End-to-End Multi-Level Encoding Methods of Visual Data Compression for Robust Monocular Visual ORB-SLAM

Omar M. Salih[1,2], József Vásárhelyi[2]

[1] Institute of Automation and Info-communicatio, University of Miskolc, Egyetemváros, 3515 Miskolc, Hungary

[2] Department of Computer Engineering, Northern Technical University, Al-Minassa Street, 41003 Mosul, Iraq

{omar.salih, jozsef.vasarhelyi}@uni-miskolc.hu

Abstract: Simultaneous localization and mapping (SLAM) has been highly studied in the last decade. It allows the estimation of the camera pose of a mobile device and the creation of a map of the surrounding environment concurrently. Recently, Visual SLAM (VSLAM) has become the most widely used state-of-the-art technique to implement SLAM tasks due to its reduced cost, lower size, and affordability. However, the intensive computation of VSLAM systems does not fit in a wide range of limited resources and energy mobile devices. A possible solution is to split its functionality between mobile devices and the edge cloud. This solution showed the necessity for efficient visual data compression methods to be integrated within VSLAM systems. This work proposes a multi-level encoding method for visual data frame compression integrated within the monocular Oriented FAST and Rotated BRIEF-SLAM (ORB-SLAM) system. The performance results of the proposed system are compared to corresponding ORB-SLAM systems adopting the most popular classical still image compression standards; the Joint Photographic Experts Group (JPEG) and the advanced version, the JPEG 2000, in terms of reconstruction quality, robot's trajectory estimation, and computational complexity.

Keywords: ORB-SLAM; data compression; JPEG; pose estimation; visual perception; localization and mapping, cloud computing

1 Introduction

In recent years, SLAM (Simultaneous Localization and Mapping) has earned great attention in research and industry due to the expansion of robotics applications. SLAM aims to create a map of an anonymous environment while concurrently determining the mobile device's trajectory and position [1]. It can also helpreduce

estimation errors when the robot is in a previously mapped area. The mobile device can be an indoor robot [2], an Autonomous Vehicle (AV) [3], an Unmanned Autonomous Vehicle (UAV) [4], or an Unmanned Ground Vehicle (UGV) [5]. The SLAM algorithm performs map construction and robot localization based on data obtained from different sensors, such as Light Detection and Ranging (LiDAR) [6], radio signals [7], or visual sensors [8]. The camera-based SLAM system, commonly known as Visual SLAM (VSLAM), is interesting nowadays due to its low cost, rich information gathering, more straightforward object detection and tracking, and compact size. VSLAM obtains environmental information from a single camera in monocular VSLAM [9], multiple cameras (stereo VSLAM [10]), or Red, Green, and Blue with Depth (RGB-D) [11] cameras for mapping and localization tasks. VSLAM is traditionally deployed onboard the robot and directly interfaced with a camera that captures non-compressed video frames. However, VSLAM algorithms demand intensive computational resources, which has been a significant challenge in the last decade. Tracking and mapping are the two general tasks to be executed in parallel, which requires high computational costs. Such demanding computing requirements need powerful onboard resources, making them less suitable for mobile devices with severe hardware constraints that decrease their range and usage time. Recent studies have been conducted to provide solutions to reduce weight, save energy, and keep mobile device sizes small by offloading these computationally intensive processing tasks to edge servers and cloud platforms [12, 13]. This practical solution exhibits the need for efficient encoding methods to compress the image data before transmission to decrease bandwidth when performing VSLAM at the edge or in the cloud. Therefore, data compression is necessary to manage and transmit visual information and preserve as much information as possible, especially in high-latency communication environments. While image compression provides the clear advantages of reducing data load and minimizing bandwidth requirements, it introduces challenges for VSLAM, such as degradation of feature quality, mismatched feature descriptors, and increased computational complexity [14].

Image compression is the technique of reducing the image and video volumes by representing the spatial pixels in different compressed domains using fewer bits while maintaining visual information to a certain extent. In general, image compression approaches can be classified into two major categories: lossless and lossy compression. Lossy compression methods involve reducing the size of an image and perceiving every detail of the original image by eliminating statistical redundancy only [15]. The counterpart method is lossy compression, which aims to achieve high compression ratios by discarding some of the less significant image information for perception. Lossy compression is popular in machine vision tasks as it provides more data reduction capabilities [16]. However, it may influence image quality adversely, making features more intricate to detect and track during visual SLAM operation and affecting localization accuracy negatively. There are two different approaches in the literature regarding cloud-

based machine vision applications. The first transmits the image-extracted features of pre-processed raw data on the mobile device. This approach is referred to as Analyze then Compress (ATC). The second is compressing the raw data directly and transmitting it to the server cloud, which is referred to as Compress then Analyze (CTA). These approaches differ in the order of steps, the amount of data to be transmitted, and the end-to-end latency from the moment of visual data capturing to the destination [14].

Many studies have researched the offloading of VSLAM architecture on edge devices and cloud platforms to perform trajectory mapping and localization of mobile reboots. Riazuelo et al. [17] proposed and implemented a cloud framework for Cooperative Tracking and Mapping (C^2TAM). The proposed architecture involves implementing a centralized map on the cloud server while tracking is still running on mobile devices. In this manner, the mobile devices send the detected keyframes to the cloud server in a non-compressed format of 640×480 Red, Green, and Blue (RGB) images. That means an average throughput of 1 MBs is required for transmission. Despite the study not evaluating the transmission rate of keyframes, transmitting all raw-sized keyframes demands significant bandwidth. In [18], the Portable Network Graphic (PNG) standard was used to compress the extracted keyframes in collaborative 3D dense visual odometry assisted by the cloud. The cloud integrates the received keyframes, m erging them with maps constructed from other robots, then optimizes the pose and relays it to the mobile device. Fabrizio et al. [19] analyzed the influence of lossy data compression on the data size and accuracy. The H264 video codec is adapted to reduce the size of depth data captured by a Kinect camera and 3D LiDAR on a mobile robot. The study aimed to solve the problem of transmitting range data streams over low bandwidth networks. The proposed method proved that highly compressed depth images can still be used in dense mapping algorithms. In contrast, Jingao et al. presented a real-time VSLAM for edge agents. The proposed architecture integrates lossy compression for raw data encoding. It demonstrated that the lower bit rates introduced by lossy compression have a negative effect on the feature extraction quality [20]. In the literature, many compression standards are introduced. These methods can be categorized as classical methods and learn-based methods. Classical methods use predefined mathematical techniques such as transforms and entropy encoding. In contrast, learn-based image compression methods use machine learning techniques to compress data to higher compression ratios. However, classical methods are still a reasonable choice for mobile device implementation due to their lower computational complexity compared to the massive computational demands of learned methods [21, 22, 23].

The main contribution of this article is to present novel multi-encoding visual data compression methods used to compress input frames to high compression ratios while preserving the information quality. It examines the feasibility of integrating within the CTA and ATC frameworks of ORB-VSLAM. The proposed method is compared to corresponding VSLAM architectures using the JPEG standard and

the more advanced version, the JPEG 2000 standard. The JPEG is the typical and widely used traditional lossy compression standard, while the JPEG 2000 is a more efficient lossy compression standard employing multiresolution capabilities based on the Discrete Wavelet Transform (DWT) [24]. Inherently, this work also demonstrates the effect of using lossy compression on the system performance in terms of data size, execution time, and trajectory estimation accuracy. The subsequent sections of this article are outlined as follows: Section 2 provides an in-depth explanation of the ORB-VSLAM architecture and operational mechanism. Section 3 presents the proposed end-to-end multi-encoding method in detail, while Section 4 demonstrates the experimental results achieved from the implementation of the proposed methods, with a comparative analysis against the JPEG and JPEG 2000 standards. Finally, the conclusion section encapsulates the key findings of the study and indicates further future directions.

2 Monocular ORB-SLAM

Visual SLAM estimates the pose and reconstructs a map of the surrounding environment using data collected from visual sensors. It continuously updates the global map to decrease drifts, allowing localization and loop-closing detection. Generally, VSLAM methods are divided in to direct and indirect (feature-based) categories. Direct methods work directly on raw pixel data for pose estimation and reduce photo-metric errors, while indirect methods extract feature points (key points) and match them with subsequent ones. There are various feature detection algorithms, including Oriented FAST and Rotated BRIEF (ORB) [25], Speeded Up Robust Features (SURF) [26], Scale-Invariant Feature Transform (SIFT) [27], and Features from Accelerated Segment Test (FAST) [28].

A popular VSLAM framework is the ORB-SLAM proposed by [29]. It leverages ORB features for robust and efficient feature detection, extraction, and matching. Figure 1 illustrates the typical architecture of VSLAM [1]. The operation starts with map initialization, where two consecutive frames are used to construct 3D map points by extracting and matching the corresponding features. After the map is initialized successfully, the tracking extracts and matches features of the input frames to the map to localize the camera with each frame. Then, local mapping manages and optimizes the map by performing bundle adjustment of the current location in the environment. Finally, the loop closure distinguishes large loops and corrects camera trajectory drifts with the aid of a Bag of Words (BoW). The BoW is a technique that detects previously visited areas and loops by comparing the current pose with the previously mapped ones. Successful loop closing detection enhances the accuracy of the map and the camera trajectory, emphasizing the robustness and reliability of VSLAM [30].

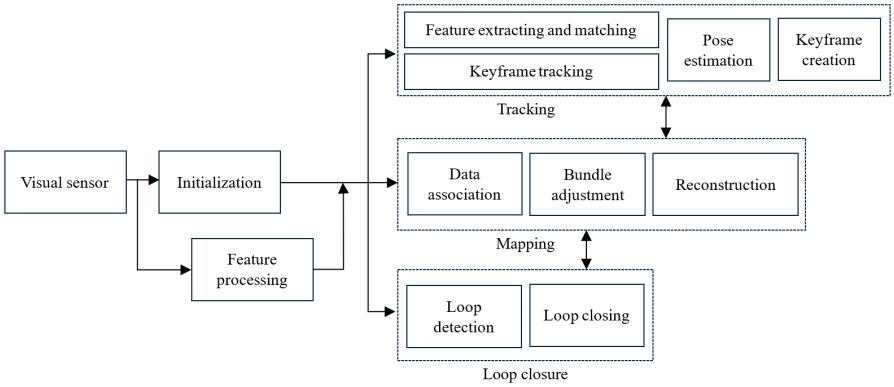


Figure 1
ORB-SLAM Architecture

3 Methodology

This section demonstrates the end-to-end multi-level encoding methods tailored for the ORB-SLAM system. The proposed methods aim to optimize image compression while preserving essential visual information presented in video frames as features.

3.1 Proposed Encoding Method

The encoding algorithm is implemented on the mobile device, which receives the raw frame input data directly from the attached camera. It can be used in both CTA and ATC frameworks. In CTA, the mobile device compresses each input frame and transmits the encoded data to the cloud. The cloud decodes the received encoded data and processes the ORB-SLAM tasks. Regarding the ATC framework, the mobile device deploys both the proposed image encoding algorithm along with the tracking module to maintain instant real-time operation. In this case, the tracking is accomplished on the raw-sized data while the encoded frames are transmitted to the cloud for map reconstruction and loop-closing operation. These framework options are selected depending on the application at hand and the computational power of the mobile device itself. Figure 2 illustrates the proposed encoding method's pipeline. The proposed method develops the basic JPEG compression standard for more efficient ORB-SLAM performance. The method starts by subdividing the input frame matrix into 8×8 sub-frames followed by Discrete Cosine Transform (DCT) to convert the frame to its frequency domain. For every sub-frame, 64 frequency components are produced,

one as a DC component and the other 63 as AC components. The DC component (average intensity) holds the most significant energy of the frame, while the AC components represent the variant frequency details. The DCT coefficients are then quantized using matrix and scalar quantization. The JPEG quantization table is updated by replacing the value of the DC component with 1 as well as the values of its five nearest AC components. The process of excluding the DC and the nearest five AC components from quantization is to increase the quality of the compressed frame by preserving image details. However, this process will decrease the compression ratio since it increases compressed frame quality. A multi-level encoding method is proposed to provide a high compression ratio for the encoded frames and perceive their quality as high. The quantized coefficients are converted into a 1D array by applying a zigzag scan, then a novel multi-level encoding algorithm is applied to reduce the 1D array size by a "number of levels" factor based on the user selection. Figures 3 and 4 demonstrate the multi-level encoding concept with two case study examples, a single-level and two-level encoding, respectively.

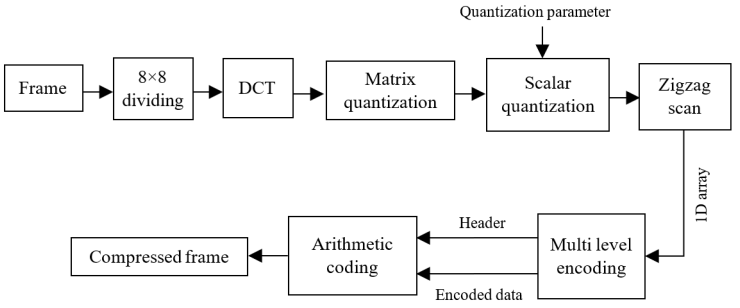


Figure 2
The proposed image compression pipeline

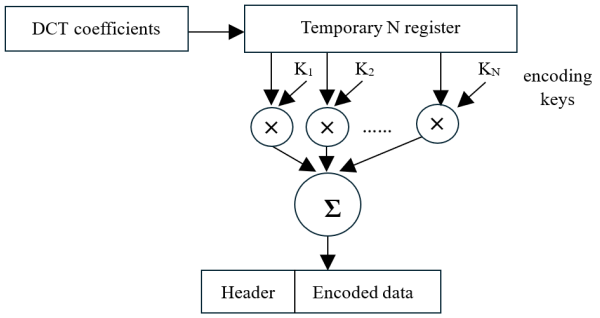


Figure 3
A single-level encoding implementation

The multi-level encoding method encodes data based on randomly generated keys in the range of 0 to 1, which can be generated automatically or simply set manually. In every encoding level, the data is encoded based on the multiplying and summation operations of the selected keys with corresponding coefficient data based on the same number of keys selected in that level. The encoded information is stored as a table, which includes the original data probabilities as a header, as well as the encoded data without any duplication, and then it is further encoded later using arithmetic coding. Finally, the compressed data of the frame (in the CTA framework) or keyframe (in the ATC framework) is sent to the cloud server for tracking, mapping, and loop-closing operations.

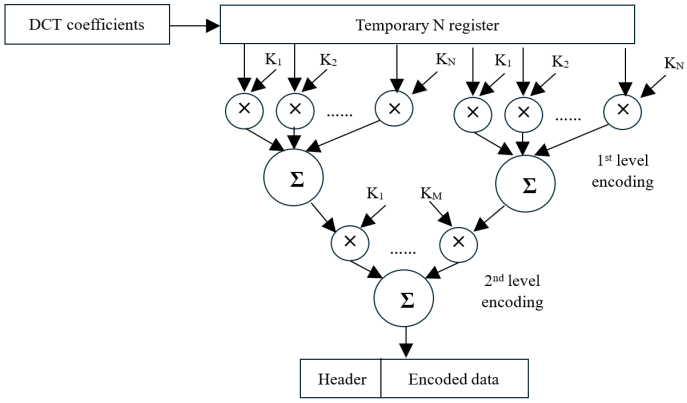


Figure 4
A two-level encoding implementation

3.2 Proposed Decoding Method

The decoding pipeline is identical to the compression steps but in the opposite order. The proposed decompression block diagram is illustrated in Figure 5. The decompression algorithm is deployed on the cloud server. The received data stream is first arithmetically decoded to retrieve the header and the encoded data by the proposed encoding algorithm in the compression phase. After this step, the data can be decoded using two possible methods. The first one is to use the header data to reconstruct the original data. In this method, the data can be decoded instantly without any extra computations. The other method involves discarding the header data during the compression phase and estimating the original data using a fast sequential search with the aid of the same encoding keys adopted during compression, as can be demonstrated in Figure 6. This means we need to implement the same encoding module that was used previously in compression. This will increase the compression ratio since it just keeps the compressed data and discards the extra header data. However, the reconstruction time will be

significantly increased due to the massive estimation iterations needed. This could be less problematic if we take into consideration harnessing the great cloud computing capabilities. For the scope of this work, the second method is considered to achieve a high compression ratio.

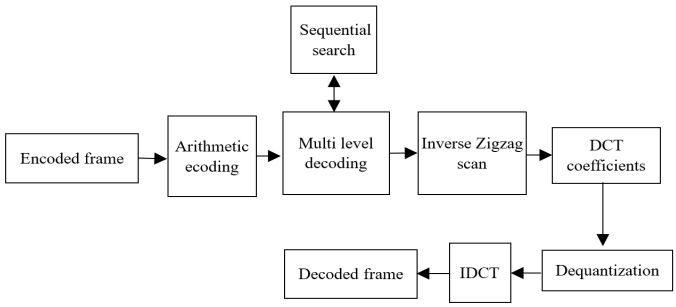


Figure 5
The proposed image decoding pipeline

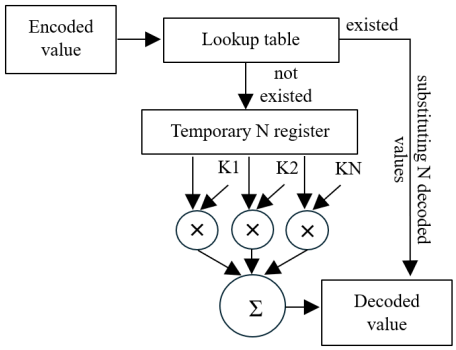


Figure 6
Mechanism of coefficient estimating using sequential search

4 Experimental Results and Discussion

The proposed method was implemented using MATLAB R2022a, running on an Intel Core i7-12700H processor, NVIDIA GeForce RTX3070 Graphical Processing Unit (GPU) with 8 GB of memory, and 32 GB of RAM (Random Access Memory). The TUM dataset [31] is used to evaluate the implementation results and to compare the performance of the proposed methods with the traditional JPEG and JPEG 2000 standards. It is a widely used dataset for evaluating the performance of VSLAM systems providing RGB images, depth

maps, and the camera trajectory's ground truth [32]. The RGB images and ground truth of the sequence "freiburg3 long office household" are used in this work. It is an indoor dataset captured using the Asus Xtion sensor containing 2585 frames recorded at a frame rate of 30 fps. The trajectory diameter of the dataset is 5.12 m ×4.89 m×0.54 m, and its duration is 87.09 s with a resolution of 640×480 pixels. The ending of the trajectory overlaps well with the beginning to reveal loop closure.

Firstly, the traditional ORB-SLAM is implemented when using full-sized raw visual data. The resulting estimated trajectory graph performance of this implementation is illustrated in Figure 7, which shows three trajectory graphs: the actual trajectory, the ORB-SLAM estimated trajectory, and the optimized trajectory. The map points are always corrected and updated according to the optimized pose trajectory graph, which is calculated from the actual trajectory and the estimated trajectory. The Root Mean Square Error (RMSE) is provided to evaluate the accuracy of the trajectory at each step of the simulation. The RMSE measures the absolute deviation between the estimated and ground truth trajectories. The initial coordinate starts at (0,0,0), and according to the dataset used in this work, the robot moves in the clockwise direction as indicated in Figure 7 and for the subsequent figures.

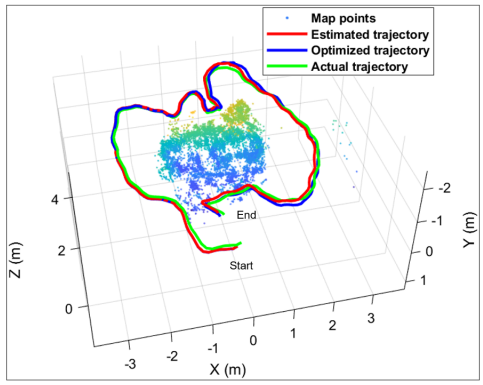


Figure 7
Standard ORB-SLAM trajectory estimation performance (RMSE=0.22114)

Secondly, the ORB-SLAM system is implemented by integrating the JPEG standard to encode the input data frames. The process of decoding, tracking, mapping, and loop closing is deployed as a single module emulating the cloud processing. The cloud module receives the encoded data, decodes it, and performs the trajectory estimation. This cloud-based ORB-SLAM architecture will be also applied to both the JPEG 2000-based system and the proposed system. In general, the purpose of using lossy image compression is to provide high compression ratios by quantizing the coefficients obtained from the discrete transformation. Therefore, additional different scalar Quantization Factors (QFs) are applied to

increase the compression ratio. Furthermore, this will demonstrate the trade-off between visual data quality and trajectory estimation accuracy. Figure 8 displays the trajectory estimation performance of the ORB-SLAM utilizing the JPEG standard as an end-to-end encoding and decoding framework. It illustrates the trajectory estimation among different QFs in the range of 5 to 20 with a step value of 5. It can be deduced from Figure 8 that JPEG compression guaranteed robust ORB-SLAM function among QF up to 15, and the loop closure in these cases has been detected accurately. However, when the quantization is increased up to 20, the trajectory drifts, and the system cannot continue mapping the remaining environment because not enough features are detected in this area. The reason behind this failure is that JPEG compression adds compression noise introduced as artifacts in the reconstructed frame, thereby negatively affecting the process of detecting and matching ORB features. Table 1 summarizes the ORB-SLAM performance metrics that adopted the JPEG compression. It previews the compression ratio, Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and execution time against different quantization values.

Thirdly, the ORB-SLAM is implemented by integrating the JPEG 2000 standard within the ORB-SLAM architecture to encode and decode data frames.

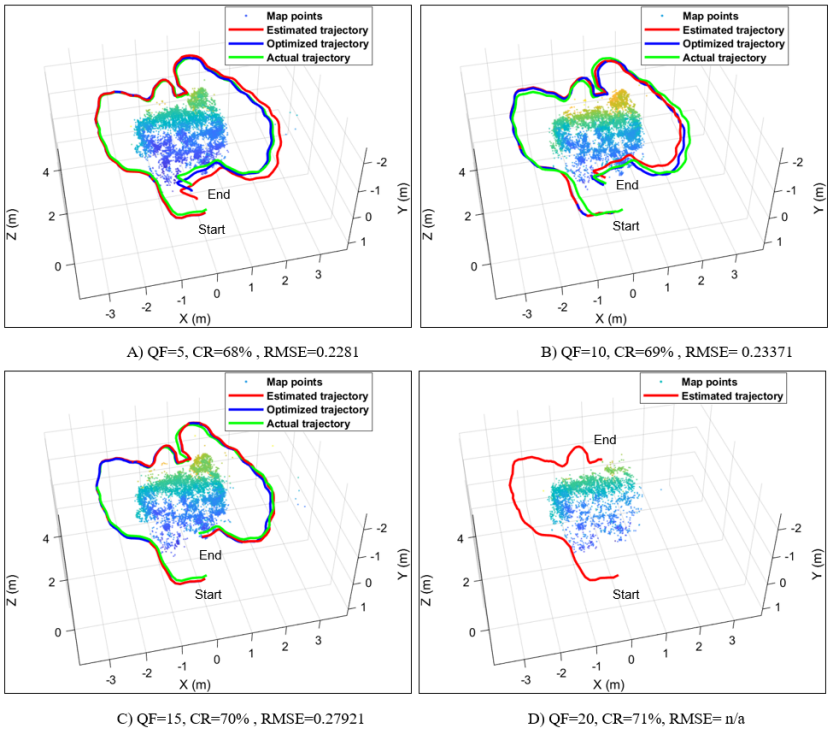


Figure 8
ORB-SLAM trajectory estimation performance adopting JPEG compression

The same procedure of splitting the cloud-based ORB-SLAM tasks between the mobile device and the cloud demonstrated in JPEG-based system implementation is applied. Unlike the JPEG standard, JPEG 2000 compresses the input frames according to a desired Compression Ratio (CR) parameter rather than specifying a quantization parameter. Figure 9 shows the trajectory estimation performance of the ORB-SLAM system employing the JPEG 2000 as an end-to-end encoding and decoding framework across different CRs. Examining Figure 9, the JPEG 2000 maintained a robust trajectory estimation up to a CR value of 97%, while at a higher value (CR= 98%) the system didn't detect loop closure. Table 2 displays the encoding and decoding performance metrics of the JPEG 2000-based system.

Table 1
Encoding performance metrics of the ORB-SLAM integrated JPEG compression

Sample frame Nr.	Original size [KB]	QF	Compressed size [KB]	PSNR [dB]	SSIM	CR [%]	Exe. time [s]	
							Enc.	Dec.
1132	472	5	153	34.8052	**0.9579**	68	0.3457	0.5991
		10	144	33.2016	0.9330	69	0.2341	0.5305
		15	141	32.3478	0.9148	70	0.2279	0.5279
		20	**140**	29.5512	0.8842	**71**	0.2217	**0.5071**
2398	505	5	166	**34.8084**	0.9414	67	0.2283	0.5394
		10	164	33.2872	0.9354	68	0.2205	0.5162
		15	163	32.4000	0.9167	68	**0.2195**	0.5139
		20	162	30.7533	0.8874	70	0.2306	0.5212

Table 2
Encoding performance metrics of the ORB-SLAM integrated JPEG 2000 compression

Sample frame Nr.	Original size [KB]	CR [%]	Compressed size [KB]	PSNR [dB]	SSIM	Exe. time [s]	
						Enc.	Dec.
1132	472	73	134	44.03	**0.9823**	0.110	0.440
		82	89.8	41.43	0.9721	0.136	0.412
		85	72	40.09	0.9647	0.112	0.444
		91	41	37.49	0.9483	0.195	0.532
		97	15	34.82	0.9279	0.192	0.492
		98	**8.9**	29.53	0.8647	0.212	0.417
2398	505	73	137.7	**44.69**	0.9819	0.112	0.432
		82	90	42.11	0.9708	0.121	**0.403**
		85	71.9	40.99	0.9460	0.115	0.515
		91	40.9	38.65	0.9485	**0.109**	0.636
		97	14.3	36.28	0.9303	0.181	0.636
		98	9	29.53	0.8835	0.162	0.512

Finally, ORB-SLAM is implemented by adopting the proposed multi-level encoding algorithm as an end-to-end visual data encoding and decoding. The trajectory estimation plots of the corresponding quantization and compression ratio conditions in both the adopted JPEG and the JPEG 2000 methods are

applied. The obtained simulation results of the trajectory estimation performance are presented in Figure 10 and Figure 11, while Table 3 summarizes the compression performance metrics of two implementations: a single-level and a two-level encoding scheme.

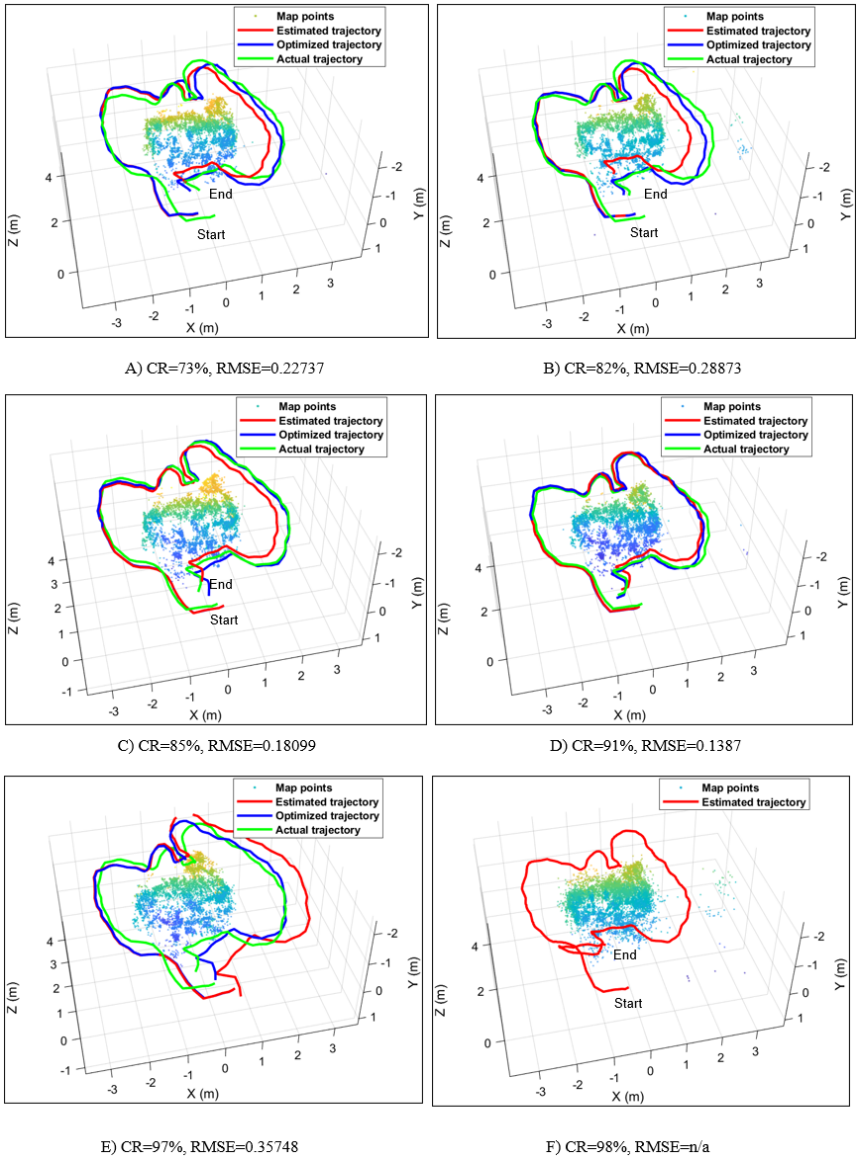


Figure 9
Trajectory estimation performance of ORB-SLAM system adopting JPEG 2000

The PSNR and SSIM metrics presented in Table 3 for both methods are the same since they achieved the same decompression quality. The reason behind the equivalent quality of the two schemes is that the multi-level encoding is lossless and does not add additional lossy compression to the encoded data.

Comparing the results of the proposed system to the JPEG-based system, at all QF cases (5 to 20), the proposed method showed robust pose estimation, and the loop closure was detected successfully as shown clearly in Figure 10. Higher values of QFs are further applied beyond the quantization values used in the JPEG-based ORB-SLAM method to evaluate the proposed method's performance under severe quantization. The proposed method showed robust trajectory estimation up to a QF value of 75. However, at higher quantization (QF=100) the proposed system failed to detect loop closure, these cases can be shown in Figure 11. Given the encoding metrics in Tables 1 and 2, it can be demonstrated that the ORB-SLAM integrated with the proposed system overcomes the equivalent JPEG-based system in terms of compression ratio, PSNR, and SSIM. This clear superiority is attributed to the multi-encoding module added to the compression system. However, this addition increased the execution time, which can be seen clearly in low QF values.

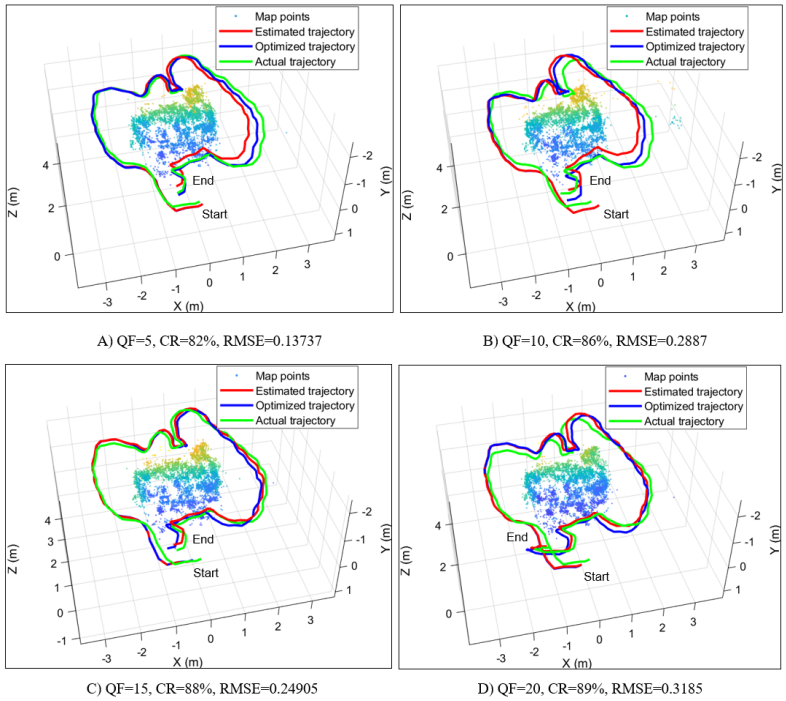
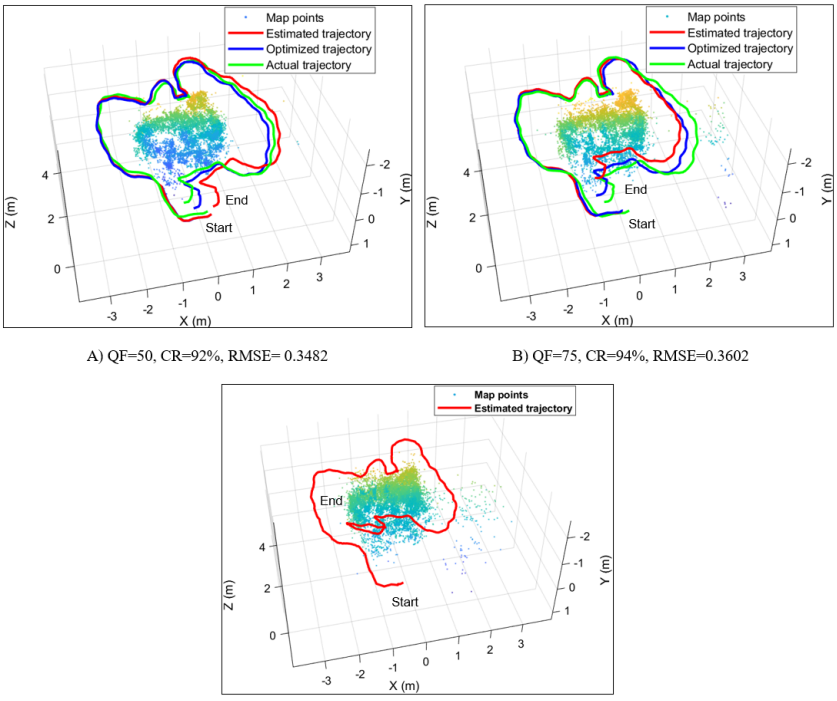


Figure 10
Multi-level based-ORB-SLAM trajectory estimation performance at lower QFs and CRs

As the QF increased, more duplicated values and zeros were present in the data, yielding faster processing, low header and encoded data size, and higher compression ratios. Nevertheless, this higher execution time does not affect real-time operation if we consider the transmission latency, which is beyond the scope of this work; transmitting smaller data sizes could compensate for the difference in execution time.



A) QF=50, CR=92%, RMSE= 0.3482 B) QF=75, CR=94%, RMSE=0.3602

C) QF=100, CR=96%, RMSE=n/a

Figure 11

Multi-level based-ORB-SLAM trajectory estimation performance at higher QFs and CRs

Comparing the results of the proposed system to the JPEG 2000-based system, the two systems exhibited close behaviors, with the JPEG 2000-based system demonstrating slightly better performance in decoding frame quality and execution time. Nevertheless, this slight encoding metrics advantage did not significantly enhance the overall ORB-SLAM system's performance. The proposed multi-level encoding system successfully estimated the path at a compression ratio of 96%, while the JPEG 2000-based system successfully estimated the path at a compression ratio of up to 97%. This minor difference came at the expense of increased computational complexity. The three implemented systems adopting the JPEG, the JPEG 2000, and the multi-level encoding are analyzed according to the computational complexity by calculating

the total basic operations: Multiplications (Mults.), Additions (Adds.), bit operations (Bit ops.), comparisons (Comps.), shifts, Memory Reads (M/R), and Memory Writes (M/W) for each stage. Tables 4, 5, and 6 show the computational complexity metrics for the three implemented systems processing a single input frame, while Figure 12 visualizes these complexity metrics.

Table 3
Encoding performance metrics of the ORB-SLAM integrating the proposed multi-level method

QF	PSNR [dB]	SSIM	Single level implementation				Two levels implementation			
			size [KB]	CR [%]	Exe. time [s]		size [KB]	CR [%]	Exe. time [s]	
					Enc.	Dec.			Enc.	Dec.
Frame sequence 1132, Frame size 472 KB										
5	35.2333	0.9515	87	82	0.7773	0.7201	86	82	0.8800	0.8307
10	33.5493	0.9398	68	86	0.6951	0.6093	67	86	0.7252	0.6966
15	32.6696	0.9328	59	88	0.6274	0.5198	58	88	0.7370	0.6451
20	32.2144	0.9271	54	89	0.5139	0.4351	50	89	0.6045	0.6160
50	31.0675	0.8961	38	92	0.4203	0.3960	30	94	0.4826	0.4378
75	30.1511	0.8721	32	93	0.3831	0.3106	22	95	0.4163	0.3926
100	29.2537	0.8501	**27**	94	**0.3013**	0.2877	**17**	**96**	**0.3937**	**0.3616**
Frame sequence 2398, Frame size 505 KB										
5	**35.3702**	**0.9517**	104	79	0.7932	0.7213	102	80	1.0329	0.9067
10	33.7746	0.9410	81	83	0.6231	0.6134	79	84	0.8411	0.7957
15	33.0977	0.9345	69	86	0.5932	0.4929	67	87	0.7535	0.6757
20	32.6712	0.9291	64	87	0.4982	0.4036	60	89	0.6904	0.6510
50	32.2985	0.8945	42	92	0.3864	0.3289	37	93	0.5370	0.4647
75	30.2647	0.8662	32	94	0.3225	**0.2854**	28	94	0.4656	0.4186
100	29.2861	0.8439	29	94	0.3019	0.2978	23	95	0.4337	0.3908

Table 4
Complexity analysis metrics of JPEG encoding (processing frame sequence 1132)

Stage	Mults.	Adds.	Bit ops.	Comps.	Shifts	M/R	M/W	Total ops.
Compression Metrics								
DC shift	0	307200	0	0	0	307200	307200	921600
DCT	307200	537600	0	0	0	307200	307200	1459200
Quantization	307200	0	307200	0	307200	307200	307200	1536000
Huffman cod.	0	0	399360	337920	61440	307200	61440	1167360
Total ops.	614400	844800	706560	337920	368640	1228800	983040	5084160
Decompression Metrics								
Huffman dec.	0	0	92160	61440	61440	61440	307200	583680
Dequantization	307200	0	307200	0	307200	307200	307200	1536000
IDCT	307200	537600	0	0	0	307200	307200	1459200
Inv. DC shift	0	307200	0	0	0	307200	307200	921600
Total ops.	614400	844800	399360	61440	368640	983040	1228800	4500480

Table 5
Complexity analysis metrics of JPEG 2000 (processing frame sequence 1132)

Stage	Mults.	Adds.	Bit ops.	Comps.	Shifts	M/R	M/W	Total ops.
Compression Metrics								
DC shift	0	307200	0	0	0	307200	307200	921600
DWT	8184000	6547200	0	0	3273600	3273600	1636800	22915200
Quantization	307200	0	0	307200	307200	307200	307200	1536000
Tier1 coding	0	0	3932160	2949120	0	1310720	655360	8847360
Tier2 coding	614400	614400	3072000	0	0	614400	307200	5222400
Total ops.	9105600	7468800	7004160	3256320	3580800	5813120	3213760	39442560
Decompression Metrics								
Tier2 decoding	0	307200	1843200	0	0	307200	307200	2764800
Tier1 decoding	0	0	2621440	983040	0	655360	327680	4587520
Dequantization	307200	0	0	0	307200	307200	307200	1228800
IDWT	6547200	4910400	0	0	1636800	1636800	1636800	16368000
Inv. DC shift	0	307200	0	0	0	307200	307200	921600
Total ops.	6854400	5524800	4464640	983040	1944000	3213760	2886080	25870720

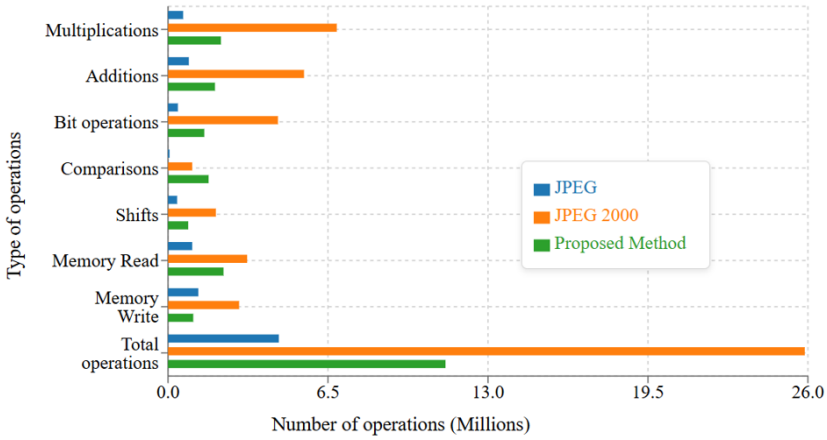
Table 6

Complexity analysis metrics of the proposed method using a single-level encoding (processing frame sequence 1132)

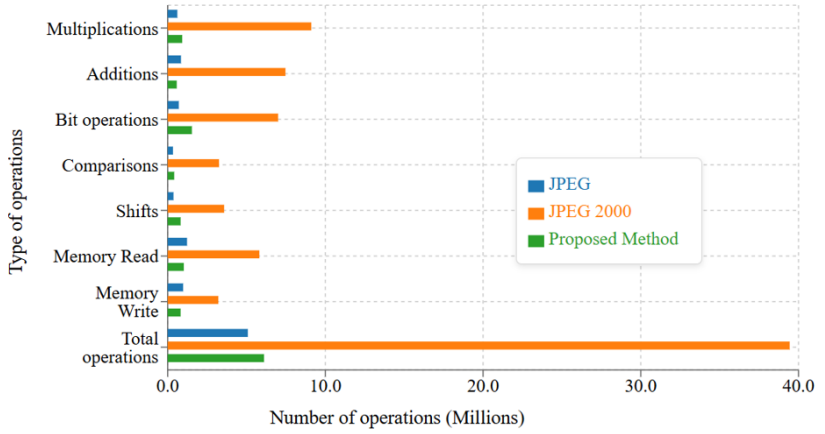
Stage	Mults.	Adds.	Bit ops.	Comps	Shifts	M/R	M/W	Total ops.
Compression Metrics								
DCT	307200	268800	307200	4800	0	307200	307200	1502400
Quantization	307200	0	307200	0	0	307264	307200	1228864
Minimization	307200	204800	102400	307200	0	307200	102400	1331200
Arithmetic coding	0	102400	819200	102400	819200	102400	102400	2048000
Total ops.	921600	576000	1536000	414400	819200	1024064	819200	6110464
Decompression Metrics								
Arithmetic decoding	0	102400	819200	102400	819200	102400	102400	2048000
Sequential search	1536000	1536000	307200	1536000	0	1536000	307200	6758400
Dequantization	307200	0	307200	4800	0	307264	307200	1233664
IDCT	307200	268800	38400	4800	0	307200	307200	1233600
Total ops.	2150400	1907200	1472000	1648000	819200	2252864	1024000	11273664

Examining the results presented in Tables 4, 5, 6, and Figure 12, the proposed method exhibited a clear advantage in terms of the computational complexity of both encoding and decoding modules compared to JPEG 2000. This obvious advantage comes with relatively equivalent trajectory estimation efficiency. On the other hand, the JPEG system shows slightly less complexity than the proposed system. However, the JPEG system showed much worse performance compared to the JPEG 2000 and the proposed systems. This makes the proposed system a reasonable compromise between the three implemented systems to be used in mobile robots. Another outcome of this work is to study the impact of lossy data compression on the reliability of ORB-SLAMs. The encoding performance results

presented in Tables 1, 2, and 3 demonstrated that ORB-SLAM is less robust when PSNR values go below 30 and SSIM values below 0.87.



a) Computational complexity of the encoding systems



b) Computational complexity comparison of the decoding systems

Figure 12
Computational complexity of the three implemented systems

Conclusions

This article presents a novel multi-level image encoding method tailored for cloud-based ORB-SLAM systems. The proposed method demonstrated outstanding performance compared to ORB-SLAM architectures employing JPEG and JPEG 2000 compression standards. The proposed system achieved superior performance compared to a corresponding JPEG-based system in terms of decoded frame quality metrics and robot trajectory estimation with only a minimal

increase in computational complexity. In addition, the proposed method demonstrated comparable performance results to the JPEG 2000-based system. In contrast, the mathematical operations analysis showed the high efficiency of the proposed system in terms of computational complexity compared to the JPEG 2000-based system, making it a practical solution for cloud-based ORB-SLAM systems. A future direction includes neglecting the compressed data header and harnessing the cloud computing power to directly decode the results without estimating them, which can significantly increase the compression ratio. Furthermore, research is needed for efficient hardware implementation using cutting-edge devices, like Field Programmable Gate Arrays (FPGA), since it is well suited to deploying the multiply accumulated operation presented in the proposed method.

References

[1] A. Tourani, H. Bavle, J. L. Sanchez-Lopez, and H. Voos, "Visual SLAM What are the Current Trends And What To Expect?," Sensors, Vol. 22, No. 23, p. 9297, 2022

[2] I. Vallivaara, J. Haverinen, A. Kemppainen, and J. Röning, "Magnetic Field-based SLAM Method for Solving the Localization Problem in Mobile Robot Floor-cleaning Task," in 2011 15th International Conference on Advanced Robotics (ICAR), pp. 198-203, IEEE, 2011

[3] Q. Zou, Q. Sun, L. Chen, B. Nie, and Q. Li, "A Comparative Analysis of LiDAR SLAM-based Indoor Navigation for Autonomous Vehicles," IEEE Transactions on Intelligent Transportation Systems, Vol. 23, No. 7, pp. 6907-6921, 2021

[4] T. Yang, P. Li, H. Zhang, J. Li, and Z. Li, "Monocular Vision SLAM-based UAV Autonomous Landing in Emergencies and Unknown Environments," Electronics, Vol. 7, No. 5, p. 73, 2018

[5] Z. Liu, H. Chen, H. Di, Y. Tao, J. Gong, G. Xiong, and J. Qi, "Real-time 6D LiDAR SLAM in Large Scale Natural Terrains for UGV," in 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 662-667, IEEE, 2018

[6] G. Lu, H. Yang, J. Li, Z. Kuang, and R. Yang, "A Lightweight Real-time 3D LiDAR SLAM for Autonomous Vehicles in Large-scale Urban Environment," IEEE Access, Vol. 11, pp. 12594-12606, 2023

[7] B. Amjad, Q. Z. Ahmed, P. I. Lazaridis, M. Hafeez, F. A. Khan, and Z. D. Zaharis, "Radio SLAM: A Review on Radio-based Simultaneous Localization and Mapping," IEEE Access, Vol. 11, pp. 9260-9278, 2023

[8] A. Macario Barros, M. Michel, Y. Moline, G. Corre, and F. Carrel, "A Comprehensive Survey of Visual SLAM Algorithms," Robotics, Vol. 11, No. 1, p. 24, 2022

[9] J-C. Trujillo, R. Munguia, S. Urzua, E. Guerra, and A. Grau, "Monocular Visual SLAM-based on a Cooperative UAV–target System," Sensors, Vol. 20, No. 12, p. 3531, 2020

[10] J. Mo, M. J. Islam, and J. Sattar, "Fast Direct Stereo Visual SLAM," IEEE Robotics and Automation Letters, Vol. 7, No. 2, pp. 778-785, 2021

[11] B. Li, Y. Guo, Z. Mi, Y. Yang, and M. S. Obaidat, "A Novel Data Compression Technique Incorporated with Computer Offloading in RGB-D SLAM," in 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), pp. 1-5, IEEE, 2019

[12] S. Eger, R. Pries, and E. Steinbach, "Evaluation of Different Task Distributions for Edge Cloud-based Collaborative Visual SLAM," in 2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP), pp. 1-6, IEEE, 2020

[13] A. J. Ben Ali, M. Kouroshli, S. Semenova, Z. S. Hashemifar, S. Y. Ko, and K. Dantu, "Edge-SLAM: Edge-assisted Visual Simultaneous Localization and Mapping," ACM Transactions on Embedded Computing Systems, Vol. 22, No. 1, pp. 1-31, 2022

[14] J. Hofer, P. Sossalla, C. L. Vielhaus, J. Rischke, M. Reisslein, and F. H. Fitzek, "Comparison of Analyze-then-compress Methods in Edge-assisted Visual SLAM," IEEE Access, 2023

[15] M. A. Rahman, M. Hamada, and J. Shin, "The Impact of State-of-the-art Techniques for Lossless Still Image Compression," Electronics, Vol. 10, No. 3, p. 360, 2021

[16] A. J. Hussain, A. Al-Fayadh, and N. Radi, "Image Compression Techniques: A Survey in Lossless and Lossy Algorithms," Neurocomputing, Vol. 300, pp. 44-69, 2018

[17] L. Riazuelo, J. Civera, and J. M. Montiel, "C^{2}TAM: A Cloud Framework for Cooperative Tracking and Mapping," Robotics and Autonomous Systems, Vol. 62, No. 4, pp. 401-413, 2014

[18] G. Mohanarajah, V. Usenko, M. Singh, R. D'Andrea, and M. Waibel, "Cloud-based Collaborative 3D Mapping in Real-time with Low-cost Robots," IEEE Transactions on Automation Science and Engineering, Vol. 12, No. 2, pp. 423-431, 2015

[19] F. Nenci, L. Spinello, and C. Stachniss, "Effective Compression of Range Data Streams for Remote Robot Operations using H.264," in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3794-3799, IEEE, 2014

[20] J. Xu, H. Cao, Z. Yang, L. Shangguan, J. Zhang, X. He, and Y. Liu, "{SwarmMap}: Scaling up Real-time Collaborative Visual {SLAM} at the

Edge," in 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), pp. 977-993, 2022

[21] D. Mishra, S. K. Singh, and R. K. Singh, "Deep Architectures for Image Compression: A Critical Review," Signal Processing, Vol. 191, p. 108346, 2022

[22] S. Jamil, M. Piran, and M. Rahman, "Learning-driven Lossy Image Compression; A Comprehensive Survey," arXiv preprint arXiv:2201.09240, 2022

[23] M. A. Rahman and M. Hamada, "Lossless Image Compression Techniques: A State-of-the-art Survey," Symmetry, Vol. 11, No. 10, p. 1274, 2019

[24] I. A. Urbaniak, "Using Compressed JPEG and JPEG2000 Medical Images in Deep Learning: A Review," Applied Sciences 14, No. 22: 10524, 2024

[25] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An Efficient Alternative to SIFT or SURF," In 2011 International conference on computer vision, pp. 2564-2571, IEEE, 2011

[26] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up Robust Features (SURF)," Computer Vision and Image Understanding, Vol. 110, No. 3, pp. 346-359, 2008

[27] D. G. Lowe, "Distinctive Image Features from Scale-invariant Keypoints," International Journal of Computer Vision, Vol. 60, pp. 91-110, 2004

[28] E. Rosten and T. Drummond, "Machine Learning for High-speed Corner Detection," in Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9, pp. 430-443, Springer, 2006

[29] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," IEEE Transactions on Robotics, Vol. 31, No. 5, pp. 1147-1163, 2015

[30] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-source SLAM System for Monocular, Stereo, and RGB-D Cameras," IEEE Transactions on Robotics, Vol. 33, no. 5, pp. 1255–1262, 2017

[31] J. Sturm, N. Engelhard, F. Endres, W. Burgard and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 573-580, 2012

[32] B. Al-Tawil, T. Hempel, A. Abdelrahman, and A. Al-Hamadi. "A review of visual SLAM for robotics: evolution, properties, and future applications," Frontiers in Robotics and AI, Vol. 11, 1347985, 2024