Hexapod gait planning and obstacle avoidance algorithm

Yixuan Guo
University of Toledo

Follow this and additional works at: http://utdr.utoledo.edu/theses-dissertations

Recommended Citation
http://utdr.utoledo.edu/theses-dissertations/2085

This Thesis is brought to you for free and open access by The University of Toledo Digital Repository. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of The University of Toledo Digital Repository. For more information, please see the repository's About page.
A Thesis

entitled

Hexapod gait planning and Obstacle avoidance algorithm

by

Yixuan Guo

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the

Master of Science Degree in Mechanical Engineering

Dr. Manish Kumar, Committee Chair

Dr. Abdollah A. Afjeh, CommitteeMember

Dr. Mohammad Elahinia, CommitteeMember

Dr. Patricia R. Komuniecki, Dean

College of Graduate Studies

University of Toledo

May 201
An Abstract of

Hexapod gait planning and Obstacle avoidance algorithm

By

Yixuan Guo

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Mechanical Engineering

The University of Toledo

May 2016

The purpose of this research is to design a gait planning method for hexapod robots. On that basis, this thesis proposes a control strategy with which to achieve stable walking in complex and unknown environments.

First of all, the kinematics of the motion of the hexapod robot is analyzed on the basis of the hexapod robot’s mechanical structure utilizing the D-H method. Its forward and inverse kinematics equations are derived on the basis of the series-parallel structure. The workspace of the hexapod robot’s foot is analyzed. The kinematic equations are verified via simulation. The gait of the hexapod robot is planned. Strategies of tripod gait’s start-stop cycle as well as quadruped and five-feet gait start-stop in stance phase are proposed, which guarantees stable and continuous walking. By combining the method of foot planning trajectory and kinematic equations, several typical gaits are simulated in Adams, and the gait planning method is verified. A strategy of walking on complex terrains is proposed utilizing several gait combinations, which is also verified in Matlab. The research uses Arduino and a 32 channel servo controller in order to control each leg and servo.

Second, the research focuses on path planning of these robots. Mobile robot path planning technology is one of the important areas in mobile robot technology research.
Path planning is also the fundamental technique for mobile robots to complete other activities. This thesis discusses the topic of researching mobile robot obstacle avoidance, introduces the background of this research field and its significance and trends, and analyses several theories that are commonly used in mobile robot path planning. Meanwhile, it introduces several algorithms for mobile robot path planning, including a detailed description of the convex hull algorithm for obstacle avoidance under the OPENCV environment. The activity range of robot is a 2D plane which is a visual range from camera. The objective is to find a route from start point to target point that avoids the obstacles. The thesis uses a convex hull algorithm in order to calculate the shortest route from this visual range, which has multiple polygons.
Acknowledgments

My deepest gratitude goes first and foremost to my advisor Dr. Manish Kumar, for his constant encouragement and guidance. He always gave me most useful suggestion when I had problem in my research. I am extremely fortunate to have him as an advisor.

Also, I would thank my committee members, Dr. Mohammad Elahunia and Dr. Abdollah A. Afjeh. They gave me a lot of valuable comments and suggestions.

Last my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. I also owe my sincere gratitude to my friends and my fellow classmates who gave me their help and time in listening to me and helping me work out my problems during the difficult course of the thesis. I am really thankful to have a very good group in our CDS lab.
# Table of Contents

Acknowledgments ........................................................................................................ iii

1 Introduction ................................................................................................................ 1
   1.1 Subject Background and Significance ................................................................. 1
   1.2 Research Status .................................................................................................. 2
   1.3 Research Status of Hexapod Gait Control .......................................................... 6
   1.4 Main Topic of Research ....................................................................................... 6

2 Literature Review ........................................................................................................ 9

3 Kinematics Analysis .................................................................................................... 12
   3.1 Introduction ........................................................................................................ 12
   3.2 Structure Introduction ......................................................................................... 12
   3.3 Swing Leg Tandem Structure Position Analysis ................................................. 14
      3.3.1 Single Leg Analysis ..................................................................................... 14
      3.3.2 Inverse Kinematics Analysis ..................................................................... 17
      3.3.3 Speed Analysis ......................................................................................... 18
   3.4 Analysis Parallel Structure of Support Leg ....................................................... 18
      3.4.1 Kinematic Analysis ..................................................................................... 19
      3.4.2 Inverse Kinematic Analysis ..................................................................... 20
   3.5 Gait Planning and Simulation ............................................................................ 21
      3.5.1 Gait of Turning On the Spot ...................................................................... 24
      3.5.2 Trajectory Planning Base on the Typical Gait ........................................... 24
3.5.3 Trajectory Planning of Swing Phase .............................................. 27
3.5.4 Trajectory Planning of Stance Phase ............................................ 30
3.6 Kinematics Simulation ................................................................. 31
3.6.1 Three-legged Gait Walk Straight ................................................ 32
3.7 Chapter Summary ........................................................................ 36
4 Avoidance Algorithm ..................................................................... 37
4.1 Description of Obstacle Avoidance Algorithm .............................. 37
4.2 Several Obstacle Avoidance Algorithms Introduced ..................... 38
  4.2.1 Visual Graph Method ............................................................... 38
  4.2.2 Free Space Method ................................................................. 39
  4.2.3 Grid Method ........................................................................ 39
  4.2.4 Topological Method ............................................................... 40
4.3 Convex Hull Algorithm Description ............................................. 40
  4.3.1 Single Obstacle .................................................................. 42
  4.3.2 The Model of Multi Obstacles ................................................. 45
4.4 Convex Hull Algorithm Process ................................................... 46
4.5 Chapter Summary ........................................................................ 52
5 Robot Movement Control .............................................................. 53
  5.1 Serco Motor Control .................................................................. 53
    5.1.1 Servo Motor Zero Modulation .............................................. 54
    5.1.2 Servo Motor Rotation ......................................................... 59
  5.2 Robot Gait ................................................................................ 60
    5.2.1 Forward Movement .............................................................. 61
    5.2.2 Back Movement ................................................................. 64
    5.2.3 Turn Left .......................................................................... 64
Chapter 1

Introduction

1.1 Subject Background and Significance

In several operations or work environments, there exists a lot of regions that difficult to reach for human beings, these include outer space, mines, counterterrorism sites, ruins, burning building, and so on. Usually, researchers have developed several kinds of robots to complete exploration in these special environments. One kind of robot is the wheeled robot. The other kinds include track or crawling robots. However, the wheeled and crawling robots have limited capacity in terms of crossing barriers. They are efficient on relatively flat ground. This research focuses on multi-legged, walking robots. Multi-legged walking robot can cross the barriers easily, and a lot of freedom can make it work more flexibly. The adaptability of multi-legged walking robots is much better than that of wheeled and crawler robots. This is the research focus.

Multi-legged walking robots is bases on bionics, and improvements there of the structure and control methods. However as Canadian Scientists J.Angeles said, “Technological development of the walking robot is ahead of his theory and research”. [1] That is, we have all kinds of robots; however, the control method problem is still
difficult to solve.

We need to solve this problem using bionic research on multi-legged robots so they can adapt to the environment. This is the development direction of robot design and control theory.

Hexapod robots have superior stability. They can adapt to different environments and even balance when they lose one leg. Therefore, research and development of hexapod robots is very important.

1.2 Research Status

The earliest annals record about robots is a mechanical horse that was designed by Rygg in 1893. Since then, after hundreds of years of development, multiple bionic robot achievements have been made:

Research and development of hexapod

Hexapod Genghis was successfully developed in 1989. The main function of Genghis is outer planetary exploration, like the exploration of Mars. Genghis is 35cm long, has a 25cm span, and weighs 1kg. It has two rotation freedoms. Each degree of freedom is driven by a servo, which is based on position control. It is equipped with two inclinometers, two tactile sensors, six infrared vision sensors; therefore, Genghis can walk on the ground. Several years later, in an MIT artificial intelligence lab, a hexapod called Attila was designed with the same function in mind. Attila is second generation hexapod. It is 35cm long, has a 30cm span, and each leg has three degrees of freedom. One of the degrees of freedom is for climbing. It can support the weight of the robot. Use modular control system each leg and trunk control is as a separate process from the control module. It has 150 sensors, 23 drives, and 11 processors, which give it powerful detection and fault tolerance.

Shallow landmines are a big problem for troops, so the IS Robotics company
develop hexapod robots called Ursula [7] and Ariel [8] (show in Image 1-3 and 1-4). The main purpose of these robots is shallow mine clearance. Ursula is the first generation robot. Each leg has two rotation degrees of freedom, and each side of the body has three legs. Each leg can move up and down. The leg can locate barriers and cross them by using sensors. At the same time, the body is equipped with a pressure sensor, so it can respond to water.

Ariel is the second generation robot. Each leg has two rotation degrees of freedoms. Each DC can drive one degree of freedom. The hip joint can move $\pm 90^\circ$, and the knee joint can move $\pm 135^\circ$. This robot can not only can move forward and backward but also when it is upside down. Ariel can take water into its body in order to reduce buoyancy. After reaching land, the water can flow out. Compared to Ursula, Ariel has stronger abilities in terms of working and performing tasks.

In 1992, Martin Frik from Duisburg University started development of the hexapod called Tarry I [9]. It can walk on flat ground. If we put it in an unknown environment, it can independently probe to find the path. Each leg has three rotation degrees of freedom driven by R/C servos. In total, it has 18 degrees of freedom. In 1988, the second generation, Tarry II [10], started development. It has same structure as Tarry I, but uses a powerful motor, has a reduced body weight, each leg has a sensor on the end, and it is able to detect contact with the ground. The front portion has ultrasonic sensors in order can detect relatively large obstacles. Each leg has a force sensor. The body has an inclination gyro in order to make the position of the body true. Tarry II uses artificial neural networks called Walknet; this can make the robot go straight or curved and use different speeds and directions.

A scientist named Roger developed Robot I [11] (as shown in Image 1-6). Each leg has two degrees of freedom, one rotation degree of freedom, and one movement
freedom. Inverse solution of kinematics by leg controller, control the end of leg moving by given the trajectory. The robot’s gait coordination controller used the neural network controller gait design by Beer for the first time, and it uses the Walknet control system, which was design by Cruse.

Robot II [12] was design based on Robot I, and some changes were made. Each leg has four degree of freedom and three rotational freedoms. Each leg has three parts, and the drive of joint approve 4w, use potentiometer to measuring the angle of each joint. It uses strain gauge force sensors in order to detect axial force, and it uses a distribution control structure. Robot III [13] and Robot IV [14] were design based on the structure of cockroaches. In order to study the motion mechanism of cockroaches, two robots were made with the same structure and size. The foreleg has 5 degrees of freedom. The middle leg has 4degrees of freedom, and the hind leg has 3 degrees of freedom. Each joint has six strain gauges. It can accurately measure three-dimensional force. It also uses the Walknet control structure. Robot IV uses artificial muscles as drive. This method can reduce two-thirds of the weight compared to cylinder driven robots.

The University of California-Berkeley successful developed a six legged robot called R Hex [15.16]. Each leg has only one degree of freedom. The leg shape is camber. The robot has tri axial angle optical fiber and an accelerometer. It can detect the body position and adjust it at any time. It can detect the load at end point of each leg. This robot is easy to control, and it is good at running and climbing.

Stanford University developed a robot called RiSE [17.18]. The main function of this robot is climbing. Each leg has two degrees of freedom. This robot also has a fiber optic gyroscope (FOG) and accelerator in order to detect the position. Each joint integrated joint angular position sensor. The material of the leg is a new adhesive
material, and it also has a contact sensor on leg; therefore, this robot has an advantage in terms of climbing. RiSE and RHex are good at climbing.

As Canada scientist J.Angeles points out, now we have advantage on hexapod; however we are not good at control theory. Since 1960, the first hexapod which can use control system, the researcher of hexapod is focus on motion planning and control.

**Gait Planning and Research Status of Hexapods**

Since 1968, Mcghee and Frank have made a point about the fluctuation gait of quadruped robots. An increasing number of people are focused on hexapod robots.

People usually use the method of fixed gait, such as three-legged gait, quadruped gait, fluctuation gait, and so on.

The researchers got achievement about the gait planning this year. Here the most representative is two methods.

The rule based on particle German scientist Cruse by research of stick insect, propose six basic rules work on legs [19,20]. These six rules can work on adjacent legs. The condition of the leg can infect the other legs, and propose the concept of front limited condition and hind limited condition, by adjust the front and hind limited condition to find a best gait planning of the robot. Scientist Quinn also made achievements base on stick insects and proposed a control method for gait planning. Scientist Fielding proposes a rule base on manual inspection [21].According the test result to judge the right condition of the leg. The method of particle rule is only based on the simple planning of the legs and the environment the robot if in, simply and practical and has flexibility and robustness.

Base on the theory CPG use CPG method can get rhythm gait without sensor, simulation the lower nerve center of biological can make robot automatic generation.
the rhythm gait to control it walking, CPG has advantage on control the robot walking fast, the particle control is good at point control [22,23]. The scientist Joseph using CPG theory on robot Lobster to comply the rhythm stable gait, using the concept of finite state, control by the neuron structure.

1.3 Research Status of Hexapod Gait Control

Two Types Method of Hexapod Gait Control Structure:

In a centralized control structure, there is only one central controller in the robot system. It can collect all kinds of information from sensors, and then process information unified, complex the parameters, to planning the body and the end point of trajectory, including kinematic calculations, use this control method can solve the problem of robot environmental adaptability; however, this method is computationally intensive and complex. It is difficult to achieve [24].

Distributed control structure distributed control can according to different method to control different sub-control module to control the gait of robot. According to the different structure of distributed system can make the control method base on connection, base on function, base on behavior. The CPG control structure of Lobster is based on distributed control by interconnecting CPS in order to implement robot motion control. Cruse make appoint about that the Walknet control structure is based on distributed control structure. Walknet categorizes robots based on support, swing, coordination and so on. Brooks makes a point about inclusive systems based on different sub-control modules and robot classification by behavior [25]. Clear high-grade and low-grade behavior, the higher the level the higher the priority. It can control the low-grade behavior.

1.4 Main Topic of Research

The gait planning and control system of hexapod robots still have deficiencies,
especially in unknown environments. The topic of this thesis is proposed the method about gait planning and control system is stable environment. It includes the following points:

Kinematic analysis of hexapod kinematic analysis is to research the relationship between the position of the bunk and each angle of joint, and the kinematic analysis is the base of the control research. The structure of a hexapod is a combination of series of parallel structures, so we need to conduct research from perspective and parallelism by creating a D-H model, derivation the forward and inverse kinematic equipment of swinging leg and support leg and doing space analysis of leg.

Gait planning of hexapod first establish several parameters base on survey the stick insect, derivation the relationship between walking speed and gait pattern, analysis the classic gait, propose the interpolation method based on typical gait of the foot end.

Kinematic analysis of hexapod robot: Kinematic analysis of six-legged robot is the relationship between the position of robot torso and attitude of each robot joint angle. Kinematic analysis is also the basis for control. A hexapod has a complex combination of series and parallel structures in structure. It is necessary from the perspective of each series and parallel, through the establishment the D-H model, derivation the positive and inverse kinematics equations of swinging leg and support leg and foot end work space analysis.

Research on Obstacle Avoidance Algorithm: Path planning is part of portfolio optimization. The core model for path planning is the portfolio optimization model. This model is a kind of nonlinear 0-1 programming model. The core algorithm of path planning is the shortest path algorithm. There are many algorithms with which to calculate the shortest route, such as the Dijkstra algorithm, Floyd-Warshall algorithm,
the algorithm based on heuristics search called the A*(A-star) algorithm, the algorithm based on dynamic programming called the D* algorithm, neural network algorithm, and so on. The research purpose of this chapter is to find a shortest route from start point to target and avoid the obstacle in the plane scene with obstacles. Among them, any obstacle area, position, and shape can be calculated, and all the obstacles remain static. This scene is called a static scene. In this static scene, we use a convex hull algorithm in order to find the target and shortest path.
Chapter 2

Literature review

Hexapod robot, with abundant gait, redundant structure, and discrete ground contact, had high adapt ability and extensive application prospect when encountering special to topographical and unpredictable environment such as obstacles, ditches and soon. As an emerging research field, international research on the hexapod robot was still in its infancy, and no mature technologies could be referenced. Thus, it is necessary to review current achievements systematically and intensively to promote further development of hexapod robot. This paper surveyed the development status of hexapod robots. Key technologies of the hexapod robot were discussed in detail, including bionic structure design, gait planning, multi-legged coordinated control, and etc. The future development trends of hexapod robot were conducted vision and outlook in the end.

With the rapid development of science and technology, there is an urgent demand for a kind of robot that can work in special environments. Compared to wheeled and tracked robots, multi-legged robots, especially hexapods, are able to adapt to the environment, and at the same time, do less damage to the environment. Hexapod robots are flexible, and their redundant structure also guarantees walking stability, even after leg amputations. However, the unique structures of hexapod robots make
motion planning and control more complex, which becomes the key to the development of hexapod robots.

After walking robot hundred years of development, made great progress, to following summary, mainly through the following stage:

(1) The first stage, use mechanical and hydraulic control to make robot movement.
(2) The second phase, use computer technology to controlled robot.
(3) The third stage, robotics technology has entered a new stage of development with versatility and autonomy.

J.Angeles has his view in the literature [1], he said, “Technological development of the walking robot is ahead of his theory and research”. That is also said a control strategy is still a difficult problem to solve.

Schilling M, Cruse H, Arena P. present their views in the literature [2], multi-legged walking robot has a lot of types, the most representative of the robot with two feet, four feet, six and eight-foot robot, hexapod bionic robot as a typical one which has better superior stability than bipedal and quadruped robot. Hexapod able to adapt to different environments and has variety of gait.

With the development of bionics, the real Hexapod Robot has been produced from 1989, such as hexapod Genghis[5], Attila[6], Ursula [7]and Ariel [8]. Each leg has two rotation degrees of freedom, and each side of the body has three legs. Each leg can move up and down. The leg can locate barriers and cross them by using sensors. In 1992,Martin Frik form Duisburg University started development of the hexapod called Tarry I [9]. Each leg has a force sensor. The body has an inclination gyro in order to make the position of the body true. Tarry II uses artificial neural networks called Walknet; this can make the robot go straight or curved and use different speeds and directions.
The University of California-Berkeley successful developed a six legged robot called R Hex [15.16]. Stanford University developed a robot called RiSE [17.18]. The robot control method is relatively simple, and the ability to run and climbing.

Research results

Germany Cruise et al. [25, 26] by study stick insect, presented six basic rules working on each leg. Joseph et al [29,30] use of CPG theory generated rhythm steady gait on robot Lobster. Brooks et al. [34] various acts of the robot in accordance with the classification level, while clearly high-level and low-level behavior, the higher the rating, the higher the priority.
Chapter 3

Kinematics Analysis

3.1 Introduction

A hexapod robot is a kind of walking robot that has a high redundancy degree of freedom (DOF), coupling multi-branched, combination of series and parallel [26]. The hexapod robot has six legs. Each leg has three DOFs, so six legs have eighteen DOFs, and the redundancy degree makes the kinematics analysis more complex than in other general robots. The structure of a hexapod is the structure of a combination of series and parallel, when the legs of robot swing are serial mechanism, when the legs are supporting, the legs with the trunk is mechanism of parallel. The kinematics analysis of a hexapod involves analysis of the joint data of the eighteen legs, the position at the end of legs, and the mathematic relationship between trunk position and trunk posture.

From the perspective of series-parallel to derivation the function of forward and inverse kinematics, at the same time analysis the moving space of the end of leg, lay the next step theoretical foundation of gait planning.

3.2 Structure Introduction

This hexapod’s mechanical structure is based on bionics. Each leg has three rotational DOFs (three joints), and its six legs are symmetrically distributed on the two
sides of the trunk. From the image, this hexapod’s structure is based on a walking stick.

![Diagram of hexapod](image)

F3-1 leg model

In image 2-1-b, o is the origin point of the torso-base section coordinate, x is the coordinate axis, which is along the flank side of the trunk, y is the coordinate axis, which is along the direction of forward motion, z is the coordinate axis, which direction is vertical the trunk plant straight up. In order to establish these three axes, follow the right-hand rule. α is the rotation angle between the trunk and base section joint. β is the rotation angle between the base section and femur. γ is the rotation angle between the femur and tibia. ψ is tibia downward deflection angle.

Establish hexapod robot on the basis of walking insect. As shown in image 3-2, the robot consists of a trunk and symmetrical distribution of six legs (L1,L2,L3,R1,R2, and R3). Each leg has three parts: base section, femur, and tibia (the real robot cancel the base section but the distance between two steering engine cannot be ignored), and it also has three joints: torso-base section joint, base section-femur joint, femur-tibia joint (in the real robot, we use steering engine to instead joint).
F3-2 hexapod leg model

3.3 Swing Leg Tandem Structure Position Analysis

At the processing of walking, the leg is belonging to tandem structure. We need to identify the relationship between the angle of the steering engine and the position of each leg end.

3.3.1 Single Leg Analysis

F3-3 single leg D-H coordinates

Establish the D-H coordinate of the robot, as show in image2-3. Give six
number(1,2,3,4,5,6) to six relevant legs(L1,L2,L3,R1,R2,R3). We chose legR2 as NO.5 object. O_{50}-X_{50}Y_{50}Z_{50} is the torso-base section joint coordinates. O_{51}-X_{51}Y_{51}Z_{51} is the coordinates of the base section-femur joint. O_{52}-X_{52}Y_{52}Z_{52} is the coordinates of the femur-tibia joint. O_{53}-X_{53}Y_{53}Z_{53} is the coordinates of the end of the leg.

O_{b}-X_{b}Y_{b}Z_{b} is the trunk coordinates.

The robot leg adjacent joints D-H transformation matrix is:

\[
\begin{bmatrix}
\cos \theta_{ij} & -\sin \theta_{ij} & \sin \theta_{ij} \sin \alpha_{ij} & a_{ij} \cos \theta_{ij} \\
-\sin \theta_{ij} & \cos \theta_{ij} & \sin \theta_{ij} \cos \alpha_{ij} & a_{ij} \sin \theta_{ij} \\
0 & 0 & \cos \alpha_{ij} & d_{ij} \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

(3-1)

\(\theta_{ij}\) is the rotation angle of NO.ith leg joint; \(\alpha_{ij}\) is the included angle between the joint of No.ith leg and axis; \(a_{ij}\) is the length of the No.ith leg; \(d_{ij}\) is the space between the No.ith leg and each public vertical. The six legs have the same structure, so we can get each leg’s Denavit-Hartenbera matrix parameter, as show in chart3-1.

<table>
<thead>
<tr>
<th>joint</th>
<th>(\theta_{ij})</th>
<th>(\alpha_{ij})</th>
<th>(a_{ij})</th>
<th>(d_{ij})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(\theta_{i1})</td>
<td>(-90^\circ)</td>
<td>(l_1)</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>(\theta_{i2})</td>
<td>0</td>
<td>(l_2)</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>(\theta_{i3})</td>
<td>0</td>
<td>(l_3)</td>
<td>0</td>
</tr>
</tbody>
</table>

Chart3-1 the D-H matrix parameter of hexapod robot

Using the D-H parameters, we can calculate the transformation matrix of the adjacent joints of legR2:

\[
0_A^{51} = \begin{bmatrix}
\cos \theta_{51} & 0 & \sin \theta_{51} \cos \theta_{51} & l_1 \cos \theta_{51} \\
\sin \theta_{51} & 0 & -\cos \theta_{51} & l_1 \sin \theta_{51} \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

(3-1)
The transformation matrix of the torso-base section coordinates at the end of R2 leg.

\[
0_{A3} = 0_{A1} 1_{A2} 2_{A2}
\]

\[
\begin{bmatrix}
\cos \theta_{52} & -\sin \theta_{52} & 0 & l_2 \cos \theta_{52} \\
\sin \theta_{52} & \cos \theta_{52} & 0 & l_2 \sin \theta_{52} \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  \hspace{1cm} (3-2)

\[
2_{A3} =
\begin{bmatrix}
\cos \theta_{53} & -\sin \theta_{53} & 0 & l_3 \cos \theta_{53} \\
\sin \theta_{53} & \cos \theta_{53} & 0 & l_3 \sin \theta_{53} \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  \hspace{1cm} (3-3)

Because the six legs have the same structure and build the same coordinates, we can get the transformation matrix for the end of the No.\textit{i}th leg of torso-base section coordinates.

\[
0_{A3} = 0_{A1} 1_{A2} 2_{A2}
\]

\[
\begin{bmatrix}
\cos \theta_{12} & -\sin \theta_{12} & 0 & l_2 \cos \theta_{12} \\
\sin \theta_{12} & \cos \theta_{12} & 0 & l_2 \sin \theta_{12} \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  \hspace{1cm} (3-4)

We can calculate the base section joint coordinates of the position at the end at the No. \textit{i}th leg $\begin{bmatrix} 0_{p_{ix}}, 0_{p_{iy}}, 0_{p_{iz}} \end{bmatrix}^T$

\[
\begin{bmatrix}
0_{p_{ix}} \\
0_{p_{iy}} \\
0_{p_{iz}}
\end{bmatrix} =
\begin{bmatrix}
l_1 \cos \theta_{i1} + l_2 \cos \theta_{i1} \cos \theta_{i2} + l_3 \cos \theta_{i1} \cos \theta_{i2} \\
l_1 \sin \theta_{i1} + l_2 \sin \theta_{i1} \cos \theta_{i2} + l_3 \sin \theta_{i1} \cos \theta_{i2} \\
l_2 \sin \theta_{i2} + l_3 \sin \theta_{i1} \sin \theta_{i2}
\end{bmatrix}
\]  \hspace{1cm} (3-5)

We regard the trunk coordinates, $O_b$-\textit{X}_{b}\textit{Y}_{b}\textit{Z}_{b}$, as reference coordinates. The direction of axis \textit{X}_{b} is body transverse. Axis \textit{Z}_{b} is vertical to the trunk. Axis \textit{Y}_{b} is determined by the right-hand rule. The transformation matrix at the end of the leg coordinates and trunks coordinate:
\[ bA_{13} = bA_{10}^0 A_{13} = T_{\text{trans}}(x_{i0}, y_{i0}, z_{i0}) R(z, \psi_{i1}) R(y, \psi_{i2}) R(x, \theta_{i3}) \quad (3-6) \]

\( x_{i0}, y_{i0}, z_{i0} \) is the origin point of the base section joint coordinates at the trunk coordinates. \( \psi_{i1} \) is the angle between the base section joint and the horizontal plane. \( \psi_{i2} \) is the angle between the leg plane and the trunk’s x-z plane. We can determine the leg endpoint at the position of trunk coordinates.

\[
\begin{bmatrix}
  b p_{ix} \\
  b p_{iy} \\
  b p_{iz}
\end{bmatrix}^T = \\
\begin{bmatrix}
  x_{i0} - s\phi_{i1}(1s\phi_{i1} + l_2 s\phi_{i1} c\theta_{i2} + l_3 s\phi_{i1} c\phi_{i1} c\theta_{i23}) - c\phi_{i1}s\phi_{i2}(l_2 s\phi_{i1} c\theta_{i2} + l_3 s\phi_{i1} c\phi_{i1} c\theta_{i23}) + c\phi_{i1}c\phi_{i2}(l_1 c\phi_{i1} + l_2 c\phi_{i1} c\theta_{i2} + l_3 c\phi_{i1} c\theta_{i23}) \\
  y_{i0} + c\phi_{i1}(1s\phi_{i1} + l_2 s\phi_{i1} c\theta_{i2} + l_3 s\phi_{i1} c\phi_{i1} c\theta_{i23}) - s\phi_{i1}s\phi_{i2}(l_2 s\phi_{i1} c\theta_{i2} + l_3 s\phi_{i1} c\phi_{i1} c\theta_{i23}) + s\phi_{i1}c\phi_{i2}(l_1 c\phi_{i1} + l_2 c\phi_{i1} c\theta_{i2} + l_3 c\phi_{i1} c\theta_{i23}) \\
  z_{i0} - s\phi_{i2}(1s\phi_{i2} + l_2 s\phi_{i2} c\phi_{i2} c\phi_{i2} c\theta_{i23}) - c\phi_{i2}(l_1 c\phi_{i2} + l_2 c\phi_{i2} c\phi_{i2} c\theta_{i23})
\end{bmatrix}
\]

\[ (3-7) \]

### 3.3.2 Inverse Kinematics Analysis

As a result of inverse kinematics analysis, we know the position of the end point of leg at trunk reference coordinate system, and then Anti solve the angle of three joints at the leg.

\[ \theta_{i1} = \arctan \left( \frac{q_{iy}}{q_{ix}} \right) \quad (3-8) \]

\[ \theta_{i2} = \arctan \left( \frac{q_{iz} (l_2 + l_3 c\theta_{i3}) + l_3 s\theta_{i3} \sqrt{M_1^2 + N_1^2}}{q_{iz} l_3 s\phi_{i3} - (l_2 + l_3 c\theta_{i3}) \sqrt{M_1^2 + N_1^2}} \right) \quad (3-9) \]

\[ \theta_{i3} = -\arccos \left( \frac{M_1^2 + N_1^2 + q_{iz}^2 - l_3^2}{2l_2 l_3} \right) \quad (3-10) \]

There \( M_i = q_{ix} - l_1 c\theta_{i1} \),

\[ N_i = q_{iy} - l_1 s\theta_{i1} \quad (3-11) \]

\[ q_{ix} = c\phi_{i1} c\phi_{i2} p_{ix} + s\phi_{i1} c\phi_{i2} p_{iy} - c\phi_{i2} p_{iz} - c\phi_{i1} c\phi_{i2} x_{i0} - s\phi_{i1} c\phi_{i2} y_{i0} + s\phi_{i2} z_{i0} \quad (3-13) \]

\[ q_{iy} = -s\phi_{i1} p_{ix} + c\phi_{i1} p_{iy} + s\phi_{i1} x_{i0} - c\phi_{i1} y_{i0} \quad (3-14) \]
\[ q_{i2} = c\phi_{i12}p_{i2} + c\phi_{i11}s\phi_{i2}p_{ix} - c\phi_{i11} s\phi_{i2}x_{i0} + s\phi_{i11} s\phi_{i2} p_{iy} - s\phi_{i11} s\phi_{i2} y_{i0} - c\phi_{i12} z_{i0} \]

(3-15)

Here, we can use end point positions to calculate the rotation angle of the three joints.

### 3.3.3 Speed Analysis

Speed analysis is the analysis of the relationship between the end point line velocity and the angular velocity. We can determine the relationship using the speed Jacobin matrix \( J(\theta) \):

\[
\begin{bmatrix}
    v_{ix} \\
    v_{iy} \\
    v_{iz}
\end{bmatrix}
= J(\theta_i)
\begin{bmatrix}
    \dot{\theta}_{i1} \\
    \dot{\theta}_{i2} \\
    \dot{\theta}_{i3}
\end{bmatrix}
\]  

(3-16)

\( v_{ix}, v_{iy}, v_{iz} \) is the line speed of the x, y, z axis direction, and \( \dot{\theta}_{i1}, \dot{\theta}_{i2}, \dot{\theta}_{i3} \) means the angular velocity of joint1 joint2 and joint3. The speed Jacobin matrix of I leg is

\[
J_{mn}(\theta_i) = \frac{\partial p_m(\theta_i)}{\partial \theta_{in}} (m=1,2,3 \ n=1,2,3)
\]  

(3-17)

\( p_m(\theta_i) \) is the coordinate of end point.

\[
J(\theta_5) = \begin{bmatrix}
    -s\theta_1 (l_3 c\theta_{523} + l_2 c\theta_{52} + l_1) & c\theta_1 (l_3 s\theta_{523} - l_2 s\theta_{52}) & -l_3 c\theta_1 s\theta_{523} \\
    c\theta_1 (l_3 c\theta_{523} + l_2 c\theta_{52} + l_1) & s\theta_1 (-l_3 s\theta_{523} - l_2 s\theta_{52}) & -l_3 s\theta_1 s\theta_{523} \\
    0 & -l_3 c\theta_{523} - l_2 c\theta_{52} & -l_3 c\theta_{523}
\end{bmatrix}
\]  

(3-18)

We can calculate the end point linear speed from the angular velocity of the joint. Using the same way, we can also calculate the angular velocity of a joint from the end point linear speed.

\[
\begin{bmatrix}
    \dot{\theta}_{i1} \\
    \dot{\theta}_{i2} \\
    \dot{\theta}_{i3}
\end{bmatrix}
= [J^{-1}(\theta_i)]
\begin{bmatrix}
    v_{ix} \\
    v_{iy} \\
    v_{iz}
\end{bmatrix}
\]  

(3-19)

### 3.4 Analysis Parallel Structure of Support Leg

When the hexapod robot is walking, there must be more than three legs in contact with ground in order to maintain stability. At this time, the support leg, the robot trunk, and the ground constitute a parallel coupling mechanism.
Kinematics Analysis is done according to the position of the end leg and the rotation angle of each joint to solve the position of the trunk [27]. Inverse kinematics analysis is done according to the position of the trunk and end point of the leg in order to calculate the rotation angle of the joint.

### 3.4.1 Kinematic Analysis

When there are N (N≥3) support legs, the robot trunk, support leg, and the ground can constitute the closed kinematic chain D-H coordinate. We can use three legs to analyze. As shown in the image, the support legs are L1, L3, and R2.

[Image of parallel D-H coordinate system]

F3-4 Parallel D-H coordinate system

$\sum o_b$ is the trunk reference coordinate system, $o_b$ is the center point of the robot, $x_b$ is the direction parallel to the direction of the robot moving forward. It is also point to the right side axis. $y_b$ is the axis which the direction is the robot forward direction. $z_b$ is the axis which the direction is vertical the trunk plane and point up. Comply with the right-hand screw rule. $\sum o_{im}(n=0\sim2)$ means the joint n in the leg i. $\sum o_{i3}$ represent the reference coordinate system of end point of leg i. $\sum o_G$ is the reference coordinate system of the ground. The body coordinate $\sum o_b$ relative to ground coordinate system $\sum o_G$ transformation matrix is:
If \((X_b, Y_b, Z_b)\) is the body coordinate that is under the ground reference coordinate, and the initial position of the body is same with the ground reference coordinate system. First, let the body coordinate system \(\sum_o\) rotation \(\alpha\) degree around z axis, rotation \(\beta\) degree around y axis and then rotation \(\gamma\) degree around x axis. We can get a new coordinate system after translation \((X_b, Y_b, Z_b)\):

\[
G_{T_b} = \begin{bmatrix}
    n_{bx} & o_{bx} & a_{bx} & x_b \\
    n_{by} & o_{by} & a_{by} & y_b \\
    n_{bz} & o_{bz} & a_{bz} & z_b \\
    0 & 0 & 0 & 1
\end{bmatrix} = G_{T i3}^{-i3} A_{i2}^{-i2} A_{i1}^{-i1} A_{i0}^{-i0} T_b \tag{3-20}
\]

If we know the rotation angle of each joint of a support leg, then from the joint rotation angle and the robot structure parameters, we can get the end point coordinate \((b^\upsilon_i, b^\nu_i, b^\omega_i)\) in the body coordinate, and the end point coordinate \((G^X_i, G^Y_i, G^Z_i)\):

\[
[G^X_i, G^Y_i, G^Z_i, 1]^T = G_{T_b} [b^\upsilon_i, b^\nu_i, b^\omega_i] \tag{3-22}
\]

When the three support legs (L1, L3, R2) are in contact with the ground,

\[
\begin{bmatrix}
    G^X_1 & G^X_3 & G^X_5 \\
    G^Y_1 & G^Y_3 & G^Y_5 \\
    G^Z_1 & G^Z_3 & G^Z_5
\end{bmatrix} = G_{T_b} \begin{bmatrix}
    b^\upsilon_1 & b^\upsilon_3 & b^\upsilon_5 \\
    b^\nu_1 & b^\nu_3 & b^\nu_5 \\
    b^\omega_1 & b^\omega_3 & b^\omega_5
\end{bmatrix} \tag{3-23}
\]

we can get six parameters of positions, \(X_b, Y_b, Z_b, \alpha, \beta, \gamma\).

### 3.4.2 Inverse Kinematic Analysis

Inverse kinematic analysis is from the position and attitude of the body and the position of leg end point, to solve the joint rotation angle of the support leg.

\[
[G^X_i, G^Y_i, G^Z_i, 1]^T = G_{T_b}^{-i0} A_{i1}^{-i1} A_{i2}^{-i2} A_{i3}^{-i3} T_0 \tag{3-24}
\]
3.5 Gait planning and simulation

During the walk, the leg of the robot is a regular periodic motion, in each periodic it has two conditions, one condition is swing state, this state include leg lift up, forward swing and fall down on the ground [28]. The other state is support state, it means the leg contact the ground, the leg load the robot capacity and swing back, support the robot until it into swing state and lift up the ground.

To analysis the relationship between the walking velocity and different state of this kind of regular periodic motion robot. We define the concept of some parameter below.

(1) **Transform position** it means the transform position of the two states of the leg during the hexapod walking process, the transform position of the leg from the swing state to support state is called ahead transform position (ATP), otherwise, the transform position of leg from support state to swing state, is called rear transform position (RTP).

(2) **Extremity position** it means during the walking process, the legs comparatively of hexapod can swing the maximum distance, is called extremity position. The transform position and the extremity position maybe not the same position, sometime the position is coincide and sometime is not coincide.

(3) **Stance time** ($t_{\text{stance}}$) the single leg in the period of swing state-support state, the time of the support leg is called stance time.

(4) **Swing time** ($t_{\text{swing}}$) the single leg in the period of swing state-support state, the time of the swing state is called swing time.

(5) **Covering coefficient** ($\beta$) the single leg in the period of swing state-support state, the ratio of support time to total time.
\[ \beta = \frac{t_{\text{stance}}}{T} = 1 - \frac{t_{\text{swing}}}{T} \] (3-25)

Among these, \( T = t_{\text{stance}} + t_{\text{swing}} \) is the single leg period of swing state-support state.

(6) **Stability factor** \( (\delta) \) it describe the average number of the support leg during the process of walking. If assume the total number of leg is \( n \), so

\[ \delta = n \times \beta \] (3-26)

(7) **Moving velocity** \( (v) \) the moving velocity of robot is only related with the velocity of leg which is in the swing state, assume the each distance of step during the walking is \( s \), so

\[ v = \frac{s}{t_{\text{stance}}} \] (3-27)

into the formula above

\[ v = \frac{s}{t_{\text{swing}}} \left( \frac{1}{\beta} - 1 \right) = \frac{s}{t_{\text{swing}}} \left( \frac{n}{\delta} - 1 \right) \] (3-28)

Generally, during the walking process of hexapod, the time of swing time and distance of step is definite, the time of the support state is undefined, which is called covering coefficient and stability factor, from the equation, if the robot covering coefficient and stability factor is smaller (which is at a moment of support state the number of the leg is smaller), the walking velocity is more fast, however the stability factor is need over than 3, if the number of the support leg is smaller than 3, the robot is at instability state; otherwise, if the robot covering coefficient and stability factor is bigger, the walking velocity is more slower, otherwise, the robot covering coefficient and stability factor is bigger, the robot is more stable. It also said in this situation which the time of swing state and the distance of step is confirm, the robot walking velocity and stable coefficient or covering coefficient are correspondence.
Assume the distance of step is $s=100\text{mm}$, $t_{\text{swing}}=1\text{s}$, the curve relationship of hexapod velocity and stable factor $\delta$.

From the figure we can know, when the stable factor $\delta$ is 3, the robot can reach the maximum velocity; when the stable factor $\delta$ is 6, the robot velocity is 0.

Three-legged gait is the fastest gait which can satisfy the conditions, the three-legged gait is the highest efficiency gait, now analysis two kinds of three-legged gait.

Assume $t_{\text{swing}}=1\text{s}$, $t_{\text{stance}}=1\text{s}$, L1, L3, R2, when $t=0\text{s}$, is zero phase of support state, $2\pi$ phase of swing state, it is begin to support; corresponding the R1, R3, L2 is the other state, when $t=0\text{s}$, is zero phase of swing state, $2\pi$ phase of support state, it is begin to swing. Otherwise we make the initial phase reverse, is the other state of three-legged gait, the phase diagram of two kinds three-legged gait. The black line in the diagram, stand for leg in the support state, blank space stand for swing state.
a) $t=0$  

b) $t=\frac{T}{2}$  

c) $t=T$  

d) $t=\frac{3T}{2}$  

e) $t=2T$

F3-12 Three-legged gait walking figure

$O_{i0}$ is the origin of the body-base section coordinate system of NO.i leg. $O_{i3}$ is the origin of the coordinate system of No.i leg end point, $a$ is the initial attitude, $e$ is the termination attitude, $a \sim b$ is the initial gait of three-legged gait, L1, L3, R2 are support legs, support half step to backwards, L2, R1, R3 are swing legs, swing half step to forward, and then is the regular three-legged gait, $b \sim c \sim d$ is the process of the three-legs walking, $d \sim e$ is the termination attitude of the three-legged gait. L2, R1, R3 are support legs, support half step to backwards. L1, L3, R2 are swing legs, swing half step to forward.

### 3.5.2 Gait of turning on the spot

Spot turn gait need use five-legged gait, if use three-legged gait to spot turn, for example, L1, L3, R2 lift up and L2, R1, R3 rotary support, and then L2, R1, R3 lift up and L1, L3, R2 rotary support, when in the half of this process, at $t=\frac{T}{2}$ all of six legs are turned in one direction, it will make the weight shift, so we use five-legged to spot turn to make sure the hexapod can make a turn stable.
A and g are the initial attitude and termination attitude of spot turn, first according to the rotation angle $\theta$ and robot structure parameters to calculate the coordinate of end point of each joint at turning and then start spot turn. $t=0 \sim 1s$ is the process of leg L1 start swing and the endpoint fall down to the final position, other legs rotary support, the body can turn $\theta/6$; $t=1 \sim 2s$ is the process of leg L2 start swing and the endpoint fall down to the final position, the body can turn $\theta/6$ again; $t=2 \sim 3s$ is the process of leg L3 start swing and the endpoint fall down to the final position, the body can turn $\theta/6$ again; $t=3 \sim 4s$ is the process of leg R1 start swing and the endpoint fall down to the final position, the body can turn $\theta/6$ again; $t=4 \sim 5s$ is the process of leg
R2 start swing and the endpoint fall down to the final position, the body can turn $\theta/6$ again; t=5~6s is the process of leg R3 start swing and the endpoint fall down to the final position, the body can turn $\theta/6$ again. After 6s the body complete counterclockwise turn by $\theta$, the every leg has same position as t=0s.

Point turning gait

This simulation of the point turning gait is designed on the basis of the five-gait, first of all is determine the center of rotation (here is the geometric center of the robot torso), and the angle the robot need to rotation (here is simulation positioning 15 degree). Calculate the coordinate of the six feet-end in the rotation center coordinates, then calculate the coordinate of the six feet-end which is after point turning in the rotation center coordinates. With reference to method of the foot end trajectory interpolation to polynomial interpolation. Inverse solution for joint angle is show in the figure.

![Graphs showing joint rotation angles for L1, R1, L2, R2](images)
Observe the simulation results showed that six-legged robot gait planning of turning on the spot can well complete the task of selecting the desired situation.

Extraction leg L1 three joints angular velocity and angular acceleration shows in the figure. The figure shows the angular velocity curve is continuous and smooth, Angular acceleration curve is continuous, this ensures that the robot torso and leg smoothness during movement, can effectively reduce the impact force or torque.

3.5.3 Trajectory planning base on the typical gait

Trajectory planning is planning the work space of leg endpoint include leg endpoint trajectory planning at swing phase and support phase, at swing phase need to choose appropriate maximum lift height, it not only can meet the requirement of cross the obstacle but also can reduce energy consumption, at support phase the leg need bonded ground and make the body steady move, during this process it still
satisfy the velocity and accelerate characteristic to reduce impact. It can be summarized as: trajectory is continuous point which stratify the characteristic of velocity, position, accelerate at workspace. polynomial interpolation to show:

\[ \bar{p}(t) = \sum_{i=0}^{n} \bar{c}_i t^i \]  
\[ (3-29) \]

In this equation, \( \bar{p}(t) = [x(t)y(t)z(t)]^T \) shows the end point three direction position component changes over time, the orbit of end point coordination changes over time, \( \bar{c}_i = [c_{ix} c_{iy} c_{iz}]^T \) polynomial parametric.

In this equation, from \( c_0 \) to \( c_n \) has \( n+1 \) unknown quantity, so need position which determined by time, velocity and accelerate or some know condition like this can solve the orbit of end point by this polynomial. This hexapod robot has same shade and size of each leg, so it also has same method of foot end trajectory planning, the only different is the angle between each leg and body. Thus it has some parameter error at the process in inverse solution joint angles. However, it need different demand conditions of each leg during swing phase and support phase at the walking process of hexapod robot, so it has slightly different method of trajectory planning. In this formula, \( \bar{p}(t) = [x(t)y(t)z(t)]^T \) indicate the coordinate a series of points which is change by the robot foot endpoint work in the space. \( \bar{c}_i = [c_{ix} c_{iy} c_{iz}]^T \) is polynomial coefficient vector of \( t \). Because this six legs has same mechanical structure and geometry, so we can trajectory planning for any leg, assume the swing leg complete the process of swing use time \( t_2 \), go through the highest point of the track use time \( t_1 \), we can get seven restrictions.

Starting position: \( \bar{p}_0 = \bar{p}(0) = [x_0, y_0, z_0]^T \)  

End position: \( \bar{p}_2 = \bar{p}(t_2) = \sum_{i=0}^{n} \bar{c}_i t_2^i \)  

Intermediate position: \( \bar{p}_1 = \bar{p}(t_1) = \sum_{i=0}^{n} \bar{c}_i t_1^i \)  

\[ (3-30) \]
\[ (3-31) \]
\[ (3-32) \]
Starting speed: \( \ddot{V}_0 = [\dddot{p}(0)]' \) \hspace{1cm} (3-33)

Terminal speed: \( \ddot{V}_2 = [\dddot{p}(t_2)]' \) \hspace{1cm} (3-34)

Starting accelerate: \( \dddot{a}_0 = [\dddot{p}(0)]'' \) \hspace{1cm} (3-35)

Terminal accelerate: \( \dddot{a}_2 = [\dddot{p}(t_2)]'' \) \hspace{1cm} (3-36)

By using the above seven restrictions, we can get uniquely determined six polynomial about \( t \), this six polynomial constitute the curve is swinging foot trajectory.

Substituting above conditions into \( \dddot{p}(t) = \sum_{i=0}^n \dddot{c}_i t^i \). We can obtain the following equations.

\[
\begin{align*}
\dddot{p}_0 &= \dddot{c}_0 \\
\dddot{v}_0 &= \dddot{c}_1 \\
\dddot{a}_0 &= 2\dddot{c}_2 \\
\dddot{p}_1 &= \dddot{c}_0 + \dddot{c}_1 t_1 + \dddot{c}_2 t_1^2 + \dddot{c}_3 t_1^3 + \dddot{c}_4 t_1^4 + \dddot{c}_5 t_1^5 + \dddot{c}_6 t_1^6 \\
\dddot{p}_2 &= \dddot{c}_0 + \dddot{c}_1 t_2 + \dddot{c}_2 t_2^2 + \dddot{c}_3 t_2^3 + \dddot{c}_4 t_2^4 + \dddot{c}_5 t_2^5 + \dddot{c}_6 t_2^6 \\
\dddot{v}_2 &= \dddot{c}_1 + 2\dddot{c}_2 t_2 + 3\dddot{c}_3 t_2^2 + 4\dddot{c}_4 t_2^3 + 5\dddot{c}_5 t_2^4 + 6\dddot{c}_6 t_2^5 \\
\dddot{a}_2 &= 2\dddot{c}_2 + 6\dddot{c}_3 t_2 + 12\dddot{c}_4 t_2^2 + 20\dddot{c}_5 t_2^3 + 30\dddot{c}_6 t_2^4 \\
\end{align*}
\]

(3-37)

Use Matlab to solve the equations:

\[
\begin{align*}
\dddot{c}_0 &= \dddot{p}_0 \\
\dddot{c}_1 &= \dddot{v}_0 \\
\dddot{c}_2 &= \frac{1}{2} \dddot{a}_0 \\
\dddot{c}_3 &= \frac{3\dddot{p}_1 - 3\dddot{p}_2 + t_2 \dddot{v}_2}{t_1 (t_1 - t_2)^2} - \frac{\dddot{p}_0 - \dddot{p}_1}{t_1^3} \\
&= -\frac{20\dddot{p}_0 - 20\dddot{p}_2 + 12t_2 \dddot{v}_0 + 8t_2 \dddot{v}_2 + 3t_2^2 \dddot{a}_0 - t_2^2 \dddot{a}_2}{2t_2^3} - \frac{\dddot{p}_1 - \dddot{p}_2}{(t_1 - t_2)^3} \\
&\quad - \frac{-t_2^2 \dddot{a}_2 + 6t_2 \dddot{v}_2 + 12\dddot{p}_1 - 12\dddot{p}_2}{2t_2^2 (t_1 - t_2)} - \frac{3\dddot{p}_0 - 3\dddot{p}_1 + t_2 \dddot{v}_0}{t_2^3 t_1} \\
&\quad - \frac{t_2^2 \dddot{a}_0 + 6t_2 \dddot{v}_0 + 12\dddot{p}_0 - 12\dddot{p}_1}{t_1 t_2^2} \\
\end{align*}
\]

(3-38)
\[ \bar{c}_4 = \frac{-1}{t_1^3(t_2^4 + t_1 t_2^3)(t_1 - t_2)(-t_2^2 + t_1 t_2)} \left[ t_2^2 \left( t_1 \bar{a}_2 \bar{v} + \frac{3t_1^6 \bar{a}_0}{2} + 15t_1^5 \bar{v}_0 \right) 
+ 12t_1^5 \bar{v}_2 \right] - 15t_1^6 \bar{p}_0 + 15t_1^6 \bar{p}_2 
+ t_2^6 \left( \frac{3t_1^2 \bar{a}_0}{2} + 3t_1 \bar{v}_0 + 3\bar{p}_0 - 3\bar{p}_1 \right) - t_2^5 \left( 3t_1^3 \bar{a}_0 - \frac{t_1^3 \bar{a}_2}{2} \right) 
- t_2^4 \left( 27t_1^5 \bar{p}_2 - 27t_1^5 \bar{p}_0 + 8t_1^6 \bar{v}_0 + 7t_1^6 \bar{v}_2 \right) 
- t_2^4(10t_1^2 \bar{v}_0 + 5t_1^3 \bar{v}_2) 
+ t_2^3 \left( 3t_1^5 \bar{a}_0 - \frac{3t_1^5 \bar{a}_0}{2} - 15t_1^3 \bar{p}_0 + 15t_1^3 \bar{p}_2 \right) \] 

(3-41)

\[ \bar{c}_5 = \frac{-1}{t_1^3 t_2(-t_2^4 + t_1 t_2^3)(t_1 - t_2)(-t_2^2 + t_1 t_2)} \left[ t_2^4 \left( \frac{3t_1^4 \bar{a}_0}{2} - 3t_1^4 \bar{a}_0 + 15t_1^3 \bar{v}_0 \right) 
+ \right] - 15t_1^6 \bar{p}_0 + 15t_1^6 \bar{p}_2 + t_2^6 \left( \frac{3t_1^2 \bar{a}_0}{2} + 3t_1 \bar{v}_0 + 3\bar{p}_0 - 3\bar{p}_1 \right) 
- t_2^5 \left( 3t_1^3 \bar{a}_0 - \frac{t_1^3 \bar{a}_2}{2} \right) - t_2^4 \left( 27t_1^5 \bar{p}_2 - 27t_1^5 \bar{p}_0 + 8t_1^6 \bar{v}_0 + 7t_1^6 \bar{v}_2 \right) 
- t_2^4(10t_1^2 \bar{v}_0 + 5t_1^3 \bar{v}_2) 
+ t_2^3 \left( 3t_1^5 \bar{a}_0 - \frac{3t_1^5 \bar{a}_0}{2} - 15t_1^3 \bar{p}_0 + 15t_1^3 \bar{p}_2 \right) \] 

(3-42)

3.5.4 Trajectory planning of swing phase

The process of the walking leg is from lift up, swinging and fall down is called swing phase, so the position of end point is a curve from rear translation position to front translation position. The first thing to determine the space trajectory of swing phase is to know the front limit position coordinate and rear limit position.
coordinate. At same time, to satisfy the stability and continuity of lift up point and fall down point, it need the velocity and accelerate of two points. Rear limit position as the time start point at swing phase. Assume swing phase time $t_{\text{swing}} = t_2$, the time from rear translation position to highest point is $t_1$, we can get three positions know amount start point, highest point, fall down point and the know amount velocity and accelerate of start point and fall down point, total has seven know amount. For height direction (Z direction) at the highest point the velocity is zero. Thus we get eight know amount. From the equation, we can only determine the six polynomial trajectory at x y direction, and seven polynomial trajectory at z direction.

3.5.5 Trajectory planning of stance phase

Stance phase is the process of end point from landing on the ground to stance the body until lift up. Thus, the trajectory planning of stance phase is to planning a curve from front limit position to rear limit position. If the trunk movement does not any special requirement, so we only need the coordinate of landing point and lift up point. At the same time, in order to meet characteristic of rise and fall, we need know the velocity and accelerate of the landing point and lift up point. We start time of stance phase at front translation position, if the stance time $t_{\text{stance}} = t_s$, it there does not has any special requirement in the middle of this process, we only need give six know conditions which is the position, velocity and accelerate of start and end points.
\[
\begin{align*}
\{ & P_0^* = P^*(0) = [x_0^*, y_0^*, z_0^*]^T v_0^* = \dot{P}^*(0), \quad \alpha_0^* = \ddot{P}^*(0) \\
& P_s^* = P^*(t_s) = [x_s^*, y_s^*, z_s^*]^T v_s^* = \dot{P}^*(t_s), \quad \alpha_0^* = \ddot{P}^*(t_s) \\
\end{align*}
\]  

(3-47)

Put this six restrictions into formula (3-47), we can uniquely determine three five polynomials at three directions. Three combined constitute the trajectory of stance phase. If the trunk movement has special requirement, we can according to require trunk trajectory, because the foot end point is stable comparatively to geodetic system, this time the trajectory of trunk compare to ground is same the trajectory of foot end point compare to trunk but only opposite direction.

### 3.6 Kinematics simulation

In order to verify the correctness of kinematics model and kinematics equations, use sinusoidal to represent the three components track of food track in trunk coordinate system, than solve the three joint angular position and angular velocity use deduced kinematics equations by using Matlab and ADAMS.

Take R2 as reference leg, R2 foot end of the three coordinate components transformation curve in the trunk coordinate system.

![Graphs showing foot end position x-t and y-t](image)

F3-16 Foot end position x-t  

F3-17 Foot end position y-t
At MATLAB using kinematic equations to solve Torso - the base section, the base section - femur, femur - tibia three joints rotation angle, show in figure 3-14 to 3-16, angular velocity show in figure 3-17 to 3-22.
From the curve in the figure shows, foot track for polynomial interpolation with a six-legged robot with good smoothness, The foot end performance good motion
characteristics through the whole process of the track.

Simulation of typical gait, we set the plane environment, for three feet gait, according to previous planning phase combine the foot end trajectory planning method to realize walk straight and it also can turning on the spot.

3.6.1 Three-legged gait walk straight

If three-legged gait steps is 50mm, the maximum lift height of the foot end is 50mm, swing phase time is 1s, support phase time is 1s, requirements for plane straight walk. According to the foot end space analysis, This step and the foot end of the maximum lift height within the range of swing.

Start and end of the velocity and acceleration are both zero, Foot end track respect torso shows in figure3-24. Combined with the foot end trajectory planning polynomial interpolation method to calculate the trajectory of the foot end, according equation 3-8, 3-9, 3-10 to solve the each joint angle, Joint angles import into the ADAMS to realize the robot walking.

F3-26 Three-legged gait of the foot end track
3.7 Chapter summary

Basic mechanical structure of hexapod establishment D-H coordinate system of a swinging leg, derived positive and inverse kinematics equations of swinging leg serial mechanism, on this basis, calculated and analyzed foot end workspace, deduced speed Jacobian matrix; Establishment a multi-support legs and torso D-H coordinate system, establishment positive and inverse kinematics equations of parallel mechanism constituted by support leg and torso. And finally by using Matlab and Adams simulation to Verify the correctness of the equations of motion.
Chapter 4

Avoidance algorithm

4.1 Description of obstacle avoidance algorithm

As one of the greatest inventions in the 20th century, robotics has experienced fifty years of development, and progress has been made by robots. At the present stage, since more advanced technology, robotics is more and more important, because the robotics can show the development level of automation & information technology and control system and integration technology. Robot technology belongs to cross-technology, it's an integrated technology containing internet technology, control theory, information sensing technology, mechanics, artificial intelligence, electronics, bionics, and so on. It's one of the widely used technologies in the world and is also a main sign of industrial automation level. One of the core sections to constitute robotics is moving robot research, one of the core sections of moving robot is avoidance and planning path. Avoidance and planning path is an essential part of robot navigation technology. At the same time, it's also a key part of robot. It shows the interactivity between robot and surroundings, it's the basic of robot successfully complete the task. Therefore, core technology of robot application is avoidance and planning path, adopt a suitable avoidance path can reduce the operation hours and also can reduce resource input. The classic description of avoidance and planning path is: in a scene with obstacles, find the route from start
point to target and through the obstacle without collision.

Path planning is belonging to portfolio optimization, the core model of path planning is portfolio optimization model, this model is a kind of nonlinear 0-1 programming model. The core algorithm of path planning is shortest path algorithm. There are a lot of algorithms to calculate the shortest route, such as Dijkstra algorithm, Floyd-Warshall algorithm, the algorithm based on heuristics search called A*(A-star) algorithm, the algorithm based on dynamic programming called D* algorithm, neural network algorithm and so on. The research purpose of this chapter is to find a shortest route from start point to target and avoid the obstacle in the plane scene with obstacles. Among them, any obstacle area, position and shape can be calculated and all the obstacles keep static, this scene is called static scene.

4.2 Several obstacle avoidance algorithms introduced

For the static scene which environment information is completely know, we already have a lot of methods to planning path

4.2.1 Visual graph method.

At 1968, Nilsson propose this method. This method regard robot as a single point, connected other point, like target point, robot and polygon obstacle, request every point cannot pass though the obstacle, it also means the straight line must be continuously. Therefore, the problem of optimal path transform to the shortest distance between start point and target point through this viewpoint. This method is easy to find the shortest route, however we regard both robot and obstacle as point it may create an issue that the distance between robot and obstacle is too small. From this, the method Tangent Graph and Voronoi Diagrams are improve the visual graph method. Voronoi Diagrams method use tangent to represent the arc, it solve the problem in V-graph method that the obstacle cannot be arc. Tangent graph method
prevents the robot to touch the obstacle and at the same time stretch the path.

4.2.2 Free space method.

This method by using definitions all the basic form to create free space, also use a connected graph to represent free space, by searching connected graph to expand the path planning. This method is very flexible, the position of start point and target cannot lead to re-establish the connected graph, however the number of obstacle is increase by improve the complex degree of algorithm and not in all case can find best path. Propose the algorithm base on free space to against respectively for a single target point and multiple intermediate points, by using Weighted graph to represent free space, it can effective solve the single target path planning, however for the multiple intermediate points the computational of iterations and searching is very large.

4.2.3 Grid method.

At 1968, W.E.Howden propose this method. Divided the robot working environment to multiple element mesh with binary information, the core idea is recording the grid as unit of environment information, the environment divided to many grids and also has high resolution and then by using optimization to complete path planning. In this method, the size of the grid can direct impact effect: if the size of grid divided too large, planning time will reduced, the environmental resolution will reduce too, cannot immediately determine the route in the complex environment; if the size of grid divided too small, the environmental resolution will increase, thus it will determine the route immediately, however the planning time will increase and environment information storage will increase.
4.2.4 Topological method

Topological method make the spatial planning divided to many subspace which has topological properties, created topologic network based on connectivity. Find the topologic route from start point to target at network, finally calculate the geometric path by using topologic path. The basic idea of topological method is Dimensionality reduction, which is make the problem of find path from high dimensional geometry space transform to discrimination continuity in low-dimensional topologic space, the advantage is greatly reduce search space by using topologic characteristics, the algorithm complexity is only depends on the number of obstacle and topologic is always didn't need the exact location, it also has better robustness of location error. The disadvantage is the process of created topologic network is very complex, especially increase the number of obstacle. However still has problem of correct the exist topologic network and improve graphics processing speed.

4.3 Convex hull algorithm description

Problem in this chapter is in the plane static scene find the best route avoid the obstacle from the start point to target in the viewable area. To solve this problem in this chapter I use the Visual graph method.

Mathematical methods of robot’s path planning and avoidance in the static scene. Created medal of visual graph method

Node definition: We called the obstacle vertex at convex polygon in a static scene, and robots start point and target point as node at static scene. The shortest route is a broken line between start point and target point, except start point and target point, each endpoint of segment is obstacle vertex, thus the shortest route it only has relationship with polygon boundary.

Viewpoint definition: Connected two arbitrarily node points with segment in a
static scene, if this segments didn't has intersect with any obstacle we call it "visible". If the obstacle is round, so we regard the arbitrarily point has the same side with the round boundary as viewpoint.

Against the specific situation of the avoidance, correct the definition of node and viewpoint. The definition of expansion node is the arbitrary position at the environment space towards to the convex polygon vertex which correspond tangent cut point, robot start point and target point. Use segment to connected arbitrary two expansion nodes in a static scene. If this segments disjoint with the obstacle so considered this segment is visible, this time the node which correspond the visible segment is called viewpoint.

Make extended processing of all obstacles in the static scene, we distinguish the "outside viewpoint" and "inside viewpoint" depend on the relationship of position. If the viewpoint from any position in environment space make two tangent to obstacle and get cut point is called outside viewpoint, at one position and the inside obstacle at most has two outside viewpoints. If there still has other viewpoint against this obstacle it is belong inside point, inside viewpoint is invalid viewpoint. if the obstacle is round obstacle, so the outside viewpoint is cut point.

Therefore, the static scene after extended processing about the obstacle, the outside viewpoint include the cut point correspond the obstacle vertex of convex polygon, the cut point of round obstacle, the start point and the target point of robot called effective viewpoint in the static scene.

Analyzing conditions of effective viewpoint:

Two arbitrary viewpoints connected form a segment and disjoint with the boundary of extended obstacle; the condition of arbitrary viewpoint connected form a
segment and disjoint with the extended obstacle boundary is transform to the distance
between visible segment and all obstacle boundary must more than a unit, the
mathematical model.

Assume there are two segments AB and CD in static scene, the coordinate of
point A is \((x_{i2}, y_{i2})\), the coordinate of point B is \((x_{j2}, y_{j2})\), the coordinate of point C is
\((x_p, y_p)\), the coordinate of point D is \((x_q, y_q)\), A and B are arbitrary two expansion
node, scilicet it is the coordinate of the start point and target point of robot, segment
AB is a visible line, CD is a boundary of the obstacle.

The shortest route model of single target.

There are a lot of research of robot avoidance and path planning, at static scene,
the robot avoidance and path planning are based on a know shape of obstacles. There
is the mathematic model below.

4.3.1 Single obstacle

Assume there are two points A and B in the plane scene, there are the start point
and target point. The obstacle projection area is \(S\), \(S\) is a convex polygon, \(V(S)\) is
vertex collection of area \(S\), \(\Omega(S)\) is vertex collection of convex polygon \(S\) except edge
point \(V(S)\).

Assume convex polygon \(S\) and the a point A out of \(S\), take a arbitrary point
\(P\in V(S)\), use \(V(L(AP))\) to represent the vertex of line \(L(AP)\) which is connected by \(AP\).
If \(V(L(AP))\cap \Omega(S)=\emptyset\) and \(|AP|\) is the shortest distance of \(AP\), so we called vertex \(P\)
is viewpoint on the convex polygon \(S\), record \(P\).

Assume two point \(P_1, P_2\) are the point on convex polygon \(S\) and boundary of \(S\),
use \(S_1(P_1P_2)\) and \(S_2(P_1P_2)\) represented the distance from \(P_1\) to \(P_2\) by different
boundary. If \(S_1(P_1P_2)<S_2(P_1P_2)\), so called \(S_1(P_1P_2)\) is the shortest distance between
boundary point \(P_1\) and \(P_2\) of convex polygon, record \(S(P_1P_2)\).
Assume two points A and B which are located on the convex polygon S and outside S, exist two viewpoints \( P_A \) and \( P_B \), if the shortest route from point A to point B must though convex polygon S, so the shortest route from point A and point B is \( L(AP_A)+S(P_A P_B)+L(P_B P) \).

Certificate: if above proposition is wrong, thus there exist a point Q, make the shortest route from point A to point B through Q.

\[ \text{a)} \]

\[ \text{b)} \]
F4-1 Polygon obstacle

Point Q(x, y) is above the convex polygon S, point H is the intersection of extension segment AP_A and segment BP_B, broken line AQ+QB didn't intersect with convex polygon, the length is:

\[ AQ+QB = \sqrt{x^2 + y^2} + \sqrt{(x_h - x)^2 + y^2} \]

AQ+QB reduce with y reducing, when \( y \rightarrow y_1 \), Q→H, make AQ, QB and convex polygon tangent, cut point is \( P_A \) and \( P_B \). Absolutely, AQ+QB is the shortest route. As we know, \( \angle P_AOG \) is less than 90°, thus the arc \( P_AG \) on the convex polygon is less than arc \( P_AH \); similarly the arc \( P_BG \) on the convex polygon is less than arc \( P_BH \). Thus \( L(AP_A) + S(P_AP_B) + L(P_BP) < AG + GB \), so the \( L(AP_A) + S(P_AP_B) + L(P_BP) \) is the shortest broken line.

Consider a rout does not pass through any obstacle, assume it respectively intersect at point \( P_A' \) and \( P_B' \) with extension cord \( OP_A \) and \( OP_B \), the route length between A and P is \( \overline{AP_A'} \), absolutely \( \overline{AP_A'} > \overline{AP_A} \), and \( AP_A \perp OP_A \), so \( \overline{AP_A'} > \overline{AP_A} \), thereby, \( \overline{AP_A'} > \overline{AP_A} \). Similarly available \( \overline{BP_B'} > \overline{AP_A} \).
Compare the length of $\overline{P_A'P_B'}$ and $\overline{P_AP_B}$, cause the new route $A'P_B'B$ isn’t though the obstacle, so $OP_A' > OP_A$, $OP_B' > OP_B$, thus $\overline{P_A'P_B'} > \overline{P_AP_B}$. That is $A'P_B'B$ is the shortest route than others.

4.3.2 The model of multi obstacles

From the figure, assume A and B are start point and target, when star planning path, set up the result queue is empty, make the point A to current starting point $A'$, make the target B to current target $B'$, first connect the line $A'B'$ between current starting point $A'$ to current target $B'$, if this line intersect the obstacle, find the intersection which has the shortest distance with current start point $A'$, the point P in figure. Calculate the point Q which is from current start point $A'$ and bypass this obstacle, at this time put target point $B'$ into stack. Now regard point Q as current target $B'$, connect the line from current start point $A'$ to current target $B'$; if not
intersection exist, put current start point into queue, regard current target B' as new current start point, take off the target from stack as new current target Repeat the process, until the target point is empty. Thus the solid black line in figure is shortest route in this case.

Algorithm flowchart:

F 4-4 Convex hull algorithm flowchart

4.4 Convex hull algorithm process

From the figure we can get the camera visible area, there are two boundary
projections in the visible area, called polygon barriers. In order to facilitate description, definition a point x which is outside the polygon and a point p which is on the polygon boundary. Define a point x out of polygon and a point P on the polygon connect point x and p, if line xp has one intersection with polygon, so we called the point p is edge point. We assume the robot from left corner to bottom right corner, celled this two points start point and target point. Before the robot movement, first store the start point into start point variable and store the target into target variable.

Receive the viewable area from the webcam and created the coordination. Assume the start point at left corner, the target is at bottom right corner and set up the coordinate of start point and target. If there is no obstacle in the viewable area, the shortest route is the line from start point to target. Calculate if there any obstacle between the start point and target, if has obstacle on the route, we assume the return value is 1, if there no obstacle, we assume the return value is 0. If there has obstacle in the viewable area so the shortest route is invalid however this route intersect the five-point star and get two intersection points, show like below F5-5.

By using webcam, we can get a viewable area called effective active area. First assume:

1. Moving robot is active at the two dimension viewable area.
2. There are already know limited numbers of static state convex polygon obstacle in the moving robot workspace.
3. Under ideal conditions, we use a point to representation the robot so called point robot.

From the webcam we can get a viewable area like the image below and assume the five star and rectangular are obstacle. It is shown in F5-5.
F4-5 Polygon obstacle

1. Connect the point x and point p, calculate if the line xp between the start point and target has intersection with obstacle polygon. If has intersection, recording the two intersections which near the start point, record x and y.

2. Than calculate the edge point on the polygon which is near the two intersections x and y, show in figure P.

3. Connect current start point and point p, if this line doesn’t has intersection with obstacle polygon, than put the current start point into date queue, make the point p as current start point.

4. Connect current start point and target, judgment if this line has intersection between current start point and obstacle polygon, shows in the figure at rectangular, than calculate the edge point P’ on the current polygon, show in the figure --- edge point P’.

5. The line from current start point p and point P’ has intersection with obstacle polygon, otherwise, put current target into stack, make current P’ as current target.
6. Connect current start point p and current target P', this line PP' has intersection with obstacle polygon, calculate the edge point P'' on the obstacle polygon.

7. Connect current start point P and edge point P'', has no intersection with obstacle polygon, so put the start point p into queue, make P'' as current start point, connect current start point P'' and current target P', the line P''P' has no intersection with obstacle polygon, so put current start point into queue.

8. Make current target P' as current start point, make the stack top element as current target. Connect current start point and current target, if the line doesn't has intersection with obstacle polygon, put the current start point into the queue, make current target as current start point, put the top element out of stack, this time the stack is empty. Put current start point into queue, the line connection by point in the queue is the trajectory of the robot. Algorithm is complete.
4.5 Chapter summary

In this chapter, we discuss about several currently using avoidance algorithm, such as visual graph method, free space method, grid method and convex hull algorithm. We analyze the advantages and disadvantages of various algorithms, use convex hull algorithm in OPENCV environment to planning route and get the shortest route.
Chapter 5

Robot movement control

Six-legged insects, animals showed a strong ability to adapt to walking, stepped crossing different obstacle, now the robot is also an urgent need to have the capacity to adapt to changing conditions and counteract interference. So these basic principles of animal behavior can guide the development of hexapod [29.30]. Animal control six-legged insect walking system does not depend on the exact calculation of the interference obstacles [31.32]. This chapter will establish a control structure of hexapod base on servo motor control, the robot can autonomously stable walking in a complex and uncertain environment.

5.1 Servo motor control

Servo motor is also called executive motors, in the automatic control system as actuator, converted the received electrical signal to angular displacement or angular velocity of motor shaft. Servo motor can control the velocity and position very accurate. This robot use continuous rotation servo motors.
5.1.1 Servo motor zero modulation

Continuous rotation servo motors is different with general motors, continuous rotation servo motors has three external lines, orange line is signal line, red line is power positive electrode the brown line is ground line.

General motor has two external lines, one is power line the other one is ground line. Dashboard by using signal line transmission PWM signal, transmission PWM signal to control spinning speed of continuous rotation servo motors, the PWM there is pulse width modulation, in PWM signal as control system has widely used in technical field of power electronics. The control technology of PWM has advantage like simple control, flexible and dynamic response.

From figure 5.3 shows, high level duration is 1.5ms, repeat pulse sequence of PWM is low level duration 20ms. Use this pulse sequence to control servo motor
which after zero calibration, the servo motor can't rotation. If this time the motor rotation, thus means the motor requires calibration and servo motor zeroing.

From figure 4.4 shows, high level duration is 1.3ms, repeat pulse sequence of PWM is low level duration 20ms. This pulse sequence is full clockwise rotation of servo motor control pulse series. And the high level duration from 1.3ms to 1.5ms, clockwise rotation of servo motor speed descending order.

From figure 5.5 shows, high level duration is 1.7m/s, repeat pulse sequence of PWM is low level duration 20ms. This pulse sequence is full counterclockwise rotation of servo motor control pulse series. And the high level duration from 1.5ms to 1.7ms, counterclockwise rotation of servo motor speed descending order.
Servo motor has only three lines, we connect the servo motor 3 line (orange line, red line, brown line). The connect method between the servo motor and Arduino dashboard is: connect the brown line to Arduino dashboard power ground pins (GND); connect the red line to Arduino dashboard power supply pins (VCC); connect orange line to Arduino dashboard signal pins (S). Assume there are two motors, connect one orange line to Arduino dashboard #3 pin, the other connect #4 pin.

(2) Servo motor zeroing procedures

```cpp
void setup()
{
  // Assume 3 pins is outputpins
  pinMode(3, output);
}

void loop()
{
  //-------- right motor zero modulation to control pulse -----------
  // Servo motor zeroing control pulse.
```

F 5-6 connect Arduino dashboard and servo motor
// Set 3 pins is high level
digitalWrite(3,HIGH);
// Continued high level 1500us
delayMicroseconds(1500);
// Set 3 pins is low level
digitalWrite(3,LOW);
//--------left motor zero modulation to control pulse--------
// Set #3 pins is high level
digitalWrite(4,HIGH);
// Continued high level 1500us
delayMicroseconds(1500);
// Set #4 pins is low level
digitalWrite(4,LOW);
// Continued low level 20ms
delay(20);
}

Servo motor zeroing procedures used a new function, function
delayMicroseconds(x1). Function delayMicroseconds(x1) and delay(x2) are all delay
function, the different between this two function is delay(x2) is millisecond delay
function, unit is ms; and delayMicroseconds(x1) is microsecond delay function, unit
is us (1000us=1ms). Function delayMicroseconds(x1) is none return value function,
input variable x1 is unsigned int. In the program, function delayMicroseconds(x1)
effect is make pulse signal keep high level 1500us. Figure 4.6 shows the servomotor
zeroing procedure workflow.
First write the servomotor zero modulation program into Arduino Programming environment edit area. Then click verify to compiling and checking the program. If the write in program doesn't has omissions, the program is adopted, finally connect power of the Arduino dashboard. Use USB line to connect dashboard and computer click upload to download the program which is after compile into dashboard. After download check the servomotor is static or not, if the servo motor didn't static, thus need zero modulation.
5.1.2 Servo motor rotation

Servo motor has same circuit connection method with motor zero modulation. One motor orange line connected to Arduino dashboard #3 pin, and the other one orange line connected to #4 pin. Red line and brown line connected to power and ground. Write down the clockwise rotation codes into edit area of Arduino editing environment.

```c
void setup()
{
    // Set #3, #4 pins is output pins
    pinMode(3, OUTPUT);
    pinMode(4, OUTPUT);
}
```
pinMode(4,OUTPUT);
}
void loop()
{
	//--------Right motor clockwise rotation control pulse--------
	// Set #3 pin is high level pin
	digitalWrite(3,HIGH);
	// Continue high level 1700us
	delayMicroseconds(1700);
	// Set #3pin is low level pin
	digitalWrite(3,LOW);

	//--------left motor clockwise rotation control pulse--------
	// Set #4 pin is high level pin
	digitalWrite(4,HIGH);
	// Continue high level 1700us
	delayMicroseconds(1700);
	// Set #4 pin is high low pin
	digitalWrite(4,LOW);
	// Continue low level 20ms
	delay(20);
}

First write the servomotor zero modulation program into Arduino Programming environment edit area. Then click verify to compiling and checking the program. If the write in program doesn't has omissions, the program is adopted, finally connect power of the Arduino dashboard. Use USB line to connect dashboard and computer click upload to download the program which is after compile into dashboard. When the high level of pulse time continue 1700us, this pulse is full clockwise rotation pulse.

5.2. Robot gait

The basic gait of six leg robot is Tripod gait, this gait has very easy method three-three alternative
5.2.1 Forward movement:

First step: #1 lift up, #3 lift up, #5 lift up;

Second step: #1 forward, #3 forward, #5 forward;

F5-10 Servo angle of first step
F5-11 Servo angle of second step

Third step: #1 landing, #3 landing, #5 landing;

F 5-12 Servo angle of third step

Forth step: #2 lift up, #4 lift up, #6 lift up;
F 5-13 Servo angle of forth step

Fifth step: #2 forward, #4 forward, #6 forward; #1 homing, #3 homing, #5 homing (S1, S9, S28 adjusted to 1500).

F 5-14 Servo angle of fifth step

Sixth step: #2 landing, #4 landing, #6 landing;

F 5-15 Servo angle of sixth step

Seven step: #1 forward, #3 forward, #5 forward; #2 homing, #4 homing, #6 homing (S5, S24, S32 adjusted to 1500)
5.2.2 Back movement:

First step: #1 lift up, #3 lift up, #5 lift up;
Second step: #1 go back, #3 go back, #5 go back;
Third step: #1 landing, #3 landing, #5 landing;
Forth step: #2 lift up, #4 lift up, #6 lift up;
Fifth step: #2 go back, #4 go back, #6 go back; #1 homing, #3 homing, #5 homing (S1, S9, S28 adjusted to 1500).
Sixth step: #2 landing, #4 landing, #6 landing;
Seventh step: #1 forward, #3 forward, #5 forward; #2 homing, #4 homing, #6 homing (S5, S24, S32 adjusted to 1500).

5.2.3 Turn left

First step: #1 lift up, #3 lift up, #5 lift up;
Second step: #1 forward, #3 forward, #5 go back;
Third step: #1 landing, #3 landing, #5 landing;
Forth step: #2 lift up, #4 lift up, #6 lift up;
Fifth step: #2 forward, #4 go back, #6 go back; #1 homing, #3 homing, #5 homing (S1, S9, S28 adjusted to 1500).
Sixth step: #2 landing, #4 landing, #6 landing;

Seven step: #2 homing, #4 homing, #6 homing (S5, S24, S32 adjusted to 1500)

5.2.4 Turn right

First step: #1 lift up, #3 lift up, #5 lift up;

Second step: #1 go back, #3 go back, #5 forward;

Third step: #1 landing, #3 landing, #5 landing;

Forth step: #2 lift up, #4 lift up, #6 lift up

Fifth step: #2 go back, #4 forward, #6 forward; #1 homing, #3 homing, #5 homing (S1, S9, S28 adjusted to 1500).

Sixth step: #2 landing, #4 landing, #6 landing;

Seven step: #2 homing, #4 homing, #6 homing (S5, S24, S32 adjusted to 1500)

5.3 Controller

To design the motion controller of robot base on use microprocessor AT89S52 and servo TR213, the control signal of servo is pulse width modulated signal (pwm) with cycle 20ms, pulse width from 0.5-2.5ms, corresponding the helm position from 0° -180°, changes linearly. During the control signal provide certain pulse, the output shaft maintained at the appropriate angle, there is a reference circuit internal the servo, generating a reference signal with cycle 20ms width 1.5ms, by using Internal comparator, determine the direction and magnitude, generating the rotation signal.

The initial servo angle is 0° position, to make to leg move forward and back, need to go through initialization to make all servos zeroing. There are 12 servos to control the robot move, the servos which number is even make leg shaft vertical movement it control the leg lift up and lift down, the servos with odd number make the leg shaft horizontal movement, if control the leg move forward and back. By using the port P0
and P1 in AT89S52 to this 12 servos, to achieve the basic gait. The servos of the robot leg and the relationship with the port P0 and P1 shows in the chart.

<table>
<thead>
<tr>
<th></th>
<th>Servo1, Servo2</th>
<th>P0.0, P0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2</td>
<td>Servo3, Servo4</td>
<td>P0.2, P0.3</td>
</tr>
<tr>
<td>A3</td>
<td>Servo5, Servo6</td>
<td>P0.4, P0.5</td>
</tr>
<tr>
<td>B1</td>
<td>Servo7, Servo8</td>
<td>P1.0, P1.1</td>
</tr>
<tr>
<td>A2</td>
<td>Servo9, Servo10</td>
<td>P1.2, P1.3</td>
</tr>
<tr>
<td>B3</td>
<td>Servo11, Servo12</td>
<td>P1.4, P1.5</td>
</tr>
</tbody>
</table>

Chart 5.1 The servos of the robot leg and the relationship with the port P0 and P1

5.3.1 The multiple servos sharing control

The control signal for single servo show in figure, the cycle T is 20ms, the change time is t and the variation range is from 0.5-2.5ms.

![F5-17 single servo control signal](image)

The control signal for single servo has 17.5ms maintain low level. In 20ms, first set the P0.0 in high level, inverse it after time t, the servo shaft will rotation to the corresponding height of time t. The maximum value of t is 2.5ms. After 2.5ms set the P0.0 to high level will not affect P0.0 control the servo, so can control up to 8 signals within 20ms(20ms/2.5ms=8), use multi-servo-sharing control to make 12 servos coordinated Control, the twelve servos divided into two groups, timer 0 control servos from 1-6, timer 1 control servos from 7-12. Six-way servo control pulse shows in figure.
F5-18 six-way servos control signal

Each timer has 12 interrupts within one cycle (6 servos has 12 interrupts). Timer
counts do not take up CPU time, can work simultaneously, Ignore the interrupt time
of two timers, so can parallel control servo. In one cycle (20ms) can control 12 servos,
12 servos control the usage of CPU time is the time for perform 24 interrupt program.
The relationship between the Initial timer ,pulse width and servo rotation angle
shows in chart.

<table>
<thead>
<tr>
<th>Timer initial</th>
<th>1.300</th>
<th>1.400</th>
<th>1.450</th>
<th>1.550</th>
<th>1.620</th>
<th>1.630</th>
<th>1.800</th>
<th>1.820</th>
<th>1.830</th>
<th>2.000</th>
<th>2.540</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse width</td>
<td>1.154</td>
<td>1.262</td>
<td>1.316</td>
<td>1.425</td>
<td>1.500</td>
<td>1.532</td>
<td>1.696</td>
<td>1.718</td>
<td>1.728</td>
<td>1.906</td>
<td>2.500</td>
</tr>
<tr>
<td>Rotation angle</td>
<td>58.86</td>
<td>68.58</td>
<td>73.44</td>
<td>83.25</td>
<td>90</td>
<td>92.88</td>
<td>107.66</td>
<td>109.6</td>
<td>110.5</td>
<td>126.4</td>
<td>180</td>
</tr>
</tbody>
</table>

chart5-2 The relationship between the Initial timer ,pulse width and servo rotation angle

5.3.2 Initialization module design

Initialization module is designed mainly to complete the initial setup procedure,
mainly include timer initialization, servo position initialization. Timer initialization is
set up timer initial value and set the interrupt priority. Set the servo initialize position
for all motor positions of each joint servo is 90° , make the robot on standby station.
Initialization program is as follows:
TMOD = 0x11;

flag1 = flag2 = 1;

TH0 = TH1 = - (1620 / 256); // timing 1.5ms

TL0 = TL1 = - (1620 / 256);

EA = 1;

ET0 = 1; TR0 = 1; PT0 = 1;

ET1 = 1; TR1 = 1; PT1 = 1;

_set_time(*1,2*/11620,/*3,4*/1620,1620,/*5,6*/1620,1620,/*7,8*/1620,1620,/*9,10*/1620,1620/*11,12*/1620,1620/

5.3.3 Time setting function

Time setting function _set_time( ) use fir setting the initial value of timer.

_set_time( unit x1, unit x2, unit x3, unit x4, unit x5, unit x6, unit x7, unit x8, unit x9, unit x10, unit x11, unit x12)

  t1 = x1, t2 = x2, t3 = x3, t4 = x4, t5 = x5, t6 = x6, t7 = x7, t8 = x8, t9 = x9, t10 = x10, t11 = x11, t12 = x12;

  When response the timer interrupt service program, the initial value assigned to the timer according to program flow, to achieve the servo rotation angle control.

5.3.4 Timer interrupt service routine

Timer 0 and Timer 1 and Timer 1 operation mode, use the flag 1, flag 2 this two flags. Timer 0 uses the flag1, timer1 uses the flag2. First judgment flag after entering the interrupt flag, and then entering the execute program which is corresponding flag.
(1) Timer0 interrupts service function void timer0 (void) interrupt1:

void timer0 (void) interrupt 1
{
    switch (flag1)
    {
        case 1: { p00=1; TH0=-(t1/256); TL0=-(t1%256); break; } // set high level to first entering interrupt p00 port, make the timer to a new initial value, timing the servo which connect the port p00 rotation to target angle.
            
        case 2: { p00=0; TH0=-(2540-t1/256); TL0=-(2540-t1)/256; break; } // second entering interrupt, set p00 to low level, timing is 2.5ms-(p00 high level holding time)
            
        case 3: { p01=1; TH0=-(t2/256); TL0=-(t2%256); break; }
        case 4: { p01=0; TH0=-(2540-t2/256); TL0=-(2540-t2)%256; break; }
        case 5: { p02=1; TH0=-(t3/256); TL0=-(t3%256); break; }
        case 6: { p02=0; TH0=-(2540-t3/256); TL0=-(2540-t3)%256; break; }
        case 7: { p03=1; TH0=-(t4/256); TL0=-(t4%256); break; }
        case 8: { p03=0; TH0=-(2540-t4/256); TL0=-(2540-t4)%256; break; }
        case 9: { p04=1; TH0=-(t5/256); TL0=-(t5%256); break; }
        case 10: { p04=0; TH0=-(2540-t5/256); TL0=-(2540-t5)%256; break; }
        case 11: { p05=1; TH0=-(t6/256); TL0=-(t6%256); break; }
        case 12: { p05=0; TH0=-(8150-t6/256); TL0=-(8150-t6)%256; flag=0; break; } //
            
            default; flag1=0;
        }
    }

    flag1++
}

(2) Timer1 interrupts service function void timer1 (void) interrupt3

has same program structure with timer0 interrupts service function void timer0 (void)
interrupt1.

5.4 Experimental results and analysis

Robots with axial symmetrical shape are mainly composed of two parts, body and legs, the distance between two feet is 311mm, height is 135mm, length is 310mm, weight is 1360g, the leg make up by using three parts base section, femur and tibia. The base section has two servos to make the leg lift up and down and move forward and back. The maximum angle of forward and back is 45°, lift up is 30°. Robot forward speed is 3.75cm/s, steps is 10.3cm, rotation angle for each step is 10.5°.

Experiments show that, according to the result of research robot gait, robot movement control by using the algorithm of motion controller, movement speed can be adjusted by changing the time interval servo controls the pulse width and each leg movement, its smooth movement.

By analyzing the gait of hexapod robot, propose the base movement of robot, accordance with the idea of sharing control multiple servos, using two timer of AT89S52 to control the 12 servos timing pulse, to achieve the basic Hexapod robot motion controller design so that the control system has been the basis of expansion modules are implemented, Intelligent robot will also be upgraded with various functional modules and continue to develop.
Chapter 6

Conclusion and future work

6.1 Conclusion

In this thesis, I mainly discuss two aspects, one is how to control servo motor to make hexapod move and the other is about how to make the robot avoid obstacles. Use Arduino dashboard to control servo motor, this robot has 18 servo motors, in order to facilitate control, I use Arduino dashboard connect 32 servo controller to control the hexapod.

After that, the robot not only can walking but also need avoid obstacles, I find a lot of thesis about obstacle avoidance algorithm, now the most popular method is Ant Colony Algorithm, Artificial Potential Field, Convex Hull Algorithm and so on, after analyzing the pros and cons, I decided use convex hull algorithm to avoid obstacles, because in real environment, most obstacle is convex polygon, use this algorithm can easily find the contour and edge points so we can get the shortest rout without obstacle.

6.2 Future works

Robotics is a very interesting field, we can use more algorithms and more sensors to perform more functions.

(1) Do more detail works, such as correction software parameters to make the robot
has better movement and operation more flexible, diverse and smooth.

(2) Depth application of the sensor module.

(3) Combined with image processing technology, build a mobile control platform.
References


planning and control [M]. Huazhong University of Science and Technology Press, 2006: 114-117.


Appendix

Appendix. A The program of food end generate track

close all; clear all; clc;

% Polynomial fit coefficients Functions
syms c0 c1 c2  c3  c4  c5 c6 t1 t2 a0 a2 v0 v2 p0 p1 p2 t;
F1 = c0+c1*t1+c2*t1^2+c3*t1^3+c4*t1^4+c5*t1^5+c6*t1^6;
F2 = c0+c1*t2+c2*t2^2+c3*t2^3+c4*t2^4+c5*t2^5+c6*t2^6;
F3=diff(F2,t2);
F4=diff(F3,t2);
f1=p0-c0; f2=v0-c1; f3=a0-2*c2; f4=p1-F1; f5=p2-F2; f6=v2-F3; f7=a2-F4;
[c0 c1 c2  c3  c4  c5 c6]=solve(f1, f2, f3, f4, f5, f6, f7, 'c0', 'c1', 'c2', 'c3', 'c4', 'c5', 'c6');
c=simplify([c0;c1;c2;c3;c4;c5;c6])

% Track the foot end of the fitting function, solving the function of velocity and acceleration.
pt=c0+c1*t+c2*t^2+c3*t^3+c4*t^4+c5*t^5+c6*t^6;
vt=c1+2*c2*t+3*c3*t^2+4*c4*t^3+5*c5*t^4+6*c6*t^5;
at=2*c2+6*c3*t+12*c4*t^2+20*c5*t^3+30*c6*t^4;
pt=subs(pt,{p0,p2,p1,v0,v2,a0,a2,t1,t2},{[20;10;-15],[20;10;-15],[23;0;-11],[0;0;0],[0;0;0],[0;0;0],[0;0;0],0.3,0.6});
vt=subs(vt,{p0,p2,p1,v0,v2,a0,a2,t1,t2},{[20;10;-15],[20;10;-15],[23;0;-11],[0;0;0],[0;0;0],[0;0;0],[0;0;0],0.3,0.6});
at=subs(at,{p0,p2,p1,v0,v2,a0,a2,t1,t2},{[20;10;-15],[20;10;-15],[23;0;-11],[0;0;0],[0;0;0],[0;0;0],[0;0;0],0.3,0.6});
digits(5);
pt=vpa(pt)
vt=vpa(vt)
at=vpa(at)
t=0:0.01:0.6;
pt=c0+c1*t+c2*t.^2+c3*t.^3+c4*t.^4+c5*t.^5+c6*t.^6;
vt=c1+2*c2*t+3*c3*t.^2+4*c4*t.^3+5*c5*t.^4+6*c6*t.^5;
at=2*c2+6*c3*t+12*c4*t.^2+20*c5*t.^3+30*c6*t.^4;
pt=subs(pt,{p0,p2,p1,v0,v2,a0,a2,t1,t2},{[20;10;-15],[20;10;-15],[23;0;-11],[0;0;0],[0;0;0],[0;0;0],[0;0;0],0.3,0.6});
vt=subs(vt,{p0,p2,p1,v0,v2,a0,a2,t1,t2},{[20;10;-15],[20;10;-15],[23;0;-11],[0;0;0],[0;0;0],[0;0;0],[0;0;0],0.3,0.6});
at=subs(at,{p0,p2,p1,v0,v2,a0,a2,t1,t2},{[20;10;-15],[20;10;-15],[23;0;-11],[0;0;0],[0;0;0],[0;0;0],[0;0;0],0.3,0.6});
figure('name'.'In the x, y, z direction of displacement, velocity and acceleration')
% x, y, z direction curve image
subplot(3,3,1)
plot(t,pt(1,:)),xlabel('time t(s)'),ylabel('x-axis displacement (cm)')
subplot(3,3,2)
plot(t,pt(2,:)),xlabel('time t(s)'),ylabel('y-axis displacement (cm)')
subplot(3,3,3)
plot(t,pt(3,:)),xlabel('time t(s)'),ylabel('z-axis displacement (cm)')
% Velocity curve image of x, y, z directions.
1. subplot(3,3,4)
plot(t,vt(1,:)),xlabel('time t(s)'),ylabel('x-axis speed (cm/s)')
subplot(3,3,5)
plot(t,vt(2,:)),xlabel('time t(s)'),ylabel('y-axis speed (cm/s)')
subplot(3,3,6)
plot(t,vt(3,:)),xlabel('time t(s)'),ylabel('z-axis speed (cm/s)')
% Acceleration curve image of x, y, z directions.
subplot(3,3,7)
plot(t,at(1,:)),xlabel('time t(s)'),ylabel('x-axis acceleration (cm/s^2)')
subplot(3,3,8)
plot(t,at(2,:)),xlabel('time t(s)'),ylabel('y-axis acceleration (cm/s^2)')
subplot(3,3,9)
plot(t,at(3,:)),xlabel('time t(s)'),ylabel('z-axis acceleration (cm/s^2)')
% Space curve path
figure('name','Space curve path')
plot3(pt(1,:),pt(2,:),pt(3,:)),grid on, xlabel('x(cm)'), ylabel('y(cm)'), zlabel('z(cm)')
% Function image of each joint angle, angular velocity and angular velocity
\[
t_1=0.005:0.01:0.595;
t_2=0.01:0.01:0.59;
\]
theta(1,:) = atan(pt(2,:)/pt(1,:));
x = pt(1,:)-cos(theta(1,:))*3.2;
y = pt(2,:)-sin(theta(1,:))*3.2;
z = pt(3,:);
theta(3,:) = -acos((x.^2+y.^2+z.^2-18^2-18.8^2)./(2*18*18.8));
theta(2,:) = -atan(18.8*sin(theta(3,:))./(18+18.8*cos(theta(3,:))))+atan(z./sqrt(x.^2+y.^2));
omega = diff(theta,1,2)/0.01;
alpha = diff(omega,1,2)/0.01;
figure('name','Each joint angle, angular velocity and angular image')
% Function image of each joint angle.
subplot(3,3,1)
plot(t,theta(1,:)),xlabel('time t(s)'),ylabel('base section rotation angle (rad)')
subplot(3,3,2)
plot(t,theta(2,:)),xlabel('time t(s)'),ylabel('femur rotation angle (rad)')
subplot(3,3,3)
plot(t,theta(3,:)),xlabel('time t(s)'),ylabel('tibia rotation angle (rad)')
% Function image of each joint angular velocity.
subplot(3,3,4)
plot(t1,omega(1,:)),xlabel('time t(s)'),ylabel('base section angular velocity (rad/s)')
subplot(3,3,5)
plot(t1,omega(2,:)),xlabel('time t(s)'),ylabel('femur angular velocity (rad/s)')
subplot(3,3,6)
plot(t1,omega(3,:)),xlabel('time t(s)'),ylabel('tibia angular velocity (rad/s)')
% Function image of each joint angular velocity.
subplot(3,3,7)
plot(t2,alpha(1,:)),xlabel('time t(s)'),ylabel('base section angular acceleration (rad/s^2)')
subplot(3,3,8)
plot(t2,alpha(2,:)),xlabel('time t(s)'),ylabel('femur angular acceleration (rad/s^2)')
subplot(3,3,9)
plot(t2,alpha(3,:)),xlabel('t(s)'),ylabel('tibia angular acceleration (rad/s^2)')

**Inverse kinematics**

clear all; clc
syms o1 o2 o3 l1 l2 l3 px py pz x y z;

% Transformation matrix
t01=DH(o1,0,0,0);
t12=DH(o2,0,l1,pi/2);
t23=DH(o3,0,l2,0);
t34=DH(0,0,l3,0);
t24=simplify(t23*t34);
t14=simplify(t12*t24);
t04=simplify(t01*t14);
%
p4=[0;0;0;1];
p0=simplify(t04*p4);
% o1, o3
eqns=[cos(o1)*(l3*cos(o2 + o3) + l2*cos(o2)), x;
      sin(o1)*(l3*cos(o2 + o3) + l2*cos(o2)), y;
      l3*sin(o2 + o3) + l2*sin(o2), z];
l=simplify(sum(eqns(:,1).^2));
r=simplify(sum(eqns(:,2).^2));
F1=l-r;
F2=atan(py/px)-o1;
[o1,o3]=solve(F1,F2,'o1','o3')

**Appendix. B Servo motor control algorithm**

```
int val;
void setup()
{
  Serial.begin(115200); // 32 servo motor controller push button to 1
}
void loop()
{
  val=Serial.read();
  if(val=='s')
  {
    Serial.println("#0P2200#1P2100#2P800#4P2200#5P2100#6P1400#12P2200#13P2100#14P1400#16P800#17P900#18P2000#20P800#21P900#22P1600#28P800#29P900#30P1600T1000"); // input 's' for stand
    delay(1000);
```
if(val=='l')
{
    Serial.println("#0P2200#1P2200#2P800#4P2200#5P2200#6P1400#12P2200#13P2200#14P1400#16P800#17P800#18P2000#20P800#22P1600#28P800#29P800#30P1600T1000");// input 'l' for squat
delay(1000);
}
if(val=='l')
{
    Serial.println("#0P800#1P2200#2P800#4P800#5P2200#6P1400#12P800#13P2200#14P1400#16P2200#17P800#18P2000#20P2200#21P800#22P1600#28P2200#29P800#30P1600T1000");// input 'F' lift all of the leg
delay(1000);
}
if(val=='0')
{
    Serial.println("#0P2200T1000");//input 0 make servo #0 move
delay(1000);
}

**Forward movement:**

First step: #1 lift up, #3lift up, #5 lift up;
Seventh step: #1 lift up, #3lift up, #5 lift up;
#2P1500#3P838#4P894#6P1500#7P1000#8P781#10P1500#11P810#12P838#13P1500#14P1852#15P2106#17P1500#18P2331#19P2106#21P1500#22P1993#23P2021T300
Second step: #1forward, #3 forward, #5 forward;
Eighth step: #1 forward, #3 forward, #5 forward; #2 homing, #4 homing, #6 homing (S5, S24, S32 adjusted to 1500).
Third step: #1 landing, #3 landing, #5 landing;
#2P1200#3P1034#4P894#6P1500#7P1000#8P781#10P1200#11P1100#12P838#13P1500#14P1852#15P2106#17P1800#18P1966#19P2106#21P1500#22P1993#23P2021T300
Forth step: #2 lift up, #4 lift up, #6 lift up;
#2P1500#3P1034#4P894#6P1500#7P781#8P781#10P1500#11P1100#12P838#13P1500#14P2219#15P2106#17P1500#18P1966#19P2106#21P1500#22P2247#23P2021T300
Fifth step: #2 forward, #4 forward, #6 forward; #1 homing, #3 homing, #5 homing (S1, S9, S28 adjusted to 1500).
#2P1500#3P1034#4P894#6P1200#7P1000#8P781#10P1500#11P1100#12P838#13P1800#14P2219#15P2106#17P1500#18P1966#19P2106#21P1800#22P2247#23P2021T300
Sixth step: #2 landing, #4 landing, #6 landing;
#2P1500#3P1034#4P894#6P1200#7P1000#8P781#10P1500#11P1100#12P838#13P1800#14P1852#15P2106#17P1500#18P1966#19P2106#21P1800#22P2021#23P2021T300
Back movement:
First step:
#2P1500#3P838#4P894#6P1500#7P1000#8P781#10P1500#11P1800#12P838#13P1500#14P1852#15P2106#17P1500#18P1966#19P2106#21P1500#22P1993#23P2021T300
Second step:
#2P1800#3P838#4P894#6P1500#7P1000#8P781#10P1500#11P1800#12P838#13P1500#14P2219#15P2106#17P1500#18P1966#19P2106#21P1500#22P1993#23P2021T300
Third step:
#2P1800#3P1034#4P894#6P1500#7P1000#8P781#10P1800#11P1100#12P838#13P1500#14P1852#15P2106#17P1200#18P1966#19P2106#21P1500#22P1993#23P2021T300
Fourth step:
#2P1500#3P1034#4P894#6P1500#7P781#8P781#10P1500#11P1100#12P838#13P1500#14P2219#15P2106#17P1500#18P1966#19P2106#21P1500#22P2247#23P2021T300
Fifth step:
#2P1500#3P1034#4P894#6P1800#7P1000#8P781#10P1500#11P1100#12P838#13P1200#14P2219#15P2106#17P1500#18P1966#19P2106#21P1200#22P2247#23P2021T300
Sixth step:
#2P1500#3P1034#4P894#6P1800#7P1000#8P781#10P1500#11P1100#12P838#13P1200#14P1852#15P2106#17P1500#18P1966#19P2106#21P1200#22P2021#23P2021T300
Turn left
First step:
Second step:

Third step:

Forth step:

Fifth step:

Sixth step:

Turn right

First step:

Second step:

Third step:

Forth step:

Fifth step:

Sixth step:
int pushButton1 = 3;
int pushButton2 = 4;
void setup()
{
  Serial.begin(115200);
pinMode(pushButton1, INPUT);
pinMode(pushButton2, INPUT);
}
void loop()
{
  int buttonState1 = digitalRead(pushButton1);
  int buttonState2 = digitalRead(pushButton2);
  Serial.print(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P1098 #11 P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1430 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P1849 T400");
  delay(400);
  Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P1098 #11 P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1430 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P1849 T1000");
  delay(2000);
  if (buttonState1 == HIGH && buttonState2 == HIGH)
  {
    Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P1098 #11 P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1430 #21 P1430 #23 P1384 #24 P1323 #25 P1500 #27 P1337 #28 P1384 #29 P1849 T350");
    delay(380);
    Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P898 #11 P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1323 #25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
    delay(380);
    Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P898 #11 P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P1849 T350");
    delay(380);
    while(buttonState1==HIGH && buttonState2==HIGH)
    {
      Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1723 #9 P1098 #11 P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1230 #21 P1630 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1184 #29 P2049 T350");
      delay(380);
      Serial.println(" #3 P1663 #4 P1593 #5 P1277 #7 P1547 #8 P1723 #9 P1098 #11 P1686 #12 P1593 #13 P1277 #19 P1407 #20 P1230 #21 P1430 #23 P1384 #24 P1523 #25 P1700 #27 P1337 #28 P1184 #29 P2049 T350");
      delay(380);
      Serial.println(" #3 P1663 #4 P1593 #5 P1277 #7 P1547 #8 P1523 #9 P1098 #11 P1686 #12 P1593 #13 P1277 #19 P1407 #20 P1430 #23 P1384 #24 P1523 #25 P1700 #27 P1337 #28 P1384 #29 P1849 T350");
      delay(380);
Serial.println(" #3 P1663 #4 P1793 #5 P1277 #7 P1547 #8 P1523 #9 P1098 #11
P1686 #12 P1793 #13 P1277 #19 P1407 #20 P1430 #21 P1430 #23 P1384 #24 P1323
#25 P1700 #27 P1337 #28 P1384 #29 P1849 T350");
delay(380);
Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P898 #11
P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1323
#25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P898 #11
P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523
#25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
}

if (buttonState1 == LOW && buttonState2 == HIGH)
{
    Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P1098 #11
P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1323
#25 P1500 #27 P1337 #28 P1384 #29 P1849 T350");
delay(380);
    Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11
P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523
#25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
    Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11
P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523
#25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
    Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P1098 #11
P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523
#25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
    Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1723 #9 P1298 #11
P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523
#25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
    Serial.println(" #3 P1663 #4 P1593 #5 P1677 #7 P1547 #8 P1523 #9 P1098 #11
P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523
#25 P1500 #27 P1337 #28 P1384 #29 P1849 T350");
delay(380);
    Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11
P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523
#25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
    while(buttonState1 == LOW && buttonState2 == HIGH)
    {
        Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P1098 #11
P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1323
#25 P1500 #27 P1337 #28 P1384 #29 P1849 T350");
delay(380);
        Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11
P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523
#25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
        Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1723 #9 P1298 #11
P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523
#25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
        Serial.println(" #3 P1663 #4 P1593 #5 P1677 #7 P1547 #8 P1723 #9 P1098 #11
P1686 #12 P1593 #13 P1677 #19 P1407 #20 P1430 #21 P1430 #23 P1384 #24 P1523
#25 P1700 #27 P1337 #28 P1384 #29 P1849 T350");
delay(380);
    }
}
Serial.println(" #3 P1663 #4 P1593 #5 P1677 #7 P1547 #8 P1523 #9 P1098 #11 P1686 #12 P1593 #13 P1677 #19 P1407 #20 P1430 #21 P1430 #23 P1384 #24 P1523 #25 P1700 #27 P1337 #28 P1384 #29 P1849 T350");
delay(380);
Serial.println(" #3 P1663 #4 P1793 #5 P1677 #7 P1547 #8 P1523 #9 P1098 #11 P1686 #12 P1793 #13 P1677 #19 P1407 #20 P1430 #21 P1430 #23 P1384 #24 P1323 #25 P1700 #27 P1337 #28 P1384 #29 P1849 T350");
delay(380);
Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11 P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1323 #25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11 P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1630 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P1849 T350");
delay(380);
}
if (buttonState1 == HIGH && buttonState2 == LOW)
{
  Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11 P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1230 #21 P1630 #23 P1384 #24 P1323 #25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
  Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11 P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1230 #21 P1430 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P1849 T350");
delay(380);
  Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11 P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1430 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P2049 T350");
delay(380);
  Serial.println(" #3 P1663 #4 P1593 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11 P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1430 #21 P1430 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P1849 T350");
delay(380);
}
while(buttonState1 == HIGH && buttonState2 == LOW)
{
  Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11 P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1230 #21 P1230 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P1649 T350");
delay(380);
  Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P1298 #11 P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1230 #21 P1230 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P1649 T350");
delay(380);
  Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P898 #11 P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1230 #21 P1230 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P1649 T350");
delay(380);
  while(buttonState1 == HIGH && buttonState2 == LOW)
  {
    Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P898 #11 P1686 #12 P1593 #13 P1477 #19 P1407 #20 P1230 #21 P1230 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1184 #29 P1549 T350");
    delay(380);
Serial.println(" #3 P1663 #4 P1593 #5 P1277 #7 P1547 #8 P1723 #9 P1098 #11 P1686 #12 P1593 #13 P1277 #19 P1407 #20 P1230 #21 P1430 #23 P1384 #24 P1523 #25 P1300 #27 P1337 #28 P1184 #29 P1849 T350");
  delay(380);
  Serial.println(" #3 P1663 #4 P1593 #5 P1277 #7 P1547 #8 P1523 #9 P1098 #11 P1686 #12 P1593 #13 P1277 #19 P1407 #20 P1430 #21 P1430 #23 P1384 #24 P1523 #25 P1300 #27 P1337 #28 P1384 #29 P1849 T350");
  delay(380);
  Serial.println(" #3 P1663 #4 P1793 #5 P1277 #7 P1547 #8 P1523 #9 P1098 #11 P1686 #12 P1793 #13 P1277 #19 P1407 #20 P1430 #21 P1230 #23 P1384 #24 P1323 #25 P1500 #27 P1337 #28 P1384 #29 P1649 T350");
  delay(380);
  Serial.println(" #3 P1663 #4 P1793 #5 P1477 #7 P1547 #8 P1523 #9 P1098 #11 P1686 #12 P1793 #13 P1477 #19 P1407 #20 P1430 #21 P1230 #23 P1384 #24 P1523 #25 P1500 #27 P1337 #28 P1384 #29 P1649 T350");
  delay(380);
}
Appendix C.  Avoidance algorithm

// OpencvTest.cpp : definition the entry point of console application program

//

#include "stdafx.h"

/* This is a standalone program. Pass an image name as a first parameter of the program.
Switch between standard and probabilistic Hough transform by changing "#if 1" to
"#if 0" and back */

#include <cv.h>
#include <highgui.h>
#include <math.h>
#include <highgui\highgui.hpp>
#include <imgproc\imgproc.hpp>
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
using namespace cv;
using namespace std;
const int PointNumMax = 1000;
//variable of image change
Mat src;
Mat dst;
Mat gray_dst;
Mat drawing;
Mat lsdst;
//variable of calculate convex hull

int thresh = 100;

int max_thresh = 255;

RNG rng(12345);

vector<vector<Point>> contours;

vector<Vec4i> hierarchy;

vector<vector<Point>> hull(PointNumMax);

//point set operation through the planned toute

CvPoint

RoutePoint[PointNumMax],EndPointStack[ PointNumMax],CandiPointStack[PointNumMax];

int RouteNum=0,EndPointNum=0,CandiPointNum=0;

//variable of path planning process

CvPoint center1,center2,CrossPoint1,CrossPoint2;

CvPoint CurrentPoint1,CurrentPoint2,Point1,Point2,KeyPoint1,KeyPoint2;

float CurrentK,CurrentB,Dist1,Dist2,Alpha1,Alpha2;

vector<Point> stack(100);

int StackTop=0;

int Lsx,Lsy,CrossFlag=0,iCurrent;

double IsCross;

CvPoint2D32f lsPoint;

float k;                //line slope between starting point to target point

int b;                   //line intercept

float Tri_a,Tri_b,Tri_c,TriA;

//Declare variables, ê?iCon Loop control variables, ê?TorF: êo0 Terminator Cycle,
Decreasing, $i$ Increment, $x$Direct: $x$-axis direction, $i$?0 $y$-axis direction

```c
int iCon, iConMax, iDirect, TorF, xDirect;

// Function header
void thresh_callback(int, void*);
```

//calculate if there ant obstacle between the start point to target, if have the return value is 1, if not the return value is 0

```c
int DetectCross(CvPoint StartPoint, CvPoint EndPoint, int *iPtr, CvPoint* Point1, CvPoint* Point2)
{
    CrossFlag=0; //if there has obstacle need bypass

    if (abs(StartPoint.x-EndPoint.x)>abs(StartPoint.y-EndPoint.y))
    {
        iCon=StartPoint.x;
        iConMax=EndPoint.x;
        
        k=(float)(EndPoint.y-StartPoint.y+0.5)/(float)(EndPoint.x-StartPoint.x+0.5); //to calculate the slope between the start point and target
        b=(int)(StartPoint.y-k*StartPoint.x);
        xDirect=1;
    }
    else
    {
        iCon=StartPoint.y;
    }
}
```
iConMax=EndPoint.y;

k=(\texttt{float})(\texttt{float})(\texttt{EndPoint}.x-\texttt{StartPoint}.x+0.5)/\texttt{EndPoint}.y-\texttt{StartPoint}.y+0.5);  //to calculate the slope between the start point and target

\[b=(\texttt{int})(\texttt{StartPoint}.x-k*\texttt{StartPoint}.y);\]

\[\texttt{xDirect}=0;\]

\textbf{if} (iCon>iConMax)
\{
\texttt{iDirect}=0;
\}

\textbf{else}
\{
\texttt{iDirect}=1;
\}

TorF=1;

//Cycle to determine whether the intersection

\textbf{while} (TorF)
\{

//******************************************************************************

\textbf{if} (\texttt{xDirect}==1)
\{

\texttt{lsPoint}.x=(\texttt{float})iCon;
\texttt{lsPoint}.y=(\texttt{float})(k*(iCon+20)+b);
\}

//******************************************************************************
else
{
    lsPoint.x=(float)(k*(iCon+20)+b);
    lsPoint.y=(float)iCon;
}
if (!CrossFlag)
{
    for (int i=0;i<contours.size();i++)
    {
        IsCross=pointPolygonTest(contours[i],lsPoint,0);
        if (IsCross>=0)
        {
            *iPtr=i;
            Point1->x=(int)lsPoint.x;
            Point1->y=(int)lsPoint.y;
            CrossFlag=1;
            i=contours.size();
        }
    }
}
else
{
    IsCross=pointPolygonTest(contours[*iPtr],lsPoint,0);
    if (IsCross<0)
    {

Point2->x=(int)lsPoint.x;
Point2->y=(int)lsPoint.y;
iCon=iConMax;
}
}
if (iDirect==0)
{
    iCon=iCon-10;
    if (iCon<iConMax)
        TorF=0;
}
else
{
    iCon=iCon+10;
    if (iCon>iConMax)
        TorF=0;
}
}
if (CrossFlag)
{
    //   EndPointNum++;
    //   EndPointStack[EndPointNum].x=EndPoint.x;
    //   EndPointStack[EndPointNum].y=EndPoint.y;
    return 1;
}
```c
else
{
    return 0;
}

void CalcEndPoint(CvPoint StartPoint, CvPoint Point1, CvPoint Point2, int contoursNo, CvPoint *EndPoint)
{

    // find the intersection point with convex hull, convex hull is i Current, ?two intersections are Point1 and Point2

    float s1, s2;
    KeyPoint1.x = Point1.x;
    KeyPoint2.x = Point1.x;
    KeyPoint1.y = Point1.y;
    KeyPoint2.y = Point1.y;
    Alpha1 = 0;
    Alpha2 = 0;

    for (int i = 0; i < contours[contoursNo].size(); i++)
    {

        s1 = (Point1.x - contours[contoursNo][i].x) * (Point1.x - contours[contoursNo][i].x);
        s2 = (Point1.y - contours[contoursNo][i].y) * (Point1.y - contours[contoursNo][i].y);
        Tri_a = sqrt(s1 + s2);
    }
}```
s1=(StartPoint.x-contours[contoursNo][i].x)*(StartPoint.x-contours[contoursNo][i].x);

s2=(StartPoint.y-contours[contoursNo][i].y)*(StartPoint.y-contours[contoursNo][i].y);

Tri_b=sqrt(s1+s2);

s1=(StartPoint.x-Point1.x)*(StartPoint.x-Point1.x);

s2=(StartPoint.y-Point1.y)*(StartPoint.y-Point1.y);

Tri_c=sqrt(s1+s2);

TriA=(Tri_b*Tri_b+Tri_c*Tri_c-Tri_a*Tri_a)/(2*Tri_b*Tri_c);

TriA=acos(abs(TriA));

if(contours[contoursNo][i].y>(k*contours[contoursNo][i].x+b)) //fixed on the top
{
    if (TriA>Alpha1)
    {
        Alpha1=TriA;
        KeyPoint1.x=contours[contoursNo][i].x;
        KeyPoint1.y=contours[contoursNo][i].y;
    }
}
else//belowe
{
    if (TriA>Alpha2)
    {
    
}
Alpha2=TriA;
KeyPoint2.x=contours[contoursNo][i].x;
KeyPoint2.y=contours[contoursNo][i].y;
}
}

//calculate the distance between the point and line
Dist1=abs(k*KeyPoint1.x-Point1.y+b)/sqrt(k*k+1);
Dist2=abs(k*KeyPoint2.x-Point2.y+b)/sqrt(k*k+1);
if (Dist1>Dist2)
{
CandiPointStack[CandiPointNum].x=KeyPoint1.x;
CandiPointStack[CandiPointNum].y=KeyPoint1.y;
CandiPointNum++;
EndPoint->x=KeyPoint2.x;
EndPoint->y=KeyPoint2.y;
}
else
{
CandiPointStack[CandiPointNum].x=KeyPoint2.x;
CandiPointStack[CandiPointNum].y=KeyPoint2.y;
CandiPointNum++;
EndPoint->x=KeyPoint1.x;
EndPoint->y=KeyPoint1.y;
}
int main(int argc, char** argv)
{

    //const char* filename = argc >= 2 ? argv[1] : "pic1.png";
    src = imread("D:\\OPENCV\\Test\\5.jpg" , 1 ); //readingt the data of image send to src
    cvtColor( src, gray_dst, CV_BGR2GRAY ); //tranlate src to Grayscale image
    blur(gray_dst,gray_dst,Size(3,3));
    cvNamedWindow( "Source", CV_WINDOW_AUTOSIZE ); //show the image on the screen
    imshow( "Source", src );

    //createTrackbar( "Threshold:", "Source", &thresh, max_thresh, thresh_callback );
    thresh_callback( 0, 0 );
    IplImage* srcPtr=(IplImage*)&src;
    Scalar color;
    color.val[0]=255;
    color.val[1]=255;
    color.val[2]=255;
    color.val[3]=0;

    //set the coordinate of start point and target
    center1.x=10;
    center1.y=10;
    center2.x=src.cols-50;
    center2.y=src.rows-10;
//show the start point and target

circle(drawing,center1,4,color,4,255);
circle(drawing,center2,2,color,2,255);
cvNamedWindow( "Hull demo", 1 );
imshow( "Hull demo", drawing );

//begin to calculate the route

CurrentPoint1.x=center1.x;
CurrentPoint1.y=center1.y;
CurrentPoint2.x=center2.x;
CurrentPoint2.y=center2.y;
RoutePoint[RouteNum].x= CurrentPoint1.x;
RoutePoint[RouteNum].y= CurrentPoint1.y;
RouteNum++;

int Cross=DetectCross( CurrentPoint1,CurrentPoint2,&iCurrent,&CrossPoint1,&CrossPoint2);

if (!Cross)
{
    RoutePoint[RouteNum]=CurrentPoint2;
    EndPointNum=-1;
}
else
{
    EndPointNum++;
    EndPointStack[EndPointNum]=CurrentPoint2;
CalcEndPoint(CurrentPoint1,CrossPoint1,CrossPoint2,iCurrent,&CurrentPoint2);
circle(drawing,CurrentPoint2,2,color,2,255);
circle(drawing,CrossPoint1,2,color,2,255);
circle(drawing,CrossPoint2,2,color,2,255);
cvNamedWindow( "Hull demo", 1 );
imshow( "Hull demo", drawing );
cvWaitKey(0);
}

while (EndPointNum>=0)
{
    Cross=DetectCross( CurrentPoint1,CurrentPoint2,&iCurrent,&CrossPoint1,&CrossPoint2);
circle(drawing,CrossPoint1,2,color,2,255);
circle(drawing,CrossPoint2,2,color,2,255);
cvNamedWindow( "Hull demo", 1 );
imshow( "Hull demo", drawing );
cvWaitKey(0);
    if (!Cross)
    {
        RoutePoint[RouteNum]= CurrentPoint2;
        RouteNum++;
        CurrentPoint1=CurrentPoint2;
        CurrentPoint2=EndPointStack[EndPointNum];
        EndPointNum--;
    }
}
// CalcEndPoint(CurrentPoint1, CrossPoint1, CrossPoint2, iCurrent, &CurrentPoint2);

else
{
    if (pointPolygonTest(contours[iCurrent], CurrentPoint2, 0) >= 0)
    {
        RoutePoint[RouteNum] = CurrentPoint2;
        RouteNum++;
        CurrentPoint1 = CurrentPoint2;
        CurrentPoint2 = EndPointStack[EndPointNum];
        EndPointNum--;
    }
    else
    {
        EndPointNum++;
        EndPointStack[EndPointNum] = CurrentPoint2;
        CalcEndPoint(CurrentPoint1, CrossPoint1, CrossPoint2, iCurrent, &CurrentPoint2);
    }
}
circle(drawing, CurrentPoint2, 2, color, 2, 255);
cvNamedWindow( "Hull demo", 1 );
imshow( "Hull demo", drawing );
cvWaitKey(0);
}

while (pointPolygonTest(contours[iCurrent], CurrentPoint1, 0) >= 0)
{

if (xDirect)
{
    if (yDirect)
    {
        CurrentPoint1.x=CurrentPoint1.x+20;
        CurrentPoint1.y=k*CurrentPoint1.x+b;
    }
    else
    {
        CurrentPoint1.x=CurrentPoint1.x-20;
        CurrentPoint1.y=k*CurrentPoint1.x+b;
    }


    if (yDirect)
    {
        CurrentPoint1.y=CurrentPoint1.y+20;
        CurrentPoint1.x=k*CurrentPoint1.y+b;
    }
    else
    {
        CurrentPoint1.y=CurrentPoint1.y-20;
        CurrentPoint1.x=k*CurrentPoint1.y+b;
    }
}


else
RoutePoint[RouteNum] = CurrentPoint1;
RouteNum++;

for (int i=0; i<RouteNum-1; i++)
{
    line(drawing, RoutePoint[i], RoutePoint[i+1], color);
}

cvNamedWindow( "Hull demo", 1 );
imshow( "Hull demo", drawing );
cvWaitKey(0);

/** @function thresh_callback */
void thresh_callback(int, void* )
{
    Mat src_copy = src.clone();
    Mat threshold_output;

    // Detect edges using Threshold
    threshold( gray_dst, threshold_output, thresh, 255, THRESH_BINARY );

    // Find contours
    findContours( threshold_output, contours, hierarchy, CV_RETR_TREE,
        CV_CHAIN_APPROX_SIMPLE, Point(0, 0) );
/// Find the convex hull object for each contour

for( int i = 0; i < contours.size(); i++ )
    { convexHull( Mat(contours[i]), hull[i], false ); } 

/// Draw contours + hull results

drawing = Mat::zeros( threshold_output.size(), CV_8UC3 ); 

for( int i = 0; i< contours.size(); i++ )
    {

        drawContours( drawing, contours, -1, Scalar(255),2 );
    
    }
}