PERANCANGAN LALUAN MENGGUNAKAN ALGORITMA A* UNTUK PELBAGAI DESTINASI

YONG VOON YAU
AZIZI ABDULLAH

Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia

ABSTRAK

Perancangan laluan merupakan satu penyelesaian bagi banyak masalah seperti kepintaran buatan seperti automasi robot. Bagaimana robot merancang laluan untuk masalah cari pelbagai destinasi? Kebanyakan algoritma tertumpu kepada penyelesaian titik tunggal. Kertas ini membincang bagaimana algoritma A* boleh diapliaksi kepada masalah perancangan laluan dan penambahbaikan untuk pelbagai destinasi. Satu perisian akan dibangunkan untuk mengambarkan perancagan laluan mengguna algoritma A* untuk pelbagai destinasi.

PENGENALAN

Perkembangan sains dan teknologi telan memperkembangkan konsep rumah pintar, di mana robot mudah alih memainkan peranan yang sangat penting. Terdapat keperluan yang sangat berat dalam kehidupan seharian untuk robot berfungsi secara berterusan dalam persekitaran yang mengandungi destinasi bercelaru dan berbilang. Perancangan laluan untuk robot mudah alih yang bekerja dalam persekitaran sedemikian adalah tugas yang mencabar dan telah menarik lebih banyak perhatian para sarjana di seluruh dunia. Manakalah ini mempersembahkan kaedah perancangan laluan berurutan dan berbilang bagi robot mudah alih dalam persekitaran dengan halangan dinamik. Masalah perancangan laluan boleh difahami sebagai masalah pengoptimuman dengan kekangan. Ia merujuk kepada bagaimana robot mencari satu optimum atau menghampiri laluan optimum dengan prestasi tertentu (seperti jarak terpendek, kurang masa, atau penggunaan tenaga minimum) dari titik permulaan ke titik destinasi dalam persekitaran dengan halangan.

Perancangan laluan adalah penentuan laluan bahawa robot mesti mengambil untuk melawati setiap titik dalam satu alam sekitar dan jalan adalah pelan lokus geometri menunjuk ke ruang yang diberikan di mana robot itu perlu dilalui. Secara amnya, masalah perancangan jalan adalah mencari laluan dengan menghubungkan lokasi yang berbeza dalam persekitaran seperti graf, maze dan jalan raya. Perancangan laluan “enable” robot mudah alih untuk melihat halangan dan menjana laluan optimum supaya elakkan mereka.

Terdapat banyak kajian dalam literatur mengenai perancangan laluan. Bergantung pada kaedah dan prinsip yang digunakan dalam perancangan laluan, kaedah perancangan jalur robot dibahagikan kepada dua jenis: kaedah perancangan jalur tradisional dan kaedah perancangan jalur pintar. Kaedah perancangan jalur tradisional termasuk "*visibility graph method, grid decoupling method, graph searching method, and artificial potential method*". Kaedah perancangan jalur pintar termasuk "*fuzzy logic , ant colony algorithm, neural networks, and genetic algorithm*".

Untuk menyelesaikan kekurangan algoritma ini, para sarjana telah banyak melakukan penyelidikan. Dalam sesetengah kerja tertentu, semut akan jatuh ke dalam perangkap

PENYATAAN MASALAH

Pada masa kini, walaupun banyak algoritma boleh menyelesaikan masalah perancangan laluan tetapi kebanyakan algoritma perancangan laluan seperti algoritma A * menggunakan satu titik hingga satu titik untuk alamat laluan antara titik. Apabila pelbagai destinasi untuk perancangan laluan diperlukan, algoritma-algoritma ni tak dapat menyelesai sempurna.

Pada masa kini, memang ada algoritma boleh menyelesai pelbagai destinasi. Tetapi ia menyelesaikan pelbagai destinasi dengan mengguna titik ke titik bermaksud algoritma ni akan mengira laluan dari titik A ke titik B, kemudian titik B ke titik C. Walaupun laluan dari titik A ke titik C kemudian ke titik B mungkin lebih pendek.

Kecuali ini, algotritma mungkin tidak dapat mengira laluan kalau ada halangan antara titik. Contohnya, titik A, B, C dan D. Antara titik B dan titik C ada halangan dan tak ada laluan antara titik B dan C. Tradisional algoritma akan mengira laluan sampai titik B sahaja, kerana ia mungkin tidak akan memintas titik yang tidak dapat dicapai.

OBJEKTIF

Objektif projek adalah:

I. Untuk memahami bagaimana dilaksanakan dan menyelesaikan perancangan laluan dengan pakai algoritma A* untuk masalah pelbagai destinasi.
II. Reka bentuk dan menbangunkan aplikasi yang menyelesai pelbagai destinasi
III. Mengesahkan aplikasi tersebut bahawa ia dapat navigasi robot ke pelbagai destinasi dengan berkesan dan dopercayai.

KAJIAN KESUSTERAAN

Algorithma Dijkstra

Algoritma Dijkstra, yang dikandung oleh ahli sains komputer Belanda Edsger Dijkstra pada tahun 1956 dan diterbitkan pada tahun 1959, adalah algorithma carian graf yang menyelesaikan masalah laluan terpendek tunggal-tunggal untuk graf dengan laluan pinggir bukan negative kos, menghasilkan pokok laluan terpendek. Algoritma ini sering digunakan dalam penghalaan dan sebagai subrutin dalam graf lain algoritma.

Untuk titik sumber (simpul) yang diberikan dalam graf, algoritma mendapati laluan dengan kos terendah (iaitu laluan terpendek) antara puncak itu dan setiap puncak lain. Ia juga boleh digunakan untuk mencari kos jalan terpendek dari puncak tunggal ke satu titik tujuan tunggal dengan menghentikan algoritma sekali laluan terpendek ke puncak destinasi telah ditentukan. Sebagai contoh, jika titik-titik grafik mewakili bandar dan kos jalan pinggir mewakili jarak memandu antara pasang kota yang dihubungkan dengan jalan langsung, algoritma Dijkstra dapat digunakan untuk mencari jalan terpendek antara satu bandar dan semua bandar lain. Akibatnya, jalan terpendek pertama digunakan secara meluas dalam protokol routing rangkaian, kebanyakannya terutamanya IS-IS dan OSPF (Buka Laluan Terpanjang Pertama).

Algoritma ini berfungsi seperti berikut:

1. Pilih sumber “vertex”
2. Tentukan set S untuk “vertices”, dan inisiakannya ke “emptyset”. Memandangkan algoritma itu berjalan, set S akan menyimpan “vertices” yang mana laluan terpendek telah dijumpai.
3. Labelkan titik sumber “vertex” dengan 0, dan masukkannya ke S.
4. Pertimbangkan setiap “vertex” yang tiada di S yang dihubungkan oleh “edge” dari “vertex” yang baru dimasukkan. Labelkan “vertex” yang tiada di S dengan label “vertex” + “length of edge”.
    a. Tetapi jika “vertex” tiada dalam S sudah dilabelkan, labelnya yang baru akan menjadi min (label baru adalah “vertex” + “length of edge”, label lama)
5. Pilih satu “vertex” tiada di dalam S dengan label terkecil, dan tambahkannya ke S.
6. Ulang dari langkah ke 4, sehingga destinasi “vertex” berada di S atau tidak ada titik berlabel tiada di S.

Algoritma A*

A * adalah algoritma carian yang dimaklumkan, atau carian terbaik, bermakna ia menyelesaikan masalah dengan mencari di antara semua jalan yang mungkin untuk penyelesaian (matlamat) yang menanggung kos terkecil (jarak paling rendah, masa terpendek). Ia digubal dari segi graf berwajaran: bermula dari nod tertentu graf, ia membina "tree of path" bermula dari node itu, meluaskan laluan satu langkah pada satu masa, sehingga salah satu laluannya berakhir pada nod gol yang telah ditetapkan. Pada setiap lelaran gelung utama, A * perlu menentukan yang mana laluan separa untuk berkembang menjadi satu atau lebih jalan yang lebih lama. Ia berbuat demikian berdasarkan anggaran kos (jumlah "weight") yang masih pergi ke nod gol. Khususnya, A * memilih laluan yang meminimumkan

$f(v) = h(v) + g(v)$

di mana n adalah nod terakhir di jalan, g (n) adalah kos laluan dari nod permulaan ke n, dan h (n) adalah heuristik yang menganggarkan kos jalan paling murah dari n ke matlamat. Heuristik adalah masalah yang khusus. Untuk algoritma untuk mencari laluan terpendek sebenar, fungsi heuristik mesti diterima, bermakna bahawa ia tidak pernah memlebihi kos sebenar untuk mencapai nod matlamat yang terdekat.

Pelaksanaan tipikal A * menggunakan giliran keutamaan untuk melakukan pemilihan nod kos minimum (anggaran) yang berulang untuk berkembang. Antrian keutamaan ini dikenali sebagai set terbuka. Pada setiap langkah algoritma, nod dengan nilai f (x) yang terendah dikeluarkan dari barisan, nilai f dan g jiran-jirannya dikemas kini dengan sewajarnya, dan jiran-jiran ini ditambah ke barisan. Algoritma berterusan sehingga nod matlamat mempunyai nilai f yang lebih rendah daripada mana-mana nod dalam baris gilir (atau sehingga barisan kosong). Nilai f bagi matlamat ialah panjang laluan terpendek, oleh kerana h pada matlamat adalah "zero" dalam heuristik yang boleh diterima.

Algoritma yang diterangkan setakat ini memberi kita hanya kepanjangan untuk laluan terpendek. Untuk mencari langkah-langkah sebenar, algoritma ini boleh disemak semula dengan mudah supaya setiap simpul di laluan menjejaki pendahulunya. Selepas algoritma ini dijalankan, nod akhir akan menunjuk kepada pendahulunya, dan sebagainya, sehingga beberapa pendahulunya nod adalah nod permulaan.

Algoritma D*

Seperti algoritma Dijkstra dan A *, D * mengekalkan senarai nod yang akan dinilai, dikenali sebagai “OPEN list”. Nod ditandakan sebagai mempunyai salah satu daripada beberapa negeri:

NEW, bermakna ia tidak pernah diletakkan pada “OPEN list”.

OPEN, bermakna ia kini berada di “OPEN list”.

CLOSED, bermakna ia tidak lagi dalam “OPEN list”.

RAISE, menunjukkan kosnya lebih tinggi daripada kali terakhir ia berada di “OPEN list”.

LOWER, menunjukkan kosnya lebih rendah daripada kali terakhir ia berada di “OPEN list”.

Pengembangan D*

Algoritma ini berfungsi dengan memilih susunan nod dari senarai OPEN dan menilainya. Ia kemudian menyebarkan perubahan nod ke semua nod jiran dan meletakkannya pada “OPEN list”. Proses penyebaran ini disebut "pengembangan". Berbeza dengan kanonik A *, yang mengikuti jalan dari awal hingga akhir, D * bermula dengan mencari mundur dari nod matlamat. Setiap nod diperluas mempunyai penunjuk belakang yang merujuk kepada nod seterusnya yang membawa kepada sasaran, dan setiap nod tahu kos yang tepat untuk sasaran. Apabila nod awal adalah nod seterusnya yang akan diperluaskan, algoritma dilakukan, dan jalan ke matlamat boleh didapati dengan hanya mengikuti backpointers.

Pengendalian halangan

Apabila halangan dikesan di sepanjang laluan yang dimaksudkan, semua mata yang terjejas akan diletakkan semula pada “OPEN list”, kali ini ditandakan RAISE. Sebelum nod RAISED meningkat dalam kos, bagaimanapun, algoritma memeriksa jiran-jirannya dan meneliti sama ada ia dapat mengurangkan kos simpul. Jika tidak, keadaan RAISE disebarkan kepada semua keturunan nod, iaitu nod yang mempunyai backpointers kepadanya. Nod ini kemudiannya dinilai, dan keadaan RAISE diteruskan, membentuk gelombang. Apabila nod RAISED dapat dikurangkan, pendukungnya dikemas kini, dan melepasi keadaan LOWER kepada jirannya. Gelombang-gelombang RAISE dan LOWER ini adalah jantung D *.

Pada ketika ini, keseluruhan siri-siri lain dihalang daripada menjadi "tersentuh" oleh ombak. Oleh itu, algoritma hanya bekerja pada mata yang terjejas oleh perubahan kos.

Satu lagi kebuntuan berlaku Kali ini, kebuntuan tidak boleh dilangkau begitu elegan. Tiada mata yang dapat mencari laluan baru melalui jiran ke destinasi. Oleh itu, mereka terus menyebarkan peningkatan kos mereka. Hanya di luar saluran boleh dijumpai mata, yang boleh membawa ke destinasi melalui laluan yang berdaya maju. Inilah bagaimana dua gelombang yang lebih rendah berkembang, yang berkembang sebagai titik-titik yang tidak dapat dijangkau dengan maklumat laluan baru.

Algoritma "Depth First Search (Dfs)"

Algoritma DFS adalah algoritma yang luas, sering digunakan sebagai blok bangunan untuk jenis topologi, penyambungan dan ujian planariti, di antara banyak aplikasi lain. Dalam masalah DFS "traversal" kami berminat mencari ibubapa, pra-pesanan dan pasca pesanan untuk setiap nod dalam graf. Contohnya, untuk graf pada Rajah 1.

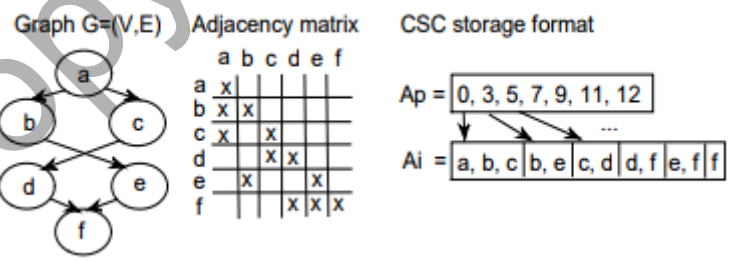
Hubungan nod- nod adalah :

nod = {a, b, c, d, e, f}

pra-pesanan = {0, 1, 4, 5, 2, 3}

post-order = {5, 2, 4, 3, 1, 0}

ibubapa = {∅, a, a, c, b, e}



Rajah 1. Contoh DFS

Peta (Maps)

Peta merupakan gambaran simbolik yang menekankan hubungan antara unsur-unsur objek, wilayah atau tema. Terdapat banyak jenis peta, dan boleh dikategorikan mengikuti sifat seperti peta statik, peta dinamik dan teknik perwakilan peta.

Peta Statik

Peta statik maksudnya merupakan pe ta yang tetap, biasanya adalah imej mandiri yang boleh dipaparkan. Halangan tidak akan tambah di dalam peta selepas lukisan peta adalah selesai walaupun halangan sudah ditambah di persekitaran yang sebenar. Peta statik tidak boleh berinteraksi dengan aplikasi lain, pengguna tak boleh mengubahsuai atribut peta.

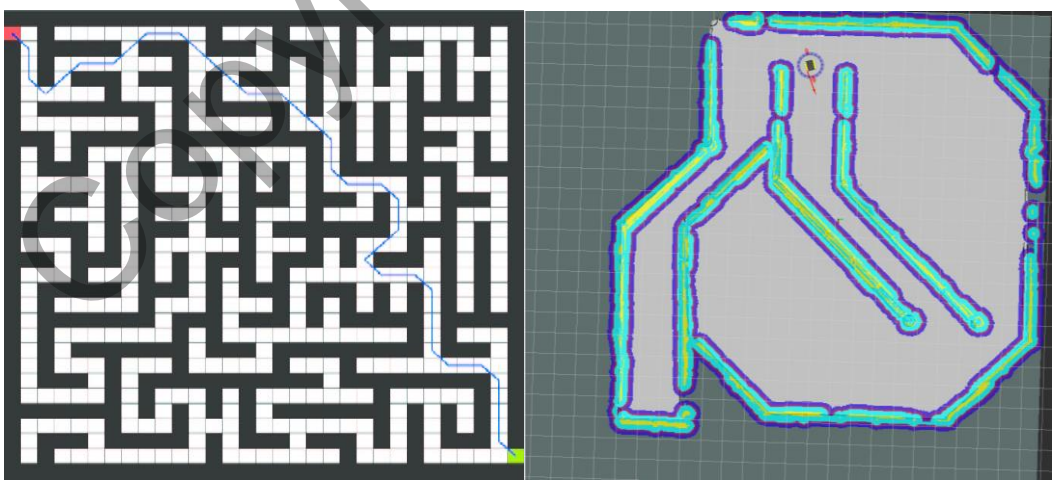
Peta Dinamik

Peta dinamik merupakan peta yang boleh berinteraksi dengan pengguna dan aplikasi lain. Contohnya, robot yang pakai peta dinamik akan menambahkan halangan dalam peta kalau robot mengesan barang yang muncul di depan dengan dikemas kini dalam masa nyata.

Occupancy Grid Map

Peta metrik ("Occupancy grid map") merujuk kepada keluarga algoritma konputer dalam "probabilistic robotics" untuk robot mudah alih yang menangani masalah menghasilkan peta dari data pengukuran yang berisik dan tidak pasti.

Kaedah ini mentakrifkan matriks 2D sel, setiap satu mewakili kawasan persegi tertentu dalam persekitaran. Pendekatan "occupancy grid map" menyerah kepada setiap sel kebarangkalian diduduki oleh halangan. Hasil pendekatan adalah matriks memori yang menyimpan kebarangkalian ini. Selain itu, kebaikan ini adalah kemudahan membuat dan mengemas peta. Rajah 2 menunjukan peta gird occupancy.

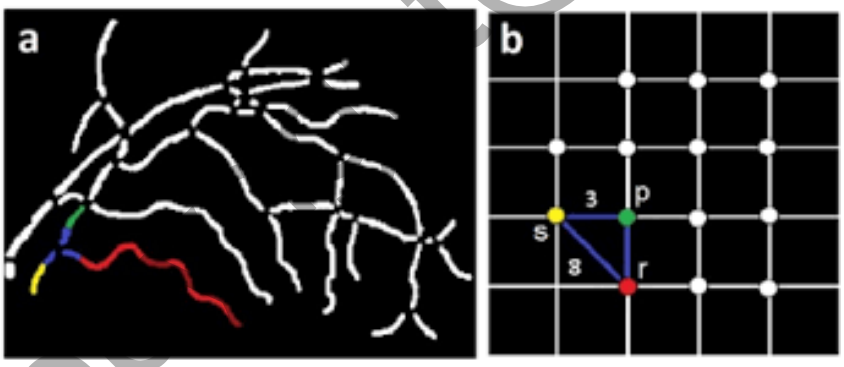


Rajah 2 Contoh Peta Grid Occupancy

GRAPH-BASED MAP

“Graph-based Map” adalah di mana paksi x dan y axis sesuai dengan jarak: kaki, meter, apa sahaja yang mahukan. Jadi, koordinat x-y mewakili lokasi semacam.

“Map graph” boleh diwakili sebagai gabungan "segi empat setengah grafik bipartite planar". Iaitu, biarkan G = (U, V, E) menjadi graf bipartit planar, dengan bipartition (U, V). Kuadrat G adalah grafik lain pada set vertex yang sama, di mana dua simpang bersebelahan di dalam alur apabila mereka paling banyak dua langkah di samping G. Separuh setengah persegi atau bipartit adalah subgraph yang diimbulkan dari satu sisi bipartisi (katakan V): set verteksnya adalah V dan ia mempunyai kelebihan antara setiap dua titik di V yang merupakan dua langkah di antara G. Separuh persegi ialah graf peta. Ia boleh diwakili secara geometri dengan mencari pelan planar G, dan memperluaskan setiap puncak V dan tepi bersebelahannya ke dalam rantau berbentuk bintang, supaya wilayah-wilayah ini menyentuh di puncak U. Sebaliknya, setiap “map graph” boleh diwakili sebagai setengah persegi dengan cara ini. Rajah 3 menunjukan peta berasaskan graf.



Rajah 3 “Graph-based Map”

REKABENTUK

Algoritma A * adalah satu algoritma perancangan laluan yang paling terkenal. Ia adalah algoritma komputer yang digunakan secara meluas dalam perancangan laluan dan graf traversal, proses merancang laluan yang diarahkan dengan berkesan di antara titik permulaan dan destinasi, yang disebut nod. Algoritma ini menggunakan carian heuristik dan mencari berdasarkan laluan terpendek. Ia ditakrifkan sebagai algoritma terbaik, kerana menggunakan formula di bawah untuk menilai nilai setiap sel dalam peta.

$$f(v)=h(v)+g(v)$$

Di mana:

h (v) : jarak heuristik antara sel sumber ke sel semasa.

g (v) : jaeak laluan dari sel semasa ke sel sasaran. f (v) :

setiap sel bersebelahan yang boleh dicapai akan dinilaikan.

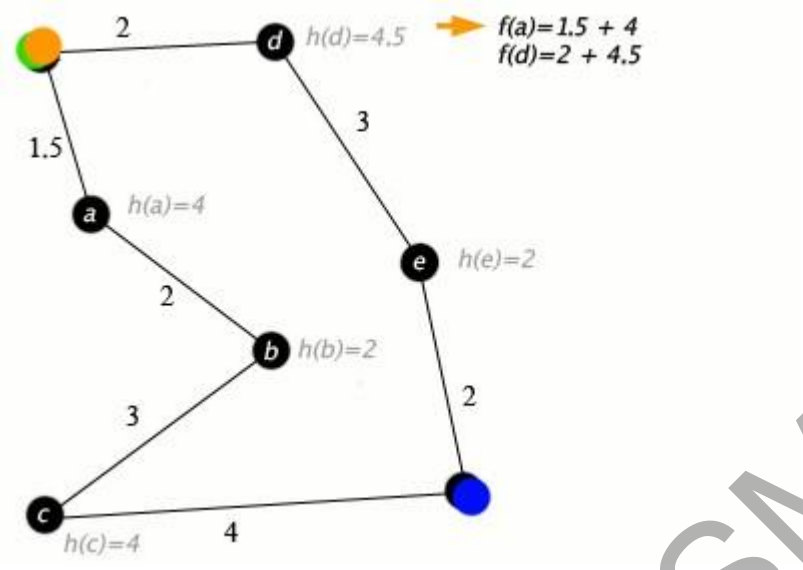
Sel-sel dengan nilai terendah f (v) dipilih sebagai urutan seterusnya. Kelebihan algoritma ini adalah bahawa jarak yang digunakan sebagai kriteria boleh diadaptasi, diubah suai atau jarak lain boleh ditambah dan memberikan pelbagai pengubahsuaian prinsip ini.

Algoritma A * berfungsi seperti ini:

1. Pada permulaan, tambah lokasi permulaan untuk senarai terbuka dan kosongkan senarai tertutup
2. Walaupun terdapat langkah-langkah seterusnya yang lebih mungkin dalam senarai terbuka dan kami belum menemui sasaran:
   a. Pilih langkah seterusnya yang paling mungkin (berdasarkan kedua-dua kos heuristik dan laluan)
   b. Keluarkannya dari senarai terbuka dan tambahkannya pada penutup
   c. Pertimbangkan setiap jiran dari langkah. Untuk setiap jiran:

   i. Kirakan kos laluan untuk mencapai jiran ii. Sekiranya kos kurang daripada kos yang diketahui untuk lokasi ini maka keluarkannya dari senarai terbuka atau tertutup (kerana sekarang kami telah menemui laluan yang lebih baik)

   iii. Jika lokasi tidak berada di dalam senarai terbuka atau tertutup maka rekod kos untuk lokasi dan tambahkannya ke senarai terbuka (ini bermakna ia akan dipertimbangkan dalam carian seterusnya). Catat bagaimana kami sampai ke lokasi ini.



Rajah 1.1 Contoh algoritma A*

Algoritma A* Cuma menyelesaikan laluan untuk satu destinasi. Untuk mencapaikan pelbagai destinasi, kaedah-kaedah akan dipakaikan. Dan kaedah yang akan pakai untuk menyelesaikan pelbagai destinasi adalah "Brute-force". Kaedah ini akan mengira semua laluan terpendek yang mungkin antara titik(termasuk titik permulaan dan titik destinasi) dan memilih laluan yang paling dekat. Algorithm di bawah menunjukan bagaimana "Brute-force" berfungsi:

- Memakai algoritma A * kepada setiap sepasang titik yang unik.
- Laluan terpendek dan jarak antara semua titik akan rekod.
- Akhirnya, pakai algoritma yang mencuba semua kombinasi jalan yang berbeza untuk mencari yang terbaik.

RAJAH KONTEKS

Dalam rajah konteks ini, ia menunjukkan interaksi antara pengguna dan komponen sistem untuk mendapatkan laluan terpendek. Di bawah adalah langkah untuk mensimulasikan:

1. Pengguna masuk imej ke sistem perancangan laluan.

2. Sistem akan hantar imej ke komponen proses imej untuk memproseskan imej ke peta metric.

3. Komponen proses imej kembalikan peta metrik ke sistem.

4. Pengguna menanda titik permulaan dan destinasi dalam peta metrik.

5. Sistem akan hantar peta metrik dan data(kedudukan titik dalam map) ke komponen algorithma A*.

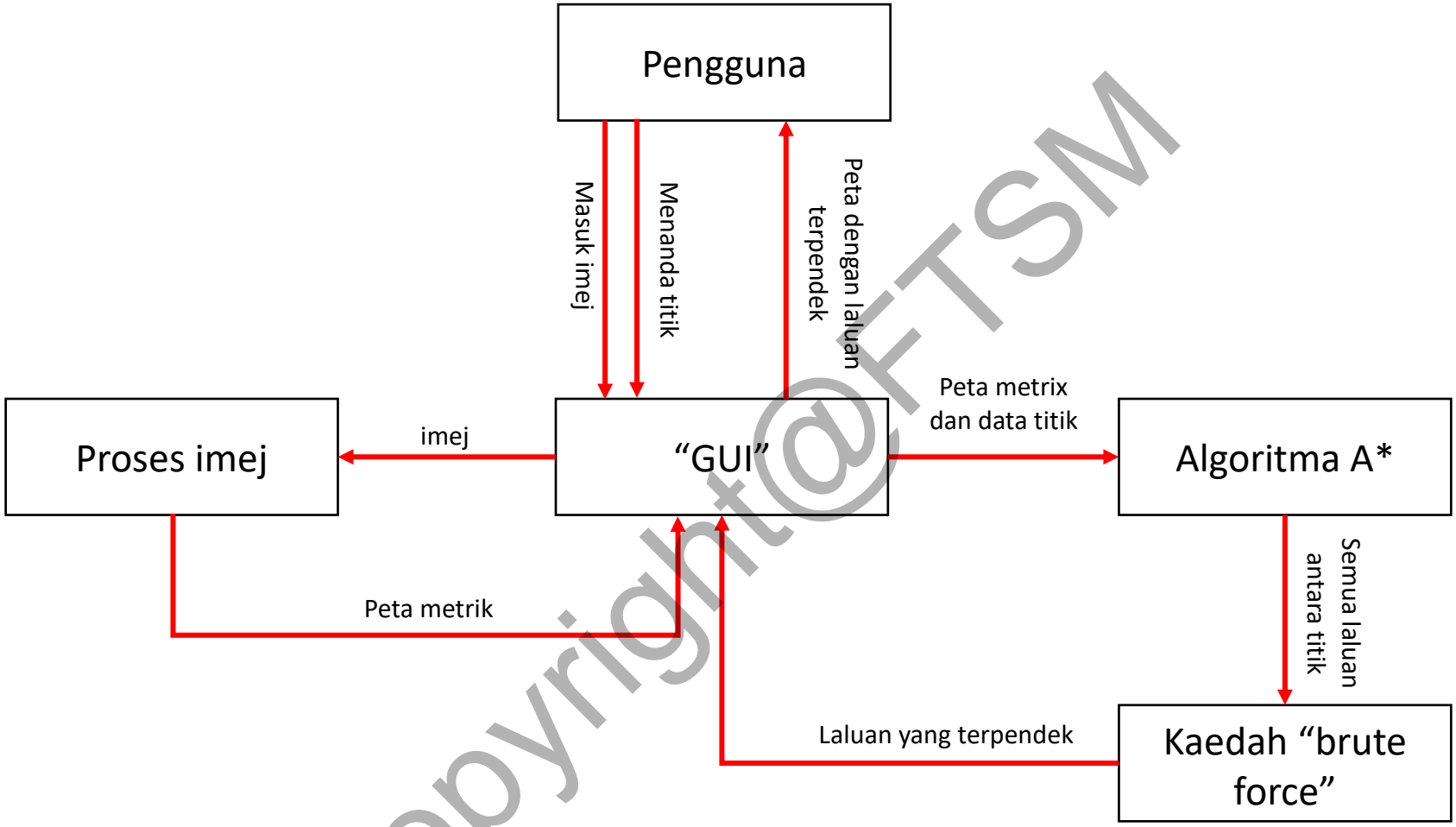
6. Komponen A* akan mengira semua mencari semua laluan terpendek yang mungkin antara titik dan hantar ke komponen “Brute Force”.

7. Komponen “Brute Force” akan cuba semua kombinasi laluan yang mungkin untuk dapat laluan terpendek.

8. Komponen “Brute Force” akan hantar keputusan ke sistem.

9. Pengguna dapat keputusan daripada sistem

Rajah 4 menunjukan gambarajah konteks bagi sistem Perancangan Laluan Menggunakan Algoritma A* untuk Pelbagai Destinasi



Rajah 4 menunjukkan rajah konteks tahap teratas

ALGORITHM KSELEURUHAN

Gambar rajah 4.6 merupakan rajah carta aliran untuk sistem ini. Bawah adalah langkah-langkah aliran untuk SPLA*.

1. Permulaan sistem
2. Pengguna memasukkan imej ke sistem.
3. Sistem akan memproses imej ke peta metric dan menunjuk di “GUI”.
4. Pengguna menanda titik permulaan dan destinasi.
5. Sistem pre-proses peta metric untuk memenuhi kekangan robot saiz.
6. Sistem akan mengira semua laluan antara titik.
7. Sistem akan mengguna kaedah “brute-force” untuk membanding jarak jauh antara titik untuk memilih jarak terpendek dan menyisihkan titik destinasi tersebut.
8. Sistem akan periksa sama ada laluan termasuk semua titik destinasi, ulang langkah 6 jika ada titik destinasi tidak dimasukkan. Sebaliknya, pergi langkah seterusnya.
9. Menunjukkan laluan terpendek kepada pengguna.

ALGORITHAM BRUTE-FORCE

Dalam algoritma ini, ia akan sentiasa menjalan jika pathList masih ada rekod. Dan ia akan keluarkan rekod yang tidak pakai lagi supaya tidak akan cari semula point_des yang sudah lalu.

```
Start = {s,ab,c,d,e}
point_des = {a,b,c,d,e}
FinalList = {}

Put A*Path in the pathList with point_start(point_des,distance)
while the pathList is not empty{
        while point_start = Start.haveNext(){
                compare all point_start(point_des,distance)
                shortpath = point_start of shortest path
                next = point_des of shortest path
                add shortpath to FinalList
                remove all shortpath(point_des,distance) in pathList
```

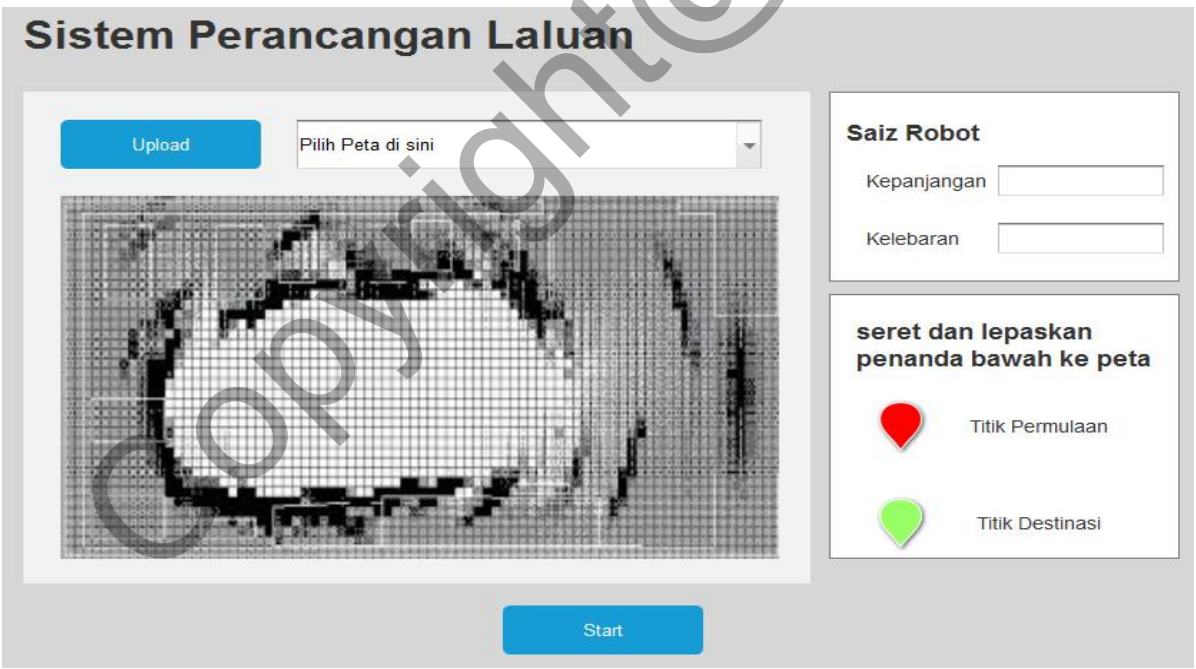
```
                remove all Start(next, distance) in PathLIst
                }
        }
Add all the distance in the FinalList
sketch to map follow the sequence of Final List
```

Rajah 4.6 menunjukkan Carta Aliran untuk sistem keseluruhan

REKA BENTUK ANTARAMUKA

Rajah 5 merupakan GUI utama untuk sistem ini. Dalam GUI ini, pengguna boleh memilih peta yang pernah pakai atau memasukkan peta yang baru ke sistem. Kemudian, peta metric akan menunjukkan dalam GUI ini, dan pengguna boleh "drag & drop" penanda berada di GUI Kanan.

Akhirnya, penguna boleh mulakan proses mencari laluan, ia akan membuka satu tingkah GUI baru selepas dapat laluan.



Rajah 5 GUI utama

KESIMPULAN

Tujuan projek ini adalah memahami perancangan laluan dengan membanding beberapa penyelesaian yang sedia ada dan keperluan untuk perancangan laluan. Dengan kefahaman ini, mereka bentuk sistem mengikuti keperluan yang ditentukan dengan mengguna algoritma yang sesuai. Hasil sistem untuk projek ini hendaklah memenuhi semua keperluan untuk mecari laluan yang terdekat dengan tepat.

ACKNOWLEDGEMENT

The authors would like to extend their appreciation and gratitude to FRGS/1/2016/ICT02/UKM/02/5.

RUJUKAN

A Hussein, A Al-Kaff, A de la Escalera, 2015. Autonomous Indoor Navigation of Low-Cost Quadcopters ROS, 2017 Coordinate Frames for Mobile Platforms http://www.ros.org/reps/rep-0105.html

Azad Noori, Farzad Moradi, 2015. Simulation and Comparison of Efficency in Pathfinding algorithms in Games, 232-237.

David Gonzalez-Arjona, Alberto Sanchez, Fernando Lopez-Colino, Angel de Castro and Javier Garrido, 2013. Simplified Occupancy Grid Indoor Mapping Optimized for Low-Cost Robots, 960-968

František DuchoĖ*a & Andrej Babineca & Martin Kajana & Peter BeĖoa & Martin Floreka & Tomáš Ficoa & Ladislav Jurišicaa, 2014. Path planning with modified A star algorithm for a mobile robot, 61-68.

Maritime University of Szczecin, 2012. A comparison between Dijkstra algorithm and simplified ant colony optimization in navigation, 26-28.

Nawaf Hazim Barnouti & Sinan Sameer Mahmood Al-Dabbagh & Mustafa Abdul Sahib Naser, 2016. Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm, 18-21.

P. Urcola, M. T. Lazaro, J. A. Castellanos and L. Montano, Generation of Probabilistic Graphs for Path Planning from Stochastic Maps, 2-3

Ross Graham & Hugh McCabe & Stephen Sheridan, 2003. Pathfinding in Computer Games", 64-67.

Rasika M. Kangutkar, 2017. Obstacle Avoidance and Path Planning for Smart Indoor Agents, 10-18.

Wikipedia, 2017. Occupancy grid mapping https://en.wikipedia.org/wiki/Occupancy_grid_mapping

Wikipedia, 2017. Map graph https://en.wikipedia.org/wiki/Map_graph

Mijo Cikeš, Marija Đakulovi and Ivan Petrovi, The Path Planning Algorithms for a Mobile Robot based on the Occupancy Grid Map of the Environment – A Comparative Study, 2-3 J.O. Wallgrün, 2010