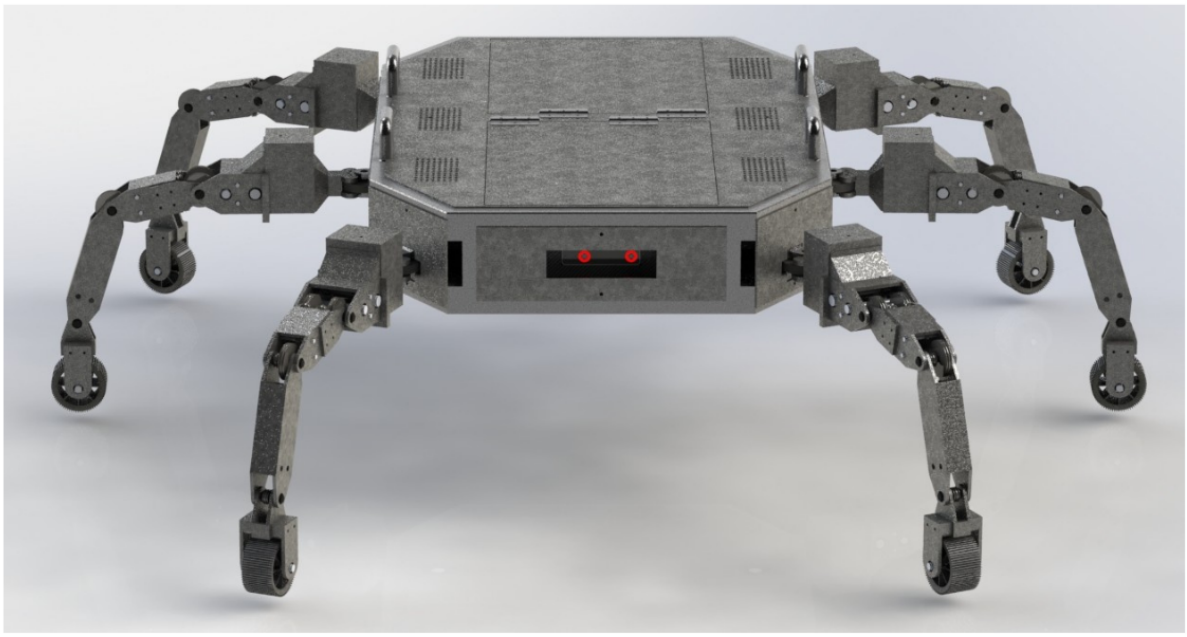
TACTICAL UNMANNED GROUND VEHICLE FOR CLOSE-QUARTERS SURVEILLANCE & COMBAT



Aman Malekar (180100012)
Barath Krishna S. (180100027)
Kritti Sharma (180100060)
Mitalee R. Oza (180100067)
Shubham Agrawal (180041000)
Rishab Khantwal (180100095)
Kratik Bhadoriya(180100059)

Contents

1 Introduction

1.1 Motivation & Problem Statement

During the 2nd Counter-Terrorism week, India's representative, Mr. T. S. Tirumurti, quoted to the United Nations that India has been a victim of terrorism, especially from across the border, over the last several decades[1]. Despite of the Line of Control barrier, which consists of double-row of fencing and concertina electrified wire 8–12 ft in height which is also connected to a network of motion sensors, thermal imaging devices, lighting systems and alarms with thousands of landmines, the incidents of infiltrators trying to sneak in are not uncommon. During the conduct of Counter-Insurgency (CI)/ Counter-Terrorist (CT) operations, accurate and real time inputs are critical for success. For the desired high degree of situational awareness and effect on targets, manual methods of surveillance and combat are being employed as of now in India. This results in collateral damage as well as casualties of our own soldiers. In the past 30 years, 6,500+ soldiers were martyred in terrorist attacks[2].

We aimed at developing a tactical Unmanned Ground Vehicle (UGV) for **surveillance** for accurate real time inputs and obviating the inherent risks to soldiers. The solution to this problem statement will be extremely useful for Infantry, RR and Special Forces Battalions.

As was quoted in an executive summary, dismounted ground combat troops carry 40 to 65 kg or more in combat. Heavy loads reduce mobility, increase fatigue, and reduce mission performance[3]. The development of improvised armours, usually make the job of soldiers more difficult due to increased payload. As an engineer, we find it our moral responsibility to come up with a **compact design solution** which is optimised for both, the tasks at hand and payload for soldiers. We aim to achieve an **optimized payload** of $\sim$ 40 kg. India shares its borders with Bangladesh, Bhutan, China, Myanmar, Nepal, Pakistan and Sri Lanka. India-Pakistan border includes desert, marshy-muddy land, valleys, snowy mountains, rocky mountains, glaciers and wild grassfield. We aim at **maneuvering in difficult terrains**.

Particularly, we aimed at achieving the following design goals:

- Carry payloads upto $\sim$ 40 kg apart from the weight of the robot (We have been able to achieve about $\sim$ **70** kg payload capacity after several iterations to our design)
- Ability to maneuver in difficult unstructured terrains
- Surveillance of surrounding region with 360 degree monitoring
- Rescue aid in case of hostages
- Ability to do 2-way communication for negotiations
- Point and Shoot with Friend and Foe distinction
- Enemy detection with camouflage detection

Figure 1: Indian Soldiers employed for surveillance of India-Pakistan (left) and India-China Borders (right), day in - day out.

[1] https://www.indiatoday.in/world/story/cross-border-terrorism-countries-safe-havens-india-un-1819660-2021-06-26
[2] https://www.satp.org/satporgtp/countries/india/states/jandk/data$_s$*heets/annual_casualties.htm*
[3] https://www.cnas.org/publications/reports/the-soldiers-heavy-load-1

1.2 Cost Analysis

The solution was aimed at saving our soldiers from potential threats and minimising the cost of the proposed solution at the same time. A constraint faced by the army right now is the budget limit for recruiting more soldiers and expanding the army. The cost distribution described below clearly shows that the solution achieved a lower annual cost than the the average salary of a soldier. The robot also aims to carry out certain tasks which pose an unnecessary threat to the lives of soldiers.

Annual cost of a Soldier ~ Rs. 20,80,000

Domain	Cost
Salary of one soldier	Rs. 15,00,000
Operations & Maintenance	Rs. 2,00,000
Procurement (Arms, Equipments, etc)	Rs. 1,00,000
Research, Development, Test & Evaluation	Rs. 50,000
Revolving & Management Funds	Rs. 50,000
Military Construction	Rs. 1,00,000
Family Housing	Rs. 1,20,000
Base, OCO and Emergency	Rs. 2,00,000
Defense Health and Education Program	Rs. 30,000
Commissary Subsidy	Rs. 30,000

One time cost of UGV ~ Rs. 8,00,000

Domain	Cost
Actuator (Servos for Joints)	Rs. 4,50,000
Motors, Wheels, Bearing & Nuts	Rs. 60,000
Materials	Rs. 30,000
Manufacturing	Rs. 1,00,000
Sensors & Computations Units	Rs. 1,00,000
Battery	Rs 30,000

Annual cost of UGV ~ Rs. 14,15,000

Kevlar & Battery coating, bearing wears	Rs. 15,000
Operator Salary & Station Costs	Rs. 14,00,000

2 Mechanical Design

2.1 Chassis Design

Chassis is the home to the majority of the features and actuation mechanisms deployed on the hexapod. The highlighting features of the hexapod are described as below:

- **Smaller Modular Bot**: It is a four-wheeled differential drive modular unit capable of operating independently when detached from the hexapod. It includes the entire sensor stack and the computational unit required for the smooth working of the hexapod. This feature can be handy while approaching compact spaces. This unit can further be deployed with payloads (like explosives, etc.) as per the mission objectives.
- **Closed Storage Unit (Medical + Ammunition)**: This compartment is designed for storing first aid kits, ammunition and other smaller substances in a similar size range.
- **Payload Platform (Strap on Unit)**: The top platform is for storing weapons payloads like guns, rocket launchers, etc. This platform is accompanied by strap hooks for securing the payload with the platform.
- **Motors with Tensioners**: This subassembly helps in actuating all the leg joints without transmitting the non-axial transmission loads to the motor shafts.
- **Exhaust**: The exhaust system includes four exhaust fans and corresponding vents for the smooth airflow over batteries. The vents are placed on the top platform and the fans on the bottom, creating a cooling air current from top to bottom.
- **Outer Shell (Kevlar Skin)**: The outer shell of the hexapod is built with Kevlar, enabling the hexapod to work in severe conditions, keeping the internals of the pod safe.

2.2 Leg Design

The leg design is inspired by the notion of a Low Inertia Manipulator with High Stiffness and Strength. Essentially the leg is designed in such a way that all the joint motors are fixed at the chassis and the load transmission is done using pulleys and wires. This enables us to minimise leg inertia while keeping the operational torque the same. The leg design has two components: Link length optimisation and Transmission mechanism. The description is as follows:

- **Link Length Optimisation**: Link lengths are optimised for maximising speed and nominal height, minimising size and average operating torque. The shape of the links is optimised for minimising weight and potential failure instances.
- **Transmission**: The transmission method involves multiple strings and pulley systems for the actuation of the joints using motors mounted at the base. This transmission method effectively helps us to minimise the inertia of the links, hence reducing the motors' operating torques. The larger joint pulleys provide an additional mechanical advantage, further decreasing the torque requirements. Further, we used knuckle joints and a belt drive with a tensioner to transfer power. This was done to make sure we don't strain the servo shaft.

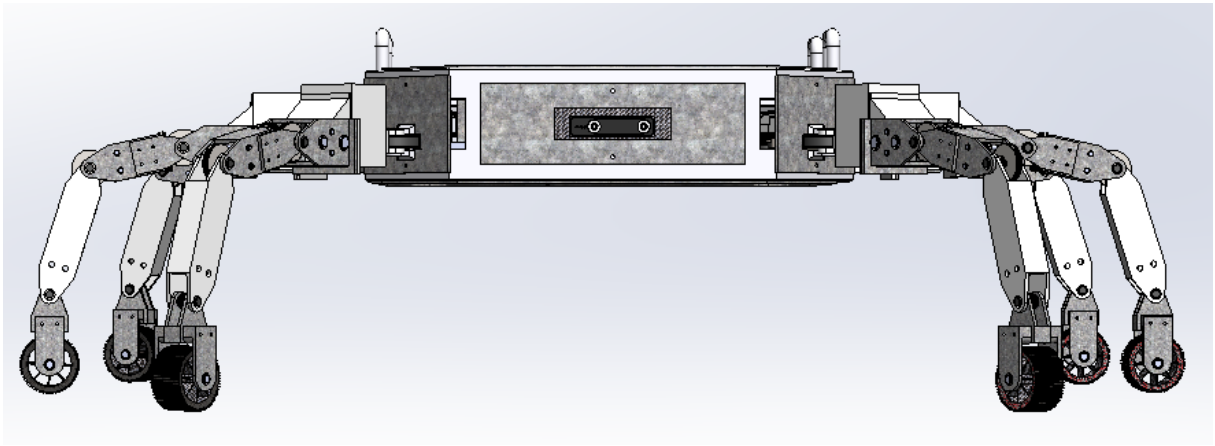


Figure 2: Front view of the Hexapod Robot.

2.3 Specifications & Renders

CHASSIS	Description				
	Length(mm)	Breadth(mm)	Height(mm)	Specifications	Count
Smaller Modular Bot	320	270	80	-	1
Closed Storage Unit	400	320	100	-	1
Payload Platform	946	646	-	-	1
Joint Motors (A1, A2)	-	-	-	DYNAMIXEL MX-64T	12
Joint Motors (A3, A4)	-	-	-	DYNAMIXEL MX-28AR	12
Hexapod Wheel Motors	-	-	-	150 RPM High Torque 12 Volt DC Gear Motor 32 kg-cm torque	6
Exhaust fan	40	40		-	4
Camera	-	-	-	Intel Realsense D435	1
Communication Unit	-	-	-	XBee 3 Pro Module - RP-SMA Antenna	1
Computational Unit	-	-	-	Jetson AGX Xavier	1

LEG	Description				
	Length(mm)	Breadth(mm)	Height(mm)	Specifications	Count
Link 1	237	-	-	-	6
Link 2	130	-	-	-	6
Link 3	180	-	-	-	6
Link 4	95	-	-	-	6
Transmission wires		(d =) 1			24

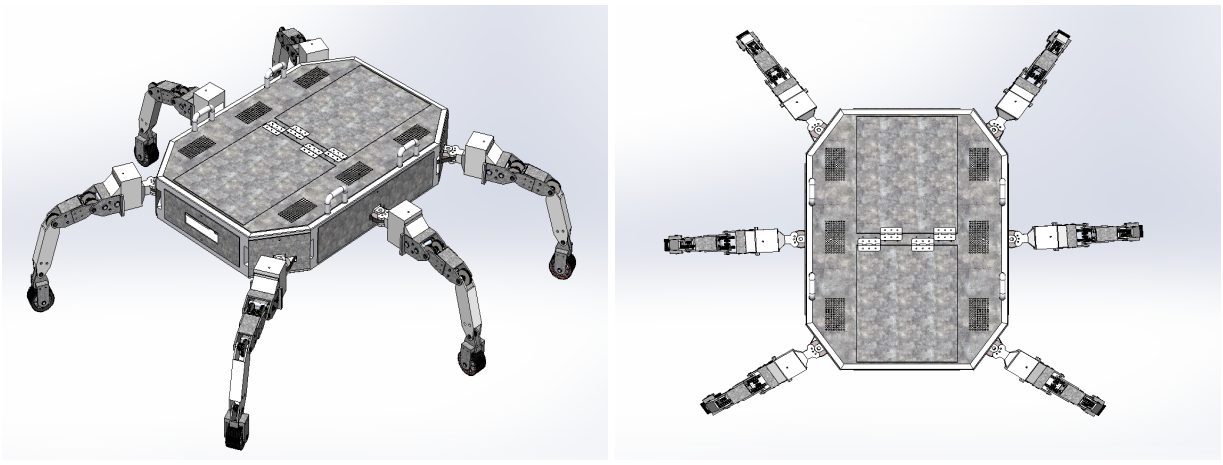


Figure 3: Perspective view and top view of the bot

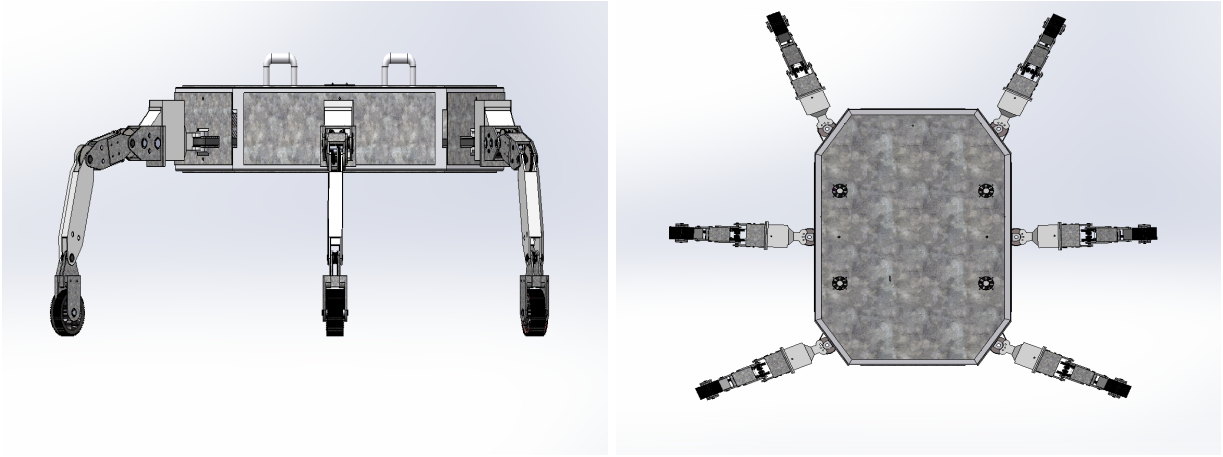


Figure 4: Side view and bottom view of the bot

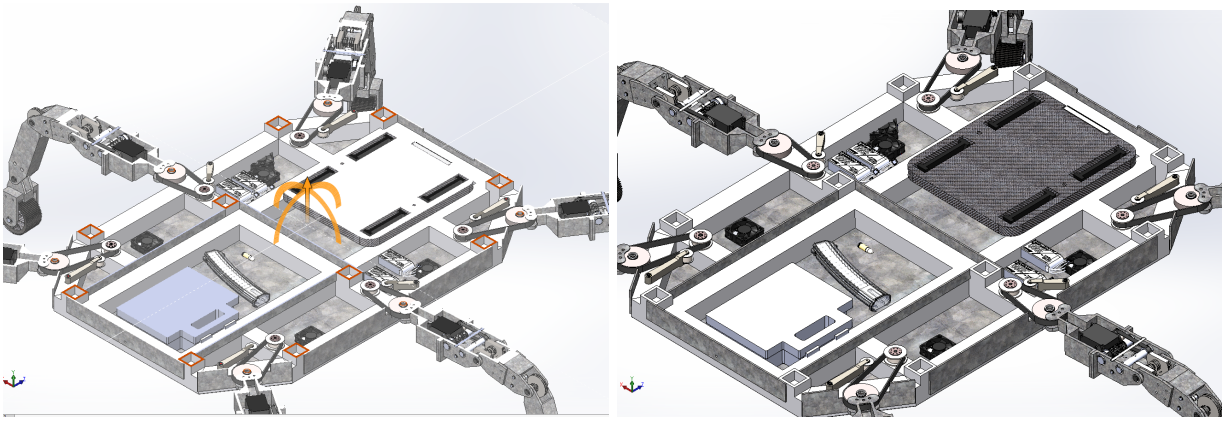


Figure 5: Motor with Tensioner

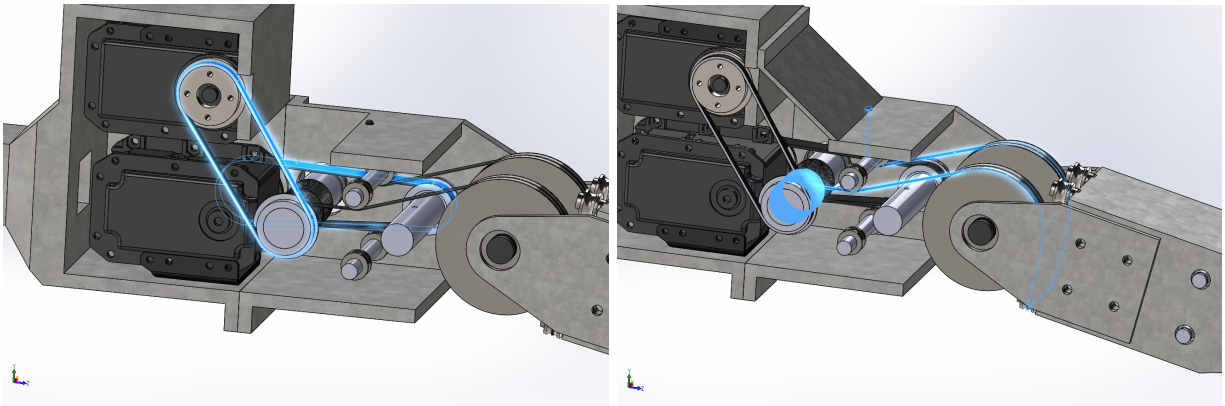


Figure 6: Transmission a2 belt and string

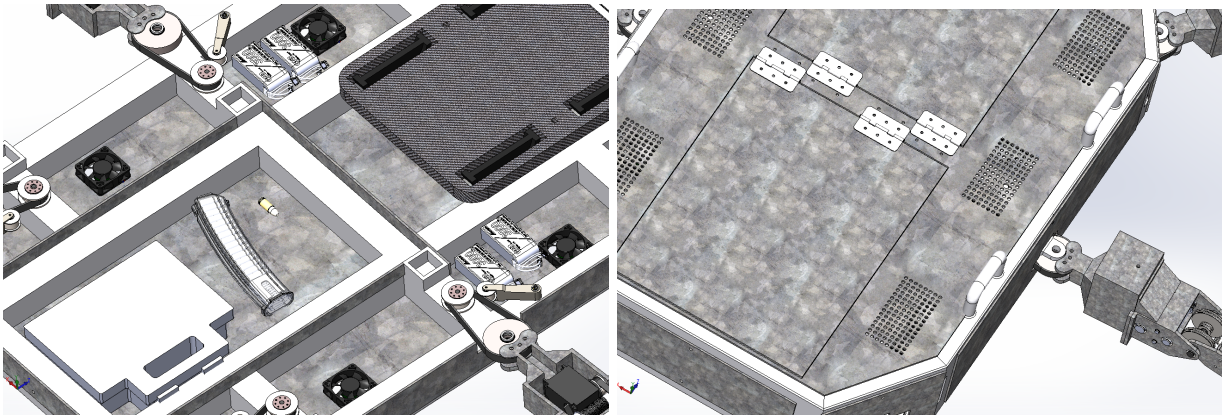


Figure 7: Exhaust Fans and Vents

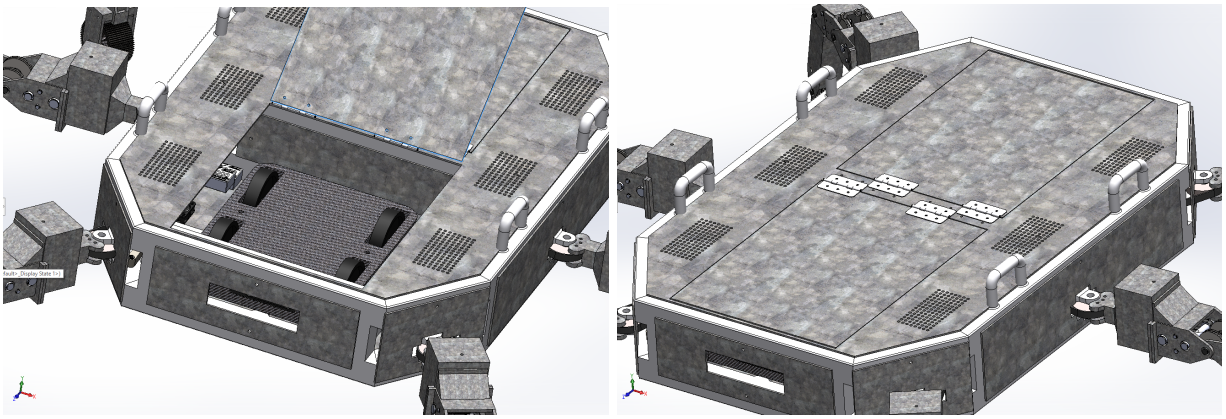


Figure 8: Modular Bot and Payload Platform

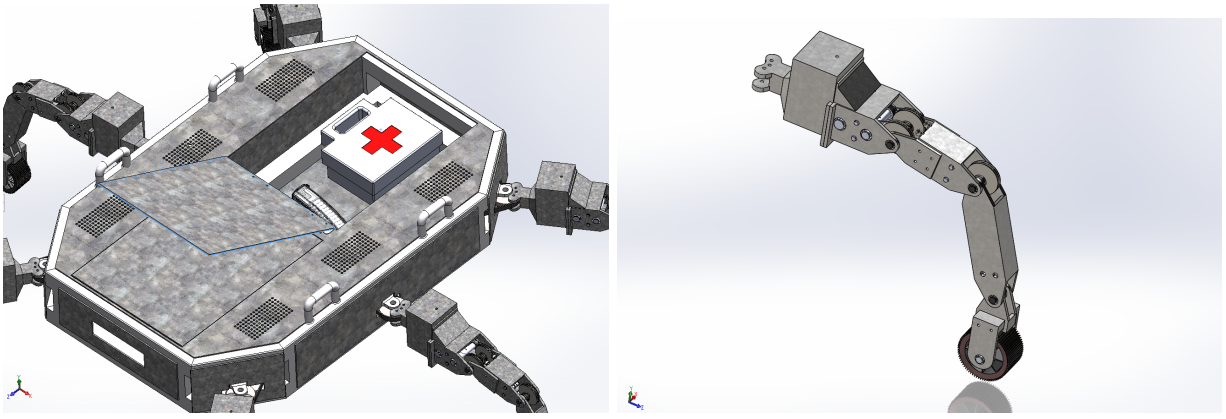


Figure 9: Closed storage and leg

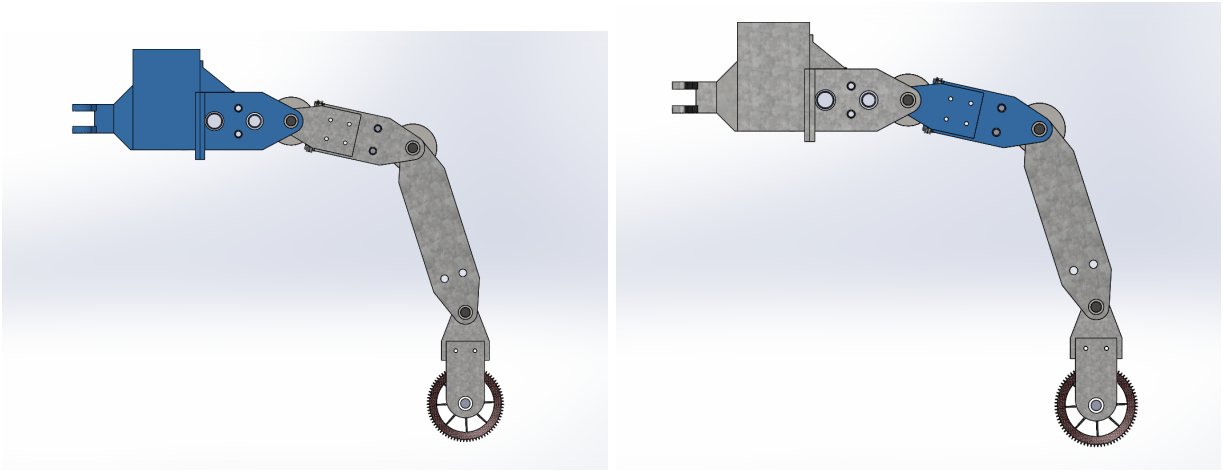


Figure 10: Link 1 and Link 2

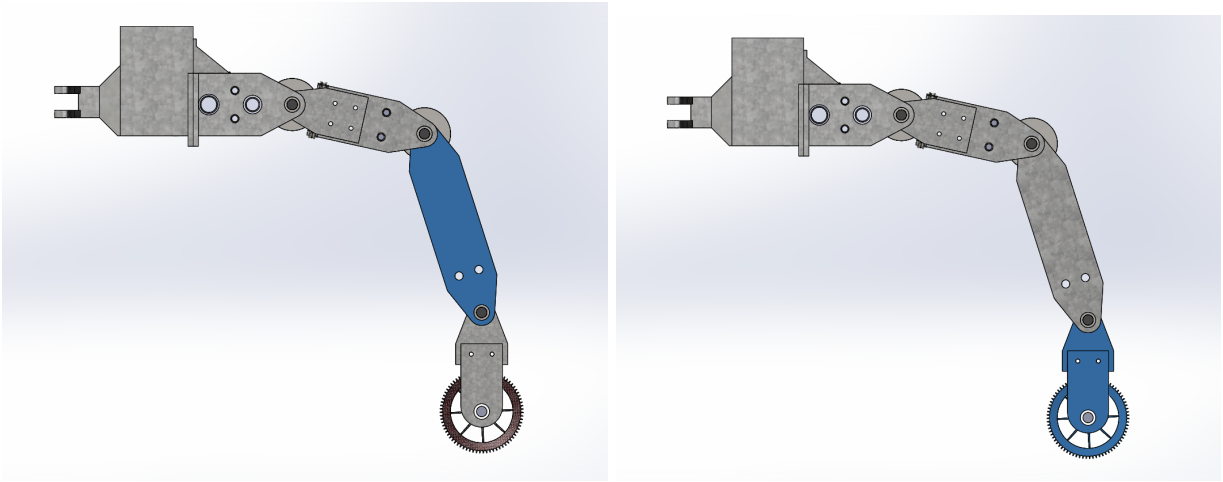


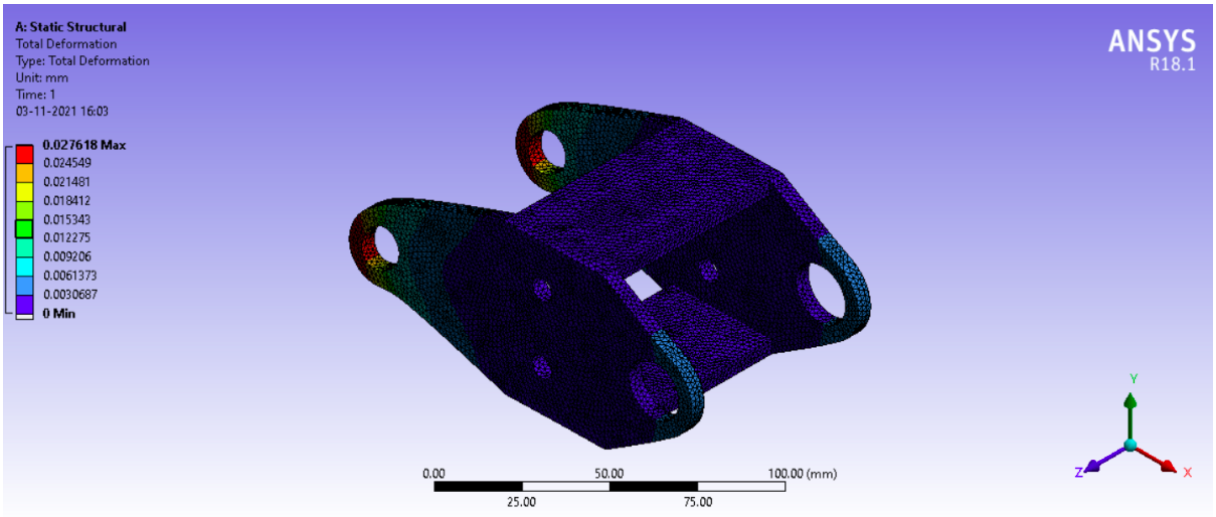
Figure 11: Link 3 and Link 4

2.4 Finite Element Analysis

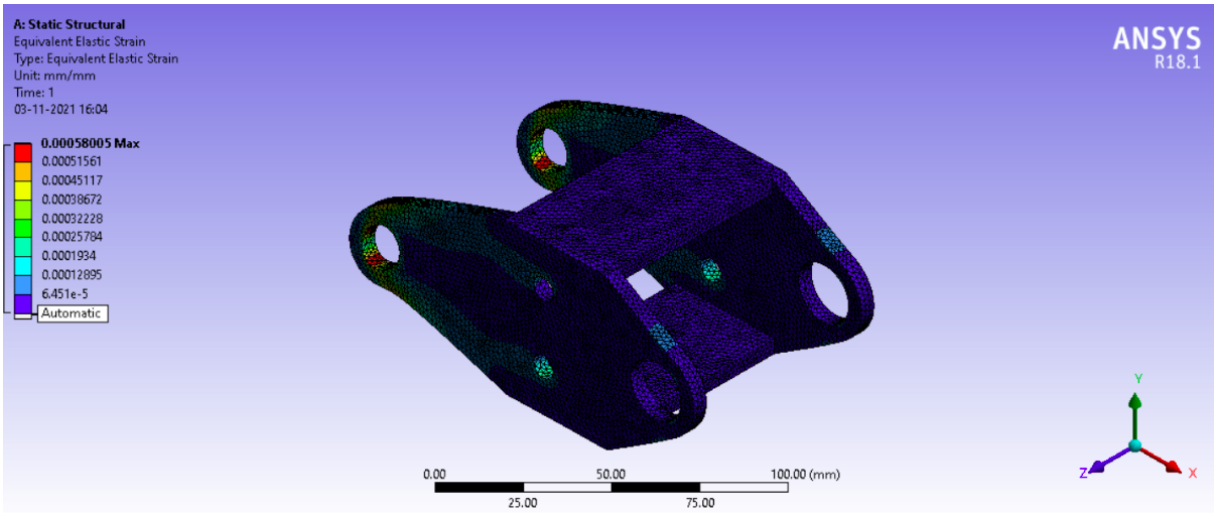
2.4.1 Analysis on the Link 1

Link 1 connects joints 1 and 2. Thus at the maximum load condition, it experiences torques of 8.624 Nm and 32.558 Nm at the respective joints. The analysis below is performed at this maximum load condition. The material used for this link is aluminium.

Deformation: The maximum observed value of deformation is 0.0276 mm, which is quite insignificant.



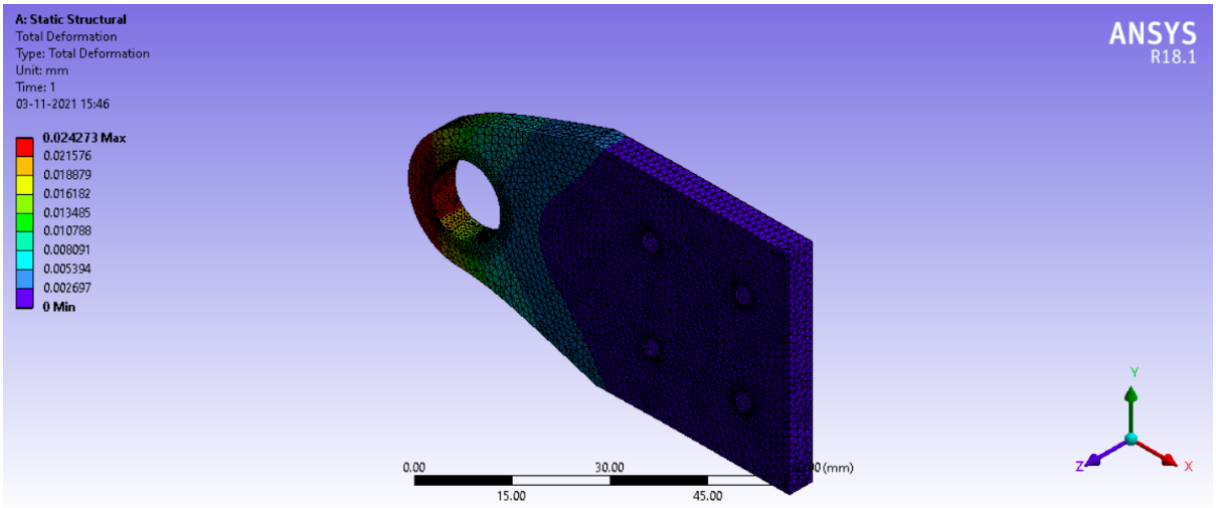
Strain: The maximum observed value of strain is $5.8e-4$, which is well within the elastic limits of aluminium.



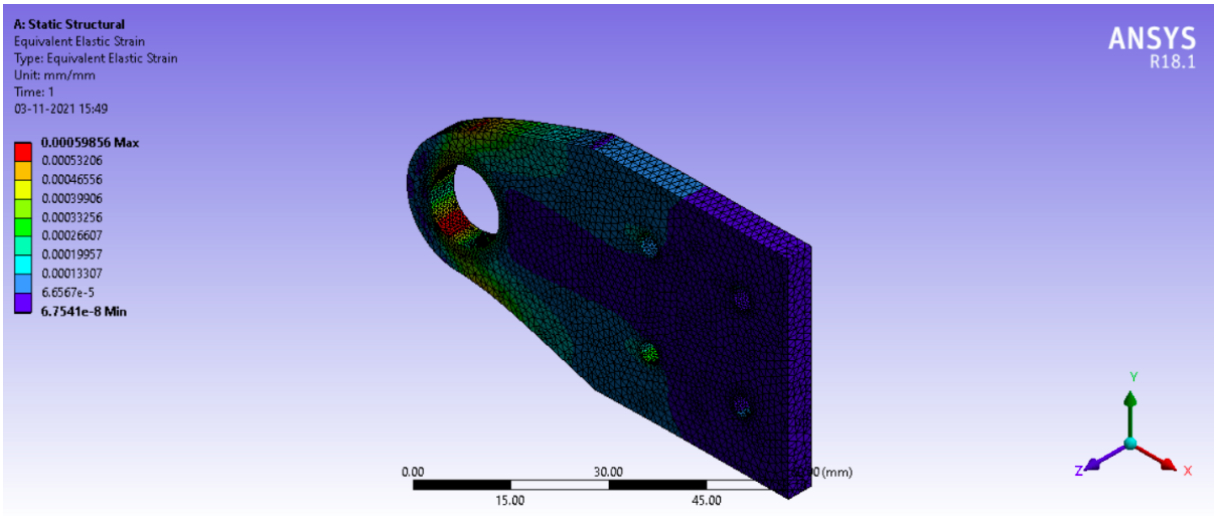
The safety factor was observed to be well above 1 for all regions in the part.

2.4.2 Analysis on the connector between Joint 2 and Link 2

This part connects link 2 to joint 2. A connector is used instead of directly attaching the link to the joint, since such a connector allows for easy maintenance and service of the motor that is placed inside. The connector is fixed to link 2 at one end and a maximum torque of 32.558Nm is experienced at the other end. The material used is aluminium.
Deformation: Maximum value of deformation = 0.024 mm



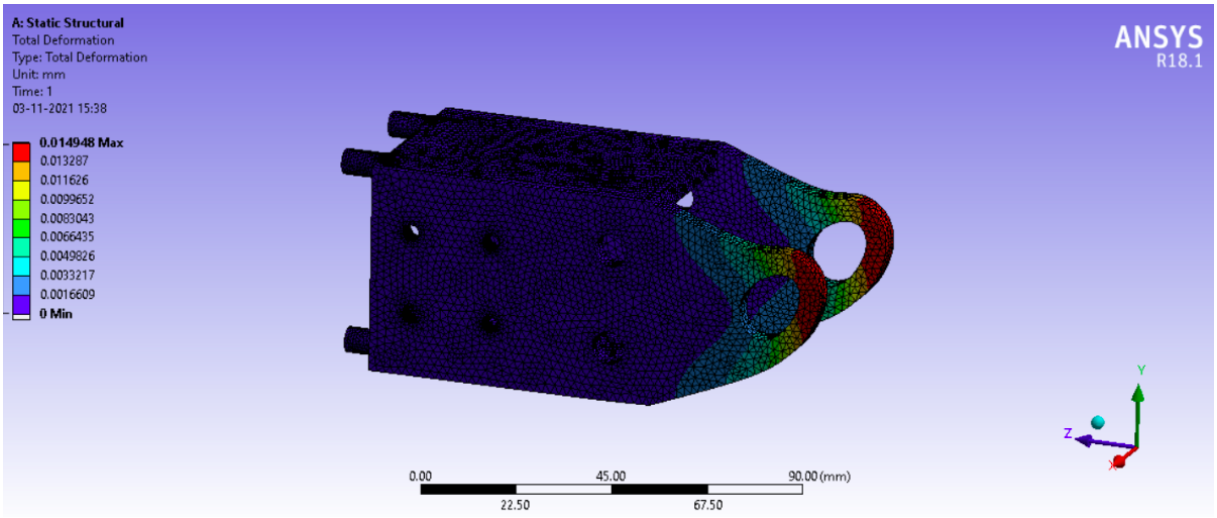
Strain: Maximum value of strain = $5.9856e-4$



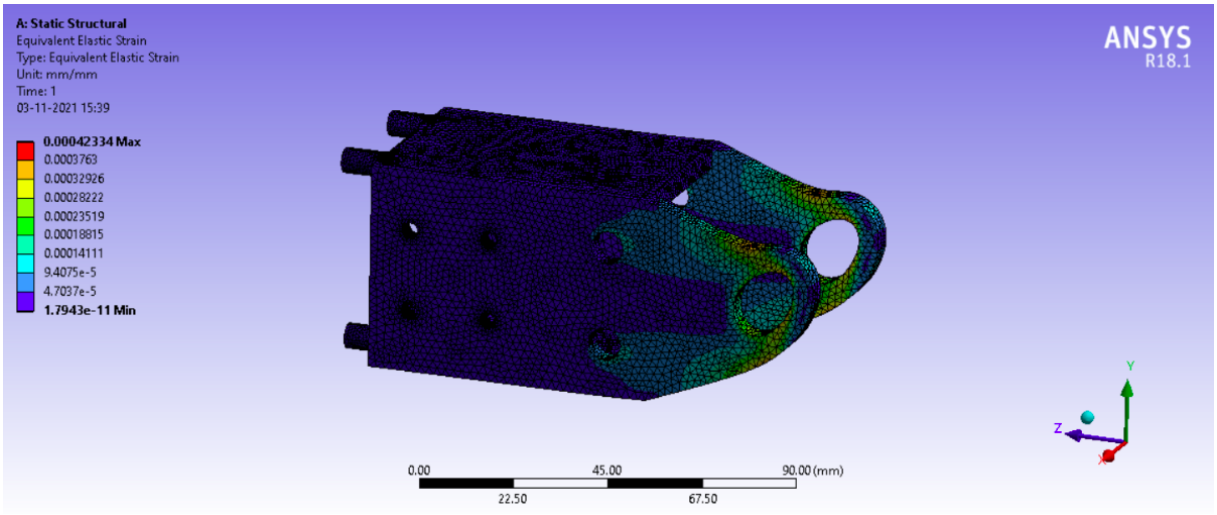
These conditions are well within the elastic limits of aluminium. Also, sufficient safety factors were observed at all regions of the part. Thus, aluminium is a suitable material for this part.

2.4.3 Analysis on Link 2

Link 2 in combination with the connector shown above houses the motor that drives joint 3. Link 2 is fixed to the connector at one end and is attached to joint 3 at the other end. Consequently, at maximum load condition, link 2 experiences a torque of 24.4156 Nm at its attachment point to joint 3. The material used is aluminium
Deformation: The maximum deformation observed in the part is 0.01495 mm.



Strain: The maximum value of strain observed is $4.233e-4$.

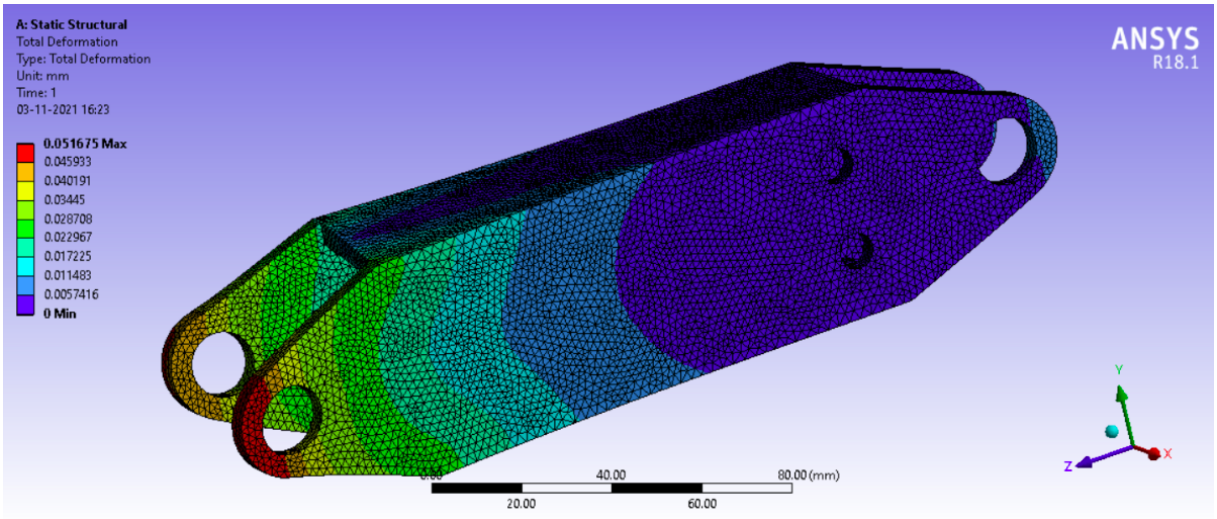


Since the values of deformation and strain are well within the elastic limits of aluminium, it is a suitable material for this part. Note that the safety factor was well above the requirements for all the regions in the part.

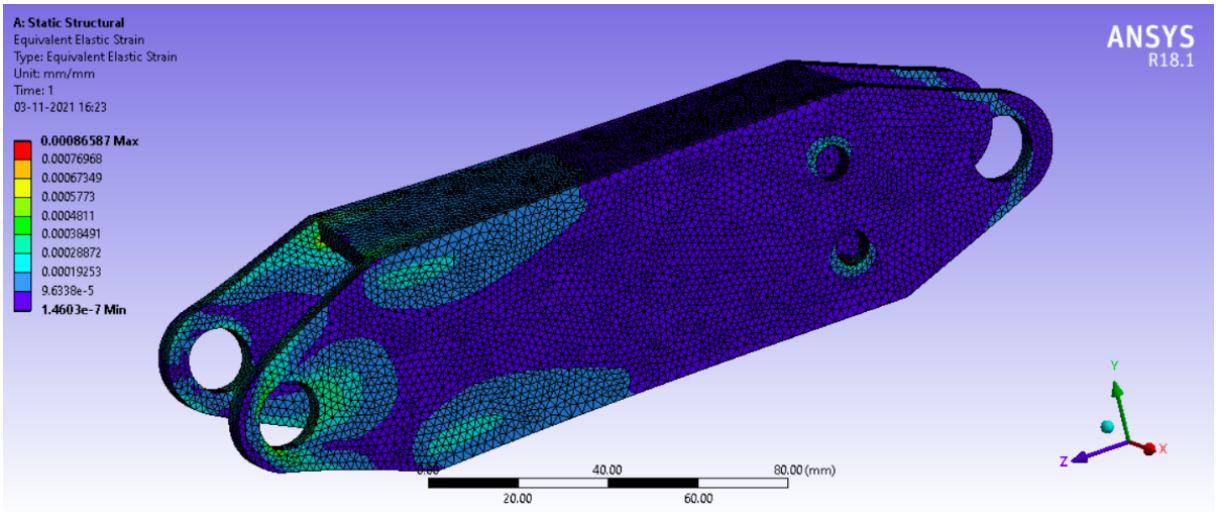
2.4.4 Analysis on Link 3

Link 3 connects joints 3 and 4, and experiences torques of 24.4156 Nm and 10 Nm at the respective attachment points at maximum load condition. Thus, the deformation, strain and safety factor were observed for the part with aluminium being selected as the material.

Deformation: The maximum observed deformation is 0.0517 mm.



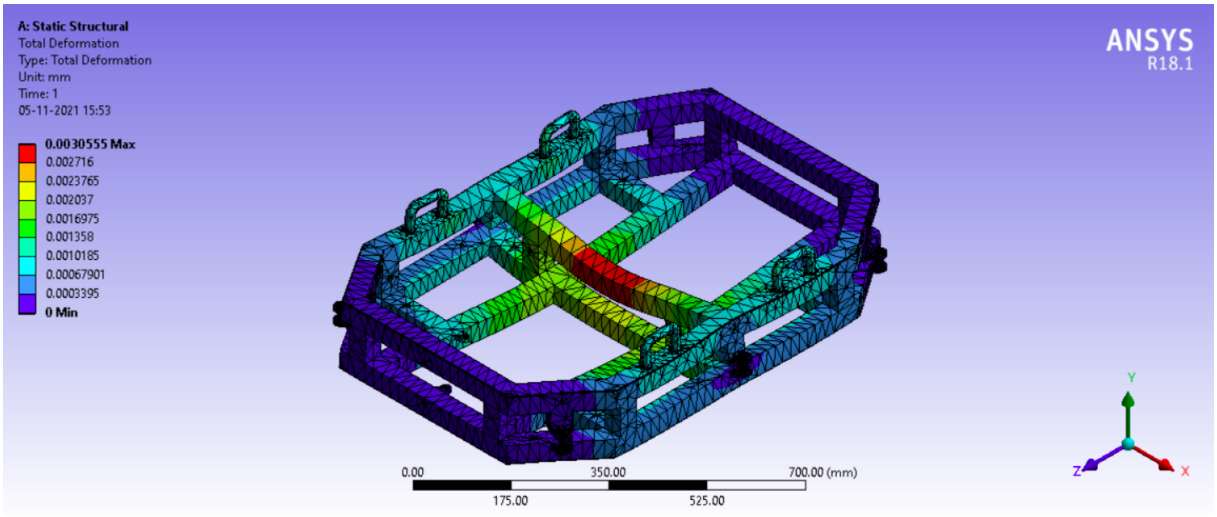
Strain: The maximum observed value of strain is $8.66e-4$.



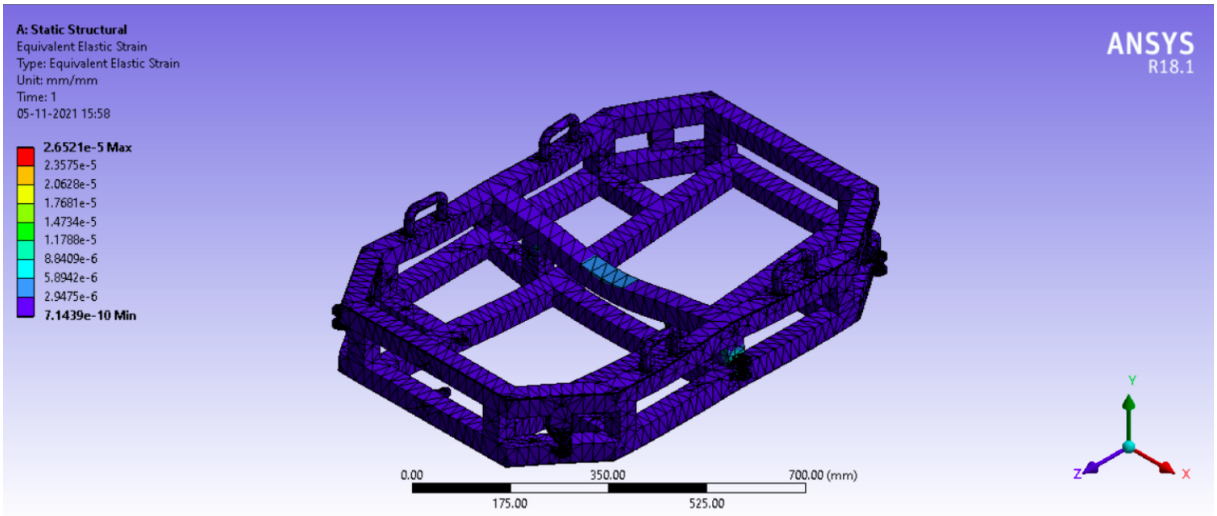
There was also sufficient safety factor at all regions of the part. Due to this reason and the fact that the deformation and strain values are well within the elastic limits of aluminium, the material was finalised as aluminium.

2.4.5 Analysis on Chassis

The chassis is the most important structural component of the bot, acting as the skeleton of the bot and housing multiple different components. The major loads on the chassis are the vertical load due to the expected payload of $\sim 60\ kg$ and the torques at the 6 joints. The material used for the simulation is aluminium.
Deformation: The maximum deformation comes out to be 0.0306 mm.



Strain: The maximum observed value of strain is $2.652e-5$.



The chassis being an integral structural component, needs sufficient safety factor at all the regions. This is due to the fact that it might experience impulses or jerks that have not been accounted for. Aluminium ensures that all these requirements are met, and was thus finalised as the material for the chassis.

2.4.6 Topology Optimization

Since there is sufficient safety factor at many regions of the parts shown, there is a big potential to reduce the weight of the bot using topology optimization. The underlying concept here is that material can be removed from areas of the part where there is no significant load or strain. Thus, this helps in reducing the weight of the part without compromising the strength or load bearing capacity.

2.5 Failure Modes and Effects Analysis

Process step	Potential failure mode	Potential failure effects	SEVERITY	Potential causes	OCCURRENCE	Current process controls	DET	RPN	Actions recommended	Actions taken	NSewV	NOecC	NDeET	NRePN
What is the step?	In what ways can the step go wrong?	What is the impact on the customer if the failure mode is not prevented or corrected?	10	What causes the step to go wrong? (i.e., How could the failure mode occur?)	10	What are the existing controls that either prevent the failure mode from occurring or detect it should it occur?	10	1000	What are the actions for reducing the occurrence of the cause or for improving its detection? You should provide actions on all high RPNs and on severity ratings of 9 or 10.	What were the actions implemented? Include completion month/year (then recalculate resulting RPN).	10	10	10	1000
Tire Tracks	The tracks can get damaged overtime	Restricted Movement	8	Rugged Terrain and Heavy Loads	3	NIL	8	192	Using rigid tire tracks instead of tubed tires.	Tubed tires were replaced by rigid tire tracks	6	3	5	90
Actuators	Current Overload and Heating	Overheating can cause malfunction	7	High Temperature Weather and Obstacles in motion under the influence high payload	3	Detecting Current Surges and accordingly commanding controller to stop the motion momentarily	4	84	Incorporating the safety features into the algorithm	Exhaust fans installed near the batteries and actuators	5	2	3	30
Base Plate	Development of cracks	The bot can disintegrate	9	Impulse imparted by heavy payload	2	Putting higher tolerance limit on the base plate payload carrying capacity and using a padding to reduce damage due to impulses	6	108	FEA based on study of the loads through experimental data gathered from the field	FEA completed, safety factors and design changes were incorporated accordingly	6	2	4	48
Link 2 - Link 3 Movement	Links can break at the joint	The bot will remain dysfunctional, will affect the battle strategy	9	Under high payload carrying cycles, the breakage can happen	5	Detection of preliminary cracks through thorough inspection	6	270	Better Material used to ensure longer cycles, but at increased cost	Material selection and link cross-section updated based on FEA results	6	4	4	96
Joint 2 Movement	Transmission wires can break	The bot will remain dysfunctional, will affect the battle strategy	9	Continuously varying high tension in the wires can lead to failure due to fatigue	4	Diameter of the wires are chosen after accounting for peak loads and safety factor	5	180	Better Material used to ensure longer cycles, but at increased cost; Smoother gait transitions can reduce sudden jerks, hence impulsive loading	Material selection and link cross-section updated based on FEA results	6	3	4	72

Figure 12: FMEA Worksheet: SEV = How severe is effect on the customer? OCC = How frequent is the cause likely to occur? DET = How probable is detection of cause? RPN = Risk priority number in order to rank concerns; calculated as SEV x OCC x DET

3 Controls

3.1 Central Pattern Generators (CPG) Framework

3.1.1 Intuition behind CPG

Central Pattern Generator is a relatively new theory being applied in robotics, specifically in problems dealing with locomotion. This theory is in fact motivated by the Central Pattern Generators existing in nervous system of vertebrates and some invertebrates. CPG.s are biological neural circuits that produce rhythmic outputs in the absence of rhythmic input.They produce tightly coupled patterns of neural activity that drive rhythmic motor behaviors such as walking, swimming, breathing, or chewing.
For example, consider the task of walking. Now one possible mechanism of walking can be this: you get a sensory input each time you step down, based on which you decide the next action and so on. While this seems intuitively correct, this is not what happens and has been shown through several experiments. While walking, the CPG frame work generates a rhythmic pattern. Let us for instance focus on the joints of the hip and legs. The CPG frame work generates a rhythmic pattern for each of these joints (angle as a function of time). It is obvious to see that these patterns wont be independent of each other. They would be kinematically compatible. We note that once a feasible rhythmic pattern has been generated, a person can walk on an even surface without any necessary sensory inputs. But practically, this is not what happens. While CPG.s don't require sensory input to generate rhythmic patterns, we can incorporate sensory inputs by changing the parameters of the CPG. Just for the sake of this example, assume that the CPG produces sinusoidal function for each of the joint angles which have different amplitudes, phase difference. Now as per the surroundings, we can vary the properties of the sinusoidal wave form, to generate rythmic patterns compatible to the surface. Thus, CPG.s are only fine tuned by sensory feedback. In fact, just by changing the parameters, we can change our action from walking to running.
A similar theory is used for robot locomotion. All the joint angles have some periodic trajectory as a function of time dependent of some parameters. These trajectories are of course not independent, for instance, they might have a fixed phase difference. Now depending on the surroundings, we change the parameters, to alter the movement such that it is compatible with the surroundings. CPG parameters maybe derived from experience, data-driven optimization methods and Reinforcement learning. Here we used a RL based approach. CPG just generates the required trajectory for each joint angles, for actually implementing it, a separate controller such as PID is used to give the motors appropriate torques calculated from the system dynamics.

3.1.2 Application to given Problem

For the given problem, a 3D two-layer artificial center pattern generator (CPG) network has been adopted to generate locomotion for the hexapod. The parameters of CPG are further fine tuned to increase the adaptability of the robot. Here, to begin with we consider a simpler hexapod with only 3 joints per leg. Note that, once we have derived the trajectory for such a system, we can determine the end effector trajectory. We can then apply inverse kinematics to determine the joint motion for our system. We consider the following mechanical design for a leg: (Nomenclature: Joint i_j represents the j^{th} joint of the i^{th} leg ($i \in 1, ..., 6$, $j \in 1, 2, 3$). $j = 1$ represents hip joint, 2 and 3 represents knee and ankle joint respectively)

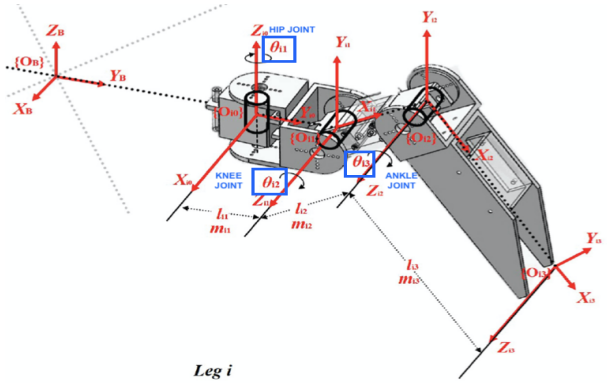


Figure 13: Mechanical Structure of the robotic leg for controls analysis

A 2 layer CPG system is considered, where the first layer is responsible for generating basic locomotion patterns

such as tripod locomotion, quadruped locomotion and five legs support locomotion. The second layer controls the limb motion. The first layer or the body layer gives motion to the hip joints while the second layer or the limb layer gives control input to the knee and ankle joints based on the inputs given to the hip joints.

In tripod locomotion, 6 legs are separated into 2 sets of legs which work alternatively while in quadruped locomotion,

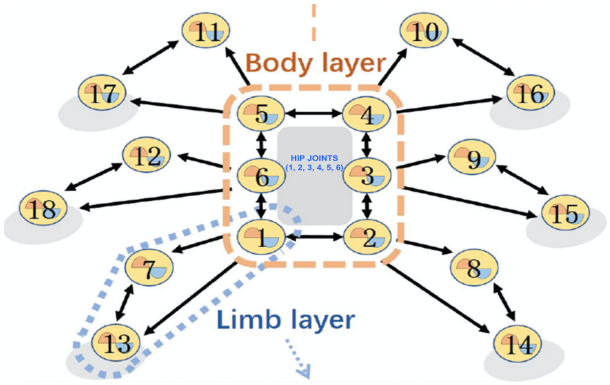


Figure 14: 2 Layer structure of the CPG framework - Body layer and Limb Layer

the 6 legs are divided into groups of 4 groups which move successively. The body layer is responsible for giving appropriate control inputs to the hip joints to decide whether the mode of locomotion is tripod or quadruped. Note that these control inputs have to be perfectly coordinated with well defined phase differences to achieve the given modes of locomotion. This is achieved by defining differential equations with coupled dynamics for all the hip joints. These differential equations are designed to converge in limit cycles which thus provide periodic coordinated motion to all the hip joints. The body layer consists of six hip oscillators with bidirectional couplings.

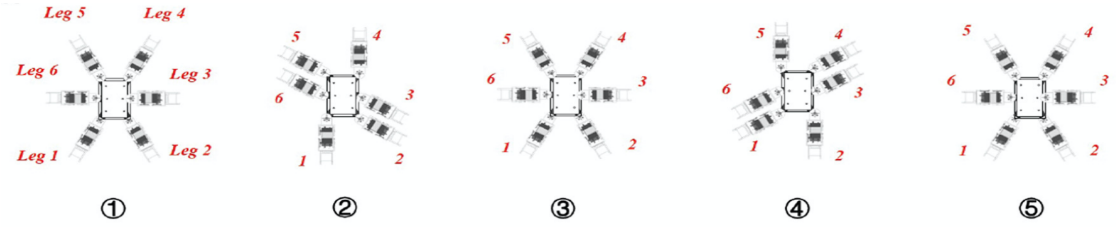


Figure 15: Tripod locomotion of a hexapod

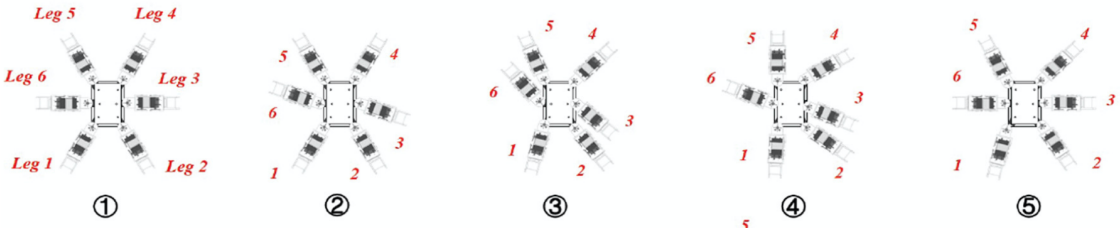


Figure 16: Quadruped locomotion for a hexapod

The limb layer includes three oscillators in association with the hip joint, the knee joint and the ankle joint in every leg, where the knee joint oscillator and ankle joint oscillator are interconnected with bidirectional coupling, but the oscillator pair is uni-directionally controlled by the corresponding hip oscillator in the body layer. Hence the CPG model can be described as a set of coupled Hopf oscillators. The solution to these differential equations generates a stable limit cycle, that is a periodic trajectory for the knee and ankle joints, dependent on hip joint trajectory. Thus, the CPG framework defines periodic coordinated trajectories for all the joints. The exact motion

depends on the parameters of the differential equations. For an even terrain, we can choose suitable parameters and keep them constant, but this approach wont work on an unstructured terrain. For an unstructured terrain, the form of the differential equations remains the same, we vary the parameters based on sensor inputs. This has been covered in the next subsection.

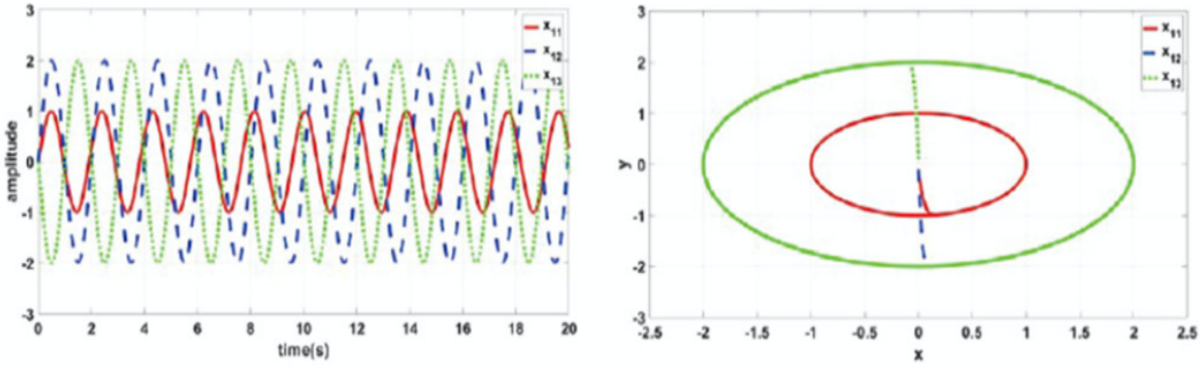


Figure 17: A. Control Inputs generated for leg joints as a function of time B. Limit Cycles solutions for leg joints

3.2 Locomotion Optimization

3.2.1 Reinforcement Learning

The problem at hand when determining the locomotion of a hexapod can be very well modelled as a Markov Decision Process (MDP). An MDP M is completely defined by (S, A, T, R, γ), where S is the set of states $\{s_1, s_2, ..., s_n\}$ with $|S| = n$; A is the set of actions $\{a_1, a_2, ..., a_k\}$ with $|A| = k$; T is the transition function defined for $s, s' \in S$ and $a \in A$ as $T(s, a, s')$ which gives the probability of reaching s' by starting at s and taking action a; R is the reward function defined for $s, s' \in S$ and $a \in A$ as $R(s, a, s')$ which is the (numeric) reward for reaching s' by starting at s and taking action a and γ is the discount factor, where a larger γ implies farther look-ahead. The agent is born in some state s^0 and takes action a^0. Then, the environment generates and provides the agent next state $s^1 \sim T(s^0, a^0, .)$ and reward $r^0 = R(s^0, a^0, s^1)$. At next time step $t = 1$, the agent is in state s^1 and takes action a^1. The environment generates and provides the agent next state $s^2 \sim T(s^1, a^1, .)$ and reward $r^1 = R(s^1, a^1, s^2)$. This continues for the entire episode, thus generating a trajectory $(s^0, a^0, r^0, s^1, a^1, r^1, ...)$. The action a^t taken at any time step t, from the state s^t, is decided by the policy $\pi : S \longrightarrow A$, which is assumed to be markovian, deterministic and stationary. The set of all possible policies is denoted by Π. The state value function is defined as the total discounted reward $V^\pi(s) = \mathbb{E}(r^0 + \gamma r^1 + \gamma^2 r^2 .....|s^0 = s)$. The action value function $Q^\pi(s, a)$ is the expected long-term reward from starting at s, taking a at $t = 0$ and following π for $t \geq 1$ and is mathematically written as $Q^\pi(s, a) = \mathbb{E}(r^0 + \gamma r^1 + \gamma^2 r^2 .....|s^0 = s, a^0 = a, a^t = \pi(s^t) for t \geq 1)$.
In typical reinforcement learning problems, the objective is to maximize the expected long term reward, which is defined as:

$$J(\theta) = \mathbb{E}(\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta) = \sum_{t=i}^{T-1} P(s_t, a_t|\tau) r_{t+1}$$

where θ represent the various policy parameters, i is an arbitrary starting point in the trajectory, $P(s_t, a_t|\tau)$ is the probability of the occurrence of s_t, a_t, given the trajectory τ. The objective is therefore, to compute θ^* which characterizes the optimal policy π^*, which maximizes the expected long term reward.
An MPD which characterizes a hexapod cannot have complete freedom since a hexapod can encounter actions which lead to collisions, falls, and other inaccessible locomotion for a real robot. Therefore, the actions of the hexapod robot should be constrained by several conditions such as acceleration, velocity, and torque constraints, which ensures the robot safe exploration. In a constrained MDP, we then have additional parameters: cost function $C_l : S \times A \times S \longrightarrow \mathbb{R}$ which maps transition tuples into costs with the limit c_l. The expected discounted cost with l constraints is then given by:

$$J_{C_l}(\pi) = \mathbb{E}(\sum_{t=0}^{\infty} \gamma^t C_l(s_t, a_t, s_{t+1}))$$

with the set of feasible policies $\Pi_C = \{\pi \in \Pi : \forall l, J_{C_l}(\pi) \leq c_l$.

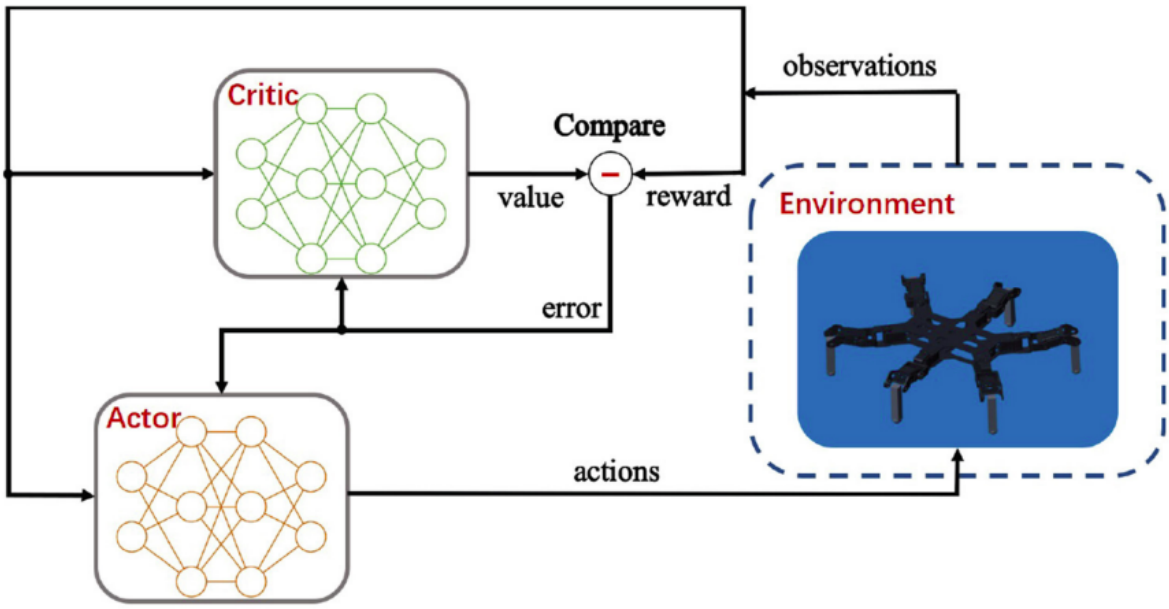


Figure 18: The reinforcement learning architecture based on the Deep Deterministic Policy Gradient algorithm for adaptive locomotion control.

3.2.2 Deep Deterministic Policy Gradient Algorithm

The adaptive locomotion control of a hexapod robot is a challenging task due to the high-dimensional observations, continuous state space and continuous action space. The Deep Deterministic Policy Gradient (DDPG) conquers this problem by employing Q-learning and Policy gradients. The adaptive control policy π, parameterized by θ^{π}, is learnt by combining policy gradient with an actor-critic network. We need to update the parameter vector θ^{π} in the direction of the action gradient, which is given by:

$$\nabla_{\theta^{\pi}} J(\pi) = \mathbb{E}(\nabla_a Q(s,a) \nabla_{\theta^{\pi}} \pi(s))$$

where $Q(s,a)$ is the action-value function, which needs to be estimated. DDPG being an actor-critic technique consists of two models: Actor and Critic. The actor is a policy network that takes the state as input and offers a control policy π that outputs the exact action (continuous), instead of a probability distribution over actions (hence, deterministic!). The critic is a Q-value network, parameterized by θ^{Q}, that takes in state and action as input and outputs the approximate action-value function. The actor network and critic network are approximated by deep neural networks, which are learned via policy gradient method and error back propagation, respectively.
To increase stability during training of these deep neural networks, the DDPG algorithm includes target critic network (parameterized by $\theta^{Q'}$) and target actor network (parameterized by $\theta^{\pi'}$) to calculate Q-value for next state. The target networks (Q', π') are delayed networks compared to the current networks (Q, π). The weights of targets are updated periodically based on the current networks. We then transfer over only a fraction of the current weights to the target weights as follows:

$$\theta^{Q'} = k\theta^{Q} + (1-k)\theta^{Q'}$$
$$\theta^{\pi'} = k\theta^{p}i + (1-k)\theta^{pi'}$$

As with other deep reinforcement learning techniques, DDPG relies on the use of replay buffer for stability. The replay buffer needs to maintain a balance of old and new experiences. At each time step, both the actor network and the critic network are updated by sampling a mini batch (of size S) uniformly from the buffer.

The loss function to be minimized can then be written as follows:

$$L = \frac{1}{S}\sum_{t=1}^{\infty}((r_t + \gamma Q'(s_{t+1}, \pi'(s_{t+1}|\theta^{\pi'})|\theta^{Q'})) - Q(s_t, a_t|\theta^{Q}))^2$$

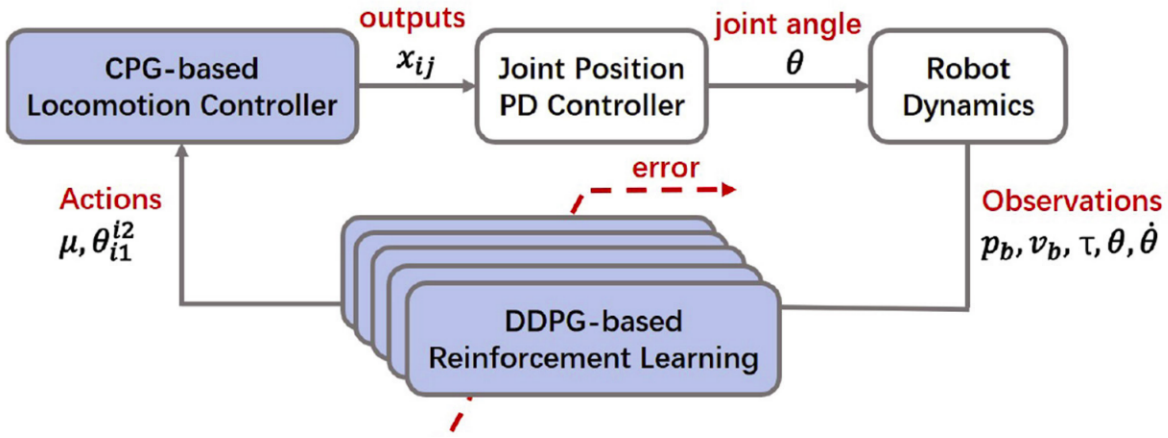


Figure 19: The proposed control scheme has a cascaded structure with a feedback loop, consisting of three parts: (1) Joint Position PD Controller (2) Two-layer CPG locomotion controller (3) DDPG-based RL motion controller.

3.2.3 MDP Formulation for Hexapod Robot

The observation parameters for our MDP are the dynamics parameters $\boldsymbol{p_b}$ and $\boldsymbol{v_b}$, which are the robot body position and velocity vector respectively; the body orientation $\mathcal{O}$; joint torques $\boldsymbol{\tau}$; joint angles $\boldsymbol{\theta}$; joint angle velocity $\dot{\boldsymbol{\theta}}$; amplitude and phase difference in the limb layer of the CPG, A. Therefore, the MDP observation vector at any time instance t is $\boldsymbol{o_t} = \{\boldsymbol{p_b}, \boldsymbol{v_b}, \mathcal{O}, \boldsymbol{\tau}, \boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, A\}$. The coupling parameters of the limb layer of the CPG which determine the intra-limb coordination as well as the adaptation to different terrain types constitute the action space of our MDP. Therefore, the action vector at any time instance t is $a_t = \{\mu, \theta_{i_1}^{i_2}\}$, where μ represents the amplitude and $\theta_{i_1}^{i_2}$ represents the phase difference between the hip joints i_1 and i_2. The reward function is designed to encourage faster heading velocity, v_x, and penalize higher energy consumption, $|\tau.\dot{\theta}| + \tau^2$. Therefore, the reward at any time instance t is computed as $K_1 v_x - K_2(|\tau.\dot{\theta}| + \tau^2)$, where K_1, K_2 are positive constants. The joint speed and torque are the limits of the actuator performance of the robot. Therefore, the costs to be added to control these two are $||max(|\dot{\theta}| - \dot{\theta}_{max}, 0)||^2$ and $||max(|\tau| - \tau_{max}, 0)||^2$, respectively. We can also penalize the bot from swinging too much by using $||\mathcal{O}||^2$ and $||z_c - z_{co}||^2$ where the first term is for orientation and second represents the deviation of the bot center height from its original height. We also need to add safety constraints to penalize the bot from exceeding the maximum allowed joint speed ($\dot{\theta} > \dot{\theta}_{max}$) and joint torque ($\dot{\tau} > \dot{\tau}_{max}$). We also need to train the bot so that it does not fall ($z_c < 0$) or roll ($\mathcal{O} > \mathcal{O}_{limit}$). So, if the train policy tries to violate any of these safety constraints, we terminate the episode directly. The training steps explored in this episode are abandoned and a negative terminal reward is added to the last training step. This control policy avoids inefficient explorations of constrained regions.
The critic network and actor network of our DDPG algorithm are parameterized as deep neural networks. The critic network is composed of five hidden layers including three fully-connected layers, with two of them having a ReLU activation. The actor network consists of six hidden layers including three fully-connected layers, with two of them having a ReLU activation and one having Tanh activation. The output from this DDPG-based RL motion controller modulates the proposed two-layer CPG parameters.

4 Automation

4.1 Enemy Detection: Camouflage detection

The main task that we want our bot to accomplish is the detection of enemies including which are camouflaged in forest trees. These enemies are very challenging to identify even for humans. For this we will train a Machine learning model which could segment the camouflaged objects from rest of surrounding. It will give our soldiers the prior information where enemies are located and also an option to shoot them using the bot.

4.1.1 Datasets

We first started to look for the relevant datasets for our task of camouflage detection and ended up over two standard camouflage segmentation datasets. We would use these datasets for training and testing our models. We will consider the model which would work good on these for our task.

- CAMO dataset

  Camouflaged Object (CAMO) dataset specifically designed for the task of camouflaged object segmentation. It focus on two categories, i.e., naturally camouflaged objects and artificially camouflaged objects, which usually correspond to animals and humans in the real world, respectively.

  Camouflaged object images consists of 1250 images (1000 images for the training set and 250 images for the testing set). Non-camouflaged object images are collected from the MS-COCO dataset (1000 images for the training set and 250 images for the testing set).

  Google Drive link to download CAMO dataset

- COD10K

  COD10K contains 10,000 images (5,066 camouflaged, 3,000 background, 1,934 noncamouflaged), divided into 10 super-classes, and 78 sub-classes (69 camouflaged, nine non-camouflaged) which are collected from multiple photography websites.

4.1.2 SINet-V2

- **Motivation**

  SINet-V2 is currently state of the art model for camouflage segmentation with the highest E-measure of 88.2 and S-measure of 82.0 on the CAMO dataset.

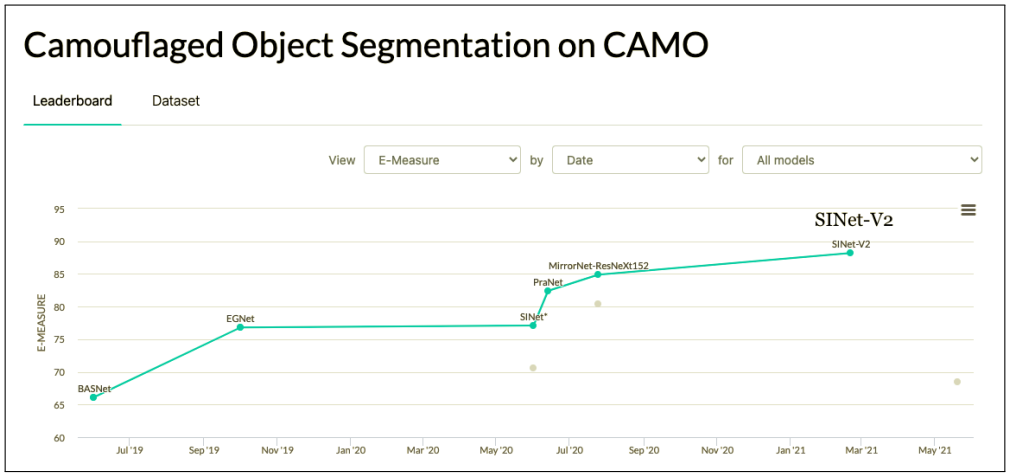


Figure 20: Performance of different models on CAMO dataset

SINet-V2 is currently state of the art model for camouflage segmentation with the highest E-measure of 88.7 and S-measure of 81.5 on the COD10k dataset.

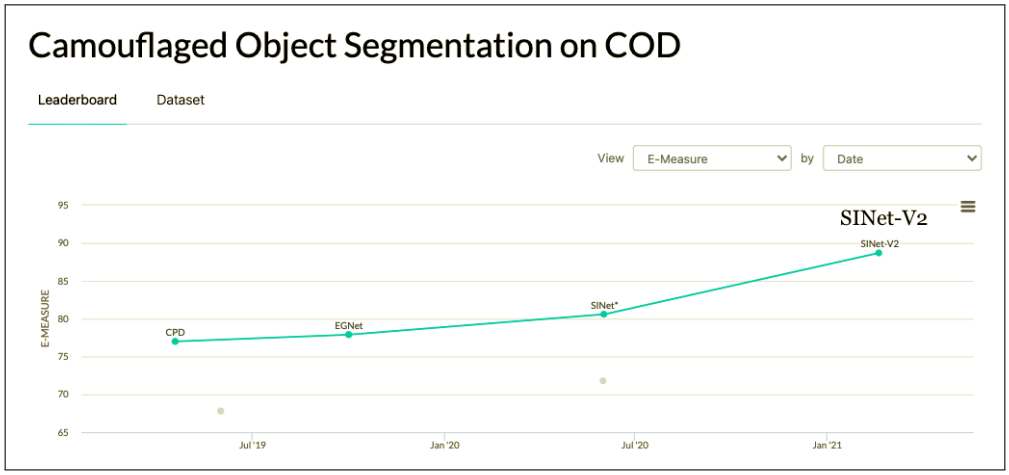


Figure 21: Performance of different models on COD10K dataset

- **Architecture**

  SINet-V2 consists of three main components:

  - Texture enhanced module (TEM) - The TEM is introduced to mimic the textural structure of receptive fields in the human visual system. It is used to capture fine-grained textures with the enlarged context cues.
  - Neighbor connection decoder(NCD) - The NCD is responsible for locating the candidates with the assistance of the TEM. It is able to provide location information.
  - Group-reversal attention (GRA) - The GRA blocks reproduce the identification stages of animal predation. These work collaboratively to refine the coarse prediction from the deeper layer.

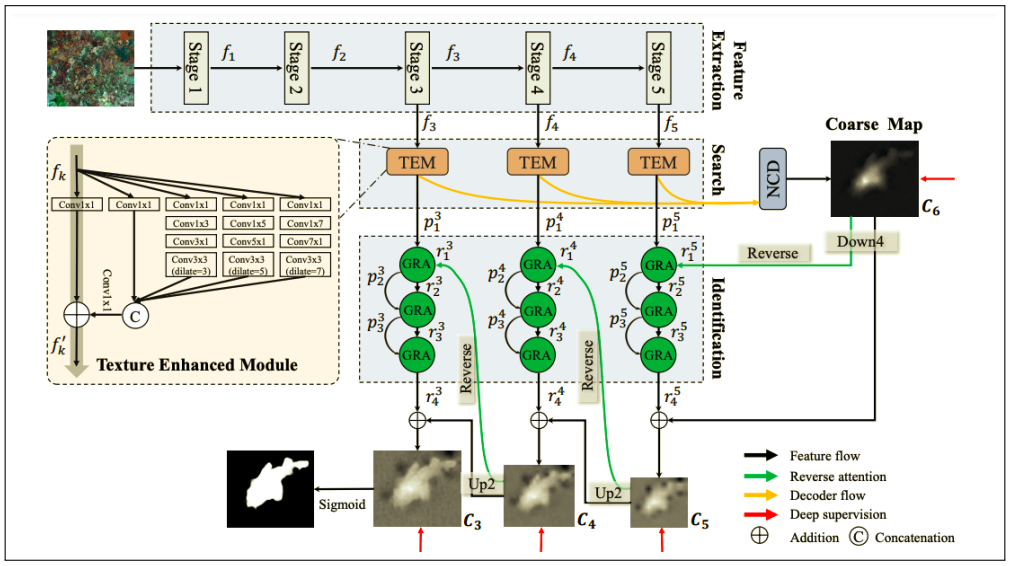


Figure 22: The pipeline of SINet framework

- **Implementation and results**

  We downloaded the GitHub repository and used a pretrained model.
  The training and testing experiments are conducted using PyTorch with a single NVIDIA Tesla K80 GPU of 12 GB Memory. Our dataset contains the images of soldiers in forest, among which many are camouflaged. We test our model on these images and our model successfully detects the enemies.

Figure 23: Model performance on Forest data

One of the main aspect of any machine learning algorithm is how well it performs on dataset with different background. Since India is large country sharing border with many countries with different environment, it becomes important that our model is generalizable and able to detect enemies even in different environment than forests

Figure 24: Model performance on different environments

4.2 Automatic shooting technology

The shooting would be use depending on the requirements of mission, If the mission is only for surveillance then only smoke gun would be used. In case of combat mission firing guns would be used and the camera would be mounted directly on gun so as to see the shooting view.

There will be a automatic gun installed in the bot which would shoot in given direction when it receives the signal. The shooting permission would come from base and the angle to shoot would be given by the another pipeline.

After detecting the enemy using Machine learning, we will find the most appropriate point to shoot. The pixel location of that point would be passed to the depth camera which would give the global location of that point wrt to our bot. Using the location, we would calculate the unit vector which is the angle to shoot. After this our bot would send the signal to base that we are ready to shoot. If the permission is given then the bot would shoot at that angle.

We can in future also integrate this pipeline with which base can choose the mode of attack. The soldiers would be able to decide whether they want to kill the enemy or make unconscious.

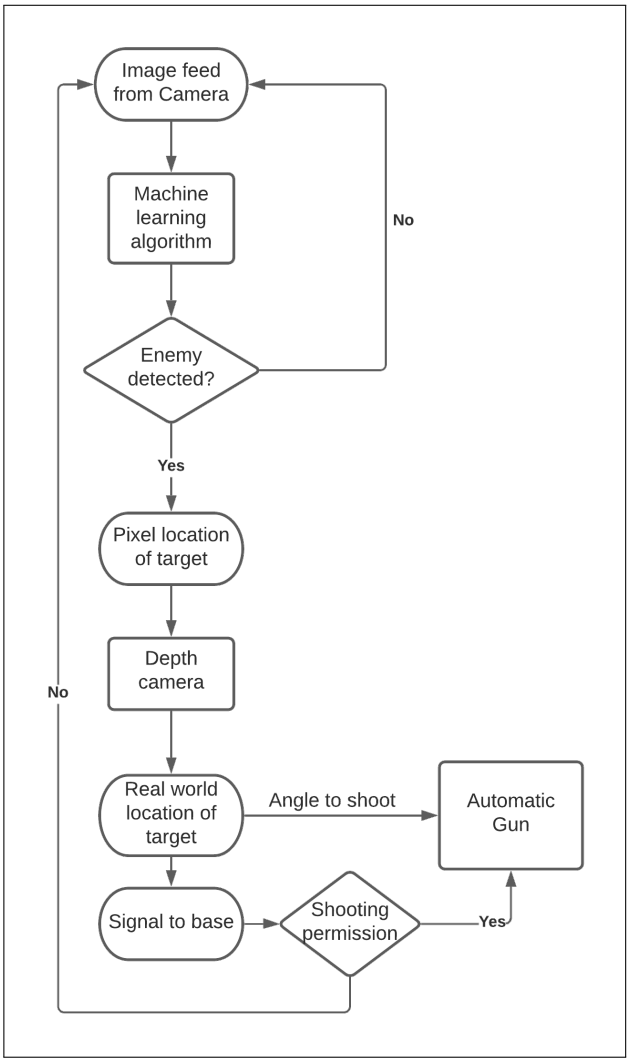


Figure 25: Pipeline for automatic shooting

4.3 Simultaneous Localization and Mapping (SLAM)

Global localization is very important for the bot to work in the autonomous mode. The localization will serve 2 important needs. The localization would serve as feedback to the controller when commanded to move from position A to position B by giving real-time feedback of its measured position. The Mapping of the environment would help the soldiers to know the environment ahead of them even at night because of mapping. The main goal of global localization would be to create an approximate map of the environment as well as sending the continuous position of the device to the base. Simultaneous localization and Mapping would serve as providing the live position of the device to soldiers. If soldiers are unaware of the environment ahead, they can send the bot to get the idea

Implementation We use Visual-Inertial SLAM (Simultaneous Localisation and Mapping) system, ORB- SLAM3 for accurate pose estimation. ORB-SLAM is a versatile and accurate SLAM solution for Monocular, Stereo and RGB-D cameras. It is able to compute in real-time the camera trajectory and a sparse 3D reconstruction of the scene in a wide variety of environments. We use RGB-D camera for depth estimation. Intel D435 camera has an ideal range of 0.3 - 3m. We will use this implementation https://github.com/raulmur/ORB SLAM2 of ORBSLAM 2. Furtur using ROS and Gazebo in simulation environment, we shall tune the parameter's of ORBSLAM.

References

- Wenjuan O. et al., 2021, Adaptive Locomotion Control of a Hexapod Robot via Bio-Inspired Learning, Frontiers in Neurorobotics, Volume 15, Pages 1, DOI: 10.3389/fnbot.2021.627157, URL
- Deriving Policy Gradients and Implementing REINFORCE by Chris Yoon, Dec'30, 2018
- Understanding Actor Critic Methods and A2C by Chris Yoon, Feb'6, 2019
- Deep Deterministic Policy Gradient: Theory and Implementation by Sunny Guha, Jun'1, 2020