A stereo vision geometric descriptor for place recognition and its GPU acceleration for autonomous vehicles applications

Mohammed CHGHAF[1], Sergio RODRIGUEZ FLOREZ[1], Abdelhafid EL OUARDI[1]

[1]Laboratoire SATIE, ENS Paris-Saclay, CNRS, Université Paris-Saclay,
Bâtiment 660 DIGITEO, rue Noetzlin, 91190, Gif-sur-Yvette, France
Mohammed.Chghaf@universite-paris-saclay.fr, Sergio.Rodriguez@universite-paris-saclay.fr,
Abdelhafid.Elouardi@universite-paris-saclay.fr

Résumé – Dans cet article, nous présentons un descripteur global géométrique dédié à la reconnaissance de lieux basé sur des nuages de points issues d'une une caméra stéréo pour des applications de véhicules autonomes. Nous présentons d'abord l'approche utilisée pour enregistrer la structure 3D de l'espace visible. Ensuite, nous proposons une optimisation paramétrique pour obtenir les meilleures performances en couplant les caractérisitiques intrinsèques du capteur utilisé (caméra stéréo) et l'algorithme utilisé. Enfin, nous proposons une implémentation optimisée GPU basée sur CUDA. Par rapport à un CPU, les temps de traitement sur GPU sont accélérés 7 fois sur une Jetson AGX Xavier et 30 fois avec une GeForce RTX 3080.

Abstract – In this paper we present a geometric global descriptor dedicated for place recognition in autonomous vehicles applications that is based on stereo camera generated point clouds. We first present the approach used to record the 3D structure of the visible space. Then, we propose a parametric optimization to achieve the best performance by coupling the dedicated sensor (Stereo Camera) and the used algorithm. Finally, we propose a GPU implementation based on CUDA. Compared to a CPU, processing times on GPU are accelerated 7 times on a Jetson AGX Xavier and 30 times using a GeForce RTX 3080.

1 Introduction

Simultaneous Localization and Mapping (SLAM) has been widely studied over the last years for autonomous ground vehicles (AGV). Place recognition in particular is a core problem in SLAM. In fact, recognizing the past places and adding loop pose constraints to the pose graph can effectively reduce the cumulative error and improve the positioning accuracy of an AGV. Many approaches have been proposed in the literature to treat the place recognition problem. In the context of AGV, it is often based on one of these two sensors: Camera or LiDAR.

Existing Cameras provide a large amount of possibilities both in quality and quantity of the relevant data to be used for loop closure detection. Using this sensor, different approaches have been proposed. Image-to-image place recognition is commonly carried out using the Bag-of-Words model [1]. In general, first, features (like ORB [2]) are extracted from selected keyframes. These features are then grouped into vectors and used to select loop closure candidates following the BoW scheme [3].

Unlike Camera-based place recognition, LiDARs have the ability to generate rich geometrical information in textureless environments. Existing methods mainly focus on the extraction of the structural information from point clouds and using it as a global descriptor to find loop candidates. Scan Context [4] is a global descriptor that encodes the 3D point cloud into a matrix describing the structural information of the scene based on the height, the azimuthal and the radial information of the points. Alternatively, Tomono et al. in [5] used geometric segments (e.g. planes and lines) extracted from the point cloud to create a virtual scan. These scans are then used to find correspondences between segments in order to achieve loop detection.

Although Camera-based place recognition is efficient in environments that contain rich features, their efficiency is limited when the scenes lack of textures. On the other hand, LiDAR-based place recognition can take advantage of the structural information present in the scene. Nevertheless, LiDAR-based systems remain more expensive than Camera-based systems.

In this paper, we present the following main contributions:

- Adaptation of the LiDAR Scan Context to stereo camera point cloud context.
- A parametric optimization protocol of sensor-algorithm coupling to achieve better performance in terms of loop detection accuracy.
- Implementation and acceleration of the proposed global descriptor on GPU using CUDA

The remainder of this paper is structured into five sections, organized as follows: Section 2 presents the problem statement. In Section 3 we discuss the parameters' analysis process and the metrics used for the evaluation. Section 4 presents the GPU implementation of the proposed approach. Section 5 provides a comparative analysis of the reported results. Further, a conclusion and perspectives are drawn.

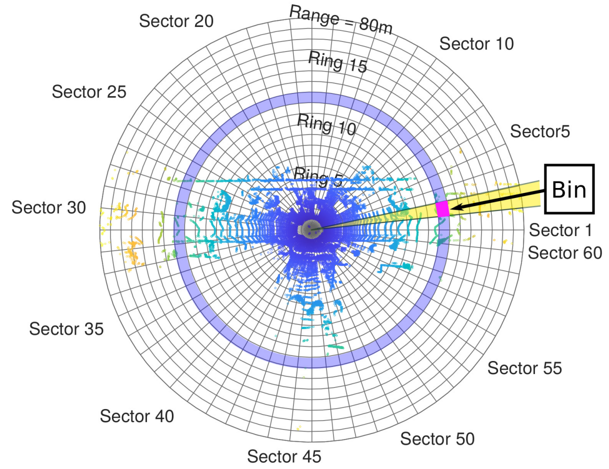

Figure 1: The partition applied to LiDAR scan. It is divided into bins (like pink area) according to azimuthal (from 0 to 2π) and radial (from center to maximum range) information. The yellow and blue areas are the sector and the ring respectively.

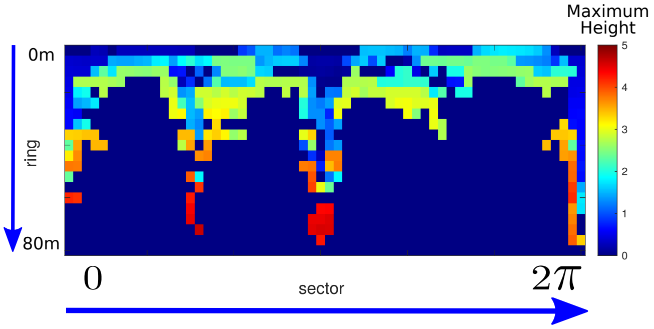

Figure 2: Illustration of the Scan Context.

2 Problem presentation

The *Scan Context* is a global descriptor used for LiDAR point-clouds. First, the 3D scan is divided into azimuthal and radial bins in the sensor coordinate and in an equally spaced fashion. Figure. 1 shows an illustration of this division. Each scan acts as a global keypoint and thus is referred to as an egocentric place descriptor. N_s and N_r are the number of sectors and rings, respectively (Fig. 2 shows an illustration of a created *Scan Context*). In this case, the maximum sensing range of a LiDAR sensor is $L_{max} = 80m$, the radial gap between rings is $\frac{L_{max}}{N_r}$ and the central angle 2π of a sector is equal to N_s . In this illustration, the parameters are fixed $N_s = 60$ and $N_r = 20$, same as the authors in [4]. In the created matrix, the value of the bin is determined by the maximum height value of the points inside the bin.

Due to the limited FoV and the limited range of Cameras, the fixed parameters for LiDAR are not adapted to the camera context. Indeed, the FoV and the range must be further discretized to obtain a reliable representation of the current scene. The Figure 3 shows the limitations of the Field-of-View of Stereo Camera generated point cloud and its maximum range along with the corresponding captured image. The geometrical information of points detected on the left buildings are well represented in Sectors 1 to 3 (colored in yellow). On the other hand, the electrical pole and the fence on the right are represented by the bins in Sectors 55 and 56 (colored in pink).

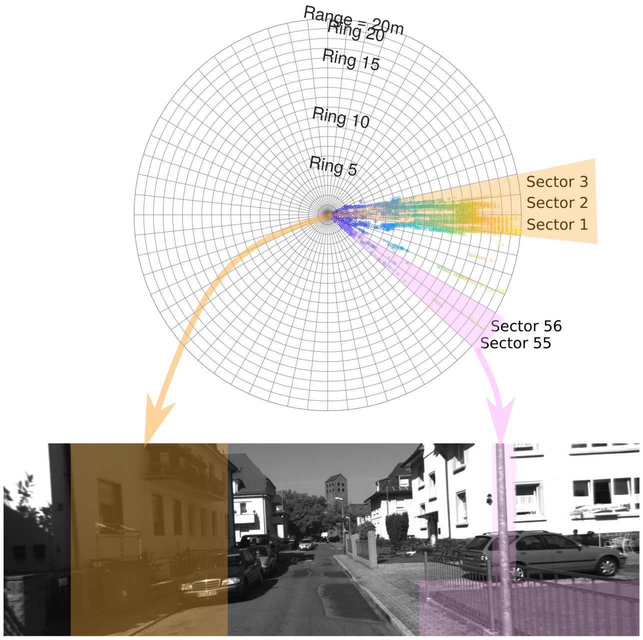

Figure 3: Illustration of the partition applied to 3D scan from Stereo Camera and its corresponding left image.

By contrast, increasing the number of sectors and rings used in the process will significantly increase the computing time of the descriptor. Therefore, reducing the overall performance of the system. Hence, an algorithm architecture co-design approach is necessary to harness the capabilities of this descriptor without losing in timing performances.

3 Parametric optimization

In order to decide whether a place is revisited or not, the *Scan Context distance* is computed. This distance has to satisfy an acceptance threshold for the place to be classified as revisited. The *Scan Context* is represented as a matrix and the *Scan Context distance* between frame t and k is the sum of distances between columns at a same index. A cosine distance is used to compute a distance between two column vectors at the same index, c_i^t and c_i^k. In addition, the sum is divided by the number of sectors for normalization, like in Equ. 1.

$$d(SC_t, SC_k) = \frac{1}{N_s} \sum_{i=1}^{N_s} \left(1 - \frac{c_i^t \cdot c_i^k}{||c_i^t|| \cdot ||c_i^k||} \right) \tag{1}$$

Our parametric optimization protocol of sensor-algorithm coupling consists of evaluating the algorithm on different sequences with different fixed numbers of sectors and rings. The accuracy of each configuration is analyzed using the precision-recall

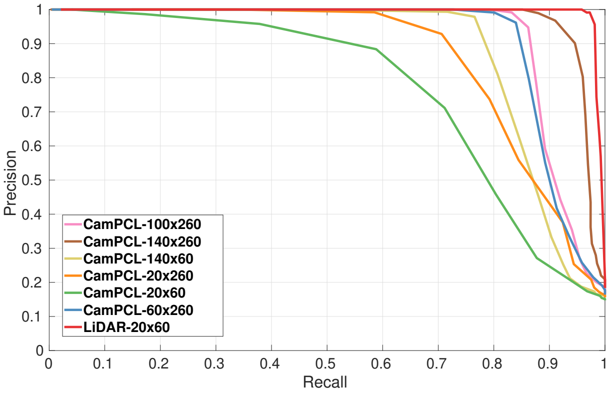

Figure 4: Precision-recall curve for the Sequence 05 of KITTI dataset [6].

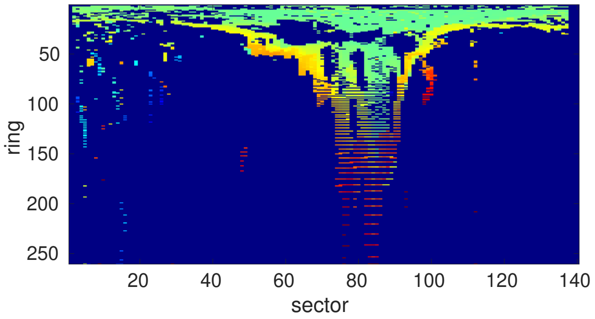

Figure 5: Illustration of the Scan Context adapted to point cloud generated from Stereo Camera.

curve. This curve shows the trade-off between precision and recall for different thresholds. In our context, we vary the acceptance threshold as a revisited place in the range of 0 and 1. A large area under the curve indicates a high recall and a high precision. High precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier achieves accurate results (high precision), and returns mostly all positive results (high recall).

Figure. 4 shows the result of this analysis applied on the sequence 05 of the KITTI dataset [6]. *CamPCL* means that it is a point cloud obtained from Camera. The first following number is the number of the rings. The second following number is the number of sectors. Overall, LiDAR-based Scan Context still outperforms the stereo Camera-based descriptor. However, by increasing the number of sectors and rings used to describe the scene, we can achieve comparable accuracy to the LiDAR-based Scan Context. Despite the fact that it is limited in its Field-of-View (90°) and in its range (20 m), the parametric study shows that fixing $N_s = 260$ and $N_r = 140$ can guarantee satisfying results for low-cost applications. Figure. 5 shows an illustration of the *Scan Context* created from a stereo camera generated point cloud where the number of sectors was fixed to 140 and the number of rings was fixed to 260.

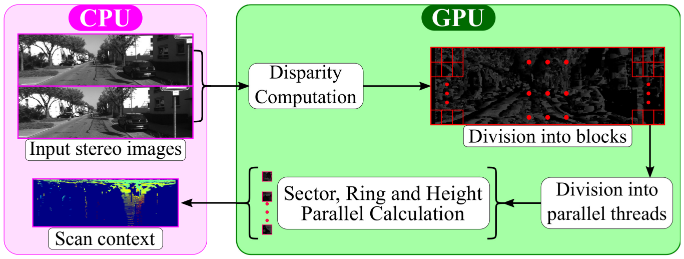

Figure 6: Global parallelization scheme of the developed approach.

4 GPU Implementation

The Scan Context creation based on point cloud generated from stereo camera can be efficiently parallelized on the GPU since all the calculations can be accomplished out independently.

In order to generate this descriptor, we first calculated the disparity map. We used the CUDA implementation proposed by Hernandez-Juarez et al. in [7].

The next step is to generate the matrix representing the *Scan Context* based on the generated point cloud from the disparity map. By taking advantage of the SIMT (Single Instruction Multiple Threads) programming model offered by CUDA [8] for NVIDIA GPUs, computing the *Scan Context* can be carried out by launching many *kernels* executed in parallel by all the threads on the CUDA-Cores. The same *kernel* is executed in parallel by all the threads. With the CUDA architecture, threads are organized hierarchically. They are indeed grouped into *thread blocks*, themselves grouped into *thread block grids*.

The proposed GPU implementation starts by computing the disparity map using the input stereo images. The obtained disparity map is then divided into blocks that regroup multiple threads. Finally, the implemented *kernel*, calculates the depth, the azimuth angle and the height of each pixel in the disparity map. The computation results of these parallel threads are then used to build the scan context. Figure. 6 illustrates the proposed pipeline and the functional blocks that were parallelized on GPU. Algorithm 1 presents the pseudocode of the proposed kernel. The $blockDim$ is fixed depending on the dimensions of the disparity map created.

5 Results

The proposed approach was tested on the KITTI dataset [6]. Each image is of dimensions 1241 × 376. The experiments using a CPU implementation are carried out on a laptop with an Intel i7-11800H CPU (2.30GHz) and 32GB memory. The GPU implementation was performed using a GeForce RTX 3080 (1.37 GHz, 16 GB memory and 6144 CUDA-Cores) and a Jetson AGX Xavier (1.38 GHz, 16 GB and 512 CUDA-Cores). In these implementations, disparity maps were divided into a grid of 39×47 blocks of 256 threads. Table 1 summarizes the timing performances achieved using various *CamPCL Scan Context* settings.

Algorithm 1: Pseudocode of the Scan Context Calculation Kernel

```
Input  : $Disp$ = Disparity map
Output: $SC$ = Scan Context Descriptor
$i = blockDim.x \times blockIdx.x + threadIdx.x$
$j = blockDim.y \times blockIdx.y + threadIdx.y$
if $(i < Disp.rows)$ and $(j < Disp.cols)$ then
    if $(Disp(i,j) > 0)$ and $(Disp(i,j) < MaxDisp)$ then
        Depth = ComputeDepth(Disp(i, j));
        if $(Depth < 20)$ then
            height = computeHeight(Depth, i);
            ringId = determinRing(Depth, j);
            sectorId = determinSector(Depth, j);
            if $(SC(ringId, sectorId) < height)$ then
                $SC(ringId, sectorId) = height$;
            end
        end
    end
end
```

Overall, we successfully reduce the computation time by 7 orders of magnitude using the Jetson AGX and 30 orders of magnitude usning the RTX 3080. In this study, we focus mainly on the performance achieved by parallelizing all the computational load on GPU. Indeed, the CPU-GPU architecture needs additional time to upload the stereo images from the CPU to the GPU and then to download the computed *Scan Context* to the CPU. In our case, this transfer time is estimated on average to be 67ms, but it will strongly depend on the architecture of the deployed system. It will vary according to the CPU-GPU interface bus and the memory access time.

6 Conclusion

In this work, we presented an adaptation of the LiDAR Scan Context to stereo camera point cloud context. First, a parametric optimization and study was carried out to determine the sensor-algorithm coupling and to find the best combination of parameters necessary to achieve better performance in terms of loop detection accuracy. Second, based on these parameters, we showed that we can achieve satisfying results in loop closure detection for a low-cost application. Finally, a GPU implementation was proposed that takes into account the parameters specified by the previous sensor-algorithm optimization protocol. We show that a clear improvement in performance is achieved at a lower overall cost. In perspective, an in-depth study will focus on the application of the same approach by combining both Camera and LiDAR data. In particular, we will propose a SLAM system dedicated for autonomous ground vehicle using both LiDAR and Camera. Future work will also focus on the extension of this geometric descriptor to odometry rather than only place recognition. In order to meet real-time requirements of an embedded system, an algorithm-architecture mapping approach will be necessary.

Dimensions of	CPU	Jetson	GeForce
Stereo vision		AGX Xavier	RTX 3080
Scan Context	(time in ms)	(time in ms)	(time in ms)
20×60	223.92	32.38 **(x6.91)**	7.41 **(x30.21)**
20×260	228.09	34.32 **(x6.64)**	8.35 **(x27.31)**
60×260	235.97	35.69 **(x6.61)**	8.41 **(x28.05)**
100×260	256.35	36.84 **(x6.95)**	8.37 **(x30.62)**
140×60	258.81	33.34 **(x7.76)**	8.63 **(x29.98)**
140×260	280.57	37.06 **(x7.57)**	8.76 **(x32.02)**

Table 1: Comparison of the per-frame processing time on CPU, on a laptop GPU (the GeForce RTX 3080) and on an GPU for embedded applications (Jetson AGX Xavier)

.

References

[1] NISTER, David et STEWENIUS, Henrik. Scalable recognition with a vocabulary tree. In : 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). IEEE, 2006. p. 2161-2168.

[2] RUBLEE, Ethan, RABAUD, Vincent, KONOLIGE, Kurt, et al. ORB: An efficient alternative to SIFT or SURF. In : 2011 International conference on computer vision. IEEE, 2011. p. 2564-2571.

[3] GÁLVEZ-LÓPEZ, Dorian et TARDOS, Juan D. Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics, 2012, vol. 28, no 5, p. 1188-1197.

[4] KIM, Giseop et KIM, Ayoung. Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. In : 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2018. p. 4802-4809.

[5] TOMONO, M. Loop detection for 3D LiDAR SLAM using segment-group matching. Advanced Robotics, 2020, vol. 34, no 23, p. 1530-1544.

[6] GEIGER, Andreas, LENZ, Philip, et URTASUN, Raquel. Are we ready for autonomous driving? the kitti vision benchmark suite. In : 2012 IEEE conference on computer vision and pattern recognition. IEEE, 2012. p. 3354-3361.

[7] HERNANDEZ-JUAREZ, Daniel, CHACÓN, Alejandro, ESPINOSA, Antonio, et al. Embedded real-time stereo estimation via semi-global matching on the GPU. Procedia Computer Science, 2016, vol. 80, p. 143-153.

[8] BUCK, Ian. Gpu computing with nvidia cuda. In : ACM SIGGRAPH 2007 courses. 2007. p. 6-es.