2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Enhancing V-SLAM Keyframe Selection with an Efficient ConvNet for Semantic Analysis

Iñigo Alonso[1] Luis Riazuelo[1] Ana C. Murillo[1]

***Abstract*— Selecting relevant visual information from a video is a challenging task on its own and even more in robotics, due to strong computational restrictions. This work proposes a novel keyframe selection strategy based on image quality and semantic information, which boosts strategies currently used in Visual-SLAM (V-SLAM). Commonly used V-SLAM methods select keyframes based only on relative displacements and amount of tracked feature points. Our strategy to select more carefully these keyframes allows the robotic systems to make better use of them. With minimal computational cost, we show that our selection includes more relevant keyframes, which are useful for additional posterior recognition tasks, without penalizing the existing ones, mainly place recognition. A key ingredient is our novel CNN architecture to run a quick semantic image analysis at the onboard CPU of the robot. It provides sufficient accuracy significantly faster than related works. We demonstrate our hypothesis with several public datasets with challenging robotic data.**

I. Introduction

Visual SLAM is an essential task running in the back-end of many robotic systems, but the mapping itself is often not the final goal on the robot missions. In recent years, it is more and more common to have robots or teams of robots communicating with a central station or the rest of the team members to achieve more sophisticated or high-level tasks.

Our work explores the possibilities of selecting more carefully the keyframes that the V-SLAM uses for mapping and place recognition, in order to be able to re-use them for additional posterior tasks. This way, we enable more efficient use of those keyframes that need to be stored and probably transmitted. Currently, additional data would need to be used for those posterior tasks, such as recognition of objects or elements of interest in the mapped environment.

State-of-the-art approaches for visual recognition tasks have witnessed a significant boost and outstanding performances lately thanks, among other reasons, to deep learning based solutions. There have been only a few years since Convolutional Neural Networks (CNNs) caught significant attention [17] and have already been adopted for numerous commercial products. Although CNN models inference time is very short compared to the training time, it is usually required to have high-end GPU(s) to run inferences in near real-time. Unfortunately, these GPUs are often not available in robotic platforms, which present restrictions incompatible with the use of high-end GPUs, such as small robots or drones that cannot hold the extra weight or afford the extra power consumption.

[1] I. Alonso, L. Riazuelo and A.C. Murillo are at DIIS - I3A, Universidad de Zaragoza, Spain. {inigo, riazuelo, acm}@unizar.es

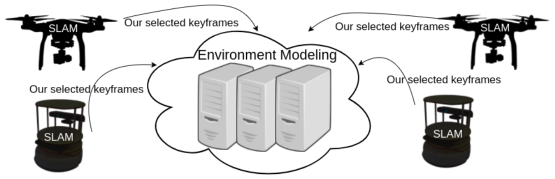


Fig. 1. Our approach runs a smart keyframe selection at each robot onboard CPU. Selected keyframes are stored and/or shared across the system, typically for further more complex processing.

That is why in the last years, we see an increasing interest on Deep Learning solutions for real-time applications on low-power GPUs [10], [23], [27]. They get results close to the state-of-the-art at much lower computational and energy cost, as discussed in more detail later. Nevertheless, even these architectures cannot run on CPUs with the required execution times, although still CPU is the only computing source available for many robots.

This work presents a novel and efficient strategy to include additional criteria to commonly used V-SLAM keyframe selection. Our contribution is twofold:

- A novel strategy for more meaningful keyframe selection, while a robot is mapping its environment, which runs efficiently on the robot CPU.
- A new CNN architecture for semantic segmentation (MiniNet) developed to be able to run on the robot CPU and serve as quick semantic filtering of frames.

Our approach obtains more representative keyframes with little extra cost and, by re-using the keyframes for multiple tasks, avoids extra computations or communications. This is particularly relevant in multi-robot settings where computation and communications bottlenecks are critical. Multi-robot teams often have several nodes with heterogeneous computational capabilities, as illustrated in Fig. 1. They present scenarios where efficiently selecting the most representative information is important to minimize the amount of information shared. For example, the well-known DARPA Subterranean (SubT) Challenge[1], presents a real use-case where communication restrictions are very strong, therefore selecting carefully what to transmit is critical.

The proposed MiniNet gets comparable results to state-of-the-art on segmentation tasks with few classes. It can run onboard the robot CPU to perform tasks such as the presented semantic based keyframe selection, but other applications could benefit from this architecture. The keyframe selection

[1] https://www.subtchallenge.com/

proposed is shown to pick more representative information for high-level recognition tasks (text reading), without losing more basic navigation information (relevant information for place recognition).

II. Related Work

The most relevant literature to our contributions is related to selecting relevant keyframes within sequences and to efficient convolutional neural network architectures.

A. Keyframe selection

Selecting the most representative and valuable frames out of a sequence is, in fact, a visual summarization problem. Depending on the problem and scenario, the criteria and the meaning of valuable information differs. For general video summarization, the selection targets the most representative frames which condense all the events of the entire video. As for many other tasks, deep learning based approaches are leading current state of the art, such us applying recurrent methods [35], CNNs for ranking methods [32] or semantic embeddings [34] for summarization tasks.

For more specific applications, such as surveillance, more specific contents need to be selected, and additional restrictions, such as computational resource or execution time, need to be considered [18]. This type of approaches is closer to our goals since these restrictions also affect robotic applications. Mobile robots need to perform several real-time tasks in parallel (e.g., V-SLAM or visual recognition algorithms), and cannot afford to apply heavy techniques such as the ones used in general video summarization. Well-known VS-LAM algorithms, such as ORBSLAM2 [21], need to select keyframes to reduce the data used for tracking and place recognition tasks. When modeling the environment with multi-robot teams [26], [29], these keyframes become also the information shared among the robots but still follow the standard VSLAM selection criteria, even though other nodes with higher capabilities could perform more demanding tasks if we would select more carefully what to share.

B. CNN architectures for low computational environments

Many works lately focus on reducing CNNs memory and computational cost, which directly affects energy consumption and inference time. Some approaches focus on the training phase (e.g., joint training and distillation [25], [9], [28]), others on parameters data type (e.g., quantized [11] or binary [5] networks) or post-processing methods (e.g., pruning [20], [8]) and others on novel architecture operations (e.g. depthwise separable convolutions [14], dilated convolutions [33] and self normalizing neural networks [16]).

Our work is focused on running efficiently semantic segmentation tasks. CNNs for semantic segmentation typically follow an encoder-decoder structure: an encoder which learns features while reducing the resolution and a decoder which upsamples the learned features and maps them into the segmentation result. Recent works towards efficient segmentation architectures inlcude Deeplab-v3 [3], [2] and ERFNet [27], that use atrous convolutions [33] to avoid the need

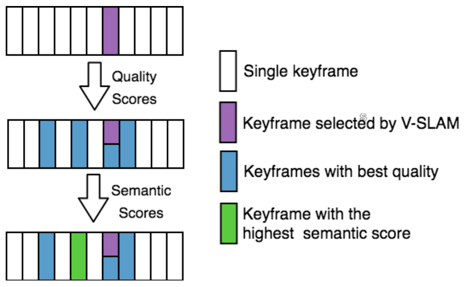

Quality
Scores
Semantic
Scores
Single keyframe
Keyframe selected by V-SLAM
Keyframes with best quality
Keyframe with the highest semantic score

Fig. 2. Keyframe selection proposed. (Top) Define a window of keyframe candidates around each selected keyframe by VSLAM; (middle) compute the quality score to reduce the set; (bottom) pick the top best quality frame according to our semantic score. Best viewed in color.

to reduce much the input resolution. Many architectures targeting efficiency, e.g., ERFNet [27] and ENet [23], perform several consecutive early downsampling operations for a quick reduction of the input resolution and they have light decoders with very few parameters and layers.

Our proposed architecture is inspired by many of these recent works, focused on efficient semantic segmentation tasks but, differently from other works, considering the feasibility of CPU execution. We focus on semantic segmentation because it provides information about the whole image scene (pixel-level semantic labels), essential to have a quick frame content analysis.

III. Efficient semantic based keyframe selection on the robot CPU.

Our proposed approach to select the most relevant keyframes has been designed with two requirements in mind: 1) A versatile framework to combine several scoring functions about the *relevance* of the keyframes for both local robot operations and global team goals. 2) Run at acceptable rates locally on the robot CPU, i.e., around 20-30 fps.

To account for the first requirement, we propose a hierarchical scoring system to select the most relevant keyframes processed during mapping (summarized in Fig. 2). To account for the limited computational resources, we introduce a novel CNN architecture, *MiniNet*, that enables a rough but very fast semantic segmentation on CPU.

A. Keyframe selection algorithm

The core idea of our keyframe selection algorithm is to boost typical VSLAM criteria (select a new keyframe when the established geometric change is reached) by selecting a higher quality and meaningful keyframes for posterior place recognition of other robots, relocalization, and further visual analysis tasks. As Fig. 2 illustrates, our system runs an evaluation on a window around each of the original VSLAM keyframes and selects the overall best combining two criterion:

a) Image quality criterion: we first set a candidate window around each keyframe selected by the VSLAM. We define the *quality* of an image I as a combination of two scores, defined in eq. (1). The first score, blurriness score sc_{BL} defined in eq. (2), is based on the Energy of

Laplacian [24], where ∂I is the Laplacian of I. The higher the value of this score, the more likely to be selected. The second score, brightness score sc_{BR} defined in eq. (3), is based on the total luminance on the image pixels. The image I is converted to LAB colorspace and the L channel values are zero-centered and added. The higher this score, the lower the image quality. To keep the computational cost low, we use a 112×112 resolution. Before combining the two scores, we independently normalize each one dividing by the corresponding maximum value in each candidates window.

$$sc_{quality}(I) = norm(sc_{BL}(I)) * norm(sc_{BR}(I)) \quad (1)$$

$$sc_{BL}(I) = \sum_{(i,j)\in\Omega(x,y)} \partial I(i,j)^2 \quad (2)$$

$$sc_{BR}(I) = \frac{1}{\sum_{(i,j)\in\Omega(x,y)} zero\_center(L(i,j))^2} \quad (3)$$

b) Semantic content criterion: the second part of the selection algorithm focuses on the image semantic content which may be relevant for the high-level tasks to be performed. This step only evaluates the Q frames with higher $sc_{quality}$ score and computes the semantic score for each of them. As a concrete use case to demonstrate this step, let us think of a system focused on finding textual information in the environment. However, note that the proposed *MiniNet* for quick semantic filtering, detailed in next subsection III-B, can be fine-tuned for different target semantic classes.

The proposed semantic score is based on a rough semantic segmentation, achieved efficiently by the proposed *MiniNet*. This score, eq. (4), is computed as the ratio of image pixels that belong to the target class, penalizing the ratio of pixels from the target class which lay on the image border. This penalizes images where the target objects are very likely to be only partially visible, e.g., if a text region is next to the border, it is likely to have only half of a sign visible:

$$sc_{semantic}(I) = \frac{\sum_{(i,j)\in\Omega(x,y)} Text(i,j)}{1 + (\sum_{(i,j)\in\Omega(x,y)} Text_{border}(i,j))}, \quad (4)$$

where $Text(i,j)$ is the text segmentation value of image pixel i,j (1 for text pixels, 0 otherwise). Same values for $Text_{border}(i,j)$, text on image borders.

B. MiniNet network architecture

The proposed architecture for semantic segmentation[2] is designed to efficiently run on CPU, which increases the applicability of CNNs for robotic tasks with execution time restrictions. In this work, MiniNet is used to build the $sc_{semantic}$ score, eq. (4). However, we should note that it could be beneficial on its own for many other visual tasks run on restricted robotic platforms, independently of the use of a VSLAM algorithm or not. *MiniNet* architecture is inspired by several prior works on CNNs for low computation

[2]Link to the official available implementation: https://github.com/Shathe/MiniNet.

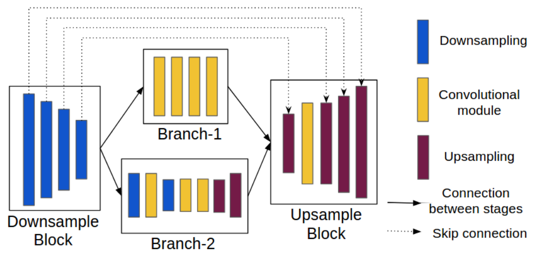


Fig. 3. MiniNet architecture diagram. See Table I for further detail.

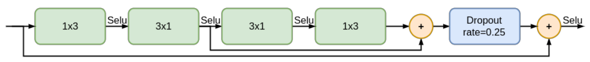


Fig. 4. *Convolutional module*: Four separable convolutions with two residual connections. Lastly a dropout layer is applied to help dealing with overfitting.

TABLE I

MiniNet ARCHITECTURE. IT HAS FOUR MAIN BLOCKS: DOWNSAMPLE, TWO CONVOLUTIONAL BRANCHES AND UPSAMPLE.

	Name	Type	Input	Output size
Downsample	d1	downsampling	image	256x128x12
	d2	downsampling	d1	128x64x24
	d3	downsampling	d2	64x32x48
	d4	downsampling	d3	32x16x96
Branch-1	branch_1_1	module rate=1	d4	32x16x96
	branch_1_2	module rate=2	branch_1_1	32x16x96
	branch_1_3	module rate=4	branch_1_2	32x16x96
	branch_1_4	module rate=8	branch_1_3	32x16x96
Branch-2	d5	downsampling	d4	16x8x192
	branch_2_1	module rate=1	d5	16x8x192
	d6	downsampling	branch_2_1	8x4x386
	branch_2_2	module rate=1	d6	8x4x386
	branch_2_3	module rate=1	branch_2_2	8x4x386
	up1	upsampling	branch_2_3	16x8x192
	branch_2_4	module rate=1	up1	16x8x192
	up2	upsampling	branch_2_4, d5	32x16x96
Upsample	up3	upsampling	branch_1_4, up2, d4	64x32x96
	module_up	module rate=1	up3	64x32x48
	up4	upsampling	module_up, d3	128x64x24
	up5	upsampling	up4, d2	256x128x12
	output	upsampling	up5 , d1	512x256xN

environments, as discussed in Sec. II, in particular ERFNet [27] and ENet [23], with the particularity that our work takes into account the best options for execution on CPU.

The *MiniNet* blocks (detailed in Figure 3 and Table I) are the following:

a) Downsampling block: the purpose of this block is to reduce the resolution to a reasonable one on which to perform thorough feature extraction. The input resolution is 512x256 which is a reasonable input size compared to the state-of-the-art[23], [13], [3]. Our downsample operation performs a depth-wise separable convolution with a stride of 2x2. The proposed architecture performs four downsample operations, leading to a 32x16 resolution.

b) Two convolutional branches: the network is split into two parallel convolutional branches. These branches use our *convolutional module* (see Fig. 4). This module is based on the ERFNet Non-bottleneck-1D module [27]. Our main modifications are:

- Include a sum operation between the two decomposed convolutions to conserve the output of the intermediate convolution.

• Remove the Batch Normalization [12] (and Relu [22]) and adding in stead Selu activations, i.e., self normalizing neural networks [16], gaining a $\times 2$ of speed in CPU.

• Change standard convolutions for separable convolutions. Performing depth-wise separable instead of standard convolutions reduces around 2-3 times the computation [10].

• Instead of using the standard dropout, we perform the alpha dropout [16].

The *branch-1 block* consists of four consecutive *convolutional modules* with different dilatation rates. This branch performs parameter-efficient feature extraction on a higher resolution (32x16) than the other branch thanks to dilated convolutions. We cannot afford to add more than four convolutional modules at this resolution for real-time performance on CPU. This fact shows the differences between CPU and low-powered GPUs, where 10-20 modules of this type can be processed and even at higher resolutions.

The *branch-2 block* follows the regular encoder architecture with no-dilated convolutions working on a tiny resolution. This branch plays a very important role in this architecture allowing more time-efficient feature extraction. This branch consists of applying several downsampling and convolutional modules up to 8x4 resolution performing the feature extraction and then, upsampled the features up to the initial size (32x16).

c) Decoder block: our upsample operation consists of a transposed convolution (kernel 3x3 and stride of 2x2). The two convolutional branches are concatenated, upsampled and applied a convolutional module. Then, we concatenate the features with skip connections from the downsample part and perform an upsample operation. This is repeated until getting the initial 512x256 resolution.

IV. Experiments

This section validates the effectiveness of the keyframe selection algorithm presented and evaluates the performance of the proposed *MiniNet*.

A. MiniNet performance and suitability

The following two experiments compare its performance to state-of-the-art CNNs on common segmentation benchmarks, detailed later in each experiment. The first one, Cityscapes, is a more general multi-class segmentation benchmark, in order to have a general evaluation of the reach of *MiniNet*. The second one, COCO-Text, is a more specific binary-segmentation benchmark, to evaluate the network on the more specific type of tasks expected to be part of the keyframe selection proposed.

Training details: MiniNet has been trained for 90K iterations on the Cityscapes dataset and for 20K iterations on the COCO-Text dataset using a batch size of 32. We use Adam optimizer [15] with initial learning rate of $n = 10^{-3}$ and polynomial learning rate decay. We optimize it through the cross-entropy loss function commonly used to train segmentation models. To account for class imbalance,

TABLE II
SEGMENTATION RESULTS ON CITYSCAPES ONLINE BENCHMARK

	CityScapes (19-classes) Metrics*			
	Cla-**IoU**	*Cat*-**IoU**	**GPU** (s)	**CPU** (s)
DeeplabV3+ [3]	82.1	92.0	0.512	14.392
ERFNet [27]	69.7	87.3	0.024	0.571
ENet [23]	58.3	80.4	0.013	n/a
RTSeg [30]	58.3	80.2	n/a	n/a
MiniNet (ours)	40.7	70.5	**0.004**	**0.018**

**Cla* = Class; *Cat* = Category; IoU = Intersection over Union metric
**GPU* = forward pass time on Titan X; *CPU* = forward pass time on Intel i5-8600k

TABLE III
SEGMENTATION RESULTS ON COCO-TEXT

	Text Segmentation (binary) Metrics*					
	GPU(s)	**CPU**(s)	GFlops	**R**	**P**	**IoU**
DeeplabV3+[3]	0.512	14.392	102.85	58.85	43.90	33.29
ERFNet [27]	0.024	0.571	55.21	52.66	39.73	29.27
MiniNet (ours)	**0.004**	**0.018**	**1.06**	52.61	36.69	27.63

* *R* = Recall; *P* = Precision; IoU = Intersection over Union metric
**GPU* = forward pass time on Titan X; *CPU* = forward pass time on Intel i5-8600k

we use the median frequency class balancing, as applied in SegNet [1]. To smooth the resulting class weights, we propose to apply a power operation: $w_c = (\frac{f_{median}}{f_c})^i$, with f_c being the frequency of class c and f_{median} the median of all frequencies.

The image augmentation applied in all experiments consists of left-right flips, small spatial shifts and scales (up to 10%) and small contrast normalization (α between 0.90 and 1.20). Additionally, for the binary segmentation experiment, we include *black-and-white* augmentation, i.e., randomly converting the RGB image into a grayscale one. This helps to generalize better to gray-scale test images (very common in robotics).

1) Multi-class segmentation experiment: This first experiment compares the state-of-the-art with *MiniNet* results on the Cityscapes dataset [4], an urban scene dataset commonly used to evaluate semantic segmentation approaches. This evaluation is done automatically on the dataset official benchmark site by submitting the test predictions. Table II shows the performance of our approach, using the public benchmark metrics, for the most relevant methods to our work published on that site: Deeplabv3+ is currently the overall state-of-the-art, while ERFNet and ENet are the best on low-power GPUs considering the trade-off between performance and speed. If available, we report the execution times for GPU. For CPU times, we have computed the mean of 5 executions (ran with the authors' available code). Note that we were not able to run ENet on CPU due to the lack of CPU implementation of the required operation *MaxPooling with argmax*). *MiniNet* is able to run $\times 3$ faster than ENet on GPU, but gets 18% less Cla-IoU and 10% less Cat-IoU. Thus, as far as GPU is concerned, the trade-off between speedup and loss of IoU seems proportional in both cases. However, concerning the CPU time performance, note that MiniNet is over $\times 30$ times faster than ERFNet (while in GPU is $\times 6$ times faster).

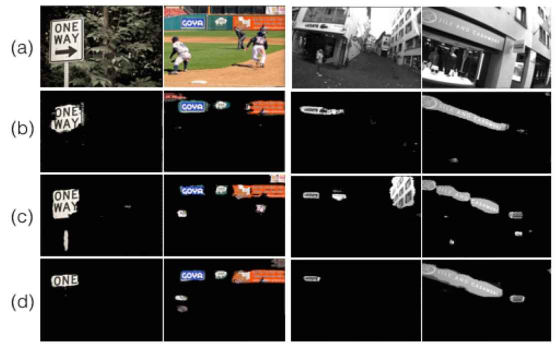


Fig. 5. Segmentation from COCO-Text (left) and V4RL data (rigth). (a) input image, (b) MiniNet, (c) Deeplabv3+ and (d) ERFNet segmentations.

These results confirm the proposed architecture gets reasonable accuracy in general tasks while achieving much faster execution, especially in CPU, which is particularly important for our goals: quickly filtering images on each robot to select what's worth sharing for further processing.

2) Binary segmentation experiment: This experiment focuses on *text segmentation*, which may seem an easier task than the previous experiment, but it is closer to the type of quick filtering tasks that *MiniNet* is designed to work with. As we analyze in the next section, a use case of the keyframe selection strategy proposed in this work is to quickly filter keyframes on the robot where text regions seem more significant to facilitate further text reading tasks on the selected frames. For this experiment, we use the well known COCO-Text dataset [31] (a subset of machine printed and legible text images). Text in this dataset is labeled for text detection with bounding boxes, but we use them as approximated per-pixel annotations for our segmentation results. For this experiment, we trained from scratch the three architectures, i.e., MiniNet, Deeplabv3+ (a top-performing generic semantic segmentation approach) and ERFNet (a state-of-the-art for low-power GPUs that can also run on CPU) with the same configuration previously described.

Table III shows the performance of our approach and the other well-known architectures that we have trained on the same setup. The only difference is the image input resolution (which indeed affects directly the execution time), we show performance results with the resolution reported by the authors on their prior work: ERFNet 1024x512, Deeplabv3+ 512x512 and we set *MiniNet* to use 512x256. To enable more direct comparisons, note some of the relevant variations we have run and measured: ERFNet with 512x256 input takes 0.21 on CPU ($\times 8$ than *MiniNet*) and 0.008 on GPU ($\times 2$). ERFNet would need a 96x48 input to take the same time that *MiniNet* CPU forward pass.

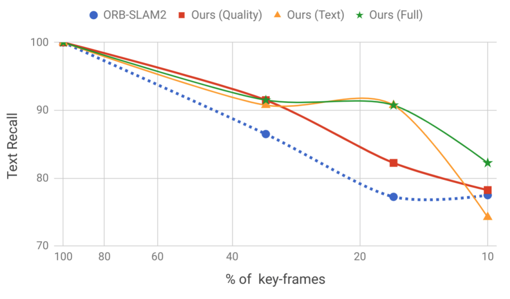
Differently, from the previous experiment, the text detection quality metrics for *MiniNet* are much closer to the other approaches, and still present a huge CPU speed-up. This demonstrates that on this type of task *MiniNet* is able to get similar results than state-of-the-art architectures for low-power GPUs while running x6-7 times faster on GPU and x30-60 times faster on CPU. Besides, the segmentation examples in Fig. 5, qualitatively confirm that MiniNet finds text regions similarly to state-of-the-art approaches. There are visual results with COCO-TEXT images as well as images from another public dataset: V4RL Urban Place Recognition Dataset [19], a challenging drone image dataset. It contains data from two drones mapping the environment and serves as a realistic robotic use case to evaluate the full system proposed in the next section. V4RL images where segmented with the CNNs trained on COCO-TEXT, without any adaptation on the model or the grayscale images.

B. Keyframe selection

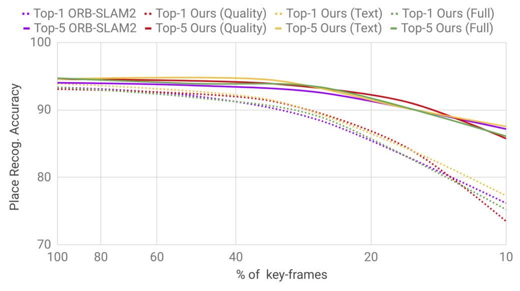
To evaluate the proposed keyframe selection we compare the relevance of the keyframes selected with it and with a state-of-the-art VSLAM algorithm, ORB-SLAM2. Typically VSLAM systems select keyframes online to perform the camera localization and store most of them to enable loop-detection/place-recognition tasks. We demonstrate how our keyframe selection method is fast enough to replace those selection strategies while it selects more relevant keyframes for additional high-level tasks to be performed on a robot team. We evaluate aspects of relevant information for place recognition, additional recognition tasks and quality of the selected images.

1) Set-up: We use the V4RL Urban Place Recognition dataset [19], with outdoor data recorded for VSLAM and place recognition applications. The configuration parameters from Sec. III-A are set as follows. The candidate window size is set as half the distance between the last keyframe selected by our system and current keyframe selected by ORB-SLAM2. The window is placed in such a way that there are twice as many elements before the original keyframe than after. The number of selected top-Q frames, sorted according to the quality score, is 1/3 of the window size. These will then be processed on a batch through the segmentation CNN. With this configuration, the average cost per frame is just 7 ms on Intel i5-8600k and 16ms on the Intel NUC (i5-6260U). As an overview, the execution time (Intel Core i5-8600k) of each keyframe selection step for one image is: *Resize* 5ms, *Blurriness* score 0.1 ms, *Brightness* score 0.1 ms and *Semantic* score 18 ms.

2) Place recognition: In these experiments, we evaluate the place recognition performance, i.e., the capability to relocate in an environment previously visited, since is an essential VSLAM capability that depends on the selected keyframes. We use the ground truth from V4RL dataset and evaluate the accuracy for place recognition. We use the DBoW2 [6] and match test query frames (from one of the drone sequences) to keyframes selected by ORB-SLAM2 algorithm or by our approach in the reference dataset (the other drone sequence, acquired at different day and time). Note we do not run a complete loop closure accuracy evaluation, but we focus on the semantic content of the keyframes. Therefore we provide accuracy of the top1 and top5 results provided by the DBoW2 algorithm. Fig. 6(b) shows that our proposal for selecting the keyframes does not lose any information with respect to the keyframes



(a)



(b)

Fig. 6. Representativity of keyframe selection strategies. (a) Average text (words) recognition recall with different % of keyframes. (b) Average place recognition accuracy (top1 and top5) obtained running DBoW2 algorithm with the different sets of selected keyframes.

selected by ORB-SLAM2. Accuracy decreases equally for both approaches when the number of selected keyframes is reduced to less than 20% of the total amount of keyframes originally selected by the standard V-SLAM algorithm.

3) Text Recognition: This experiment shows that, besides maintaining performance in original tasks, our selected keyframes are more representative and useful for additional tasks. Since the V4RL dataset does not have any semantic label, we have manually labeled visible words in *Shopping Street 1 Sequence 1*, from four different intervals of 200 frames each: frames 50 to 250, 1250 to 1450, 3650 to 3850 and 6760 to 6960. Fig. 6 (a) shows the average recall of words correctly found running the text recognition from Gupta et al. [7] on keyframes selected by ORB-SLAM2 and by our approach. It shows separated results for each individual component of our score (*quality* and *semantic*) to verify the contribution of each to the final result (*Full*). The same text reading algorithm finds more information on our keyframes, regardless of the density of keyframes stored. This demonstrates our approach is more effective in selecting the data to share with the system. We would save computation and network resources by using the same keyframes to perform multiple tasks.

4) Qualitative results: Our approach capabilities to enhance keyframe selection are highlighted in Fig. 7 (more examples available on the supplementary video). The first example shows how a better contrast frame is selected, which was just 7 frames far from the keyframe picked by ORB-SLAM2. Selecting this affects positively to the posterior image analysis. The second example shows the effect of our semantic score based on *MiniNet*: a frame containing two full

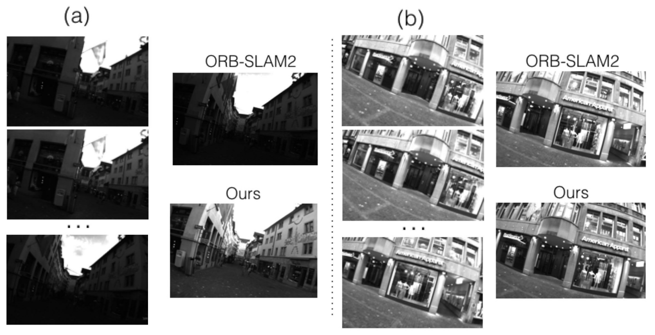


Fig. 7. Keyframes selected by our approach and ORB-SLAM2. (a) Example with strong illumination changes, our approach picks keyframe with better contrast. (b) Example with large text-signs, our approach picks a keyframe with fully visible signs.

shop signs is selected, as opposed to the keyframe picked by ORB-SLAM2, with one of them partially occluded.

V. Conclusion

We have presented a novel keyframe selection which can be integrated with state-of-the-art VSLAM systems to boost the usefulness of the keyframes. The benefits of our approach are particularly relevant for multi-robot systems or robots connected to a remote station since the proposed strategy allows the robots to be more efficient about what is shared with the team. The keyframes are selected considering multiple goals instead of purely based on the VSLAM criteria, which is what is commonly done in multi-robot mapping systems. Our experimentation with challenging drone imagery has shown that the proposed keyframe selection is more useful the ORB-SLAM2 selection strategy. Evaluating the shared keyframes in the GPU-enabled server, we get better performance on additional tasks, text recognition in our experiments, while we do not lose the capacity of recognizing revisited places using those keyframes, essential for VSLAM systems.

A key ingredient in our approach is the efficient CNN proposed for image segmentation, which analyzes the frames online at the robot onboard CPU. This efficiency is essential to incorporate our selection algorithm without penalizing the other tasks run on the robot. Our experiments cover an in-depth analysis of the proposed CNN architecture, *MiniNet*. The good results with the presented MiniNet open opportunities for additional applications based on quick video processing on low-resource nodes. Our future steps include training models for additional use cases with different semantic filters, as well as full integration of the presented selection with multi-robot semantic mapping systems.

VI. Acknowledgements

The authors would like to thank NVIDIA Corporation for the donation of a Titan Xp GPU used in this work. This research has been partially funded by the Spanish Government projects DPI2015-69376-R and DPI2016-76676-R-AEI/FEDER-UE, and Aragón regional government (DGA T45_17R/FSE).

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[2] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[3] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint:1802.02611*, 2018.

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of IEEE conf. on CVPR*, pages 3213–3223, 2016.

[5] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, 2015.

[6] D. Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Trans. on Robotics*, 28(5):1188–1197, October 2012.

[7] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *IEEE conf. on CVPR*, 2016.

[8] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[9] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[11] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016.

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *int. conf. on Machine Learning*, 2015.

[13] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *CVPR Workshops*. IEEE, 2017.

[14] L. Kaiser, A. N. Gomez, and F. Chollet. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*, 2017.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pages 972–981, 2017.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[18] G. Lu, Y. Zhou, X. Li, and P. Yan. Unsupervised, efficient and scalable key-frame selection for automatic summarization of surveillance videos. *Multimedia Tools and Applications*, 76(5):6309–6331, 2017.

[19] F. Maffra, Z. Chen, and M. Chli. Viewpoint-tolerant place recognition combining 2d and 3d information for uav navigation. In *ICRA*, 2018.

[20] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *arXiv preprint arXiv:1611.06440*, 2016.

[21] R. Mur-Artal and J. D. Tards. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Trans. on Robotics*, 33(5):1255–1262, Oct 2017.

[22] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. of Int. Conf. on Machine Learning*, 2010.

[23] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[24] S. Pertuz, D. Puig, and M. A. Garcia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415–1432, 2013.

[25] S. Ravi. Projectionnet: Learning efficient on-device deep networks using neural projections. *arXiv preprint arXiv:1708.00630*, 2017.

[26] L. Riazuelo, J. Civera, and J. M. M. Montiel. C2tam: A cloud framework for cooperative tracking and mapping. *Robotics and Autonomous Systems*, 62(4):401 – 413, 2014.

[27] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Trans. on Intelligent Transportation Systems*, 2018.

[28] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[29] P. Schmuck and M. Chli. Multi-uav collaborative monocular slam. In *2017 IEEE int. conf. on Robotics and Automation (ICRA)*, 2017.

[30] M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, and M. Jagersand. Rtseg: Real-time semantic segmentation comparative study. *arXiv preprint arXiv:1803.02758*, 2018.

[31] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.

[32] T. Yao, T. Mei, and Y. Rui. Highlight detection with pairwise deep ranking for first-person video summarization. In *Proc. of IEEE conf. on CVPR*, pages 982–990, 2016.

[33] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[34] Y. Yuan, T. Mei, P. Cui, and W. Zhu. Video summarization by learning deep side semantic embedding. *IEEE Trans. on Circuits and Systems for Video Technology*, 2017.

[35] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Video summarization with long short-term memory. In *ECCV*. Springer, 2016.