Name of student: Chieh-En Li

Name of instructor: Milind Kulkarni

Project title: Approximate K Nearest Neighbor with KDtree on GPU

Credit hours: 1

Term: Spring 2020

Brief description of the project:

In this project, I am looking for any method to optimize K nearest neighbor search with KDtree running on GPU and try to lower the time consumption without sacrifice too much accuracy.

Brief summary of the results:

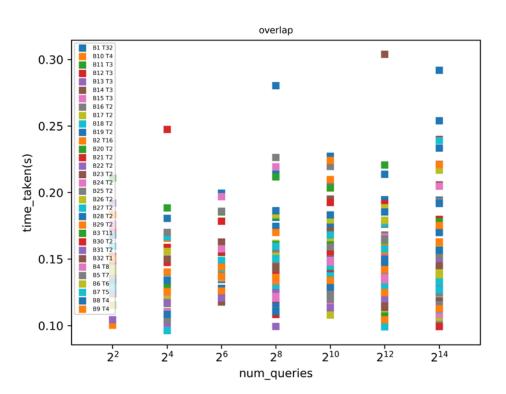
After my experiment, I found just purely control the the traverse way may not increase any time efficiency but may get a worse result. Therefore, to solve the problem may modify the original KDtree, especially decrease the tree depth, but decrease the tree depth may sacrifice the accuracy.

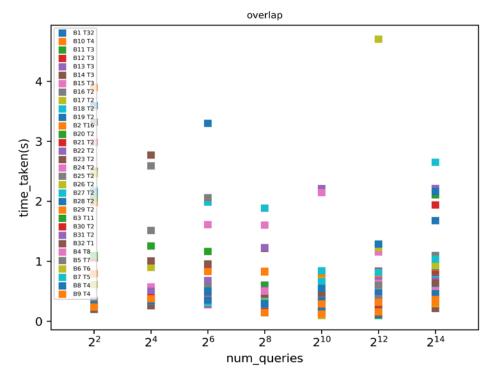
Outline for the report:

- 1. Result comparison and possible reason.
- 2. Method.
- 3. What I have learned from this experience.

Non -reversed tree order result

Reversed tree order result





In the legend "BxTy", x means block number and y means each thread number in each block. The experiment condition: same queries points and on the same machine.

As the result, reversed tree order can get an overall lower time efficiency; there is no linear relationship between time for traversing and block number. The reason that reversed tree order can lead to a relatively longer time on traversing could be the unideal spatial locality. When the machine traverse the reversed order trees, it cannot access just the just the adjacent memories blocks.

	17	esuit C	<u>Jiiibai isc</u>	<u>ni anc</u>	1 LO22	INIE NEC	35011.
Block number.	Thread number for each block.	Non –reversed tree order result	Reversed tree order result	Block number.	Thread number for each block.	Non -reversed tree order result	Reversed tree order result
1	32	250	13.5 2.5 2.5 2.5 2.6 2.6 2.6 2.6 2.6 2.6 2.6 2.6 2.6 2.6	5	7 02 03 01 01 01 01 01 01 01 01 01 01 01 01 01		### ##################################
2	ume takerits)		1000	6	0.18 - 0.16 - 0.16 - 0.12 - 0.10 - 0.	2° 23 <u>26 2</u> 9 <u>212 215</u> :	1.2 - 1.0 - (3) 0.8 - (3) 0.6 - (3)
3	11	# # # # # # # # # # # # # # # # # # #	### ### ##############################	7	5 0.17 . 0.16 . 0.15 . 0.15 . 0.15 . 0.15 . 0.15 . 0.15 . 0.12 . 0.11 . 0.10 .	20 23 26 29 212 215 21 OCC, numBithreads, per Joick Atreenum32 quiers _ max16384 quiers_min4.quiers_i	1.0 - 0.8 - 0.6 - 0.4 - 0.2 - 26 29 212 215 218
4	8		risiod, rund, lineade, joe, jocotii zireerumi 2 quem, mas 1838, quem, mostquiem, interved 2,000 1,75 1,50 1,50 1,75 1,75 1,75 1,75 1,75 1,75 1,75 1,75	8	4 0.18 -	2 ⁰ 2 ³ 2 ⁶ 2 ⁹ 2 ¹² 2 ¹⁵	2.25 2.00 1.75 1.59 3.00 0.75 0.50 0.25 2.0 25 2.0 25 2.0 25 2.0 25 2.1 25 2.1 25 2.2 25 2.2 25 2.3 25

Block number.	Thread number for each block.	Non –reversed tree order result	Reversed tree order result		Thread number for each block.	Non –reversed tree order result	Reversed tree order result
9	quayer auu	riblick, rum/s.threads, per_libick.threenum32.quiers_mas16384.quiers_min4.quiers_ 1222 - 120 - 1	### ##################################	13	S (spaceper)	nthick_um11/nreati_per_block1/neeroun32.gainn_nas_16384.gainn_nos4.gainn_nise4	15 20 28 28 28 212 211 218 num_qqqries
10	4	nblock_num10.threads_per_block&freenum32.quiers_max16384.quiers_mink.qui 1.18	2.00 1.75 1.30 2.00 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75 1.50 2.75	14	Converge		781016, rum34.87eeald, juer jäestä tirannum32.quien, marsti 398.quien, mink, quien, maercalit 2.5 2.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3
11	3	num_queries **riord_numil.hteenin_ger_tisch3/evenum32.quers_van163H.quers_nunt.quers_in 0.22 0.20 0.13 0.14 0.12 0.10 20 23 26 27 212 215 num_queries	1000 sund1.threads per Mod3 trenne32 query mod3384 query mod4pers intervers 0.355 0.305 0.20 0.15	15		### ### ##############################	10 - 10 - 27 - 27 - 27 - 27 - 217 - 218 -
12	Sine, Lakers(s)	nblock_rum12.threads_par_block3.breenum32.quiers_reax16384.quiers_mn4.quiers_ 024 -		16	2	######################################	1.0 - 1.0 - 2.3 - 2.0 - 2.0 - 2.1 -

Block number.	Thread number for each block.	Non -reversed tree order result	Reversed tree order result	Block number.	Thread number for each block.	Non –reversed tree order result	Reversed tree order result
17	2	\$100.0, rest. timest., en. \$800.0 comments acres, rest. \$100.0 com, rest. acres, re	thinks_rund17,threads_ser_block27rrensen33.gozen_mas16384.qozen_mod.qozen_ritoroxid 4. 3. 3. 3. 1. 3. 2. 1.	21	2	0.24 0.20 0.20 0.20 0.20 0.20 0.20 0.20	### ### ##############################
18	2	80100_max_periors	0 28 28 29 212 215 238 num_quaries 1000, num18 Prose (num_quaries) 200 200 200 200 200 200 200 200 200 20	22	2	18/04_mm22 8/mah_mm_lms22 8/mmm_ms23 a/mm_ms15/24 q.dmm_min4_a/mm_interval 0.22 0.20 0.20 0.20 0.10 0.18 0.14 0.12 0.10 2 ³	7.0 - 20 20 26 20 217 215 218 num, queries
19	2	**************************************	123 150 175 175 175 175 175 175 175 175 175 175	23	(i),uappi Tuni		1.0 0.8 0.4 0.3 1.0 0.9 0.4 0.10 0
20		ninos puestilinnos per junta jumentos game puestilidadem predugem primate o se	2 00 - 1.75 - 1.86 - 1.95 - 1.95 - 1.	24		022- 023- 6 018- 6 10- 014- 012- 23 23 24 25 252 253 7	1.6 1.4 1.2 2.6 2.7 2.7 2.7 2.7 2.7 2.7 2.7 2.7 2.7 2.7

	I V	Court C	<u>OTTIDATISC</u>	<u>/II alle</u>	<u> </u>	IDIC INCO	<u> </u>
Block number.	Thread number for each block.	Non –reversed tree order result	Reversed tree order result		Thread number for each block.	Non –reversed tree order result	Reversed tree order result
25	Disease and	Notes a participation of the control	1.4 1.2	29	2	sition modificates, jer jine 2 jorne mili quire, e milital quire, mid quire,	risiocs ,rum29,threads ,per_block2,treenum32,quiers _max16384,quiers _min4,quiers _intervald 0.8 0.7 0.6 0.5 0.9 0.1 20 20 20 20 20 21 22 23 24 25 22 215 218
26	2 92 92 93 93 93 93 93 93 93 93 93 93 93 93 93	8-	### ### ##############################	30	2	nblock_rum002thread_per_thox22thream=12.querr_mox16384.querr_mon4.cuerr_1 0.20 0.18 0.16 0.17 0.18 0.19 0.19 0.19 0.10 0.10 0.10 0.10 0.10	### ### ##############################
27	interest annual	10 10 10 10 10 10 10 10 10 10 10 10 10 1	20 28 26 29 222 223 228 ***Rum_queries** **Ross_mac27/invols_per-biolo2/jeronat2a_ven_mac2694.dates_res_dates_himned** 6.7 6.8 6.9 6.9 6.1 6.2 6.3 6.3 6.3 6.3 6.3 6.3 6.3	31	2	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	28 23 26 23 212 215 218 num_queries nam_discox_num31.threads_ser_block2.treenum32.quiers_max14384.quiers_mont.quiers_intervalid 0.7 0.6 0.5 0.9 0.4 0.3 0.3 0.2
28	brez Ladorfes O O O		191003, num28 threads, ser_flack2, tremum32, quers, mac16 84, quers, mor4, quers, extenses 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2	32		## ## ## ## ## ## ## ## ## ## ## ## ##	0.1 2º 2º 2º 2º 2º 2½ 215 218 num_queries 1.0 1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.

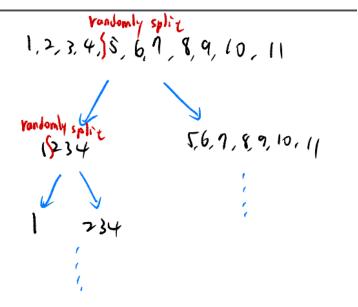
Result comparison and possible reason. Conclusion.

- Based on above results, there is no obvious best block and thread pairs to get the lowest time consumption. However, when traversing the reversed order tree, only when block number is 9,11,21,23,26,27,28,29,31,32 would get lower time on traversing but still get a higher traverse time than non-reversed tree order.
- To lower the time taken, instead of modifying the block and threads, cut the traverse depth would be a better way to enhance the time efficiency.
- To modify the inquiry point order may not work in my project because the method to accesses the inquiry points is in random when building a KDtree.

Method

• First I crease a KDtree with random spliting an array of inquiry points

as the figure shown.



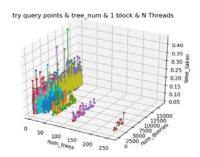
Method

- Second, I compare the inquiry points and the target with each dimension when traversing. When traverse depth increase, the compare dimension increases until reach the maximum dimension and then compare with the first dimension.
- In my experiment, I build KDtrees with three dimensions.
- To optimize the traverse process, I control the block number and threads number in eeach block as the figure shown bellow.

k_nearest_search_k1_GPU << < block_num, thread_num_per_block >> > (tree_reversed, target, tree_num, found);

What I have learned from this experience.

- Kdtree build and traverse.
- Cuda.
- Properly statistic showing.
 - A 3D graph is not intuitive for showing the concept.



- Sometimes matplotlib may not capture the correct order of a dataset, so scattering the points may be better.
- If too many data points may not clearly see the result because the noise

would be enlarged as well.

