O-POCO: Online POint cloud COmpression mapping for visual odometry and SLAM

Luis Contreras[1] and Walterio Mayol-Cuevas[2]

Abstract— **This paper presents O-POCO, a visual odometry and SLAM system that makes online decisions regarding what to map and what to ignore. It takes a point cloud from classical SfM and aims to sample it on-line by selecting map features useful for future 6D relocalisation. We use the camera's traveled trajectory to compartamentalize the point cloud, along with visual and spatial information to sample and compress the map. We propose and evaluate a number of different information layers such as the descriptor information's relative entropy, map-feature occupancy grid, and the point cloud's geometry error. We compare our proposed system against both SfM, and online and offline ORB-SLAM using publicly available datasets in addition to our own. Results show that our online compression strategy is capable of outperforming the baseline even for conditions when the number of features per key-frame used for mapping is four times less.**

I. INTRODUCTION

To be able to have a mapping platform that is capable of deciding what information to keep and what to eschew is both central to embedded decision making and appealing for memory and computational efficiency. Having a reduced and better informed representation of the environment also facilitates tasks like planning and or leads to a reduced transmission bandwidth useful for multi-agent exploration and re-exploration.

In previous work ([1]), a number of trajectory-driven *offline* compression techniques were studied. That work shown that it is possible to have compression rates of up to 80% without sacrificing relocalisation accuracy. In that work we used feature selection based on the camera traveled trajectory and its visibility, descriptor information and 3D position. We consider as input a point cloud of features and the series of cameras from winch it was generated; each map feature has associated with it both the cameras that can see it (*visibility matrix*) and a visual descriptor.

However, a crucial competence for any exploring agent is the ability to take decisions *in-situ* and be able to concurrently operate without deferring action. In the case of a visual mapping platform this means being able to decide what should be mapped and what can be ignored. This work is effort in that direction.

As in our previous offline work, we primarily use relocalisation as the performance criteria over a geometry preservation criteria more often used for map compression. This we do since we argue that relocalisation ability is more useful for most robotic action tasks than something aimed at visualisation. We also exploit trajectory information as the basis for compression, and test several criteria for selecting map features. We look at the influence of feature positions, the visual information of each descriptor and feature visibility. While we concentrate on using relocalisation as performance metric, we also analyze resulting geometry prevalence by computing the point to plane and occupancy grid error, and the relative entropy as a descriptor information distance.

This paper is organized as follows. In Section II we discuss work related to offline map compression. An online compression algorithm is then presented in Section III, in which we highlight the key concepts for performing the trajectory parametrisation used in our proposed point cloud compression. Experimental results are presented in SectionIV. The work ends outlining some discussion and conclusions.

II. RELATED WORK

There have been many proposed techniques for point cloud compression and simplification, such as those in [2], whose goal is to produce a reduced map version that is good for visualisation. These approaches involve clustering, iterative simplification and particle simulation methods.

The work in [3] presented a technique for relocalisation based on 3D points visibility prediction with respect to the query camera and a memory-based learning algorithm; however, its goal is to improve localisation given a point cloud. [4] does some compression in the sense of reducing the search space by evaluating features to improve camera localisation, but again it starts with an offline generated point cloud (generated using Structure-from-Motion).

ORB-SLAM as presented in [5] is a feature-based visual SLAM system based on key-frame selection and Bag of Words concepts ([6]); similarly, [7] presented a SLAM technique that reduces the map generation process using key-frames. These kind of techniques, even though they use previous cameras information, do no take into consideration the camera's trajectory. They present some rules to define when to capture a new key-frame and assume the points in it to be valid for constructing the final map.

OctoMap [8] is a method of probability mapping utilizing Octrees to reduce redundant information, achieving compression levels of up to 44% with respect to the original map.

Apart from [1], the other closest works to ours are offline. In [9] and [10], an offline 3D point selection method uses an optimizer tuned to also reduce relocalisation error. The

*This work was partially supported by CONACYT and the Secretaria de Educacion Publica, Mexico

Department of Computer Science, University of Bristol, United Kingdom
[1]cslact@bristol.ac.uk
[2]wmayol@cs.bris.ac.uk

reported impressive-sounding compression rate is 1 per cent, going from a number of millions of features to the order of thousands. However, we note that the result is only for the 3D components of the map since they use a descriptor matching technique ([11]) that allows to have several descriptors per point. The method thus reduces 3D points but keeps descriptors, which are larger in memory footprint than 3D points. The end effect is that every point in the compression may end with hundreds of descriptors and therefore the real per-point compression rate is much lower. Our framework, in contrast, uses a single descriptor per point, and any reduction of descriptors results in a real reduction of map points.

III. ONLINE POINT CLOUD COMPRESSION

This work aims to present an intelligent map compression technique. The map reconstruction process can use various approaches but here it follows a classical Structure from Motion (SfM) and Iterative Closest Point (ICP) technique. This provides a series of map-features and camera poses from a sequence of RGB-D images.

The compression method first models the traveled trajectory from the camera spatial positions and divides it in segments upon its curvature. A series of map-feature subsets are then generated by associating each point in the full point cloud to a specific trajectory segment based on an association rule. Finally, a number of selection criteria are proposed to compress each subset; the final compressed point cloud is the union of all compressed subsets. In this work, a sequential implementation of this algorithm takes on average 67 ms per frame in a desktop with an i7 CPU (2.80 GHz) and 8 GB RAM.

We refer to the complete compression process as Online POint cloud COmpression, or **O-POCO**, and will be detailed in the following sections.

The use of a mapping and then sampling outline allows the use of all the matched and registered data to generate the map and locate the current camera, which is especially useful in areas with poor texture. In contrast, traditional key-frame techniques criteria of usefulness is the amount of new data observed between the current frame and the latest key-frame. However, a small difference between frames may be due the short distance between them or because the lack of new features due to poor texture areas; discarding map features in the latter case may result in a loss of camera localisation.

In Figure 1 we illustrate a typical indoor mapping scene, a poorly textured corridor sequence. Incidentally SfM and ORB-SLAM get lost. For ORB-SLAM, we attempted different parameter combinations such as 1000 features per frame and FAST ([12]) minimum response of 7 (green line), 4000 features per frame and FAST minimum response of 7 (red line), and 4000 features per frame and FAST minimum response of 1 (orange line); this last case worked offline due to the amount of data per frame to be extracted and matched. Offline ORB-SLAM ended up with 93346 points and 435 key-frames while O-POCO used 600 key-frames and 30894 map features in one sequences and 30927 in a second one.

Our approach of standard SfM+ICP with O-POCO, an online judicious choice of features, is seen to work well in this scenario. We hypothesize because, by using a compact map (i.e. less points in previous frames), we consider important all new information, regardless of the amount; in standard SfM and ORB-SLAM they add a new key-frame if the number of new points is significant, a condition that is not meet in poorly textured areas.

To make a fairer comparison, we introduce the *descriptor per key-frame* (dpf) unit as the ratio of the total points in the map to the total number of key-frames used to build that map. We introduce this unit to extend the results to methods where a map feature has more than one descriptor.

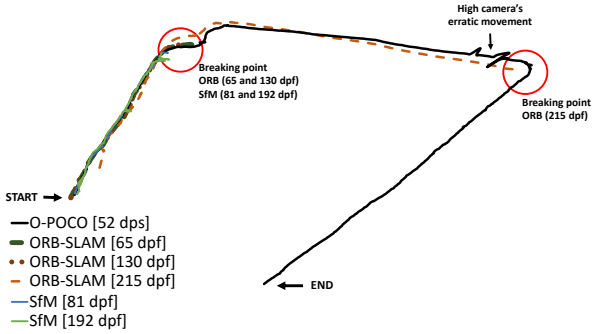


Fig. 1: SfM and ORB-SLAM vs O-POCO. Long corridor test I. Different SfM and ORB-SLAM configurations were used, but they still get lost in areas with poor texture or camera movement noise. The *descriptor per key-frame* [dpf] to compare methods is shown, and where smaller dpf is better.

A. Point Cloud Sub-Mapping

We consider a windowed version of the mapping process that consists in taking the last camera k and generate a sub-map using the features from its frame and the features associated to the previous n cameras (Figure 2). We then compress this sub-map. We add to the global map only those selected points in the current frame.

In addition, we test an alternative online compression method that consists in using only the map-features from the last camera k, and the position from the previous n cameras without considering their associated points (a kind of a key-frame compression) as shown in Figure 3.

B. Trajectory based Subsetting

The compression process takes as input a sub-map formed by a series of cameras and their associated map features, and outputs a compact version, as follows. First, we interpolate the curve that best fits the camera trajectory – which is useful to reduce noisy and mislocated cameras–, and we model this curve as a series of control points by applying *NURBS* (non-uniform rational b-splines) ([13]), as shown in Figure 4.

By construction, the control point generation is highly related with the change rate in the first and second derivative of the curve to be modeled and, therefore, to its shape. In

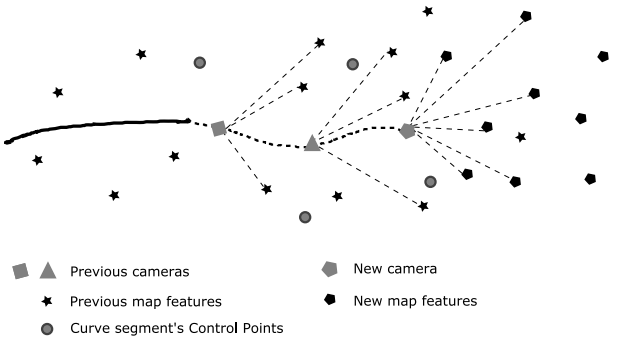


Fig. 2: O-POCO mapping process. We generate a sub-map of new features and those features in the global map which are visible from from the previous n cameras. We model the camera trajectory using the last n cameras. We aim to compress this sub-map and add the remaining points to the global map.

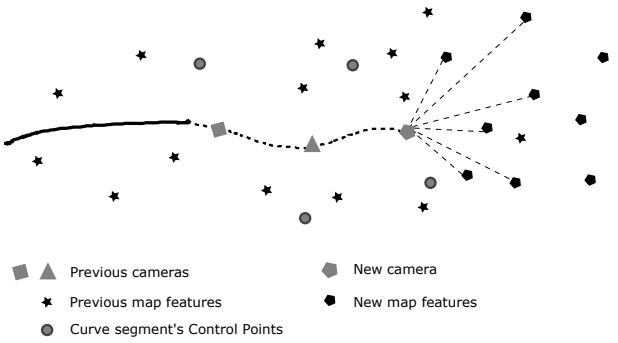


Fig. 3: Alternative key-frame mapping. We generate a sub-map by taking only the latest features and, as in the previous case, and model the camera trajectory from the last n cameras. Similarly, we compress this sub-map and add to the global map the selected points.

consequence, we prefer its use over other models because spline models encode the curvature level: in smooth trajectories, the control points tend to be close to the curve and evenly separated. On the other hand, in trajectories with high changes in shape, the control points tend to be separated from the curve and with larger distances between them.

Next, we define a series of map features subsets by dividing the traveled trajectory in segments, based on the control points information; for each control point we consider its neighboring points, and obtain the cameras in the segment they define. To avoid overlapping between consecutive segments, we considered only half of the intersection section between each segment. Thereafter, with the visibility matrix, we generate a subset of those map-features in the point cloud that can be seen from all the cameras in the current segment, as illustrated in Figure 5. We call this process Trajectory based Subsetting (or *TbS*) as detailed in the algorithm of Figure 6.

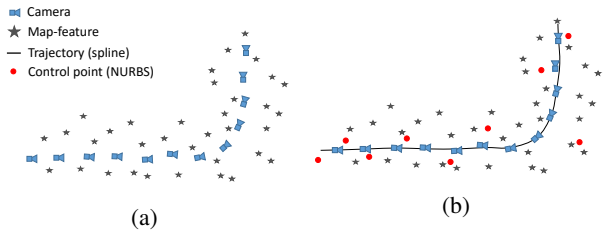


Fig. 4: We first model the camera trajectory from a series of cameras in (a) by interpolating the curve that best fits all n cameras in the window – reducing the uncertainty from noisy positions and mislocated cameras – and then modeling this curve through a series of control points using NURBS, as shown in (b).

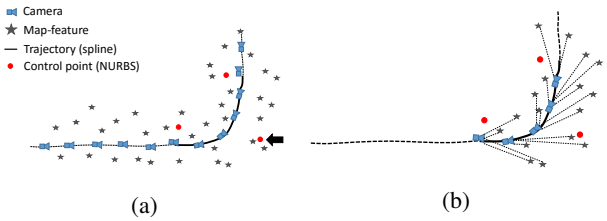


Fig. 5: In the subsetting process, for each control point in the curve model, e.g. the point indicated by the black arrow in a), we take all the cameras on the segment of the trajectory they represent – indicated as a solid line – and select all the map features those cameras can see, as shown in b).

C. Map Compression

The selection is based in the spatial position of the map features in the subset. We clusterise the subset with *k-means*, where we define k equal to a fixed maximum points per subset, and keep the closest point to each of the obtained cluster centers.

The complete Online POint cloud COmpression algorithm is shown in Figure 7.

IV. EXPERIMENTS AND RESULTS

A. Visual Odometry

First, we compare the online compression method performance for visual odometry with classical SfM and ORB-SLAM using two datasets, one publicly available (the Technische Universitat Munchen's RGB-D SLAM Dataset and Benchmark [14], Figure 8) and our own (available under request) from a long corridor with high ambiguity and zones of poor texture (Figure 9). The generated trajectory for each algorithm is shown in Figure 1, Figure 10, and Figure 11.

Even though we are using standard Structure from Motion, with no optimization in any step beyond those in OpenCV algorithms ([15]), promising results are demonstrated. In the corridor sequence, our approach overcomes online ORB-SLAM, where the latter breaks in areas with poor texture or erratic camera movements.

Require: cameras $cams$, point cloud pc, number of words NoW
Ensure: point cloud subsets $subset$
  $poly$ = spline($cams$)
  cp = NURBS($poly$)
  for all $cams$ **do**
    $segment$ = getsegment($cp(i-1)$, $cp(i+1)$)
    min_c = getmincam($segment$)
    max_c = getmaxcam($segment$)
    for cam = min_c:max_c **do**
      $subset_i$ = $subset_i$ ∪ visible(pc, cam)
    end for
  end for

Fig. 6: The Trajectory based Subsetting generates a series of point cloud subsets, based on the control points from the traveled trajectory.

B. Relocalisation

We then tested the relocalisation error in an indoors ten loops sequence traveled by three different persons to avoid any bias in the evaluation process – one sequence for mapping and the rest for testing the relocalisation performance (in total, 800 cameras to relocate), as shown in Figure 12. We generate three maps, one for each method and using the same number of key-frames – namely, offline compression (where we compress the full map from the scene, as in [1]), online key-frame compression and O-POCO, where we test several window sizes. As not all the cameras were properly relocated, for analysis purposes, we take 80 per cent of the total cameras, discarding the worst relocated, and then generate Table I to show relocalisation rates.

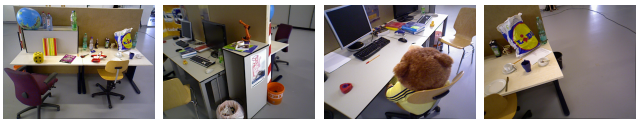
TABLE I: relocalisation mean error [in meters] for Offline compression (as in [1]), key-frame mapping and O-POCO at different windows sizes. We put in parenthesis the number of descriptors per frame as a measurement of compression performance – for comparison, SfM without compression and same key-frames uses 262 dpf.

	Relocalisation mean error [m]		
Windows size	Offline	Key-frame	O-POCO
6 cameras	0.258 (41)	2.370 (78)	2.632 (41)
10 cameras	0.424 (41)	1.573 (92)	1.405 (99)
15 cameras	0.362 (41)	1.167 (101)	0.589 (119)
20 cameras	0.328 (41)	0.839 (110)	0.203 (138)
25 cameras	0.354 (41)	0.762 (115)	0.157 (147)

We can see that, for online compression techniques, the relocalisation performance is dependent on the window size and the point cloud sub-map generation, while offline compression method stays steady at high compression rates. Results shows O-POCO overcoming all methods with big enough windows sizes by giving up on compression performance; in consequence, while relocalisation performance increases, big window sizes cause more processing time and memory footprint consumption. Figure 13 shows O-POCO relocalisation performance at different precision rates – i.e. proportion of successfully relocated cameras; for example, 90 per cent of the testing cameras are within 30 cm mean error for a windows size of 25 while, for a window size of 15, 90 percent of the testing cameras are within a mean error of 1.11 meters.

Require: frame sequence $frame$, compression rate q, window size win
Ensure: compressed map map
  if $i < win$ **then**
    (map_i, cam_i) = SfM($frame_i$)
    map = map ∪ map_i
  end if
  if $i == win$ **then**
    (map_i, cam_i) = SfM($frame_i$)
    map = map ∪ map_i
    $subset$ = TbS(map, cam_i)
    n = count($subset$), $size$ = $q * n$
    $centroids$ = clustering($subset$[position], $size$)
    map = knn($subset$[position], $centroids$)
  end if
  if $i > win$ **then**
    case *key-frame*
      (map_{tmp}, cam_i) = SfM($frame_i$)
    case *windowing*
      (map_i, cam_i) = SfM($frame_i$)
      map_{tmp} = map ∪ map_i
    (map_{sub}, cam_{sub}) = SUBMAP(map_{tmp}, cam_{i-win}:cam_i)
    $subset$ = TbS(map_{sub}, cam_{sub})
    n = count($subset$), $size$ = $q * n$
    $centroids$ = clustering($subset$[position], $size$)
    map_i = knn($subset$[position], $centroids$)
    map = map ∪ map_i
  end if

Fig. 7: O-POCO: Online POint cloud COmpression. We use classical SfM for mapping. Once we capture the first *win* cameras, we apply a Trajectory based Subsetting and selection by spatial position to this map. For each new camera we apply two windowing criteria: *key-frame*, where we take the last *win* cameras and only the 3D points from the current camera and we compress this reduced map, and add to the point cloud only the selected points; the other case generates the reduced map using the newest points and the points from the last *win* cameras.



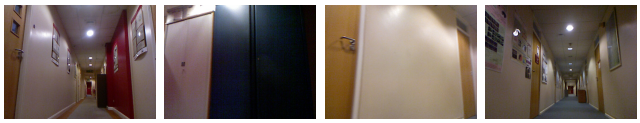
(a) (b) (c) (d)

Fig. 8: Typical views in the TUM [14] long office sequence.

C. Information Conservation

We introduce a series of metrics to measure other information levels apart from point cloud size. In all cases, from two different long corridor sequences, and using the same number of key-frames, we performed a mapping process without compression (135 dpf on average) and using O-POCO (51 dpf), and analyzed several information layers as follows.

1) Descriptor information's relative entropy: To measure information loss we use the relative entropy between the Bag of Words (BoW) distributions before and after compression.



(a) (b) (c) (d)

Fig. 9: Long corridor views with poor texture in b) and c) that causes real-time ORB-SLAM to break.

Fig. 10: SfM and ORB-SLAM vs O-POCO in the TUM [14] long office sequence. We see some drift introduced in SfM and O-POCO due to the standard mapping process.

We generate a BoW from the descriptors in the full map and take the histogram of words as a distribution of the descriptor information; then, we obtain the histogram of words in the compressed map and use the relative entropy to measure of the distance between such two distributions ([16]), as follows

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \quad (1)$$

where $p(x)$ is the histogram in the resulting compressed map and $q(x)$ the distribution in the full uncompressed map. We tested two sequences from the same area, the long corridor, where we obtained for each sequence a full and a compressed map.

We performed the following analysis. From the two different explorations in the same long corridor, for each full map – i.e., before compression – we obtained the word vocabulary (bag of words, or BoW), and with them we calculated a histograms of words (HoW) for the compressed maps: we compared and cross-compared vocabularies to generate those histograms, having an average entropy of 0.019 in the first case (using the same sequence for both the BoW and HoW) and 0.034 in the latter (using different sequences for the BoW and HoW).

Therefore, there is no loss of information between the original map and their compressed version either from the same sequence or a different one – big values would mean that we are not observing all the same words after compressing or re-exploring the same area; in this case, values close to zero mean that we are not loosing descriptor information after compressing.

2) Map-feature occupancy grid: Here we tested the occupancy prevalence in the compressed map in relation with the full map. We use a cell size of 1 centimeter per side and

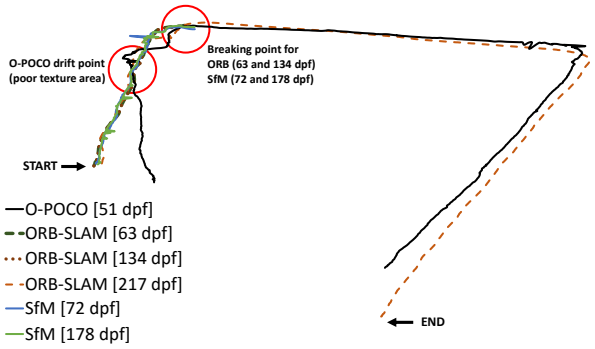


Fig. 11: SfM and ORB-SLAM vs O-POCO. Long corridor test II. Different ORB-SLAM configurations were tested, where only offline ORB-SLAM seems to performs well; our online algorithm suffers from some drift in the poor textured areas but keeps working – we believe that an optimization process will help to correct some of this drift.

(a) (b) (c) (d)

Fig. 12: Typical views in the lab sequence used to measure the relocalisation performance.

conventional binary similarity measures ([17]). Results are shown in Table II.

We can see that most of the occupied space is preserved in the compressed map version, with respect with the full point cloud from the same sequence, within a grid size of 1cm per side, which is useful for navigation. We then compared the compressed version with a full re-exploration map from the same area, after a registration process, and we got an intersection around of 40 per cent, within a 10cm cell size, despite the randomness in the feature detection process and point cloud alignment errors, as shown in Figure 14. This means that, beyond alignment problems, the mapping process in two different runs takes almost half of the same points.

TABLE II: Occupancy grid intersection [in per cent]. We compared two sequences from a long corridor, A and B, where we compares compressed map with full map spatial properties.

map	testing	intersection (1cm)	intersection (10cm)
map_A	full_A	97.0	99.9
map_A	full_B	4.1	40.6
map_B	full_A	3.6	39.2
map_B	full_B	97.0	99.9

3) Point cloud's geometry error: Here we measure the map geometry preservation between the compressed and full (non-compressed) maps, as described in [1], by the RMS geometrical error. (Figure 15). Similarly to the occupancy

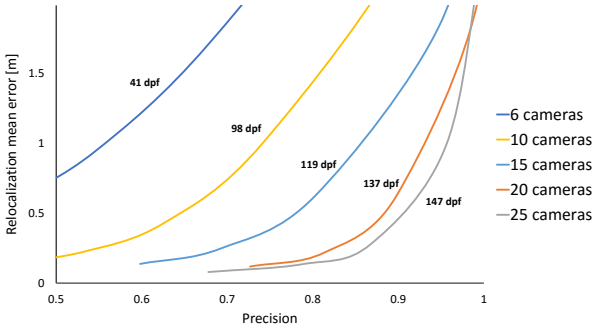


Fig. 13: O-POCO's relocalisation and compression performance versus precision rate – i.e. proportion of correctly relocated cameras – at different sub-map window sizes.

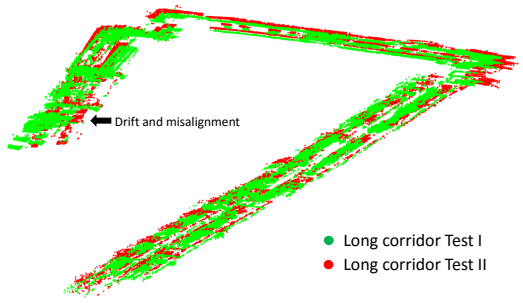
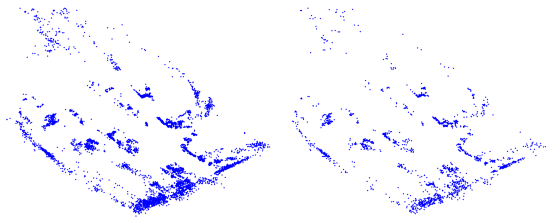


Fig. 14: Long corridor Test I and II point cloud registration. We observe some misalignment that shifts the registration.

error, we tested the two long corridor sequences, generating both full and compressed maps for each of them. We first compare the geometry between full maps and their self-compressed version, where no perceptible error was detected (0.003 meter error); then, we cross-compare a full map from one sequence with a compressed map from a different sequence – after a registration process –, having an average error of 0.525 meters.

We observe that the geometry is highly preserved when compressed. This is validated when a second sequence navigating the same space preserves the global geometry with an error of around 50 cm. This implies the method is consistently performing its compression approach.



(a) (b)

Fig. 15: Comparison between a) full and b) compressed Point Cloud (lab sequence upper view). Even though a decrease in density is clear, we observe a preservation in geometry.

V. CONCLUSIONS

Using classical mapping via SfM and ICP, we present O-POCO, an Online POint cloud COmpression technique useful for visual odometry and SLAM. A windowed sub-mapping is applied to perform map-feature selection to compress the map. We show cases when O-POCO outperforms SfM and ORB-SLAM in visual odometry in poorly textured areas due to the proposed mapping and then compressing strategy. We evaluate with both publicly available datasets and our own. We demonstrate O-POCO compression performance by introducing a *descriptors per key-frame* unit and compare several sub-mapping window sizes and baselines configurations, outperforming even in cases when we use four times less information. Furthermore, we performed a rigorous analysis of its effectiveness in relocalisation and information conservation, where we got low relocalisation mean error (about 20 cm), attributable to the descriptor entropy preservation; at the same time, spatial and geometry information is also preserved, which is useful for tasks beyond relocalisation such as path planning and navigation.

REFERENCES

[1] L. Contreras and W. Mayol-Cuevas, "Trajectory-driven point cloud compression techniques for visual slam.," in *IROS*, IEEE, 2015.

[2] M. Pauly, M. Gross, and L. P. Kobbelt, "Efficient simplification of point-sampled surfaces," in *Proceedings of the Conference on Visualization*, pp. 163–170, IEEE, 2002.

[3] P. F. Alcantarilla, K. Ni, L. M. Bergasa, and F. Dellaert, "Visibility learning in large-scale urban environment.," in *International Conference on Robotics and Automation*, pp. 6205–6212, IEEE, 2011.

[4] T. Sattler, B. Leibe, and L. Kobbelt, "Improving image-based localization by active correspondence search," in *European Conference on Computer Vision*, pp. 752–765, Springer-Verlag, 2012.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "Orb-slam: a versatile and accurate monocular slam system.," *Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] D. Galvez-Lopez and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[7] K. Pirker, M. Rther, and H. Bischof, "Cd slam - continuous localization and mapping in a dynamic world.," in *International Conference on Intelligent Robots and Systems*, pp. 3990–3997, IEEE, 2011.

[8] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.

[9] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. T. Furgale, "The gist of maps - summarizing experience for lifelong localization.," in *ICRA*, pp. 2767–2773, IEEE, 2015.

[10] M. Dymczyk, S. Lynen, M. Bosse, and R. Siegwart, "Keep it brief: Scalable creation of compressed localization maps.," in *IROS*, pp. 2536–2542, IEEE, 2015.

[11] S. Lynen, M. Bosse, P. T. Furgale, and R. Siegwart, "Placeless place-recognition.," in *International Conference on 3D Vision*, pp. 303–310, IEEE, 2014.

[12] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection.," in *ECCV*, pp. 430–443, Springer, 2006.

[13] D. Rogers, *An introduction to NURBS: with historical perspective*. Morgan Kaufmann, first ed., 2001.

[14] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems.," in *IROS*, pp. 573–580, IEEE, 2012.

[15] G. Bradski *Dr. Dobb's Journal of Software Tools*.

[16] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[17] S. seok Choi and S. hyuk Cha, "A survey of binary similarity and distance measures," *Journal of Systemics, Cybernetics and Informatics*, pp. 43–48, 2010.