

BINARY FEATURE MATCHING METHODS AND SYSTEMS

TECHNICAL FIELD

- 5 The present disclosure relates to feature matching and in particular to matching binary features in real time or near real time.

BACKGROUND

- 10 Feature matching plays an important role in many computational tasks. For example, feature descriptor matching is an essential step for an autonomous robot to localize itself during navigation. However, it is often difficult to achieve the matching in real-time due to the limited on-board computing resources. Localization is necessary for an autonomous robot to determine its position with respect to its environment, so that
- 15 it can perform safe and reliable navigation. In recent years, visual localization which relies on the image feed from cameras, has become increasingly popular as cameras are low cost and provide rich scene information. The steps in visual localization involves extracting feature descriptors from each image frame, and finding correspondences between these feature descriptors in consecutive image frames.
- 20 The process of finding correspondences is known as feature descriptor matching. These correspondences enable pose estimation, which tracks the camera (robot) movement in the environment over time. It is evident that feature descriptor matching needs to perform in real-time (at camera frame rate) with high matching precision. However, existing methods that offer high precision are complex and do not lend
- 25 themselves well for realizations on resource constrained computing devices on-board the robots.

- Previous works have leveraged on field programmable gate arrays (FPGAs) to accelerate feature descriptor matching. These works adopt different types of feature
- 30 descriptors for matching, i.e., binary descriptors e.g., binary robust independent elementary features (BRIEF), Oriented FAST and Rotated BRIEF (ORB), and non-binary descriptors e.g., scale-invariant feature transform (SIFT). Prior to matching, the descriptors are extracted through key-point detection and descriptor computation.

SIFT descriptors lead to more time consuming matching since it relies on the Euclidean distance calculation to identify correspondences. In addition, the large non-binary descriptors incur high memory requirement. In addition, the large non-binary descriptors incur high memory requirement. In order to reduce the complexity of the descriptors, recent SLAM (simultaneous localization and mapping) visual localization frameworks such as ORB-SLAM2 and ORB-SLAM3 adopt BRIEF binary descriptors.

All the above-mentioned hardware architectures employ the Linear Exhaustive Search (LES) method for descriptor matching. In the LES method, each query descriptor has to be compared with every reference descriptor, which incurs very time-consuming matching process.

SUMMARY

According to a first aspect of the present disclosure, a feature matching method of matching a query binary feature descriptor with a reference binary feature descriptor is provided. The method comprises: allocating reference binary feature descriptors to bins corresponding to leaf nodes of a balanced binary search tree based on bit values of selected bit positions of the reference binary feature descriptors; traversing the binary search tree using the query binary feature descriptor to reach a leaf node and identifying a reference binary feature descriptor allocated to a bin corresponding to the leaf node; and matching the query binary feature descriptor to the identified reference binary feature descriptor if a distance measure between the query binary feature descriptor and the identified reference binary feature descriptor is less than a threshold.

In an embodiment, the method further comprises storing the allocated reference binary feature descriptors for each respective bin.

In an embodiment, descriptor values corresponding to the selected bit positions are removed prior to storing the allocated reference binary feature descriptors for each respective bin.

In an embodiment, the balanced binary search tree has a pre-defined number of levels and a corresponding pre-defined number of leaf nodes and wherein any reference binary feature descriptors exceeding bin sizes of leaf nodes are discarded.

- 5 In an embodiment, the reference binary feature descriptors are allocated to the bins by selecting indices of the reference binary feature descriptors which have a probability of a binary value 1 closest to 0.5 at each level of the balanced binary search tree.
- 10 In an embodiment, the reference binary feature descriptors and query binary feature descriptor are image feature descriptors.

In an embodiment, the distance measure is thresholded by performing a ratio test.

- 15 In an embodiment, the ratio test comprises performing a multiplication by 0.5 which is performed by a one-bit shift operation.

In an embodiment, the distance measure is a Hamming distance measure.

- 20 According to a second aspect of the present disclosure, a feature matching system for matching a query binary feature descriptor with a reference binary feature descriptor is provided. The system comprises: a binary search tree module configured to allocate reference binary feature descriptors to bins corresponding to leaf nodes of a balanced binary search tree based on bit values of selected bit positions of the reference binary
- 25 feature descriptors; and to traverse the binary search tree using the query binary feature descriptor to reach a leaf node and identify a reference binary feature descriptor allocated to a bin corresponding to the leaf node; and matching module configured to match the query binary feature descriptor to the identified reference binary feature descriptor if a distance measure between the query binary feature
- 30 descriptor and the identified reference binary feature descriptor is less than a threshold.

In an embodiment, the system is provided as an integrated circuit.

In an embodiment, the binary search tree module is further configured to store the allocated reference binary feature descriptors for each respective bin.

5 In an embodiment, the binary search tree module is further configured to remove descriptor values corresponding to the selected bit positions prior to storing the allocated reference binary feature descriptors for each respective bin.

10 In an embodiment, the balanced binary search tree has a pre-defined number of levels and a corresponding pre-defined number of leaf nodes and the binary search tree module is further configured to discard any reference binary feature descriptors exceeding bin sizes in leaf nodes.

15 In an embodiment, the binary search tree module is further configured to allocate reference binary feature descriptors to the bins by selecting indices of the reference binary feature descriptors which have a probability of a binary value 1 closest to 0.5 at each level of the balanced binary search tree.

20 In an embodiment, the reference binary feature descriptors and query binary feature descriptor are image feature descriptors.

In an embodiment, the distance measure is thresholded by performing a ratio test.

25 In an embodiment, the ratio test comprises performing a multiplication by 0.5 which is performed by a one-bit shift operation.

In an embodiment, the distance measure is a Hamming distance measure.

BRIEF DESCRIPTION OF THE DRAWINGS

30 In the following, embodiments of the present invention will be described as non-limiting examples with reference to the accompanying drawings in which:

FIG.1 is a block diagram showing an implementation of a feature matching system according to an embodiment of the present invention;

FIG.2 is a flow chart showing a feature matching method according to an embodiment of the present invention;

5 FIG.3 shows a 3 level binary search tree used in embodiment of the present invention;

FIG.4 is a block diagram showing an overview of a binary search tree matching module with N-depth (BSTM-N module) according to an embodiment of the present invention;

10 FIG.5 is a block diagram showing the internal architecture of the *BinarySearchTree* module of the binary search tree matching module shown in FIG.4;

FIG.6 is a block diagram showing the internal architecture of the *TreeLevelNode* units of the *BinarySearchTree* module shown in FIG.5;

15

FIG.7A and FIG.7B show data paths in the *ReferenceDescStore* module of the binary search tree matching module shown in FIG.4;

FIG.8 is a block diagram showing the internal architecture of the *QueryDescStore* module of the binary search tree matching module shown in FIG.4;

20

FIG.9 is a block diagram showing the internal architecture of the *HammingDistanceCalc* module of the binary search tree matching module shown in FIG.4;

25

FIG.10 is a block diagram showing the internal architecture of the *MatchSelector* module of the binary search tree matching module shown in FIG.4; and

FIG.11A to FIG.11C are graphs showing comparisons of matching methods.

30

DETAILED DESCRIPTION

The present disclosure provides a binary search tree matcher for binary feature matching. Binary descriptors have gained widespread use in various applications due

to their notable advantages, such as minimal storage requirements and lower computational costs during correspondence matching. Binary descriptors like BRIEF, ORB, and BRISK are popular choices for their compactness and computational efficiency, making them particularly suitable for embedded and real-time systems. In
5 computer vision, several feature descriptor matching techniques are commonly employed, including the following.

Image-to-image feature matching which is used for tasks like pose estimation (via the Fundamental Matrix), panorama stitching, augmented reality, and 3D Reconstruction.

10

Image-to-map-points matching which is applied in scenarios such as pose estimation (using Iterative Closest Point (ICP)) and loop closure detection in visual simultaneous location and mapping (SLAM).

15 Map-points-to-map-points matching which is used in multi-map fusion scenarios in single robot and collaborative multi-robot applications.

Beyond traditional image feature descriptor matching between multiple images, the latter two cases highlight other situations where the proposed binary search tree
20 matcher can effectively establish correspondences using binary features. These cases typically involve large number of data points where rapid and accurate correspondence matching is essential. The proposed binary search tree matcher, with its ultra-low latency and improved precision, offers a promising alternative for such scenarios, providing significant benefits in both speed and precision which is
25 especially beneficial in resource-constrained environments such as drones and autonomous robots, where real-time processing is crucial.

Further, beyond feature matching approaches stated above, the proposed binary search tree matcher may find applications in the following.

30

Network Intrusion Detection: In network intrusion detection systems (NIDS), detecting unusual or malicious activity from vast amounts of network traffic is challenging due to the volume and complexity of data. Binary feature descriptors help reduce this complexity by transforming packet characteristics into compact binary vectors, which

are easier to manage and compare. By using binary search trees for matching these descriptors, the speed of detecting intrusions could be significantly improved. The BST structure allows for rapid matching processes, which are critical in real-time threat detection scenarios. The binary feature descriptors effectively capture key patterns and abnormalities, which can be efficiently matched against known attack signatures using the proposed, which will enhance both speed and detection accuracy.

Document Retrieval: Binary feature descriptors play a crucial role in document retrieval by transforming text features, such as sets of words or shingles (i.e., word sequences), into binary vectors through methods like MinHash. This transformation simplifies the process of comparison between documents. By utilizing binary search trees (BSTs), the system can efficiently organize and match these binary signatures, significantly accelerating the retrieval process. The proposed binary search tree matcher mechanism reduces the search space within the reference set, enhancing efficiency. This is especially important in large-scale databases where millions of documents need to be compared in real-time, ensuring swift identification of similar or near-duplicate documents.

Fingerprint Matching: Fingerprint matching is an essential component in biometric identification systems. Binary feature descriptors representing the whole signature is highly effective in fingerprint applications due to their ability to capture fingerprint minutiae in a compact form, enabling faster comparisons. Using binary search tree-based matching techniques helps in organizing these binary descriptors to expedite the matching process, which is crucial when matching an individual against large fingerprint databases.

FIG.1 is a block diagram showing an implementation of a feature matching system according to an embodiment of the present invention. The feature matching system 150 may be a field programmable gate array (FPGA) hardware based system. As shown in FIG.1, the inputs to the feature matching system 150 are features extracted from a reference image 110 and a query image 120. A feature extractor 130 extracts reference image feature descriptors 112 from the reference image 110 and query image feature descriptors 122 from the query image 120. The feature extractor 130 may implement a key point detection method which extracts points of interest. The

reference image feature descriptors 112 and query image feature descriptors 122 are binary feature descriptors.

5 The feature matching system 150 comprises a binary search tree module 152, a matching module 154 and a memory 156. As will be described in more detail below, the binary search tree module 152 is configured to allocate reference image binary feature descriptors to leaf nodes a balanced binary search tree and traverses the balanced binary search tree using query image binary feature descriptors. The matching module 154 is configured to determine a distance measure between an
10 identified reference image binary feature descriptor and a query image binary feature descriptor and to determine a match if the distance measure is less than a threshold.

The memory 156 stores a reference descriptor store 162 which stores reference image binary feature descriptors, a query descriptor store 164 which stores query image
15 binary feature descriptors and a binary search tree store 166 which stores index data allocating the reference image binary feature descriptors to a balanced binary search tree. The binary search tree store 166 may be stored inside configuration registers that can be configured during runtime.

20 As shown in FIG.1, the output of the feature matching system 150 is a set of correspondences 170 between reference image feature descriptors 112 and query image feature descriptors 122.

As described above, the present disclosure relates to an integrated circuit architecture
25 (e.g., may be implemented on a field programmable gate array (FPGA) or an application-specific integrated circuit (ASIC)) for binary feature descriptor matching in real-time wherein salient points of a query image and a reference image are matched accordingly based on feature descriptors extracted from the images.

30 The architecture, having a matching module, may comprise the following features:

1. A balanced N-Level Binary Search tree (BST) module that partitions the feature descriptors into 2^N bins based on selected bit indices of descriptors, wherein any excess descriptors beyond the capacity are discarded;

2. The module incorporates a streaming architecture where the binary tree construction and matching process utilize internal on-chip memory;
- 5 3. A ratio-test based match selection mechanism to improve matching precision in binary feature descriptor matching.

The architecture adopts a modular and resource-efficient pipelined design that scales easily to accommodate different bin sizes. FPGA implementation of the architecture
10 shows that it is significantly faster than the best-performing FPGA-based feature matcher in the literature. The architecture is approximately 10X faster than the commonly-used Linear Exhaustive Search (LES) method on FPGA.

FIG.2 is a flow chart showing a feature matching method according to an embodiment
15 of the present invention. The method 200 shown in FIG.2 may be carried out by the feature matching system 150 shown in FIG.1.

In step 202, reference binary feature descriptors are allocated to bins corresponding to leaf nodes of a balanced binary search tree. The allocate is made based on bit
20 values of selected bit positions of the reference binary feature descriptors.

In step 204, the balanced binary search tree is traversed using a query binary feature descriptor to reach a leaf node of the balanced binary search tree.

25 In step 206, a distance measure between the reference binary feature descriptor corresponding to the leaf node of the balanced binary search tree and the query binary feature descriptor. If the distance measure is less than a threshold, then the query binary feature descriptor is matched to the reference binary feature descriptor corresponding to the leaf node and an indication of the match is generated.

30

The feature matching systems and methods of the present disclosure have the following advantages. The binary search tree matching architecture may be implemented as a streaming architecture where the input data is processed as it arrives without the need for external storage. There is no need for off-chip memory

access during construction of the binary search tree or during the matching process. The streaming architecture leads to higher speed and lower power consumption as it does not require off-chip memory access during the binary tree construction and matching process.

5

The binary search tree matching architecture uses a predefined balanced binary search tree (i.e., fixed number of bins), and therefore there is no need to adjust the tree structure at runtime. Users can determine a fixed balanced binary search tree that is suitable for the given application beforehand in order to avoid in-deterministic tree size during runtime that requires the use of external memories.

10

The proposed FPGA-based binary search tree matching (BSTM) overcomes the high complexity matching requirement in all the previously reported hardware feature descriptor matchers, which use the LES method. By relying on a hardware-efficient binary search tree (BST), the need to exhaustively match each descriptor from the query image with all the descriptors in the reference image is avoided.

15

In order to boost the matching precision, the ratio-test outlier rejection mechanism is integrated into the processing. It is shown that the ratio-test outlier rejection of the proposed binary search tree matcher can surpass the matching precision of the distance threshold-based calculation that is commonly used in binary feature descriptor matching.

20

The design may be implemented on the SoC based Xilinx MPSoC Ultrascale+ platform and evaluated with widely-used datasets. The experiments demonstrate the ultra-low latency of the proposed binary search tree matcher design compared to existing hardware-based feature descriptor matchers, with only moderate resource utilization.

25

Embodiments of the present invention may process binary key point descriptors to implement binary descriptor matching. Given a reference image I and a query image J , descriptor matching attempts to find pixels in I and pixels in J that correspond to the same physical point in 3D world. The output is a set of correspondence pairs, which can be used in projective geometry to determine the camera pose between the two images.

30

Prior to matching, the descriptors are extracted through key-point detection and descriptor computation for each image. Key-point detection determines the interest points (e.g., corners) from an image frame. Commonly-used key point detection methods include the Harris corner detector and the features from accelerated segment test (FAST) corner detector. In the descriptor computation step, the descriptor is extracted from an image patch that is centered at the detected key-points. Descriptors describe the local appearance of key-points, and they are represented using a vector of binary values or floating point values. The commonly-used binary descriptors are binary robust independent elementary features (BRIEF), binary Robust invariant scalable key-points (BRISK), and Oriented FAST and Rotated BRIEF (ORB), while speeded up robust features (SURF) and scale-invariant feature transform (SIFT) are commonly-used non-binary descriptors. These descriptors are designed with the underlying idea that they will represent locally similar appearance regions across different images. A similarity metric is used to match feature descriptors between two images. In particular, the Euclidean distance is often used for matching non-binary descriptors, while the Hamming distance is employed for matching the binary descriptors. The latter refer to the number of dissimilar bit pairs among two descriptors, from I and J respectively.

Mathematically, feature matching finds the most similar key-point $p_i \in I$ with key-point $p_j \in J$ as follows.

$$p'_i = \underset{p_i}{\operatorname{argmin}} \left\{ \operatorname{dist} \left(\operatorname{desc}(p_j), \operatorname{desc}(p_i) \right) \right\}$$

Where $\operatorname{desc}(p_j), \operatorname{desc}(p_i)$ represents descriptors of key-point p_j and p_i , and $\operatorname{dist}(\dots)$ represents the similarity metric between descriptors. p'_i is the key-point in I that matches $p_j \in J$. Distance threshold mechanisms are often employed to eliminate false matches, i.e., distances lesser than a certain threshold τ will be discarded. Apart from such direct distance thresholding mechanisms, other approaches use threshold based on the ratio between the two least distant pairs.

Linear Exhaustive Search (LES) is the most commonly-used feature matching approach. It compares the distance between p_j with every $p_i \in I$ and outputs the closest match $p'_i \in I$. LES provides high matching accuracy, but has a complexity that is proportionate to the number of descriptors in J . The number of descriptors in an image typically ranges from 100 to 10,000, hence the computational cost of LES becomes prohibitive especially when it is implemented on resource constrained computing platforms that are often found on robots.

In order to overcome the high matching latency of LES, a hardware-efficient binary search tree (BST) for binary feature descriptor matching is proposed. The proposed method evaluates distances for only a selected number of reference key-points from I to find the corresponding key-point for query $p_j \in J$. The ratio-test outlier rejection mechanism is also incorporated, which has been used previously for only non-binary descriptor matching. The approach consists of three parts: Binary Search Tree, Selection of Bit Indices, and Match Selection.

The proposed method partitions the descriptor space into different bins based on the bit values of selected bit indices of descriptors. The proposed approach relies on a balanced BST with N levels, wherein the descriptor space is partitioned into 2^N bins.

FIG.3 shows a 3 level binary search tree used in embodiment of the present invention. In the example shown in FIG.3, the 3 level BST is based on bit indices $p, q, r = 2, 0, 4$, and a given query descriptor $desc_{in} = 101011$, which maps to $bin_id = 010$. The $desc_{in}$ bits at indices 2, 0, 4 are discarded at the respective level. Descriptor index 0 starts from right most bit position.

The BST shown in FIG.3 has $N = 3$ and 8 bins, where each bin stores a set of partial descriptors from the reference image. As shown in FIG.3, each bin is assigned a bin_id . The allocation of the descriptors to the bins is based on bit values of selected bit indices (positions) of the descriptors. In the example, $p, q, r \in \{K - 1, \dots, 2, 1, 0\} = \{2, 0, 4\}$, where K is the width of the descriptor, are the selected bit indices. The descriptors stored in each bin have common bit values in the p, q, r indices. For example, all the descriptors in $bin_id = 101$, have values of 1, 0 and 1 in their

respective p, q and r indices. In order to reduce the memory requirement for storing the descriptors in the bins, descriptor bits that are associated with the selected bit indices are removed (e.g., p, q and r in FIG.3). Given a query descriptor, the bit values at indices p, q and r are used to identify the corresponding bin that stores the reference descriptors for matching. The bits at indices p, q and r of the query descriptor are also discarded to facilitate the distance calculation with the reference descriptors in the corresponding bin. It is evident that compared to the LES method, the number of comparisons for each query descriptor is bounded by the number of reference descriptors from the corresponding bin.

10

Careful selection of the bit indices is necessary to ensure that the descriptors are distributed uniformly among the bins. This is important as the memory resources allocated to each bin is bounded by a fix value, i.e., MAX_DEPTH. If the number of reference descriptors that are associated with a certain bin exceeds MAX_DEPTH, the excess descriptors will be discarded leading to missed matches. In order to determine suitable bit indices, the probability of "1"s in each bit index $k \in \{(K - 1), \dots, 2, 1, 0\}$ of the set of reference descriptors is calculated. Then $|0.5 - p_{ri}|$ is calculated for each index i , where p_{ri} is the probability of "1"s in reference descriptor index i . Finally, the N indices $\{i_0, i_1, i_2, \dots, i_{N-1}\}$ are selected with the smallest values of $|0.5 - p_{ri}|$. This method enables the selection of indices that lead to descriptors with evenly distributed "1"s and "0"s in the bins. It is noteworthy that the above-mentioned process can be computed on-the-fly without incurring high computational resources, by updating a set of counters associated with each bit indices as a new descriptor is computed. Moreover, since the image content variations of each consecutive image frames are very small, the selected bit indices of the descriptors from the previous frame can be used to allocate the descriptors in the current frame to the appropriate bins.

In order to reduce false matches, existing hardware implementations for binary feature descriptor matching rely on a distance thresholding mechanism. On the other hand, hardware implementations of non-binary descriptor matching have used a ratio-test-based mechanism for match selection.

30

In the present disclosure "*thresholded*" refers to the result of applying a thresholding operation to data, typically to an image or signal. A thresholding operation involves setting a specific threshold value and then converting or modifying the data such that all values above or below the threshold are altered in a consistent manner. For example, in the context of image processing, thresholding may involve converting a grayscale image into a binary image, where all pixel values above a certain threshold are assigned one value (e.g., white) and all values below the threshold are assigned another value (e.g., black). The term "*thresholded*" therefore describes data that has been processed in this way, where the values or features of the data have been altered or segmented based on a predetermined threshold.

In the proposed design, the ratio-test based match selection mechanism based on Hamming distance for binary feature descriptor matching is used. Experiments show that this leads to more precise matches compared to the distance threshold approach. The underlying idea of the ratio test is that there should be sufficient discrimination between distances to consider two descriptors as a matching pair. The following ratio-test criteria should be met for two descriptors to be considered a match.

$$distance_{1st_min} < t_d \times distance_{2nd_min}$$

Where $distance_{1st_min}$ and $distance_{2nd_min}$ are the minimum distances of the two descriptor pairs respectively. t_d is the threshold value. The SIFT-based feature matching employs a ratio-test with $t_d = 0.6$. In the design, a ratio-test is used with $t_d = 0.5$, which replaces the multiplication with a one-bit-shift operation.

In this section, the proposed hardware architecture is described, which is denoted as BSTM-N for an N depth implementation of BST. The hardware is implemented for 256-bit ORB binary descriptor matching, which is used in well-known visual localization frameworks.

The top-level of the design consists of the AXI4LITE interface. The S AXIS and one M AXIS interface are connected to AXI-DMA to stream in query and reference descriptors, and stream out indices of matched descriptors. The S AXIS interface, which feeds data to the top hardware module, has a 512-bit data bus to accommodate

the 256-bit descriptor and 16-bit descriptor index. Unused bits are padded with 0s. The process starts by configuring config registers in the Controller module with selected bit indices calculated using the approach explained below. Then, the reference set of descriptors is fed into the BSTM module as an input stream. Next, query descriptors
5 are fed to the module using the same interface. The BSTM module outputs pairs of computed matched indices of query and reference descriptors if there are any. These outputs are transferred using the M AXIS interface of the BSTM module.

FIG.4 is a block diagram showing an overview of a binary search tree matching module
10 with N-depth (BSTM-N module) according to an embodiment of the present invention.

The BSTM-N hardware module 400 comprises of seven sub-modules as depicted in FIG.4, i.e., *BinarySearchTree* module 410, *Switch* module 420, *QueryDescStore* module 430, *ReferenceDescStore* module 440, *HammingDistanceCalc* module 450,
15 *MatchSelector* module 460, and *Controller* module 470. The Controller module 470 executes the functionality of storing selected indices of descriptors in configuration registers. *BinarySearchTree* module 410 is responsible for generating *bin_id* for an incoming descriptor and removing redundant bits. *Switch* module 420 switches between the *QueryDescStore* module 430 and the *ReferenceDescStore* module 440
20 depending upon a query or a reference, and transfers inputs to the relevant module. The *ReferenceDescStore* module 440 stores reference descriptors and their indices to compare with query descriptors. This module consists of a block random access memory (BRAM) that stores reference descriptors and their indices in the corresponding locations based on the incoming *bin_id*. The *QueryDescStore* module
25 430 is a first in first out (FIFO) module that stores query descriptors with indices to facilitate the pipeline execution of the design. The *HammingDistanceCalc* module 450 calculates the Hamming distance for binary descriptor matching. Finally, the *MatchSelector* module 460 selects correct matches using the ratio-test-based approach described in more detail below.

30

FIG.5 is a block diagram showing the internal architecture of the *BinarySearchTree* module of the binary search tree matching module shown in FIG.4.

The *BinarySearchTree* module 410 executes the task of *bin_id* generation and redundant information removal from the descriptor. The *bin_id* generated from this module determines the bin of the descriptor that will be required in subsequent modules to accelerate the matching process. The input to the *BinarySearchTree* module 410 is the same as the input to the top-level BSTM module 400. The output of the *BinarySearchTree* module 410 is *reshape_desc*, which is the descriptor generated after removing redundant bits from the input descriptor, and *bin_id* of the input descriptor. The *BinarySearchTree* module 410 comprises N *TreeLevelNode* units 412 which are labelled *TN0*, *TN1*, *TN2*... *TN N-1*.

10

FIG.6 is a block diagram showing the internal architecture of the *TreeLevelNode* units of the *BinarySearchTree* module shown in FIG.5.

Each *TreeLevelNode* unit 412 represents one level of BST. The *TreeLevelNode* unit 412 removes one input descriptor bit placed at given bit index by the *Controller* module 470 to generate output *reshape_desc* (output descriptor of a given *TreeLevelNode* unit 412) and *bin_id* (output *bin_id* of *TreeLevelNode* unit 412).

Depending on the type of input descriptor, the BSTM module 400 switches between the two descriptor-storing hardware modules. Specifically, query descriptors are stored in the *QueryDescStore* module 430, while reference descriptors are stored in the *ReferenceDescStore* module 440. Since reference descriptors and query descriptors are fed into the system as separate chunks of data streams, this switching process happens at the end of each chunk with the last signal reaching the *Switch* module 420.

FIG.7A and FIG.7B show data paths in the *ReferenceDescStore* module of the binary search tree matching module shown in FIG.4. FIG.7A shows the *ReferenceDescStore* module in a DATA_IN_STATE in which descriptors are received and stored and FIG.7B shows a DATA_OUT_STATE in which descriptors are read.

The *ReferenceDescStore* module 440 comprises of a BRAM 442 that has 2^N partitions. In this implementation, the depth of the BRAM 442 as is configured as 1024, since recent FPGA-based localization approaches have used around 700 key points per

image. Due to this fixed depth, each BRAM partition will have a maximum depth (MAX_DEPTH) of $1024/2^N$.

During the storing of descriptors, if the number of descriptors for a bin reaches the limit of MAX_DEPTH, the *ReferenceDescStore* module 440 discards further incoming descriptors to that bin. It is noteworthy that the number of descriptors that are discarded is minimized due to the bit selection mechanism.

FIG.8 is a block diagram showing the internal architecture of the *QueryDescStore* module of the binary search tree matching module shown in FIG.4.

The *QueryDescStore* stores 430 query descriptors during the descriptor-matching process. This module comprises of a FIFO memory 432 with a depth of 32, and enables the BSTM module to utilize pipelined architecture in subsequent modules to improve latency.

FIG.9 is a block diagram showing the internal architecture of the *HammingDistanceCalc* module of the binary search tree matching module shown in FIG.4.

The Hamming distance calculation is performed by the *HammingDistanceCalc* module 450. Inputs to the *HammingDistanceCalc* module 450 are query descriptors from the *QueryDescStore* module 430 and reference descriptors from *ReferenceDescStore* module 440, along with their indices. For these two inputs, the Hamming distance can be calculated in parallel using an array of XOR gates 452 followed by an adder tree 454. Since this sub-module receives reshaped descriptors from both query and reference, before feeding into the Adder tree 454, N number of bits are padded to both descriptors.

Furthermore, during the calculation of Hamming distance, indices that come as inputs along with descriptors are buffered, and *HammingDistanceCalc* module 450 outputs the relevant calculated distance.

FIG.10 is a block diagram showing the internal architecture of the *MatchSelector* module of the binary search tree matching module shown in FIG.4.

The *MatchSelector* module 460 is used to remove irrelevant matches. The ratio-test mechanism is employed for match selection, which relies on a threshold of 0.5 as follows.

$$distance_{1st_min} < (distance_{2nd_min} >> 1); t_d = 0.5$$

Inputs to the *MatchSelector* module 460 are the distances between two descriptors and indices of both query and reference descriptors. If there is any match, it outputs two indices of the matched descriptor pair. The module is pipelined and designed to accept inputs in each clock cycle. As shown in FIG.10, the minimum two distance values are stored in a min_1 register 462 and a min_2 register 464, and relevant indices of descriptors are stored in associated registers. When the last in signal is asserted, this module outputs matched index pair for query and reference, if there is a match.

An evaluation of the proposed binary search tree feature matching system will now be described. The proposed hardware was designed using System Verilog, and synthesised and implemented with Vivado 2021.2 Design Suite. The design was implemented and validated on the Kria KV260 AI Starter Kit (with K26 SOM), which is equipped with a XCK26-SFVC784-2LV-C Xilinx Zynq Ultrascale+ Mp-SoC device. This device has a quad-core Arm A53 processing system (PS) with 4GB RAM and runs at 1.33 GHz clock frequency. A PC with Intel Xeon CPU running at 3.80 GHz clock frequency was also used to benchmark of this hardware implementation.

For evaluations, the leuven, bikes, and boat datasets from the following publication are used:

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A comparison of affine region detectors," International journal of computer vision, vol. 65, pp. 43–72, 2005.

These datasets incorporate affine covariant features. leuven exhibits significant changes in lighting, bikes contains image blur, and boat is subjected to significant rotation and scale changes. Each dataset consist of 6 images with different resolutions, i.e., leuven with 921x614 images, bikes with 1000x700 images, and boat with 800x640
 5 images. These challenging datasets enable us to evaluate the performance and precision of this approach compared to existing implementations.

The precision is defined as follows.

$$10 \quad \text{precision} = \frac{\# \text{ correct matches}}{\# \text{ total matches}}$$

Where $\# \text{ total matches} = \# \text{ correct matches} + \# \text{ false matches}$.

BSTM-0 is chosen as the baseline which is equivalent to the LES method. BSTM-0
 15 utilises a BST with 0 levels and therefore, there is no *BinarySearchTree* module, and only one bin in the BRAM of *ReferenceDescStore*. For comparisons, BSTM-0 (LES method) and the proposed design (BSTM-4) are implemented on the same K26 SOM platform with the same clock frequency, and evaluated for all three datasets for the same number of descriptors.

20

The results obtained are shown in Table I below.

TABLE I: Latency comparison of BSTM proposed methodology and LES methodology implemented in PL of same K26 SOM platform

		BSTM-4	LES
		(K26 SOM - FPGA)	(K26 SOM - FPGA)
Frequency		300 MHz	300MHz
Features		Image #1: 600 features / Image #2: 600 features	
Latency (ms)	Bikes	0.216	1.718
	Leuven	0.210	1.884
	Boat	0.204	1.886
	Average	0.210	1.829
Multiplying Factor		x1	x8.71
Features		Image #1: 700 features / Image #2: 700 features	
Bikes		0.239	2.262
Latency Leuven		0.242	2.530
(ms) Boat		0.242	2.544
Average		0.241	2.445
Multiplying Factor		x1	x10.15

- 5 It can be observed that the proposed approach executes 10 times faster than the LES method. Furthermore, the approach is evaluated with existing descriptor-matching techniques for runtime performance, and the results are presented in Table II.

Comparisons are made with the following techniques:

10

Chien: C.-H. Chien, C.-J. Chien, and C.-C. Hsu, "Hardware-software co-design of an image feature extraction and matching algorithm," in 2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS). IEEE, 2019, pp. 37–41.

15

Fularz: M. Fularz, M. Kraft, A. Schmidt, and A. Kasiński, "A high-performance fpga-based image feature detector and matcher based on the fast and brief algorithms," International Journal of Advanced Robotic Systems, vol. 12, no. 10, p. 141, 2015.

Daoud: L. Daoud, M. K. Latif, H. Jacinto, and N. Rafla, "A fully pipelined fpga accelerator for scale invariant feature transform keypoint descriptor matching," Microprocessors and Microsystems, vol. 72, p. 102919, 2020.

5 **TABLE II: Latency comparison with existing descriptor matching hardware architectures**

Approach	Platform	Frequency	# Features	Latency (ms)				Multiplying Factor
				Bikes	Leuven	Boat	Average	
LES	Intel Xeon PC	3.80 GHz	615	2.728	1.657	1.996	2.127	x10.38
LES (Arm A53 PS)	ARM A53 Quad-Core	1.33 GHz	615	13.367	15.301	14.47	14.379	x70.14
Chien	FPGA	50 MHz	600	-	-	-	7.37	x35.26
Fularz	FPGA	100 MHz	512	-	-	-	4.37	x2.09
Daoud	FPGA	150 MHz	579	-	-	-	6.08	x29.09
BSTM-40	FPGA	300 MHz	615	0.197	0.212	0.206	0.205	x1

All the existing approaches employ LES for their descriptor-matching technique. Results of the LES descriptor matching running on the PC and PS of k26 SOM are also included. As the results show, the approach of the present disclosure is the fastest among all the implementations listed in Table II. Fularz15, which is the second best-performing approach, uses 32 matching cores to speed up the process, which incurs a large amount of resources to achieve microseconds latency.

15 The architecture is implemented for different numbers of BST levels (N), namely for 2 to 6. The resource utilization of the FPGA architecture is evaluated with existing architectures. These results are shown in Table III.

Comparisons are made with the following architectures:

20

- Wang: J. Wang, S. Zhong, L. Yan, and Z. Cao, "An embedded system-onchip architecture for real-time visual detection and matching," IEEE transactions on Circuits and Systems for Video Technology, vol. 24, no. 3, pp. 525–538, 2013.
- 5 Fularz: M. Fularz, M. Kraft, A. Schmidt, and A. Kasiński, "A high-performance fpga-based image feature detector and matcher based on the fast and brief algorithms," International Journal of Advanced Robotic Systems, vol. 12, no. 10, p. 141, 2015.
- Chien: C.-H. Chien, C.-J. Chien, and C.-C. Hsu, "Hardware-software co-design of an
10 image feature extraction and matching algorithm," in 2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS). IEEE, 2019, pp. 37–41.
- Daoud: L. Daoud, M. K. Latif, H. Jacinto, and N. Rafla, "A fully pipelined fpga accelerator for scale invariant feature transform keypoint descriptor matching,"
15 Microprocessors and Microsystems, vol. 72, p. 102919, 2020.
- Belmeshd: N. M. Belmessaoud, Y. Bentoutou, and M. C. El-Mezouar, "Fpga implementation of feature detection and matching using orb," Microprocessors and Microsystems, vol. 94, p. 104666, 2022.

20

TABLE III: Resource comparison with existing descriptor matching hardware architectures

Approach	Platform	LUT	FF	BRAM (kb)
Wang	XUPV5-LX110T	1635	526	2448
Fularz	XC7Z020-CLG484-1	17205	33969	306
Chien	Altera DE2i-150	10535	132.41 kb	
Daoud	Zed board	3710	6365	540
Belmeshd	XC7Z020-CLG484-1	1012	709	288
(BSTM-2)	XCK26-SFVC784-2LV-C	3114	2948	270
(BSTM-3)	XCK26-SFVC784-2LV-C	3933	3306	270
(BSTM-4)	XCK26-SFVC784-2LV-C	4647	3697	270
(BSTM-5)	XCK26-SFVC784-2LV-C	6708	4188	270
(BSTM-6)	XCK26-SFVC784-2LV-C	7741	4693	270

From the results, it can be observed that the proposed design requires only moderate resource utilization, which increases with the number of BST levels. Experiments also reveal that as the number of levels in BST increases, the latency of the proposed design reduces due to reduced number of descriptor comparison in each bin. BSTM-6 has the maximum number of levels and achieves 0.1 ms range latency for the same set of experiments carried out for comparison in Table II. It is noteworthy that BSTM-6 has lesser LUT, FF and BRAM utilization compared to Fularz, which is implemented with 32 cores.

Lastly, the precision of matches of the proposed implementation is analysed. The comparison results are shown in FIG.11A to FIG.11C. In each graph the three bars correspond to a boat dataset, a Leuven dataset and a bikes dataset in order.

FIG.11A is a graph showing a comparison of matching methods with homography with up to 5 pixels difference. Here, each method's precision is measured by determining the percentage of matched feature points that fall within a 5-pixel distance from the corresponding points projected by the homography transformation..

FIG.11B is a graph showing a comparison of matching methods with homography with up to 3 pixels difference. By reducing the allowable pixel deviation, this comparison emphasizes accuracy, reflecting each method's ability to achieve high spatial alignment precision. The 3-pixel tolerance requires closer correspondence between matched points, thus highlighting methods that are effective under higher precision demands.

FIG.11C is a graph showing a comparison of matching methods with RANSAC based Geometric Verification. This iteratively fits a geometric model to the feature matches and excludes outliers that do not conform in order to calculate the percentage of inliers. Unlike homography-based comparisons, this verification method does not just rely on pixel tolerance but also focuses on the geometric consistency of matches. The resulting precision metric represents the percentage of inliers or matches that conform to the underlying geometric transformation model, providing an assessment on accuracy of matches.

Here the ratio-test-based methodology is evaluated first by using BSTM-0 implementation and compared with the matching precision of Belmeshd using the same experiment approach indicated in their paper for the same datasets. As shown in FIG.11, ratio-test results in 100% precision for output matches for each dataset, while Belmeshd reports a lower precision. Therefore, it is evident that the ratio-test is a better choice for match selection compared to distance threshold approaches. Furthermore, FIG.11 included precision results evaluated under different metrics to select good matches and compared with the LES hardware implementation. In particular, FIG.11A is evaluated with homography matrix provided with datasets up to a difference of 5 pixels, FIG.11B is evaluated with homography matrix with up to a difference of 3 pixels, and finally, in FIG.11C with RANSAC based geometry verification approach with a threshold unit of 1. It can be seen that the proposed method produces comparable or better precision compared to the LES method.

This disclosure introduces a FPGA design for binary descriptor matching using a hardware-efficient BST. The design overcomes the high matching latency requirements of existing approaches by reducing the number of evaluations between the query and reference descriptor. The ratio test outlier rejection mechanism is integrated. It is shown that the proposed design can surpass the matching precision of the distance threshold based calculation that is commonly used in binary feature descriptor matching. The FPGA evaluations demonstrate ultralow matching latency and moderate resource utilization, compared to existing hardware implementations for descriptor matching, while maintaining a comparable precision on well-known datasets.

Whilst the foregoing description has described exemplary embodiments, it will be understood by those skilled in the art that many variations of the embodiments can be made within the scope and spirit of the present invention.

CLAIMS

1. A feature matching method of matching a query binary feature descriptor with a reference binary feature descriptor, the method comprising:
 - allocating reference binary feature descriptors to bins corresponding to leaf nodes of a balanced binary search tree based on bit values of selected bit positions of the reference binary feature descriptors;
 - traversing the binary search tree using the query binary feature descriptor to reach a leaf node and identifying a reference binary feature descriptor allocated to a bin corresponding to the leaf node; and
 - matching the query binary feature descriptor to the identified reference binary feature descriptor if a distance measure between the query binary feature descriptor and the identified reference binary feature descriptor is less than a threshold.
2. The method according to claim 1, further comprising storing the allocated reference binary feature descriptors for each respective bin.
3. The method according to claim 2, wherein descriptor values corresponding to the selected bit positions are removed prior to storing the allocated reference binary feature descriptors for each respective bin.
4. The method according to any preceding claim, wherein the balanced binary search tree has a pre-defined number of levels and a corresponding pre-defined number of leaf nodes and wherein any reference binary feature descriptors exceeding bin sizes of leaf nodes are discarded.
5. The method according to any preceding claim, wherein the reference binary feature descriptors are allocated to the bins by selecting indices of the reference binary feature descriptors which have a probability of a binary value 1 closest to 0.5 at each level of the balanced binary search tree.
6. The method according to any preceding claim, wherein the reference binary feature descriptors and query binary feature descriptor are image feature descriptors.

7. The method according to any preceding claim, wherein the distance measure is thresholded by performing a ratio test.

8. The method according to claim 7, wherein the ratio test comprises performing a multiplication by 0.5 which is performed by a one-bit shift operation.

9. The method according to any preceding claim, wherein the distance measure is a Hamming distance measure.

10. A feature matching system for matching a query binary feature descriptor with a reference binary feature descriptor, the system comprising:

a binary search tree module configured to allocate reference binary feature descriptors to bins corresponding to leaf nodes of a balanced binary search tree based on bit values of selected bit positions of the reference binary feature descriptors; and to traverse the binary search tree using the query binary feature descriptor to reach a leaf node and identify a reference binary feature descriptor allocated to a bin corresponding to the leaf node; and

matching module configured to match the query binary feature descriptor to the identified reference binary feature descriptor if a distance measure between the query binary feature descriptor and the identified reference binary feature descriptor is less than a threshold.

11. The feature matching system according to claim 10 provided as an integrated circuit.

12. The feature matching system according to claim 10 or 11, wherein the binary search tree module is further configured to store the allocated reference binary feature descriptors for each respective bin.

13. The feature matching system according to claim 12, wherein the binary search tree module is further configured to remove descriptor values corresponding to the selected bit positions prior to storing the allocated reference binary feature descriptors for each respective bin.

14. The feature matching system according to any one of claims 10 to 13, wherein the balanced binary search tree has a pre-defined number of levels and a corresponding pre-defined number of leaf nodes and the binary search tree module is further configured to discard any reference binary feature descriptors exceeding bin sizes of leaf nodes.

15. The feature matching system according to any one of claims 10 to 14, wherein the binary search tree module is further configured to allocate reference binary feature descriptors to the bins by selecting indices of the reference binary feature descriptors which have a probability of a binary value 1 closest to 0.5 at each level of the balanced binary search tree.

16. The feature matching system according to any one of claims 10 to 15, wherein the reference binary feature descriptors and query binary feature descriptor are image feature descriptors.

17. The feature matching system according to any one of claims 10 to 16, wherein the distance measure is thresholded by performing a ratio test.

18. The feature matching system according to claim 17, wherein the ratio test comprises performing a multiplication by 0.5 which is performed by a one-bit shift operation.

19. The feature matching system according to any one of claims 10 to 18, wherein the distance measure is a Hamming distance measure.