PENGESANAN PAPAN TANDA MENGUNAKKAN RASPBERRY PI DENGAN KAEDAH MULTITHREADING

Wong Soon Fook
Abdul Hadi Abd Rahman

Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia

ABSTRAK

Penggunaan Robot Pintar adalah teknologi yang sudah berleluasa pada masa sekarang. Teknik ini diikuti dengan membolehkan robot untuk memahami benda seperti manusia supaya dapat menjimatkan masa dengan berjalan dalam ulangan. Dalam perkembangan teknologi yang begitu cepatnya, suatu sistem yang berjalan secara automatik dengan hasil yang tepat sangat diperlukan dalam penyelesaian suatu pekerjaan. Dalam keperluan ini digunakan suatu robot yang memiliki kecerdasan dan keunggulan tertentu dalam suatu hal. Robot digunakan kerana ketepatan yang tinggi, kecepatan dan ketepatan terhadap penyelesaian suatu masalah yang diberikan pada awal lagi bila diperlukan waktu penyelesaian yang cukup lama dimana melebihi batas kemampuan manusia Robot digunakan untuk melakukan fungsi tertentu, seperti pengendalian bahan, untuk melakukan fungsi tersebut, robot harus memiliki kemampuan untuk memindahkan benda ke koordinat tujuan. Beberapa cara yang digunakan untuk kajian ini seperti kegunaan anotasi imej, SVM dan *Multithreading*. Hasil daripada kajian adalah untuk meningkatkan prestasi pemproses *Raspberry Pi* dengan penggunaan *Multithreading*.

1 PENGENALAN

Pemprosesan imej adalah salah satu medium yang paling ulung untuk memproses imej dan digunakan pada pelbagai teknologi dengan cara yang terbaik. Pemprosesan imej merupakan topik yang sangat menarik suatu ketika dahulu dan juga menarik sampai kini dengan topik tersebut membangun secara drastik. Dalam konteks yang lain, teknologi pemprosesan imej digunakan lagi untuk kemajuan dan keberkesanan. Disebabkan teknologi pemprosesan imej adalah penting, proses ini hendaklah berjalan dengan lebih cekap dengan penggunaan kaedah *Multithreading* yang merupakan proses yang mempunyai pelbagai tugas dan menjadikan pemprosesan imej secepat mungkin dalam faktor masa. Masa adalah faktor terpenting dalam pemprosesan imej kerana kelewatan dalam masa ataupun kelewatan dalam penghantaran kerangka imej boleh menyebabkan pelbagai masalah pada keputusan akhir dalam proses nanti. Cadangan algoritma ini telah menggunakan konsep *multithreading* dalam pemprosesan imej supaya hasil daripada pengesanan imej adalah tepat sekali.

Di samping itu, Smart Car Robot juga merupakan satu simbol kepada pemodenan dan pembangunan pada zaman ini yang giat berubah dengan cepat. Setiap satu ciri kereta dan pengangkutan dibuat adalah untuk membantu dalam menyenangkan dan keselamatan kehidupan setiap orang. Disebabkan itu, penyelidikan ini adalah berkenaan kereta berautonomi dengan penambah baikan menggunakan *multithreading* dan pemprosesan imej. Setiap ciri sistem ini dibina mengikut algoritma pemprosesan imej dan berfungsi untuk mengesan imej seperti papan tanda dengan penambahan *multithreading* kepada sistem untuk prestasi yang lebih bagus. Pengesanan imej diikuti dengan penggunaan algoritma klasifikasi iaitu *Support Vector Machine* sebagai sebuah pembelajaran mesin untuk melatih gambar supaya sistem dapat maklumat tentang ciri imej yang hendak dikesan.

2 PENYATAAN MASALAH

Penggunaan *Smart Car Robot* yang sangat cekap disebabkan robot sendiri akan bergerak dengan autonomi kerana robot memahami setiap papan tanda dan bertindak balas terhadap papan tanda yang dikesan tanpa memerlukan pengguna untuk bergerakkannya.

Sambungan kepada masalah ini ialah apabila *Smart Car Robot* dapat mengesan papan tanda tersebut dengan kamera *Pi, Raspberry Pi* yang sebagai sistem pengesanan gambar-gambar papan tanda tersebut berfungsi dengan lembab. Ini disebabkan *Raspberry Pi* yang mempunyai empat teras tetapi hanya penggunaan teras tunggal sahaja yang dapat dicapai. Penggunaan teras tunggal ini menyebabkan prestasi pada *Raspberry Pi* lambat untuk kamera *Pi* mengesan papan tanda. Gambar yang dapat dikesan dengan menggunakan algoritma *Support Vector Machine* juga adalah terhad untuk pengesanan yang cepat apabila hanya teras tunggal digunakan menyebabkan prestasi pada sistem dan pengesanan menggunakan kamera Pi akan menjadi lambat apabila lebih gambar disimpan sebagai model SVM.

3 OBJEKTIF KAJIAN

Objektif kajian ini merangkumi:

1. Menjalani ujian lebih lanjut tentang pemprosesan papan tanda menggunakan kamera *Pi* dengan beberapa pemboleh ubah untuk diuji bagi meningkatkan prestasi sistem.

2. Perbandingan *Raspberry Pi* yang berdasarakan teras tunggal dengan *multicore* melalui *multithreading* supaya penggunaan Cpu dan memori *Raspberry Pi* adalah minimum.

4 METODOLOGI

Penyelidikan atau kajian adalah sebagai satu kegiatan pengumpulan, pengolahan dan analisis data yang dilakukan secara sistematik dan cekap untuk memecahkan sesuatu persoalan. Dengan itu, tujuan menjalankan kajian ialah untuk memperoleh jawapan melalui penggunaan suatu langkah ilmiah yang sistematik dan saintifik. Dengan metodologi kajian ini, tatacara melaksanakan kajian atau tatacara untuk mencapai matlamat kajian dapat dicapai. Dalam projek ini, terdapat beberapa fasa pemprosesan yang digunakkan untuk menjalankan projek yang lengkap dan sistematik.

4.1 FASA PERANCANGAN

Fasa ini melibatkan proses mengenal pasti masalah, merancang objektif, persoalan kajian dan menentukan skop. Langkah seterusnya adalah sorotan susastera yang melibatkan pengumpulan , pencarian dan pembacaan jurnal dan kajian lepas bagi mencetus idea dan inspirasi. Fasa ini kita terdapat masalah timbul apabila peningkatan ke atas bilangan imej dalam setiap model SVM untuk pengesanan oleh kamera Pi yang bebatan menujukkan prestasi kepada ketepatan pengesanan akan menurun. Selain itu, semasa proses dijalakan pada sistem, hanya penggunaan 1 teras pada *Raspberry Pi* yang amatlah memberatkan penggunaan memori yang menyebabkan gambar yang disimpan adalah sedikit disebabkan sistem *Raspberry Pi* tidak dapat menyokong penyimpanan gambar yang banyak lalu akan melambatkan prestasinya.

4.2 FASA PENGUMPULAN DATA

Projek ini berkenaan dengan pengesanan papan tanda jadi pengumpulan gambar-gambar papan tanda adalah dari sumber *https://github.com/Moataz-E/deeplearning-traffic-signs*. Setiap gambaran yang diguna mempunyai maklumat yang berlainan. Gambaran yang dikumpul adalah dari beberapa resolusi yang akan ditetapkan kepada 4 resolusi iaitu 160x128, 240x192, 640,480 dan 1296x736. Peningkatan resolusi pada setiap kali pengesanan akan menguji kemampuan sistem dalam berfungsi dengan cekap.

Data prestasi semasa pengujian pengesanan papan tanda akan dikumpul dan diaporkan untuk mengetahui hasil prestasi dengan beberapa attibut yang digunakan seperti resolusi dan amaun imej.

4.3 FASA PEMPROSESAN

Projek lebih memfokuskan tentang prestasi dalaman daripada prestasi luaran iaitu lebih kepada prestasi oleh *Raspberry Pi* dalam kecekapan untuk menjalankan pengesanan papan tanda dengan penyimpanan gambar yang banyak dan gambar yang dari resolusi yang lebih tinggi . Terdapat proses yang dijalankan dalam projek ini iaitu *multithreading* untuk mendapatkan prestasi yang tepat. Proses pengesanan papan tanda akan dijalankan dengan pemantauan prestasi sistem *Raspberry Pi* lalu menambah *multithreading* untuk melihat perbezaan dengan koding yang sebelumnya.

4.4 FASA PENGESANAN

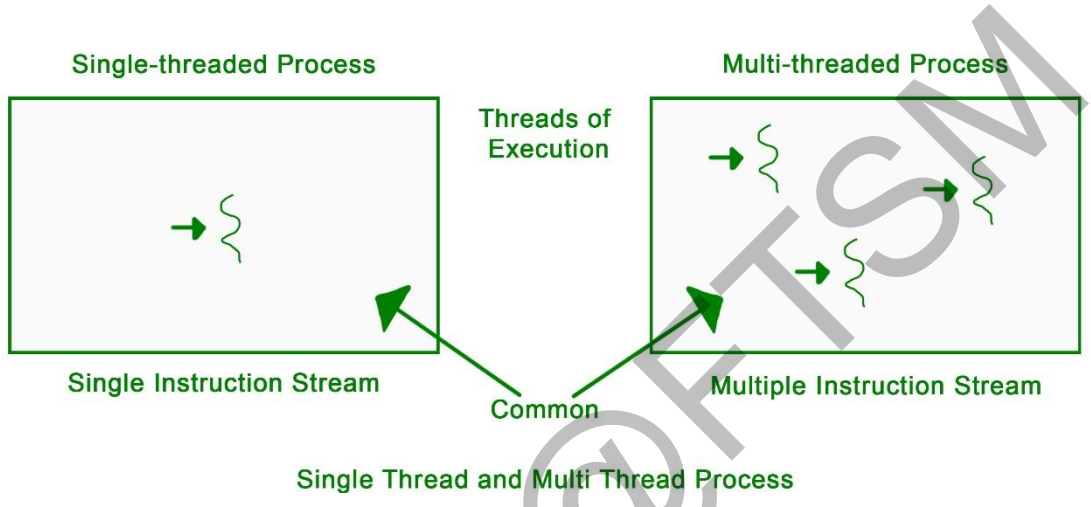
Fasa ini bertindak untuk mengesanan papan tanda yang telah dilatih menggunakan algoritma SVM.Pengesanan papan tanda dengan menggunakan kamera *Pi* dan apabila imej papan tanda yang telah dilatih dikesan, bingkai berwarna hijau, merah, biru ataupun putih akan muncul mengelilingi imej yang dikenali sebagai tanda imej tersebut adalah imej yang dikesan dengan betul.

4.5 FASA MULTITHREADING

Dalam seni bina komputer, *multithreading* adalah keupayaan unit pemprosesan pusat (CPU) (atau teras tunggal dalam pemproses berbilang teras) untuk melaksanakan pelbagai proses atau benang dengan serentak yang disokong oleh sistem pengendalian. Pendekatan ini berbeza dari *multiprocessing*. Dalam aplikasi *multithreaded*, proses dan benang berkongsi sumber teras tunggal atau berganda, termasuk unit pengkomputeran, cache CPU, dan penimbal terjemahan *lookaside* (TLB).

Di mana sistem *multiprocessing* termasuk pelbagai unit pemprosesan lengkap dalam satu atau lebih teras, *multithreading* bertujuan untuk meningkatkan penggunaan teras

tunggal dengan menggunakan paralelisme peringkat benang, serta paralelisme peringkat arahan. Oleh kerana kedua teknik itu saling melengkapi, kadang-kadang keduanya digabungkan dalam sistem yang berbilang *multithreading* CPU dan dengan CPU yang mempunyai banyak teras *multithreading* . (Wikipedia, Multithreading (computer architecture), 2019)



Rajah 1.0 Proses *Multithreading*

4.5.1 *Coarse-grained multithreading*

Jenis *multithreading* yang paling mudah berlaku apabila satu *thread* berjalan sehingga ia disekat oleh satu peristiwa yang biasanya akan mewujudkan hentian kependaman yang lama. Hentian sedemikian mungkin disebabkan cache yang harus mengakses memori cip luar, yang mungkin mengambil beratus-ratus kitaran CPU untuk data yang dikembalikan. Daripada menunggu hentian diselesaikan, pemproses *threading* akan menukar pelaksanaan ke benang lain yang sudah sedia untuk dijalankan. Hanya apabila data untuk benang terdahulu telah tiba, benarkan data sebelumnya akan diletakkan pada senarai benang siap sedia.

4.5.2 *Barrel processor*

Tujuan *multithreading* adalah untuk menghapus semua ketergantungan data yang berhenti daripada saluran paip pelaksanaan. Oleh kerana satu benang bebas dari benang lain, terdapat kemungkinan kekurang satu arahan dalam satu peringkat talian paip yang

memerlukan output daripada arahan yang lebih lama dalam perancangan. Secara konseptual, ia serupa dengan *multitasking* primtif yang digunakan dalam sistem pengendalian; analogi adalah bahawa masa yang diberikan kepada setiap benang aktif adalah satu kitaran CPU.

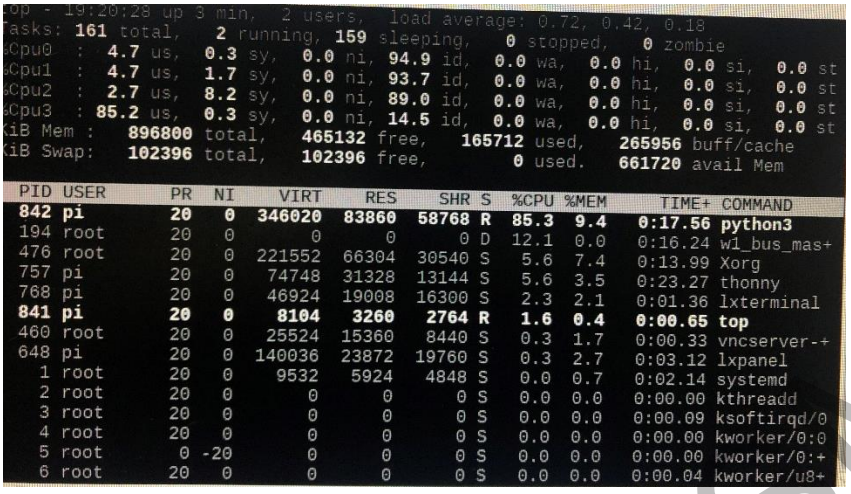
4.5.3 *Simultaneous multithreading*

Jenis *multithreading* yang paling canggih berlaku untuk pemproses *superscalar*. Sedangkan pemproses *superscalar* biasa mengeluarkan pelbagai arahan dari satu *thread* setiap kitaran CPU, dalam *multithreading* serentak (SMT) pemproses *superscalar* boleh mengeluarkan arahan dari pelbagai *thread* setiap kitaran CPU. Menyedari bahawa mana-mana *thread* tunggal mempunyai jumlah *terhad* paralelisme arahan, jenis *multithreading* ini cuba mengeksploitasi paralelisme yang terdapat di pelbagai *thread* untuk mengurangkan sisa yang berkaitan dengan slot isu yang tidak digunakan.

5 HASIL KAJIAN

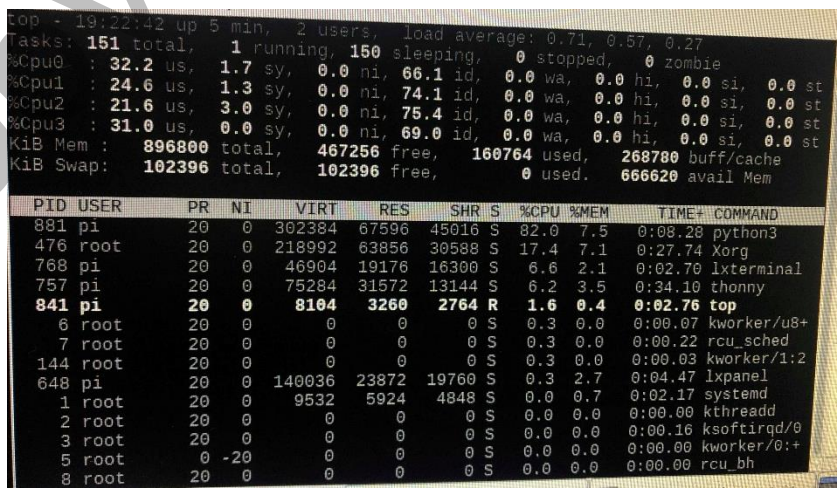
5.1 PERBANDINGAN PENGGUNAAN MEMORI CPU DENGAN MENGGUNAKAN KOD ASAL DENGAN KOD YANG DITAMBAH PENGATURCARAAN MULTITHREADING .

Rajah 5.10 menunjukkan kod asal yang dijalankan dengan pemantauan prestasi diambil semasa perjalankan kod tersebut. Pada masa ini hanya dua proses yang berjalan iaitu pengesanan model SVM dan pemantauan prestasi yang boleh dilihat pada gambar rajah atas. Daripada pemantauan sistem boleh melihat pada *process identifier* nombor 842 , penggunaan CPU adalah sebanyak 85.3% dengan memori sebanyak 9.4% oleh koding tersebut. Di atas rekod tersebut boleh ditunjuk bahawa 4 Cpu yang diguna. Dari situ penjaminan penggunaan teras dengan melihat nombor pada *us* iaitu *usage* dalam teras *Raspberry Pi*. Boleh dilihat bahawa hanya satu teras yang digunakan di sini dan susulan pengesanan masih lagi terjadi.



Rajah 5.10 Prestasi Raspberry Pi tanpa *Multithreading*

Rajah 5.11 menunjuk koding yang ditambah dengan pengaturcaraan *multithreading*. Pada masa ini hanya satu proses yang berjalan iaitu koding pengesanan model SVM. Dengan penggunaan *multithreading*, boleh dilihat penggunaan pada 4 teras Cpu telah pun dibahagikan dengan sama rata. Ini disebabkan setiap 1 model SVM menggunakan 1 benang untuk menjalankan proses daripada kod asal yang menggunakan 1 benang untuk menjalankan proses kesemua model SVM. *Usage* yang tidak lebih 50 dan kegunaan memori juga menurun daripada 9.4% kepada 7.5%. Penggunaan *multithreading* menunjukkan penambah baikan kepada prestasi.



Rajah 5.11 Prestasi Raspberry Pi dengan *Multithreading*

Jadual 5.1 Perbandingan prestasi benang dan tanpa benang

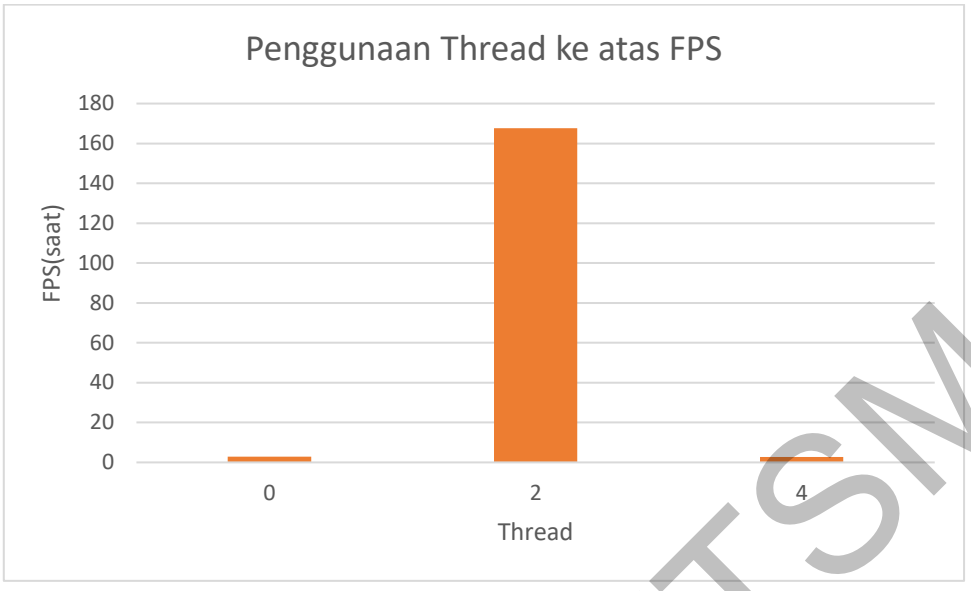
5.2 PERBANDINGAN PENGGUNAAN MEMORI CPU DALAM *FRAME PER SECOND* DENGAN KEGUNAAN *THREADING* DAN TANPA *THREADING.*

Parameters	*Without Multithreading*	*Multithreading*
Memori	*9.4%*	*7.5%*
Cpu Usage	*85.3%*	*82%*
Thread Used	*1*	*4*

Parameters	**Multithreading**		**Without Multithreading**
FPS	167.60	2.65	2.82
Memori	7.5%	7.6%	9.4%
Cpu Usage	86.2%	80.3%	85.3%
Thread Used	2	4	1

Jadual 5.2 Perbandingan prestasi benang dan tanpa benang dengan FPS

Rajah 5.2 menunjukkan graf penggunaan *thread* ke atas prestasi FPS. Ujian pertama menggunakan 0 *thread* dengan masa yang dicatat ialah 2.82 dan penggunaan 2 *thread* telah menyebabkan FPS meningkat dengan banyak iaitu 167.6. FPS pada kegunaan 4 *thread* menurun ke 2.65 disebabkan keberatan sistem untuk menjalan setiap satu *thread* yang mengandungi proses sebagai *thread* yang asing.



Rajah 5.2 Graf perbandingan penggunaan *thread* ke atas FPS

5.3 PERBANDINGAN RESOLUSI DENGAN KEGUNAAN *MULTITHREADING* PADA MEMORI DAN CPU.

Jadual 5.3 menunjukkan antara resolusi yang digunakan untuk menjalankan ujian untuk mengetahui bahawa adakah prestasi akan menambah baik jika resolusi meningkat atau pun menurun.

Resolution	Aspect Ratio	Framerates	FoV
160x128	4:3	30fps	PARTIAL
240x192	4:3	49fps	PARTIAL
640x480	4:3	42.1-60fps	FULL
1296x736	16:9	1-49fps	FULL

Jadual 5.2 Spesifikasi Resolusi

Parameters	**Resolution**			
	160x128	240x192	640x480	1296x736

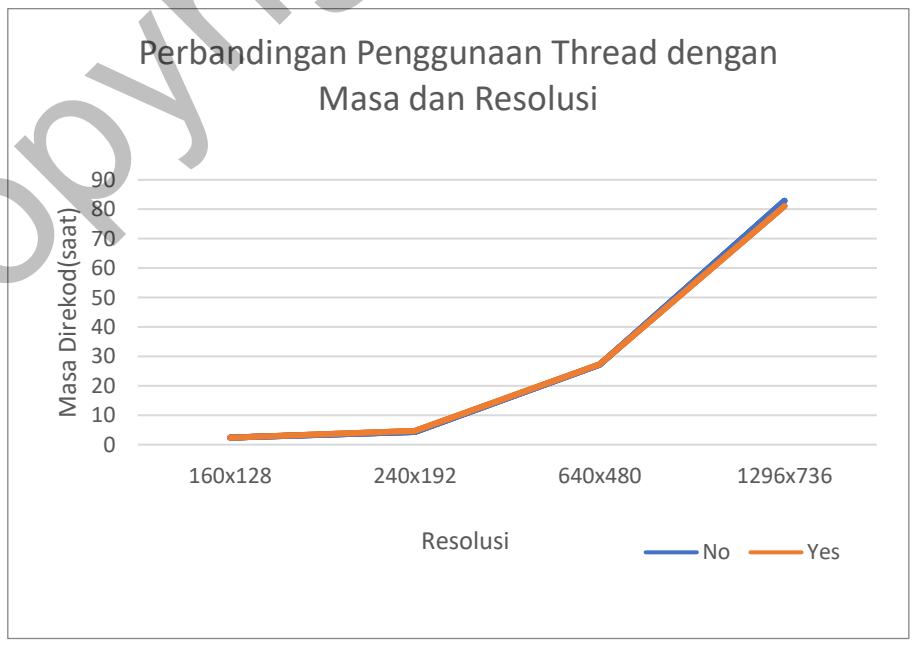
Time	2.39	4.66	27.33	81.1

Jadual 5.4 Prestasi sistem tanpa benang dengan pemboleh ubah resolusi

Parameters	**Resolution**			
	160x128	240x192	640x480	1296x736
Time	2.41	4.37	27.22	82.87
Memori	7.4%	7.7%	12.6%	9.3%
Cpu Usage	74.5%	93.2%	100.7%	99.7%

Jadual 5.5 Perbandingan prestasi menggunakan benang dengan pemboleh ubah resolusi

Rajah 5.30 menunjukkan graf meningkat dengan masa apabila penggunaan resolusi meningkat. Pebezaan penggunaan *thread* dengan tidak menggunakan *thread* hanyalah sedikit pada masa yang direkod tetapi penambah baik telah dibuat dalam sistem. Masa direkod mengikut 5 gambar yang direkod dan masa terakhir iaitu masa gambar kelima diambil digunakan untuk melakar graf. Boleh dilihat terdapat sedikit peningkatan dalam graf yang menggunakan *threading* berbanding tanpa *threading*.



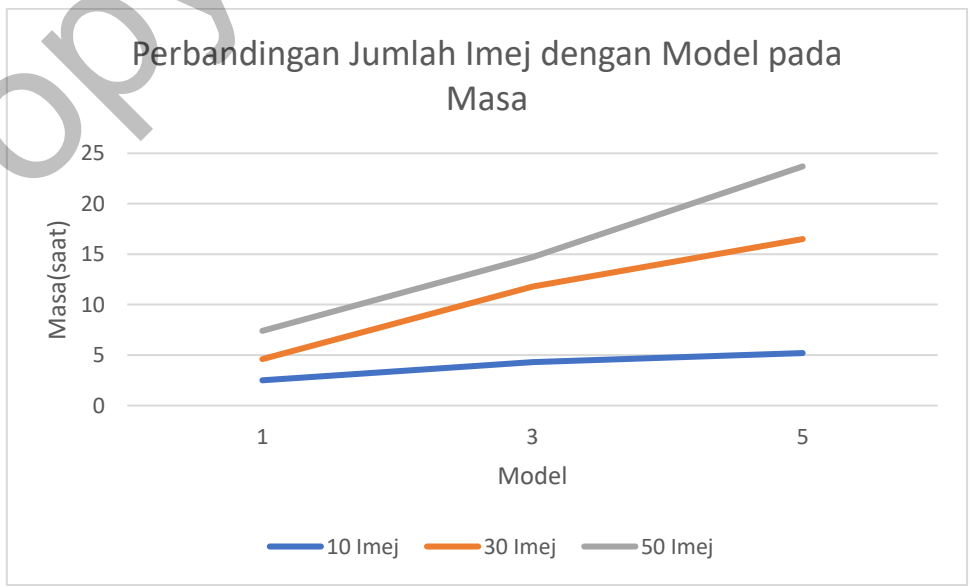
Rajah 5.30 Graf perbandingan penggunaan *thread* ke atas resolusi dengan masa

5.4 PERBANDINGAN JUMLAH GAMBAR PADA SATU MODEL DENGAN BEBERAPA MODEL DALAM SETIAP JUMLAH GAMBAR MENGGUNAKAN *MULTITHREADING*.

Model	Bilangan Imej/Masa Direkod(saat)		
	10	30	50
1	2.5	4.3	5.2
3	4.6	11.8	16.5
5	7.4	14.7	23.7

Jadual 5.6 Perbandingan prestasi menggunakan benang dengan pemboleh ubah bilangan model SVM

Rajah 5.40 menujukkan penggunaan threading untuk pemprosesan imej dengan meningkatkan imej dari 10 ke 50 dan peningkatan model SVM yang dilatih. Peningkatan masa pemprosesan buat 10 imej dalam model 1,3 dan 5 dengan seragam. Untuk imej 30 pula, masa meningkat dengan drastik pada model ke tiga dan meningkat dengan perlahan apabila ke model yang ke lima. Untuk 50 imej pula, masa meningkat secara selari dari modal 1 ke 5. Boleh dilihat pada modal yang ke 5 dengan 50 gambar yang menggunakan threading adalah lebih cepat berbanding pengesanan yang tidak menggunakan threading dengan 3 model dan 30 imej setiap model iaitu 29.33 masa pengesanan.



Rajah 5.40 Graf perbandingan no model dan jumlah imej ke atas masa.

6 KESIMPULAN

Pembangunan sistem *Smart Car* serta penujian unutk meningkatkan prestasi Raspberry Pi yang berfungsi sebagai sistem *Smart Car* ini dijangka untuk disiap pada masa yang ditentukan dengan menepati objektif untuk mencapai matlamat supaya dapat menguntungkan pengguna semasa menggunakan sistem. Dalam bab ini, kekangan dan cadangan penambahbaikan sistem untuk masa hadapan akan diusulkan.

Berdasarkan kajian yang dijalankan, terdapat beberapa cadangan yang dicadangkan untuk menambahbaikkan Sistem *Smart Car Robot*. Antaranya ialah:

- Menggunakan *deep learning* ataupun *multiprocessing* sebagai kaedah yang lain untuk projek ini.
- Menggunakan algorithma seperti Fractal yang *machine learning* yang lain seperti *Neural Network*.
- Penggunaan ROS sebagai satu metod untuk menjalankan pengesanan imej.

Kesimpulannya, Sistem *Smart Car Robot* akan dibangunkan berteraskan objektif kajian, keperluan pengguna dan reka bentuk aplikasi yang ditetapkan. Aplikasi ini diharapkan dapat memberi manfaat kepada pengguna jalan raya agar mereka dapat maklumat tentang papan tanda yang terdapat pada jalan raya juga dengan prestasi yang tinggi supaya pengguna dapat nikmati kecanggihan sistem. Di samping itu, sistem dalam Smart Car Robot akan berfungsi dengan ketepatan yang seharusnya supaya pengguna dapat menikmati kelancaran sistem tanpa kecuaian kesilapan sistem.

7 RUJUKAN

Alex Eames “Raspberry Pi 2 Performance Testing of the quad-core CPU”, Published on Feb 3, 2015 https://www.youtube.com/watch?v=f3vCVfxl0MA

Bilgin, Enis & Robila, Stefan. (2016). Road sign recognition system on Raspberry Pi. 1-5. 10.1109/LISAT.2016.7494102.
https://www.researchgate.net/publication/304189593_Road_sign_recognition_system_on_Raspberry_Pi

Mr. Vinston Raja .R1, Prem Kumar .D 2, Stanley Alfred .S 3, Thameem .M 4, "Accident Avoidance by Using Road Sign Recognition System" , International Research Journal of Engineering and Technology (IRJET) , Mar -2017 https://irjet.net/archives/V4/i3/IRJET-V4I3566.pdf

N Radhakrishnan1 , S Maruthi2, "REAL-TIME INDIAN TRAFFIC SIGN DETECTION USING RASPBERRY PI AND OPEN CV", International Journal of Advance Research in Science and Engineering, https://www.ijarse.com/images/fullpdf/1511377395_Bang281.pdf

Python Programming/Threading. (2018, December 7). Wikibooks, The Free Textbook Project. Retrieved 04:49, December 10, 2018 from https://en.wikibooks.org/w/index.php?title=Python_Programming/Threading&oldid=3497202.

xyz "Is a quad core really 4 times faster than a single core?" Thu Feb 19, 2015 11:10 pm, https://www.raspberrypi.org/forums/viewtopic.php?t=100974m/

Automatic image annotation. (2019, April 8). Wikipedia contributors. Retrieved 20 May 2019 12:54, 20 May 2019 from https://en.wikipedia.org/w/index.php?title=Automatic_image_annotation&oldid=891501149.

Support-vector machine. (2019, May 17). Wikipedia contributors. Retrieved 20 May 2019 13:24, from https://en.wikipedia.org/w/index.php?title=Support-vector_machine&oldid=897444209