High-Performance Feature Extraction for GPU-accelerated ORB-SLAMx

Filippo Muzzini*, Nicola Capodieci*, Roberto Cavicchioli†, Benjamin Rouxel*‡
*Dep. of Physics, Informatics and Mathematics - University of Modena and Reggio Emilia - Modena, Italy
†Dep. of Communication and Economics - University of Modena and Reggio Emilia - Modena, Italy
‡StatInf - Paris, France

{filippo.muzzini, nicola.capodieci, roberto.cavicchioli}@unimore.it, benjamin.rouxel@statinf.fr

***Abstract*—In the autonomous vehicles field, localization is a crucial aspect. While the ORB-SLAM algorithm is a recognized solution for these tasks, it poses challenges due to its computational intensity. Although accelerated implementation exists, a bottleneck persists in the Point Filtering phase which relies on the Distribute Octree algorithm that is not suitable for GPU processing. In this paper, we introduce a novel GPU-suitable algorithm designed to enhance the Point Filtering step, surpassing Distribute Octree. We conducted a comprehensive comparison with state-of-the-art CPU and GPU implementations, considering both computational time and trajectory accuracy. Our experimental results, demonstrate significant speed-ups up to 3x compared to previous contributions.**

***Index Terms*—GPU, ORB-SLAM, CUDA, Parallel**

I. Introduction

Autonomous vehicles (AVs) must use sensors and algorithms to perceive the external environment to localize themselves in the world. Systems like ORB-SLMA2 [1] and ORB-SLAM 3 [2] use cameras to extract features useful to localize the system. AVs systems often run on embedded boards with limited computational power. For this reason, the GPU is exploited in [3] resulting in the actual stat-of-the-art implementation of ORB-SLAM in terms of execution time. The main bottleneck is the *Point filtering* phase. We propose a novel parallel algorithm for this phase and we implement it on GPU. Our code is publicly available[1]. We evaluated our proposal against the state-of-the-art CPU and GPU implementations and we see a speed-up of up to 3x with respect to the original implementation.

II. Our novel Implementation

We focus on the *Point Filtering* phase of ORB-SLAM (for deatails see [1]) that is performed using the *Distribute Octree* algorithm on CPU also in [3] since this algorithm is not suitable for GPU implementation. We propose a new method to filter points using the GPU accelerator and we call it *Cluster-filter*. *Cluster-filter* is articulated in five steps: **Centroid** We pre-compute the cluster centroids' location based on the image size. The centroids are equally distanced to cover the image surface. They are uniformly distributed across the image, roughly forming a grid. The number of centroids corresponds to the number of needed features. **Cluster Assignment** We launch a kernel with a thread for every point extracted by FAST (see [1] for details). The kernel computes the point distance to every pre-computed centroid and links the point to the nearest one. In the end, every point is assigned to one centroid to form a cluster, and more points can be assigned to the same cluster. **Compute the Max Score** For each cluster, we identify the points with the highest score calculated by FAST. We launch a thread for every point of each level to perform a CUDA *atomicMax* with the point score. The maximum score value is assigned to the cluster. **Max score point selection** With another GPU kernel, we filter out points within clusters to only keep the point with the cluster score. Each thread for each point compares the score of the point with the score of the cluster, if the two values correspond, then the point is the best for this cluster. Otherwise, the point is marked as not selected. In case multiple points match the maximum cluster score, we randomly pick a point among them. **Last Cluster Assignment** A centroid can have no points assigned to itself. If all features are in a small region, they will be assigned to the same cluster. Since we need too many features as the cluster amount, a CUDA kernel is launched having a GPU thread for each cluster. Each thread checks if its corresponding cluster has at least a point assigned to it. In this case, it terminates; otherwise, it will iterate on the unassigned points, calculate the distance between each point and the cluster centroid, and assign to the cluster the closest one. Differently from [3], our method runs on GPU, which means it can exploit massive parallelism reducing execution time and the amount of copies between CPU and GPU.

[1]https://git.hipert.unimore.it/fmuzzini/cuda-accelerated-orb-slam

III. Experiments

We compared our implementation with the implementations of the original *ORB-SLAM2* [1] and the GPU implementation (GPU-SotA) [3] using Kitti dataset [4]. Moreover, we compared our implementation with *ORB-SLAM3* [2] original implementation and the GPU implementation (GPU-SotA) [3] using EuroC dataset [5]. We measured the computational time needed by the *Tracking* phase (macro-phase in which *Point Filtering* is placed) to process a single frame in the *Monocular* and *Stereo* versions. We also measured the trajectory errors (ATE) using the same strategies as in [3]. We ran experiments on Nvidia Xavier AGX board (512 NVIDIA cores, 8 cores ARM CPU, 32GB RAM). Table I reports the results of ORB-SLAM2. We can see that our implementation has a speedup of up to 3.2x concerning the CPU implementation and up to

979-8-3503-4859-0/DATE24/© 2024 EDAA

TABLE I: ORB-SLAM2 Results for KITTI dataset

		Stereo			Monocular		
		ORB-SLAM2	GPU-SotA	Our	ORB-SLAM2	GPU-SotA	Our
KITTI04	ATE (m)	0.805	**0.571**	0.977	1.645	0.495	**0.374**
	mean time (s)	99.090	76.0080	**70.4699**	50.5232	18.7462	**15.7954**
	min time (s)	87.643	66.0580	**61.0216**	44.1417	13.4232	**12.7979**
	max time (s)	112.162	196.5720	**89.0281**	64.7201	131.9220	**28.5660**
KITTI06	ATE (m)	2.688	**1.391**	2.594	**13.443**	22.953	110.478
	mean time (s)	95.856	80.0650	**70.6078**	51.0668	25.9693	**17.0879**
	min time (s)	77.926	57.4509	**54.3523**	41.4550	12.1427	**11.3211**
	max time (s)	450.564	**447.8280**	448.7160	311.4790	241.1960	**211.2260**
KITTI07	ATE (m)	**0.871**	1.380	0.957	**4.299**	4.768	4.551
	mean time (s)	92.128	75.8203	**71.3601**	52.5821	**18.1844**	21.2995
	min time (s)	74.167	62.4044	**56.7317**	40.8354	12.4471	**12.0155**
	max time (s)	**319.958**	359.1350	352.4080	310.9100	262.6870	**233.4490**

TABLE II: ORB-SLAM3 Results for Euroc dataset

	Stereo				Monocular			
	ATE (m)	mean time (s)	min time (s)	max time (s)	ATE (m)	mean time (s)	min time (s)	max time (s)
ORB-SLAM3	0.037	57.068	44.941	101.260	3.059	29.555	19.388	254.500
GPU-SotA	0.064	33.021	21.843	997.094	**1.101**	19.095	11.720	425.650
Our	**0.035**	**31.115**	**21.790**	**53.786**	3.170	**18.352**	**11.041**	**138.960**

1.1x concerning the GPU implementation. Our implementation shows promising results in *Stereo* trajectory error, it does not outperform GPU-SotA but the errors are aligned with the original *ORB-SLAM2* results. Table II reports the results of ORB-SLAM3. Our implementation shows a speedup of up to 1.8x concerning the CPU implementation and 1.1x concerning the GPU implementation. In terms of trajectory errors, our implementation shows the best results for the ATE in *Stereo* case. For the *Monocular* version, errors are aligned with the original *ORB-SLAM3*.

Our method, differently from the *Octree* method that takes into account the distribution of the actual features, assumes features are equally distributed. This can lead to errors since some features extracted in a frame might not be extracted in the second. On the other hand, this effect is mitigated in *Stereo* settings in which our method exhibits less computational time making itself a good choice for time-critical scenarios.

IV. Conclusion

We proposed a novel point filtering method for ORB-SLAM systems suitable to be performed on GPU. Our implementation outperforms the SotA implementation in terms of execution time. In future research, we intend to examine the impact of the Unified Memory feature on this system.

References

[1] R. Mur-Artal and J. D. Tard'os, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[2] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[3] F. Muzzini, N. Capodieci, R. Cavicchioli, and B. Rouxel, "Brief announcement: Optimized gpu-accelerated feature extraction for orb-slam systems," in *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3558481.3591310

[4] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[5] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract