Project Report
The Evolutionary Jackal

A project of the 2017 Robotics Course of the School of Information Science and Technology (SIST) of ShanghaiTech University
https://robotics.shanghaitech.edu.cn/teaching/robotics2017

Zhu Wangshu She Xinghao Chen Jinji Andre Rosendo
{zhuwsh, shexh, chenjj, arosendo}@shanghaitech.edu.cn

January 17, 2018

Abstract

Training Jackal robot to learn how to avoid obstacles in a confined space. This project will use neuron network and genetic algorithm with selection, crossover, and mutation to evolve the Jackal robot.

Introduction

We use the distance from three directions and put them into the simple neural network to output to the Jackal robot. By letting Jackal try different trajectories and directions, and then find the perfect route, iterative learning with genetic algorithm, so as to achieve the perfect obstacle avoidance.This project can have a wide range of applications, such as auto driving, such as sweeping robots, and we can also see Jackal becoming more and more intelligent in continuous learning, which is very interesting.

State of the Art

shexh

cite and present

[7] It is about the robot vision learning from nature animals. And the robot with some sensors uses to simulate the nature to make it complete the works like navigation.
[5] This paper is about the robot with evolutionary algorithm to set the control system. With simulation and in real work, the robot will know how to run in

different shape space.
[3] Similar to the paper before this, this paper is also about the robot with evolutionary algorithm. While it containing the neurons networks, spiking circuits and so on.

further

The further is about the third paper.This paper is based from the neurons communication in nature. The networks of spiking neurons can be implemented. The main carrier is a micro robot using spiking circuits. With the spiking circuits, it is easily to simulate the neurons and to control. So the neural circuit and evolutionary algorithm fit on the same chip in a few bytes of data memory. With the standard genetic algorithm, the robot can be encoded and evolved. And it can move forward and avoid the wall with the spiking networks after evolved. Further fitness gains correspond to faster and smoother trajectories. Then compare with a less generations robot, with none of the evolutionary runs could improve the fitness value, it will against the wall. The algorithm is not the main part here, and it will be talked in the others paper but it still be a main part to do this work. The authors genetically encode and evolve the signs of neurons and the connections among the neurons. The result is the robot Robots with lower fitness values tend to rotate more frequently and thus remain in the same area of the maze, but always manage to move away from the walls.

package

package:Jackal_desktop
This package is containing two packages which are Jackal_msgs and Jackal_viz. The Jackal_msgs is the messages exclusive to Jackal, especially for representing low-level motor commands and sensors. To command and monitor the robot, we need be able to use standard ROS interfaces. And the messages are between Jackal's ARM MCU and integrated PC. While Jackal_viz is visualization launchers and helpers for Jackal. It provides launchers and rviz configurations to assist with visualizing real or simulated Jackal from a desktop environment. It can help to bring up a specific rviz configuration. And some available configurations are default, navigation, gmapping and localization. The default configuration includes the RobotModel plugin, and displays Jackal in the odom frame, which will move the model about in rviz. The navigation displays Jackal in the odom frame and all the visualization information that local navigation provides including global plan, trajectory and local costmap. The gmapping configuration displays Jackal in the map frame. The local configuration also displays Jackal in the map frame and includes PoseArray plugin which display the localization system's uncertainty about the robot's pose

chenjj

cite and present

[2] This paper is about the prototype family of Alice miniature mobile robots and the improvements achieved so far. The research carried out with Alice and various real-world applications, which exceed the robot's use as a research prototype.

[6] This paper is about neuromorphic hardware model of a selective attention mechanism implemented on a very large scale integration (VLSI) chip, using analog circuits. And paper describes the characteristics of the circuits used in the architecture and present experimental data measured from the system.

further

[10] This paper introduce a framework for a class of algorithms for shortest path related problems in the presence of obstacles, such as the one-to-one shortest path problem, the one-to-many shortest paths problem and the minimum spanning tree problem, in the presence of obstacles. For these algorithms, the search space is restricted to a sparse strong connection graph that is implicitly represented and its searched portion is constructed incrementally on-the-fly during search. The time and space requirements of these algorithms essentially depend on actual search behavior. Therefore, additional techniques or heuristics can be incorporated into search procedure to further improve the performance of the algorithms. These algorithms are suitable for large VLSI design applications with many obstacles.

package

package:pointcloud_to_laserscan
This package converts a 3D Point Cloud into a 2D laser scan. This is useful for making devices like the Kinect appear like a laser scanner for 2D-based algorithms. pointcloud_to_laserscan packages helps to convert 3D sensor depth image to laser scan data. LaserScan assumes that all points are in a plane, namely the XY plane of the sensor coordinate system. The coordinates x,y,z can be computed from the sensor pose (position and orientation, which are defined by the tf frame stored in the header), the angle_min, the angle_increment, the range from ranges[] and its position in ranges[]. In our project, the package will subscribe the node date of point cloud, and receive the point cloud date from 0 to 180 degree, a total of 360 datas is taken at 0.5 degrees. Then record the minimum value of the same angle in the vlp-16 layer data, and finally publish the forward laser data to the /scan node. Then we select data->ranges[270] as sensorleft and data->ranges[180] as sensormid and data->ranges[90] as sensorright.

zhuwsh

cite and present

ALGORITHM 1 (Outline of an Evolutionary Algorithm)

$t := 0;$
initialize $P(0) := \{\vec{a}_1(0), \ldots, \vec{a}_\mu(0)\} \in I^\mu;$
evaluate $P(0) : \{\Phi(\vec{a}_1(0)), \ldots, \Phi(\vec{a}_\mu(0))\};$
while $(\iota(P(t)) \neq$ **true**$)$ **do**
recombine: $P'(t) := r_{\Theta_r}(P(t));$
mutate: $P''(t) := m_{\Theta_m}(P'(t));$
evaluate $P''(t) : \{\Phi(\vec{a}_1''(t)), \ldots, \Phi(\vec{a}_\lambda''(t))\};$
select: $P(t+1) := s_{\Theta_s}(P''(t) \cup Q);$
$t := t + 1;$
od

[1]This is the outline of an Evolutionary Algorithm. And I will talk about in more detail later.

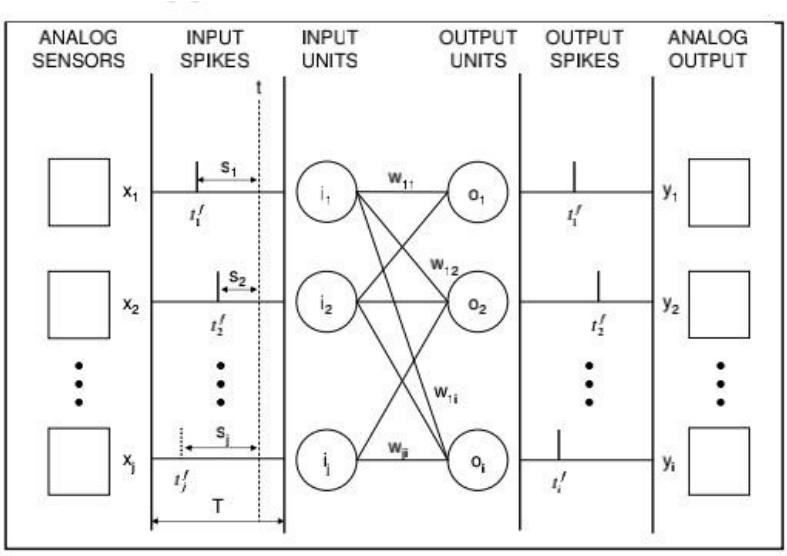

Figure 2. SRM SNN using delay coding architecture

[4]
This algorithm uses the many weights and inputs to combine the output linearly and optimizes the weights after evaluation.

[8] The neural network architecture was fixed and consisted of a single layer of synaptic weights from eight input units (clamped to the sensors) to two output units (directly connected to the motors) with mobile thresholds, logistic activa-

tion functions, and recurrent connections. Synaptic connections and thresholds were coded as floating point numbers on the chromosomes.

further

The first paper talks about three main streams of Evolutionary Algorithms (EAs), which are Evolution Strategies (ESs), Evolutionary Programming (EP), and Genetic Algorithms (GAs). They are some probabilistic optimization algorithms based on the model of natural evolution. The paper introduces the three algorithms to us and compared them in detail. The outline of an Evolutionary Algorithm has been given above. The three different algorithms use different probabilistic ways to mutate, recombine and select. But the outline is the same. The main steps are initialize, evaluate and continually doing the recombine, mutate, evaluate and select. It is very similar to the species evolution. Firstly there are some genes. Then we combine the best ones to generate the new generation. The new generations inherit some outstanding characteristic of they parents. Also they will randomly get some other features, which means the mutate. If the mutation is good, it will be left to the next generations. In every loop, they generate the new and better genes. Then after many generations, the genes can be better to adapt themselves to the environment.

package

package:ROS Navigation Stack package
I think the ROS Navigation Stack package may be very useful. It is about a 2D navigation stack that takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base. Use of the Navigation Stack on an arbitrary robot, however, is a bit more complicated. As a pre-requisite for navigation stack use, the robot must be running ROS, have a tf transform tree in place, and publish sensor data using the correct ROS Message types. It is all right because we will use the Jackal to do our project. Also, the Navigation Stack needs to be configured for the shape and dynamics of a robot to perform at a high level. It is said that the package is meant for both differential drive and holonomic wheeled robots only. It assumes that the mobile base is controlled by sending desired velocity commands to achieve in the form of: x velocity, y velocity, theta velocity. We may can use the differential drive of the two groups of Jackals wheels to get the x,y and theta. There are another two main hardware requirements, which are the a planar laser and square robot. That can be met by Jackal.

relevant to problem

present

[9]This paper is talking about the navigation system for mobile service robots working in urban environments. The system combines a 3D laser sensor with

2D algorithms for path planning and simultaneous localization and mapping, which is related to our project, 3D date is the huge amount of date that needs to be processed, which leads to new problems, to simplify the problem, in our project, we change 3D date into 2D date.
As regular 2D maps are not able to differ between obstacles and landmarks, the Colored 2D Map that is able to carry both types of information in a consistent way.
To reduce the computational costs of 3D algorithms, the paper use virtual 2D scans which allows an efficient combination of 3D perception and 2D localization, mapping and path planning, contributing the ability to use 3D data in form of virtual 2D scans not only for SLAM but also for path planning and navigation. To achieve the Colored 2D map was introduced. The benefit of this Colored 2D map is the ability to bold different information for localization and path planning in a consistent way.

System Description

Firstly we have the pointcloud2 data from velodyne vlp-16 equipped on Jackal Robot. By the pointcloud to laserscan node (cloned from github) we convert the pointcloud2 data to the laserscan data so we can easily get the sensor data of three different directions:in front of Jackal, left ahead and right ahead (figure 1).

Then we follow the outline of the Evolutionary Algorithm. We define each

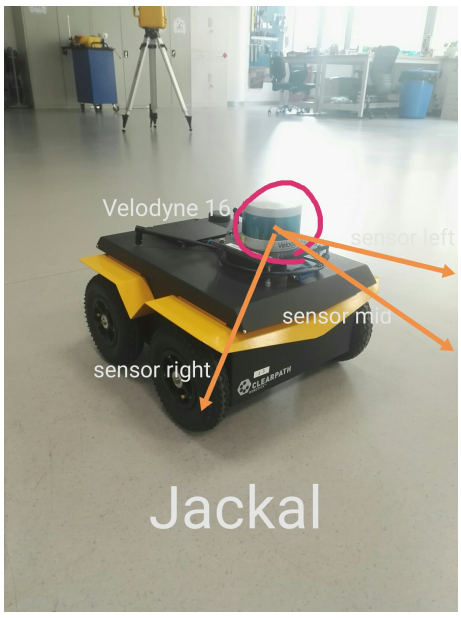

Figure 1

generation is a set of six individuals. For each individual we first randomly get fifteen weights. Use these fifteen weights and three values from laser scanner (after code 7 as follow)to linearly combine(figure 2) and output to the linear speed and turn speed.

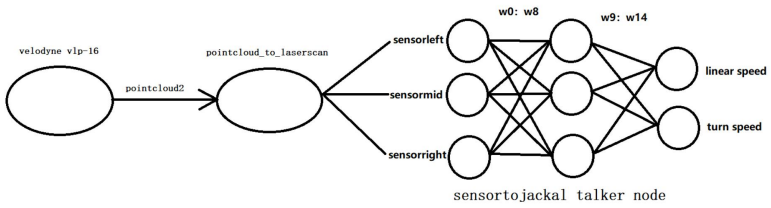

Figure 2

For each individual let the Jackal run until 20 seconds or it is too close to the obstacle. And record how far it can run as the fitness point. After six individuals have run the trial and they all get the fitness point. We will generate the next generation as following rules. The first two individuals are the best two in the former generation (which have two biggest fitness points). The next two individuals are generated by crossover which means each one has half of the weights from the best one and half from the second best one. The next two individuals are generated by mutation which means each one most of the weights from the best one and some random values. Repeat to run the new generation as the first generation continually. At last we may get a good gene (weights) adopting to the environment we made. There can be three main

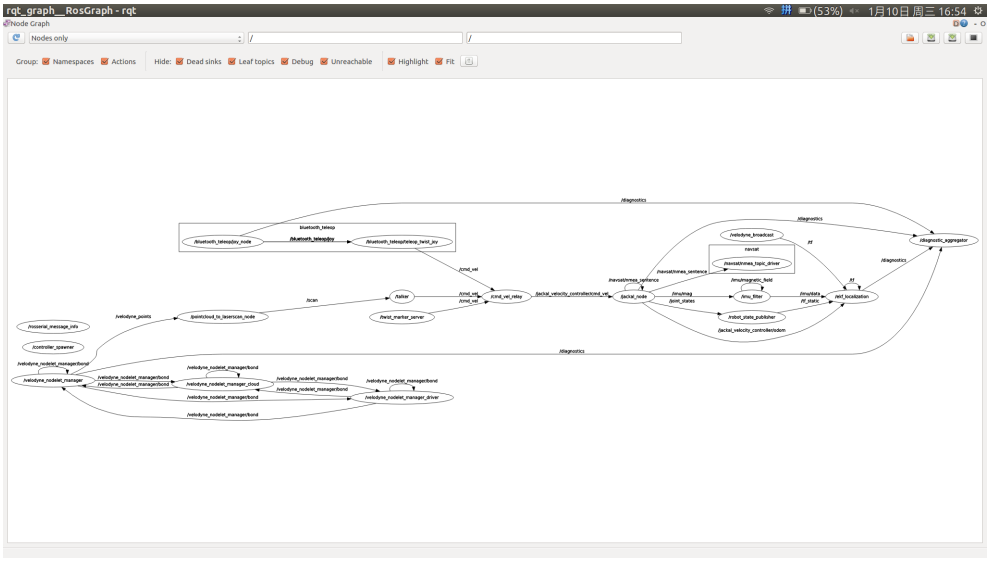

Figure 3:ROS System Structure

problems during the project. The first one is the combination algorithm (Use these fifteen weights and two values from laser scanner to combine and output to the linear speed and turn speed). The second one is the generation algorithm (We can choose some weights from the better two trials and combine to generate the new generation also with some random values). The last one is how to reset the Jackal after one trial. Ideally every trial should start at the same starting point. But reset by hand can be troublesome. Up to now we have written a series of problems codes to overcome most of the problems. We will describe how we solve them in detail by following codes.

c++cdoes

```
1.
typedef struct pop_weights{
    double weights[15];
    double fit;
}pop_weights;
void init(pop_weights *popu){
    for (int i=0;i<8;i++)
    {
        for (int j=0;j<15;j++)
            popu[i].weights[j]=rand() / double(RAND_MAX)-0.5;
    }
}
```

The first struct is defined to represent each individual trial which has fifteen weights and fitness point (fit). And the second part is the function to initialize the weights of the first generation which contains eight individuals.

```
2.
void corssover(pop_weights *popu){
    for (int i=0;i<7;i++)
    {
        popu[2].weights[i]=popu[0].weights[i];
        popu[2].weights[i+7]=popu[1].weights[i+7];

        popu[3].weights[i+7]=popu[0].weights[i+7];
        popu[3].weights[i]=popu[1].weights[i];
    }
    popu[2].weights[14]=popu[1].weights[14];
    popu[3].weights[14]=popu[1].weights[14];
}

3.
void mutation(pop_weights *popu){
    popu[4]=popu[0];
    popu[5]=popu[1];
```

```
    int j,k,l;
    l=rand()%15;
    for (int i=0;i<l;i++)
    {
    j=rand()%15;
        popu[4].weights[j]=rand() / double(RAND_MAX)-0.5;
    }
    l=rand()%15;
    for (int i=0;i<l;i++)
    {
    k=rand()%15;
        popu[5].weights[k]=rand() / double(RAND_MAX)-0.5;
    }
}
```

These two are the crossover and mutation codes. The crossover can get the forth and fifth individuals in the next generation. And mutation code can get the sixth and seventh individuals.The mutation code has changed to choose some of them by randomly choosing.

4.

```
output.linear.x=w4*(sl*w1+sm*w2+sr*w3)+w8*(sl*w5+sm*w6+sr*w7)+
w12*(sl*w9+sm*w10+sr*w11);
output.linear.y=0;output.linear.z=0;
output.angular.x=0;output.angular.y=0;
output.angular.z=w13*((sl)*w1+sm*w2+(sr)*w3)+w14*((sl)*w5+(sm)*w6+sr*w7)+
w15*((sl)*w9+(sm)*w10+(sr)*w11);
```

This is the combination algorithm. Use these fifteen weights and two values from laser scanner to combine and output to the linear speed and turn speed.

5.

```
if(countstep>=0){
...
backarray[0][countstep+tosetback]=-output.linear.x;
backarray[1][countstep+tosetback]=-output.angular.z;
...
if (countstep==timeofindividual){  //end each individual
      ...
      countstep=-counthitinterval;
      ...
    }
countstep++;
}
else {
```

```
      output.linear.x=backarray[0][-countstep];
      output.angular.z=backarray[1][-countstep];
      backarray[0][-countstep]=0;
      backarray[1][-countstep]=0;
      pub.publish(output);
      countstep++;
  }
```

To run the Jackal back to the origin automatically we wrote these codes. The backarray is the array to record the linear speed and the turn speed of the Jackal in each step. Then after the end of each individual it will read the array data one by one. Then the Jackal will run back to the point near the origin (which is not the origin exactly because of the error caused by the method limitation).

6.

```
void readweight(){                                          //read our weights
  ifstream in("/home/administrator/zsc_ws/weights.txt");  //need to change when on jackal
    string filename;
    string line;
    int i=0;
    int j=0;
    if(in) // there is the file
    {
        while (getline (in, line)) // get line
        {
          istringstream is(line);
          for(j=0;j<15;j++) is>>popu[i].weights[j];
          i++;
        }
    }
    else // no file
    {
        cout <<"no such file" << endl;
    }
    cout<<"weights"<<endl;
    for(i=0;i<6;i++) {
       cout<<i<<' ';
      for(j=0;j<15;j++) cout<<popu[i].weights[j]<<' ';
        cout<<endl;
    }
}
void file(){                                //save our weights
  FILE *p;
  if((p=fopen("weights.txt","wt"))!=NULL){
    for(int j=0;j<6;j++){
      for(int i=0;i<15;i++){
```

```
        if(i==14){
          fprintf(p,"%1f %s\n",popu[j].weights[i],"");}
        else{
        fprintf(p,"%1f %s",popu[j].weights[i],"");}
      }}}
      fclose(p);
}
```

The readweight code is to read weights from the weights.txt. And the file code is to store weights in the weights.txt. With the two functions we can easily have a look at the weights and save them for the deeper reseach.

7.

```
if (sl>2) sl=2;                              //only get the distance <=2
if (sm>2) sm=2;
if (sr>2) sr=2;
sl=2-sl;sm=2-sm;sr=2-sr;
```

We only get the distance <=2 and then change it to make it reasonable before put into the neural network.

8.

```
if((output.linear.x-formerlinearspeed)>speedmax/4)
output.linear.x=formerlinearspeed+0.05;
else if((output.linear.x-formerlinearspeed)<-speedmax/4)
output.linear.x=formerlinearspeed-0.05;
if((output.angular.z-formerangularspeed)>turnmax/4)
output.angular.z=formerangularspeed+0.05;
else if((output.angular.z-formerangularspeed)<-turnmax/4)
output.angular.z=formerangularspeed-0.05;
```

Also we have written the smoothly start and stop code to protect Jackal. We record the speed of the last step. If the speed increases or decreases too much, the code will limit the change. And the linear and angular speed limitation are also set globally.

System Evaluation

We put some boxes or some other obstacles in STAR Lab. Then let Jackal run and learn in this arena. A successful system can be go out of the arena with full speed and without crashing. Up to now the Jackal can turn left when meeting

Figure 4:Jackal in arena

the corner and avoid some big obstacles. And go out in average 2.5 generation in our 12 trials. And with the genetic algorithm we could see the improvement of Jackal.

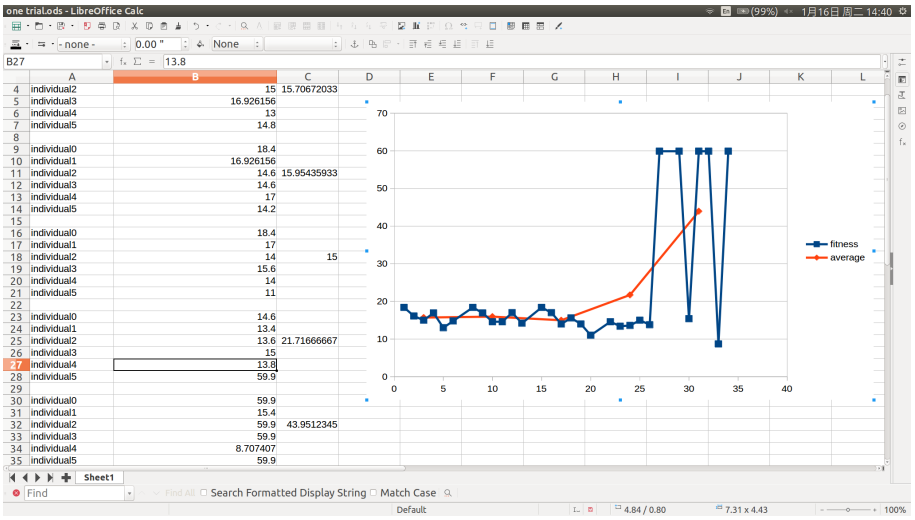

Figure 5:A typical trial

Conclusion

In this project our group use the Jackal robot equipped the velodyne vlp-16. We apply the genetic algorithm and neural network to the Jackal robot. Use the converted laser data from pointcloud(the velodyne vlp-16) and evolutionary weights to combine and out put to Jackals wheels. let Jackal continue to try and learn to optimize the gene of himself. At last Jackal can finally avoid obstacles and drive perfectly in a built space.

References

[1] Thomas Bäck and Hans-Paul Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1(1):1–23, March 1993.

[2] Gilles Caprari, Kai Arras, and Roland Siegwart. The autonomous miniature robot alice: from prototypes to applications. 09 2001.

[3] D. Floreano, N. Schoeni, G. Caprari, and J. Blynel. Evolutionary Bits'n'Spikes. In R. K. Standish, M. A. Beadau, and H. A. Abbass, editors, *Artificial Life 8*. MIT Press, 2002. In R. K. Standish, M. A. Beadau and H. A. Abbass (eds).

[4] H. Hagras, A. Pounds-Cornish, M. Colley, V. Callaghan, and G. Clarke. Evolving spiking neural network controllers for autonomous robots. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 5, pages 4620–4626 Vol.5, April 2004.

[5] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, 20(2):205 – 224, 1997. Practice and Future of Autonomous Agents.

[6] G. Indiveri. A neuromorphic VLSI device for implementing 2-D selective attention systems. *IEEE Transactions on Neural Networks*, 12(6):1455–1463, November 2001.

[7] Giacomo Indiveri and Rodney Douglas. Neuromorphic vision sensors. *Science*, 288(5469):1189–1190, 2000.

[8] Stefano Nolfi, Dario Floreano, Orazio Miglino, Francesco Mondada, Rodney Brooks, and Pattie Maes. How to evolve autonomous robots: Different approaches in evolutionary robotics. 01 1994.

[9] O. Wulf, C. Brenneke, and B. Wagner. Colored 2d maps for robot navigation with 3d sensor data. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2991–2996 vol.3, Sept 2004.

[10] S.Q. Zheng, Joon Shink Lim, and Sundararaj Iyengar. Finding obstacle-avoiding shortest paths using implicit connection graphs. 15:103 – 110, 02 1996.