2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Selection and Compression of Local Binary Features for Remote Visual SLAM

Dominik Van Opdenbosch, Martin Oelsch, Adrian Garcea, Tamay Aykut and Eckehard Steinbach

Abstract— **In the field of autonomous robotics, Simultaneous Localization and Mapping (SLAM) is still a challenging problem. With cheap visual sensors attracting more and more attention, various solutions to the SLAM problem using visual cues have been proposed. However, current visual SLAM systems are still computationally demanding, especially on embedded devices. In addition, collaborative SLAM approaches emerge using visual information acquired from multiple robots simultaneously to build a joint map. In order to address both challenges, we present an approach for remote visual SLAM where local binary features are extracted at the robot, compressed and sent over a network to a centralized powerful processing node running the visual SLAM algorithm. To this end, we propose a new feature coding scheme including a feature selection stage which ensures that only relevant information is transmitted. We demonstrate the effectiveness of our approach on well-known datasets. With the proposed approach, it is possible to build an accurate map while limiting the data rate to 75 kbits/frame.**

I. INTRODUCTION

Local visual features enable various visual analysis tasks ranging from visual search and object recognition to visual SLAM applications by providing the possibility to efficiently describe local parts of an image. In the recent past, local binary feature descriptors have been used to enable real-time visual localization and mapping in robotic applications, which is essential for exploration and navigation tasks. While the extraction of local binary features such as ORB [1] is feasible on embedded devices with both power and memory restrictions, the task of self-localization and mapping is rather complex. Therefore, we propose to outsource this task to a powerful (cloud-based) processing node and transmit only the needed information in form of local binary features to the processing node as shown in Figure 1. We name this approach *Remote Visual SLAM*, as all information is processed on a remote node. In order to reduce the amount of data, we propose a coding scheme for local binary features tailored to the particular needs of a state-of-the-art visual SLAM system, namely ORB-SLAM2 [2]. To the best of our knowledge, this is the first feature coding approach incorporating explicitly the requirements of a visual SLAM application. This paper contains the following contributions:

1) We propose an improved coding scheme for binary local features which is specifically tailored to the needs of visual SLAM based on existing work on local binary feature coding [3].

Authors are with the Department of Electrical and Computer Engineering, Chair of Media Technology, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany dominik.van-opdenbosch@tum.de

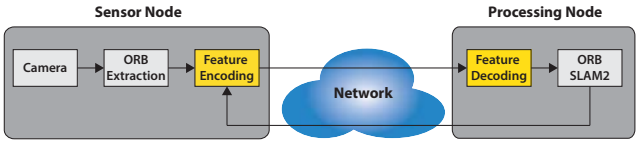


Fig. 1. Transmission path from a sensor node to a central processing node. The sensor node is responsible for capturing the image data, performing feature extraction and transmitting the coded features over a channel to the processing node. A visual analysis task such as ORB-SLAM2 is responsible of interpreting the data and providing feedback to the sensor node.

2) We use the scheme described in [4] to jointly code local and global image signatures by exploiting the statistical dependencies between the Bag-of-Words (BoW) representation and the feature descriptors. Additionally, we propose to directly reuse the Bag-of-Words representation as a global image signature for tasks like re-localization and loop closure detection.
3) We add a feature selection step to the encoder to prioritize useful features by means of a probabilistic model, which includes the current tracking state, coding mode decision and keypoint properties, to meet the real-time constraints and to stay within the available transmission capacity. In order to adapt the feature coding to the tracking state, we add a feedback channel to signal the current tracking state.

The rest of the paper is organized as follows: In Section II, we discuss the related work for visual SLAM, remote SLAM and visual feature coding. In Section III, we introduce our system architecture. Section IV details the used coding modes and the mode decision algorithm. Section V introduces the feature selection and the feedback channel. We demonstrate the effectiveness of the remote visual SLAM system in Section VI and draw conclusions in Section VII.

II. RELATED WORK

A. Visual SLAM

Despite growing computational resources, today's visual SLAM algorithms still cannot process all sensor data in real-time. State-of-the-art visual SLAM approaches use keyframe-based methods to consider only a subset of the past frames, thus lowering the computational burden. Probably the most prominent representative of the keyframe-based approaches is the Parallel Tracking and Mapping (PTAM) approach by Murray and Klein [5]. They split the visual SLAM problem into two separate problems: Tracking given a known map and mapping to extend the map. Following this idea, two different types of approaches have emerged. On the one hand, feature-based approaches such as PTAM

978-1-5386-3081-5/18/$31.00 ©2018 IEEE

use only small parts of the image around local keypoints for triangulation of map points. On the other hand, dense (DTAM [6]) or semi-dense approaches (LSD-SLAM [7]) try to minimize the photometric error between frames using the raw pixel intensities. However, (semi-) dense approaches have several drawbacks such as susceptibility to changing lighting conditions and strong viewpoint changes. While for dense approaches, the pixel intensities have to be available for the algorithm to work, feature-based approaches only need the keypoint locations and the feature descriptors. Therefore, using only local image features greatly reduces the information required to run the algorithm and makes this class of visual SLAM systems suitable for robotic applications, where memory footprint, computational complexity and the available transmission capacity to exchange information play an important role. One of the recent state-of-the-art approaches for feature-based visual SLAM is ORB-SLAM2 [8], [2], which combines most of the recent findings on visual SLAM into a single framework. Although the combination of the keyframe-based approach and binary features greatly reduces the computational requirements, the processing power in many applications is still quite limited. Especially in robotic exploration tasks, the battery consumption should be limited to a minimum to achieve long-term operation. In addition, multi-core architectures required for running tracking, mapping and loop-closing tasks in parallel are often not available. Following this constraint, we propose to outsource the visual SLAM algorithm to a central, powerful processing node and to transmit only the essential information required to run the algorithm. Furthermore, this also allows for collecting information from several robots and merging them into a single map for collaborative mapping. After processing, the current position and the map can be transmitted back to the robot. Alternatively, the map can be used for further high-level tasks at the central node such as path planning or central orchestration of a robot swarm.

B. Remote Visual SLAM

There are two different approaches for remote visual SLAM: Compress-then-analyze (CTA), where the visual information is transmitted as a compressed video sequence, and analyze-then-compress (ATC), where the visual cues in form of local features are extracted, compressed and transmitted. For example, Martinez-Carranza et al. [9] proposed a remote visual SLAM setup for ORB-SLAM where they streamed the full image information from a micro aerial vehicle (MAV) using H.264 over a WiFi connection. Riazuelo et al. [10] proposed to provide visual SLAM as a cloud service by running the expensive map optimization on a cloud server and a fast tracking on the robot. In their scheme, they upload the keyframes as RGB data to the server requiring a considerable amount of transmission rate. Schmuck et al. [11] proposed to run a visual SLAM system with a local map on each robot, but collect and merge all the information at a centralized server.

C. Feature Coding

For the CTA approach, usually the whole image or video is coded and transmitted. At the receiver node, the analysis task is carried out on the decoded image. Although there are several choices for feature-preserving image [12] and video coding approaches [13], most parts of the image might be irrelevant for the targeted task and keypoint locations are not well preserved through the compression step [14]. In consequence, we apply an ATC-based scheme optimized for visual SLAM. For visual feature coding, the Compact Descriptors for Visual Search (CDVS) standard has been proposed by MPEG [15], focusing mainly on descriptor compression for computationally demanding SIFT-like [16] features optimized for visual search. A visual SLAM system relies on accurate keypoint locations for triangulation, whereas in visual search the keypoint position is often only used for a subsequent geometric consistency check. The proposed coding method for the keypoint locations within the MPEG-CDVS standard uses a location histogram [17], [18], which includes a quantization step leading to a loss in accuracy. Apart from the MPEG-CDVS approach, there are other approaches for coding both descriptors and keypoints. Starting with real-valued descriptors like SIFT and SURF [19], Baroffio et al. proposed to code visual features extracted from video sequences using a scheme close to hybrid video coding [20]. Their approach supports intra- and inter-frame coding to exploit both redundancies within the descriptor itself and correlation between matching features of successive frames. While this work was based on real-valued descriptors, binary descriptors such as ORB [1] became a faster alternative suitable for most computer vision tasks. Following this development some work has been done to compress binary features [21], [3]. We use the general structure of [3] for coding binary features in our remote ORB-SLAM2 framework.

III. System Architecture

The overall system architecture is depicted in Figure 1. A sensor node, e.g. a robot, is responsible for acquiring image data as well as extracting and coding the binary features. We send the coded features over a communication network to a powerful processing node running the monocular version of ORB-SLAM2. A feedback channel returns information about the current tracking state and possibly the result of the SLAM system including map and position information. The structure of the feature coding system is shown in Figure 2. The proposed system for feature coding is similar to hybrid video coding and based on the findings of Baroffio et al. [3], consisting of coding modes for intra-frame coding, inter-frame coding, skipping features and a mode for static scenes coined inter-frame skip. We highlight the parts that are different in our approach compared to previous works. First, we replace the intra-frame coding from Baroffio et al. by an approach which uses a visual vocabulary as shared knowledge as introduced in [4]. The visual word information can directly be reused as part of

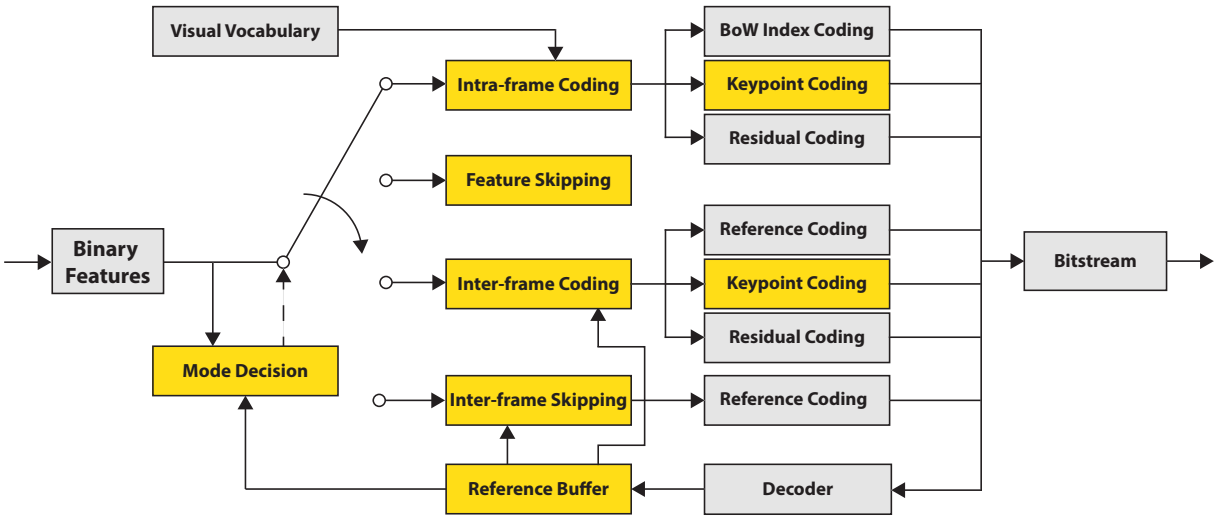


Fig. 2. Overview of the proposed feature coding architecture. The main coding modes are intra-frame and inter-frame. Additionally, the mode decision can skip several features to stay within a certain bit-budget or time limit. We have highlighted the main parts that have been modified compared to previous feature compression systems (yellow).

the Bag-of-Words global image signature employed in ORB-SLAM2 for re-localization and loop closing. The inter-frame coding exploits the temporal correlation between successive frames and selects features within a certain search area in the previous frames as reference for the current feature. In this respect, we extend the previous work [3] by adding the support for multiple reference frames. For each coding mode, we have extended the keypoint coding by a lossless coding mode for the keypoint position to provide as accurate keypoint positions as possible. Before starting the actual coding, the best coding mode decision and the resulting rate for each feature are calculated and the features are then sorted according to a probabilistic model. We include statistics about useful features depending on the tracking state of the visual SLAM system, the mode decision and the keypoint properties. The goal is to prioritize the features, which are more relevant for the visual SLAM task. If the transmission capacity or a time limit is reached, the remaining features are skipped and not transmitted.

IV. Feature Coding

For the sake of consistency, we use a similar notation as [3] in the rest of our paper to describe the system. A local visual feature usually consists of two parts. First, the properties such as position in the frame, scale and orientation are denoted as keypoint. The second part is the binary visual descriptor, in form of a binary string $\mathbf{d} \in [0,1]$ with length D ($D = 256$ for ORB features). The total cost in terms of the required number of bits for any coding mode is a combination of the individual costs for coding the descriptor, the keypoint, the visual word index for intra-coding or the reference feature for inter-coding. For the coding mode m, we use the following notation: $m \in \{I, P, S\}$, where I denotes intra-frame coding, P denotes inter-frame predictive coding and S the inter-frame skip mode. We will detail the costs for each coding mode in the following.

A. Intra-Coding

The intra-frame coding mode differs from the one introduced in [3]. The authors exploit the correlation within a binary feature descriptor by modeling the descriptor as a first-order Markov source. They try to find a permutation of the descriptor elements that minimizes the cost of coding neighboring elements and use this information to code the reordered descriptor elements based on their predecessor. However, the authors of ORB [1] state that they have performed a training of the pixel tests in order to minimize the correlation among the descriptor entries. Therefore, we decided to use the visual words, which are used in ORB-SLAM2 for fast feature matching and as part of the global image signature for loop closure detection and re-localization. In our approach, we use the visual words as reference for coding the feature descriptors and employ the visual vocabulary as shared knowledge. This allows us to directly reuse the already calculated visual words in ORB-SLAM2. The details for the residual coding can be found in [4], which we will briefly explain in the following. Additionally, we introduce a lossless keypoint coding method using the scale pyramid employed for ORB features. The total cost for intra-frame coding can be written as

$$R^I_{n,i} = R^{I,BoW}_{n,i} + R^{I,des}_{n,i} + R^{I,kpt}_{n,i} \tag{1}$$

where $R^{I,BoW}_{n,i}$ is the cost for signaling the visual word index, $R^{I,des}_{n,i}$ for coding the descriptor part and $R^{I,kpt}_{n,i}$ for transmitting the keypoint information for feature i from image n. In the following, we will calculate the lower bound for this rate. We use arithmetic coding throughout our framework to approach this lower bound.

1) Visual Word Index Coding:

The visual vocabulary C consists of S visual words $\mathbf{c}_v$ arranged in an hierarchical vocabulary tree

$$C = \{\mathbf{c}_1, \mathbf{c}_2, \dots \mathbf{c}_S\}. \tag{2}$$

The size S of the hierarchical vocabulary is calculated as $S = b^l$, where b is the branching factor and l is the maximum depth of the tree. First, we assign each descriptor to its closest representative in the visual vocabulary such that

$$v^* = \arg\min_v H(\mathbf{d}_{n,i}, \mathbf{c}_v), \tag{3}$$

where H denotes the Hamming distance between the descriptor and the visual word and v^* the best matching visual word index. It is worth mentioning that the hierarchical structure implements a very fast but only approximate nearest neighbor search. We assume the probability of the visual words to be uniformly distributed. So we need

$$R_{n,i}^{I,BoW} = \log_2(S) \tag{4}$$

bits to signal the visual word index v^*.

2) Descriptor Coding:
After we have identified the closest match, we compute the residual vector $\mathbf{r}_{n,i}^I = \mathbf{d}_{n,i} \oplus \mathbf{c}_{v^*}$. In binary space, this translates into an XOR operation between the descriptor $\mathbf{d}_{n,i}$ and the visual word $\mathbf{c}_{v^*}$. We then apply arithmetic coding on the resulting residual vector. In general, the lower bound for coding a binary string of length D with h denoting the number of non-zero values and p_0 the probability of a symbol being zero is given by

$$R_{res}(h, p_0) = -(D-h) \cdot log_2(p_0) - h \cdot log_2(1-p_0). \tag{5}$$

For the intra-frame coding, we can calculate $h_{n,i}^I$ as the non-zero entries of the residual vector $\mathbf{r}_{n,i}^I$, which corresponds to the Hamming distance between the descriptor and the matching visual word as

$$h_{n,i}^I = H(\mathbf{d}_{n,i}, \mathbf{c}_{v^*}). \tag{6}$$

Using (5), we can calculate the lower bound for the required rate for coding the intra-frame residual:

$$R_{n,i}^{I,des} = R_{res}(h_{n,i}^I, p_0^I). \tag{7}$$

In the following, we will address the coding of the keypoint properties for intra-frame coding.

3) Keypoint Coding:
We define a keypoint $\mathbf{k}_{n,i} = [x, y, \sigma, \hat{\theta}]$ as the set of properties used in ORB-SLAM2, where x and y are the position within the image, $\sigma \in [1, N_\sigma]$ denotes the level of the scale pyramid where the feature has been extracted and $\hat{\theta} \in [1, N_{\hat{\theta}}]$ denotes the quantized keypoint orientation. For the quantization of the orientation, we divide the circle into $N_{\hat{\theta}}$ equally spaced orientation bins. In contrast to previous work [3], which used BRISK features, we focus on the specific properties of ORB features. While BRISK features introduce an additional interpolation step by fitting a 1D parabola to estimate the final scale and keypoint position, ORB only operates on distinct scale levels. This opens the possibility to employ lossless keypoint position coding by converting the keypoint position back to the scale level where the keypoint has been detected. In their respective scale level, the keypoint positions are extracted at integer positions and do not require quantization as used in prior work. By first signaling the keypoint scale level, we can transmit the scaled keypoint positions and rescale them back at the decoder. The scaling factor f from the original image resolution to the resolution of the scale level σ is denoted as

$$f(\sigma) = 1.2^\sigma. \tag{8}$$

In ORB-SLAM2, the default scaling factor is 1.2. The scaling of keypoints from original image coordinates x, y to rescaled integer scale level coordinates x_s, y_s can be written as

$$x_s = \frac{x}{f(\sigma)}, y_s = \frac{y}{f(\sigma)}. \tag{9}$$

This also allows for adapting the number of bits spent for coding the keypoint position according to the image size in the corresponding scale level. The image dimensions $height(\sigma)$ and $width(\sigma)$ per level of the scale pyramid are calculated as

$$height(\sigma) = \frac{N_y}{f(\sigma)}, \quad width(\sigma) = \frac{N_x}{f(\sigma)}, \tag{10}$$

where N_y and N_x denote the original image height and width. The cost for coding the keypoint $\mathbf{k}_{n,i}$ is the sum of the costs for coding σ, $\hat{\theta}$, x_s and y_s:

$$\begin{aligned} R_{n,i}^{I,kpt} = &\log_2(N_\sigma) + \log_2(N_{\hat{\theta}}) \\ &+ \log_2(height(\sigma)) + \log_2(width(\sigma)). \end{aligned} \tag{11}$$

This coding scheme allows for lossless coding of the keypoint position while at the same time lowering the costs for the transmission. In previous work [3], the authors used a quantization step to quarter pixel resolution thereby $\log_2(4 \cdot N_y) + \log_2(4 \cdot N_x)$ bits are required for the keypoint position. In our case, the keypoint orientation is encoded in a lossy manner. We use $N_\sigma = 8, N_{\hat{\theta}} = 32$.

B. Inter-Coding

The inter-frame coding mode makes use of the temporal correlation between successive frames and is similar to [3]. In order to minimize the coding cost, we look for the best matching feature j^* within the previous frames according to the calculated rate. We restrict the search to a window of size $r = 20$ and to neighboring scales as $\Delta\sigma = \pm 1$. We perform a search for the minimum coding costs for the descriptor and the keypoint depending on the reference feature j as

$$j^* = \arg\min_j (R_{n,i}^{P,des}(j) + R_{n,i}^{P,kpt}(j)). \tag{12}$$

The cost for coding a feature using predictive coding is given by the costs for signaling the reference feature, the residual between the descriptors and the differential keypoint between the current and the reference feature as

$$R_{n,i}^P = R_{n,i}^{P,ref} + R_{n,i}^{P,des}(j^*) + R_{n,i}^{P,kpt}(j^*). \tag{13}$$

In contrast to previous work [3], our system also supports multiple reference frames, which allows for an efficient feature coding in the presence of short-term occlusions.

1) Reference Coding:
The number of bits required to signal the reference feature is given by

$$R_{n,i}^{P,ref} = \log_2(N_{ref}^P), \tag{14}$$

where N_{ref}^P is the total number of the features from the reference frames.

2) Descriptor Coding:
Similar to the intra-frame coding mode we can derive the costs for coding the descriptor and the keypoint part. The cost for coding the descriptor using inter-frame coding is defined by the Hamming distance between the current descriptor and the candidate as

$$h_{n,i}^P(j^*) = H(\mathbf{d}_{n,i}, \mathbf{d}_{j^*}), \tag{15}$$

where j^* denotes the best reference feature in the buffer and

$$R_{n,i}^{P,des}(j^*) = R_{res}(h_{n,i}^P(j^*), p_0^P) \tag{16}$$

denotes the lower bound for the amount of bits needed to transmit the residual vector using (5).

3) Keypoint Coding:
In order to efficiently transmit the keypoint, we calculate the difference of all keypoint properties with respect to the reference keypoint. For this coding mode, we train probability tables $p_{\Delta p}$ for the difference in position individually for each scale level. We train an additional table $p_{\Delta\sigma}$ for the difference in scale and a table $p_{\Delta\hat{\theta}}$ for the difference in angle bins. The rate of coding the keypoint difference is then given by

$$\begin{aligned} R_{n,i}^{P,kpt} = &- log_2(p_{\Delta p}(\Delta x_s, \Delta y_s, \sigma_{n,i})) \\ &- log_2(p_{\Delta\sigma}(\Delta\sigma)) - log_2(p_{\Delta\hat{\theta}}(\Delta\hat{\theta})), \end{aligned} \tag{17}$$

where Δx_s and Δy_s denote the difference in x- and y-direction, $\Delta\sigma$ the difference in scale level and $\Delta\hat{\theta}$ the difference in orientation bins. For the keypoint displacement, we scale the difference back to the scale level of the current keypoint to ensure integer accuracy.

$$\begin{aligned} \Delta x_s &= round(\frac{x_{n,i} - x_{j^*}}{f(\sigma_{n,i})}) \\ \Delta y_s &= round(\frac{y_{n,i} - y_{j^*}}{f(\sigma_{n,i})}). \end{aligned} \tag{18}$$

Depending on the settings for the scaling factor and maximum allowed scale difference, this produces also unique solutions for features matched across neighboring scales.

C. Inter-frame Skip Mode

If the Hamming distance between matching features is smaller than a threshold $t_h = 5$ and there is zero motion for the particular feature, we copy the reference feature. This results in a lossy compression but can help to drastically reduce the amount of data if no motion is present in the scenery. The threshold accounts for the feature descriptors being subject to sensor noise and lighting changes, for example from flickering light sources, which causes the descriptor to slightly deviate from frame to frame.

D. Mode Decision

For every feature we can calculate the rate needed for every coding mode and select the mode that achieves minimum rate. All the information is now available to calculate a feature relevance score $r_{n,i}$, sort the features accordingly and start the encoding. After each feature, we check if either the bit budget is exhausted or the time limit is reached. The coding mode for each feature is signaled with two additional bits leaving room for future extensions.

V. Feature Selection

In order to ensure constant time encoding and meeting the bitrate constraints, we have added a feature selection step for the encoding part. For now, we do not change the ORB feature extraction parameters but rather decide based on the feature properties whether a visual feature is useful for the remote visual SLAM system or not. Similar to Francini et al. [22], we have measured various relevance statistics for the features used for map point creation in the visual SLAM system. For this purpose, we have used ORB-SLAM2 to build a map using our coding framework. We then analyzed the feature properties that are observations of map points. We ended up to use the detector response s given by the FAST corner detector [23] used in ORB and the coding mode decision from the feature coding algorithm for the relevance score. A stronger detector response indicates a high contrast corner, which should be more reproducible in different views of the same scene compared to low contrast corners. Inter-coding mode is used for features that are visible in consecutive frames. Intuitively, the more often a corner is seen in different views, the higher the chance of being triangulated to a map point. In order to formulate the problem, we follow the notation of Francini et al. [22] and introduce the indicator $c = 1$ for features being an observation of a map point and $c = 0$ for features not attached to a map point. For the response value, we use a bin size of 20 and quantize the response range $s \in [0, \ldots, 180]$ accordingly. Values exceeding the range are assigned to the highest bin. For the quantized response value $\hat{s} \in [0, N_{\hat{s}}]$, we can calculate the probability of correct matches using Bayes formulation

$$P(c = 1|\hat{s} = \hat{s}_{n,i}) = \frac{P(c = 1 \cap \hat{s} = \hat{s}_{n,i})}{P(\hat{s} = \hat{s}_{n,i})}. \tag{19}$$

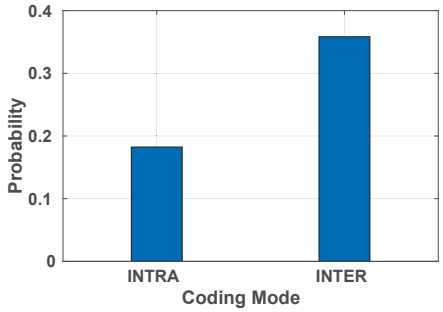
For the coding mode, we distinguish only between the intra and inter-frame coding modes:

$$P(c = 1|m = m_{n,i}) = \frac{P(c = 1 \cap m = m_{n,i})}{P(m = m_{n,i})}. \tag{20}$$

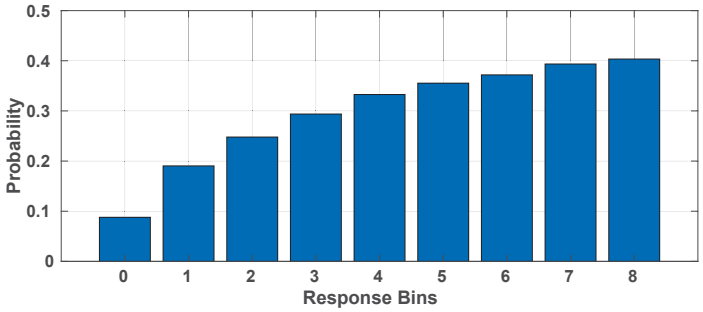
The resulting statistics are shown in Figure 3. For now, we assume both properties to be independent. This results in a relevance score obtained by a multiplication of the individual scores

$$r_{n,i} = P(c = 1|\hat{s} = \hat{s}_{n,i}) \cdot P(c = 1|m = m_{n,i}). \tag{21}$$

Based on these criteria, we can calculate a score for every feature in the current image and transmit only relevant



(a) Triangulation probability given the coding mode.



(b) Triangulation probability given the detector response.

Fig. 3. Triangulation probability used for feature selection based on a map of the EuRoC MH01 sequence.

features up to a certain time limit or bit budget. We also considered the keypoint orientation $\hat{\theta}$ and the scale σ as a feature property. While the orientation did not provide additional information, we discovered that higher scale levels show a higher triangulation probability, but the prioritization of certain levels reduces scale invariance leaving ORB-SLAM2 unable to keep tracking. In contrast, the authors of ORB-SLAM2 specifically try to distribute the feature across the image and the scale space to make the tracking more scale invariant [8].

A. Feedback Channel

A reliable initialization is crucial for a visual SLAM system. If the initial map is erroneous, the subsequent tracking has little chance to estimate the correct frame position. The authors of ORB-SLAM2 double the number of ORB features for the frames in the uninitialized monocular case. They use only the features extracted at the first level of the scale pyramid to estimate the initial transformation between two frames. While the features detected at other scale levels are also stored and can later be used for creating map points and loop closing, we try to avoid sending twice the number for initialization and therefore keep the number constant and prioritize the features at the first scale level during the initialization phase. In order to signal when to return to the usual mode decision, we have added a feedback channel where information about the current tracking state is fed back to the sender. More specifically, we signal when the initialization is completed or when tracking is lost. We then take appropriate action such as sending more features from the first scale level.

VI. Experimental Evaluation

For the intra-coding scheme, we extracted ORB features from the MIRFlickr 1M [24] dataset as training data. We trained a vocabulary using an hierarchical implementation of Bag-of-Words for binary descriptors, namely DBoW2 [25], with branching factor $b = 10$ and depth $l = 5$, using the descriptors from the first 100.000 images of the dataset. We refrain from using the visual vocabulary included in ORB-SLAM2, which was created with $b = 10$ and $l = 6$ due to size constraints. Our goal is to run on embedded devices

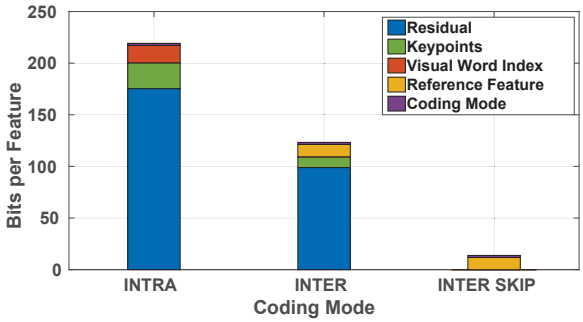


Fig. 4. Bits per feature for all coding modes using 4 reference frames.

TABLE I
Timing and Rate for fr3/long_office (Median)

	INTRA-only	INTER r=1	INTER r=4
Encoding (ms)	19.41	21.02	26.32
Decoding (ms)	25.41	23.03	23.61
kbits/frame	219.20	161.10	140.56

Timing results are the median over ten executions using 1000 features for varying numbers of reference frames r.

with limited memory footprint. Therefore, we opted in for the smaller vocabulary taking only 14.8 MB compared to 145.3 MB of the original vocabulary. We have trained the probabilities p_0^m using the machine hall (MH) sequences from the EuRoC MAV dataset [26]. The properties for the feature selection have been obtained from a map of the MH01 sequence. For the evaluation, we have used the TUM RGB-D [27] dataset consisting of sequences with a resolution of 640x480 and 30 fps using the monocular version of ORB-SLAM2. For all experiments, we employed the default settings provided by ORB-SLAM2 extracting 1000 features per frame. We ran both encoder and decoder simultaneously on a single computer equipped with a core i7-3770 processor and 16 GB RAM and using the Robotic Operation System (ROS) [28] for communication. First, we show the feature size of the individual coding modes in Figure 4. For the intra-coding mode we spend on average 175.36 bits for the descriptor coding, 24.93 bits for the keypoint properties and

16.91 bits for the visual word. Compared to previous work [4], which uses the quantization-based keypoint position coding, we could reduce the number of bits from 223.33 to 217.20 using our new lossless keypoint coding approach exploiting the scale pyramid. We signal the mode with 2 additional bits, resulting in 219.20 bits per feature for the intra-coding. For the inter-coding, we use 11.97 bits for coding reference features using 4 reference frames, 98.88 bits for the descriptor and 10.42 bits for the keypoint differences, resulting in 123.27 bits per feature including the coding mode. The inter-skip mode only needs to send the reference feature index and the coding mode. We compare the resulting bitrate and coding times for different number of reference frames in Table I. When using four reference frames, we can achieve a reduction of 20.54 kbits/frame compared to a single reference frame resulting in 64.12% of the size of intra-mode only. Next, we have evaluated the effectiveness of our feature selection step by limiting the available bit budget to 75 kbits/frame. The SLAM performance is measured in terms of absolute trajectory RMSE error and is shown in Table II. As monocular SLAM does not provide any scale, the resulting trajectory has been aligned and scaled with the ground truth. Although we have drastically reduced the bitrate to 75 kbits/frame, which is 20.83% of the uncompressed size of 360 kbits/frame (assuming 1000 features per frame and 360 bits per feature [4]), we are able to process the sequences with only slight loss in accuracy. In order to provide more insight on the effects of feature selection, we show in Figure 5 the bitrate profile, the map points used for tracking, the number of keyframes and the number of map points contained in the map at each frame of the *fr3/long_office* sequence. We compare our feature selection to random sampling by using 500 out of 1000 features. By favoring features with both strong responses and inter-coding as the selected coding mode, we optimize for tracking stability and bitrate at the same time. An indicator for the tracking stability is the number of map points that have been tracked in the current frame. The more map points are used for estimating the current frame pose, the more reliable is the result. In addition, the number of both keyframes and map points in the map can be significantly reduced by using features that can be tracked very well. This in return reduces the memory footprint of the map and furthermore reduces the computational burden for ORB-SLAM2, as shown in Table III. We are able to reduce the median time to track a single frame to 11.63 ms using only a subset of features compared to 18.80 ms using 1000 features. Although the time required for the sender and receiver for sequential processing using 1000 features exceeds the time between frames of 33.33 ms at 30 fps, we can achieve real-time capabilities by running the encoding of the last frame and the ORB extraction of the current frame in parallel. We performed additional tests on an embedded platform using an NVIDIA Jetson TX2. Without any specific optimization, we were able to encode 1000 features within 34.2 ms using one and 50.4 ms using four reference frames at 1.4 GHz while using roughly 5.5 watt. At 2.0 Ghz it took 25.3 ms and 34.6 ms respectively while using about 6.7 watt. Until now, we use parallelization only for the mode decision leaving room for further improvement.

TABLE II

COMPARISON OF ACCURACY RMSE (CM) FOR TUM RGB-D

Sequence	75 kbits/frame	1K features/frame
fr2/desk	1.18	**1.08**
fr2/desk_person	1.35	**1.18**
fr3/long_office	1.35	**1.23**
fr3/nostr_tex_near_wl	1.57	**1.40**
fr3/sit_halfsphere	1.91	**1.81**
fr3/str_tex_far	**1.11**	1.14
fr3/str_tex_near	1.37	**1.31**

Results are the median values over 10 executions for each sequence. The trajectories have been aligned and scaled to the ground truth using 7DoF alignment by similarity transformation using the provided dataset tools.

TABLE III

TIMING RESULTS FOR FR3/LONG_OFFICE IN MILLISECONDS (MEDIAN)

		75 kbits/frame	1K features/frame
Sender	ORB	13.50	
	Encoding	17.38	26.32
	Total	31.36	40.47
Receiver	Decoding	13.40	23.61
	Tracking	11.63	18.80
	Total	25.39	43.85

Timing results are the median over 10 executions. The number of reference frames was set to 4.

VII. CONCLUSIONS

In this paper, we propose a framework for efficient compression of local binary features tailored for monocular remote visual SLAM applications. By using an efficient probabilistic feature selection process, we are able to reduce the required bitrate to 75 kbits/frame, which results in 20.83% of the uncompressed size while being able to run ORB-SLAM2 with only slight degradation in accuracy. In addition, we are able to reduce the computation time of ORB-SLAM2 and the resulting map size by only using selected features. Future directions of research include a stereo-view coding mode and collaborative mapping. Supplementary material is available online.

ACKNOWLEDGMENT

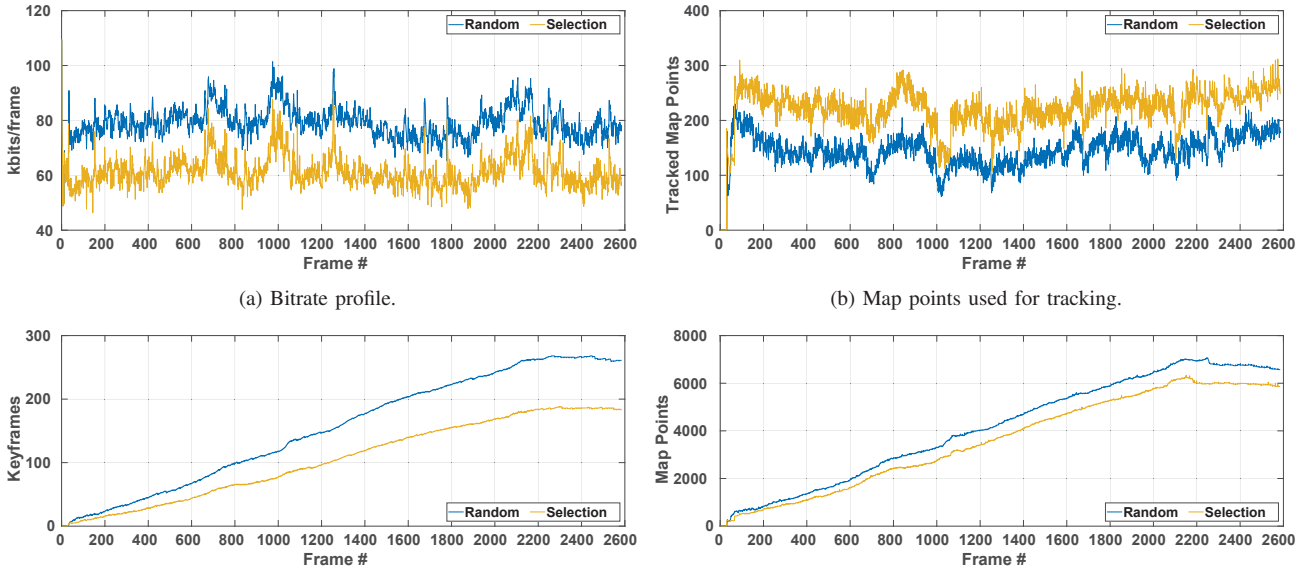
This work is supported by the space agency of the German Aerospace Center with funds from the Federal Ministry of Economics and Technology on the basis of a resolution of the German Bundestag under the reference 50NA1515.

REFERENCES

[1] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2564–2571.

[2] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. PP, no. 99, pp. 1 – 8, 2017.

`https://rebrand.ly/icra18`



(a) Bitrate profile.

(b) Map points used for tracking.

(c) Evolution of the number of keyframes in the map.

(d) Evolution of the number of map points in the map.

Fig. 5. Comparison of map properties using a random selection of 500 features and our proposed feature selection scheme on the *fr3/long_office* sequence.

[3] L. Baroffio, A. Canclini, M. Cesana, A. Redondi, M. Tagliasacchi, and S. Tubaro, "Coding Local and Global Binary Visual Features Extracted from Video Sequences," *IEEE Transactions on Image Processing*, vol. 24, no. 11, pp. 3546–3560, 2015.

[4] D. V. Opdenbosch, M. Oelsch, A. Garcea, and E. Steinbach, "A Joint Compression Scheme for Local Binary Feature Descriptors and their Corresponding Bag-of-Words Representation," in *IEEE Conference on Visual Communications and Image Processing (VCIP)*, 2017.

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007, pp. 225–234.

[6] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2320–2327.

[7] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Direct Monocular SLAM," *European Conference on Computer Vision (ECCV)*, vol. 8690, pp. 834–849, 2014.

[8] 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.

[9] J. Martinez-Carranza, N. Loewen, F. Marquez, E. O. García, and W. Mayol-Cuevas, "Towards autonomous flight of micro aerial vehicles using ORB-SLAM," in *Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, 2015.

[10] L. Riazuelo, J. Civera, and J. M. Montiel, "C2TAM: A Cloud framework for cooperative tracking and mapping," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 401–413, 2014.

[11] P. Schmuck, "Multi-UAV collaborative monocular SLAM," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3863–3870.

[12] J. Chao, H. Chen, and E. Steinbach, "On the design of a novel JPEG quantization table for improved feature detection performance," in *IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 1675 – 1679.

[13] J. Chao, R. Huitl, E. Steinbach, and D. Schroeder, "A Novel Rate Control Framework for SIFT/SURF Feature Preservation in H.264/AVC Video Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, pp. 958–972, 2015.

[14] J. Chao and E. Steinbach, "Keypoint Encoding for Improved Feature Extraction From Compressed Video at Low Bitrates," *IEEE Transactions on Multimedia*, vol. 18, no. 1, pp. 25–39, 2016.

[15] L.-y. Duan, V. Chandrasekhar, J. Chen, J. Lin, Z. Wang, T. Huang, B. Girod, and W. Gao, "Overview of the MPEG-CDVS Standard," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 179–194, 2016.

[16] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[17] S. S. Tsai, D. Chen, G. Takacs, V. Chandrasekhar, J. P. Singh, and B. Girod, "Location coding for mobile image retrieval," in *Mobile Multimedia Communications Conference (ICST)*, 2009, pp. 1–7.

[18] S. S. Tsai, D. Chen, and G. Takacs, "Improved Coding for Image Feature Location Information," *Proc. SPIE*, vol. 8499, 2012.

[19] 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.

[20] L. Baroffio, M. Cesana, A. Redondi, M. Tagliasacchi, and S. Tubaro, "Coding visual features extracted from video sequences," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2262–2276, 2014.

[21] A. Redondi, L. Baroffio, J. Ascenso, M. Cesana, and M. Tagliasacchi, "Rate-accuracy optimization of binary descriptors," in *IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 2910 – 2914.

[22] G. Francini, S. Lepsoy, and M. Balestri, "Selection of local features for visual search," *Signal Processing: Image Communication*, vol. 28, no. 4, pp. 311–322, 2013.

[23] E. Rosten and T. Drummond, "Machine Learning for High-speed Corner Detection," in *European Conference on Computer Vision (ECCV)*, 2006, pp. 430–443.

[24] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *ACM International Conference on Multimedia Information Retrieval*, vol. 4, no. November, 2008, pp. 39–43.

[25] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[26] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Y. Siegwart, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1–7, 2016.

[27] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 573–580.

[28] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: An open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.