Sl.no	Topic	Page.no
1.	**Abstract :**	1
2.	**1.0 Introduction** **1.1 General Methodology** **1.2 Terrain Mapping Technology**	2
3.	**2.0 Literature Survey** **2.1 Hazards** **2.2 Human Rescue Technique** **2.2.1 Scenario** **2.2.2 Destination**	3
4	**2.2.3 What went wrong?** **2.2.4 Logistics**	4
5.	**2.2.5 Approx view of the chamber** **2.2.6 Site Map**	6
6.	**2.2.7 The floor planning of rescue operation**	7
7.	**3.0 Drawbacks of Existing Systems** **4.0 Problem Statement** **5.0 objectives**	8
8.	**6.0 PLATFORM** **6.1 Legged Robot** **6.1.1 HEXAPOD**	9
9.	**7.0 Design Considerations**	10
10.	**8.0 Profile and Execution** **8.1 Software Used** **8.2 Design** **8.2.1 PART 1 LEG**	11
11.	**8.2.2 PART 2 LEG CONNECTORS** **8.2.3 PART 3 COMPLETE LEG WITH SERVO** **8.2.4 PART 4 MAIN PLATE**	12
12.	**9.0 STATIC STABILITY**	13
13.	**10.0 COMPLETE VIEW WITH SERVO IMPLANTATION** **10.1 Front View** **10.1.1 Front View Exploded**	14
14.	**10.2 Side View** **10.2.1 Side View Exploded**	15
15.	**10.3 Top View**	16

	10.3.1 Top View Exploded	
16.	**10.4 Isometric Explosion** **11.0 Mathematical Model** **11.1 Static Stability Criteria**	17
17.	**11.1.1 Free Body Diagram**	18.
18.	**11.1.1.1 Terminologies**	19.
19.	**11.2 Doing a Torque Balance Equation**	20
20.	**11.2.1 HIP EQUATION**	21
21.	**11.3 Simulation Based on Model** **11.3.1 Walking backward servo angle calculation** **11.3.2 Walking forward servo angle calculation**	22
22.	**12.0 THE HEXAPOD CONTROL EQUATIONS** **12.1 Inverse Kinematics** **12.1.1 Direct Inputs** **12.2 Robot Geometry**	23.
23.	**12.3Command Inputs** **12.4 Primary Geometric Calculations** **12.5 C.C.O Calculation**	24
24.	**12.5 C.C.O Calculation**	26
25.	**13.0 Hexapod flow of control** **14.0 The code blocks**	48
26.	**15.0 Automatic Stabilization Function**	54
27	**15.0 Automatic Stabilization Function**	56
28.	**16.0 Flow chart on how the image processing algorithm works**	59
29.	**17.0 Flow chart on the ML algorithm efficacy**	62
30.	**18.0 Result .** **19.0 Reference**	64.

Abstract :

Cave exploration has been a pretty interesting topic for the majority of daredevils out there , it so happens that the majority of them get trapped inside the cave they tend to explore and thereby falling to their doom .

The cave systems , however, being complex are not endless .

The emergence of robotics has now paved the way for limitless exploration of even such complex systems .

Introducing : Project CEEFRR.

CEEFRR is a Hexapod with 18[Degree of Freedom] under development for the sole purPOSe of cave exploration , excavation and rescue .

With the on board Microprocessor , cloud uplink and also the backend Terrain mapping algorithm [ANYMAL] . We could open up limitless POSsibilities of exploration and first response services offered to personnel involved in geological experiments .

Keyword : Hexapod , IOT , Image processing , Embedded System , Robotics.

1.0 INTRODUCTION :

1.1 General Methodology :

Cave rescue is a highly specialized field of wilderness rescue in which injured, trapped or lost cave explorers are medically treated and extracted from various cave environments.

Cave rescue borrows elements from firefighting, confined space rescue, rope rescue and mountaineering techniques but has also developed its own special techniques and skills for performing work in conditions that are almost always difficult and demanding. Since cave accidents, on an absolute scale, are a very limited form of incident, and cave rescue is a very specialized skill, normal emergency staff are rarely employed in the underground elements of the rescue. Instead, this is usually undertaken by other experienced cavers who undergo regular training through their organizations and are called up at need.

Cave rescues are slow, deliberate operations that require both a high level of organized teamwork and good communication. The extremes of the cave environment (air temperature, water, vertical depth) dictate every aspect of a cave rescue. Therefore, the rescuers must adapt skills and techniques that are as dynamic as the environment they must operate in.

1.2 Terrain Mapping Technology :

Terrain Analysis is the analysis and interpretation of topographic features through geographic information systems. Such features include slope, aspect, viewshed, elevation, contour lines, flow, upslope flowlines and downslope flowlines. The intention is to build mathematical abstraction of surface terrain in order to delineate or stratify landscapes and create an understanding of relationships between ecological processes and physical features.Slope measures the change in elevation over the change in horizontal POSition.Aspect is defined as the horizontal direction of the slope of topographic feature.Elevation refers to the vertical distance above a fixed points, most commonly sea level.

2.0 Literature Survey

2.1 Hazards

A novice's apprehension before a caving trip is healthy and an awareness of POSsible hazards helps you avoid them. Some of the dangers of caving.

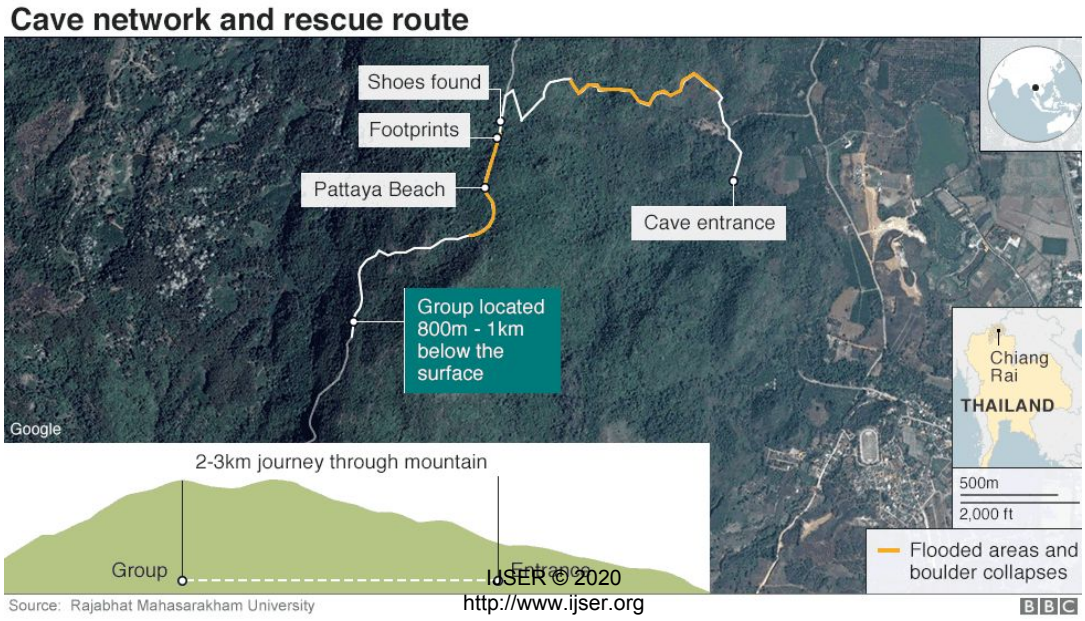
- Getting lost
- Running out of light
- Hypothermia
- Passages flooding
- Falling rocks
- Poor footing, falling
- Falling down pits

2.2 Human Rescue Technique :

Case Study : Thailand's cave rescue

2.2.1 Scenario : On 23 June 2018 , 13 boys went exploring in Thailand's Chiang Rai province with their football coach - and ended up trapped deep inside a cave underneath a mountain.

2.2.2 Destination : Tham Luang cave



2.2.3 What went wrong? :

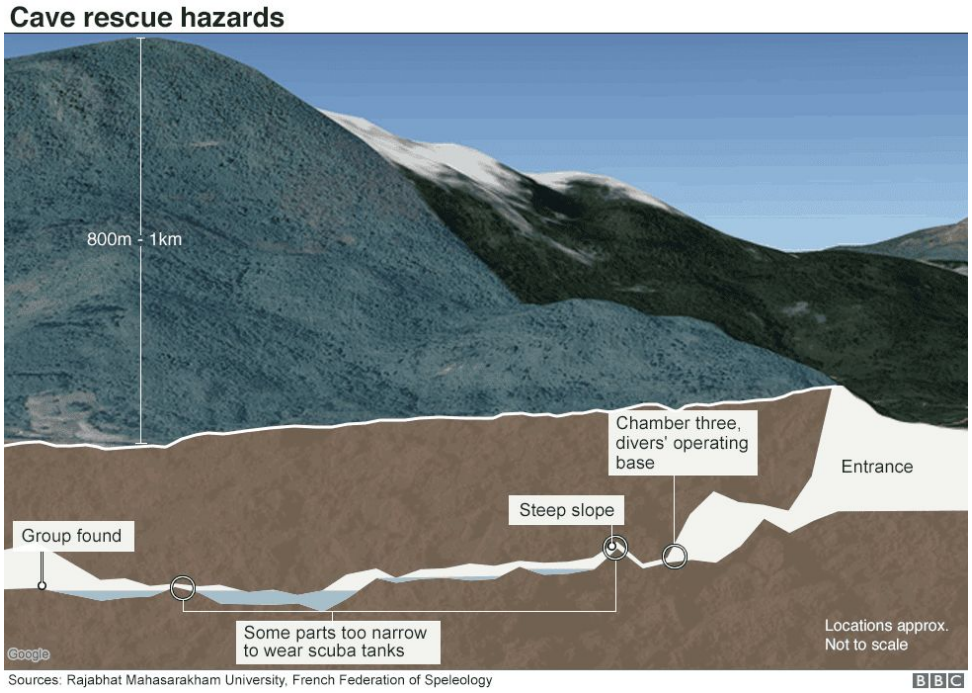
Sudden abrupt rainfall , flooding the inner chambers of the cave thereby preventing the cave escape route , the water clogging made the students stranded in a single chamber in the cave for a week .

2.2.4 Logistics :

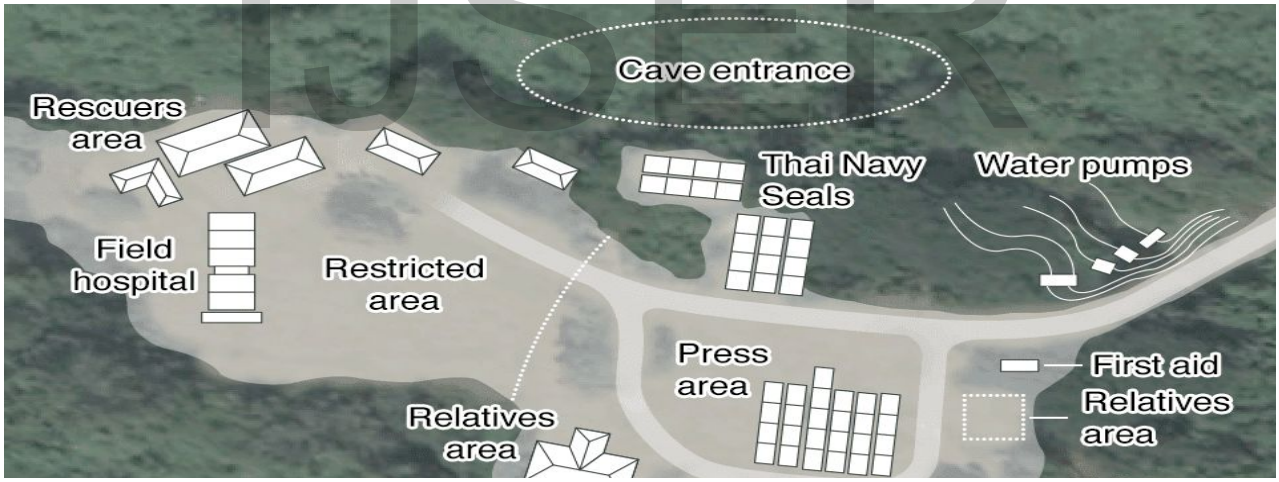
S.no	Name	Qty/nature
1.	No of students	13
2.	Time taken for rescue	1 week and 3 hours
3.	Severity of incident	Class 4 on National Geological Accident scale
4.	No of chambers in cave	58
5.	Depth at which students were found	800 m
6.	No of chambers flooded	32
7.	No of unflooded chambers	26
8.	Reason for delay	1. The rainfall had clogged all ways 2. Communication with students was cutoff because the entirety of cave entrance was shut down because of sudden land slide 3. The rescue team had to plan an alternate way to reach the chamber through a series of other chambers with the danger of cave collapsing on them 4. There were 3 major injuries to cave rescuers .

		5. Majority of cave channels were not accessible to humans. 6. In cave abnormalities such as Soft lime formation lead to severe degradation of equipments 7. The closeness to crust and the depth of cave made the human intervention into the pristine area highly hazardous and quite imPOSsible.
9.	No of Casualties	0
10.	Rescue Team	17 members
11.	Cost of Rescue	$13500
12.	Side Effect	1. 8 were unaffected 2. 3 suffered from nitrogen narcosis because of week long exPOSure to damp and mineral rich atmosphere of cave along with ancient pristine environment which had a huge nitrogen concentration 3. 2 suffered from Hypothermia

2.2.5 Approx view of the chamber

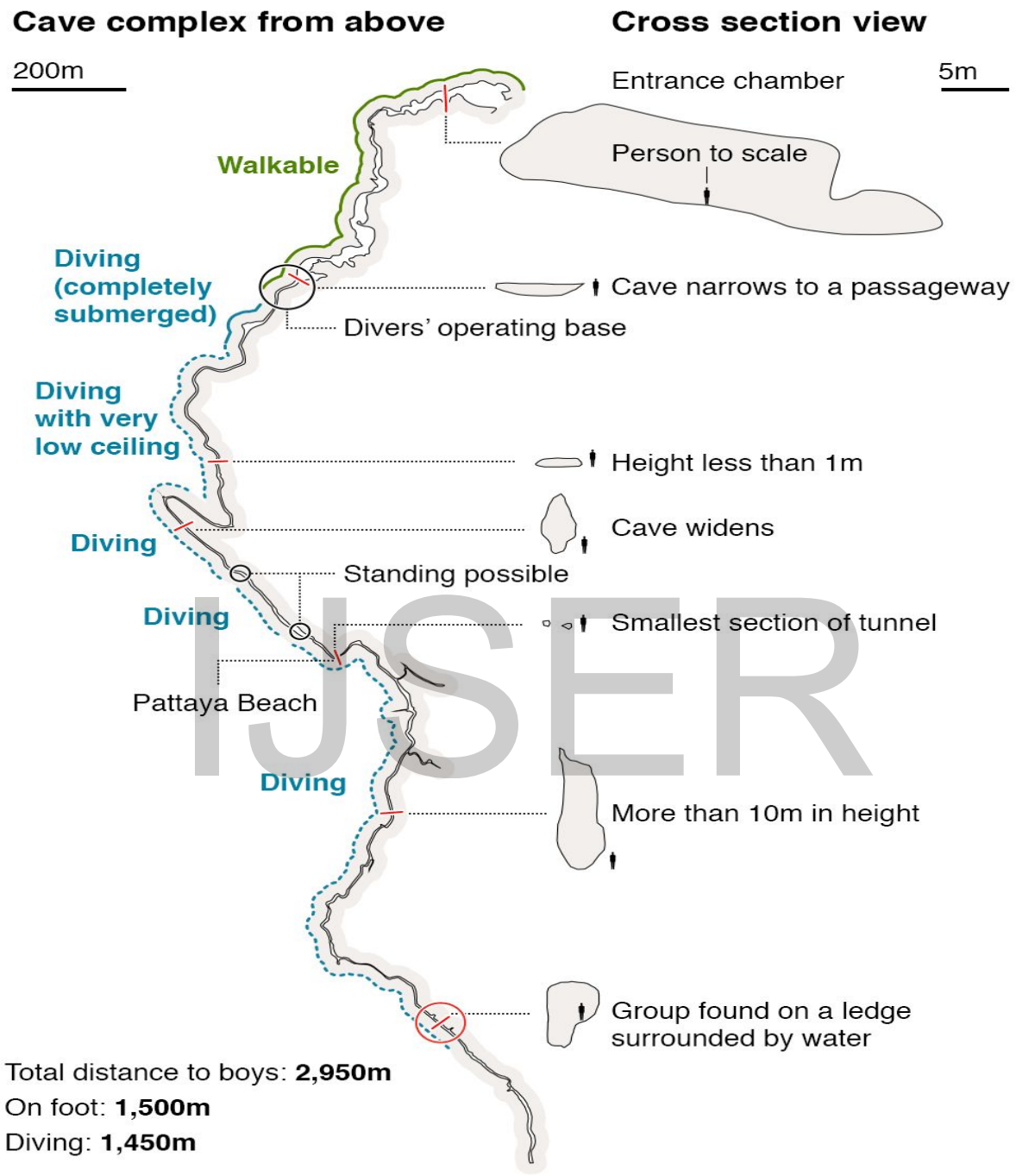


2.2.6 Site Map :



An entire private army had carried out the rescue operation with the most expensive decision of using the **NAVY SEALS[AN ELITE GROUP OF MERCENARIES]** for special war tactics .

2.2.7 The floor planning of rescue operation :



Cross-sections of the cave taken from 1986 survey

Sources: French Federation of Speleology, Rajabhat Mahasarakham University, Digitalay BBC

3.0 Drawbacks of Existing Systems

- Time consuming
- Less efficient
- Costly
- Based on assumptions
- Not predictable
- Not versatile
- Not scalable

4.0 Problem Statement:

To develop a Legged robot[**Embedded system**] that is scalable , versatile , fast , accurate , cheap , based on logic , scalable through principles of **IOT** .
A system that is dynamic and can trace terrains and can act both manual and autonomous for exploration of cave terrain and turf .
A system that collects data and stores it in the cloud and processes them for further insight about the situation .

5.0 objectives :

S.no	Objectives
1.	Scalable
2.	Versatile
3.	Fast
4.	Accurate
6.	Cheap
7.	Logic Based implementation
8.	IOT Backend [Class 4 IOT implementation]

6.0 PLATFORM

6.1 Legged Robot

The motor behaviors of animals are characterized by numerous features .Several of these basic features, such as self-organization, vestibular reflexes, and compliance, play fundamental roles in achieving adaptive and versatile locomotion behaviors. Self-organization of locomotion represents the capability of autonomously spontaneous locomotion generation . Vestibular reflexes and compliance can extend the functionality of self-organized locomotion in response to unexpected situations, such as abrupt changes in the ground plane and external perturbation. Therefore, understanding the biological principles of these properties contributes to revealing the underlying mechanisms of adaptive locomotion generation , and the subsequent development of advanced artificial legged robots). However, it is not convenient to investigate the locomotor principles by means of animal experiments alone, because, in general, it is difficult to perform repeated measurements of the variables or quantities of unrestrained animal behaviors

6.1.1 HEXAPOD :

Hexapod (6-legged) robots are most commonly configured in either two rows of 3 legs (3+3) or at 60 degrees from each other and equal distance from the center.

A dynamically stable walking robot must be in motion in order to prevent it from falling over. If the robot were to stop while walking the center of mass would cause it to fall over.

A statically stable robot can be stopped at any point during its gait and it will not fall over. In the case of a hexapod, so long as 3 legs are always in contact with the ground and the center of mass is located within the triangle formed by three feets, it will be statically stable.

7.0 Design Considerations

S.no	Name	Tool/Method /Qty
1.	Type	Hexapod
2.	Stability	Static
3.	Weight	3200g
4.	Centre of Mass	Controlled by Algorithm [Rapid stability and PID]
5.	Feedback	Closed feedback - POSition COG, COM
6.	Motor Torque	Stall torque Min : 10Kg-cm Max: 20Kg-cm
7.	Battery Specification	11.1V 3S 6C 10000mAh
8.	Expected Operating time	15 min Peak capacity
9.	No of motors	18
10.	Motor type	Metal Gear Servo
11.	Motor constraint	Angle of Rotation : 0-180 degree
12.	Motor min operating voltage	4.4 V
13.	Motor Max operating voltage	6.2V
14.	Motor Burn voltage	8.4 V
15.	Motor Burn current	2200 mAh
16.	Prototyping	Rapid prototyping/3D Printing
S.no	**Name**	**Tool/Method /Qty**
17.	Material	1. Aluminium Alloy 2mm Plate/Sheet-100x100x2mm

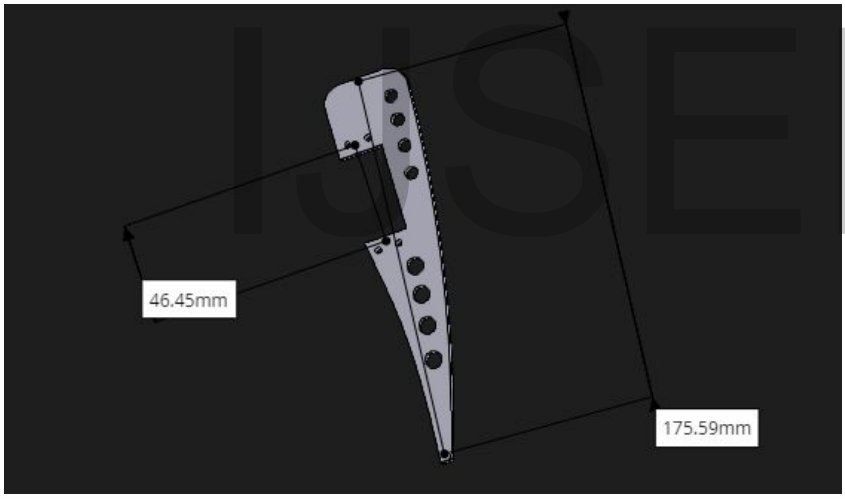
		2. ABS filament 3. Acrylic Sheets
18.	Infill	30%
19.	Cross slicing	X
20.	Cooling mechanism	None

8.0 Profile and Execution

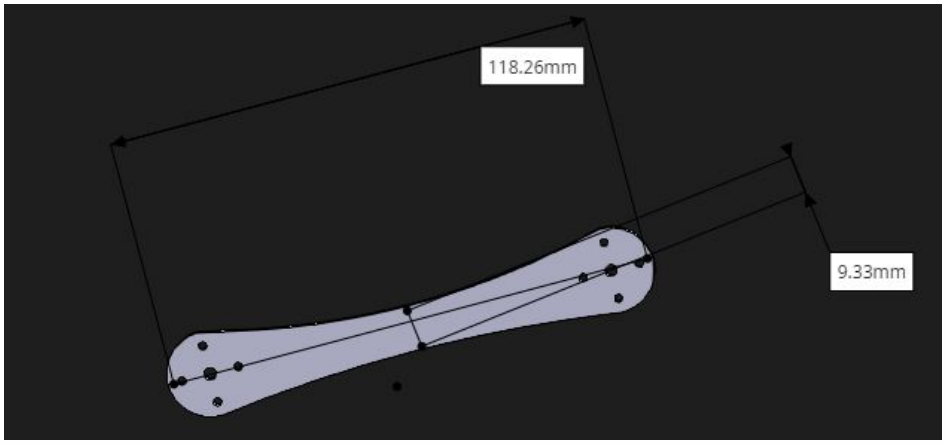
8.1 Software Used : Solid Works

8.2 Design :

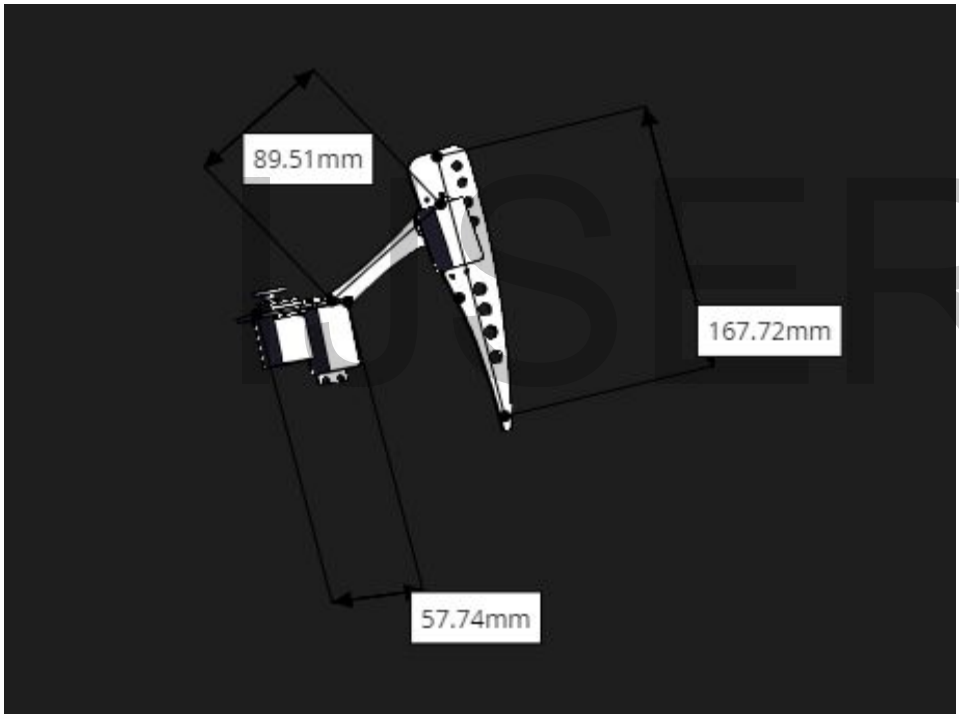
8.2.1 PART 1 LEG:



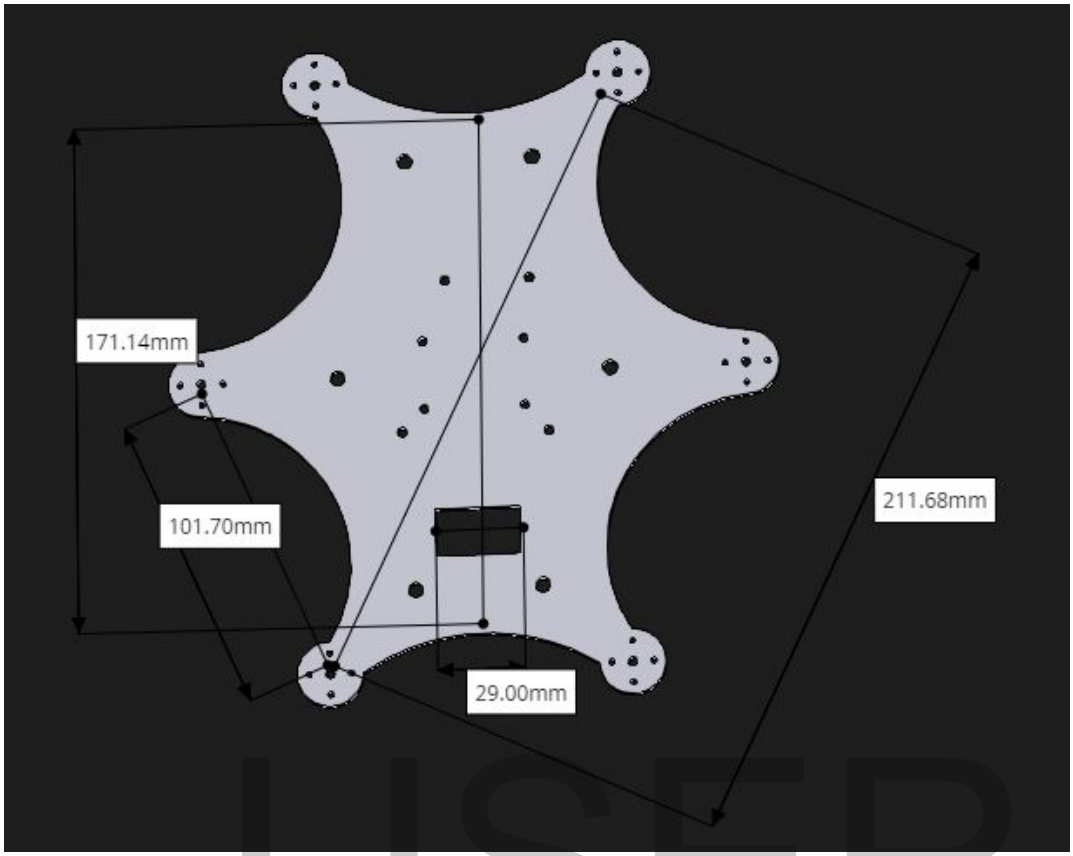
8.2.2 PART 2 LEG CONNECTORS



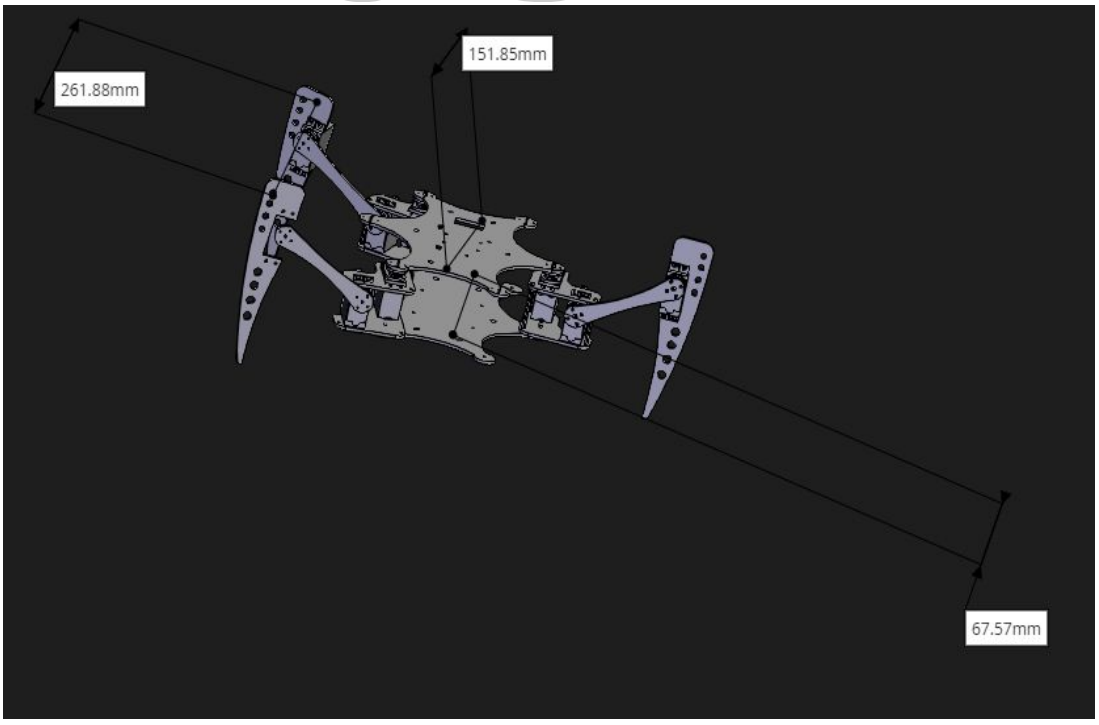
8.2.3 PART 3 COMPLETE LEG WITH SERVO



8.2.4 PART 4 MAIN PLATE

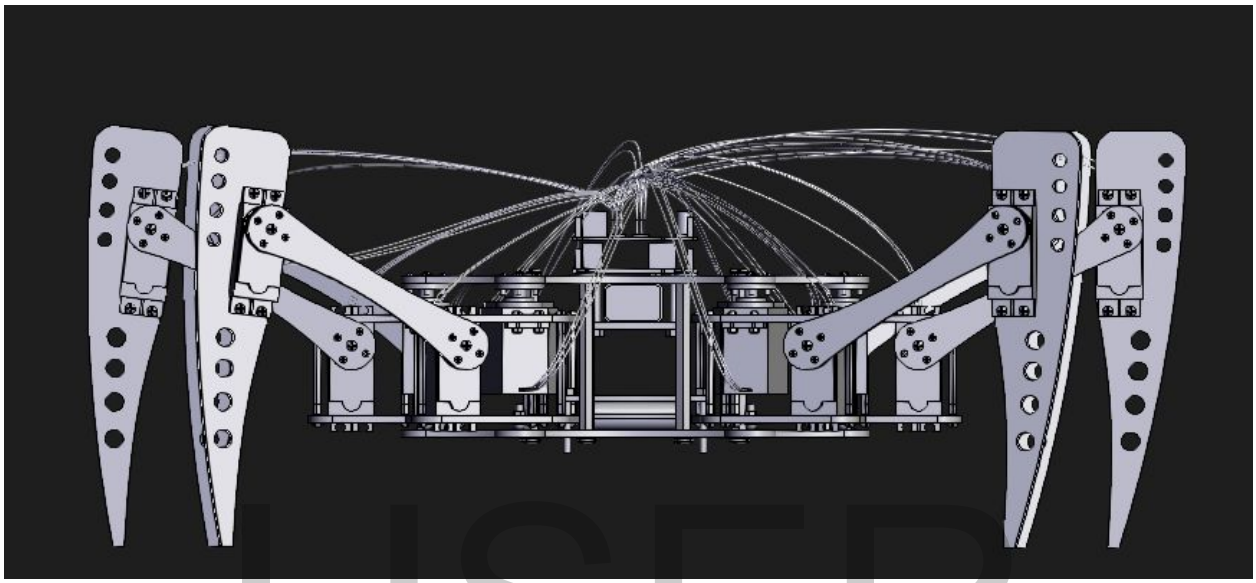


9.0 STATIC STABILITY

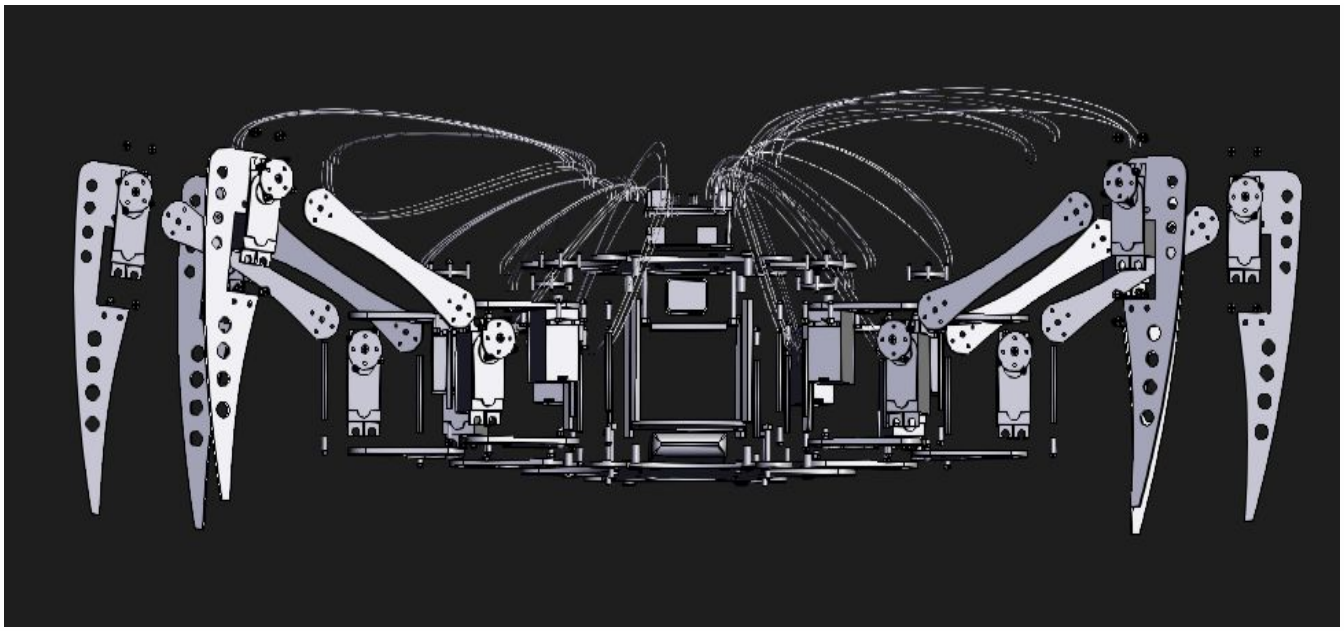


10.0 COMPLETE VIEW WITH SERVO IMPLANTATION

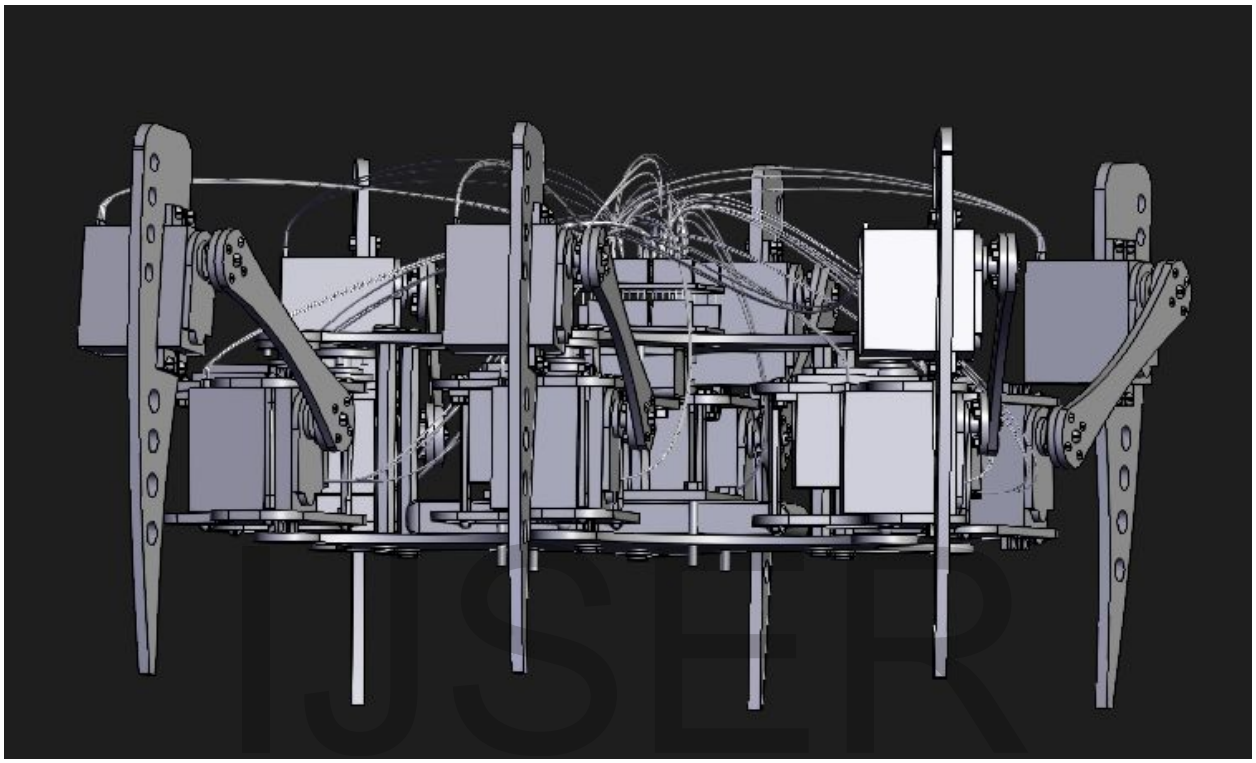
10.1 Front View:



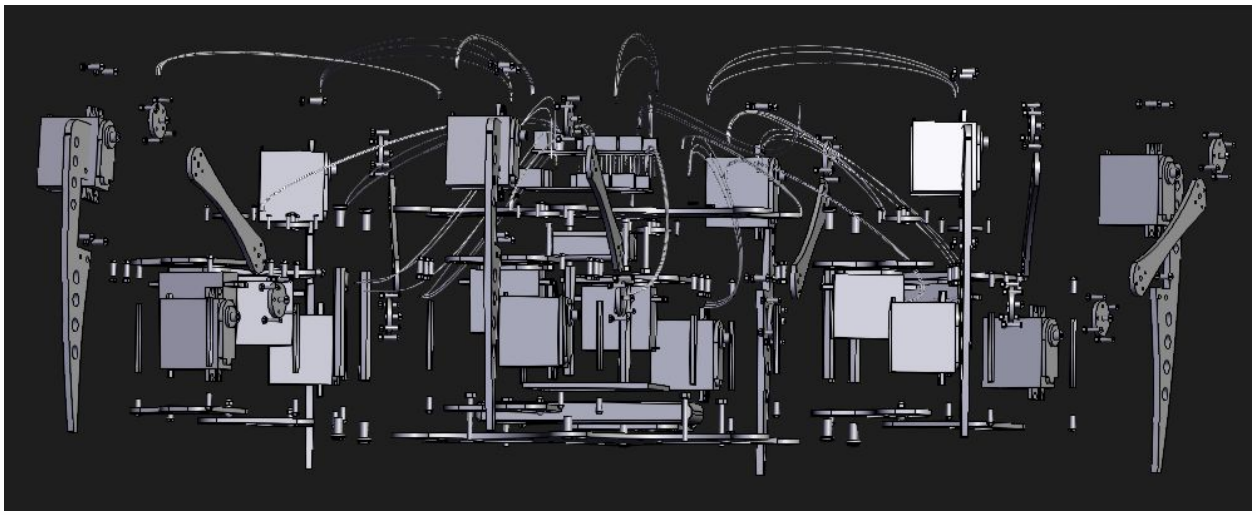
10.1.1 Front View Exploded



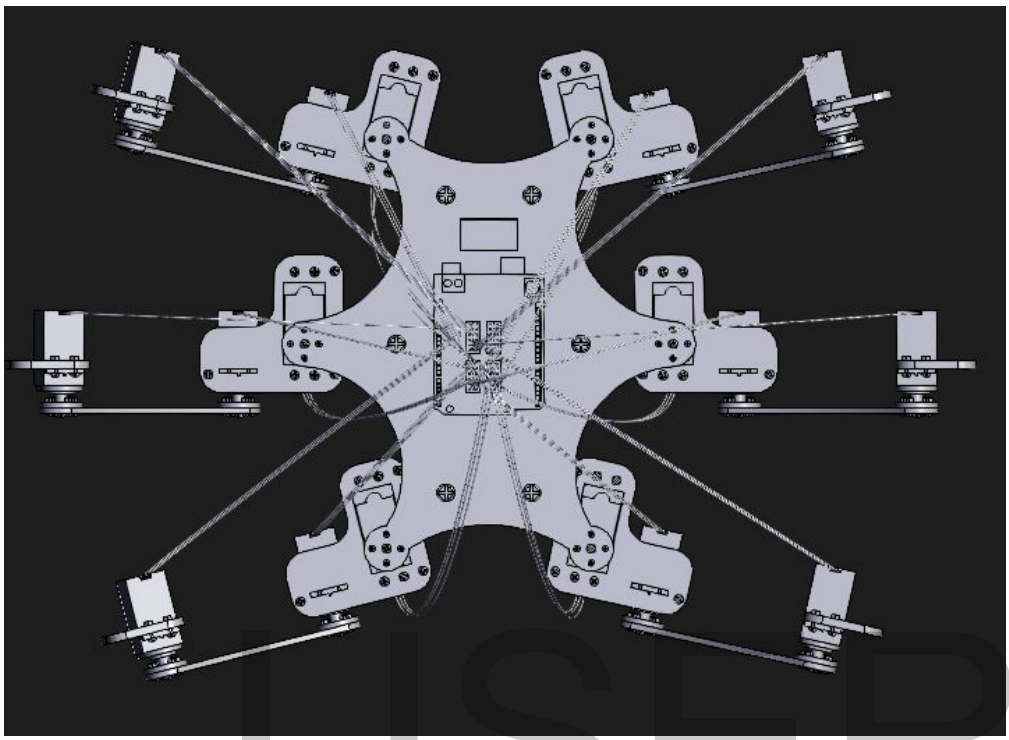
10.2 Side View



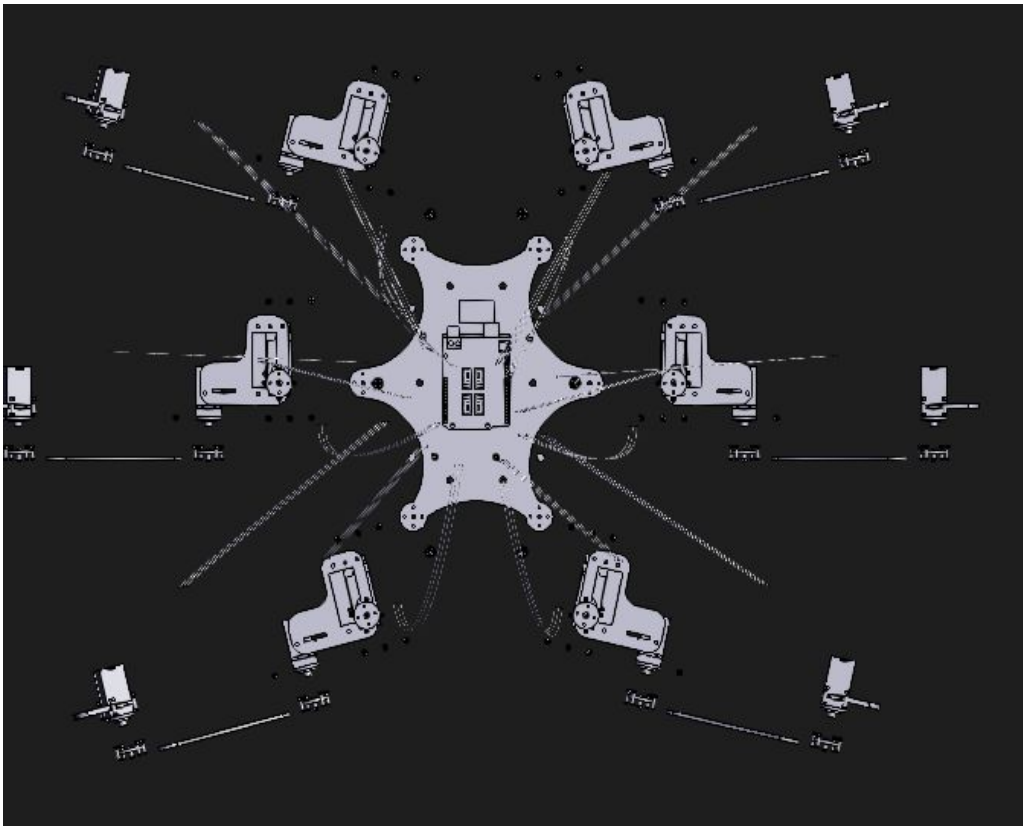
10.2.1 Side View Exploded



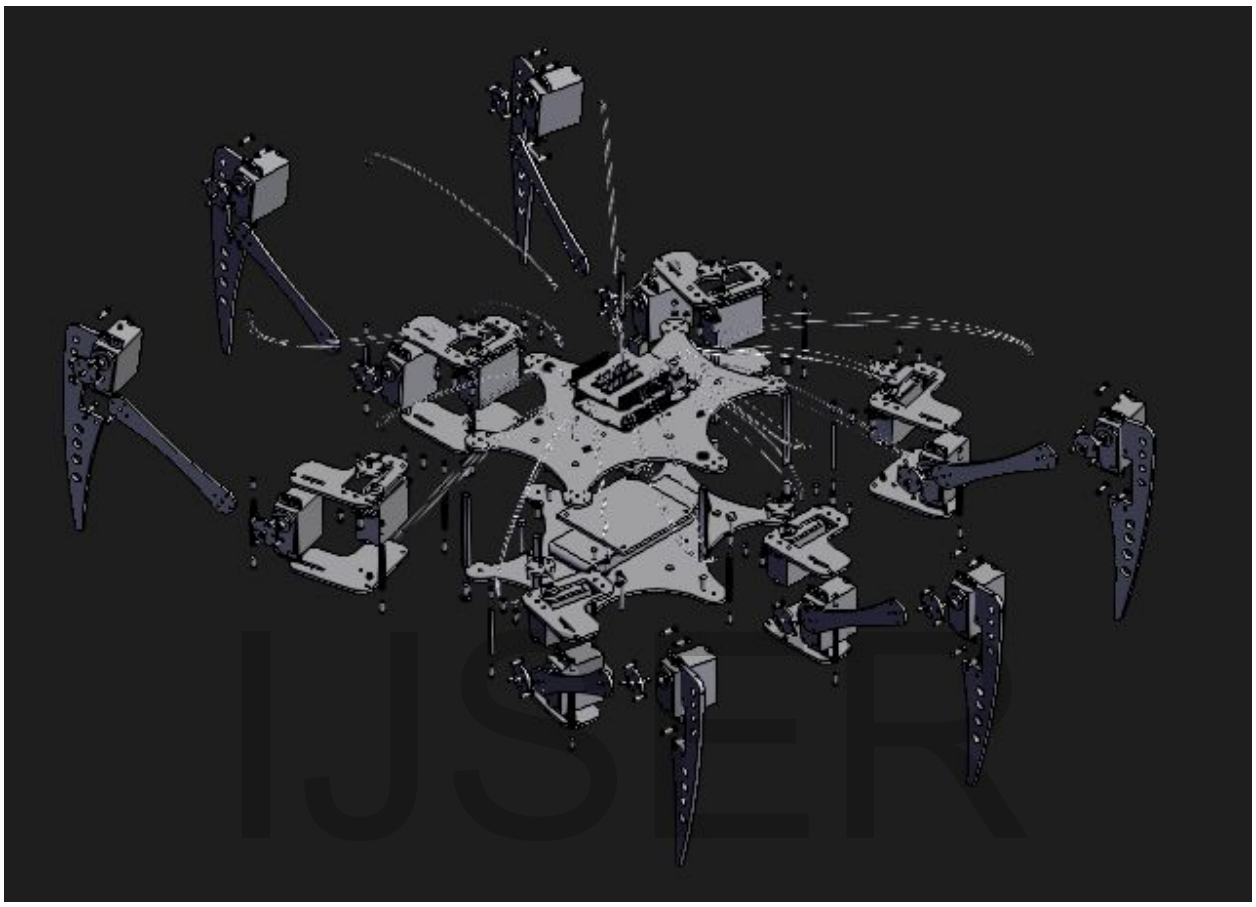
10.3 Top View



10.3.1 Top View Exploded



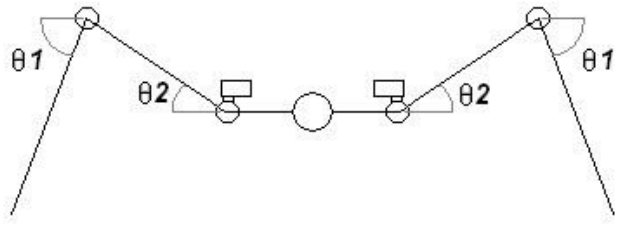
10.4 Isometric Explosion



11.0 Mathematical Model

11.1 Static Stability Criteria

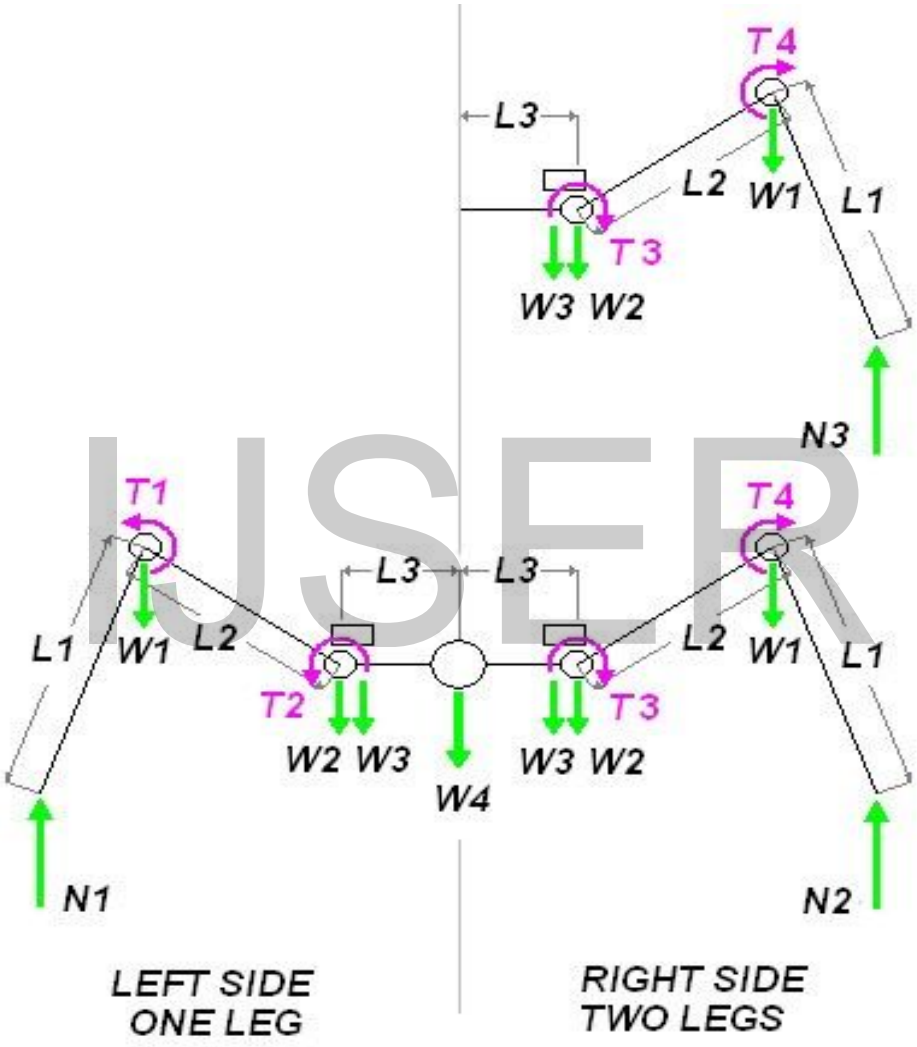
A statically stable robot can be stopped at any point during its gait and it will not fall over. In the case of a hexapod, so long as 3 legs are always in contact with the ground and the center of mass is located within the triangle formed by these feet, it will be statically stable.



To make calculations more accurate we make the following assumptions

1. 6 legs, configured as two rows of 3.
2. All the legs are identical

11.1.1 Free Body Diagram



11.1.1.1 Terminologies :

S.no	Terminology	Explanation
1.	NI, N2,N3	Normal Reaction : It is a force extended by the ground on the bot , W.R.T the force extended by the bot to the ground
2.	L1,L2,L3	Length between each actuator
3.	W1,W2,W3	Weight of Each Actuator [W1=W2=W3]
4.	T1,T2,T3	Torque at each Joint
5.	W4	Weight at centre when statically stable
6.	Wf	Weight of frame
7.	Welec	Weight of controller+component other than (actuator, battery)
8.	Wbat	Weight of battery
9.	Wleg	Weight of each leg

Based on the Free Body Diagram:

We deduce that

1. W4 = Wf + Welec + Wbat + 3*(Wleg) **[Statically stable]**

2. N2 and N3 are considered equal as they are supporting almost entire acting weight
3. Three legs must support the entire weight of the robot, as well as their own weight
   N1 + 2N2 = W4 + 6*(W1+W2+W3)

11.2 Doing a Torque Balance Equation :

$$\sum T_{LeftFoot} = -W1{*}l1{*}\cos\theta1$$
$$-W2{*}(L1{*}\cos\theta1+L2{*}\cos\theta2)$$
$$-W3(L1{*}\cos\theta1+L2{*}\cos\theta2)$$
$$-W4{*}(L1{*}\cos\theta1+L2{*}\cos\theta2+L3)$$
$$-2W3{*}(2L3+L1{*}\cos\theta1+L2{*}\cos\theta2)$$
$$-2W2{*}(L1{*}\cos\theta1+L2{*}\cos\theta2+2L3)$$
$$-2W1{*}(L1{*}\cos\theta1+2L2{*}\cos\theta2+2L3)$$
$$+2N2{*}(2L1\cos\theta1+2L2{*}\cos\theta2+2L3)$$

$$\sum T_{knee} = T1-N1{*}(L1{*}\cos\theta1)$$
$$-W2{*}(L2{*}\cos\theta2)-W3{*}(L2{*}\cos\theta2)$$
$$-W4{*}(L2{*}\cos\theta2+L3)-2W3{*}(L2{*}\cos\theta2+2L3)$$
$$-2W2{*}(L2{*}\cos\theta2+2L3)$$
$$-2W1{*}(2L2{*}\cos\theta2+2L3)$$
$$+2N2{*}(2L2{*}\cos\theta2+2L3+L1{*}\cos\theta1)\}$$

11.2.1 HIP EQUATION :

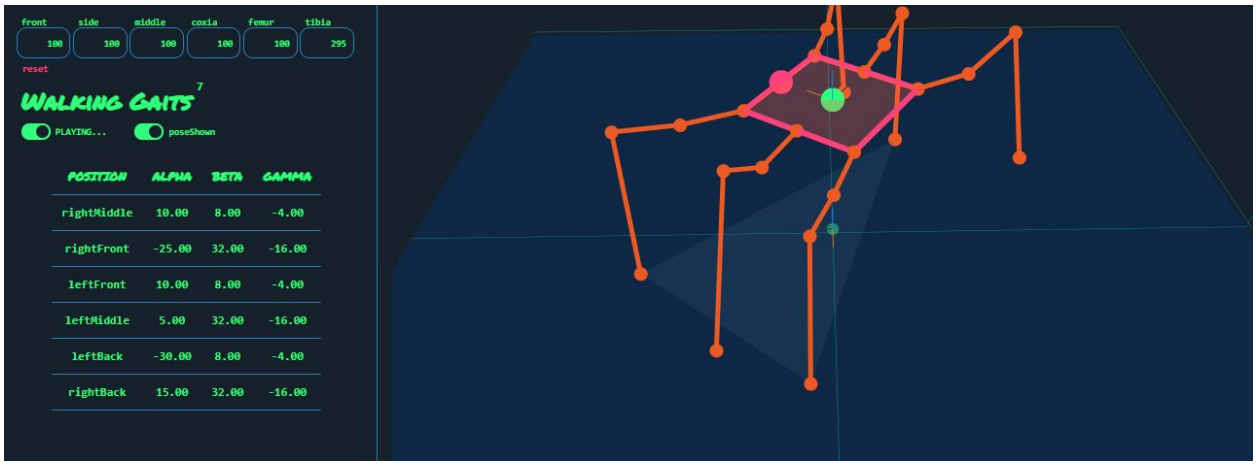
$$\sum T_{hip} = T2 - N1*(L1*\cos\theta 1 + L2*\cos\theta 2)$$
$$+W1*(L2*\cos\theta 2) - W4*L3 - 2W2*L3 - 2W3L3$$
$$-2W1*(2L3 + L2*\cos\theta 2)$$
$$+2N2(2L3 + L2*\cos\theta 2 + L1\cos\theta 1)$$

A torque balance about the left shoulder actuator gives:

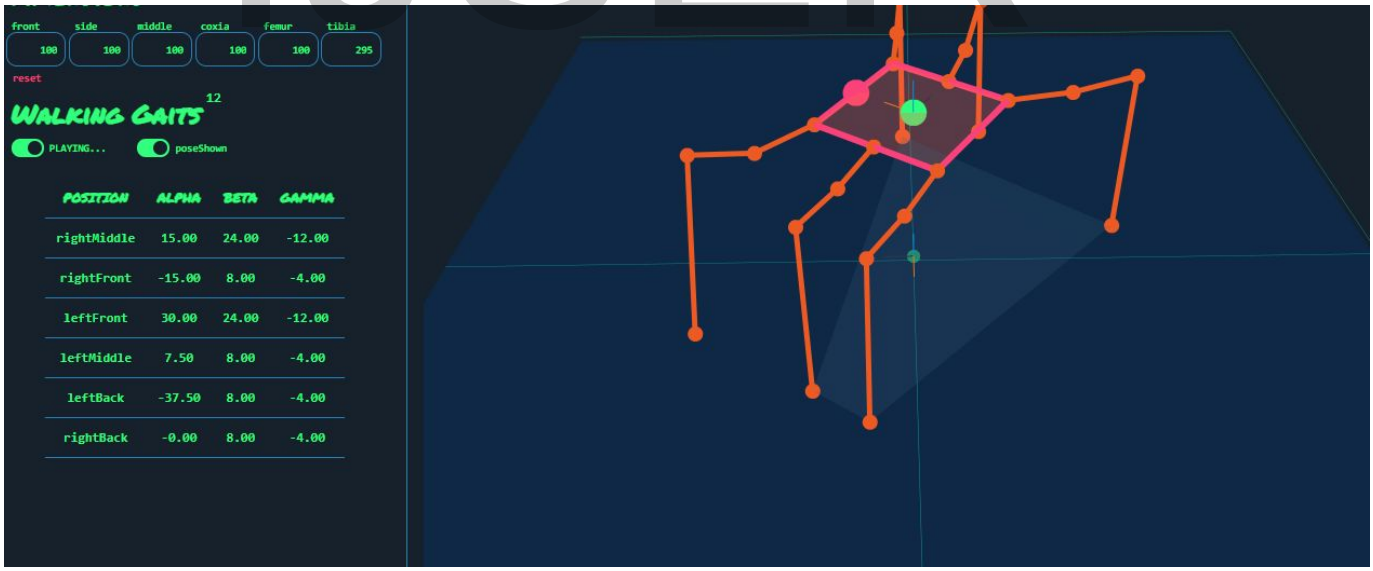
$$\sum T5 = T1 - F1(L1*\cos\theta 1 + L2*\cos\theta 2) - L3Wx$$
$$+2*F2*(2*L3 + L1*\cos\theta 1 + L2*\cos\theta 2)$$

11.3 Simulation Based on Model :

11.3.1 Walking backward servo angle calculation :



11.3.2 Walking forward servo angle calculation :



12.0 THE HEXAPOD CONTROL EQUATIONS

12.1 Inverse Kinematics :

Inverse kinematics refers to the use of the kinematics equations of a robot to determine the joint parameters that provide a desired POSition of the end-effector. Specification of the movement of a robot so that its end-effector achieves a desired task is known as motion planning. Inverse kinematics transforms the motion plan into joint actuator trajectories for the robot.

Inverse Kinematics (IK) is basically a mathematical method to make robots move in a particular way. Building robots that move around on legs is quite complex. Legged robots can have as many as eight legs (or more in theory, although I have never found an example of a hobby robot with more than eight legs), each of which typically has at least three servos. This mechanical complexity means that the math, the inverse kinematics, of legged robots is quite complex.

Robotic arms have three servos that control the POSition of the end of the arm, called the end effector. Legged robots have legs that are a bit like a bunch of upside-down robot arms. By changing the angle of each joint of the arm, we can change the POSition of the end effector. Controlling hexapod robots requires the use of **inverse kinematics.**

The inverse kinematics process for calculating the 18 servo angles is an actual process, with a start point and an endpoint.

12.1.1 Direct Inputs

The first step is to input information about the robot geometry and commands for the robot's movement. The geometric information is constant. This is information over which we,have direct control.

12.2 Robot Geometry

B.S.L ==D.I

CL ==D.I

FL ==D.I

TibiaLength ==D.I

12.3Command Inputs

POSX == C.I

POSY == C.I

POSZ == C.I

RotX == C.I

RotY == C.I

RotZ == C.I

12.4 Primary Geometric Calculations

From the geometry of the robot, the next step is to calculate other basic geometric relationships. Note that because this information is based on the robot's geometry, it is constant.

12.5 C.C.O Calculation

When we are sending commands to the hexapod to move in a particular way, those move commands refer to the POSition of the center of the hexapod. If we command the hexapod to move forward by 10cm, we are telling the center of the hexapod to move forward by that distance, and we compute the POSitions of all 18 servos on all six legs accordingly.

BCO 1 = B.S.L/2

BCO 2 = sqrt(B.S.L^2 - BCO 1^2)

Body Center Offset X

Leg		
1	BCO X_1	BCO 1
2	BCO X_2	B.S.L
3	BCO X_3	BCO 1

4	BCO X_4	-BCO 1
5	BCO X_5	-B.S.L
6	BCO X_6	-BCO 1

Body Center Offset Y

Leg		
1	BCO Y_1	BCO 2
2	BCO Y_2	0
3	BCO Y_3	-BCO 2
4	BCO Y_4	-BCO 2
5	BCO Y_5	0
6	BCO Y_6	BCO 2

12.6 Initial Feet POSitions

All feet POSitions expressed with respect to the coxa servo of the corresponding leg. In other words, we basically use two different reference frames for our IK calculations. In the previous

section we calculated the POSitions of the coxa servos of all six legs using the center of the hexapod as the origin point. Now, in this section, we will calculate the feet POSitions using the coxa servo of each leg as the origin point.

Leg 1

FeetPOSX_1	cos(60/180*PI)*(CL + FL)
FeetPOSZ_1	TibiaLength
FeetPOSY_1	sin(60/180*PI)*(CL + FL)

Leg 2

FeetPOSX_2	CL + FL
FeetPOSZ_2	TibiaLength
FeetPOSY_2	0

Leg 3

FeetPOSX_3	cos(60/180*PI)*(CL + FL)
FeetPOSZ_3	TibiaLength
FeetPOSY_3	sin(-60/180*PI)*(CL + FL)

Leg 4

FeetPOSX_4	-cos(60/180*PI)*(CL + FL)
FeetPOSZ_4	TibiaLength
FeetPOSY_4	sin(-60/180*PI)*(CL + FL)

Leg 5

FeetPOSX_5	-(CL + FL)

FeetPOSZ_5	TibiaLength
FeetPOSY_5	0

Leg 6

FeetPOSX_6	-cos(60/180*PI)*(CL + FL)
FeetPOSZ_6	TibiaLength
FeetPOSY_6	sin(60/180*PI)*(CL + FL)

Body IK

The equations for each leg are of the same general form.

Leg 1

TotalY_1	FeetPOSY_1 + BCO Y_1 + POSY

TotalX_1	FeetPOSX_1 + BCO X_1 + POSX
DistBodyCenterFeet_1	sqrt(TotalY_1^2 + TotalX_1^2)
AngleBodyCenterX_1	PI/2 - atan2(TotalY_1, TotalX_1)
RollZ_1	tan(RotZ * PI/180) * TotalX_1
PitchZ_1	tan(RotX * PI/180) * TotalY_1
BodyIKX_1	cos(AngleBodyCenterX_1 + (RotY * PI/180)) * DistBodyCenterFeet_1 - TotalX_1
BodyIKY_1	(sin(AngleBodyCenterX_1 + (RotY * PI/180)) * DistBodyCenterFeet_1) - TotalY_1
BodyIKZ_1	RollZ_1 + PitchZ_1

Leg 2

TotalY_2	FeetPOSY_2 + BCO Y_2 + POSY
TotalX_2	FeetPOSX_2 + POSX + BCO X_2
DistBodyCenterFeet_2	sqrt(TotalY_2^2 + TotalX_2^2)
AngleBodyCenterX_2	PI/2 - atan2(TotalY_2, TotalX_2)
RollZ_2	tan(RotZ * PI/180) * TotalX_2
PitchZ_2	tan(RotX * PI/180) * TotalY_2
BodyIKX_2	cos(AngleBodyCenterX_2 + (RotY * PI/180)) * DistBodyCenterFeet_2 - TotalX_2
BodyIKY_2	(sin(AngleBodyCenterX_2 + (RotY * PI/180)) * DistBodyCenterFeet_2) - TotalY_2

BodyIKZ_2	RollZ_2 + PitchZ_2

Leg 3

TotalY_3	FeetPOSY_3 + BCO Y_3 + POSY
TotalX_3	FeetPOSX_3 + BCO X_3 + POSX
DistBodyCenterFeet_3	sqrt(TotalY_3^2 + TotalX_3^2)
AngleBodyCenterX_3	PI/2 - atan2(TotalY_3, TotalX_3)
RollZ_3	tan(RotZ * PI/180) * TotalX_3
PitchZ_3	tan(RotX * PI/180) * TotalY_3
BodyIKX_3	cos(AngleBodyCenterX_3 + (RotY * PI/180)) * DistBodyCenterFeet_3 - TotalX_3

BodyIKY_3	(sin(AngleBodyCenterX_3 + (RotY * PI/180)) * DistBodyCenterFeet_3) - TotalY_3
BodyIKZ_3	RollZ_3 + PitchZ_3

Leg 4

TotalY_4	FeetPOSY_4 + BCO Y_4 + POSY
TotalX_4	FeetPOSX_4 + BCO X_4 + POSX
DistBodyCenterFeet_4	sqrt(TotalY_4^2 + TotalX_4^2)
AngleBodyCenterX_4	PI/2 - atan2(TotalY_4, TotalX_4)
RollZ_4	tan(RotZ * PI/180) * TotalX_4
PitchZ_4	tan(RotX * PI/180) * TotalY_4

BodyIKX_4	cos(AngleBodyCenterX_4 + (RotY * PI/180)) * DistBodyCenterFeet_4 - TotalX_4
BodyIKY_4	(sin(AngleBodyCenterX_4 + (RotY * PI/180)) * DistBodyCenterFeet_4) - TotalY_4
BodyIKZ_4	RollZ_4 + PitchZ_4

Leg 5

TotalY_5	FeetPOSY_5 + BCO Y_5 + POSY
TotalX_5	FeetPOSX_5 + BCO X_5 + POSX
DistBodyCenterFeet_5	sqrt(TotalY_5^2 + TotalX_5^2)
AngleBodyCenterX_5	PI/2 - atan2(TotalY_5, TotalX_5)

RollZ_5	tan(RotZ * PI/180) * TotalX_5
PitchZ_5	tan(RotX * PI/180) * TotalY_5
BodyIKX_5	cos(AngleBodyCenterX_5 + (RotY * PI/180)) * DistBodyCenterFeet_5 - TotalX_5
BodyIKY_5	(sin(AngleBodyCenterX_5 + (RotY * PI/180)) * DistBodyCenterFeet_5) - TotalY_5
BodyIKZ_5	RollZ_5 + PitchZ_5

Leg 6

TotalY_6	FeetPOSY_6 + BCO ZY_6 + POSY
TotalX_6	FeetPOSX_6 + BCO X_6 + POSX
DistBodyCenterFeet_6	sqrt(TotalY_6^2 + TotalX_6^2)

AngleBodyCenterX_6	PI/2 - atan2(TotalY_6, TotalX_6)
RollZ_6	tan(RotZ * PI/180) * TotalX_6
PitchZ_6	tan(RotX * PI/180) * TotalY_6
BodyIKX_6	cos(AngleBodyCenterX_6 + (RotY * PI/180)) * DistBodyCenterFeet_6 - TotalX_6
BodyIKY_6	(sin(AngleBodyCenterX_6 + (RotY * PI/180)) * DistBodyCenterFeet_6) - TotalY_6
BodyIKZ_6	RollZ_6 + PitchZ_6

Leg IK

Leg 1

NewPOSX_1	FeetPOSX_1 + POSX + BodyIKX_1

NewPOSZ_1	FeetPOSZ_1 + POSZ + BodyIKZ_1
NewPOSY_1	FeetPOSY_1 + POSY + BodyIKY_1
CoxaFeetDist_1	sqrt(NewPOSX_1^2 + NewPOSY_1^2)
IKSW_1	sqrt((CoxaFeetDist_1 - CL) ^2 + NewPOSZ_1^2)
IKA1_1	atan((CoxaFeetDist_1 - CL)/NewPOSZ_1)
IKA2_1	acos((TibiaLength^2 - FL^2 - IKSW_1^2)/(-2 * IKSW_1 * FL))
TAngle_1	acos((IKSW_1^2 - TibiaLength^2 - FL^2)/(-2 * FL * TibiaLength))
IKTibiaAngle_1	90 - TAngle_1 * 180/PI

IKFemurAngle_1	90 - (IKA1_1 + IKA2_1) * 180/PI
IKCoxaAngle_1	90 - atan2(NewPOSY_1, NewPOSX_1) * 180/PI

Leg 2

NewPOSX_2	FeetPOSX_2 + POSX + BodyIKX_2
NewPOSZ_2	FeetPOSZ_2 + POSZ + BodyIKZ_2
NewPOSY_2	FeetPOSY_2 + POSY + BodyIKY_2
CoxaFeetDist_2	sqrt(NewPOSX_2^2 + NewPOSY_2^2)
IKSW_2	sqrt((CoxaFeetDist_2 - CL) ^2 + NewPOSZ_2^2)

IKA1_2	atan((CoxaFeetDist_2 - CL)/NewPOSZ_2)
IKA2_2	acos((TibiaLength^2 - FL^2 - IKSW_2^2)/(-2 * IKSW_2 * FL))
TAngle_2	acos((IKSW_2^2 - TibiaLength^2 - FL^2)/(-2 * FL * TibiaLength))
IKTibiaAngle_2	90 - TAngle_2 * 180/PI
IKFemurAngle_2	90 - (IKA1_2 + IKA2_2) * 180/PI
IKCoxaAngle_2	90 - atan2(NewPOSY_2, NewPOSX_2) * 180/PI

Leg 3

NewPOSX_3	FeetPOSX_3 + POSX + BodyIKX_3

NewPOSZ_3	FeetPOSZ_3 + POSZ + BodyIKZ_3
NewPOSY_3	FeetPOSY_2 + POSY + BodyIKY_2
CoxaFeetDist_3	sqrt(NewPOSX_3^2 + NewPOSY_3^2)
IKSW_3	sqrt((CoxaFeetDist_3 - CL) ^2 + NewPOSZ_3^2)
IKA1_3	atan((CoxaFeetDist_3 - CL)/NewPOSZ_3)
IKA2_3	acos((TibiaLength^2 - FL^2 - IKSW_3^2)/(-2 * IKSW_3 * FL))
TAngle_3	acos((IKSW_3^2 - TibiaLength^2 - FL^2)/(-2 * FL * TibiaLength))
IKTibiaAngle_3	90 - TAngle_3 * 180/PI

IKFemurAngle_3	90 - (IKA1_3 + IKA2_3) * 180/PI
IKCoxaAngle_3	90 - atan2(NewPOSY_3, NewPOSX_3) * 180/PI

Leg 4

NewPOSX_4	FeetPOSX_4 + POSX + BodyIKX_4
NewPOSZ_4	FeetPOSZ_4 + POSYZ + BodyIKZ_4
NewPOSY_4	FeetPOSY_4 + POSY + BodyIKY_4
CoxaFeetDist_4	sqrt(NewPOSX_4^2 + NewPOSY_4^2)
IKSW_4	sqrt((CoxaFeetDist_4 - CL) ^2 + NewPOSZ_4^2)

IKA1_4	atan((CoxaFeetDist_4 - CL)/NewPOSZ_4)
IKA2_4	acos((TibiaLength^2 - FL^2 - IKSW_4^2)/(-2 * IKSW_4 * FL))
TAngle_4	acos((IKSW_4^2 - TibiaLength^2 - FL^2)/(-2 * FL * TibiaLength))
IKTibiaAngle_4	90 - TAngle_4 * 180/PI
IKFemurAngle_4	90 - (IKA1_4 + IKA2_4) * 180/PI
IKCoxaAngle_4	90 - atan2(NewPOSY_4, NewPOSX_4) * 180/PI

Leg 5

NewPOSX_5	FeetPOSX_5 + POSX + BodyIKX_5

NewPOSZ_5	FeetPOSZ_5 + POSZ + BodyIKZ_5
NewPOSY_5	FeetPOSY_5 + POSY + BodyIKY_5
CoxaFeetDist_5	sqrt(NewPOSX_5^2 + NewPOSY_5^2)
IKSW_5	sqrt((CoxaFeetDist_5 - CL) ^2 + NewPOSZ_5^2)
IKA1_5	atan((CoxaFeetDist_5 - CL)/NewPOSZ_5)
IKA2_5	acos((TibiaLength^2 - FL^2 - IKSW_5^2)/(-2 * IKSW_5 * FL))
TAngle_5	acos((IKSW_5^2 - TibiaLength^2 - FL^2)/(-2 * FL * TibiaLength))
IKTibiaAngle_5	90 - TAngle_5 * 180/PI

IKFemurAngle_5	90 - (IKA1_5 + IKA2_5) * 180/PI
IKCoxaAngle_5	90 - atan2(NewPOSY_5, NewPOSX_5) * 180/PI

Leg 6

NewPOSX_6	FeetPOSX_6 + POSX + BodyIKX_6
NewPOSZ_6	FeetPOSZ_6 + POSZ + BodyIKZ_6
NewPOSY_6	FeetPOSY_6 + POSY + BodyIKY_6
CoxaFeetDist_6	sqrt(NewPOSX_6^2 + NewPOSY_6^2)
IKSW_6	sqrt((CoxaFeetDist_6 - CL) ^2 + NewPOSZ_6^2)

IKA1_6	atan((CoxaFeetDist_6 - CL)/NewPOSZ_6)
IKA2_6	acos((TibiaLength^2 - FL^2 - IKSW_6^2)/(-2 * IKSW_6 * FL))
TAngle_6	acos((IKSW_6^2 - TibiaLength^2 - FL^2)/(-2 * FL * TibiaLength))
IKTibiaAngle_6	90 - TAngle_6 * 180/PI
IKFemurAngle_6	90 - (IKA1_6 + IKA2_6) * 180/PI
IKCoxaAngle_6	90 - atan2(NewPOSY_6, NewPOSX_6) * 180/PI

Servo Angles

Leg 1

CoxaAngle_1	IKCoxaAngle_1 - 60
FemurAngle_1	IKFemurAngle_1

TibiaAngle_1	IKTibiaAngle_1

Leg 2

CoxaAngle_2	IKCoxaAngle_2
FemurAngle_2	IKFemurAngle_2
TibiaAngle_2	IKTibiaAngle_2

Leg 3

CoxaAngle_3	IKCoxaAngle_3 + 60
FemurAngle_3	IKFemurAngle_3
TibiaAngle_3	IKTibiaAngle_3

Leg 4

CoxaAngle_4	IKCoxaAngle_4 - 240
FemurAngle_4	IKFemurAngle_4
TibiaAngle_4	IKTibiaAngle_4

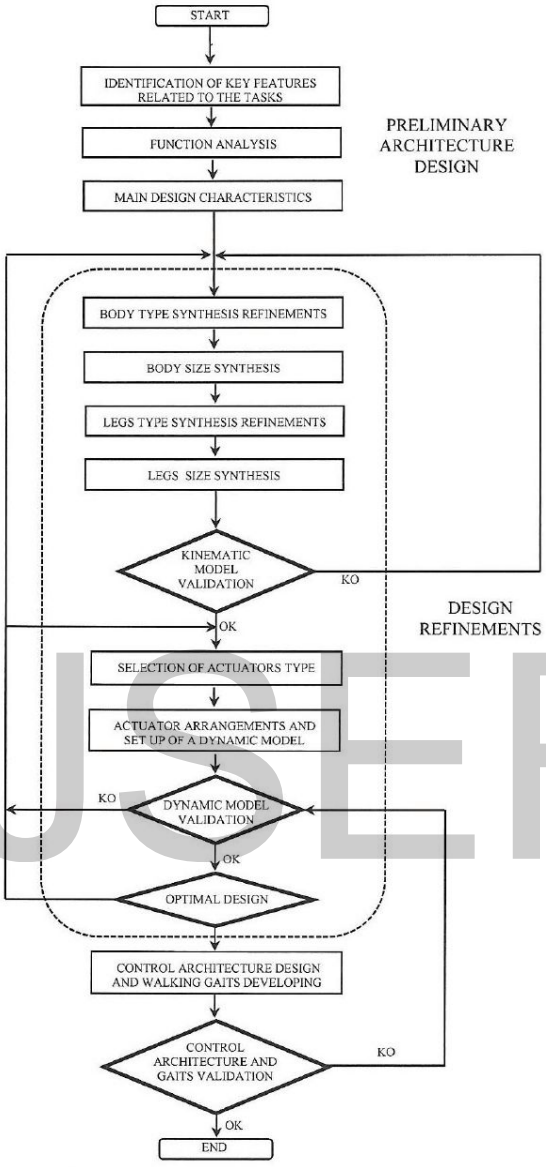
Leg 5

CoxaAngle_5	IKCoxaAngle_5 - 180
FemurAngle_5	IKFemurAngle_5
TibiaAngle_5	IKTibiaAngle_5

Leg 6

CoxaAngle_6	IKCoxaAngle_6 - 120
FemurAngle_6	IKFemurAngle_6
TibiaAngle_6	IKTibiaAngle_6

13.0 Hexapod flow of control



14.0 The code blocks :

```
import move
import RPIservo
init_pwm0 = 300
init_pwm1 = 300
init_pwm2 = 300
init_pwm3 = 300
init_pwm4 = 300
init_pwm5 = 300
```

```
init_pwm6 = 300
init_pwm7 = 300
init_pwm8 = 300
init_pwm9 = 300
init_pwm10 = 300
init_pwm11 = 300
init_pwm12 = 300
init_pwm13 = 300
init_pwm14 = 300
init_pwm15 = 300
for i in range(0,16):
        exec('pwm%d=RPIservo.init_pwm%d'%(i,i))
def init_all():
        pwm.set_pwm(0, 0, pwm0)
        pwm.set_pwm(1, 0, pwm1)
        pwm.set_pwm(2, 0, pwm2)
        pwm.set_pwm(3, 0, pwm3)
        pwm.set_pwm(4, 0, pwm4)
        pwm.set_pwm(5, 0, pwm5)
        pwm.set_pwm(6, 0, pwm6)
        pwm.set_pwm(7, 0, pwm7)
        pwm.set_pwm(8, 0, pwm8)
        pwm.set_pwm(9, 0, pwm9)
        pwm.set_pwm(10, 0, pwm10)
        pwm.set_pwm(11, 0, pwm11)
        pwm.set_pwm(12, 0, pwm12)
        pwm.set_pwm(13, 0, pwm13)
        pwm.set_pwm(14, 0, pwm14)
        pwm.set_pwm(15, 0, pwm15)
init_all()
def robotCtrl(command_input, response):
                global direction_command, turn_command
                if 'forward' == command_input: #move forward
                        direction_command = 'forward'
                        move.commandInput(direction_command) #move backward
                elif 'backward' == command_input:
                        direction_command = 'backward'
                        move.commandInput(direction_command)
                elif 'DS' in command_input: #stop
                        direction_command = 'stand'
```

```
                move.commandInput(direction_command)
            elif 'left' == command_input: #turn left
                turn_command = 'left'
                move.commandInput(turn_command)
            elif 'right' == command_input: #turn right
                turn_command = 'right'
                move.commandInput(turn_command)
            elif 'TS' in command_input: #stop
                turn_command = 'no'
                move.commandInput(turn_command)
            elif 'lookleft' == command_input: # the platform turns left
                P_sc.singleServo(12, 1, 7)
            elif 'lookright' == command_input: #the platform turns right
                P_sc.singleServo(12,-1, 7)
            elif 'LRstop' in command_input: #the platform stops turning left and right
                P_sc.stopWiggle()
            elif 'up' == command_input: #platform turns upward
                T_sc.singleServo(13, -1, 7)
            elif 'down' == command_input: #platform turns downward
                T_sc.singleServo(13, 1, 7)
            elif 'UDstop' in command_input: #the platform stops turning up and down
                T_sc.stopWiggle()
def functionSelect(command_input, response):
            global direction_command, turn_command, SmoothMode, steadyMode,
            functionMode
            if 'scan' == command_input:
                pass
            elif 'findColor' == command_input:
                flask_apPOSmodeselect('findColor')
            elif 'motionGet' == command_input: # Moving object detection
                flask_apPOSmodeselect('watchDog')
            elif 'stopCV' == command_input:
                flask_apPOSmodeselect('none')
                switch.switch(1,0)
                switch.switch(2,0)
                switch.switch(3,0)
            elif 'KD' == command_input: #Self-balancing
                move.commandInput(command_input)
            elif 'automaticOff' == command_input: # switch to default fast gait
                move.commandInput(command_input)
```

```
                elif 'automatic' == command_input: # switch to slow gait
                        move.commandInput(command_input)
                elif 'trackLine' == command_input: # track line on
                        flask_apPOSmodeselect('findlineCV')
                elif 'trackLineOff' == command_input: # track line off
                        flask_apPOSmodeselect('none')
                elif 'speech' == command_input: # turn on the alarm light
                        RL.police()
                elif 'speechOff' == command_input: # turn off the alarm light
                        RL.pause()
def move_thread():
global step_set
stand_stu = 1
if not steadyMode:
'''
#The default value of steadyMode is equal to True when the initial value is 0, and then else is not executed. When executing the self-balancing action, setting the initial value of steadyMode to 1
, which is if not steadyMode: equal to False, then it will execute else.
'''
if direction_command == 'forward' and turn_command == 'no':
if SmoothMode:
'''
#The default value of SmoothMode is equal to False when the initial value is 0, and it w
ill execute else; when switching to slow gait, setting the initial value to 1, you will get if Smoot
hMode: equal to True, then it will execute dove ().
'''
dove(step_set,35,0.001,DPI,'no') # slow gait of moving forward
step_set += 1
if step_set == 5:
step_set = 1
else:
move(step_set, 35, 'no') # fast gait of moving forward
time.sleep(0.1)
step_set += 1
if step_set == 5:
step_set = 1
elif direction_command == 'backward' and turn_command == 'no':
if SmoothMode:
dove(step_set,-35,0.001,DPI,'no') #slow gait of moving backward
```

```
step_set += 1
if step_set == 5:
step_set = 1
else:
move(step_set, -35, 'no') # fast gait of moving backward
time.sleep(0.1)
step_set += 1
if step_set == 5:
75
step_set = 1
else:
pass
if turn_command != 'no': # gait of turning left and right
if SmoothMode:
dove(step_set,35,0.001,DPI,turn_command)
step_set += 1
if step_set == 5:
step_set = 1
else:
move(step_set, 35, turn_command)
time.sleep(0.1)
step_set += 1
if step_set == 5:
step_set = 1
else:
pass
if turn_command == 'no' and direction_command == 'stand':
stand()
step_set = 1
pass
else: #Self-balancing
steady_X()
steady()
class RobotM(threading.Thread):
def __init__(self, *args, **kwargs):
super(RobotM, self).__init__(*args, **kwargs)
self.__flag = threading.Event() #The default flag is False, set the flag to True when calli
ng set() self.__flag.clear()
def pause(self):
#print('.....pause.....')
```

```
self.__flag.clear()
def resume(self):
self.__flag.set()
def run(self):
while 1:
self.__flag.wait() #wait for set()
move_thread()
pass
rm = RobotM()
rm.start()
rm.pause()
76
def commandInput(command_input):
global direction_command, turn_command, SmoothMode, steadyMode
if 'forward' == command_input:
direction_command = 'forward' #move forward
rm.resume()
elif 'backward' == command_input:
direction_command = 'backward' #move backward
rm.resume()
elif 'stand' in command_input:
direction_command = 'stand' #stand at attention
rm.pause()
elif 'left' == command_input:
turn_command = 'left' #turn left
rm.resume()
elif 'right' == command_input:
turn_command = 'right' #turn right
rm.resume()
elif 'no' in command_input: #stop
turn_command = 'no'
rm.pause()
elif 'automaticOff' == command_input: #Switch to fast gait
SmoothMode = 0
steadyMode = 0
rm.pause()
elif 'automatic' == command_input: #Switch to slow gait
rm.resume()
SmoothMode = 1
elif 'KD' == command_input: #Self-balancing
```

```
steadyMode = 1
rm.resume()
```

15.0 Automatic Stabilization Function

Based on the automatic stabilization function of the robot realized by MPU6050 , after starting this function, you can place the robot on a panel, and then tilt the panel. The robot will keep the body balanced by changing the height of the corresponding leg.

```
def steady_X(): #Initializing the servo in the X axis direction
if leftSide_direction:
pwm.set_pwm(0,0,pwm0+steady_X_set)
pwm.set_pwm(2,0,pwm2)
pwm.set_pwm(4,0,pwm4-steady_X_set)
else:
pwm.set_pwm(0,0,pwm0+steady_X_set)
pwm.set_pwm(2,0,pwm2)
pwm.set_pwm(4,0,pwm4-steady_X_set)
if rightSide_direction:
pwm.set_pwm(10,0,pwm10+steady_X_set)
pwm.set_pwm(8,0,pwm8)
pwm.set_pwm(6,0,pwm6-steady_X_set)
else:
pwm.set_pwm(10,0,pwm10-steady_X_set)
pwm.set_pwm(8,0,pwm8)
pwm.set_pwm(6,0,pwm6+steady_X_set)
def steady(): #Adjust the Y direction of the servo to maintain the self-balance
global X_fix_output, Y_fix_output
if mpu6050_connection:
"""
Determine whether the MPU6050 is connected '"
accelerometer_data = sensor.get_accel_data()
"""
Read the data of MPU6050
'"
78
X = accelerometer_data['x']
X = kalman_filter_X.kalman(X)
Y = accelerometer_data['y']
```

```
Y = kalman_filter_Y.kalman(Y)
"""
Carry out Kalman filtering on the data
'''
X_fix_output += -X_pid.GenOut(X - target_X)
X_fix_output = ctrl_range(X_fix_output, steady_range_Max, -steady_range_Max)
Y_fix_output += -Y_pid.GenOut(Y - target_Y)
Y_fix_output = ctrl_range(Y_fix_output, steady_range_Max, -steady_range_Max)
#LEFT_I
left_I_input = ctrl_range((X_fix_output + Y_fix_output), steady_range_Max, steady_range_Min)
left_I(0, 35, left_I_input)
#LEFT_II
left_II_input=ctrl_range((abs(X_fix_output*0.5)+Y_fix_output),steady_range_Max,steady_range_M
in)
left_II(0, 35, left_II_input)
#LEFT_III
left_III_input = ctrl_range((-X_fix_output + Y_fix_output), steady_range_Max, steady_range_Mi
n)
left_III(0, 35, left_III_input)
#RIGHT_III
right_III_input = ctrl_range((X_fix_output - Y_fix_output), steady_range_Max, steady_range_Mi
n)
right_III(0, 35, right_III_input)
#RIGHT_II
right_II_input=ctrl_range((abs(-X_fix_output*0.5)-Y_fix_output),steady_range_Max,steady_rang
e_Min)
right_II(0, 35, right_II_input)
#RIGHT_I
right_I_input = ctrl_range((-X_fix_output-Y_fix_output), steady_range_Max, steady_range_Min)
right_I(0, 35, right_I_input)
```

16.0 Image recognition using YOLO CNN :

```
from importlib import import_module
import os
```

```
from flask import Flask, render_template, Response
from camera_opencv import Camera
app = Flask(__name__)
def gen(camera):
while True:
frame = camera.get_frame()
yield (b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
@apPOSroute('/')
def video_feed():
return Response(gen(Camera()), mimetype='multipart/x-mixed-replace; boundary=frame')
if __name__ == '__main__':
apPOSrun(host='0.0.0.0', threaded=True)

import time
import threading
try:
from greenlet import getcurrent as get_ident
except ImportError:
try:
from thread import get_ident
except ImportError:
from _thread import get_ident
class CameraEvent(object): """An Event-like class that signals all active clients when a new frame is
available. """ def __init__(self):
self.events = {}
def wait(self): """Invoked from each client's thread to wait for the next frame."""
ident = get_ident()
if ident not in self.events:
# this is a new client
# add an entry for it in the self.events dict
# each entry has two elements, a threading.Event() and a timestamp
self.events[ident] = [threading.Event(), time.time()]
return self.events[ident][0].wait()
def set(self): """Invoked by the camera thread when a new frame is available.""" now = time.time()
remove = None
for ident, event in self.events.items():
if not event[0].isSet():
# if this client's event is not set, then set it
# also update the last set timestamp to now
event[0].set()
```

```
event[1] = now
else:
# if the client's event is already set, it means the client
# did not process a previous frame
# if the event stays set for more than 5 seconds, then assume
# the client is gone and remove it
if now - event[1] > 5:
remove = ident
if remove:
del self.events[remove]
def clear(self):
class BaseCamera(object):
thread = None # background thread that reads frames from camera
frame = None # current frame is stored here by background thread
last_access = 0 # time of last client access to the camera
event = CameraEvent()
def __init__(self): """Start the background camera thread if it isn't running yet."""
if BaseCamera.thread is None:
BaseCamera.last_access = time.time()
# start background frame thread
BaseCamera.thread = threading.Thread(target=self._thread)
BaseCamera.thread.start()
# wait until frames are available
while self.get_frame() is None:
time.sleep(0)
def get_frame(self): """Return the current camera frame.""" BaseCamera.last_access = time.time()
# wait for a signal from the camera thread
BaseCamera.event.wait()
BaseCamera.event.clear()
return BaseCamera.frame
@staticmethod
def frames(): """Generator that returns frames from the camera."""
raise RuntimeError('Must be implemented by subclasses.')
@classmethod
def _thread(cls): """Camera background thread."""  print('Starting camera thread.')
frames_iterator = cls.frames()
for frame in frames_iterator:
BaseCamera.frame = frame
BaseCamera.event.set() # send signal to clients
```

```
time.sleep(0)
# if there hasn't been any clients asking for frames in
# the last 10 seconds then stop the thread
if time.time() - BaseCamera.last_access > 10:
frames_iterator.close()
print('Stopping camera thread due to inactivity.')
break
BaseCamera.thread = None
```

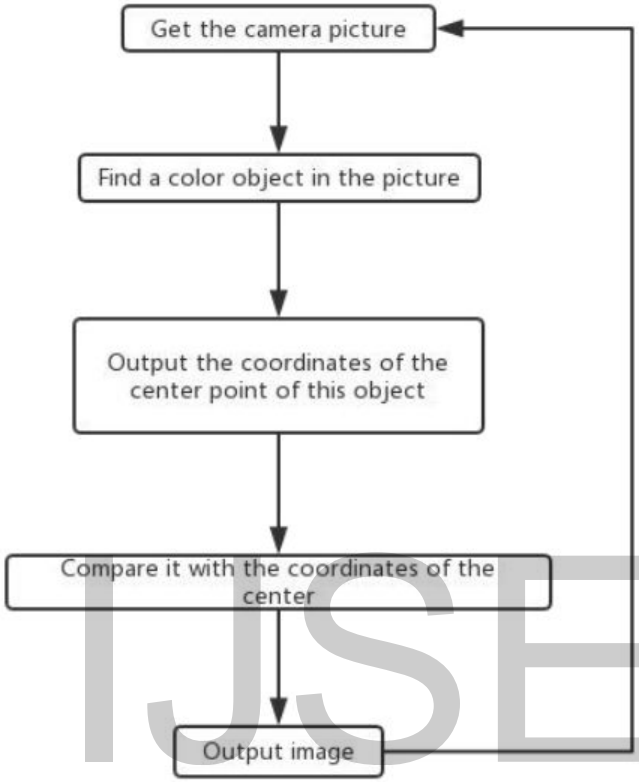
●We use OpenCV for color recognition using the HSV color space. Before introducing the code, we first need to understand the color space and why we use the HSV color space instead of the more common RGB color space for color recognition

. ●**Color space**: ·Color space is how colors are organized. With the help of color space and tests for physical devices, fixed analog and digital representations of colors can be obtained. The color space can be defined by just picking some colors at random, for example, the Pantone system just takes a specific set of colors as samples, and then defines the name and code for each color; it can also be based on a strict mathematical definition, such as Adobe RGB , SRGB.

●**RGB color space**: ·RGB uses an additive color mixing method because it describes the ratio of various "lights" to produce colors. The light is continuously superimPOSed from dark to produce color. RGB describes the values of red, green and blue light. RGBA is to add alpha channel to RGB to achieve transparency effect. Common color spaces based on RGB mode are sRGB, Adobe RGB, and Adobe Wide Gamut RGB.

●**HSV color space**: ·HSV (Hue: Hue, Saturation: Saturation, Brightness; Value), also known as HSB (B refers to Brightness) is commonly used by artists, because compared with the term of addition and subtraction, the use of hue, saturation and other concepts to describe color is more natural and intuitive. HSV is a variant of the RGB color space, its content and color scale are closely related to its source-RGB color space ·Using the HSV color space in OpenCV's color recognition function can make the recognition result more accurate, less affected by ambient light, and it is very convenient to define the color range, because what you are looking for during color recognition is not a certain color, but a certain one Range of colors, so use the HSV color space that is more in line with human eye habits for color recognition

16.0 Flow chart on how the image processing algorithm works



Set target color, HSV color space
''' colorUpper = nPOSarray([44, 255, 255])
colorLower = nPOSarray([24, 100, 100])
font = cv2.FONT_HERSHEY_SIMPLEX
class Camera(BaseCamera):
video_source = 0
def __init__(self):
if os.environ.get('OPENCV_CAMERA_SOURCE'):
Camera.set_video_source(int(os.environ['OPENCV_CAMERA_SOURCE']))
super(Camera, self).__init__()
@staticmethod
def set_video_source(source):

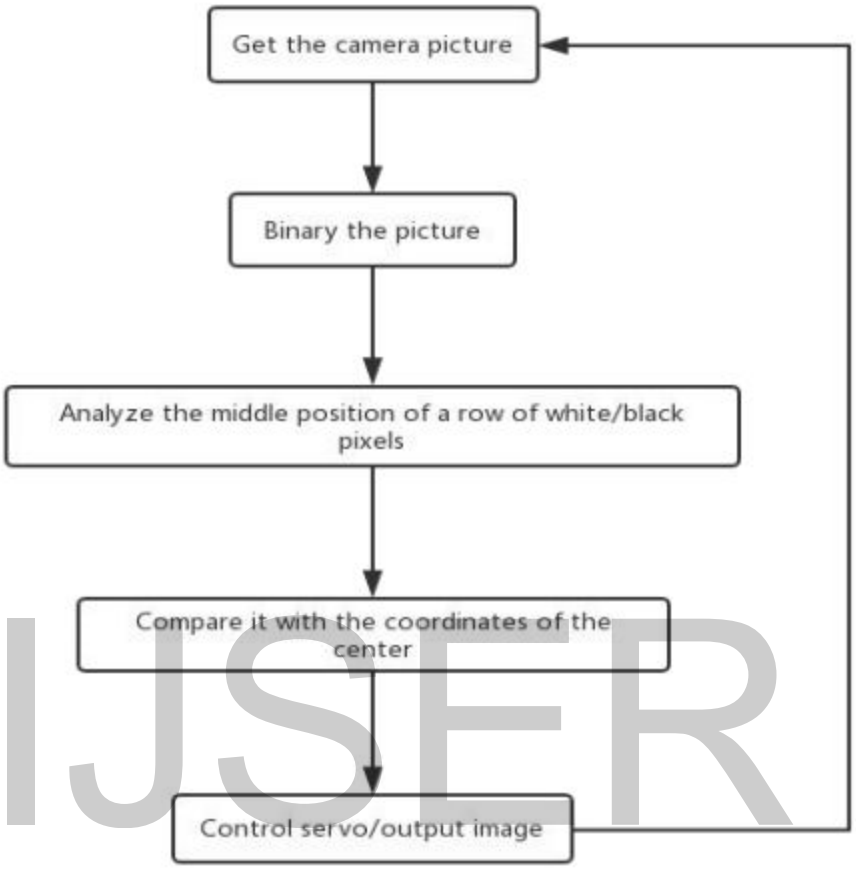
```
Camera.video_source = source
@staticmethod
def frames():
camera = cv2.VideoCapture(Camera.video_source)
if not camera.isOpened():
raise RuntimeError('Could not start camera.')
while True:
# read current frame
img = camera.read() #Get the picture captured by the camera
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV) #Convert captured images to HSV color space
mask = cv2.inRange(hsv, colorLower, colorUpper) #Traverse the colors in the target color range in the HSV
color space, and turn these color blocks into masks
mask = cv2.erode(mask, None, iterations=2) #Corrosion of small pieces of mask (noise) in the picture
becomes small (small pieces of color or noise disappear)
mask = cv2.dilate(mask, None, iterations=2) #Inflate, and resize the large mask that was reduced in the
previous step to its original size
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2] #Find a few masks in the picture
center = None
if len(cnts) > 0: #If the number of whole masks in the picture is greater than one
''' Find the coordinates of the center point of the object of the target color and the size of the object in
the picture
''' c = max(cnts, key=cv2.contourArea)
((box_x, box_y), radius) = cv2.minEnclosingCircle(c)
M = cv2.moments(c)
center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
X = int(box_x)
Y = int(box_y)
''' Get the center point coordinates of the target color object and output
''' print('Target color object detected')
print('X:%d'%X)
print('Y:%d'%Y)
print('-------')
''' Write text on the screen：Target Detected
''' cv2.putText(img,'Target Detected',(40,60), font, 0.5,(255,255,255),1,cv2.LINE_AA)
```

''' Draw a frame around the target color object
''' cv2.rectangle(img,(int(box_x-radius),int(box_y+radius)),
(int(box_x+radius),int(box_y-radius)),(255,255,255),1)
else:
cv2.putText(img,'Target Detecting',(40,60), font, 0.5,(255,255,255),1,cv2.LINE_AA)
print('No target color object detected')
# encode as a jpeg image and return it

HSV\color	Black	Grey	White	Red	Orange	Yellow	Green	Cyan	Blue	Purple
H_min	0	0	0	0|156	11	26	35	78	100	125
H_max	180	180	180	10|180	25	34	77	99	124	155
S_min	0	0	0	43	43	43	43	43	43	43
S_max	255	43	30	255	255	255	255	255	255	255
V_min	0	46	221	46	46	46	46	46	46	46
V_max	46	220	255	255	255	255	255	255	255	255

yield cv2.imencode('.jpg', img)[1].tobytes()

17.0 Flow chart on the ML algorithm efficacy




```
import os
import cv2
from base_camera import BaseCamera
import numpy as np
import time
import threading
import imutils
''' Set the color of the line, 255 is the white line, 0 is the black line
''' lineColorSet = 255
''' Set the horizontal POSition of the reference, the larger the value, the lower, but not greater than the vertical resolution
of the video (default 480)
''' linePOS = 380
```

```
class Camera(BaseCamera):
video_source = 0
def __init__(self):
if os.environ.get('OPENCV_CAMERA_SOURCE'):
Camera.set_video_source(int(os.environ['OPENCV_CAMERA_SOURCE']))
super(Camera, self).__init__()
@staticmethod
def set_video_source(source):
Camera.video_source = source
@staticmethod
def frames():
camera = cv2.VideoCapture(Camera.video_source)
if not camera.isOpened():
raise RuntimeError('Could not start camera.')
while True:
img = camera.read() #Get the picture captured by the camera
'''
Convert the picture to black and white, and then binarize (the value of each pixel in the picture is
255
except 0)
''' img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
retval, img = cv2.threshold(img, 0, 255, cv2.THRESH_OTSU)
img = cv2.erode(img, None, iterations=6) #Use Corrosion Denoising
colorPOS = img[linePOS] #Get an array of pixel values for linePOS
try:
lineColorCount_POS = nPOSsum(colorPOS == lineColorSet) #Get the number of pixels of line
color
(line width)
lineIndex_POS = nPOSwhere(colorPOS == lineColorSet) #Get the horizontal POSition of the
end
point of the line in the linePOS line
''' Use the endpoint POSition and line width to calculate the POSition of the center point of the
line
''' left_POS = lineIndex_POS[0][lineColorCount_POS-1]
right_POS = lineIndex_POS[0][0]
center_POS = int((left_POS+right_POS)/2)
print('The POSition of the center point of the line is：%d'%center_POS)
except:
''' If no line is detected, the line width above 0 will cause an error, so that you know that no line
has
```

```
been detected
''' center_POS = 0
print('No line detected')
''' Draw a horizontal reference line
''' cv2.line(img,(0,linePOS),(640,linePOS),(255,255,64),1)
if center_POS:
''' If a line is detected, draw the center point of the line
''' cv2.line(img,(center_POS,linePOS+300),(center_POS,linePOS-300),(255,255,64),1)
# encode as a jpeg image and return it
yield cv2.imencode('.jpg', img)[1].tobytes()
```

18.0 Result .

The project CEEFRR is by every means a best alternative to the present existing system of robots or human intervention for the cave rescue and would revolutionize the way of rescue operations carried out in subterranean cave system with complex geography and frigid environment .

1. Hexapod systems can be used to perform complex 21 DOF tasks in an energy efficient manner
2. Hexapod systems can very well be deployed in subterranean environments.
3. The body is less abrasion resistant, we need to be more careful about its deployment .
4. The hexapod is statically stable .

19.0 Reference

1.Meyrowitz, A. Let al. , "Autonomous vehicles, " Proceedings of the IEEE, vol. 84, no. 8, Aug 1996pp. 1147-1164 doi: 10. 1109/5. 533960J.

2.Poole, Harry H. , "Types of Robots" in Fundamentals of Robotics Engineering, 1st ed. Netherlands: Springer, 1989, pp. 27-51.

3.Brooks, Rodney A. "A robot that walks; emergent behaviors from a carefully evolved network" in Neural computation, vol. 1, no. 2, Cambridge: MIT Press, 1989, pp. 253-262 doi: 10. 1162/neco. 1989. 1. 2. 253

4.Simpson, Mark M. , and Tim Jordanides. "Genghis: An Intelligent Autonomous Mobile Robot" inExpert Systems and Robotics, Berlin Heidelberg: Springer, 1991, pp. 663-672.

5.Hardarson, Freyr. "Locomotion for difficult terrain" in A Survey Study, Dept. Mach. Des. , Royal Inst. Technol. , Stockholm, Sweden, 1998. FLEXChip Signal Processor (MC68175/D), Motorola, 1996.

6.Ding, Xilun, et al. "Locomotion analysis of hexapod robot" inClimbing and Walking Robots, 2010, pp. 291-310.

7.GaoJianhua, "Design and Kinematic Simulation for Six-DOF Leg Mechanism of Hexapod Robot, " Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference, pp. 625, 629, 17-20 Dec. 2006 doi: 10. 1109/ROBIO. 2006. 340272

8.Wettergreen, David, and Chuck Thorpe. "Gait generation for legged robots, "IEEE International Conference on Intelligent Robots and Systems. 1992.

9.Saranli, Ulucet al. , "RHex: A simple and highly mobile hexapod robot, " The I+++++++++++++++nternational Journal of Robotics Research, vol. 20, no. 7, 2001, pp. 616-631.
10. Lewis, M. Anthony et al. , "Genetic algorithms for gait synthesis in a hexapod robot" inRecent trends in mobile robots, vol 11, Singapore: World Scientific Publishing Co. , 1994, pp. 317-331.
11. Askeland, Donald and Wendelin Wright, "Polymers" in Essentials of Materials Science & Engineering, 3rd ed. USA: Cengage Learning, 2014, pg. 546
12. Carter, G. F. et al. , "Nonferrous Alloys" in Material science and Engineering, USA: ASM International, 2011, pg. 127
13. Mullen, J. B. , "Speed Torque" in Popular Mechanics, vol. 106, no. 4, pp245-248, Oct. 1956.