Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation

Mathieu Labbé, *Student Member, IEEE*, and François Michaud, *Member, IEEE*

Abstract—In appearance-based localization and mapping, loop-closure detection is the process used to determinate if the current observation comes from a previously visited location or a new one. As the size of the internal map increases, so does the time required to compare new observations with all stored locations, eventually limiting online processing. This paper presents an online loop-closure detection approach for large-scale and long-term operation. The approach is based on a memory management method, which limits the number of locations used for loop-closure detection so that the computation time remains under real-time constraints. The idea consists of keeping the most recent and frequently observed locations in a working memory (WM) that is used for loop-closure detection, and transferring the others into a long-term memory (LTM). When a match is found between the current location and one stored in WM, associated locations that are stored in LTM can be updated and remembered for additional loop-closure detections. Results demonstrate the approach's adaptability and scalability using ten standard datasets from other appearance-based loop-closure approaches, one custom dataset using real images taken over a 2-km loop of our university campus, and one custom dataset (7 h) using virtual images from the racing video game "Need for Speed: Most Wanted."

Index Terms—Appearance-based localization and mapping, bag-of-words approach, dynamic Bayes filtering, place recognition.

I. Introduction

AUTONOMOUS robots that operate in real-life settings must be able to navigate in large, unstructured, dynamic, and unknown spaces. Simultaneous localization and mapping (SLAM) [1] is the capability, which is required by robots to build and update a map of their operating environment and to localize themselves in it. A key feature in SLAM is to recognize previously visited locations. This process is also known as loop-closure detection, referring to the fact that coming back to a previously visited location makes it possible to associate this location with another one recently visited.

For most of the probabilistic SLAM approaches [2]–[13], loop-closure detection is done locally, i.e., matches are found between new observations and a limited region of the map, de-

termined by the uncertainty associated with the robot's position. Such approaches can be processed under real-time constraints at 30 Hz [14] as long as the estimated position is valid, which cannot be guaranteed in real-world situations [15]. As an exclusive or complementary alternative, appearance-based loop-closure detection approaches [15]–[19] generally detect a loop closure by comparing a new location with all previously visited locations, independently of the estimated position of the robot. If no match is found, then a new location is added to the map. However, a robot that operates in large areas for a long period of time will ultimately build a very large map, and the amount of time required to process new observations increases with the number of locations in the map. If computation time becomes larger than the acquisition time, a delay is introduced, making updating and processing the map difficult to achieve online.

Our interest lies in developing an online appearance-based loop-closure detection approach that can deal with large-scale and long-term operation. Our approach dynamically manages the locations that are used to detect loop closures, in order to limit the time required to search through previously visited locations. This paper describes our memory management approach to accomplish appearance-based loop-closure detection, in a Bayesian framework, with real-time constraints for large-scale and long-term operation. Processing time, i.e., the time required to process an acquired image, is the criterion used to limit the number of locations kept in the robot's working memory (WM). To identify the locations to keep in WM, the solution that is studied in this paper consists of keeping the most recent and frequently observed locations in WM, and transferring the others into long-term memory (LTM). When a match is found between the current location and one stored in WM, associated locations that are stored in LTM can be remembered and updated. This idea is inspired from observations made by psychologists [20], [21] that people remember more the areas where they spent most of their time, compared with those where they spent less time. By following this heuristic, the compromise made between search time and space is, therefore, driven by the environment and the experiences of the robot.

Because our memory management mechanism is made to ensure satisfaction of real-time constraints for online processing (in the sense that the time required to process new observations remains lower or equal to the time to acquire them, as in [5], [13], [14], and [47]), independently of the scale of the mapped environment, our approach is named real-time appearance-based mapping (RTAB-Map[1]). An earlier version of RTAB-Map has been presented in [22]. This paper presents in more detail the

improved version tested with a much wider set of conditions, and is organized as follows. Section II reviews appearance-based loop-closure detection approaches. Section III describes RTAB-Map. Section IV presents experimental results, and Section V presents limitations and possible extensions to our approach.

II. RELATED WORK

For global loop-closure detection, vision is the sense that is generally used to derive observations from the environment because of the distinctiveness of features extracted from the environment [23]–[25], although successful large-scale mapping using laser range finder data is possible [26]. For vision-based mapping, the bag-of-words [27] approach is commonly used [16], [18], [19], [28], [29] and has been shown to perform online loop-closure detection for paths of up to 1000 km [30]. The bag-of-words approach consists in representing each image by visual words that are taken from a vocabulary. The visual words are usually made from local feature descriptors, such as scale-invariant feature transform (SIFT) [31]. These features have high dimensionality, making it important to quantize them into a vocabulary for fast comparison instead of making direct comparisons between features. Popular quantization methods are vocabulary tree [28], approximate k-means [32], or kd tree [31]. By linking each word to related images, comparisons can be done efficiently over large datasets, as with the term frequency-inverse document frequency (TF-IDF) approach [27].

The vocabulary can be constructed offline (using a training dataset) or incrementally constructed online, although the first approach is usually preferred for online processing in large-scale environments. However, even if the image comparison using a pretrained vocabulary (as in FAB-MAP 2.0 [30]) is fast, real-time constraint satisfaction is effectively limited by the maximum size of the mapped environment. The number of comparisons can be decreased by considering only a selection of previously acquired images (referred to as key images) for the matching process, while keeping detection performance nearly the same compared with using all images [33]. Nevertheless, processing time for each image acquired still increases with the number of key images. In [34], a particle filter is used to detect transition between sets of locations referred to as categories, but again, the cost to update the place distribution increases with the number of categories.

Even if the robot is always revisiting the same locations in a closed environment, perceptual aliasing, changes that can occur in dynamic environments or the lack of discriminative information, may affect the ability to recognize previously visited locations. This leads to the addition of new locations in the map and, consequently, influences the satisfaction of real-time constraints [35]. To limit the growth of images to match, pruning [36] or clustering [37] methods can be used to consolidate portions of the map that exceed a spatial density threshold. This limits growth over time, but not in relation to the size of the explored environment.

Finally, memory management approaches have been used in robot localization to increase recognition performance in dynamical environments [38] or to limit memory used [39]. In contrast, our memory management is used for online localiza-
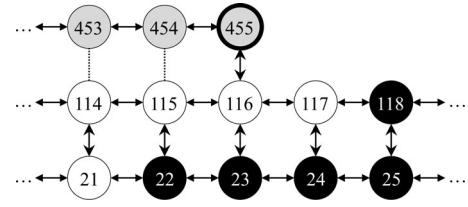

Fig. 1. Graph representation of locations. Vertical arrows denote loop-closure links and horizontal arrows denote neighbor links. Dotted links show undetected loop closures. Black locations are those in LTM, white ones are in WM, and gray ones are in STM. Node 455 is the current acquired location.

tion and mapping, where new locations are dynamically added over time.

III. ONLINE APPEARANCE-BASED MAPPING

The objective of our study is to provide an appearance-based localization and mapping solution independent of time and size, to achieve online loop-closure detection for long-term operation in large environments. The idea resides in only using a limited number of locations for loop-closure detection so that real-time constraints can be satisfied, while still gaining access to locations of the entire map whenever necessary. When the number of locations in the map makes processing time for finding matches greater than a time threshold, our approach transfers locations less likely to cause loop-closure detection from the robot's WM to LTM so that they do not take part in the detection of loop closures. However, if a loop closure is detected, neighbor locations can be retrieved and brought back into WM to be considered in future loop-closure detections.

As an illustrative example used throughout this paper, Fig. 1 shows a graph representation of locations after three traversals of the same region. Each location is represented by an image signature, a time index (or age), and a weight, and locations are linked together in a graph by neighbor or loop-closure links. These links represent locations that are near in time or in space, respectively.

Locations in LTM are not used for loop-closure detection. Therefore, it is important to choose carefully which locations to transfer to LTM. A naive approach is to use a first-in first-out (FIFO) policy, pruning the oldest locations from the map to respect real-time constraints. However, this sets a maximum sequence of locations that can be memorized when exploring an environment. If the processing time reaches the time threshold before loop closures can be detected, pruning the older locations will make it impossible to find a match. As an alternative, a location could be randomly picked, but it is preferable to keep in WM the locations that are more susceptible to being revisited. As explained in Section I, the idea that is studied in this paper is based on the working hypothesis that locations seen more frequently than others are more likely to cause loop-closure detections. Therefore, the number of times a location has been consecutively viewed is used to set its weight. When a transfer from WM to LTM is necessary, the location with the lowest weight is selected. If many locations have the same lowest weight, the oldest one is transferred.

Fig. 2 illustrates the RTAB-Map memory management model. The perception module acquires an image and sends it to sensory
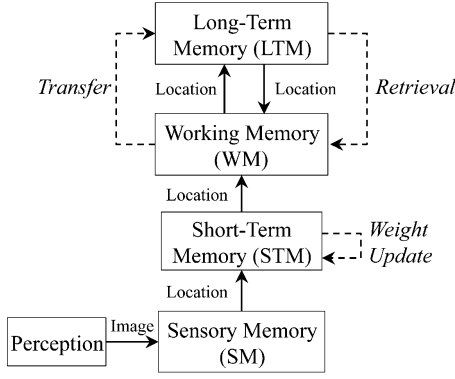
Fig. 2.   RTAB-Map memory management model.

Algorithm 1 RTAB-Map

1: $time \leftarrow$ TIMENOW()   ▷ TIMENOW() returns current time
2: $I_t \leftarrow$ acquired image
3: $L_t \leftarrow$ LOCATIONCREATION(I_t)
4: **if** z_t (of L_t) is a bad signature (using T_{bad}) **then**
5:     Delete L_t
6: **else**
7:     Insert L_t into STM, adding a neighbor link with L_{t-1}
8:     Weight Update of L_t in STM (using $T_{\text{similarity}}$)
9:     **if** STM's size reached its limit (T_{STM}) **then**
10:         Move oldest location of STM to WM
11:     **end if**
12:     $\boldsymbol{p}(S_t|L^t) \leftarrow$ Bayesian Filter Update in WM with L_t
13:     Loop Closure Hypothesis Selection ($S_t = i$)
14:     **if** $S_t = i$ is accepted (using T_{loop}) **then**
15:         Add loop closure link between L_t and L_i
16:     **end if**
17:     Join $trash$'s thread   ▷ Thread started in TRANSFER()
18:     RETRIEVAL(L_i)   ▷ LTM → WM
19:     $pTime \leftarrow$ TIMENOW() $- time$ ▷ Processing time
20:     **if** $pTime > T_{\text{time}}$ **then**
21:         TRANSFER()   ▷ WM → LTM
22:     **end if**
23: **end if**

memory (SM). SM evaluates the image signature to reduce data dimensionality and to extract useful features for loop-closure detection. Then SM creates a new location with the image signature and sends it to short-term memory (STM). STM updates recently created locations through a process referred to as weight update. If weight update considers that the new location is similar to the last one in STM, it merges them into the new one, and it increases the weight of the new location.

STM is used to observe similarities through time between consecutive images for weight updates, while the role of the WM is to detect loop closures between locations in space. Similar to [17], RTAB-Map does not use locations in STM to avoid loop closures on locations that have just been visited (because most of the time, the last location frequently looks similar to the most recent ones). The STM size T_{STM} is set based on the robot velocity and the rate at which the locations are acquired. When the number of locations in STM reaches T_{STM}, the oldest location in STM is moved into WM. RTAB-Map evaluates loop-closure probabilities with a discrete Bayesian filter by comparing the new location with the ones in WM. A loop closure is detected and locations are linked together when a high loop-closure probability is found between a new and an old location in WM. Two steps are then keys to ensuring that locations more susceptible to cause future loop-closure detections are in WM while keeping the WM size under an online limit tractable by the Bayesian filter. The first step is called retrieval. Neighbor locations of the highest loop-closure probability, for those which are not in WM, are brought back from LTM into WM, increasing the probability of identifying loop closures with future nearby locations. The second step is called transfer. If the processing time for loop-closure detection is greater than the time threshold T_{time}, the oldest of the least viewed locations (i.e., oldest locations with smallest weights) are transferred to LTM. The number of transferred locations depends on the number of locations that are added to WM during the current cycle.

Algorithm 1 illustrates the overall loop-closure detection process, which is explained in detail in the following sections.

A. Location Creation

The bag-of-words approach [27] is used to create signature z_t of an image, which is acquired at time t. An image signature is represented by a set of visual words that are contained in a visual vocabulary incrementally constructed online. We chose to use an incremental rather than a pretrained vocabulary to avoid having to go through a training step for the targeted environment.

Using OpenCV [40], speeded-up robust features (SURF) [41] are extracted from the image to derive visual words. Each visual word of the vocabulary refers to a single SURF feature's descriptor (a vector of 64 dimensions). Each SURF feature has a strength referred to as feature response. The feature response is used to select the most prominent features in the image. To avoid bad features, only those over a feature response of T_{response} are extracted. A maximum of $T_{\text{maxFeatures}}$ SURF features with the highest feature response are kept to have nearly the same number of words across the images. If few SURF features are extracted (under a ratio T_{bad} of the average features per image), the signature is considered to be a bad signature and is not processed for loop-closure detection. This happens when an image does not present discriminative features, such as a white wall in an indoor scene.

For good signatures, to find matches with words that are already in the vocabulary (a process referred to as the quantization step), SURF features are compared using the distance ratio between the nearest and the second-nearest neighbor [called the nearest-neighbor distance ratio, (NNDR)]. As in [31], two features are considered to be represented by the same word if the distance with the nearest neighbor is less than T_{NNDR} times the distance to the second-nearest neighbor. Because of the high dimensionality of SURF descriptors, a randomized forest of four kd trees (using FLANN [42]) is used. This structure increases the efficiency of the nearest-neighbor search when matching descriptors from a new signature with the ones associated with each word in the vocabulary (each leaf of the kd trees corresponds to a word in the vocabulary). The randomized kd-tree approach was chosen over the hierarchical k-means

Algorithm 2 Create location L with image I

1: **procedure** LOCATIONCREATION(I)
2:    $f \leftarrow$ detect a maximum of $T_{\mathrm{maxFeatures}}$ SURF features from image I with SURF feature response over T_{response}
3:    $d \leftarrow$ extract SURF descriptors from I with features f
4:    Prepare nearest-neighbor index (build kd-trees)
5:    $z \leftarrow$ quantize descriptors d to vocabulary (using kd-trees and T_{NNDR})
6:    $L \leftarrow$ create location with signature z and weight 0
7:    **return** L
8: **end procedure**

---

approach because of its lower tree-build time [42]. FLANN does not provide an interface for incremental changes to its search indexes (such as randomized kd trees or hierarchical k-means tree); therefore, they need to be reconstructed online at each iteration as the vocabulary is modified. The kd trees are built from all the SURF descriptors of the words that are contained in the vocabulary. Then, each descriptor that is extracted from the image is quantized by finding the two nearest neighbors in the kd trees. For each extracted feature, when T_{NNDR} criterion is not satisfied, a new word is created with the feature's descriptor. The new word is then added to the vocabulary and z_t. If the match is accepted with the nearest descriptor, its corresponding word of the vocabulary is added to z_t.

A location L_t is then created with signature z_t and time index t; its weight initialized to 0 and a bidirectional link in the graph with L_{t-1}. The summary of the location creation procedure is shown in Algorithm 2.

B. Weight Update

To update the weight of the acquired location, L_t is compared with the last one in STM, and similarity $\mathfrak{s}$ is evaluated using

$$\mathfrak{s}(z_t, z_c) = \begin{cases} N_{\mathrm{pair}}/N_{z_t}, & \text{if } N_{z_t} \geq N_{z_c} \\ N_{\mathrm{pair}}/N_{z_c}, & \text{if } N_{z_t} < N_{z_c} \end{cases} \quad (1)$$

where N_{pair} is the number of matched word pairs between the compared location signatures, and N_{z_t} and N_{z_c} are the total number of words of signature z_t and the compared signature z_c, respectively. If $\mathfrak{s}(z_t, z_c)$ is higher than a fixed similarity threshold $T_{\mathrm{similarity}}$ (ratio between 0 and 1), the compared location L_c is merged into L_t. Only the words from z_c are kept in the merged signature, and the newly added words from z_t are removed from the vocabulary: z_t is cleared, and z_c is copied into z_t. The reason why z_t is cleared is that it is easier to remove the new words of z_t from the vocabulary because their descriptors are not yet indexed in the kd trees (as explained in Section III-A). Empirically, we found that similar performances are observed when only the words of z_t are kept or that both are combined, using a different $T_{\mathrm{similarity}}$. In all cases, words that are found in both z_c and z_t are kept in the merged signature; the others are generally less discriminative. To complete the merging process, the weight of L_t is increased by the weight of L_c plus one, the neighbor and loop-closure links of L_c are redirected to L_t, and L_c is deleted from STM.

C. Bayesian Filter Update

The role of the discrete Bayesian filter is to keep track of loop-closure hypotheses by estimating the probability that the current location L_t matches one of an already visited location stored in the WM. Let S_t be a random variable that represents the states of all loop-closure hypotheses at time t. $S_t = i$ is the probability that L_t closes a loop with a past location L_i, thus detecting that L_t and L_i represent the same location. $S_t = -1$ is the probability that L_t is a new location. The filter estimates the full posterior probability $\boldsymbol{p}(S_t|L^t)$ for all $i = -1, \ldots, t_n$, where t_n is the time index that is associated with the newest location in WM, expressed as follows [29]:

$$\boldsymbol{p}(S_t|L^t) = \eta \underbrace{\boldsymbol{p}(L_t|S_t)}_{\mathrm{Observation}} \underbrace{\sum_{i=-1}^{t_n} \underbrace{\boldsymbol{p}(S_t|S_{t-1}=i)}_{\mathrm{Transition}} p(S_{t-1}=i|L^{t-1})}_{\mathrm{Belief}}$$

$$(2)$$

where η is a normalization term, and $L^t = L_{-1}, \ldots, L_t$. Note that the sequence of locations L^t includes only the locations that are contained in WM and STM. Therefore, L^t changes over time as new locations are created or when some locations are retrieved from LTM or transferred to LTM, in contrast with the classical Bayesian filtering where such sequences are fixed.

The observation model $\boldsymbol{p}(L_t|S_t)$ is evaluated using a likelihood function $\mathfrak{L}(S_t|L_t)$. The current location L_t is compared using (1) with locations that correspond to each loop-closure state $S_t = j$, where $j = 0, \ldots, t_n$, giving a score $s_j = \mathfrak{s}(z_t, z_j)$. The difference between each score s_j and the standard deviation σ is then normalized by the mean μ of all non-null scores, as in (3) [29].

$$p(L_t|S_t = j) = \mathfrak{L}(S_t = j|L_t) = \begin{cases} \frac{s_j - \sigma}{\mu}, & \text{if } s_j \geq \mu + \sigma \\ 1, & \text{otherwise.} \end{cases}$$

$$(3)$$

For the new location probability $S_t = -1$, the likelihood is evaluated using

$$p(L_t|S_t = -1) = \mathfrak{L}(S_t = -1|L_t) = \frac{\mu}{\sigma} + 1 \quad (4)$$

where the score is relative to μ on the σ ratio. If $\mathfrak{L}(S_t = -1|L_t)$ is high (i.e., L_t is not similar to a particular location in WM, as $\sigma < \mu$), then L_t is more likely to be a new location.

The transition model $\boldsymbol{p}(S_t|S_{t-1} = i)$ is used to predict the distribution of S_t, given each state of the distribution S_{t-1} in accordance with the robot's motion between t and $t-1$. Combined with $p(S_{t-1} = i|L^{t-1})$ (i.e., the recursive part of the filter), this constitutes the belief of the next loop closure. The transition model is expressed as in [29].

1) $p(S_t = -1|S_{t-1} = -1) = 0.9$: the probability of a new location event at time t given that no loop closure occurred at time $t - 1$.
2) $p(S_t = i|S_{t-1} = -1) = 0.1/N_{\mathrm{WM}}$ with $i \in [0; t_n]$: the probability of a loop closure event at time t given that no loop closure occurred at $t - 1$. N_{WM} is the number of locations in WM of the current iteration.
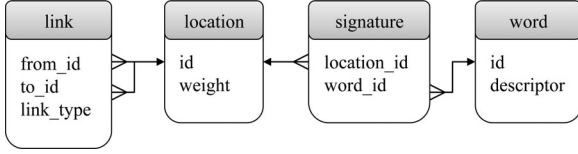
Fig. 3.   Database representation of the LTM.

3) $p(S_t = -1 | S_{t-1} = j) = 0.1$ with $j \in [0; t_n]$: the probability of a new location event at time t given that a loop closure occurred at time $t - 1$ with j.

4) $p(S_t = i | S_{t-1} = j)$ with $i, j \in [0; t_n]$: the probability of a loop-closure event at time t given that a loop closure occurred at time $t - 1$ on a neighbor location. The probability is defined as a discretized Gaussian curve ($\sigma = 1.6$) centered on j and where values are non-null for a neighborhood range of 16 neighbors (for $i = j - 16, \ldots, j + 16$). Within the graph, a location can have more than two adjacent neighbors (if it has a loop-closure link) or some of them are not in WM (because they were transferred to LTM). The Gaussian's values are set recursively by starting from $i = j$ to the end of the neighborhood range (i.e., 16); then $p(S_t >= 0 | S_{t-1} = j)$ is normalized to sum 0.9.

D. Loop-Closure Hypothesis Selection

When $\boldsymbol{p}(S_t | L^t)$ has been updated and normalized, the highest loop-closure hypothesis $S_t = i$ of $\boldsymbol{p}(S_t | L^t)$ is accepted if the new location hypothesis $p(S_t = -1 | L^t)$ is lower than the loop-closure threshold T_{loop} (set between 0 and 1). When a loop-closure hypothesis is accepted, L_t is linked with the old location L_i: the weight of L_t is increased by the one of L_i, the weight of L_i is reset to 0, and a loop-closure link is added between L_i and L_t. The loop-closure link is used to get neighbors of the old location during retrieval (see Section III-E) and to set up the transition model of the Bayes filter (see Section III-C). Note that this hypothesis selection differs from our previous work [22]. The old parameter T_{minHyp} is no longer required, and locations are no longer merged on loop closures (only a link is added). Not merging locations helps to keep different signatures of the same location for better hypothesis estimation, which is important in a highly dynamic environment or when the environment changes gradually over time in a cyclic way (e.g., day–night or weather variations).

E. Retrieval

After loop-closure detection, neighbors not in WM of the location with the highest loop-closure hypothesis are transferred back from LTM to WM. In this study, LTM is implemented as an SQLite3 database, following the schema illustrated in Fig. 3. In the link table, the *link type* tells if it is a neighbor link or a loop-closure link.

When locations are retrieved from LTM, the visual vocabulary is updated with the words that are associated with the corresponding retrieved signatures. Common words from the retrieved signatures still exist in the vocabulary; therefore, a reference is added between these words and the correspond-

---

Algorithm 3 Retrieve neighbors of L from LTM to WM

1: **procedure** RETRIEVAL(L)
2:     $L_r[] \leftarrow$ load a maximum of two neighbors of L from LTM (with their respective signatures $z_r[]$)
3:     Add references to $L_r[]$ for words in $z_r[]$ still in vocabulary
4:     Match old words (not anymore in vocabulary) of $z_r[]$ to current ones in vocabulary
5:     Not matched old words of $z_r[]$ are added to vocabulary
6:     Insert $L_r[]$ into WM
7: **end procedure**

---

ing signatures. For words that are no longer present in the vocabulary [because they were removed from the vocabulary when the corresponding locations were transferred (see Section III-F)], their descriptors are quantized in the same way as in Section III-A (but reusing the kd trees and doing a linear search on SURF descriptors not yet indexed to kd trees of the new words added to vocabulary) to check if more recent words represent the same SURF descriptors. This step is important because the new words that are added from the new signature z_t may be identical to the previously transferred words. For matched descriptors, the corresponding old words are replaced by the new ones in the retrieved signatures. However, all references in LTM are not immediately changed because this operation is expensive in terms of computational time. Instead, they are changed as other signatures are retrieved, and the old words are permanently removed from LTM when the system is shut down. If some descriptors are still unmatched, their corresponding words are simply added to vocabulary.

Algorithm 3 summarizes the retrieval process. Because loading locations from the database is time consuming, a maximum of two locations are retrieved at each iteration (chosen inside the neighboring range that is defined in Section III-C). When more than two locations can be retrieved, nearby locations in time (direct neighbors of the hypothesis) are prioritized over nearby locations in space (neighbors added through loop closures). In Fig. 1 for instance, if location 116 is the highest loop-closure hypothesis, location 118 will be retrieved before location 23. This order is particularly important when the robot is moving, where retrieving next locations in time is more appropriate than those in space. However, if the robot stays still for some time, all nearby locations in time will be retrieved, followed by nearby locations in space (i.e., 23, 24, 22, 25).

F. Transfer

When processing time for an image is greater than T_{time}, the oldest locations of the lowest weighted ones are transferred from WM to LTM. To be able to evaluate appropriately loop-closure hypotheses using the discrete Bayesian filter, neighbors of the highest loop-closure hypothesis are not allowed to be transferred. The number of these locations is, however, limited to the finite number of nearby locations in time (accordingly to the neighboring range defined in Section III-C), to avoid "immunizing" all nearby locations in space (which are indefinite in terms of numbers). T_{time} is set empirically to allow the robot to process online the perceived images. Higher T_{time} means that

Algorithm 4 Transfer locations from WM to LTM

1: **procedure** TRANSFER( )
2: $\quad nwt \leftarrow 0$ ▷ number of words transferred
3: $\quad nwa \leftarrow$ number of new words added by L_t and retrieved locations
4: $\quad$ **while** $nwt < nwa$ **do**
5: $\quad\quad L_i \leftarrow$ select a transferable location in WM (by weight and age), ignoring retrieved locations and those in recent WM (using T_{recent})
6: $\quad\quad$ Move L_i to $trash$
7: $\quad\quad$ Move words w_i which have no more references to any locations in WM to $trash$
8: $\quad\quad nwt \leftarrow nwt + \text{SIZE}(w_i)$
9: $\quad$ **end while**
10: $\quad$ Start $trash$'s thread to empty $trash$ to LTM
11: **end procedure**
---

more locations (and implicitly more words) can remain in WM, and more loop-closure hypotheses can be kept to better represent the overall environment. T_{time} must, therefore, be determined according to the robot's CPU capabilities, computational load, and operating environment. If T_{time} is determined to be higher than the image acquisition time, the algorithm intrinsically uses an image rate that corresponds to T_{time}, with 100% CPU usage.

Because the most expensive step of RTAB-Map is to build the nearest-neighbor index (line 4 of Algorithm 2), processing time per acquired image can be regulated by changing the vocabulary size, which indirectly influences the WM size. Algorithm 4 presents how the visual vocabulary is modified during the transfer process. A signature of a location transferred to LTM removes its word references from the visual vocabulary. If a word no longer has reference to a signature in WM, it is transferred into LTM. While the number of words that are transferred from the vocabulary is less than the number of words that are added from L_t or the retrieved locations, more locations are transferred from WM to LTM. At the end of this process, the vocabulary size is smaller than before the new words from L_t and retrieved locations were added, thus reducing the time required to create the nearest-neighbor index (kd trees) from the vocabulary for the next image. Saving the transferred locations into the database is done asynchronously using a background thread, leading to a minimal time overhead for the next iteration when joining the thread on line 17 of Algorithm 1 ($pTime$ then includes transferring time).

The way to transfer locations into LTM influences long-term operation when WM reaches its maximum size, in particular when a region (a set of locations) is seen more often than others. Normally, at least one location in a new region needs to receive a high weight through the weight update to replace an old and high weighted one in WM, in order to detect loop closures when revisiting this region. However, if there is no location in the new region that receives a high weight, loop closures could be impossible to detect unless the robot comes back to a high weighted location in an old region, and then moves from there to the new one. To handle this situation and to improve our previous work [22], a subset of the highest weighted locations (defined by $T_{\text{recent}} \times N_{\text{WM}}$) after the last loop closure detected are not allowed to be transferred to LTM. In this way, when the robot explores a new region, there are always high weighted

T_{STM}	30	SURF dimension	64
$T_{\text{similarity}}$	20%	SURF T_{NNDR}	0.8
T_{recent}	20%	SURF $T_{\text{maxFeatures}}$	400
		SURF T_{bad}	0.25

locations of this region in WM until a loop closure is detected. If the number of locations after the last loop closure detected in WM exceeds $T_{\text{recent}} \times N_{\text{WM}}$, these locations can be transferred like the ones in WM (i.e., with the criterion of the oldest of the lowest weighted locations). T_{recent} is a ratio fixed between 0 and 1. A high T_{recent} means that more locations after the last loop closure detected are kept in WM, which also leads to a transfer of a higher number of old locations to LTM.

IV. RESULTS

The performance of RTAB-Map is evaluated in terms of precision–recall metrics [30]. Precision is the ratio of true-positive loop-closure detections to the total number of detections. Recall is defined as the ratio of true-positive loop-closure detections to the number of ground-truth loop closures. To situate what can be considered good recall performance, for metric SLAM, recall of around 20–30% at 100% precision (i.e., with no false positives) is sufficient to detect most loop-closure events when the detections have uniform spatial distributions [30]. Note, however, that the need to maximize recall depends highly on the SLAM method, which is associated with the loop-closure detection approach. If metric SLAM with excellent odometry is used, a recall ratio of about 1% could be sufficient. For less accurate odometry (and even no odometry), a higher recall ratio would be required.

Using a MacBook Pro 2.66 GHz Intel Core i7 and a 128 Gb solid-state hard drive, experimentation is done on ten community datasets and two custom datasets using parameters presented in Table I. These parameters were set empirically over all datasets to give good overall recall performances (at precision of 100%), and remained the same (if not otherwise stated) to evaluate the adaptability of RTAB-Map. The only SURF parameter changed between experiments is T_{response}, which is set based on the image size. T_{time} is set accordingly to the image rate of the datasets. As a rule of thumb, T_{time} can be about 200–400 ms smaller than the image acquisition rate at 1 Hz, to ensure that all images are processed under the image acquisition rate, and even if the processing time goes over T_{time} (T_{time} then corresponds to the average processing time of an image by RTAB-Map). Therefore, for an image acquisition rate of 1 Hz, T_{time} can be set between 600 and 800 ms. For each experiment, we identify the minimum T_{loop} that maximizes recall performance at 100% precision. With the use of these bounded datasets, the LTM database could have been placed directly in the computer RAM, but it was located on the hard drive to simulate a more realistic setup for timing performances. Processing time $pTime$ is evaluated without the SURF feature extraction step (lines 2 and 3 of Algorithm 2), which is done by the camera's thread in the implementation.

TABLE II
EXPERIMENTAL CONDITIONS AND RESULTS OF RTAB-MAP ON COMMUNITY DATASETS

Data set	NC	CiC	L6I	L6O	70 km	NCO	CrC	BI	BO	BM
# images	1073	1237	388	531	5511	1626	692	1757	2277	2147
Image size	1280x480	1280x480	240x192	240x192	1600×460^2	2048x618	480x270	320x240	320x240	320x240
Image rate	≈ 0.5 Hz	≈ 0.5 Hz	1 Hz	0.5 Hz	≈ 1 Hz	1 Hz	2 Hz	1 Hz	1 Hz	1 Hz
T_{response}	1000	1000	10	10	1000	1000	75	50	50	50
T_{time} (s)	1.4	1.4	0.7	1.4	0.7	0.7	0.35	0.7	0.7	0.7
Max $pTime$ (s)	1.77	1.73	0.74	1.58	0.94	0.94	0.44	0.86	0.87	0.87
Max dict. size $\times 10^3$	110	112	52	111	57	56	28	57	57	57
Max WM size	410	377	334	373	162	259	101	220	210	204
Min T_{loop}	0.11	0.08	0.14	0.07	0.11	0.10	0.09	0.43	0.17	0.18
Precision (%)	100	100	100	100	100	100	100	100	100	100
Recall (%)	**89**	**81**	**98**	**95**	**59**	**92**	**95**	**82**	**56**	**72**
Recall (%) from other approaches [ref. #]	47 [16]	37 [16] 80 [44]	78 [44] 80 [29]	71 [29]	49 [30]	≈ 7 [46] 38 [48]	87 [44]	58 [47]	6 [47]	28 [47]

A. Community Datasets

We conducted tests with the following community datasets: NewCollege (NC) and CityCentre (CiC) [16]; Lip6Indoor (L6I) and Lip6Outdoor (L6O) [29]; 70 km [30]; New-CollegeOmni (NCO) [43]; CrowdedCanteen (CrC) [44]; BicoccaIndoor-2009-02-25b (BI), BovisaOutdoor-2008-10-04 (BO); and BovisaMixed-2008-10-06 (BM) [45] datasets. NC and CiC datasets contain images that are acquired from two cameras (left and right), totaling 2146 and 2474 images, respectively, of size 640×480. Because RTAB-Map takes only one image as input in its current implementation, the images from the two cameras were merged into one, resulting in 1073 and 1237 images, respectively, of size 1280×480. For datasets with an image rate depending on vehicle movement or over 2 Hz, some images were removed to have approximately an image rate of 1 Hz (i.e., keeping 5511 of the 9575 panoramic images for the 70 km dataset). For NCO and CrD datasets, because they contain panoramic images taken by a vehicle slower than for the 70 km dataset, $T_{\text{similarity}}$ is increased to 35% (compared with 20% for all other datasets). T_{time} is set to 1.4 s for images that are acquired every 2 s (0.5 Hz), to 0.7 s for images that are acquired every second (1 Hz), and to 0.35 s for images acquired every half second (2 Hz).

Table II summarizes experimental conditions and results. Recall performance corresponds to the maximum recall performance observed at 100% precision, and precision–recall curves are shown in Fig. 4. Compared with other approaches that also used these datasets, RTAB-Map achieves better recall performances at 100% precision (with improvements up to 54%, as shown at the bottom of Table II) while respecting real-time constraints. The maximum processing time $pTime$ is always under the image acquisition time.

B. Université de Sherbrooke Dataset

The dataset used for this test is made of images that are taken over a 2-km loop of the Université de Sherbrooke (UdeS) campus, traversed twice, as illustrated by Fig. 5. A total of 5395 images of 640×480 resolution at 1 Hz were captured with a hand-held webcam, over 90 min. The dataset contains a variety
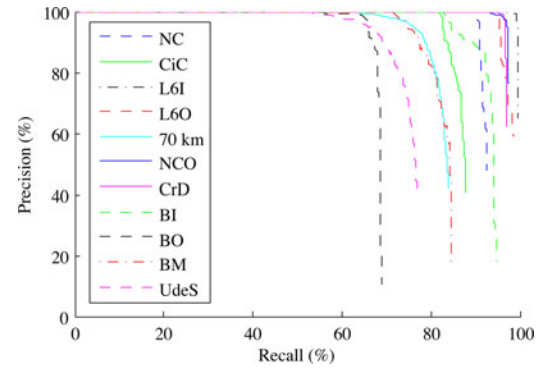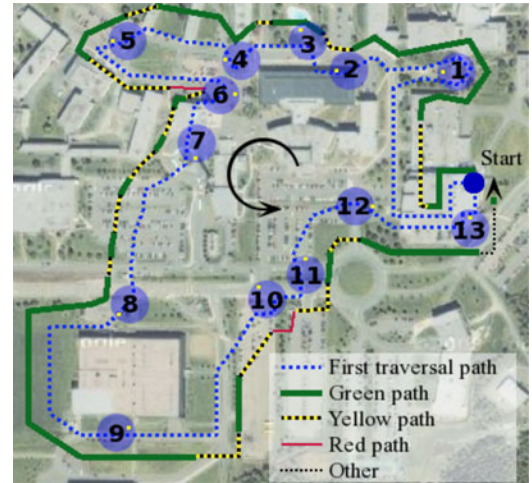

Fig. 4. Precision–recall curves for each dataset.

Fig. 5. UdeS dataset aerial view. The first traversal is represented by the dotted line. The second traversal is represented by a line located around the first. The start/end point is represented by the circle. The small white dots in the waypoint ID numbers represent camera orientation at this location. Recall performance is from the test case with $T_{\text{time}} = 0.7$ s.

of environment conditions: indoor and outdoor, roads, parkings, pedestrian paths, trees, a football field, with differences in illumination and camera orientation. To study the ability of RTAB-Map to transfer and to retrieve locations based on their occurrences, we stopped at 13 waypoints during which the

T_{time} (s)	∞	0.95	0.90	0.85	0.80	0.75	**0.70**	0.65	0.60	0.55	0.50	0.45	0.40	0.35	0.30	0.25
Max $pTime$ (s)	10.9	1.11	1.08	1.05	1.01	0.94	**0.87**	0.81	0.74	0.72	0.61	0.56	0.50	0.46	0.37	0.32
Max dict. size $\times 10^3$	714	76	73	68	65	60	**57**	52	49	44	41	36	33	29	24	21
Max WM size	2870	302	280	265	256	240	**221**	205	184	169	155	136	118	103	85	69
Min T_{loop}	0.19	0.11	0.09	0.09	0.09	0.10	**0.10**	0.11	0.11	0.10	0.15	0.12	0.12	0.12	0.10	0.14
Precision (%)	100	100	100	100	100	100	**100**	100	100	100	100	100	100	100	100	100
Recall (%)	34	47	48	51	54	48	**51**	49	50	53	41	44	47	39	34	11

camera remained still for around 20–90 s, leading to 20–90 images of the same location. After processing the images of the first traversal, it is expected that the related locations will still be in WM and that RTAB-Map will be able to retrieve nearby locations from LTM to identify loop closures. T_{response} is set to 150.

Table III presents results using different T_{time} to show the effect of memory management on recall performances. When $T_{\text{time}} = \infty$, all locations are kept in WM; therefore, loop-closure detection is done over all previously visited locations. The resulting maximum processing time is 10.9 s, which makes it impossible to use online (image acquisition time is 1 s). With $T_{\text{time}} <= 0.75$ s, processing is done online (i.e., processing time is always lower than the image acquisition time).

For $T_{\text{time}} \in [0.95; 0.35]$ s, recall varies between 39% and 54%, and this variation is caused by the choice of locations kept in WM. Small computation time variations and different T_{time} explain why some locations are transferred or retrieved in some experiments while they are not in others. Lower recall performance at $T_{\text{time}} = \infty$ is caused by the presence of a large vocabulary. As words are added to vocabulary, the matching distance between SURF features becomes smaller because of the T_{NNDR} matching criterion. Although this provides more precise matches, the matching process is more sensitive to noise. Note also that with a larger WM, there are more chances that a new location (in the presence of dynamic environmental changes) would also be found similar to another old location still in WM (which would have been in LTM with a smaller WM), creating more false positives. Conducting tests to determine the optimal size of WM for the best recall could be done, but this would be highly dependent on the environment and the path followed by the robot, and would not necessarily satisfy real-time constraints. For $T_{\text{time}} < 0.35$ s, the WM size becomes too small, and RTAB-Map is unable to detect as many loop closures. When T_{time} is reached (because the retrieved locations cannot be immediately transferred), old locations with large weights are transferred instead, making it difficult to detect loop closures if retrieval does not bring back appropriate locations.[2]

To illustrate more closely the results that are obtained for $T_{\text{time}} = 0.7$ s, at the end of the trial each waypoint is represented in WM by at least one location with a high weight. The other locations with smaller weights that are still in WM are the ones where the images did not change much over time (like the football field). At its maximum, the vocabulary has 57 000 visual words for 251 locations (221 in WM + 30 in STM). With
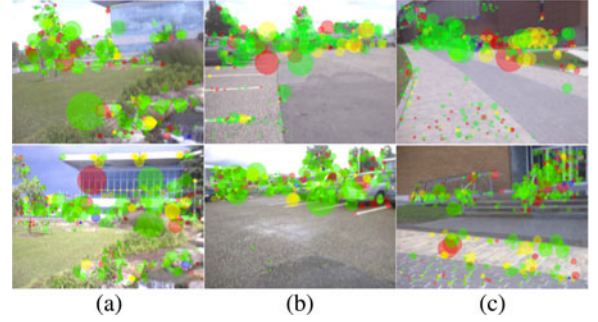
---

[2]The five-view omnidirectional images were stitched using The Panorama Factory software.

Fig. 6. Examples (with visual words represented as circles) of loop-closure hypotheses that are insufficient (under T_{loop}) to detect loop closures, caused by changes in (a) illumination conditions or (b and c) camera orientation. New words are colored in green or yellow, and words already in the vocabulary are colored in red or blue.

$T_{\text{maxFeatures}} = 400$, there are then about 227 unique words and 173 common words per location. The high number of unique words is attributable to T_{NNDR} criterion on feature quantization. Fig. 5 illustrates recall performance over the 2-km loop, categorized using three colored paths.

1) *Green* paths identify valid loop-closure hypotheses. Ground truth was labeled manually, which is based on similar visual appearance and proximity between images. The images from the second traversal are not taken at exactly the same position and the same angle compared with the first traversal. Therefore, a match within a margin of 10 locations is considered acceptable for loop closure.

2) *Yellow* paths indicate an insufficient loop-closure probability under T_{loop}. However, a yellow path also means that retrieval works as expected, i.e., RTAB-Map is able to retrieve appropriate transferred signatures (those near ground-truth loop closures) as previously visited locations are encountered. Fig. 6(a) illustrates an example. A change in illumination conditions from the first traversal (top image) caused a change in the visual words that are extracted from the image of the second traversal (bottom image). Most of the words extracted in the top image are from the tree on the left, compared with the building section in the bottom image.

3) *Red* paths identify when there is no location in WM that could be matched with the current location. A transition from a yellow to a red path occurs as follows. The likelihood of the observed location with the corresponding location of the first traversal still in WM is too low (i.e., words extracted are too different or are too common) and is lower than with other locations in the map. Because
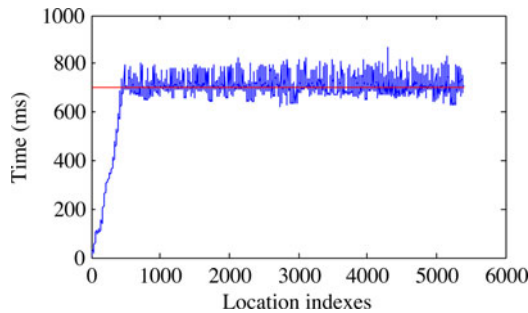
Fig. 7. Processing time in terms of locations processed over time with the UdeS dataset. T_{time} is set at 0.7 s, as shown by the horizontal line.

the associated loop-closure hypothesis is no longer the highest, nearby locations of the real loop closure cannot be retrieved from LTM to WM. Therefore, the next observed locations do not have any locations in WM that can be used to find loop closures. Fig. 6(b) and (c) illustrates what happens at the beginning of two red paths (before the waypoints 6 and 10, respectively). Over several consecutive images, the camera was not oriented in the same direction as in the first traversal, and RTAB-Map was not able to retrieve neighbor locations from LTM because the new acquired locations were more similar to locations in another part of the map. However, the wrong loop-closure hypotheses (or false positives) during a red path stayed low under T_{loop}, and thus, they were not accepted. A transition from a red to a yellow path happens when the camera returns to a location still in WM, increasing the associated loop-closure hypothesis to become the highest and resulting in a retrieval of neighbor locations from which loop closures can be detected.

4) *Other* indicates paths that are different from the ones taken during the first traversal, and therefore, there are no loop closures to find.

Finally, Fig. 7 shows the processing time for each acquired image with $T_{\text{time}} = 0.7$ s. As expected, once the processing time has reached $T_{\text{time}} = 0.7$ s (after 444 images), the memory management is triggered, and the processing time remains close to T_{time}, with an average processing time of 0.67 s and a maximum processing time of 0.87 s.

C. "Need for Speed: Most Wanted" Dataset

For this experiment, the video game "Need for Speed : Most Wanted" (NFSMW) was used to acquire images by driving around (while respecting speed limits) city Area 1 about one hundred times and then moving to Area 2 for another hundred traversals. Fig. 8 illustrates these two areas. This dataset was generated to evaluate RTAB-Map in two specific conditions:

1) frequently observing the same locations;
2) moving to new locations after observing the same area for a long period of time.

Even though the environment is synthetic, there are many large changes in illumination conditions (sun and shadows move slowly; there are also bright sunrises and random storms) that make it very challenging for loop-closure detection over long-
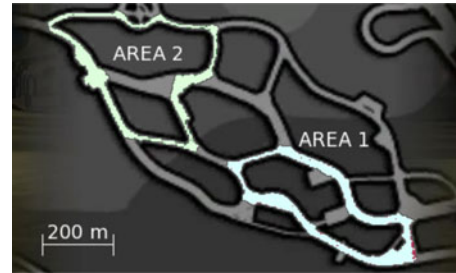

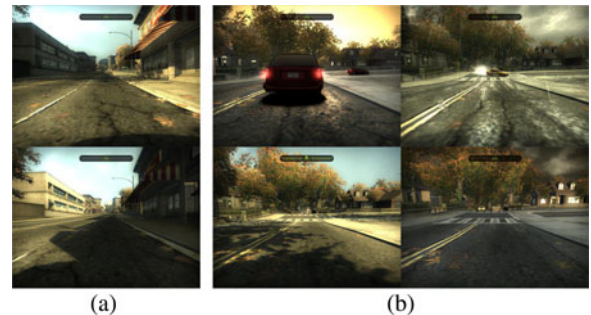Fig. 8. Map of NFSMW dataset showing Areas 1 and 2.

Fig. 9. Samples of the NFSMW dataset. In (a), the sun shines from the east (bottom) or the west (top); note the high color and contrast differences for the left and right buildings caused by the dynamic shadows. In (b), four different atmospheric conditions are shown for the same location (over 30 min).
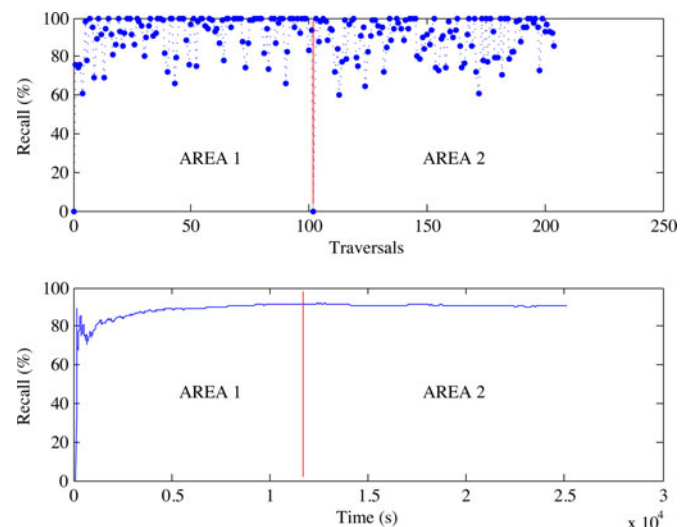

Fig. 10. Recall performance over traversals (top) and time from $t = 0$ (bottom) for the NFSMW dataset (at 100% precision).

term operation. Fig. 9 illustrates examples of such changes. A total of 25 098 images of 640×480 size were taken at 1 Hz, totalizing 7 h of driving. Because of the presence of the head-up display in the images and that the lower portions are generally made of common and repetitive road textures, the upper 10% and lower 40% of the images are not used for SURF features extraction. T_{response} is set to 150.

The upper portion of Fig. 10 illustrates recall performances that are computed for each traversal. The performances for the 102 traversals of Area 1 and the 103 traversals of Area 2 are delimited by the red vertical line. Recall of 0% is observed for

the first traversal of Area 1 and the first traversal of Area 2, as expected. Recall variations are caused by changes in illumination conditions during the traversals. If a storm happens when sunny locations are retrieved, loop closures are not detected until the storm finishes or a darker version of the encountered locations is retrieved. Generally, red paths end on road intersections, which corresponds to locations with higher weights (caused by having the vehicle stop) and where neighbors (by loop closures) with different illumination conditions are retrieved. For Area 1 and Area 2, recall at 100% precision varies from 60% to 100%.

The lower portion of Fig. 10 shows recall performances over time at 100% precision, considering all loop closures that are detected from $t = 0$. After encountered most of the changes in illumination conditions between $t = 0$ and $t = 5000$ (which corresponds to 41 traversals), the average recall performance stabilizes around 89%, for a minimum $T_{\text{loop}} = 0.10$. With a ground truth of 24 800 loop closures and 89% recall, there are 2728 duplicated locations in the global graph of the 280 locations of Area 1 and Area 2. These duplicated locations create new paths in the global map. Fig. 1 illustrates such a case. Locations 453 and 454 are duplicates of 114 and 115, respectively, because the environment changed too much; location 455 then has two paths that represent the same real location. In practice, it is likely that one of the paths will eventually be transferred to LTM, keeping only one version of the real location in WM. However, keeping two paths representing the same locations in WM may be beneficial, especially in dynamical environments with cyclic atmospheric changes like in this dataset. Locations in dark conditions could have almost no features similar to their versions in bright conditions; then loop closures are found alternately between dark and bright versions.

Looking more closely at the transition between the two areas, when moving to Area 2, the WM only contains locations of Area 1 with large weights. If a set of the new locations would not have been kept in the recent part of WM (as explained in Section III-F), loop closures would have been impossible to detect if no location received a high weight (from weight update) to replace locations in WM from Area 1. After the first traversal of Area 2, the recent part of WM was populated mostly by locations that represent road intersections (having higher weights because the vehicle stopped). The first loop closure that was detected on the second traversal was found on the first intersection encountered during the first traversal of Area 2. Next, locations were then retrieved, and a recall of 100% at 100% precision was achieved for the second traversal.

Regarding processing time, once $T_{\text{time}} = 0.7\,\text{s}$ is reached, a mean time of $0.71\,\text{s}$ is achieved for the rest of the experiment. The maximum processing time is $0.93\,\text{s}$, thus respecting the real-time constraint of 1 Hz. At the end of the experiment, there were 19 locations of Area 1 and 38 locations of Area 2 with high weights in WM, distributed mainly on road intersections.

V. Discussion

Results that are presented in Section IV suggest that RTAB-Map can achieve good recall performances at 100% precision over diverse and large-scale environments. Real-time constraints can be satisfied independently of the scale of the environment, which is very important for long-term online mapping.

Overall, using similarity occurrences reveals to be a simple and functional method to determine which locations to keep in WM. Obviously, it has limitations when an area is seen only one time before moving to a new area for a long time. To illustrate, we conducted a trial using the NFSMW setup by doing only one traversal of Area 1 and then moving for one hundred traversals of Area 2. After the first traversal of Area 1, the highest weight of a location is 2. After 56 traversals of Area 2, all locations of Area 1 were transferred to LTM (in comparison to 19 locations remaining in WM after 103 traversals of Area 2 in Section IV-C). The number of traversals of Area 2 required to transfer all locations of Area 1 in LTM depends on the weight assigned to locations during the first traversal. Returning to Area 1 then leads to the creation of duplicated locations that cannot be associated with the locations of Area 1 stored in LTM (unless they were revisited backward from the entry point of Area 2), and these duplicated locations, if visited frequently, can remain in WM to be used in future loop closures. This illustrates the compromise to be made to satisfy real-time constraints. It may happen that infrequently visited locations get transferred to LTM without being able to be remembered back, but at the same time such locations are not used in the loop-closure detection process, allowing to speed up the process using only locations that have more chance to be revisited. Such compromise is, therefore, driven by the environment and the experiences of the robot. Note that other methods to assign weights to locations could be imagined, such as having the system identify which locations are important (and assigning directly a high weight to these locations) based on events, the robot's internal states, or even from user inputs. In addition, approaches such as sparsely or randomly sampling the LTM could be used to prevent forgetting entire areas not visited often enough.

In RTAB-Map, LTM's growth influences loop-closure detection performance over large-scale and long-term operation. To understand such influence, let us define w_w the number of words in WM and STM (i.e., the visual vocabulary), n_w the number of locations in WM, w_l the number of words in LTM, and n_l the number of locations in LTM. Time complexity for each step of Algorithm 1 is given as follows:

1) *Location creation*: Building the kd trees from the vocabulary is $O(w_w \log w_w)$, and quantizing SURF descriptors extracted from the new image to kd trees is $O(\log w_w)$. The SURF feature extraction can be considered $O(1)$ as image size is fixed.
2) *Weight update*: Updating weight is $O(1)$.
3) *Bayesian filter update*: computing observation is $O(n_w)$ and belief if $O(n_w^2)$.
4) *Loop-closure hypothesis selection*: Hypothesis selection is $O(n_w)$.
5) *Retrieval*: SURF descriptors quantization is $O(\log w_w)$. Database selection query is $O(\log[w_l + n_l])$.
6) *Transfer*: Selecting a transferrable location is $O(n_w)$. Database insertion query is $O(\log[w_l + n_l])$.

When T_{time} is reached, WM size remains fixed, bounding time complexities that are associated with w_w and n_w.
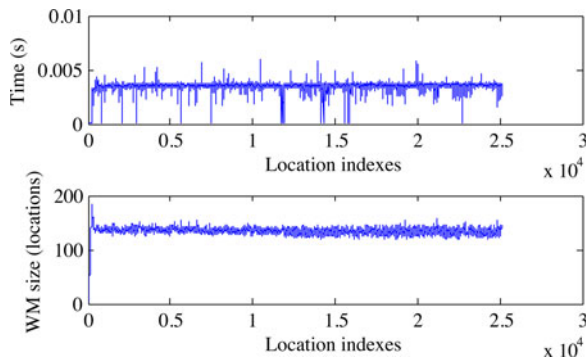
Fig. 11. (Top) Total database (LTM) access time to retrieve and transfer locations per iteration. (Bottom) WM size variation during the experiment.

However, for retrieval and transfer, time complexities also depend on LTM, and LTM size is not bounded. With the growth of LTM, T_{time} is more likely to be reached, and WM size will gradually decrease over time to satisfy real-time constraints. Theoretically, WM size may eventually become null, disabling loop-closure detection. In practice, though, the logarithmic growth in time complexity caused by LTM is very small, and WM size is not affected. For the NFSMW experiment, Fig. 11(top) shows the total database access time, which is required to retrieve and to transfer locations for each RTAB-Map iteration, for up to 7 h of use. Time growth is unnoticeable. At the end of the experiment, the database size is 3.1 GB with 6.3 million words and 25 098 locations (all merged and bad locations were kept in the database for debugging purpose). Fig. 11(bottom) shows the WM size over time. The higher variations of WM size after around 12 000 locations are mainly caused by environmental changes from Area 1 to Area 2. If necessary, a solution to LTM size would be to limit database growth by permanently removing offline some locations from the database. For instance, paths leading to the same high weighted locations could be eliminated based on the sum of the weights of locations in the paths. If the number of distant high weighted locations gets very high, important locations could ultimately be deleted, resulting in a dismembered global map (i.e., many disconnected smaller maps) if weight is the primary transfer criterion. Pruning the oldest locations (independently of the weight) may be better to preserve a unique global map, at the cost of forgetting important old locations.

In dynamic environments, the performance of RTAB-Map is also highly dependent on the quality of the SURF features extracted. We observed in our experiments that SURF features are relatively sensible to changes in illumination and shadows, reducing the number of discriminative features for more "garbage features" (or common words) in images. In RTAB-Map, at least one discriminative feature in the environment is required to find a loop closure, but if there are many "garbage features," this means that other locations also receive high likelihoods, thus shadowing the weight of the discriminative feature. In this case, (4) scores high (for a new location probability) because the standard deviation of the likelihood scores is small comparatively to the mean. Feature weighting may help in such cases by assigning a high weight to discriminative features and a lower weight

to "garbage features." However, we think that doing so would lead to more false positives. By considering all features with the same weight in RTAB-Map, many discriminative features are required for a location to score higher than others if many "garbage features" are present, decreasing the chance of false positives. We prefer avoiding finding a loop closure under such condition (like in environments that are populated with many dynamic objects or people) rather than increasing the probability of accepting a false positive.

As additional improvements, exploiting sparseness of the Bayesian filter [30] or using more efficient nearest-neighbor structures (to avoid reconstructing the whole kd trees at each iteration) may speed up the process to keep more locations in WM. However, our focus in this paper is on real-time constraint satisfaction (i.e., what should be done when computation time reaches the time threshold), and not optimizing complexities depending on WM size. Finally, to overcome the occurrences of red paths that are caused by changes in camera orientation (see Section IV-B), active localization could be triggered by detecting decreasing hypotheses, which could make the system move the camera in the right direction to let RTAB-Map retrieve appropriate locations from LTM to WM.

VI. CONCLUSION

Results that have been presented in this paper suggest that RTAB-Map, which is a loop-closure detection approach based on a memory management mechanism, is able to meet real-time constraints needed for online large-scale and long-term operation. While keeping a relatively constant number of locations in WM, online processing is achieved for each new image acquired. Retrieval is a key feature that allows RTAB-Map to reach adequate recall ratio even when transferring a high proportion of the perceived locations in LTM, which are not used for loop-closure detection. In future work, in addition to the possible extensions outlined in Section V, we plan to study how RTAB-Map can be combined with other approaches to implement a complete simultaneous localization and mapping system.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.*. Cambridge, MA, USA: MIT Press, 2005.

[2] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proc. Int. Joint Conf. Artif. Intell.*, 2003, vol. 18, pp. 1151–1156.

[3] M. Bosse, P. Newman, J. Leonard, and S. Teller, "Simultaneous localization and map building in large-scale cyclic environments using the atlas framework," *Int. J. Robot. Res.*, vol. 23, no. 12, pp. 1113–1139, Dec. 2004.

[4] A. I. Eliazar and R. Parr, "DP-SLAM 2.0," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2004, pp. 1314–1320.

[5] C. Estrada, J. Neira, and J. D. Tardós, "Hierarchical SLAM: Real-time accurate mapping of large environments," *IEEE Trans. Robot.*, vol. 21, no. 4, pp. 588–596, Aug. 2005.

[6] J. Folkesson and H. I. Christensen, "Closing the loop with graphical SLAM," *IEEE Trans. Robot.*, vol. 23, no. 4, pp. 731–741, Aug. 2007.

[7] L. Clemente, A. Davison, I. Reid, J. Neira, and J. Tardós, "Mapping large loops with a single hand-held camera," presented at the Robot.: Sci. Syst., Atlanta, GA, USA, Jun. 2007.

[8] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao–Blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[9] J.-L. Blanco, J. Fernandez-Madrigal, and J. Gonzalez, "Toward a unified Bayesian approach to hybrid metric—topological SLAM," *IEEE Trans. Robot.*, vol. 24, no. 2, pp. 259–270, Apr. 2008.

[10] J. Callmer, K. Granström, J. Nieto, and F. Ramos, "Tree of words for visual loop closure detection in urban SLAM," in *Proc. Australasian Conf. Robot. Autom.*, 2008, pp. 102–109.

[11] L. M. Paz, J. D. Tardós, and J. Neira, "Divide and conquer: EKF SLAM in O(n)," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1107–1120, 2008.

[12] P. Pinies and J. D. Tardós, "Large-scale SLAM building conditionally independent local maps: Application to monocular vision," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1094–1106, Oct. 2010.

[13] D. Schleicher, L. Bergasa, M. Ocaña, R. Barea, and E. López, "Real-time hierarchical stereo visual SLAM in large-scale environments," *Robot. Auton. Syst.*, vol. 58, no. 8, pp. 991–1002, 2010.

[14] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[15] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using visual appearance and laser ranging," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 1180–1187.

[16] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Int. J. Robot. Res.*, vol. 27, no. 6, pp. 647–665, Jun. 2008.

[17] A. Angeli, S. Doncieux, J. Meyer, and D. Filliat, "Incremental vision-based topological SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 1031–1036.

[18] T. Botterill, S. Mills, and R. Green, "Bag-of-words-driven, single-camera simultaneous localization and mapping," *J. Field Robot.*, vol. 28, no. 2, pp. 204–226, 2011.

[19] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, "View-based maps," *Int. J. Robot. Res.*, vol. 29, no. 8, pp. 941–957, Jul. 2010.

[20] R. Atkinson and R. Shiffrin, "Human memory: A proposed system and its control processes," in *Psychology of Learning and Motivation: Advances in Research and Theory,* Vol. 2, New York, NY, USA Elsevier, 1968, pp. 89–195.

[21] A. Baddeley, *Human Memory: Theory and Practice.* Abingdon, Oxon, U.K.: Psychology Press, 1997.

[22] M. Labbé and F. Michaud, "Memory management for real-time appearance-based loop closure detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 1271–1276.

[23] M. J. Milford and G. F. Wyeth, "Mapping a suburb with a single camera using a biologically inspired SLAM system," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1038–1053, Oct. 2008.

[24] P. Newman and K. Ho, "SLAM- loop closing with visually salient features," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 635–642.

[25] A. Tapus and R. Siegwart, "Topological SLAM," in *Springer Tracts in Advanced Robotics,* Vol. 46, New York, NY, USA Springer-Verlag, 2008, pp. 99–127.

[26] M. Bosse and J. Roberts, "Histogram matching and global initialization for laser-only SLAM in large unstructured environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 4820–4826.

[27] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. 9th Int. Conf. Comput. Vis.*, Nice, France, 2003, pp. 1470–1478.

[28] D. Nistèr and H. Stewènius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2006, pp. 2161–2168.

[29] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1027–1037, Oct. 2008.

[30] M. Cummins and P. Newman, "Highly scalable appearance-only SLAM– FAB-MAP 2.0," presented at the Robot.: Sci. Syst., Seattle, WA, USA, Jun. 2009.

[31] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[32] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.

[33] O. Booij, Z. Zivkovic, and B. Kröse, "Efficient data association for view based SLAM using connected dominating sets," *Robot. Auton. Syst.*, vol. 57, no. 12, pp. 1225–1234, 2009.

[34] A. Ranganathan, "Pliss: Detecting and labeling places using online change-point detection," presented at the Robot.: Sci. Syst. Conf., Zaragoza, Spain, Jun. 2010.

[35] A. J. Glover, W. P. Maddern, M. J. Milford, and G. F. Wyeth, "FAB-MAP + RatSLAM: Appearance-based SLAM for multiple times of day," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 3507–3512.

[36] M. Milford and G. Wyeth, "Persistent navigation and mapping using a biologically inspired SLAM system," *Int. J. Robot. Res.*, vol. 29, no. 9, pp. 1131–1153, Aug. 2010.

[37] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 1156–1163.

[38] F. Dayoub and T. Duckett, "An adaptive appearance-based map for long-term topological localization of mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 3364–3369.

[39] A. Pronobis, L. Jie, and B. Caputo, "The more you learn, the less you store: Memory-controlled incremental SVM for visual place recognition," *Image Vision Comput.*, vol. 28, no. 7, pp. 1080–1097, 2010.

[40] G. Bradski, "The OpenCV Library," *Dr. Dobb's J. Softw. Tools*, vol. 25, no. 11, pp. 122–125, Nov. 2000.

[41] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded up robust features (SURF)," *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

[42] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, 2009, pp. 331–340.

[43] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *Int. J. Robot. Res.*, vol. 28, no. 5, pp. 595–599, 2009.

[44] A. Kawewong, N. Tongprasit, and O. Hasegawa, "PIRF-Nav 2.0: Fast and online incremental appearance-based loop-closure detection in an indoor environment," *Robot. Auton. Syst.*, vol. 59, no. 10, pp. 727–739, 2011.

[45] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. Sorrenti, and P. Taddei, "Rawseeds ground truth collection systems for indoor self-localization and mapping," *Auton. Robots*, vol. 27, no. 4, pp. 353–371, 2009.

[46] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroeter, L. Murphy, W. Churchill, D. Cole, and I. Reid, "Navigating, recognizing and describing urban spaces with vision and lasers," *Int. J. Robot. Res.*, vol. 28, no. 11–12, p. 1406, 2009.

[47] D. Gálvez-López and J. Tardós, "Real-time loop detection with bags of binary words," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 51–58.

[48] W. Maddern, M. Milford, and G. Wyeth, "Continuous appearance-based trajectory SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3595–3600.

Mathieu Labbé (S'10) received the B.Sc.A. degree in computer engineering and the M.Sc.A. degree in electrical engineering from the Université de Sherbrooke, Sherbrooke, QC, Canada, in 2008 and 2010, respectively, where he is currently working toward the Ph.D. degree in electrical engineering.

His research interests include computer vision, autonomous robotics, and robot learning.

François Michaud (M'90) received the Bachelor's, Master's, and Ph.D. degrees, all in electrical engineering from the Université de Sherbrooke, QC, Canada, in 1992, 1993, and 1996, respectively.

After completing postdoctoral work at Brandeis University, Waltham, MA, USA in 1997, he became a faculty member with the Department of Electrical Engineering and Computer Engineering, Université de Sherbrooke, and founded IntRoLab: a research laboratory working on designing intelligent autonomous systems that can assist humans in living environments. He is the Director of the Interdisciplinary Institute for Technological Innovation. His research interests include architectural methodologies for intelligent decision-making and design of interactive autonomous mobile robots.

Dr. Michaud held a Canada Research Chair in Mobile Robots and Autonomous Intelligent Systems from 2001 to 2011. He is a member of the Association of the Advancement of Artificial Intelligence and the Ordre des ingénieurs du Québec.