Supplementary Material

Jie Ren*[1,2][0000−0003−3332−2092], Wenteng Liang*[1][0000−0002−3750−7695], Ran Yan†[1][0000−0001−6705−1644], Luo Mai[2][0000−0002−3594−1092], Shiwen Liu[1][0000−0003−1779−409X], and Xiao Liu[1][0000−0002−6073−030X]

[1] Megvii Inc., China
[2] The University of Edinburgh, United Kingdom

We provide examples to describe the JecVector data structure and the edge-based BA problem partitioning method. Further, we provide extra experimental results, including the full BAL datasets [2], the 1DSfM dataset [8] and a large synthetic dataset. We finally provide reconstruction plots to illustrate the effects of MegBA.

1 JetVector Design

To show the novel design of *JetVector*, we compare it with its predecessor: *Jet* implemented in prior BA libraries such as Ceres [1].

A *Jet* object comprises *value* and *grad* (i.e., gradients). It has atomic operators such as $+$, $-$, $*$, and / which are essential for auto-differentiation. Multiple Jet objects are contained in an Array-of-Structures (AoS). Though effective for CPUs, the use of AoS makes it difficult to coalesce GPU memory transactions, resulting in low efficiency in utilising GPUs.

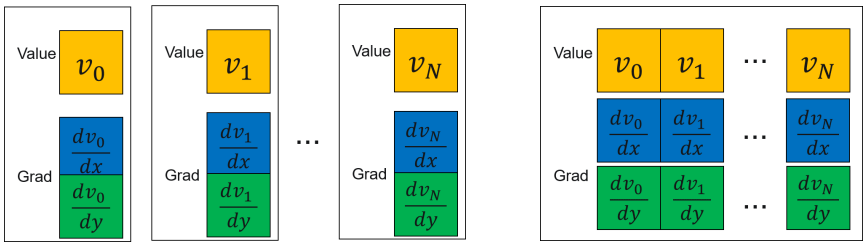
In contrast, *JetVector* is implemented as a Structure-of-Arrays which allows GPU memory transactions to be effectively coalesced. Figure 1 compares Jet and JetVector. As we can see, if we stack *Jet* in a vector, the memory addresses of *Value* and *Grad* of different each *Jet* objects are not continuous. In *JetVector*, however *Value* and *Grad* objects are organised in a continuous manner in the memory, making it friendly for GPUs.

We show how does MegBA add JetVector objects in Figure 2. Supposing there are N *Jet*s in *JetVector*. We launch N GPU threads and each GPU thread processes a *Jet* object. In this way, GPU threads will access adjacent data elements, thus coalescing memory transactions. As a result, the memory loading/storing operations in a warp (a warp is 32 consecutive GPU threads) can be achieved by a 128-byte memory transaction. Notably, the memory transactions of a naive AoS structure comprises multiple serialised 32-byte memory transactions, which can adversely affect memory performance.

2 Edge-based Partitioning of BA Problems

We provide an example that shows how MegBA partitions non-zero Hessian blocks and assigns the partitions to multiple GPUs in a load balancing manner.

* Equal contribution, work was done during their internship in Megvii Inc.
† Corresponding author.



(a) Data structure of *Jet* (b) Data structure of *JetVector*

Fig. 1: Comparison between *Jet* and *JetVector*.

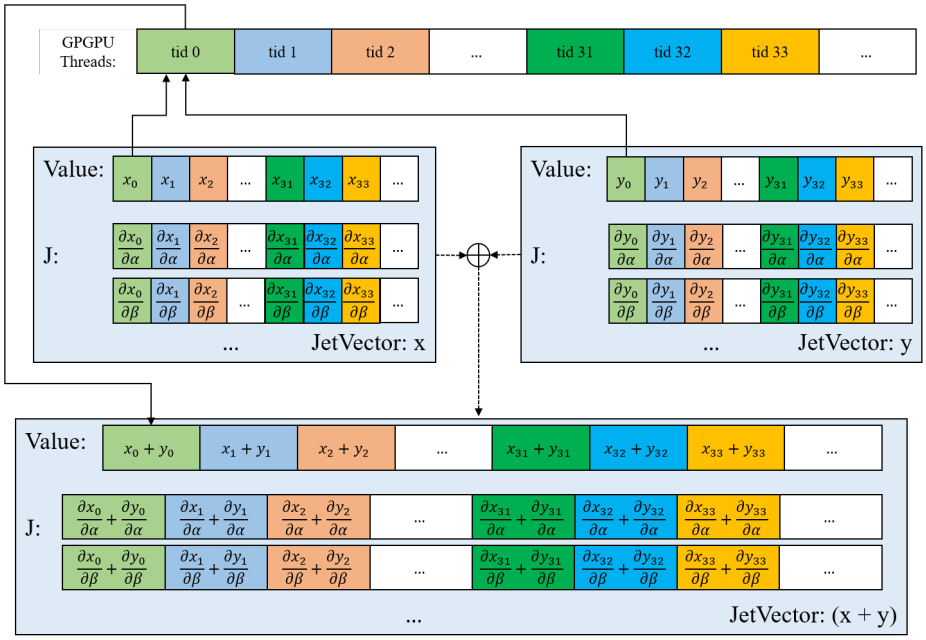
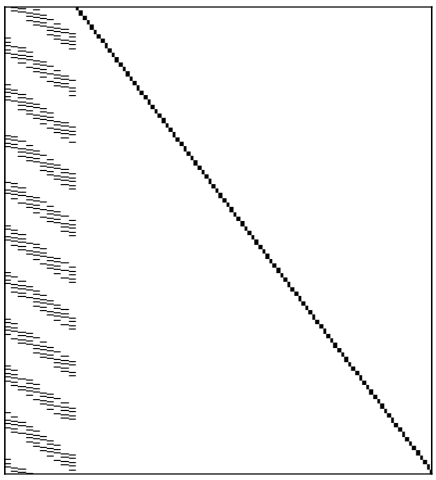


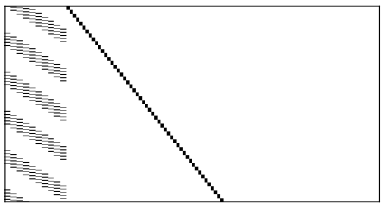
Fig. 2: Example of adding two *JetVector* objects.

Figure 3a shows the Jacobian structure of a BA problem. The black blocks refer to non-zero blocks and the white blocks refer to zero blocks. It is partitioned into two sub-matrices shown in Figure 3b and Figure 3c.

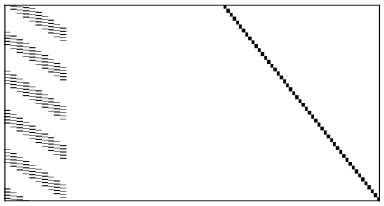
Figure 4a is the structure of the Hessian matrix to be computed by MegBA. A Hessian, though stored in the sparse format, is still too large to fit into a GPU. To address this, MegBA stores a part of a Hessian in each GPU, as shown in Figure 4b and Figure 4c. Since we have $H = H_0 + H_1$, it is guaranteed that computations based on partitioned Hessian sub-matrices are equivalent to the original computation.



(a) Jacobian J

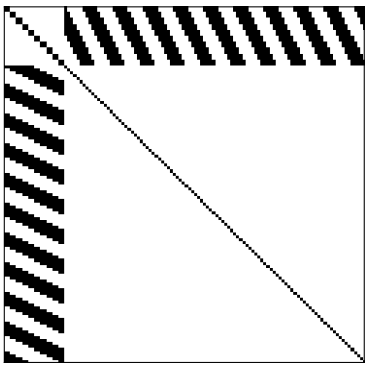


(b) Partitioned Jacobian J_0 on GPU 0

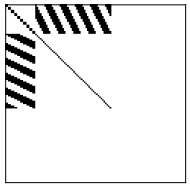


(c) Partitioned Jacobian J_1 on GPU 1

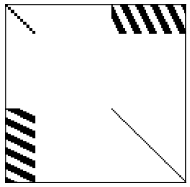
Fig. 3: A Jacobian matrix. (a) is the Jacobian of a BA problem. We partition Jacobian into two matrix blocks (b) and (c). MegBA stores these two matrix blocks on two GPUs.



(a) Hessian H



(b) Hessian H_0 on GPU 0



(c) Hessian H_1 on GPU 1

Fig. 4: A Hessian matrix. (a) is the Hessian H constructed from Jacobian J that $H = J^T J$. On each GPU, MegBA constructs a Hessian matrix so that $H_0 = J_0^T J_0, H_1 = J_1^T J_1$.

	Ceres-16		DeepLM		g2o-16		RootBA-16		MegBA-1-a		MegBA-1-m	
	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time
Trafalgar-21	0.83	0.42	0.83	1.72	0.83	0.79	0.83	**0.16**	0.83	0.58	**0.83**	0.50
Trafalgar-39	0.95	1.02	0.95	2.63	0.95	2.52	0.95	**0.38**	0.95	0.88	**0.95**	0.81
Trafalgar-50	0.70	1.04	0.70	2.62	0.70	1.90	**0.70**	**0.44**	0.70	0.97	**0.70**	0.81
Trafalgar-126	0.62	3.36	0.62	3.98	0.62	8.76	0.62	8.06	0.62	1.83	**0.62**	**1.52**
Trafalgar-138	0.53	8.11	0.53	4.00	**0.52**	12.90	0.52	11.88	0.53	1.18	0.53	**1.03**
Trafalgar-161	0.47	4.76	0.47	4.11	**0.46**	9.04	0.46	14.64	0.47	1.12	0.47	**1.11**
Trafalgar-170	0.47	6.77	0.47	4.00	**0.46**	14.23	0.46	20.65	0.47	1.40	0.47	**1.14**
Trafalgar-174	0.47	5.63	0.46	4.17	**0.45**	23.42	0.46	19.35	0.46	1.22	0.46	**1.19**
Trafalgar-193	0.46	7.13	0.46	4.17	**0.45**	16.92	0.46	18.45	0.46	2.38	0.46	**2.02**
Trafalgar-201	0.48	2.83	**0.46**	3.72	0.50	27.17	0.46	5.18	0.47	1.30	0.47	**1.22**
Trafalgar-206	**0.45**	12.80	0.45	4.36	0.45	22.26	0.45	17.30	0.46	1.52	0.46	**1.32**
Trafalgar-215	0.45	9.97	0.45	4.33	0.45	24.01	0.45	12.63	0.45	1.24	**0.45**	**1.21**
Trafalgar-225	0.44	5.55	0.44	4.49	0.44	19.43	0.44	23.96	0.44	1.35	**0.44**	**1.29**
Trafalgar-257	0.43	8.16	0.43	3.82	0.43	21.69	**0.43**	3.307	0.44	1.36	0.44	**1.15**

Table 1: The results of the Trafalgar dataset.

3 Experimental Results

We evaluate the performance of MegBA with the FP64, 1-GPU configuration and compare MegBA against Ceres [1], g2o [4], RootBA [3] and DeepLM [5]. MegBA uses the Levenberg–Marquardt algorithm and the trust-region strategy (same as Ceres).

3.1 BAL Datasets

We summarise the results of the BAL dataset in Table 1 (Trafalgar), Table 2 (Ladybug), Table 3 (Dubrovnik), Table 4 (Venice), and Table 5 (Final). As we can see in these tables, MegBA consistently outperforms all baselines by up to 24x. We also evaluated PBA [9] on the BAL dataset. PBA supports FP32 only, so we evaluate MegBA(FP32) with PBA in Table 6, showing that MegBA outperforms PBA by a large margin.

3.2 1DSfM Dataset

We further compare MegBA with the baselines using the 1DSfM [8] dataset. The statistics, such as the number of observations, of the 1DSfM dataset can be found in Table 7. Results are shown in Table 8. 1DSfM is a challenging dataset and other algorithms can converge to local optima that have several magnitudes of orders larger MSE than other algorithms, while MegBA always converges to a competitive optima.

	Ceres-16		DeepLM		g2o-16		Rootba-16		MegBA-1-a		MegBA-1-m	
	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time
Ladybug-49	0.42	1.07	0.42	2.23	0.42	2.50	0.42	0.32	0.42	0.19	**0.42**	**0.19**
Ladybug-73	0.37	1.24	0.37	2.72	0.37	2.12	0.37	0.67	0.37	0.24	**0.37**	**0.22**
Ladybug-138	0.70	2.72	0.70	3.00	0.70	31.21	0.82	0.79	0.70	0.32	**0.70**	**0.29**
Ladybug-318	0.48	3.04	0.48	2.95	0.48	65.70	0.48	3.78	0.48	0.44	**0.48**	**0.33**
Ladybug-372	0.55	4.45	0.55	2.10	0.55	83.52	0.55	2.96	0.55	0.54	**0.55**	**0.41**
Ladybug-412	0.50	4.07	0.50	3.34	0.49	86.94	0.50	2.83	0.49	0.58	**0.49**	**0.44**
Ladybug-460	0.53	6.02	0.53	3.47	0.53	243.32	0.53	5.38	0.53	0.69	**0.53**	**0.55**
Ladybug-539	0.55	8.37	0.55	3.85	0.55	65.63	0.55	5.08	0.55	0.75	**0.55**	**0.57**
Ladybug-598	0.55	8.54	0.55	3.34	**0.54**	293.69	0.55	4.16	0.55	0.79	0.55	**0.60**
Ladybug-646	0.55	9.21	0.55	3.56	0.56	200.83	0.55	5.44	0.55	0.92	**0.55**	**0.74**
Ladybug-707	0.57	15.30	0.56	4.48	0.56	283.15	0.56	6.59	0.56	0.98	**0.56**	**0.76**
Ladybug-783	0.53	7.69	0.53	5.01	0.53	539.85	**0.52**	15.95	0.53	0.98	0.53	**0.78**
Ladybug-810	0.52	9.67	0.52	4.77	0.52	391.78	0.52	13.71	0.52	1.00	**0.52**	**0.74**
Ladybug-856	0.51	13.30	0.51	4.81	0.51	114.89	0.51	8.01	0.51	1.10	**0.51**	**0.86**
Ladybug-885	0.51	15.00	0.51	5.08	0.51	308.21	0.51	8.03	0.51	1.01	**0.51**	**0.79**
Ladybug-931	0.55	15.30	0.55	4.56	0.55	220.54	0.55	11.08	0.55	1.18	**0.55**	**0.96**
Ladybug-969	0.55	17.50	0.54	5.21	0.55	197.59	0.54	7.79	0.54	1.19	**0.54**	**0.94**
Ladybug-1031	0.55	12.90	0.55	4.04	0.55	203.63	0.55	10.24	0.55	1.48	**0.55**	**1.18**
Ladybug-1064	0.55	15.40	0.56	3.17	0.55	355.78	0.55	4.47	0.55	1.23	**0.55**	**1.03**
Ladybug-1118	0.57	15.40	0.58	3.68	0.58	311.09	0.58	6.16	0.57	1.39	**0.57**	**1.11**
Ladybug-1152	0.56	13.70	0.56	3.07	0.56	488.39	0.56	7.53	0.56	1.46	**0.56**	**1.10**
Ladybug-1197	0.57	17.30	0.57	4.13	0.57	122.18	0.57	8.58	0.57	1.40	**0.57**	**1.13**
Ladybug-1235	0.56	20.30	0.58	3.34	0.56	212.62	0.57	5.89	0.56	1.30	**0.56**	**1.06**
Ladybug-1266	0.57	23.60	0.56	4.53	0.56	342.70	0.56	8.81	0.56	1.44	**0.56**	**1.18**
Ladybug-1340	0.57	26.20	0.57	4.54	2.02	453.25	0.57	9.17	0.57	1.61	**0.57**	**1.31**
Ladybug-1469	0.56	26.30	0.57	3.84	0.58	724.13	0.56	8.30	0.56	1.73	**0.56**	**1.37**
Ladybug-1514	0.56	25.20	0.56	4.69	1.18	459.05	0.56	8.61	0.56	1.59	**0.56**	**1.34**
Ladybug-1587	0.59	37.50	0.56	4.82	1.15	567.71	0.58	6.77	0.56	2.01	**0.56**	**1.58**
Ladybug-1642	0.58	36.10	0.56	3.64	0.76	127.00	0.56	16.12	0.56	1.73	**0.56**	**1.43**
Ladybug-1695	0.56	32.30	0.56	4.24	0.62	799.46	0.59	5.46	0.56	1.87	**0.56**	**1.51**
Ladybug-1723	0.56	34.50	0.57	3.93	1.96	140.69	0.56	7.05	0.56	0.93	**0.56**	**0.77**

Table 2: The results of the Ladybug dataset.

3.3 Large Synthesised Dataset

In the end, we evaluate MegBA on a larger synthetic dataset: `SynthesisedData-20000`. This dataset emulates a real-world BA problem we have in production. We use $\mathcal{U}(a, b)$ to denote a uniform distribution with a minimum as a and a maximum as b. We generate 20,000 cameras on a circle of radius 8 uniformly. We add a uniform noise $\epsilon \sim \mathcal{U}(0, 0.01)$ to camera rotation and translation, as well as a uniform noise $\epsilon \sim \mathcal{U}(0, 0.5)$ to the camera intrinsic. We generate 80, 000 points. For each point, the x, y coordinates are sampled from a uniform distribution $\mathcal{U}(-0.1, 0.1)$ and the z coordinate is sampled from a uniform distribution $\mathcal{U}(-0.03, 0.03)$. The

	Ceres-16		DeepLM		g2o-16		Rootba-16		MegBA-1-a		MegBA-1-m	
	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time
Dubrovnik-16	0.22	0.79	0.22	1.75	0.22	1.72	0.22	**0.36**	0.22	1.32	**0.22**	1.24
Dubrovnik-88	0.75	7.69	0.75	4.77	0.75	21.87	0.75	11.14	0.75	1.89	**0.75**	**1.63**
Dubrovnik-135	0.67	17.00	0.67	5.88	0.67	34.21	0.67	17.99	0.67	1.52	**0.67**	**1.50**
Dubrovnik-142	0.48	17.30	0.48	6.03	0.48	30.11	0.48	6.74	0.48	2.34	**0.48**	**2.25**
Dubrovnik-150	0.43	15.20	0.43	6.27	0.43	31.32	0.43	33.01	0.43	1.11	**0.43**	**1.00**
Dubrovnik-161	0.41	16.80	0.41	7.61	0.41	31.96	0.41	19.32	0.41	0.71	**0.41**	**0.69**
Dubrovnik-173	0.41	15.40	0.41	7.38	0.41	38.44	0.41	59.45	0.41	0.96	**0.41**	**0.84**
Dubrovnik-182	0.45	13.40	0.45	8.11	0.45	41.91	0.45	67.52	0.45	1.08	**0.45**	**1.00**
Dubrovnik-202	0.43	21.70	0.43	9.40	0.44	54.12	0.43	22.19	0.43	1.41	**0.43**	**1.29**
Dubrovnik-237	0.42	25.50	0.42	9.95	0.42	59.35	0.42	17.27	0.41	0.91	**0.41**	**0.80**
Dubrovnik-253	0.38	29.00	0.38	10.82	0.39	69.64	0.38	50.39	0.38	1.45	**0.38**	**1.24**
Dubrovnik-262	0.37	95.90	0.37	11.25	0.37	64.68	0.37	52.60	0.37	2.79	**0.37**	**2.59**
Dubrovnik-273	0.37	47.90	0.37	11.41	0.37	63.24	0.36	60.37	0.37	1.87	**0.37**	**1.78**
Dubrovnik-287	0.36	26.80	0.36	11.43	0.37	93.30	0.36	40.84	0.36	2.35	**0.36**	**2.13**
Dubrovnik-308	0.37	33.80	0.37	11.22	0.37	91.82	0.37	37.15	0.37	3.23	**0.37**	**2.88**
Dubrovnik-356	**0.39**	116.00	0.40	6.12	0.39	94.39	0.39	78.16	0.41	3.64	0.41	**3.26**

Table 3: The results of the Dubrovnik dataset.

x, y coordinates of each point are also added noise $\epsilon \sim \mathcal{U}(-0.1, 0.1)$. Each point can be captured by 1,000 cameras, and there are 80,000,000 observations in total.

The evaluation was conducted on a GPU server with 300 GB CPU memory in total and 32 GB GPU memory for each GPU. RootBA [3] and DeepLM [5] incur out-of-memory (OOM) errors and fail to complete in this experiment. g2o cannot return results in a reasonable time (1 hour) and throws an out-of-time error.

Figure 5 shows the experimental results of MegBA, Ceres, DeepLM and g2o. DeepLM and g2o both cannot solve this problem due to memory and computation limitations. Only MegBA and Ceres can process such a large dataset; however, MegBA is 20× faster than Ceres, making MegBA the state-of-the-art BA library for large BA problems (with more than 100s millions of observations).

4 GPU Utilisation

Solving BA problems on a single GPU can suffer from computation and memory bottlenecks. We are interested in if the computation bottleneck can be resolved by adding more GPUs. We used nvprof to profile MegBA and calculated its GPU utilisation. In particular, we plot the time proportion of different computation in MegBA. We repeat the same experiment with 1, 2, 4 GPUs and report the results in Figure 6. According to this figure, the profiling results show that using multiple GPUs incurs only marginal communication cost and the computation time decreases around 50%.

	Ceres-16		DeepLM		g2o-16		Rootba-16		MegBA-1-a		MegBA-1-m	
	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time
Venice-52	0.75	7.95	0.75	4.66	**0.68**	17.94	0.68	2.39	0.73	1.85	0.73	**1.59**
Venice-89	0.50	8.47	0.50	6.41	0.50	27.53	0.51	**4.51**	0.50	5.05	**0.50**	4.85
Venice-245	0.84	31.90	0.87	5.66	0.87	123.27	0.85	10.47	0.84	3.26	**0.84**	**3.06**
Venice-427	0.63	58.20	0.63	15.46	**0.62**	115.24	0.63	43.12	0.63	5.17	0.63	**5.17**
Venice-744	0.51	109.0	0.51	27.79	0.51	659.83	0.51	43.60	0.50	6.26	**0.50**	**6.04**
Venice-951	**0.44**	84.30	F	F	0.57	793.15	0.45	59.45	0.45	13.80	0.45	**13.64**
Venice-1102	0.40	146.0	0.39	32.80	0.56	585.86	0.40	94.13	0.39	14.42	**0.39**	**13.24**
Venice-1158	0.42	33.10	0.43	12.07	0.48	581.79	0.36	51.18	0.36	6.74	**0.36**	**6.32**
Venice-1184	0.77	30.10	0.35	36.32	0.46	600.57	0.35	38.16	0.34	36.24	**0.34**	**28.52**
Venice-1238	0.35	71.40	0.34	13.20	0.46	531.25	0.34	90.36	0.34	8.36	**0.34**	**7.75**
Venice-1288	0.33	153.0	0.33	19.79	0.45	630.90	0.33	95.45	0.33	8.36	**0.33**	**7.64**
Venice-1350	0.34	52.30	0.34	10.47	0.44	551.41	0.34	53.85	0.33	6.79	**0.33**	**6.43**
Venice-1408	0.35	76.50	0.35	10.72	0.47	687.48	0.35	47.00	0.34	5.66	**0.34**	**5.34**
Venice-1425	0.34	153.0	0.34	10.80	0.44	635.88	0.34	183.28	0.34	5.67	**0.34**	**5.28**
Venice-1473	0.33	110.0	0.33	43.24	0.42	571.33	0.33	61.21	0.33	7.05	**0.33**	**6.68**
Venice-1490	0.33	66.80	0.33	14.25	0.33	571.17	0.33	123.52	0.32	5.64	**0.32**	**5.32**
Venice-1521	0.33	75.60	0.33	10.98	0.33	666.63	0.33	93.50	0.32	7.47	**0.32**	**6.78**
Venice-1544	0.33	173.0	0.33	26.59	0.42	653.50	0.33	132.69	0.32	5.71	**0.32**	**5.33**
Venice-1638	0.57	75.40	0.58	30.04	**0.56**	1015	0.57	54.58	0.58	17.66	0.58	**16.60**
Venice-1666	0.48	74.00	0.48	23.48	0.45	1033	**0.43**	85.30	0.44	10.75	0.44	**10.08**
Venice-1672	0.38	260.0	0.38	33.49	0.38	991.73	**0.37**	123.52	0.40	12.20	0.40	**11.06**
Venice-1681	0.37	48.40	0.34	46.98	0.34	1067	0.34	118.23	0.34	9.16	**0.34**	**8.38**
Venice-1682	0.34	164.0	0.33	38.92	0.34	956.35	0.33	274.34	0.33	17.39	**0.33**	**15.78**
Venice-1684	0.34	207.0	0.33	43.94	0.33	1658	0.33	202.10	0.33	11.78	**0.33**	**11.25**
Venice-1695	0.37	53.40	0.34	44.99	0.34	1648	**0.33**	292.24	0.34	16.16	0.34	**14.48**
Venice-1696	0.38	43.00	0.33	47.04	0.34	1335	**0.33**	243.68	0.34	7.43	0.34	**7.06**
Venice-1706	0.34	234.0	0.34	38.17	**0.33**	1845	0.34	196.48	0.34	19.72	0.34	**18.59**
Venice-1776	0.33	192.0	0.33	46.69	0.35	1147	0.34	62.10	0.33	10.56	**0.33**	**10.02**
Venice-1778	0.33	319.0	0.33	24.44	0.34	890.55	0.34	73.94	0.33	11.96	**0.33**	**10.92**

Table 4: The results of the Venice dataset. F stands for failed.

	Ceres-16		DeepLM		g2o-16		Rootba-16		MegBA-1-a		MegBA-1-m	
	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time
Final-93	0.51	4.10	0.51	1.56	0.51	6.22	0.51	1.84	0.51	0.78	**0.51**	**0.67**
Final-394	0.56	30.10	0.56	5.10	0.56	101.72	0.56	20.07	0.56	1.10	**0.56**	**0.92**
Final-871	0.62	115.0	0.62	26.12	0.62	481.22	0.62	76.65	0.62	8.20	**0.62**	**7.75**
Final-961	0.94	44.30	0.94	12.82	0.94	710.55	0.94	348.11	0.94	3.32	**0.94**	**2.85**
Final-1936	0.89	106.0	0.89	17.03	0.89	789.57	0.89	595.28	0.89	7.03	**0.89**	**6.67**
Final-3068	1.09	31.60	1.03	5.49	1.13	26203	0.98	72.27	1.01	2.82	**1.01**	**2.40**
Final-4585	0.57	417.0	**0.56**	66.13	0.56	23221	0.56	373.30	0.57	22.22	0.57	**15.60**

Table 5: The results of the Final dataset.

	PBA(FP32)		MegBA-1-m(FP32)	
	MSE	Time	MSE	Time
Trafalgar-257	0.85	0.57	**0.44**	**0.11**
Ladybug-1695	34.13	0.37	**1.33**	**0.32**
Dubrovnik-287	0.84	1.22	**0.38**	**0.10**
Venice-89	0.86	1.32	**0.51**	**0.28**
Final-4585	1.87	4.96	**0.57**	**3.13**

Table 6: We evaluate PBA with the BAL dataset. To make a fair comparison, we compare PBA against the FP32 MegBA. Both PBA and FP32 MegBA suffer from numerical instability issues in some datasets. In addition, PBA uses texture memory, it failed to solve large-scale BA problems due to memory limitations. We chose the largest datasets that can be solved successfully by PBA to evaluate its performance. The result shows that MegBA outperforms PBA by a large margin, with maximum 12.5 times speed up while achieving a lower MSE.

Dataset	#Points	#Observations
Alamo-577	140080	816891
Ellis Island-233	9210	20500
Gendarmenmarkt-704	76964	268747
Madrid Metropolis-347	44479	195660
Montreal Notre Dame-459	151876	811757
Notre Dame-548	224153	1172145
NYC Library-334	54757	211614
Piazza del Popolo-336	29731	150161
Piccadilly-2292	184475	798085
Roman Forum-1067	223844	1031760
Tower of London-484	126648	596690
Trafalgar-5052	327920	1266102
Union Square-816	26430	90668
Vienna Cathedral-846	154394	495940
Yorkminster-429	100426	376980

Table 7: The statistics of the datasets in 1DSfM.

5 Reconstruction Plots

Finally, we show the visualisation results of MegBA. Specifically, we show the landmark point clouds (i.e. black dots) with cameras (i.e. red frames) for each dataset. The point clouds are rendered by COLMAP [6] [7].

References

1. Agarwal, S., Mierle, K., Others: Ceres solver. http://ceres-solver.org 1, 4
2. Agarwal, S., Snavely, N., Seitz, S.M., Szeliski, R.: Bundle adjustment in the large. In: European conference on computer vision (2010) 1

	Ceres-16		DeepLM		g2o-16		Rootba-16		MegBA-1-m	
	MSE	Time	MSE	Time	MSE	Time	MSE	Time	MSE	Time
Alamo	4.8E+2	219.0	2.3E+2	40.4	7.6E+5	48.9	2.3E+2	31.1	**2.2E+2**	**18.9**
Ellis Island	2.2E+3	6.3	2.3E+3	9.9	4.6E+5	2.7	**1.6E+3**	1.8	2.8E+3	**0.5**
Gen.markt	2.5E+1	63.0	6.2E+1	31.2	4.4E+4	46.0	1.5E+2	20.8	**1.6E+1**	**2.4**
M.Metropolis	8.9E+3	45.7	8.4E+3	21.2	2.8E+8	51.4	**8.0E+3**	15.8	1.1E+4	**1.1**
M.N.Dame	8.4E+2	206.0	**7.1E+2**	73.4	5.0E+5	89.1	7.3E+2	46.4	7.4E+2	**10.6**
Notre Dame	**1.5E+4**	282.0	1.6E+4	92.3	4.2E+8	68.6	2.3E+4	48.3	2.0E+4	**10.8**
NYC Library	4.8E+0	62.7	1.2E+1	37.4	**2.4E+0**	23.2	3.0E+1	17.7	4.1E+0	**1.4**
P.del Popolo	7.5E+1	26.5	8.4E+1	4.6	5.3E+6	1.4	**3.2E+1**	19.3	3.7E+1	**1.4**
Piccadilly	4.8E+2	213.0	6.3E+2	68.3	3.9E+6	21.6	3.7E+2	73.5	**3.7E+2**	**14.3**
R.Forum	6.5E+1	104.0	5.6E+1	88.5	1.1E+5	201.6	5.8E+1	105.8	**5.5E+1**	**6.0**
T.of London	1.4E+3	81.9	1.8E+3	53.5	6.6E+8	7.1	1.9E+3	45.9	**1.3E+3**	**14.7**
Trafalgar	1.7E+2	358.0	1.6E+2	112.1	4.8E+5	34.8	1.7E+2	211.3	**1.5E+2**	**15.0**
Union Square	5.4E+2	24.2	3.9E+2	5.8	2.9E+6	23.1	**2.3E+2**	8.9	3.4E+2	**0.8**
V.Cathedral	5.7E+2	139.0	5.2E+2	49.2	1.1E+5	51.6	4.9E+2	13.1	**4.9E+2**	**12.2**
Yorkminster	1.2E+1	115.0	9.2E+0	18.9	**4.2E+0**	25.9	1.0E+2	29.8	9.0E+0	**4.4**

Table 8: Results of the 1DSfM dataset.

	ceres16	DeepLM	g2o	RootBA	MegBA-1-a	MegBA-1-m
Alamo	2.16	1.86	1.96	4.39	2.88	**1.85**
Ellis Island	1.46	1.53	1.15	1.06	0.67	**0.65**
Gen.markt	1.72	1.83	1.42	1.13	1.36	**1.07**
M.Metropolis	1.62	1.73	1.33	1.14	1.15	**0.95**
M.N.Dame	2.16	2.40	1.92	3.85	2.87	**1.85**
Notre Dame	2.50	**2.39**	2.29	5.54	3.88	2.47
NYC Library	1.64	1.76	1.35	1.13	1.20	**0.98**
P.del Popolo	1.57	1.67	1.29	1.31	1.02	**0.89**
Piccadilly	2.20	2.47	2.18	3.30	2.85	**1.85**
R.Forum	2.38	2.56	2.17	3.40	3.49	**2.17**
T.of London	1.99	2.17	1.71	1.80	2.28	**1.55**
Trafalgar	2.72	2.99	3.15	4.97	4.15	**2.51**
Union Square	1.53	1.58	1.24	1.09	0.86	**0.80**
V.Cathedral	1.96	2.10	1.67	**1.36**	2.00	1.40
Yorkminster	1.81	2.02	1.52	**1.19**	1.67	1.23

Table 9: The memory overhead of the 1DSfM dataset. The unit is GB.

3. Demmel, N., Sommer, C., Cremers, D., Usenko, V.: Square root bundle adjustment for large-scale reconstruction. In: IEEE Conference on Computer Vision and Pattern Recognition (2021) 4, 6
4. Grisetti, G., Kümmerle, R., Strasdat, H., Konolige, K.: g2o: A general framework for (hyper) graph optimization. In: IEEE International Conference on Robotics and Automation (2011) 4
5. Huang, J., Huang, S., Sun, M.: Deeplm: Large-scale nonlinear least squares on deep learning frameworks using stochastic domain decomposition. In: IEEE/CVF Con-

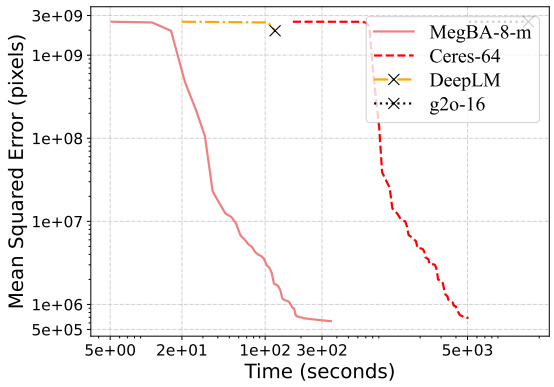


Fig. 5: Experimental results of the synthesised dataset. DeepLM throws an OOM error after the 6th step. g2o is significantly slower than all other baselines and we terminate its execution after 6 steps.

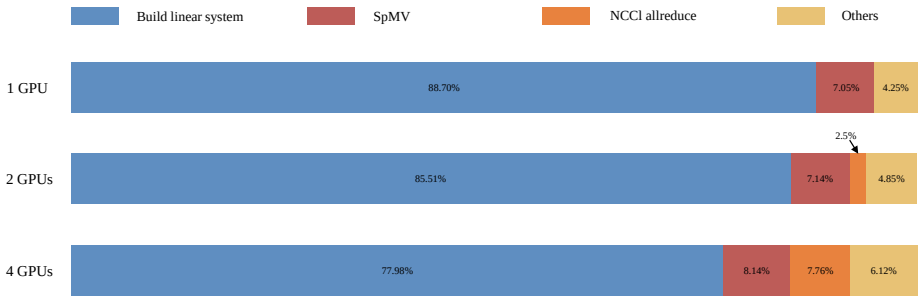
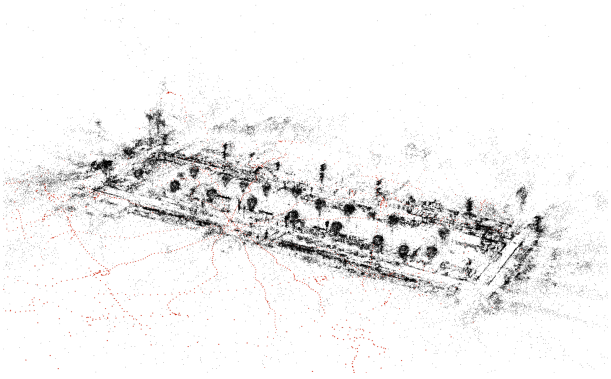
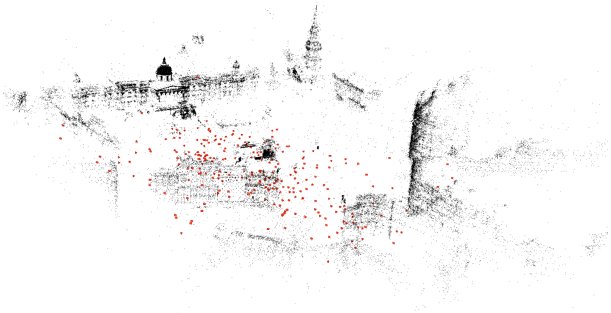


Fig. 6: The visualisation of the profiling result returned by nvprof. We use the dataset Venice-1778. SpMV denotes sparse matrix-vector multiplication. Using 2 GPUs, the all-reduce communication accounts for 2.5%, while the overall time reduces 60.6% compared with the result using 1 GPU. Using 4 GPUs, the all-reduce communication accounts for 7.8%, while the overall time reduces 69.8% compared with the result using 2 GPUs.

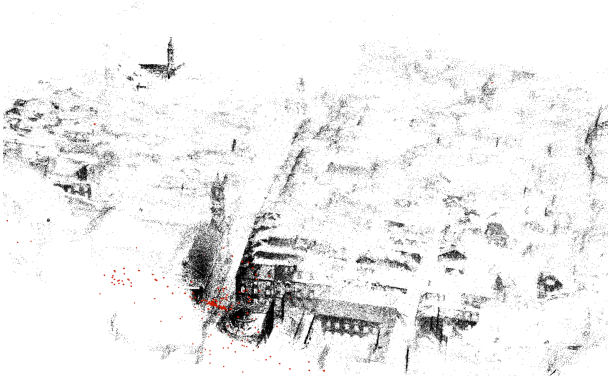
ference on Computer Vision and Pattern Recognition (2021) 4, 6
6. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (2016) 8
7. Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (2016) 8
8. Wilson, K., Snavely, N.: Robust global translations with 1dsfm. In: European Conference on Computer Vision (2014) 1, 4
9. Wu, C., Agarwal, S., Curless, B., Seitz, S.M.: Multicore bundle adjustment. In: IEEE Conference on Computer Vision and Pattern Recognition (2011) 4



Ladybug



Trafalgar Square



Dubrovnik