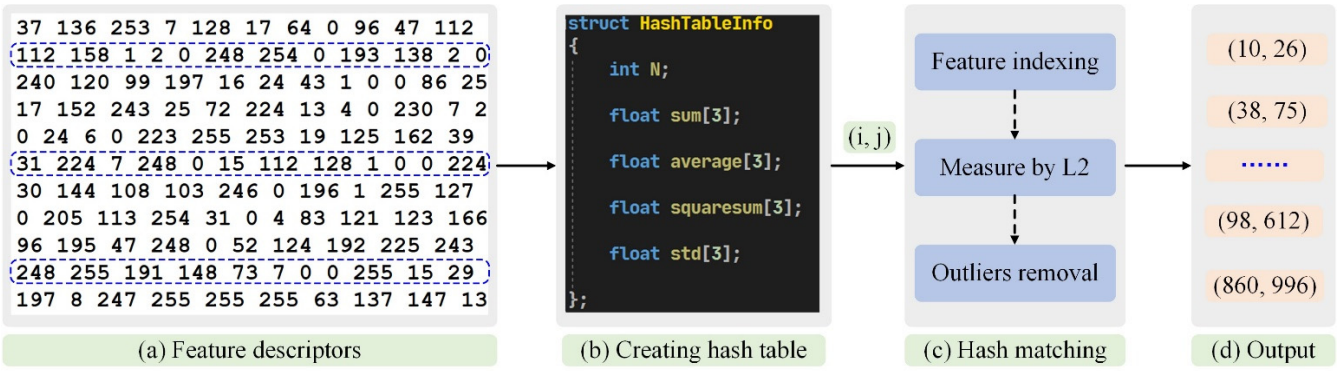

Figure 4. Flowchart of hash matching. Where (**a**) denotes the input feature descriptors, (**b**) represents the structure of the hash table in C-plus-plus language, (**c**) represents the pipeline of hash matching, and (**d**) denotes the output indexes of feature correspondences.

3.3.1. Creating a Hash Table

Descriptor matching is still a challenging task in large-scale 3D reconstruction. Although some novel approaches have been proposed for accelerating descriptor matching in image matching, these methods need further improvements regarding how they deal with high-resolution images. To reduce the computational costs of descriptor matching, in this section, we propose the creation of a big hash table to accelerate image matching via hash indexing, then boost the computational efficiency of image-based 3D reconstruction. Specifically, let *htable* be a hash table for the given descriptor collections $d = \{d_1, \cdots, d_N\}$, and the *htable* is defined as (in C-Plus-Plus language):

$$htable = \{int\ N; float\ sum[3]; float\ avg[3]; float\ squsum[3]; float\ std[3]\} \tag{7}$$

where N is the number of feature points, $sum[3]$ is used to store the sum of Haar wavelet coefficients of all feature descriptors and $squsum[3]$ is utilized to store the squared sum of Haar wavelet coefficients of all feature descriptors. Let c_{k1}, c_{k2}, and c_{k3} represent the Haar wavelet coefficients for the descriptor $d_k \in d$, which can then be calculated using the following formulas,

$$\begin{cases} c_{k1} = e_{k1} + e_{k2} - e_{k3} - e_{k4} \\ c_{k2} = -e_{k1} + e_{k2} - e_3 + e_{k4} \\ c_{k3} = e_{k1} - e_{k2} - e_{k3} + e_{k4} \end{cases} \tag{8}$$

Additionally, e_{k1}, e_{k2}, e_{k3}, and e_{k4} are the sum of descriptor elements and are defined as

$$e_{k1} = \sum_{i=0}^{i=3} \sum_{j=0}^{j=3} d_k[i * 8 + j] \tag{9}$$

$$e_{k2} = \sum_{i=0}^{i=3} \sum_{j=4}^{j=7} d_k[i * 8 + j] \tag{10}$$

$$e_{k3} = \sum_{i=4}^{i=7} \sum_{j=0}^{j=3} d_k[i * 8 + j] \tag{11}$$

$$e_{k4} = \sum_{i=4}^{i=7} \sum_{j=4}^{j=7} d_k[i * 8 + j] \tag{12}$$

Upon finding the values of c_{k1}, c_{k2}, and c_{k3}, the members of *htable* can be computed as

$$sum[index] = \sum_{k=0}^{k=N} c_{k1}, index = 0, 1, 2 \tag{13}$$

$$squsum[index] = \sum_{k=0}^{k=N} c_{k1} * c_{k1}, index = 0, 1, 2 \tag{14}$$

$$std[index] = \sqrt{\frac{squsum[index]}{N} - sum[index] * sum[index]}, index = 0, 1, 2 \tag{15}$$

$$avg[index] = \frac{sum[index]}{N}, index = 0, 1, 2 \tag{16}$$

To this end, we can create a big hash table based on Haar wavelet transformation for the input feature descriptor collections for fast indexing in the image matching stage.

3.3.2. Hash Indexing

The final stage of hash matching is to index hash table *htable* for the given feature descriptor $d_k \in d$. Brute force searching is the simplest method for indexing the hash table, but it may demand high computational costs. To mitigate this, we propose the construction of an index table *indtable* for the purpose of accelerating image matching. The *indtable* is a multi-dimensional array, which can be defined as (in C Plus Plus language) $indtable := std :: vector\langle std :: vector\langle std :: vector\langle int\rangle\rangle\rangle()$. Specifically, let inv_std_{index}=$1/std_{index}$, $index = 0,\ 1,\ 2$, and id_0, id_1, and id_2 represent indexing numbers. t_0, t_1, and t_2 denote temporal variables, namely

$$\begin{cases} t_0 = indinv\_std_0 * (c_{k0} - avg[0]) \\ t_1 = indinv\_std_1 * (c_{k1} - avg[1]) \\ t_2 = indinv\_std_2 * (c_{k2} - avg[2]) \end{cases} \tag{17}$$

The values of id_0, id_1, and id_2 can be calculated by the following formulas.

$$\begin{cases} id_0 = \min(9, \max(0, floor(1.5 * (t_0 + 3.333)))) \\ id_1 = \min(9, \max(0, floor(1.5 * (t_1 + 3.333)))) \\ id_2 = \min(9, \max(0, floor(1.5 * (t_2 + 3.333)))) \end{cases} \tag{18}$$

where $\min(x, y)$ and $\max(x, y)$ represents the minimal and maximal values between x and y, respectively. To obtain an integer that is less than x, $floor(x)$ is used. Let $bins = std :: vector\langle std :: vector\langle int\rangle\rangle()$, and $index_t = 100 * id_0 + 10 * id_1 + id_2$. Thus, $bins[index_t] = k$, where k represents the indexing number of the kth feature descriptor in the image I_n, $n \in [0, N-1]$. As a result, the value of the indexing table can be assigned using the following formula.

$$indtable[n] = bins_n,\ n \in [0, N-1] \tag{19}$$

Once we have the indexing table, we can quickly obtain the descriptor collections for the image I_n and image I_{n+1} via hash indexing. The remaining operations of feature descriptor matching are similar to [23]. More detailed information which demonstrates the performance of the HashMatch can be found in Section 4, where a comprehensive experiment conducted on several challenging datasets is provided for readers.

4. Experimental Results

The HashMatch was implemented in Visual C++ 2017 with OpenCV SDK 4.1, and Nvidia CUDA SDK 10.0 version was also used to accelerate the process of image matching. Furthermore, we evaluated the HashMatch on the benchmarking dataset and the open-source dataset and also made a comprehensive comparison to KNN + RatioTest [9], ENFT [61], MODS [62], RepMatch [32], CODE [63], D2D [64], and R2D2 [15]. It should be noted that, during the experiment, each algorithm used the original parameters.

4.1. Evaluation on the EdgeFoci Dataset

The EdgeFoci dataset [65] is a benchmark for evaluating the performance of local feature detectors and descriptors. It consists of nine sequences and each sequence has a variant number of images. The samples of the EdgeFoci dataset are provided in Figure 5, they are Boat, Graffiti, Light, Dame, Obama, Painted Ladies, Rushmore, and Yosemite, respectively. The partially matching results are provided in Figure 6 where the BFM has the most incorrect matches. The RatioTest can reject some outliers from the matching collection produced by the BFM. The ENFT has more robustness than that of the BFM and KNN + RatioTest due to the use of a two-pass matching strategy. The HashMatch can generate the most accurate feature correspondences via Haar wavelet encoding. In the case of Notre Dame, it has symmetric structures and repeated features. The BFM produced the highest number of feature correspondences for Notre Dame, but there are too many incorrect matches in the collection of generated feature correspondences. The KNN + RatioTest also has outliers because its performance suffers from the constant threshold. If a big threshold is used by the KNN, it may produce more feature correspondences than that of the KNN with a small threshold but have more incorrect matches. Otherwise, The KNN with a small threshold can reject more incorrect feature correspondence; however, it can also reject some correct ones. Although the ENFT can filter many outliers from the collection of the initial feature correspondences by the two-pass matching strategy, a few incorrect matches also exist. However, the HashMatch has the best performance among the compared methods, providing the highest number of correct feature correspondences.

Figure 5. Image pairs from the EdgeFoci dataset.

In addition to the qualitative analysis, we also provided the statistical results (number of correct feature correspondences) for the compared methods. The results are listed in Table 1, where the higher the number of correct feature correspondences, the better the performance of the corresponding method. As the stated before, the KNN + RatioTest has the worst performance and the least amount of feature correspondences. The HashMatch has the highest number of feature correspondences (only inliers) on the whole dataset, demonstrating excellent performance. Finally, for each sequence, we recorded the computational time taken by the HashMatch. These times are depicted in Figure 7. The HashMatch can match ten thousand local features in a small amount of time. Moreover, we also provide the computational time for each method in Figure 8, where it can be seen that our proposed HashMatch has the fastest speed. To this end, both qualitative and quantitative results have proven the excellent performance of the HashMatch.

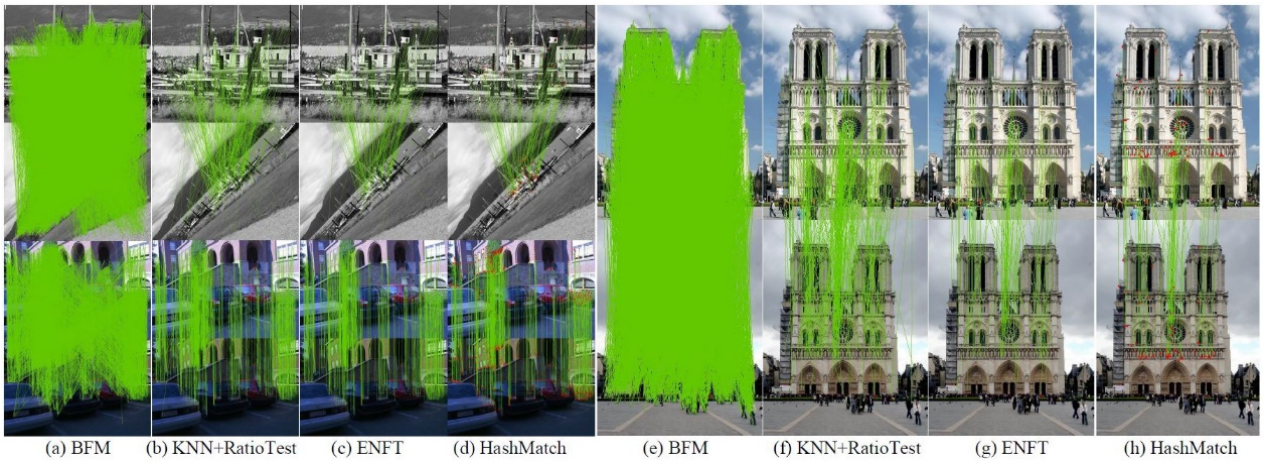

Figure 6. Matching results for Image pairs from the EdgeFoci dataset.

Table 1. Statistical results (correct matches) of the EdgeFoci dataset.

Method	Sequence							
	Boat	**Graffiti**	**Light**	**Notre Dame**	**Obama**	**Painted Ladies**	**Rushmore**	**Yosemite**
KNN + RatioTest	95	70	501	70	21	39	17	23
ENFT [61]	141	96	698	89	25	80	23	28
MODS [62]	101	83	564	142	42	47	18	32
RepMatch [32]	117	120	721	137	60	69	35	41
CODE [63]	98	109	706	110	58	53	29	47
D2D + KNN [64]	167	203	792	89	81	90	42	59
R2D2 + KNN [15]	130	178	807	120	90	83	53	62
HashMatch (Ours)	558	378	1263	307	328	407	286	613

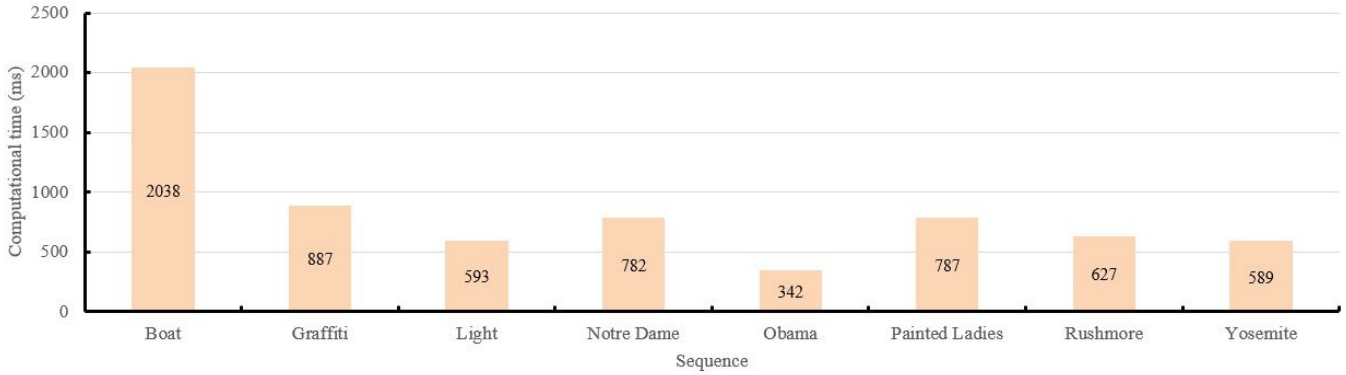

Figure 7. Computational times taken by the HashMatch on the Edge Foci dataset. These times consist of two parts, namely, feature detection time and feature matching time.

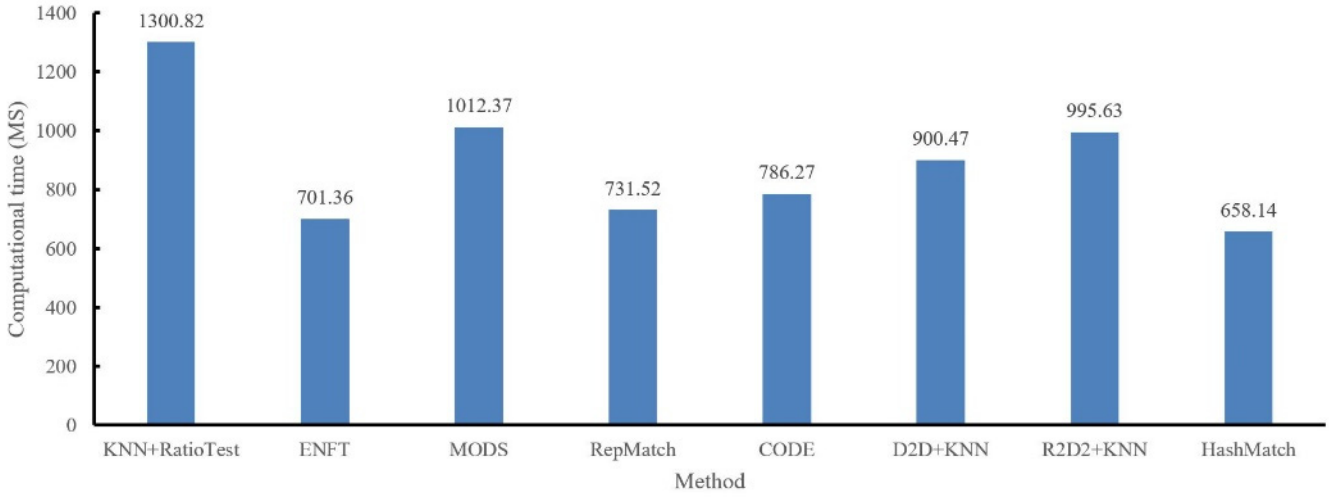

Figure 8. The calculation time of each method. According to these statistical results, the proposed method, HashMatch, has the fastest speed.

4.2. Evaluating HashMatch with Different Features

To further verify the effectiveness of the matching approach of the HashMatch, some different local features including BRISK, ORB, AKAZE, KAZE, SIFT, SURF & VGG, SIFT & VGG, have been integrated into the HashMatch, and we evaluated them using the Samper sequence of Strach's dataset (http://cvlabwww.epfl.ch/data/multiview/denseMVS.html, accessed on 10 April 2019). The statistical results are listed in Table 2, where the "Features" denote the number of input local features, and the "Matches" represent the number of generated feature correspondences. As shown in Table 2, "SURF + HM" has the highest number of feature correspondences; thus, the effectiveness of the HashMatch is proved once again. In addition to statistical results, we also provide some visualized results in Figure 9, in which all combiners can produce accurate feature correspondence. This is an excellent property of various computer vision tasks, especially in 3D reconstruction based on images.

Table 2. Statistical results (correct matches) of the Semper sequence of Strach's dataset, where HM represents HashMatch, "SURF + VGG" represents SURF detector and VGG descriptor, and "SIFT + VGG" represents SIFT detector and VGG descriptor.

Quantitative Analysis	Method							
	BRSK + HM	ORB + HM	AKAZE + HM	KAZE + HM	SIFT + HM	SURF + HM	SURF + VGG + HM	SIFT + VGG + HM
Features	19,808	20,000	11,896	11,434	38,991	45,944	45,944	38,991
Matches	3037	3857	4571	4699	6437	8539	7945	4808
Ratio of Inliers	79.34%	81.14%	88.56%	89.67	91.56	97.72	89.76	84.62

4.3. Evaluation on the Open-Source Dataset

To verify the effectiveness of the HashMatch in 3D reconstruction, we integrated it into the Bundler and evaluated it by using the open-source dataset (https://github.com/rperrot/ReconstructionDataSet, accessed on 12 April 2021). It should be noted that this dataset contains nine sequences. In our experiment, the "AvignonHotelDesMonnaies" (in the following section, we call it a "Hotel" for convenience) was adopted because it is the most challenging sequence. Figure 10 presents the samples of the Hotel sequence, there are many repeated features on the surfaces of the sampling images, and this raises a big challenge for the image matching method. The matching results of the compared methods are depicted in Figure 11, where it can be seen that the BFM has the most incorrect feature correspondences, meaning that BFM had the lowest matching precision among these image

matching methods. Although the RatioTest can remove some outliers, there are also many incorrect feature correspondences in the collection of matches. The ENFT obtained a better result than that of the KNN + RatioTest because of its use of the two-pass matching strategy. The HashMatch yielded the best results among these methods as there were no outliers in the collection of feature correspondences.

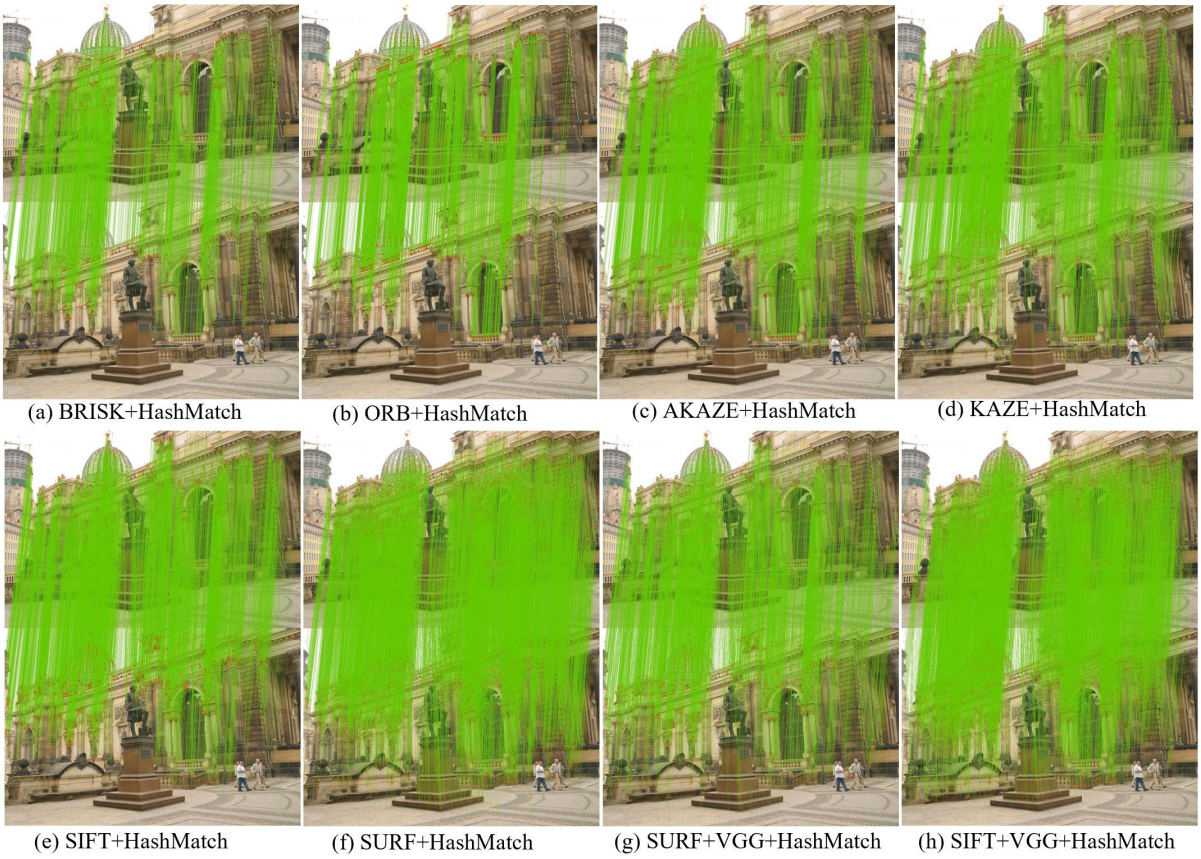

Figure 9. Feature matches of the Semper sequence, where the combination of "SURF + HashMatch" has the highest number of correct feature correspondences. Again, this proves the validity of the HashMatch method.

Figure 10. Samples of the Hotel sequence. Note that these figures have more repeated features and symmetrical structures.

Figure 11. Feature correspondences of the Hotel sequence. The HashMatch has the most correct feature correspondences. This proves that the HashMatch has more robustness to repeated features than that of the others.

Three ways are adopted to evaluate the performance of the HashMatch in 3D reconstruction, namely sparse reconstruction by Bundler, dense reconstruction by the PMVS, and surface reconstruction via the Poisson surface reconstruction. The sparse point clouds, dense point clouds, and textured models are provided in Figures 12–14, respectively. In each stage, the reconstructed model had a high geometry consistency with the real scene. Moreover, the HashMatch to one minute to generate feature correspondences. To this end, all of our results (Figures 11–14) prove the effectiveness of the HashMatch in 3D reconstruction.

4.4. Evaluation on Strecha's Dataset

To further assess the effectiveness of the HashMatch in 3D reconstruction, we integrated it into COLMAP and evaluated its effectiveness by using Strecha's dataset [66]. It should be noted that this dataset contains six sequences. In our experiment, the "Herz-Jesu-P25" is used because it is the most challenging sequence. Figure 15 presents the samples of Herz-Jesu-P25, and it can be observed that there are many repeated features on the surfaces of the sampling images, and this raises a big challenge for the image matching methods. The matching results of the HashMatch are depicted in Figure 16. Only correct feature correspondences are visible, indicating that the HashMatch had the highest matching precision compared to state-of-the-art methods. In summary, the proposed HashMatch has the best performance matching precision.

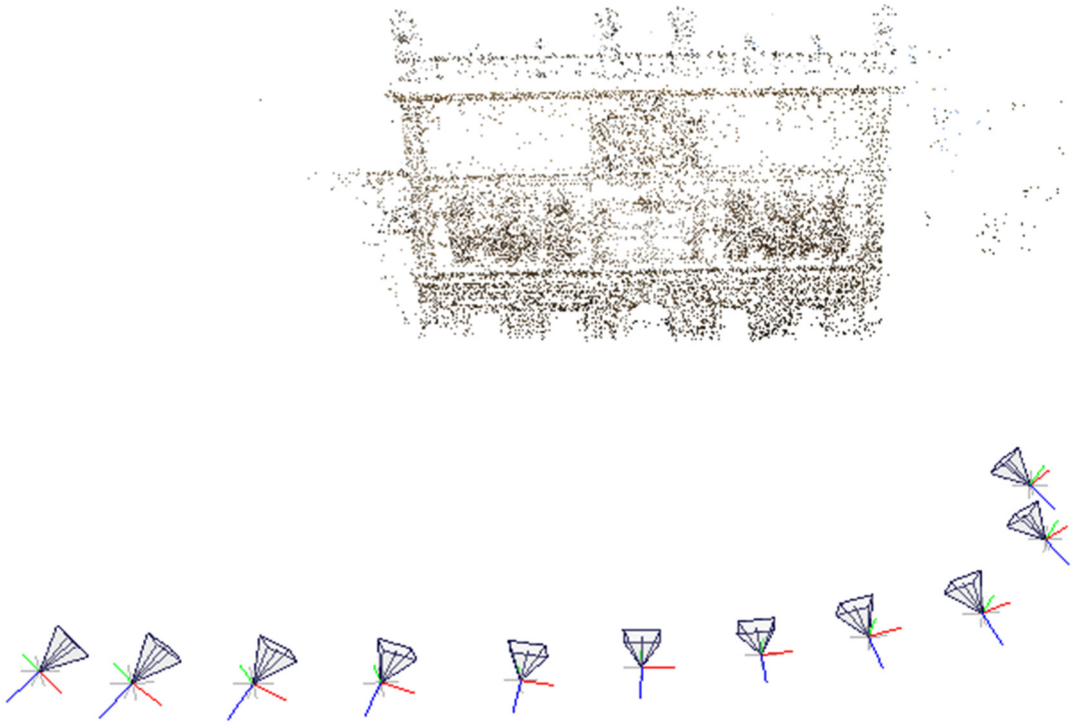

Figure 12. Sparse point clouds of the Hotel sequence, generated by Visual SfM with the HashMatch method.

Figure 13. MVS data of the Hotel sequence that was generated by the PMVS. The dense point clouds are very accurate. This is attributed to use of the HashMatch method.

Figure 14. Texture model of the Hotel sequence that was reconstructed by the Poisson surface reconstruction. The textured model has a high geometry consistency with the real scene.

Figure 15. Samples of "Herz-Jesu-P25" in Strecha's dataset. These images have many repeated features. This poses a challenge to the feature matching method.

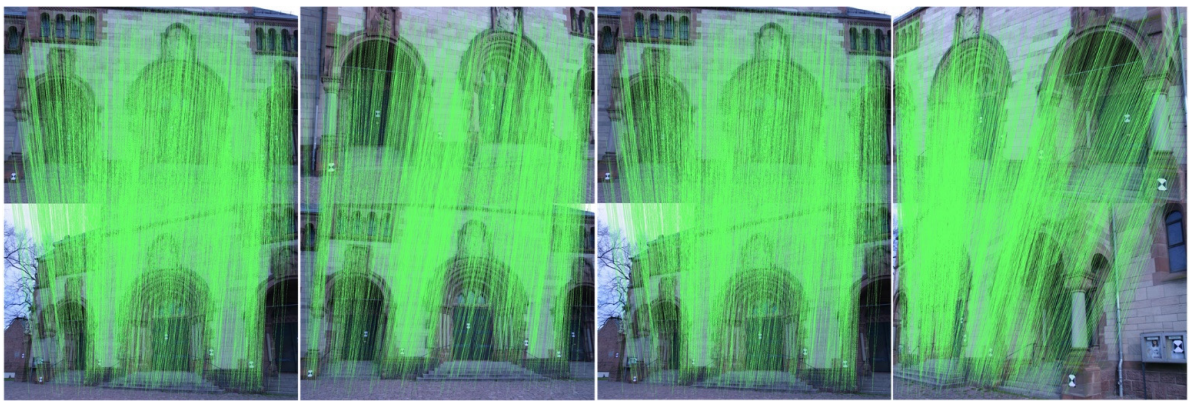

Figure 16. Feature correspondences of Herz-Jesu-P25 sequence in *Strecha's* dataset. Note that these feature correspondences are very accurate.

Three ways are used to evaluate the performance of the HashMatch in 3D reconstruction, namely sparse reconstruction by COLMAP, dense reconstruction by the PMVS, and surface reconstruction via the Poisson surface reconstruction. The sparse point clouds, dense point clouds, and textured models are provided in Figures 17–19, respectively, and in each stage, the reconstructed model has high geometry consistency with the real scene. Moreover, the HashMatch only took a few minutes (0.75 min) to generate feature correspondences. To this end, all of our results (Figures 16–19) prove the effectiveness of the HashMatch in 3D reconstruction.

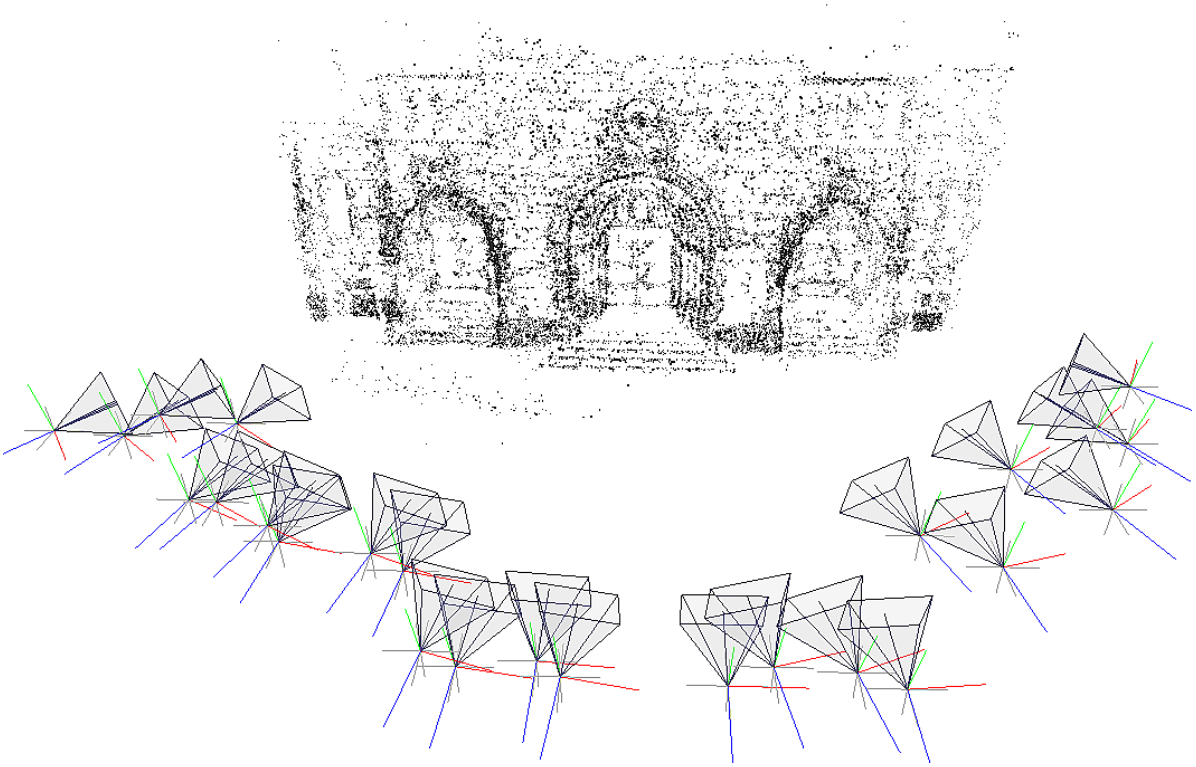

Figure 17. Sparse point clouds of Herz-Jesu-P25 sequence, generated by Visual SfM with the HashMatch. These point clouds have accurate camera poses which are very important for dense reconstruction procedures.

Figure 18. Dense point clouds of Herz-Jesu-P25 sequence.

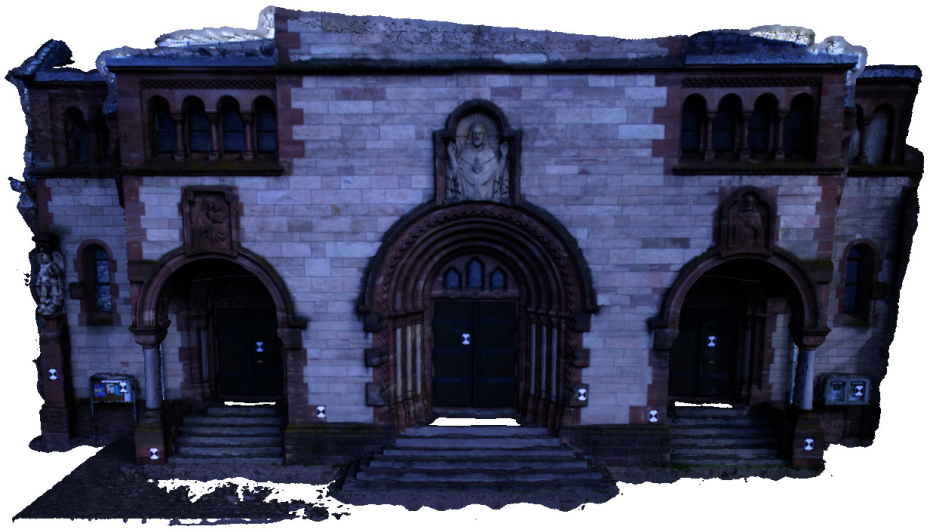

Figure 19. Texture model of Herz-Jesu-P25 sequence.

5. Conclusions

In this article, we propose an efficient image matching method for 3-D reconstruction. The proposed method was named HashMatch because it makes use of hash indexing to achieve large-scale feature matching in a short amount of time. We adopted the SURF feature to describe images, resulting in high-quality feature points and robust feature descriptors. We suggest using hash coding to generate a large hash table for the image set, so as to quickly index. Multiple verifications are utilized to reject incorrect feature correspondences. The results of systematic experiments show that the HashMatch is better than the state-of-the-art methods as it had the highest matching precision on all challenging datasets. In addition to this, the proposed method has good extensibility, which can work well with different feature detectors or feature descriptors. In short, our method is fast and accurate.

In the future, we also plan to design parallel architecture with the CUDA SDK for the HashMatch to provide more generic, fast, and accurate solutions for real-time image matching.

Author Contributions: Methodology, M.C.; supervision, H.Z.; funding acquisition, M.C., H.Z. and H.J.; Software, H.J.; investigation, H.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (No. 61876002, No. 62076005, No. 62106006, No. U20A20398), Anhui Natural Science Foundation Anhui energy Internet joint fund (No. 2008085UD07), The University Synergy Innovation Program of Anhui Province, China (No. GXXT-2020-015, No. GXXT-2021-030, No. GXXT-2021-002, No. GXXT-2021-065), National Natural Science Foundation of China and Anhui Provincial Key Research and Development Project (No. 202104a07020029)).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data underlying the results presented in this paper are not publicly available at this time but may be obtained from the authors upon reasonable request.

Acknowledgments: We are grateful for the support and help of Guofeng Zhang in providing the ENFT code.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhu, R.; Guo, Z.; Zhang, X. Forest 3D Reconstruction and Individual Tree Parameter Extraction Combining Close-Range Photo Enhancement and Feature Matching. *Remote Sens.* **2021**, *13*, 1633. [CrossRef]
2. Wang, W.; Lv, Z.; Li, X.; Han, G.; Xu, W.; Zhang, B.; Zhu, Y.; Yan, Y. Spatial Query Based Virtual Reality GIS Analysis Platform. *Neurocomputing* **2018**, *274*, S0925231217306719. [CrossRef]
3. Lv, Z.; Li, X.; Li, W. Virtual reality geographical interactive scene semantics research for immersive geography learning. *Neurocomputing* **2017**, *254*, 71–78. [CrossRef]
4. Lan, R.; Lu, H.; Zhou, Y.; Liu, Z.; Luo, X. An LBP encoding scheme jointly using quaternionic representation and angular information. *Neural Comput. Appl.* **2019**, *32*, 4317–4323. [CrossRef]
5. Lan, R.; Zhou, Y.; Liu, Z.; Luo, X. Prior Knowledge-Based Probabilistic Collaborative Representation for Visual Recognition. *IEEE Trans. Cybern.* **2020**, *50*, 1498–1508. [CrossRef]
6. Cao, M.; Zheng, L.; Jia, W.; Lu, H.; Liu, X. Accurate 3-D Reconstruction Under IoT Environments and Its Applications to Augmented Reality. *IEEE Trans. Ind. Inform.* **2021**, *17*, 2090–2100. [CrossRef]
7. Cao, M.; Zheng, L.; Jia, W.; Liu, X. Joint 3D Reconstruction and Object Tracking for Traffic Video Analysis Under IoV Environment. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 3577–3591. [CrossRef]
8. Liang, A.; Li, Q.; Chen, Z.; Zhang, D.; Zhu, J.; Yu, J.; Fang, X. Spherically Optimized RANSAC Aided by an IMU for Fisheye Image Matching. *Remote Sens.* **2021**, *13*, 2017. [CrossRef]
9. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
10. Bay, H.; Tuytelaars, T.; Van Gool, L. Surf: Speeded up robust features. In Proceedings of the European Conference on Computer Vision, Graz, Austria, 7–13 May 2006; Springer: Berlin/Heidelberg, Germany, 2006.
11. Alcantarilla, P.F.; Bartoli, A.; Davison, A.J. KAZE features. In Proceedings of the Computer Vision–ECCV, Florence, Italy, 7–13 October 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 214–227.
12. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the Computer Vision (ICCV) International Conference, Barcelona, Spain, 6–13 November 2011.
13. Yi, K.M.; Trulls, E.; Lepetit, V.; Fua, P. LIFT: Learned Invariant Feature Transform. *arXiv* **2016**, arXiv:1603.09114.
14. Suárez, I.; Sfeir, G.; Buenaposada, J.M.; Baumela, L. *BELID: Boosted Efficient Local Image Descriptor*; Springer International Publishing: Cham, Switzerland, 2019.
15. Revaud, J.; Weinzaepfel, P.; De Souza, C.; Pion, N.; Csurka, G.; Cabon, Y.; Humenberger, M. R2d2: Repeatable and reliable detector and descriptor. *arXiv* **2019**, arXiv:1906.06195.
16. Dusmanu, M.; Rocco, I.; Pajdla, T.; Pollefeys, M.; Sivic, J.; Torii, A.; Sattler, T. D2-Net: A Trainable CNN for Joint Description and Detection of Local Features. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
17. Sarlin, P.-E.; DeTone, D.; Malisiewicz, T.; Rabinovich, A. Superglue: Learning feature matching with graph neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020.
18. Zhao, J.; Ma, J.; Tian, J.; Ma, J.; Zhang, D. A robust method for vector field learning with application to mismatch removing. *CVPR 2011* **2011**, *31*, 2977–2984.
19. Wang, F.; Xue, N.; Yu, J.; Xia, G. Zero-Assignment Constraint for Graph Matching With Outliers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
20. Jiang, X.; Ma, J.; Jiang, J.; Guo, X. Robust Feature Matching Using Spatial Clustering With Heavy Outliers. *IEEE Trans. Image Process.* **2020**, *29*, 736–746. [CrossRef]
21. Zhou, L.; Zhu, S.; Shen, T.; Wang, J.; Fang, T.; Quan, L. Progressive Large Scale-Invariant Image Matching in Scale Space. In Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
22. Han, X.; Leung, T.; Jia, Y.; Sukthankar, R.; Berg, A.C. Matchnet: Unifying feature and metric learning for patch-based matching. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
23. Brown, M.; Szeliski, R.; Winder, S. Multi-image matching using multi-scale oriented patches. In Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–26 June 2005.
24. Yu, Y.F.; Xu, G.; Huang, K.K.; Zhu, H.; Chen, L.; Wang, H. Dual Calibration Mechanism based L2,p-Norm for Graph Matching. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *31*, 2343–2358. [CrossRef]
25. Yu, Y.; Xu, G.; Jiang, M.; Zhu, H.; Dai, D.; Yan, H. Joint Transformation Learning via the L2,1-Norm Metric for Robust Graph Matching. *IEEE Trans. Cybern.* **2019**, *31*, 2343–2358. [CrossRef]
26. Lucas, B.D.; Kanade, T. An iterative image registration technique with an application to stereo vision. In Proceedings of the International Joint Conference on Artificial Intelligence, Vancouver, BC, Canada, 24–28 August 1981.
27. Gehrig, D.; Rebecq, H.; Gallego, G.; Scaramuzza, D. EKLT: Asynchronous Photometric Feature Tracking Using Events and Frames. *Int. J. Comput. Vis.* **2019**, *128*, 601–618. [CrossRef]
28. Zach, C.; Gallup, D.; Frahm, J.-M. Fast gain-adaptive KLT tracking on the GPU. In Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Anchorage, AK, USA, 23–28 June 2008.
29. Hwangbo, M.; Kim, J.-S.; Kanade, T. Inertial-aided KLT feature tracking for a moving camera. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, St Louis, MS, USA, 11–15 October 2009.

30. Poling, B.; Lerman, G.; Szlam, A. Better feature tracking through subspace constraints. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014.
31. Zhao, L.; Li, X.; Xiao, J.; Wu, F.; Zhuang, Y. Metric learning driven multi-task structured output optimization for robust keypoint tracking. In Proceedings of the 29th {AAAI} Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; pp. 3864–3870.
32. Lin, W.-Y.; Liu, S.; Jiang, N.; Do, M.N.; Tan, P.; Lu, J. RepMatch: Robust Feature Matching and Pose for Reconstructing Modern Cities. In *Computer Vidion—ECCV 2016, Proceedings of the 14th European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016*; Springer: Berlin/Heidelberg, Germany, 2016.
33. Roth, L.; Kuhn, A.; Mayer, H. Wide-Baseline Image Matching with Projective View Synthesis and Calibrated Geometric Verification. *PFG–J. Photogramm. Remote Sens. Geoinf. Sci.* **2017**, *85*, 85–95. [CrossRef]
34. Lowry, S.; Andreasson, H. LOGOS: Local Geometric Support for High-Outlier Spatial Verification. In Proceedings of the International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018.
35. Zhao, C.; Cao, Z.; Li, C.; Li, X.; Yang, J. NM-Net: Mining reliable neighbors for robust feature correspondences. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.
36. Yi, K.M.; Trulls, E.; Ono, Y.; Lepetit, V.; Salzmann, M.; Fua, P. Learning to Find Good Correspondences. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
37. Ma, J.; Jiang, X.; Fan, A.; Jiang, J.; Yan, J. Image Matching from Handcrafted to Deep Features: A Survey. *Int. J. Comput. Vis.* **2021**, *129*, 23–79. [CrossRef]
38. Snavely, N.; Seitz, S.M.; Szeliski, R. Photo tourism: Exploring photo collections in 3D. In *ACM Transactions on Graphics (TOG)*; ACM: New York, NY, USA, 2006; pp. 835–846.
39. Wu, C. Towards linear-time incremental structure from motion. In Proceedings of the International Conference on 3D Vision-3DV, Seattle, WA, USA, 29 June–1 July 2013.
40. Sweeney, C.; Fragoso, V.; Höllerer, T.; Turk, M. Large Scale SfM with the Distributed Camera Model. In Proceedings of the 4th International Conference on 3D Vision (3DV), Stanford, CA, USA, 25–28 October 2016.
41. Zhu, S.; Zhang, R.; Zhou, L.; Shen, T.; Fang, T.; Tan, P.; Quan, L. Very Large-Scale Global SfM by Distributed Motion Averaging. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
42. Schönberger, J.L.; Frahm, J.-M. Structure-from-motion revisited. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
43. Heinly, J.; Schönberger, J.L.; Dunn, E.; Frahm, J.M. Reconstructing the world* in six days. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
44. Agarwal, S.; Snavely, N.; Simon, I.; Seitz, S.M.; Szeliski, R. Building rome in a day. In Proceedings of the 12th International Conference on Computer Vision, Kyoto, Japan, 27 September–4 October 2009.
45. Furukawa, Y.; Ponce, J. Accurate, Dense, and Robust Multi-View Stereopsis. In Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 18–23 June 2007.
46. Goesele, M.; Curless, B.; Seitz, S.M. Multi-View Stereo Revisited. In Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition, New York, NY, USA, 17–22 June 2006.
47. Galliani, S.; Lasinger, K.; Schindler, K. Massively Parallel Multiview Stereopsis by Surface Normal Diffusion. In Proceedings of the International Conference on Computer Vision, Las Condes, Chile, 11–18 December 2015.
48. Kuhn, A.; Hirschmüller, H.; Scharstein, D.; Mayer, H. A TV Prior for High-Quality Scalable Multi-View Stereo Reconstruction. *Int. J. Comput. Vis.* **2017**, *124*, 2–17. [CrossRef]
49. Yao, Y.; Luo, Z.; Li, S.; Fang, T.; Quan, L. Mvsnet: Depth inference for unstructured multi-view stereo. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
50. Yao, Y.; Luo, Z.; Li, S.; Shen, T.; Fang, T.; Quan, L. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–17 June 2019.
51. Kazhdan, M.; Bolitho, M.; Hoppe, H. Poisson surface reconstruction. In Proceedings of the Eurographics Symposium on Geometry Processing, Sardinia, Italy, 26–28 June 2006.
52. Zhou, K.; Gong, M.; Huang, X.; Guo, B. Data-Parallel Octrees for Surface Reconstruction. *IEEE Trans. Vis. Comput. Graph.* **2011**, *17*, 669–681. [CrossRef]
53. Waechter, M.; Moehrle, N.; Goesele, M. Let There Be Color! Large-Scale Texturing of 3D Reconstructions. In *Computer Vision—ECCV 2014, Proseedings of the 13th European Conference, Zurich, Switzerland, 6–12 September 2014*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 836–850.
54. Ummenhofer, B.; Brox, T. Global, Dense Multiscale Reconstruction for a Billion Points. *Int. J. Comput. Vis.* **2017**, *125*, 82–94. [CrossRef]
55. Kelly, T.; Femiani, J.; Wonka, P.; Mitra, N.J. BigSUR: Large-scale Structured Urban Reconstruction. *Acm Trans. Graph.* **2017**, *36*, 204. [CrossRef]
56. Ji, M.; Gall, J.; Zheng, H.; Liu, Y.; Fang, L. SurfaceNet: An End-to-End 3D Neural Network for Multiview Stereopsis. In Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
57. DeTone, D.; Malisiewicz, T.; Rabinovich, A. Superpoint: Self-supervised interest point detection and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.

58. Yang, X.; Cheng, K.-T. LDB: An ultra-fast feature for scalable augmented reality on mobile devices. In Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR) IEEE Conference, Atlanta, GA, USA, 5–8 November 2012.
59. Ziegler, A.; Christiansen, E.; Kriegman, D.; Belongie, S.J. Locally uniform comparison image descriptor. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012.
60. Levi, G.; Hassner, T. LATCH: Learned Arrangements of Three Patch Codes. *arXiv* **2015**, arXiv:1501.03719.
61. Zhang, G.; Liu, H.; Dong, Z.; Jia, J.; Wong, T.-T.; Bao, H. ENFT: Efficient Non-Consecutive Feature Tracking for Robust Structure-from-Motion. *arXiv* **2015**, arXiv:1510.08012.
62. Mishkin, D.; Matas, J.; Perdoch, M. MODS: Fast and robust method for two-view matching. *Comput. Vis. Image Underst.* **2015**, *141*, 81–93. [CrossRef]
63. Lin, W.Y.; Wang, F.; Cheng, M.M.; Yeung, S.K.; Torr, P.H.S.; Do, M.N.; Lu, J. CODE: Coherence Based Decision Boundaries for Feature Correspondence. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 34–47. [CrossRef] [PubMed]
64. Tian, Y.; Balntas, V.; Ng, T.; Barroso-Laguna, A.; Demiris, Y.; Mikolajczyk, K. D2D: Keypoint Extraction with Describe to Detect Approach. *arXiv* **2020**, arXiv:2005.13605.
65. Zitnick, C.L.; Ramnath, K. Edge foci interest points. In Proceedings of the International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011.
66. Seitz, S.M.; Curless, B.; Diebel, J.; Scharstein, D.; Szeliski, R. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition, New York, NY, USA, 17–22 June 2006.