A PROJECT REPORT ON FARM BOT

DRISHTI

A Revolutionary Concept

TEAM MEMBERS

- VIRAG SHAH
- PANELIYA TARANG
- BUTTI RAVITEJA
- NAMAN JAIN
- CHIRAG GUPTA
- HIMANSHU LADDHAD
- PRANAV PREMLANI
- ABHISHEK AGARWAL
- NAVEEN RAWAT
- RIYA DHOLAKIYA
- HARSH KAKADIYA
- PRAJWAL SONWANE

MENTORS

- YASH FALDU
- SHASHWAT MEDHIR
- DEV PATEL
- MANASVI SINHA

Acknowledgement

Before we get into the thick of things,we would like to add a few words of appreciation for the organisation and people who have been a part of this project from the beginning.This project would not be completed without their support and active guidance.we are extremely thankful to Drishti-A revolutionary concept for providing us a platform to learn so many new concepts and implement them in this practical world.

In this corona pandemic situation where students are sitting at their homes ,drishti made it possible for us to build projects and learn some new innovations through online mode.

We are very thankful to our mentors who were with us from the beginning, they supported and guided us in every situation.without their support and guidance it would not be possible to complete the project on time.

We are equally thankful to each other for the coordination and support we show in this online mode .It is difficult to do teamwork without any physical meetings and we as a team did our best to make everything workout correctly.

Abstract

We aim to design, manufacture and simulate a robot capable of performing weed removal, ploughing, sapling plantation, watering the plant and many more things which are mostly done manually in today's era. "FARMBOT" is an idea to take a run in assisting farmers and make farming easier. This project will allow you to explore and follow your imagination and implement them as a team. In farming, there are a wide range of mechanisms and algorithms which you can implement and we are mentioning some of them below.All the farming processes done are time-consuming and require a sufficient amount of labour work. This innovation will enable us to increase the farming processes as well as the range of plantation, which is very important in man's dream to colonise habitable planets.

Contents

1 Pre -project research

1.1 Drive Train:

There are two types of motion:

Holonomic motion: Where a vehicle can go in all X and Y directions.

Non-Holonomic motion: Where a vehicle is constrained to only certain motions.

There are various types of drives which perform holonomic motion.

1. Omni Drive
2. Mecanum Drive

An example of Non-Holonomic motion drive is Ackermann steering mechanisms.

1. **Omni directional drive:**

Omni wheels have smaller wheels(rollers) attached perpendicular to the circumference of another bigger wheel.

The major advantage is that the bot does not need to rotate or turn to move in any direction unlike other designs like differential drive/steering mechanisms.

2. **Mecanum Drive :**

These wheels are somewhat similar to omni wheels but instead of rollers being perpendicular to
the wheel, they are in an oblique direction. These rollers typically each have an axis of rotation at 45° to the wheel plane and at 45° to the axle line.

1.2 Pneumatic Circuit:

A pneumatic system is a system that uses compressed air to transmit and control energy. Pneumatic systems are used in controlling train doors, automatic production lines, mechanical clamps, etc
These pneumatic systems have a lot of advantages as these are widely used in many industries. Some of the common advantages are -

- High effectiveness
- High durability and reliability
- Simple design
- High adaptability to harsh environment
- Safety
- Easy selection of speed and pressure
- Economical and Environmental friendly

But It has some disadvantages too-

- Low loading
- Noise
- Relatively low accuracy

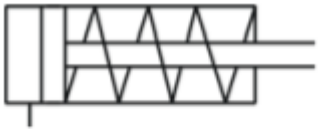
Main Basic Components of Pneumatic systems are-

- Compressor - A compressor can compress air to the required pressures. It can convert the mechanical energy from motors and engines into the potential energy in compressed air.
- Pressure regulating component - Pressure regulating components are formed by various components, each of which has its own pneumatic symbol:

(i) Filter – can remove impurities from compressed air before it is fed to the pneumatic components.
(ii) Pressure regulator – to stabilise the pressure and regulate the operation of pneumatic components.
(iii) Lubricator – To provide lubrication for pneumatic components

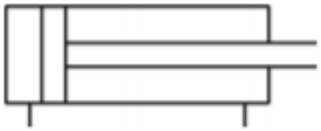
- **Execution component** - Pneumatic execution components provide rectilinear or rotary movement. Rectilinear motion is produced by cylinder pistons, while pneumatic motors provide continuous rotations. There are many kinds of cylinders, such as single acting cylinders and double acting cylinders.

    - Single acting cylinder - These are having only one entrance that allows compressed air to pass through.That means they can produce thrusts in only one direction. The piston rod is propelled in the opposite direction by an internal spring.



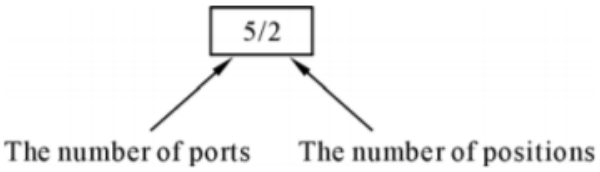
Pneumatic symbol of a single acting cylinder

    - Double acting cylinder - In a double acting cylinder, air pressure is applied alternately to the relative surface of the piston, producing a propelling force and a retracting force.



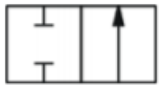
Pneumatic symbol of a double acting cylinder

- **<u>Directional control valve</u>** - Directional control valves ensure the flow of air between air ports by opening, closing and switching their internal connections. They can be classified on the basis of how many ports are there or how many switching positions exist in it. They can be denoted by following way ---



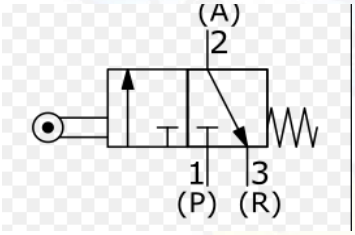
Some of the common types of valves are:

- ❖ <u>2/2 Directional control valve -</u> There will be only 2 ports and 2 switching positions.It uses the thrust from the spring to open and close the valve. The control valve can be driven manually or mechanically, and restored to its original position by the spring.



Pneumatic symbol of a 2/2 directional control valve

- ❖ <u>3/2 Directional control valve</u> - 3 ports and 2 switching positions. This type of valve can be used to control the single acting cylinder. As we can see in the diagram , when P and A are connected then pressure supply is on and when position gets switched, A and R gets connected and excess pressure or Air coming from the cylinder can be taken out and easily released into the atmosphere. The valves can be driven manually, mechanically, electrically or pneumatically.

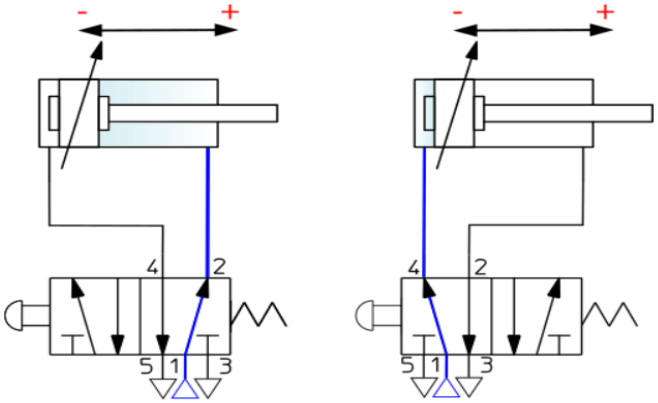


3/2 directional control valves can further be divided into two classes: Normally open type (N.O.) and Normally closed type (N.C.)

(a) Normally closed type

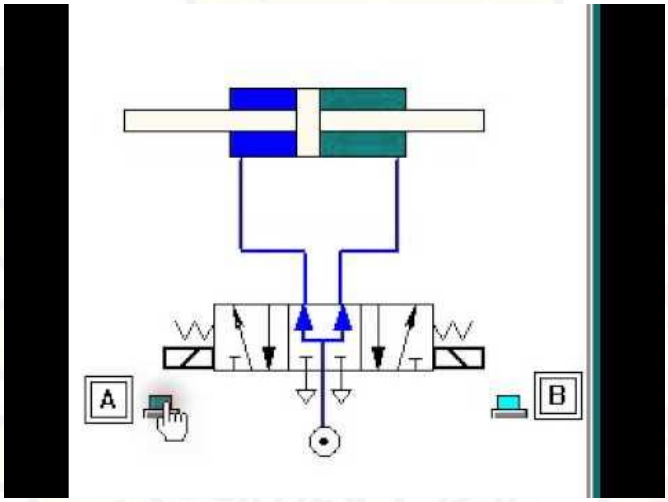
(b) Normally open type

- 5/2 Directional control valve - Just like above valves , This would have 5 ports and 2 switching positions or position valves that will put a fluid or air into one end of a double acting device as well as allowing the other end vent to exhaust. As we can see in the below image, The blue line is connected to the compressor(source of high pressure air). The first image contains 1 to 2 connections allowing the cylinder to retract and from 4 to 5 the air will be released out . Now, when the positions are switched the cylinder can be pushed outwards due to air coming into it when 1 to 4 connected and 2 to 4 air can be released out. The valves can be driven manually, mechanically, electrically or pneumatically.



- 5/3 Directional control valve - Working is the same as 5/2 Control valve. The only difference is that It has 3 positions that it can switch and used for double acting cylinders only, just like 5/2 valves.A 5/3-way valve has five ports and three states. They have two solenoids that each can control a valve state. If no

solenoid is energized, the valve returns to the central state Where it won't do anything.

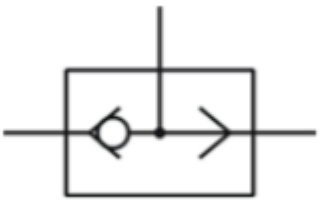


- **<u>Control valve</u>** - A control valve is a valve that controls the flow of air. Examples include non-return valves, flow control valves, shuttle valves, etc.

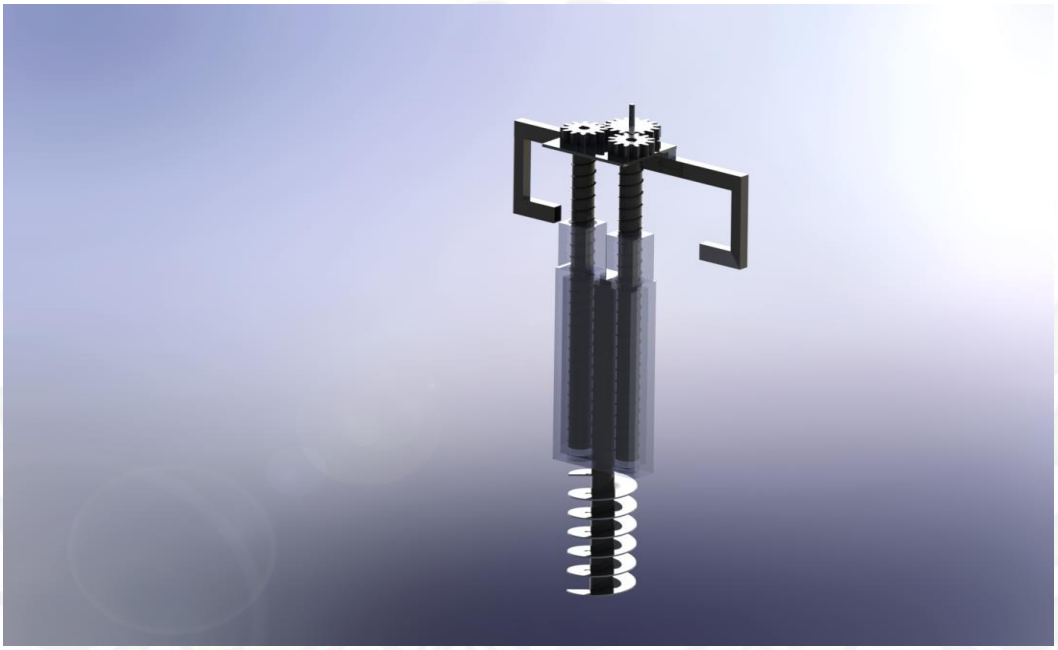
    ★ <u>Non-return valve</u> - Which allows air to flow in one direction only. When air flows in the opposite direction, the valve will close.

★ Flow control valve - We can say It is formed by a non-return valve and a variable throttle.

★ Shuttle valve - These are also known as double control or single control non-return valves. Basically, It contains 2 inlets and one outlet. When air enters through one inlet, there is a ball sphere inside it which blocks the other inlet and allows air to go from the first inlet to the outlet. If sir comes from the other way around the same thing will happen. This is how it plays the role of a 2 way control valve.



2 Soil collecting module



Before starting digging or ploughing it is advisable to first test the soil so that we can know whether it has sufficient amount of required nutrients. If soil is devoid of necessary nutrients it can lead to poor quality of the crop. So this module performs the task of collecting the soil for testing purposes. The bot collects soil from different places on the field using auger. This auger is paired with the lead screw so that it can lower itself and the rotating auger can collect the soil. As the soil is collected, auger is pulled up using a lead screw.

3 Agricultural research for sowing

- **Sowing time:**

Based on above temperature requirement it has been found that for indigenous wheat last week of October, for long duration dwarf varieties like Kalyan Sona, Arjun, etc. first fortnight of November and for short duration dwarf wheats like Sonalika, Raj 821 etc. second fortnight is the best sowing time. Under exceptionally late sown conditions it may be delayed to the latest by 1st week of December beyond which if the area is very small transplanting may be practiced.

- **Seed rate:**

Generally, a seed rate of 100 kg/ha has been found to be sufficient for most of the varieties like Kalyan Sona, Arjun, Janak, etc. Which have moderate tillering and medium sized grains. But a higher seed rate of 125 kg/ha is desirable for late sown wheat and normal sown for varieties like Sonalika, Raj 821 etc. which have bold grains and shy tillering habits.

- **Spacing and Cultivation Height (for particular crops):**

Crop	Inter row spacing (cm)	Intra row spacing (cm)	Cultivation height (cm)
Rye	40-45	15	100-110
Wheat	60-75	28	60-88
Sorghum	30	10	0.6-2
Mustard	30	10-15	90
Barley	20-30	-	60-120

Examples of sowing methods in use by farmers:

- Drilling

  In this method seed is sown by seed drill or ferti-seed drill. With the help of this implement seed drop at depth and results in uniform germination and regular stand. Seed beds should be fine and well levelled free from clods and weeds for the use of seed drill or ferti-seed drill. Seed drills are easily available in the market. They may be either bullock driven or tractor driven. Ferti-seed drill should be used wherever possible to ensure uniform depth of sowing, proper placement of fertilizers and good germination.

- Dibbling

  This method is used in cases where the supply of seed is limited. Sowing is done with the help of a small implement known as 'Dibbler'. It is a wooden or iron frame with pegs. The frame is pressed in the field and lifted and then one or two seeds are dropped by hand in each of the holes. It is not a common method because it is a very time-consuming process.

- Zero tillage technique

  This new method is used in the Rice-Wheat cropping system where showing of wheat is delayed beyond 25 November. The field preparation is one of the most important reasons, which causes delay in wheat sowing. Puddling in transplanted rice creates a hard pan in the field. After harvesting of rice crop, the field requires at least 6-8 tillage operations in ploughing and harrowing for sowing of wheat, in which generally 10-15 days are required for proper field preparation. Zero-tillage can be adopted with the following preparations. At the time of sowing there should be proper moisture in the field. At the time of sowing the seed-drill should be lifted up or lower down very slowly to avoid clocking of furrow opener by soil, otherwise seeds and fertilizer will not drill in the furrow. Seed should be treated with vitavax or Bavistin at the rate of 2.5 g/kg of wheat seed. Seed rate should be 140-150 kg/ha (20-25% higher). Sowing depth should be maintained about 5-6 cm. Light planker may be used behind the zero-tillage machine.

4 Seed planting

4.1 Digging:

1 Disc digging mechanism:

This mechanism employs a disc with a sharp edge for ploughing across the soil. This disc is attached with an axle which is connected to the lower portion of the chassis. This disc is kept such that a certain portion of the disc remains inside the soil and as the bot moves forward the disc inside the soil performs the ploughing across the soil. This is a continuous ploughing or digging mechanism.

For this purpose we designed prototypes of the disc with different designs and shapes to test the mechanism. We mainly selected two disc designs:

1) Disc with smooth curved cutting edge
2) Disc with v shaped cutting edge

For initial testing we decided to test the disc for 5 cm penetration into soil with different radius and thickness of the disc.
Advantage:

- It has a simple but efficient working principle.
- It also provides continuous ploughing motion along with the motion of the bot.

2 Cam and Follower mechanism:

In cam follower mechanism The whole mechanism can move on the chassis. It is moving such that when digging and coming out the mechanism remains stationary with respect to ground. So at that time it will dig a hole in the ground. After this, the mechanism will move on the chassis in forward direction. In this whole process chassis is continuously moving forward

Advantages:

- Compact design.
- Easy to move the whole mechanism.

Disadvantage:

- If a follower is stuck in the middle then it may break.

3 : Spiked Wheel

In this idea we thought of using a wheel with spikes on its outer surface. This spiked wheel can be attached with a motor which rotates this wheel. When this wheel will rotate onto the soil the spikes will penetrate the soil and make holes. Ten using a seed ing mechanism we can drop seed one by one into the respective holes.

The spikes that are made onto the wheel can be varied in number depending on the distance that we want to keep between two seeds. We also thought of performing seeding through this spiked wheel only by using some sort of mechanism but it seemed to be impractical so we dropped the idea.

This mechanism itself was also having some issues like we can not get sure whether we will be able to dig up to the depth we want or not. The reason for this was that when the spikes would penetrate the soil the entire weight would fall on the bot and hence if the spike would not penetrate completely and it might cause damage to the bot.This was the main disadvantage with this design.

4 : Penetrating Cone

In this idea we have discussed that there will be a cone shape part which contains a single seed at a time now when the time comes for seeding this cone will be ejected in the soil. After the ejection the end part of the cone will open and the seed will fall in that dig.after that cone will pull up .
Major disadvantage of this mechanism is that the opening section is very small so it may not open properly due to the force of soil. Which may lead to breaking the end. And after that digging the covering of soil is also a major issue because the soil which will come out will be in improper means. So we can't cover it with that soil . Hence we require extra soil.

5 : Open Cone

In this idea, instead of penetrating a cone into soil and opening it, we thought of dividing the cone in two portions and connecting the cone with a horizontal linear actuator which can open and close the two portions of the cone. Initially the cone will be in an open position and it will be lowered to penetrate into the soil. When the cone will be penetrated upto desired

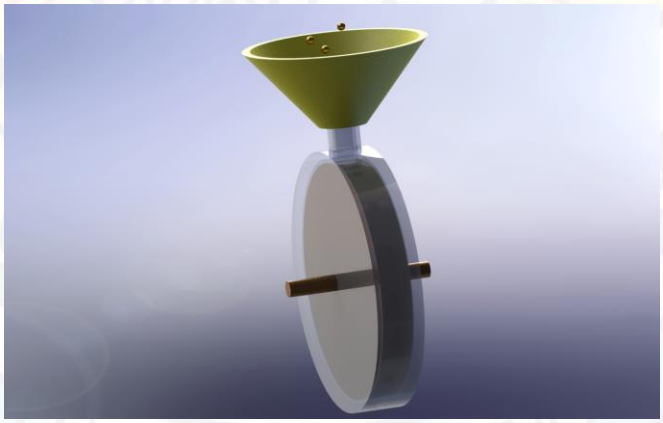
depth the horizontal linear actuator will start closing the cone. After closing the cone the cone with a certain amount of soil will be pulled up and a hole will be formed at that position. Later using the seeding mechanism we can put the seed into the hole and the soil collected with the cone can be filled back into the hole.

The main reason for rejecting this mechanism was that this mechanism involved too many complexities without complete surety of performing the digging operation that we require. Moreover the soil will be a bit harder and if the cone would not penetrate properly then it will get damaged. Also while closing the cone it is possible that soil might not get collected properly and excessive pressure might be exerted onto the cone which may cause damage to the mechanism. So these were the main disadvantages of this mechanism so we rejected this idea.

4.2 Seeding mechanism:

1 Vertical seeding wheel

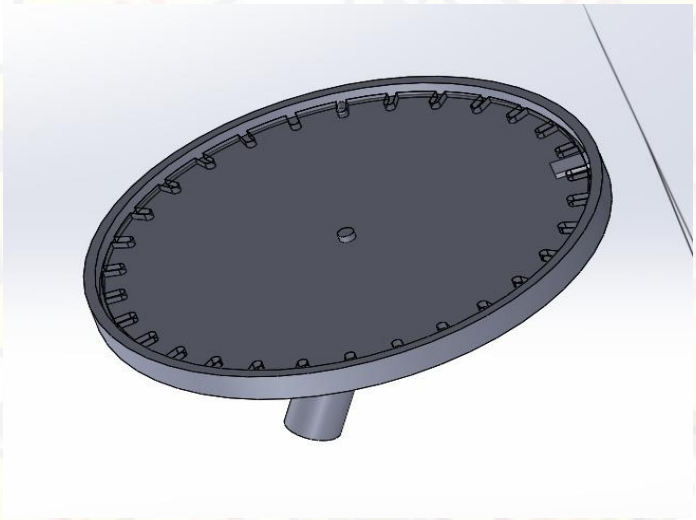
In the seeding module, there is a disk with holes of seed size and it is covered with casing. At the top of the casing there is a hole with which a funnel is attached. As soon as hole in the disk align with hole of funnel seed it will be dropped in to the disk and from the other end it will release the seed outside.



2 Inclined seeding wheel:

In this seeding module, we are using an inclined disk for which there will be a seed volume hall at periphery of the disk. And at the top most part of the cover there will be a small hall through which seed will fall into the place. But in this idea there may be a chance that seed

will jump and fall anywhere into the field as it is open from the top. And seed may also don't fall into the volume

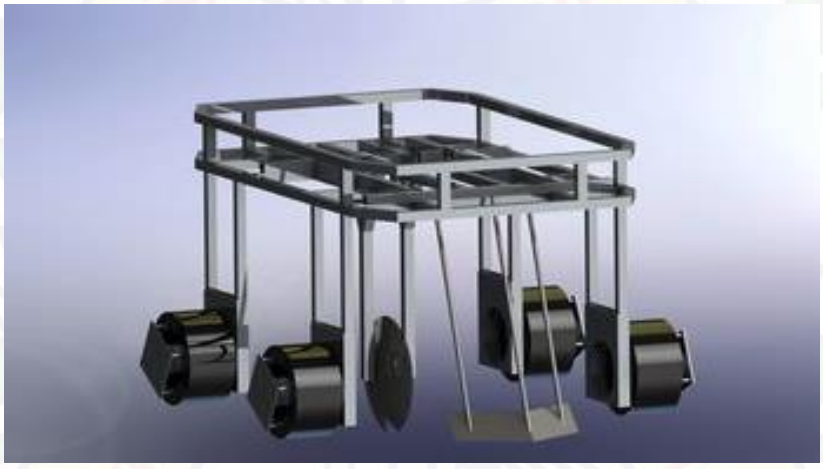


4.3 Leveler :

This module consists of a beam connected to a V-shaped plate. The main purpose of leveler is to collect the soil around the digged region and to fill the same soil back into the digged part.We have different shapes of levelers:

1)Curved
2)Flat
3)V shape
4)Square

We chose a V shaped leveler because it will properly fill the digg part with the help of soil which is removed with disk digging.



5 Research for weeding

Generally weeding is done after 1 ½ to 2 months after sowing or weedicides like 2,4 D, Avadex or Nitrofen (Tok E-25) for controlling Chenopodium sp, Angallis sp. Asphodelus sp. Phalaris sp. of weeds.

Dependent upon the most appropriate way in which the weeds must be managed, there are three different types of soil cultivators:

- Inter-row soil cultivators; operate between crop rows, enabling remarkable weed control with minimal risk to crops
- Intra-row soil cultivators; operate inside the row between the plants

- Broadcast cultivators; blind cultivation without regard for crop rows

Also available are simple and manually operated soil cultivators, as well as mechanized cultivators, which come with a variety of farming attachments and in a variety of sizes. The most recent technology developed is a totally different dimension of weed control -the precision weed cultivator- which uses a camera and computer system to distinguish weeds from plants and automatically removes the weeds.

Brush weeder

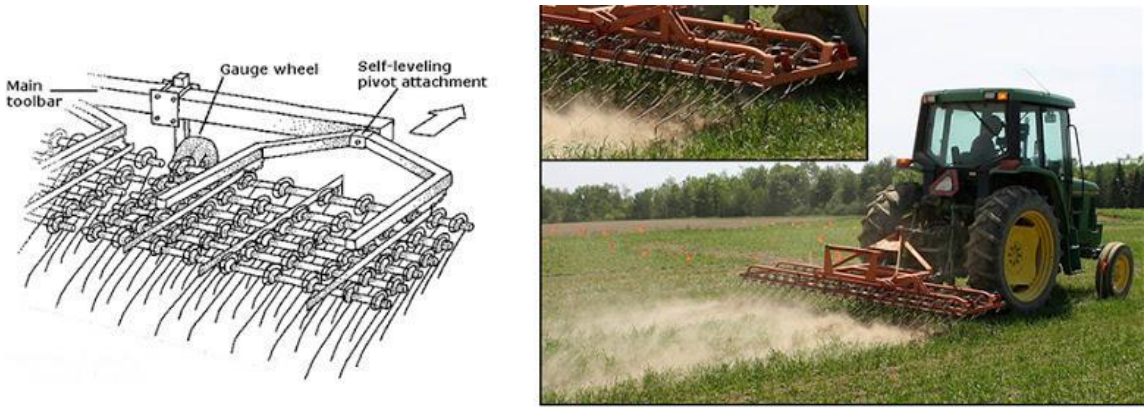
Rotary cultivator

Rolling cultivator

Brush weeders; there are several types available, including vertical and horizontal axis brushes; brushes are made of fiberglass and are flexible. These cultivators uproot, bury, or break weeds; some types require an operator to manage the cultivation so as to prevent damage to the crop. Some of them have an included shield to protect the crops.

Rotary cultivators; these cultivators have a vertical, horizontal, or oblique axis; they are equipped with blades, points, or knives that turn and pulverize the soil; used in orchard cultivation as well Rolling cultivators; consist of gangs of three to five spiders (wheels made of strong, curved, and cutting teeth or notched disks). Finger weeders; used for in-row weed management, ground driven rotating fingers push the soil and uproot weeds away from the crop row. Torsion weeders; usually mounted on an existing inter-row cultivator for additional in-row cultivation. They consist of a rigid frame with spring tines connected and bent so that two short tine segments are parallel to the soil surface and meet near the crop plant row.

Regarding harrows, farmers mostly use Flex-tine harrows, consisting of a rigid frame and a variety of fine, flexible tines. The tines destroy the weeds by vibrating in all directions.



Flex tine harrow is widely used farming implement for soil cultivation

6 Weeding Mechanism

Weeding mechanism is one the most important part of the farmbot. Weeds are basically the unwanted plants that grow anywhere in the field. These weeds need to be removed because they take up a certain portion of water and fertilizers that are meant for the plant. Also they take certain essential nutrients of the soil which are required for the plants. So weed removal is an important step in farming. Weeding is basically carried out in portions , one is inter row weeding and another is intra row weeding. In inter row weeding , the weeds present in

between two rows of crops are removed whereas in intra row weeding , the weeds present in between two successive crops are removed. Both these type of weeds should be removed and the autonomous mechanism used for this purpose is called weeding mechanism.

Weeding can be done in various ways like performing intermittent weeding in which the bot stops as it detects the weed and removes the weed using the weeding mechanism. Also we can perform continuous weeding in which the bot continues its motion and wherever it detects the weed , it activates the wedding mechanism and performs the weeding while in motion. Weeding can be performed by moving the bot in between two rows of crops or by moving the bot above the rows of crops. Below are the ideas related to weed removal or weeding mechanisms.

Ideas for weeding mechanism:

1 : Pick and collect mechanism

In this mechanism, there is a robotic arm with 3 or 4 fingers at one end with a suitable gripper and a box to collect the removed weeds. The Sensors will detect the weeds and by the help of Robotic arm weed will be removed.

This is a discontinuous mechanism.
It takes more time compared to other mechanisms

2 : Inclined Blades

This mechanism has two or three circular blades which are at inclined position with the ground. As the bot moves forward the sensors will detect the weeds and these circular blade cutters come to their respective positions and will cut the weeds.
This is slow and a continuous mechanism.

3 : Separate module type

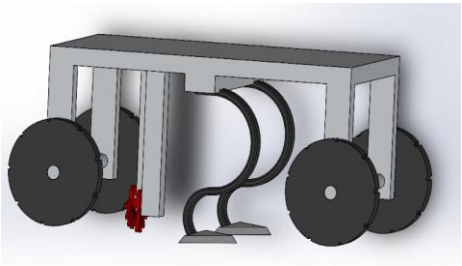
In this idea we decided to use two mechanisms for performing inter and intra row weeding. For inter row weeding we decided to use a static mechanism like a clamp that can pierce in the soil and as the bot moves forward it drags the clamp along with it and whatever weed is present in between two rows of plant will be uprooted and hence will be removed.

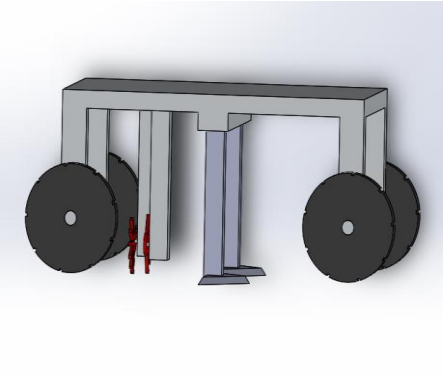
For intra row weeding a special cutter is designed in which there are few finger-like extruded portions made onto a small circular disc. This is a partial cutter in which the extruded portion is kept upto certain angle only.

As the cutter moves at the bottom position the extruded portion penetrates the soil and removes the weed that is present in between two plants. For making this mechanism efficient we kept two such cutters at a certain distance so that a larger area can be covered.

This is a continuous type mechanism so there is no need to stop the bot for weeding, the clamp performs the weeding continuously by dragging into the soil and the cutter connected with motor can be activated whenever any weed is detected in between the plants or we can also allow the cutter to move continuously and properly synchronizing the mechanism.

This mechanism has one disadvantage that the bot and the mechanism has to suffer sufficient drag and hence more force is required.



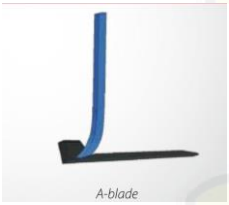


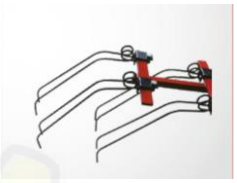
Idea 4: Inter and Intra

For inter weeding:

We have A blade or Weed harrow which pierces a little depth into soil.A blade covers more area than weed harrow but it requires more power compared to weed harrow.

Both will cover the area between the rows.





For intra weeding:

There are two L shaped blades attached to the bot, piercing little depth into the soil. As the bot moves forward, the L shaped blades are turning at an angle of 90° and coming back to 0° such that these blades covering the area between two plants.

In this way the L shaped blades are going to cut the weeds without harming the plants.

7 Simulation

Our main goal for simulation is to identified what type of effect is considerable during soil interaction with our equipment.

So we have research on it that which software is solve our soil simulation. So we have basically find two software:

1) ABAQUS

2) EDEM

7.1 Abaqus:

Abaqus is finite element analysis based software that offers powerful and complete solutions for both routine and sophisticated engineering problems covering a vast spectrum of industrial applications. In the automotive industry engineering work groups are able to consider full vehicle loads, dynamic vibration, multibody systems, impact/crash, nonlinear static, thermal coupling, and acoustic-structural coupling using a common model data structure and integrated solver technology. Best-in-class companies are taking advantage of Abaqus Unified FEA to consolidate their processes and tools, reduce costs and inefficiencies, and gain a competitive advantage. Abaqus is a heavy simulation software so it requires better configurations of laptop like requirement for graphic card and sufficiently powerful processor.

Installation guidance

Please follow the instructions in the video for installation process of ABAQUS.

https://www.youtube.com/watch?v=WA-jBPp4qtg&feature=share

Below are the steps for performing basic simulation in abaqus:

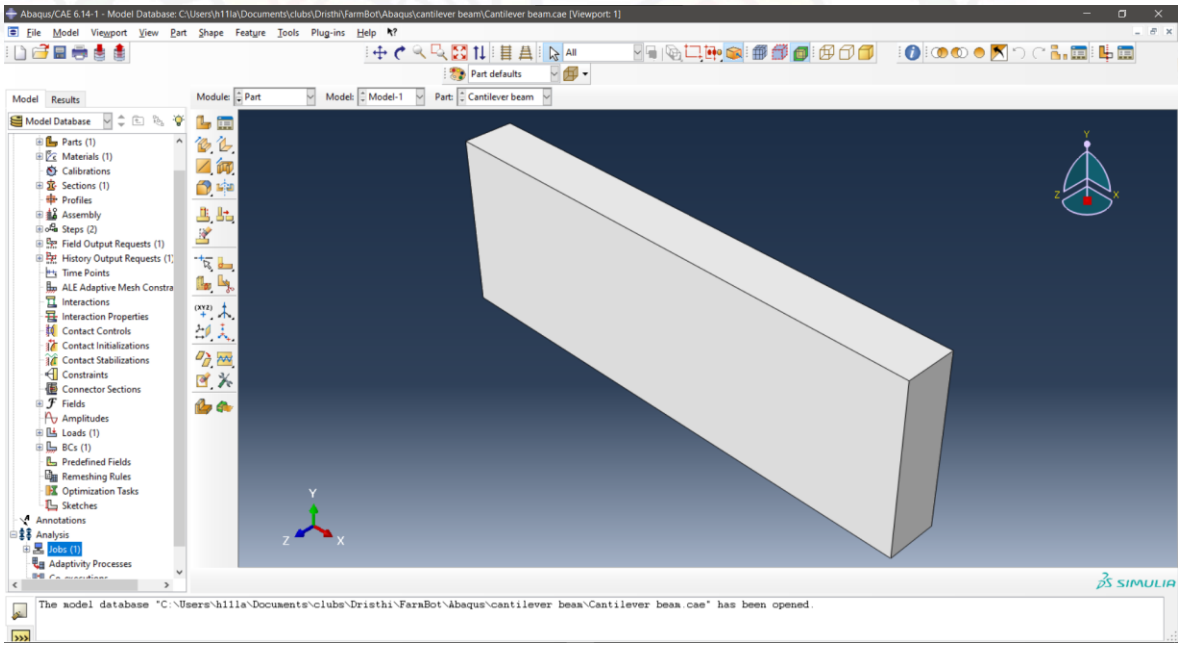
- First we need to draw our model using create part and the convert it into 3d design.
- Second it is required to provide material properties to the part. This properties require sufficient knowledge of material that we need to analyse.
- Third we need to provider step which provides that in how much step we need to apply the load.
- Using load we can provide different load condition to the model. Here we can also provide boundary conditions to the model.
- For any simulation we need to mesh the part using meshing. Using mesh setting we can control how much finer element analysis we require.
- Then using interaction we can provide how a part interact with respect to other part or certain reference.
- Also we can perform assembly inn abaqus. Assembly provides facilities to provide various constraints.

- After completing this steps we need to create job. In job we get option to decide how much processors to use, how much memory to allocate during simulation.
- After completing job creation we submit the job and after the job is simulated we can see the results.
- Abaqus can also allow importing models of different formats. For e.g we can import solidworks part file by converting it into parasolid.

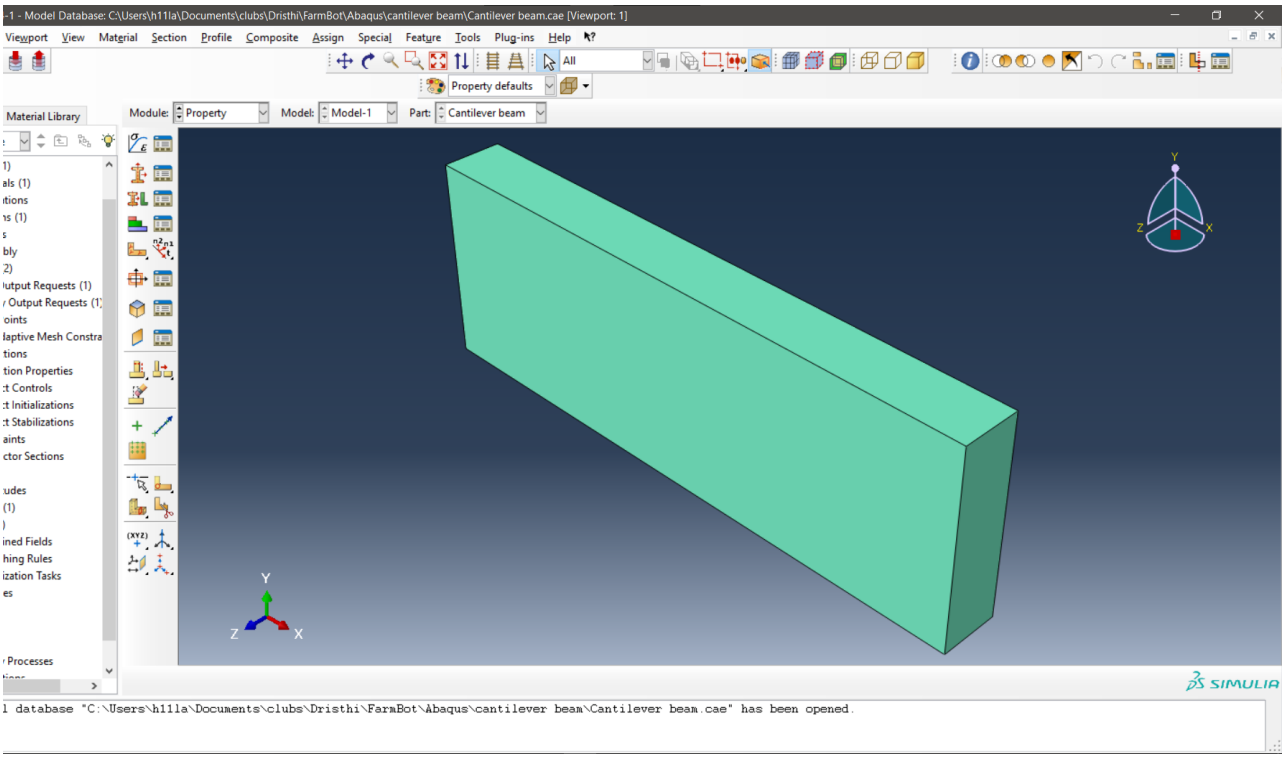
Simulation of beam bending:

Implementing the above steps to create a simple cantilever simulation. The steps are as follows:

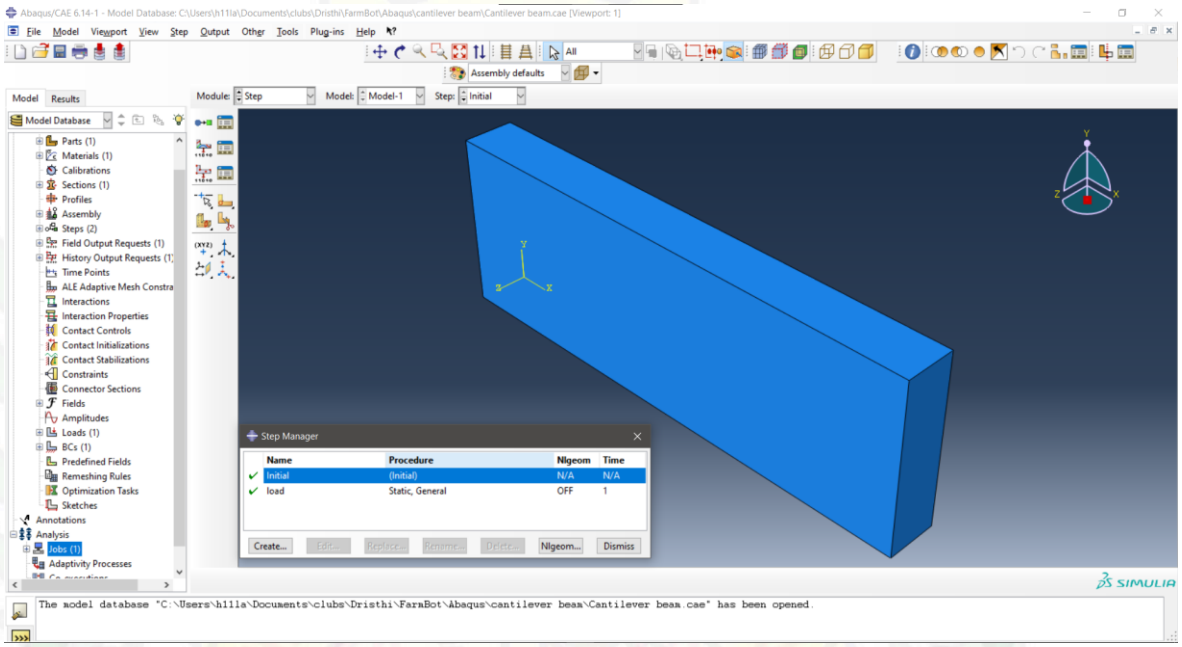
- First, we have to create a 2-D sketch with extrusion selected as the option to create a 3-D model. The model created will look like the following image:



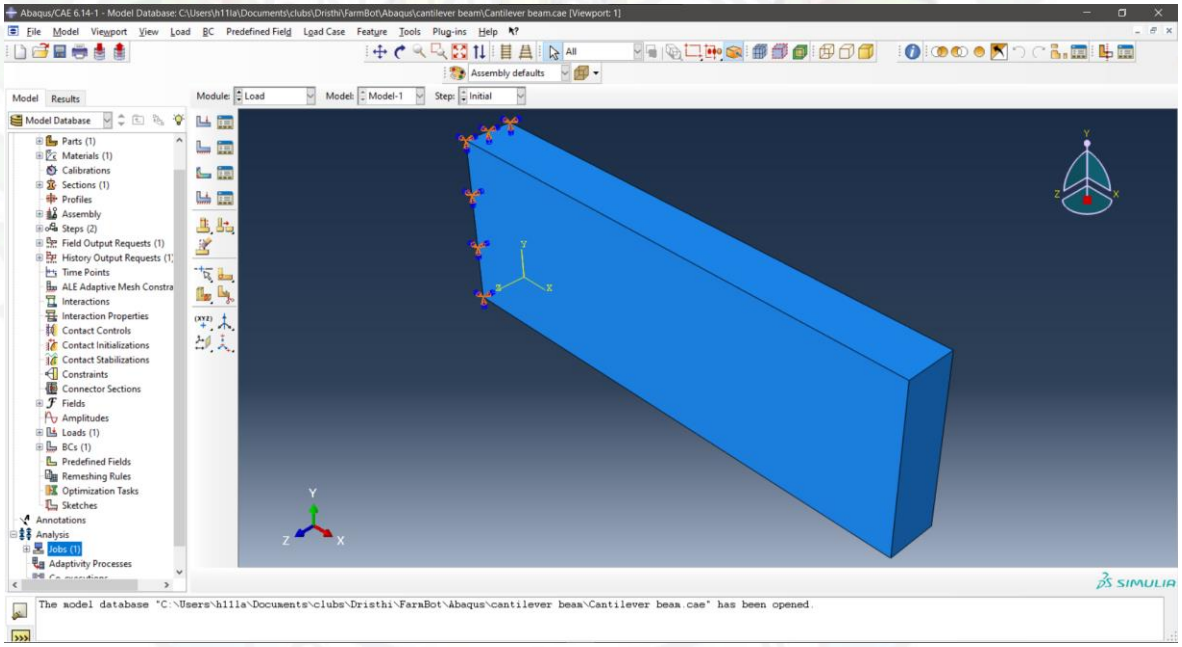
- By going to the properties tab, we apply the material we want to the bar by putting in parameters like Young's modulus, etc.



- We have created an instance so we can apply the load on the beam, also we can apply boundary conditions after this by going to the load section. This could be understood as creating a step in the process so that we can input our load parameters.

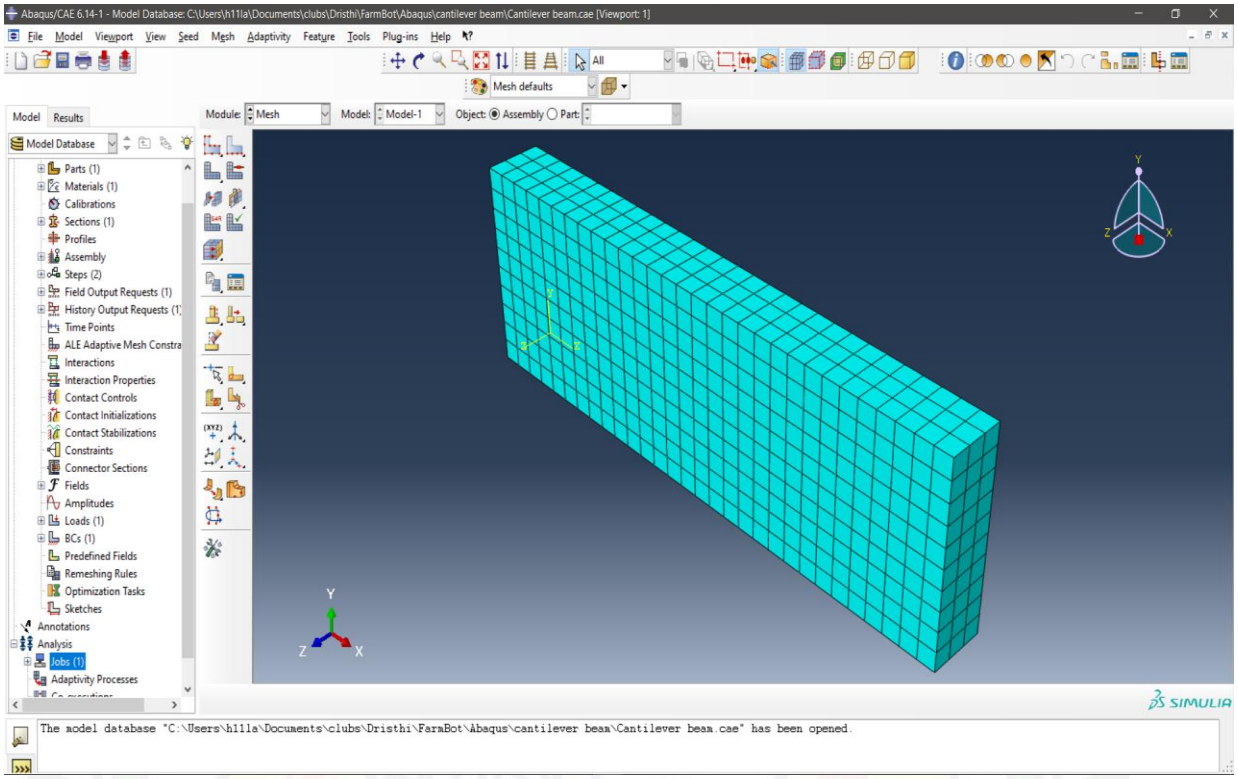


- Next by going to the load tab, we fix one face of the cantilever by encastrating it and applying load on the other face.



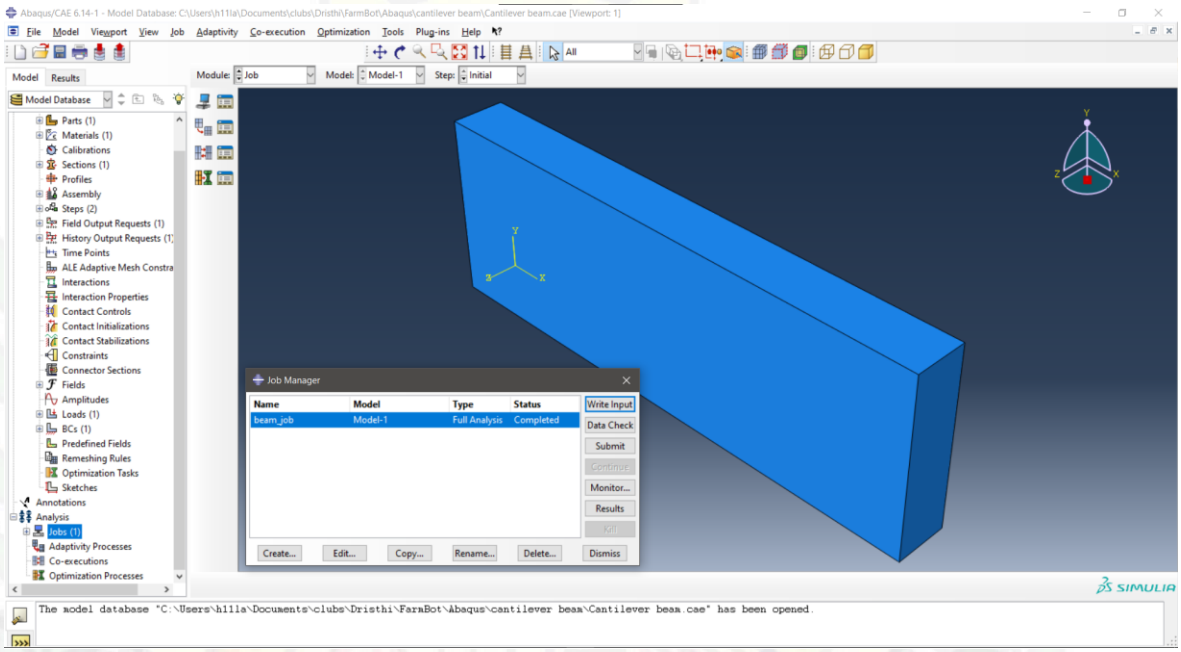
- As we are performing FEA analysis, we have to mesh our model. (Meshing is a process where we divide our model into numerous small units and intermediate

values are calculated and a rough function is developed which is then multiplied by a weighted function and integrated to get a more accurate result).



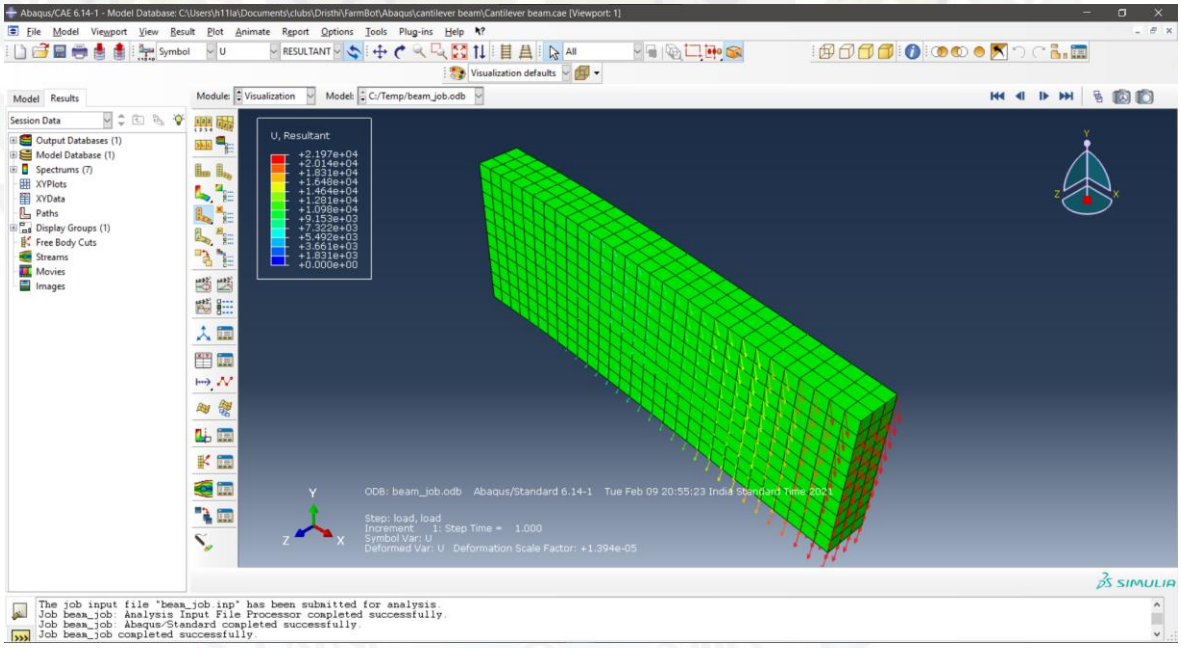
- Then we create a job and apply for a data check. If they are missing some data it will show error so that we can avoid it before the job submissions. Also we can edit the F-outputs settings to get additional info after processing our model.

- Finally we can submit our job, monitor while it is being processed and then finally see the results in the form of different graphical and visual representations.

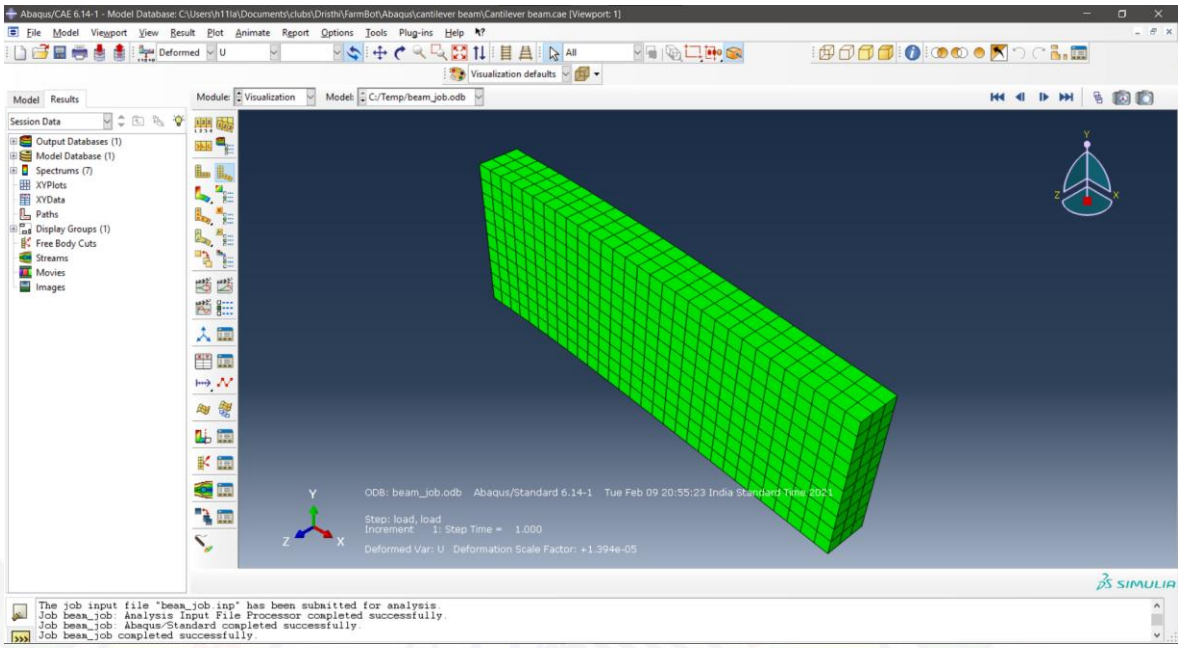


- The results will look as follows:

Force analysis of the beam:



Deformed unit:



Similarly, other results like stress, strain, and other results can also be derived.

Process for the disk digging mechanism

- First we have to import the parasolid assembly that created in the solid works
- Then we have to define the material for all the parts of assembly.(For the practise purpose we have followed same material as shown in the video)
- Now we have to create a step to perform the task
- Now we have to add the boundary conditions for making the slab fix.
- Then we apply a load for making the stress and other effects on the slab. (in this case we apply linear velocity and angular velocity to only disk
- Then we mesh the assembly part by part (Other than slab we have to provide tetrahedron meshing)
- Then we create a job and apply for a data check. If they are missing some data it will show error so that we can avoid it before the job submissions.
- Then after we submit the job. Also we can monitor the job which shows each and every detail such as energy consumed, error, warning, status, stage of the job.
- After successfully completing the job we can visualize the results of the job
- There are different types of the results that we can see such as stress, strain, deformation and many more.

7.2 EDEM:

EDEM is a discrete element simulation software based on discrete element theory of contact mechanics model. Because of the difference from traditional models, such as Bulk Mechanical Models, and Continuum Mechanics Models, the simulation results are not always consistent with the actual results in the simulation of granular material transport inconsistent. Thus, obtaining parameters is very important for the corresponding discrete element simulation. Angle of repose is the key parameter to characterize a material flow.

For learning edem ,We can take a basic introductory course from the edem learning module.(refrence:https://www.edemsimulation.com/courses/course-e2020-introduction-to-edem/).In this course we get basic idea of edem that which type of simulation we can get from this powerful software.It is purely based on DEM(discrete element method).After this course we tried different example from learning module like,**Screw Auger geometry and Rock Box geometry.** From this example we got more idea about how edem software work and also we got some idea how our simulation implement in software so that we can find our precise result.From this example and course we had not get good idea about how to make soil bed and how to move our geometry on soilbed.So we tried to make soil bed and after that we succeed to make just bed of different particle other than soil.so our research on carry on how to get soil particles in soil beds for soil particle we could find that edem itself give material library for diffrent materials an differernt soilstarterpack https://www.edemsimulation.com/blog-and-news/blog/simulating-soft-soils-edem-soils-starter-pack/#:~:text=The%20Soils%20Starter%20Pack%20has,gravels%20to%20soft%20compressible%20soils.

So basically we can get a good idea about how to access this soil starter pack to our simulation and after that we can try different examples that suit our simulation goal.

Key Terms and Steps while doing EDEM project

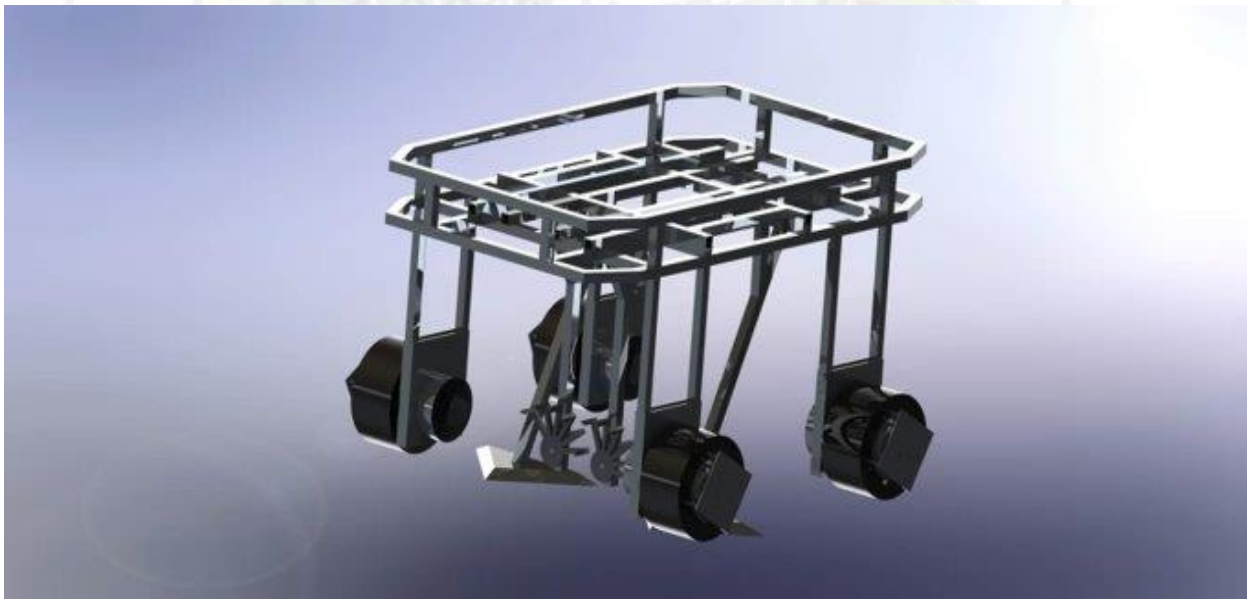
1. EDEM Creator : Setting up the model
2. EDEM Simulator : Running the simulation
3. EDEM Analyst : Exporting an input deck

Disk simulation:

Our main aim of this simulation is to show the disk rotating and moving on soil at a certain depth. We didn't get the best results of this simulation because this software takes approximately 30 min of time to complete 10 sec of simulation (it all depends on size of particle and end time). When you decrease the size of particles, the simulation increases more. Thus makes difficult for a normal processor to handle that much computation

The disk was only able to move the soil particles as shown in the below video .

https://youtu.be/lLasMKtn4WQ



8 Introduction of weed recognition

AIM:- To detect the weeds from the farm using a video camera feed.

9 Pre project research

9.1 Python

As python is a very easy learning language and very useful in machine learning, the machine learning side is kicked off by learning the python language.In that specially object oriented programming is covered.So by gaining the basic knowledge of python we started working on our project.
For the project following libraries are useful:

.Numpy

.openCV

.Matplotlib

.Pandas

.Tensorflow

.Keras

9.2 OpenCV

Since we have to work around images only so After learning python,we started learning Opencv,which is a huge open-source library for computer vision, machine learning, and image processing.

OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human.

For the project, the data augmentation is done using OpenCV functions. Following functions were used in data augmentation:

- Resizing
- Cropping
- Flipping

- Rotation
- Sharpening
- Scaling

The dataset collected is first resized by using Opencv. Here we resized all the images to 512*512 pixels.

To resize the images in the dataset the following function is used:

```
resized_image = cv2.resize(image,(512,512))
```

After resizing the dataset, it is augmented using the following necessary functions of OpenCV:

```
image1=cv2.add(image,m)
image2=cv2.subtract(image,m)
flipped_image = cv2.flip(image, 1)
sharpened_image = cv2.filter2D(image, -1, kernel_sharpening)
```

9.3 Regression

Before diving into the deep ocean of image datasets we worked on some simple datasets . This includes some csv files. We started making some of the basic machine learning models using algorithms including Linear regression and logistic regression.

Linear regression is the basic machine learning algorithm used to predict values for unseen data based on the model trained using some training dataset. It comes under supervised learning.

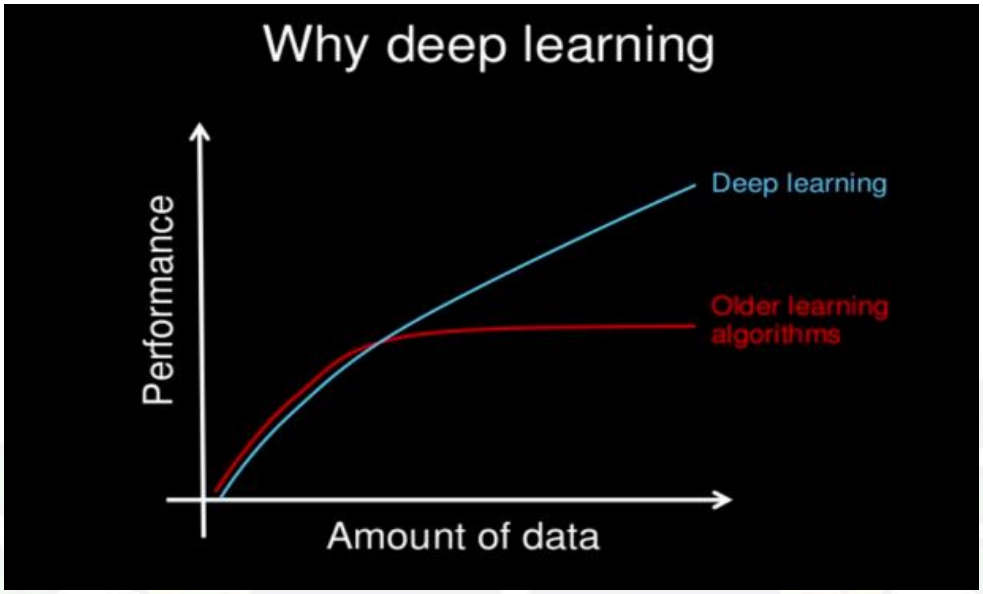
Similarly logistic regression is another model used to predict binary values(having values either 0 or 1).its like binary classification.After completing all these basics we finally come to the deep learning part .

9.4 Deep Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. Why is deep learning needed in our project although we have other machine

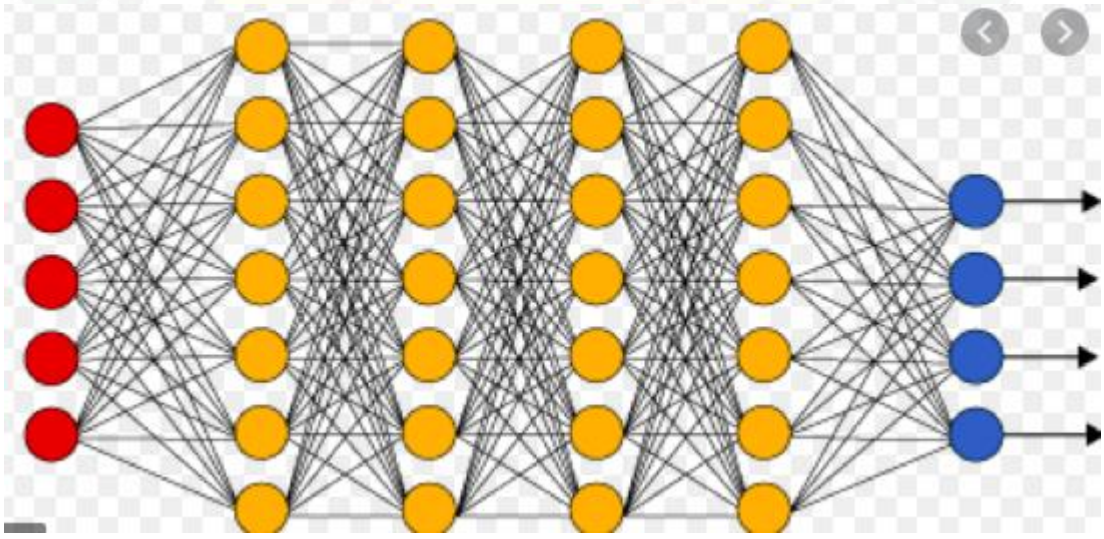
learning algorithms also ?

Ans-



Basically deep learning includes deep neural networks containing many layers and each layer contains some neurons. These neurons extract the features from the datasets and through the layers these features are transferred to the output layer and predictions are made.

Simple deep neural network having 5 layers



Deep learning applied in:

- Customer support

- Medical care
- Self-driving cars etc..

9.4.1 Deep learning frameworks

1. Tensorflow:
   TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources.

2. Pytorch
   pytorch is an optimized tensor library primarily used for Deep Learning applications using GPUs and CPUs. It is an open-source machine learning library for Python, mainly developed by the Facebook AI Research team. It is one of the widely used Machine learning libraries.

3. keras
   Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation.

9.4.2 terms related to deep learning

- Gradient descent:
- Activation function:
- Forward and backward propagation:

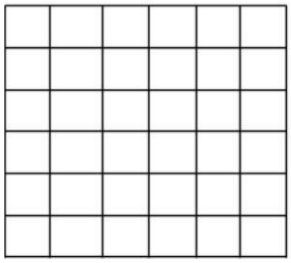
Since we need to classify images, we started working on convolutional neural networks.

9.5 Convolution Neural Networks

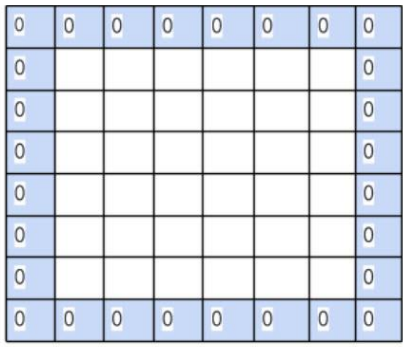
A Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data.

9.5.1 Terms related to CNN

- Padding:
  it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN. For example, if the padding in a CNN is set to zero, then every pixel value that is added will be of value zero. If, however, the zero padding is set to one, there will be a one pixel border added to the image with a pixel value of zero.



6x6 image



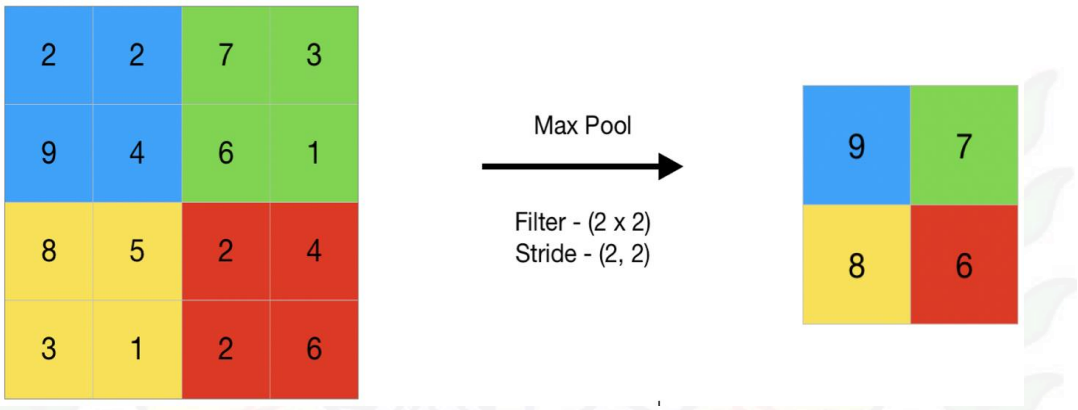
6x6 image with 1 layer of zero padding

- Strided convolutions:
  It is sometimes referred to as deconvolutions, transpose images, typically from a minimized format to a larger one. Imagine an image that has been reduced to a 2x2 pixel format. To transpose the image up to a larger format, a fractionally strided convolution reconstructs the image's spatial resolution, then performs the convolution.

- Pooling layer:
  Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.

  Max pooling is a pooling operation that selects the maximum element from the

region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.



9.5.2 Transfer Learning

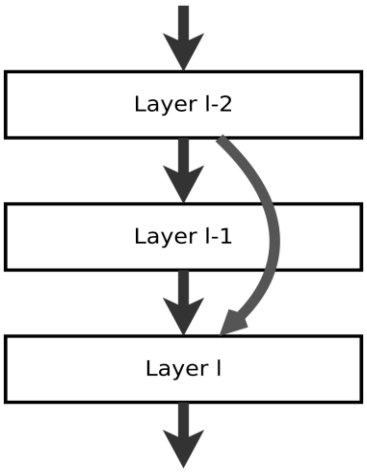
Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

9.5.3 Pre-Trained models

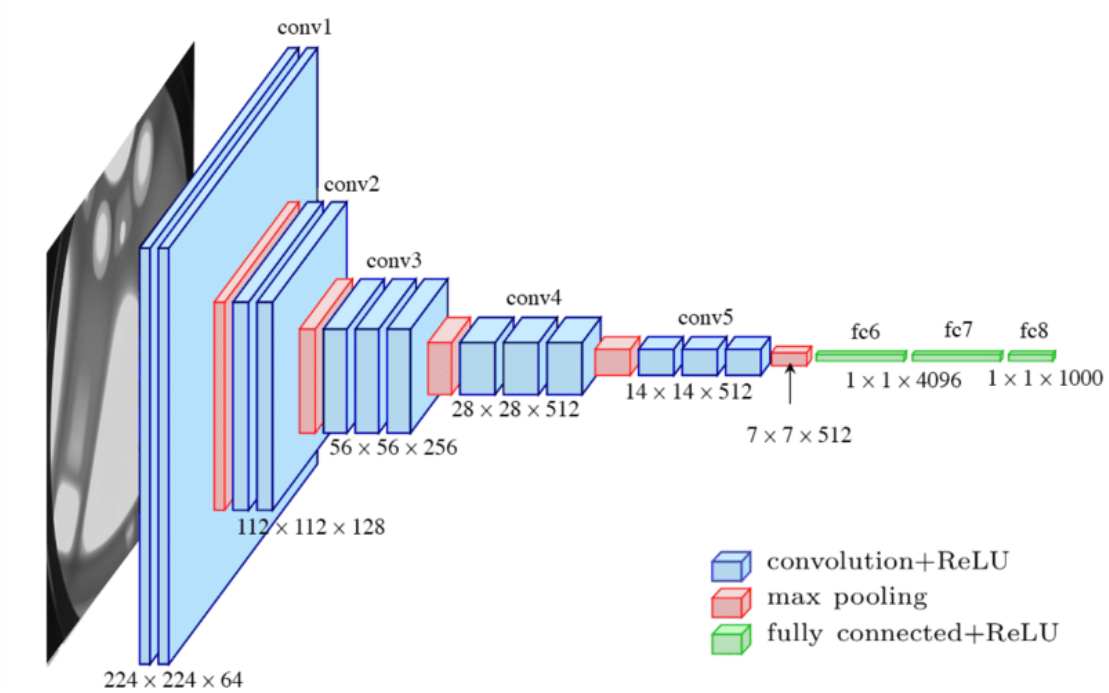
9.5.3.1 ResNet 50

A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers.



9.5.3.2 VGG Network:

VGG is a classical convolutional neural network architecture. It was based on an analysis of how to increase the depth of such networks. Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC(fully connected layers) followed by a softmax for output. The 16 in VGG16 refers to it having 16 layers that have weights. This network is a pretty large network and it has about 138 million (approx) parameters.



9.5.3.3 Inception

It is basically a convolutional neural network (CNN) which is 27 layers deep. Inception Layer is a combination of all those layers (namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage.

1×1 Convolutional layer before applying another layer, which is mainly used for dimensionality reduction.

A parallel Max Pooling layer, which provides another option to the inception layer.

9.5.3.4 Efficient 50

9.5.4 Some important terms in CNN.

- Batch size
  The batch size is a number of samples processed before the model is updated.The size of a batch must be more than or equal to one and less than or equal to the number of samples in the training dataset.

- Epoch
  The number of epochs indicates the number of passes of the entire training dataset the machine learning algorithm has completed.

- Convolution layer
  Convolution is a mathematical way of combining two signals to form a third signal. Here we used convolution to convolute the image with filters to obtain features in the image.applying convolution will result in shrinking of output image.

- Batch normalization
  Batch normalization is a layer that allows every layer of the network to do learning more independently. It is used to normalize the output of the previous layers. The activations scale the input layer in normalization. Using batch normalization learning becomes efficient also it can be used as regularization to avoid overfitting of the model. The layer is added to the sequential model to standardize the input or the outputs. It can be used at several points in between the layers of the model. It is often placed just after defining the sequential model and after the convolution and pooling layers.
- Layer Activation
  The activation function is a node that is put at the end of or in between Neural Networks. They help to decide if the neuron would fire or not. We have different types of activation functions, for example: Rectified Linear Unit (ReLU), Sigmoid function.

- Dropout
  Dropout is a regularization technique for neural network models. A Simple Way to Prevent Neural Networks from Overfitting.It is a technique where

randomly selected neurons are ignored during training. They are “dropped-out” randomly.

- Optimizers
  Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses. In simpler terms, optimizers shape and mold your model into its most accurate possible form by futzing with the weights.We have different types of optimizers.For example:Adam, RMSprop, SGD.

- Flatten
  Flatten layer converts M x N feature matrix into MN x 1 vector that is passed to the next layer/fully connected layer.

- Dense

Dense layer is a neural network that is connected deeply,which
Means each layer in the dense layer receives input from all
Neurons of its previous layer.it is the most commonly used layer.

10 Dataset Description and Augmentation

10.1 datasets

For training we use 3 types of datasets ,after training our model on different datasets available online ,the results were not that accurate as most of the images are top viewed and bot will take pictures from a certain angle so we manually collected the desired dataset and later augmented it using openCV.

initial dataset description : 3700 training images of crop and weed each .

740 testing images of crop and weed each.

Final dataset description : 4000 training images of crop and weed each

900 testing images of crop and weed each

We also tried our model with a combined initial and final dataset but the results were not satisfying so we went with the final dataset .

to increase our dataset , we augment our dataset .

10.2 Dataset augmentation techniques

- **Resizing**
- **Rotating**
- **Cropping**
- **Sharpening**
- **Flipping**

11 Models

11.1 Model 1

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 510, 510, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 510, 510, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 255, 255, 32)      0
_________________________________________________________________
dropout (Dropout)            (None, 255, 255, 32)      0
=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 510, 510, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 510, 510, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 255, 255, 32)      0
_________________________________________________________________
dropout (Dropout)            (None, 255, 255, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 253, 253, 64)      18496
_________________________________________________________________
activation_2 (Activation)    (None, 253, 253, 64)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 126, 126, 64)      0
_________________________________________________________________
dropout_1 (Dropout)          (None, 126, 126, 64)      0
_________________________________________________________________
flatten (Flatten)            (None, 1016064)           0
_________________________________________________________________
dense (Dense)                (None, 64)                65028160
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_4 (Activation)    (None, 1)                 0
=================================================================
Total params: 65,047,617
Trainable params: 65,047,617
Non-trainable params: 0
```

Accuracy: 99.8

Drawbacks/problem : model is overfit. Works fine with the seen data
But shows poor results with random google images.

Solution: add some dropout layers in between and make the model more complex by adding more layers.

MODEL 2:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 510, 510, 32)      896
_________________________________________________________________
activation (Activation)      (None, 510, 510, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 255, 255, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 253, 253, 32)      9248
_________________________________________________________________
activation_1 (Activation)    (None, 253, 253, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 126, 126, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 124, 124, 64)      18496
_________________________________________________________________
activation_2 (Activation)    (None, 124, 124, 64)      0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 62, 62, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 246016)            0
_________________________________________________________________
dense (Dense)                (None, 64)                15745088
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_4 (Activation)    (None, 1)                 0
=================================================================
Total params: 15,773,793
Trainable params: 15,773,793
Non-trainable params: 0
```

Accuracy: 98.9

Drawback: model was better than the previous one but it overfits ag. Works fine with the seen data but shows poor results with random google images.

Solution: Add some dropout layers in between and make the model more complex by adding more layers.

11.3 Model 3

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 510, 510, 32)      896
_________________________________________________________________
activation (Activation)      (None, 510, 510, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 255, 255, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 253, 253, 32)      9248
_________________________________________________________________
activation_1 (Activation)    (None, 253, 253, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 126, 126, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 124, 124, 32)      9248
_________________________________________________________________
activation_2 (Activation)    (None, 124, 124, 32)      0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 62, 62, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 60, 60, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 60, 60, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 57600)             0
_________________________________________________________________
dense (Dense)                (None, 64)                3686464
_________________________________________________________________
activation_4 (Activation)    (None, 64)                0
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_5 (Activation)    (None, 1)                 0
=================================================================
Total params: 3,724,417
Trainable params: 3,724,417
Non-trainable params: 0
```

Accuracy: 94.24

Drawbacks/problem : model was overfitted.

Solution: Add one dropout layer at the end of the third layer.

With dropout=0.2, accuracy was 95.09.

With dropout=0.25, accuracy was 97.82.

When dropout and batch normalisation were added, accuracy was 89.96.

After applying our dataset to all these models still we have not got the results we want, so we shift to some pretrained models.

11.4 Pre-Trained models

1. **RESNET50 :**

```
base_model = Sequential()
base_model.add(ResNet50(include_top=False, weights='imagenet', pooling='max'))
base_model.add(Dense(1, activation='sigmoid'))
base_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 2048)              23587712
_________________________________________________________________
dense (Dense)                (None, 1)                 2049
=================================================================
Total params: 23,589,761
Trainable params: 23,536,641
Non-trainable params: 53,120
_________________________________________________________________
```

Accuracy : 87.33

Results: when trained 1st time results are very impressive

9 out of 10 images were detected correctly.

When trained 2nd time results were opposite to the one we trained earlier. The model treated all images as crops.

So we need to apply seed to our models so that everytime we train our model the weight initialisation values will not change.

2. Vgg16:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 512, 512, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 512, 512, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 512, 512, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 256, 256, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 256, 256, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 256, 256, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 128, 128, 128)     0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 128, 128, 256)     295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 128, 128, 256)     590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 128, 128, 256)     590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 64, 64, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 64, 64, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 64, 64, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 64, 64, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 32, 32, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 16, 16, 512)       0
_________________________________________________________________
flatten (Flatten)            (None, 131072)            0
_________________________________________________________________
dense (Dense)                (None, 512)               67109376
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 81,824,577
Trainable params: 67,109,889
Non-trainable params: 14,714,688
_________________________________________________________________
```

Accuracy: 95.44

Results: Results were quite impressive . 9 out of 12 images were detected correctly.

3.EfficientNet50:

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 512, 512, 3)	0	
stem_conv (Conv2D)	(None, 256, 256, 32)	864	input_1[0][0]
stem_bn (BatchNormalization)	(None, 256, 256, 32)	128	stem_conv[0][0]
stem_activation (Activation)	(None, 256, 256, 32)	0	stem_bn[0][0]
block1a_dwconv (DepthwiseConv2D	(None, 256, 256, 32)	288	stem_activation[0][0]
block1a_bn (BatchNormalization)	(None, 256, 256, 32)	128	block1a_dwconv[0][0]
block1a_activation (Activation)	(None, 256, 256, 32)	0	block1a_bn[0][0]
block1a_se_squeeze (GlobalAvera	(None, 32)	0	block1a_activation[0][0]
block1a_se_reshape (Reshape)	(None, 1, 1, 32)	0	block1a_se_squeeze[0][0]
block1a_se_reduce (Conv2D)	(None, 1, 1, 8)	264	block1a_se_reshape[0][0]
block1a_se_expand (Conv2D)	(None, 1, 1, 32)	288	block1a_se_reduce[0][0]
block1a_se_excite (Multiply)	(None, 256, 256, 32)	0	block1a_activation[0][0] block1a_se_expand[0][0]
block1a_project_conv (Conv2D)	(None, 256, 256, 16)	512	block1a_se_excite[0][0]
block1a_project_bn (BatchNormal	(None, 256, 256, 16)	64	block1a_project_conv[0][0]
block2a_expand_conv (Conv2D)	(None, 256, 256, 96)	1536	block1a_project_bn[0][0]
block2a_expand_bn (BatchNormali	(None, 256, 256, 96)	384	block2a_expand_conv[0][0]
block2a_expand_activation (Acti	(None, 256, 256, 96)	0	block2a_expand_bn[0][0]
block2a_dwconv (DepthwiseConv2D	(None, 128, 128, 96)	864	block2a_expand_activation[0][0]
block2a_bn (BatchNormalization)	(None, 128, 128, 96)	384	block2a_dwconv[0][0]
block2a_activation (Activation)	(None, 128, 128, 96)	0	block2a_bn[0][0]
block2a_se_squeeze (GlobalAvera	(None, 96)	0	block2a_activation[0][0]
block2a_se_reshape (Reshape)	(None, 1, 1, 96)	0	block2a_se_squeeze[0][0]
block2a_se_reduce (Conv2D)	(None, 1, 1, 4)	388	block2a_se_reshape[0][0]
block2a_se_expand (Conv2D)	(None, 1, 1, 96)	480	block2a_se_reduce[0][0]
block2a_se_excite (Multiply)	(None, 128, 128, 96)	0	block2a_activation[0][0] block2a_se_expand[0][0]
block2a_project_conv (Conv2D)	(None, 128, 128, 24)	2304	block2a_se_excite[0][0]
block2a_project_bn (BatchNormal	(None, 128, 128, 24)	96	block2a_project_conv[0][0]
block2b_expand_conv (Conv2D)	(None, 128, 128, 144	3456	block2a_project_bn[0][0]

▪ ▪ ▪ ▪ ▪ ▪

block6d_se_reduce (Conv2D)	(None, 1, 1, 48)	55344	block6d_se_reshape[0][0]
block6d_se_expand (Conv2D)	(None, 1, 1, 1152)	56448	block6d_se_reduce[0][0]
block6d_se_excite (Multiply)	(None, 16, 16, 1152)	0	block6d_activation[0][0] block6d_se_expand[0][0]
block6d_project_conv (Conv2D)	(None, 16, 16, 192)	221184	block6d_se_excite[0][0]
block6d_project_bn (BatchNormal	(None, 16, 16, 192)	768	block6d_project_conv[0][0]
block6d_drop (FixedDropout)	(None, 16, 16, 192)	0	block6d_project_bn[0][0]
block6d_add (Add)	(None, 16, 16, 192)	0	block6d_drop[0][0] block6c_add[0][0]
block7a_expand_conv (Conv2D)	(None, 16, 16, 1152)	221184	block6d_add[0][0]
block7a_expand_bn (BatchNormali	(None, 16, 16, 1152)	4608	block7a_expand_conv[0][0]
block7a_expand_activation (Acti	(None, 16, 16, 1152)	0	block7a_expand_bn[0][0]
block7a_dwconv (DepthwiseConv2D	(None, 16, 16, 1152)	10368	block7a_expand_activation[0][0]
block7a_bn (BatchNormalization)	(None, 16, 16, 1152)	4608	block7a_dwconv[0][0]
block7a_activation (Activation)	(None, 16, 16, 1152)	0	block7a_bn[0][0]
block7a_se_squeeze (GlobalAvera	(None, 1152)	0	block7a_activation[0][0]
block7a_se_reshape (Reshape)	(None, 1, 1, 1152)	0	block7a_se_squeeze[0][0]
block7a_se_reduce (Conv2D)	(None, 1, 1, 48)	55344	block7a_se_reshape[0][0]
block7a_se_expand (Conv2D)	(None, 1, 1, 1152)	56448	block7a_se_reduce[0][0]
block7a_se_excite (Multiply)	(None, 16, 16, 1152)	0	block7a_activation[0][0] block7a_se_expand[0][0]
block7a_project_conv (Conv2D)	(None, 16, 16, 320)	368640	block7a_se_excite[0][0]
block7a_project_bn (BatchNormal	(None, 16, 16, 320)	1280	block7a_project_conv[0][0]
top_conv (Conv2D)	(None, 16, 16, 1280)	409600	block7a_project_bn[0][0]
top_bn (BatchNormalization)	(None, 16, 16, 1280)	5120	top_conv[0][0]
top_activation (Activation)	(None, 16, 16, 1280)	0	top_bn[0][0]
flatten (Flatten)	(None, 327680)	0	top_activation[0][0]
dense (Dense)	(None, 1024)	335545344	flatten[0][0]
dropout (Dropout)	(None, 1024)	0	dense[0][0]
dense_1 (Dense)	(None, 1)	1025	dropout[0][0]

Total params: 339,595,933
Trainable params: 335,546,369
Non-trainable params: 4,049,564

Accuracy: 94.17

Results: not impressive. Model predicts most of the test images as of weed category.

4. InceptionV3:

```
Model: "model_1"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 512, 512, 3)      0
_________________________________________________________________________________________________
conv2d (Conv2D)              (None, 255, 255, 32)      864         input_1[0][0]
_________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 255, 255, 32)   96          conv2d[0][0]
_________________________________________________________________________________________________
activation (Activation)      (None, 255, 255, 32)      0           batch_normalization[0][0]
_________________________________________________________________________________________________
conv2d_1 (Conv2D)            (None, 253, 253, 32)      9216        activation[0][0]
_________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 253, 253, 32)   96          conv2d_1[0][0]
_________________________________________________________________________________________________
activation_1 (Activation)    (None, 253, 253, 32)      0           batch_normalization_1[0][0]
_________________________________________________________________________________________________
conv2d_2 (Conv2D)            (None, 253, 253, 64)      18432       activation_1[0][0]
_________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 253, 253, 64)   192         conv2d_2[0][0]
_________________________________________________________________________________________________
activation_2 (Activation)    (None, 253, 253, 64)      0           batch_normalization_2[0][0]
_________________________________________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 126, 126, 64)      0           activation_2[0][0]
_________________________________________________________________________________________________
conv2d_3 (Conv2D)            (None, 126, 126, 80)      5120        max_pooling2d[0][0]
_________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 126, 126, 80)   240         conv2d_3[0][0]
_________________________________________________________________________________________________
activation_3 (Activation)    (None, 126, 126, 80)      0           batch_normalization_3[0][0]
_________________________________________________________________________________________________
conv2d_4 (Conv2D)            (None, 124, 124, 192 138240      activation_3[0][0]
_________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 124, 124, 192 576         conv2d_4[0][0]
_________________________________________________________________________________________________
activation_4 (Activation)    (None, 124, 124, 192 0           batch_normalization_4[0][0]
_________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 61, 61, 192)  0           activation_4[0][0]
_________________________________________________________________________________________________
conv2d_8 (Conv2D)            (None, 61, 61, 64)    12288       max_pooling2d_1[0][0]
_________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, 61, 61, 64)    192         conv2d_8[0][0]
_________________________________________________________________________________________________
activation_8 (Activation)    (None, 61, 61, 64)    0           batch_normalization_8[0][0]
_________________________________________________________________________________________________
conv2d_6 (Conv2D)            (None, 61, 61, 48)    9216        max_pooling2d_1[0][0]
_________________________________________________________________________________________________
conv2d_9 (Conv2D)            (None, 61, 61, 96)    55296       activation_8[0][0]
_________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 61, 61, 48)    144         conv2d_6[0][0]
_________________________________________________________________________________________________
batch_normalization_9 (BatchNor (None, 61, 61, 96)    288         conv2d_9[0][0]
_________________________________________________________________________________________________
activation_6 (Activation)    (None, 61, 61, 48)    0           batch_normalization_6[0][0]
_________________________________________________________________________________________________
activation_9 (Activation)    (None, 61, 61, 96)    0           batch_normalization_9[0][0]
_________________________________________________________________________________________________
average_pooling2d (AveragePooli (None, 61, 61, 192)  0           max_pooling2d_1[0][0]
_________________________________________________________________________________________________
conv2d_5 (Conv2D)            (None, 61, 61, 64)    12288       max_pooling2d_1[0][0]
_________________________________________________________________________________________________
```

■ ■ ■ ■ ■ ■

conv2d_87 (Conv2D)	(None, 14, 14, 384)	442368	activation_86[0][0]
conv2d_88 (Conv2D)	(None, 14, 14, 384)	442368	activation_86[0][0]
conv2d_91 (Conv2D)	(None, 14, 14, 384)	442368	activation_90[0][0]
conv2d_92 (Conv2D)	(None, 14, 14, 384)	442368	activation_90[0][0]
average_pooling2d_8 (AveragePoo	(None, 14, 14, 2048)	0	mixed9[0][0]
conv2d_85 (Conv2D)	(None, 14, 14, 320)	655360	mixed9[0][0]
batch_normalization_87 (BatchNo	(None, 14, 14, 384)	1152	conv2d_87[0][0]
batch_normalization_88 (BatchNo	(None, 14, 14, 384)	1152	conv2d_88[0][0]
batch_normalization_91 (BatchNo	(None, 14, 14, 384)	1152	conv2d_91[0][0]
batch_normalization_92 (BatchNo	(None, 14, 14, 384)	1152	conv2d_92[0][0]
conv2d_93 (Conv2D)	(None, 14, 14, 192)	393216	average_pooling2d_8[0][0]
batch_normalization_85 (BatchNo	(None, 14, 14, 320)	960	conv2d_85[0][0]
activation_87 (Activation)	(None, 14, 14, 384)	0	batch_normalization_87[0][0]
activation_88 (Activation)	(None, 14, 14, 384)	0	batch_normalization_88[0][0]
activation_91 (Activation)	(None, 14, 14, 384)	0	batch_normalization_91[0][0]
activation_92 (Activation)	(None, 14, 14, 384)	0	batch_normalization_92[0][0]
batch_normalization_93 (BatchNo	(None, 14, 14, 192)	576	conv2d_93[0][0]
activation_85 (Activation)	(None, 14, 14, 320)	0	batch_normalization_85[0][0]
mixed9_1 (Concatenate)	(None, 14, 14, 768)	0	activation_87[0][0] activation_88[0][0]
concatenate_1 (Concatenate)	(None, 14, 14, 768)	0	activation_91[0][0] activation_92[0][0]
activation_93 (Activation)	(None, 14, 14, 192)	0	batch_normalization_93[0][0]
mixed10 (Concatenate)	(None, 14, 14, 2048)	0	activation_85[0][0] mixed9_1[0][0] concatenate_1[0][0] activation_93[0][0]
flatten_1 (Flatten)	(None, 401408)	0	mixed10[0][0]
dense_2 (Dense)	(None, 1024)	411042816	flatten_1[0][0]
dropout_1 (Dropout)	(None, 1024)	0	dense_2[0][0]
dense_3 (Dense)	(None, 1)	1025	dropout_1[0][0]

Total params: 432,846,625
Trainable params: 411,043,841
Non-trainable params: 21,802,784

Accuracy: 95.83

Results: results were decent but not as good as vgg16 and resnet.

After testing our dataset to all these models we decided to go with the vgg16 model.

12 Video testing

After successful training and testing of model on image dataset, we tested our model on video frames.

We manually filmed a video having crops and weeds in it.

Testing Process includes :

- Video loading
- Capture frame-by-frame.
- Saves image of the current frame in jpg file
- Testing each frame and printing its class.

The results were quite similar to what we required.

13 Results

```
img_pred = image.load_img('/content/drive/MyDrive/test/weedtest1.jpg',target_size=(512,512))
img_pred = image.img_to_array(img_pred)
img_pred = np.expand_dims(img_pred , axis = 0)
```

```
[10] result = new_model.predict(img_pred)
     print(result)

     if result[0][0] ==1:
         prediction = "weed"

     else:
         prediction = "crop"

     print(prediction)
```

```
[[1.]]
weed
```

```
[ ]
```

1. Installation and Workspace:

1.1 UBUNTU INSTALLATION

To install ubuntu dual boot with window certain steps are to be followed:

Step1:

Download ubuntu 20.04 from the link below:
https://ubuntu.com/download/desktop.

Step2:

Take backup of windows.It would be helpful if somehow the system would be messed up.

Step3:

Third step is to make pendrive/disc bootable- this can be done using software rufus or belena etcher.

Step4:

Make a partition where ubuntu will be installed - we need to make a partition to install ubuntu . 80GB will be enough to install ubuntu.

To make partition, path is:
Control panel -> disk management tool.
Inside the disk management tool, right click on the drive which you want to make partition and select the volume shrink option.

NOTE: If possible try to make the partition in C-drive. It will save you from irrelevant errors.

After partition the next step is to boot the system with ubuntu using drive. For this plug-in the USB and boot the system.Then restart the system and press f9/f10 keys . It will open a window . Click on continue and further choose accordingly . It will install Ubuntu to the system.
Once ubuntu is installed , to change the operating system restart the pc and press f9/f10 and choose ubuntu as operating system.

1.2 ROS INSTALLATION

To install ROS in Ubuntu check the following link given:
http://wiki.ros.org/Installation/Ubuntu

To check whether it is successfully installed or not run command: ‘roscore’ in

terminal.If its last line is something like: ‘started core service [/rosout]’, then it is installed successfully.

1.3 CREATING WORKSPACE AND PACKAGES

To create a ROS Workspace, we follow a set of commands.
These codes include initialising it and adding some necessary files.

mkdir -p ~/my_ws/src #this creates a workspace directory names my_ws

cd ~/my_ws/src #the my_ws directory contains a src file in it for code

catkin_init_workspace #this command creates a CMakeLists.txt file for you in the src directory

cd .. #Navigate back to the workspace root directory.
Catkin_make #This command will create two new directories named “build” and “devel” and also compile the workspace.

source devel/setup.bash #Now will configure our system to use this workspace. This is done by setting up the devel file. source devel/setup.bash (source is an important command and is also very necessary.)

1.4 ROS Packages

To make ROS packages execute the following codes

cd ~/catkin_ws/src # Navigate to the src directory.

catkin_create_pkg my_pkg rospy #This command create a package with name my_pkg and python being the accepted programming language of this package for creating publishers, subscribers and all.

2. Publisher and Subscriber

Sample code for publishing to a topic:

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import Int32
rospy.init_node('topic_publisher')
rate = rospy.Rate(2)
count = 0
pub = rospy.Publisher('counter', Int32)

while not rospy.is_shutdown():
    pub.publish(count)
    count+= 1
    rate.sleep()
```

Sample code for subscribing to a topic:

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import Int32
def callback(msg):
print msg.data
rospy.init_node('topic_subscriber')
sub = rospy.Subscriber('counter', Int32, callback)
rospy.spin()
```

Before running any python file, make sure that it has execute permission. Use this command.

```
user@hostname$ chmod u + x filename
```

We can see all the topics that are running in ROS simply by the command below.

```
user@hostname$ rostopic list
```

3. Creating Custom messages:

We created the custom messages when we want to publish too many message types at a time and don't want to create individual publishers.

To create your own message type follow the steps below:

- Create a file with your defined message type as below and save with the .msg extension.

  float32 real
  float32 imaginary

- Add the lines below to your Package.xml file.

  <build_depend>message_generation</build_depend>
  <run_depend>message_runtime</run_depend>

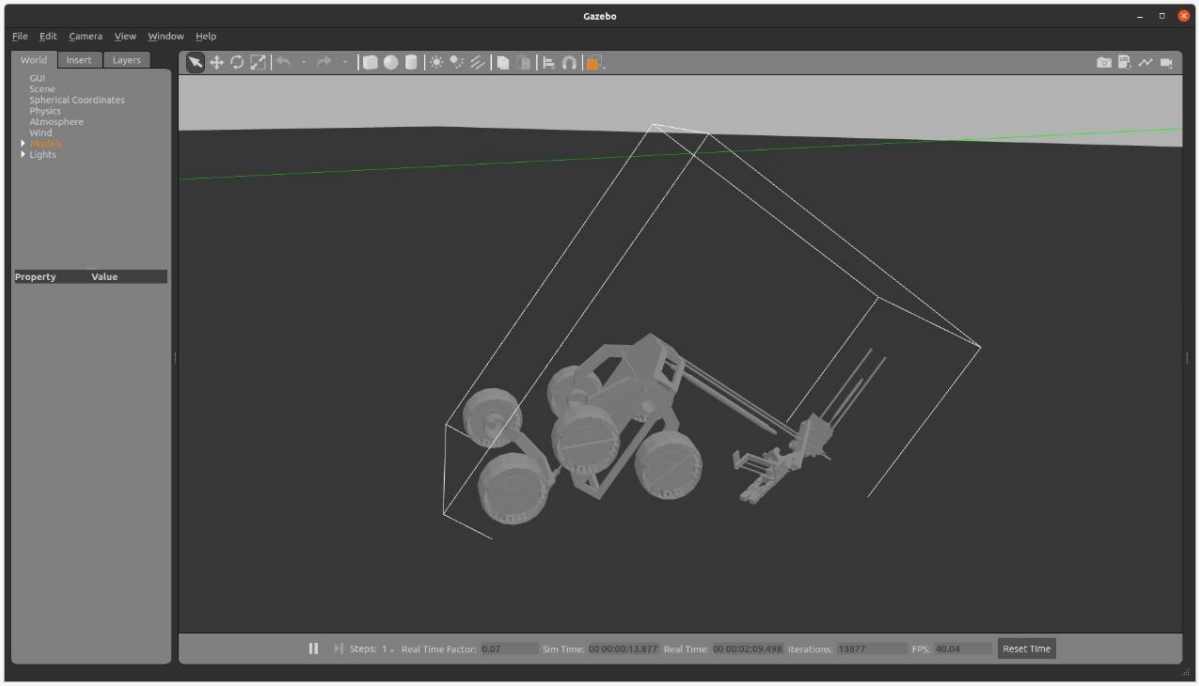
- Add “message_generation” at the end of the find_package() function in the CmakeLists.txt file.
- Add “message_runtime” at the end of the catkin_package() function in the CmakeLists.txt file.
- Add the name of the file in the add_message_files() function in CmakeLists.txt file.
- Finally uncomment the generate_messages() function and then we are all set for catkin_make.

4. Parsing the package in gazebo:

- Create the workspace as discussed above.
- In the src directory load the package directory. The package directory should include directories like urdf, meshes, config, launch.
- Navigate to the workspace directory then run the command “catkin_make”.
- If it does not show any errors then the package has parsed successfully.

Ignoring transform for child_frame_id "arm_1" from authority "unknown_publisher" because of a nan value in the transform (-nan -nan -nan) (0.000000 0.000000 0.000000 1.000000)

Ans. The problem is with the CMakeLists.txt and the Package.xml file in the parsed Package. While loading the package in the src directory of workspace, load the CMakeLists.txt and Package.xml files of the same workspace. **Do not replace the files with the ones you created while creating the package** and copying the given package files.



The model shown in figure is broken because we haven't set the limits of every joint in urdf. When the limits of every joint in urdf is correct as per the real world objectives then it won't appear as now.

5. Controlling the model in Gazebo:

To Control the model in gazebo we need to make some changes in .urdf file, gazebo.launch file, .yaml file and those changes are as follows:

5.1 Changes in .urdf file:

Adding Gazebo Plugin:

A Gazebo plugin needs to be added to your URDF that actually parses the transmission tags and loads the appropriate hardware interfaces and controller manager.

The syntax is as follows:

```
<gazebo>
 <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
  <robotNamespace>/model_name</robotNamespace>
 </plugin>
</gazebo>
```

Adding Transmissions:

The <transmission> element is used to link actuators to joints.

The syntax is as follows.

```
<transmission name="tran1">
<type>transmission_interface/SimpleTransmission</type>
<joint name="joint1">
                <hardwareInterface>EffortJointInterface</hardwareInterface>
</joint>
<actuator name="motor1">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
<mechanicalReduction>1</mechanicalReduction>
</actuator>
</transmission>
```

5.2 Changes in .yaml file:

The PID gains and controller settings must be saved in a yaml file that gets loaded to the param server via the roslaunch file. In the config folder of your package, make the following changes to your .yaml file.

1). Make a file in the config folder and its syntax will be as follows:

model_name(same as in urdf):

```
joint_state_controller:
type: joint_state_controller/JointStateController
publish_rate: 20

controller_name_1:
type: effort_controllers/JointGroupEffortController
joints: ['joint_name_1', 'joint_name_2']

controller_name_2:
type: effort_controllers/JointGroupEffortController
joints: ['joint_name_3', 'joint_name_4']
```

Further we'll tune the output using the PID controller.

5.3 Changes in .launch file:

1). Launch file provides a convenient way to start up multiple nodes and a master as well as other initialization requirements such as setting parameters.

We use the spawner tool to automatically load, start, stop and unload a controller from within a launch file. When we start spawner, it will load and start the controller. When we stop spawner (when the launch file is taken down) it will stop and unload the controller.

The robot state publisher helps us to broadcast the state of our robot to the tf transform library. The robot state publisher internally has a kinematic model of the robot; so given the joint positions of the robot, the robot state publisher can compute and broadcast the 3D pose of each link in the robot.

Now we would have to make a node in launch file and its syntax is as follows:

```
<!-- loads the controllers -->
<rosparam
file="$(find farm_bot)/config/gazebofb.yaml"
command="load" />


<node name="controller_spawner"
pkg="controller_manager"
type="spawner" ns="/fb_model"
args="r_con_position_controller l_con_position_controller
joint_state_controller"/>
```

<!-- converts joint states to TF transforms -->

<node name="robot_state_publisher"
pkg="robot_state_publisher"
type="robot_state_publisher" respawn="false" output="screen">
<remap from="/joint_states" to="/fb_model/joint_states" />

</node>

Warning: Controller Spawner couldn't find the expected controller_manager ROS interface.

Packages to be Installed for running Gazebo

1] four_wheel_steering_msgs

ROS messages for vehicles using four wheel steering.

Source: https://github.com/ros-drivers/four_wheel_steering_msgs.git

2] urdf_geometry_parser

It will extract the geometry value of a vehicle from urdf.

Source: https://github.com/ros-controls/urdf_geometry_parser.git

3] ros_controllers

Source: https://github.com/ros-controls/ros_controllers.git

We used gazebo_ros_pkgs package for rectifying the error:

Source: http://gazebosim.org/tutorials?cat=connect_ros

THE ABOVE PACKAGES ARE INSTALLED USING GIT CLONE FUNCTION.

To check the motors and controllers just execute ‘rostopic list’ while gazebo running and then execute the command ‘rostopic pub -1 topic_name std_msgs/msg_type “data:value”.

If the code works then make a python file in src of the package that takes input from the keyboard using **getkey** function and the model moves accordingly.

As the accuracy was very less and that will create a lot of disturbance in the real world and hence we thought to add PID controllers to the ROS.

6. Tuning using PID controllers:

The term PID stands for **P**roportional **I**ntegral **D**erivative. In this controller, a control loop feedback device is used to regulate all the process variables.
For an Example: If we are riding a bike and we want to maintain a constant velocity of 30 kmph then we would have to accelerate and as the vehicle increases its speed and overshoots we would decrease the acceleration and the speed falls below 30 with a less error than before, this continues and the error becomes negligible at last. This is the basic working principle of PID controllers.

P controller : P controller is mostly used in first order processes with single energy storage to stabilize the unstable process. The main usage of the P controller is to decrease the steady state error of the system.As the proportional gain factor Kp increases, the steady state error of the system decreases.
$P = K_p \times e(t)$; e(t): instantaneous error; Kp: Proportionality constant.

I controller : An **integral controller** (also called reset **controller**) can eliminate the steady-state error that occurs with a proportional **controller.**

Integral gain (Ki) - This gain helps compensate for changing system dynamics, such as varying loads, and often aids the axis in rapidly reaching the Command Position or Command Velocity.

$I = K_i \times \int_0^t (e(t) \times dt)$; e(t):error starting from initial time; Ki : Integral gain.

D controller: D stands for derivative. This term produces an output that is proportional to the rate of change of the error signal. The primary benefit of D controllers is to resist change in the system, the most important of these being oscillations. The control output is calculated based on the rate of change of the error with time. The larger the rate of the change in error, the more pronounced the controller response will be.

$$c(t) = K_d \times de/dt$$

c(t) = controller output ; *Kd* = derivative time constant ; de = change in error ; dt = change in time

Feedback Control:

When using ros_control, the "/joint_states" topic is published by an instance of the JointStateController. This is a read-only controller that does not command any joint, but rather publishes the current joint states at a configurable frequency.
We subscribed to '/joint_states' topic and got the current velocity of the bot.

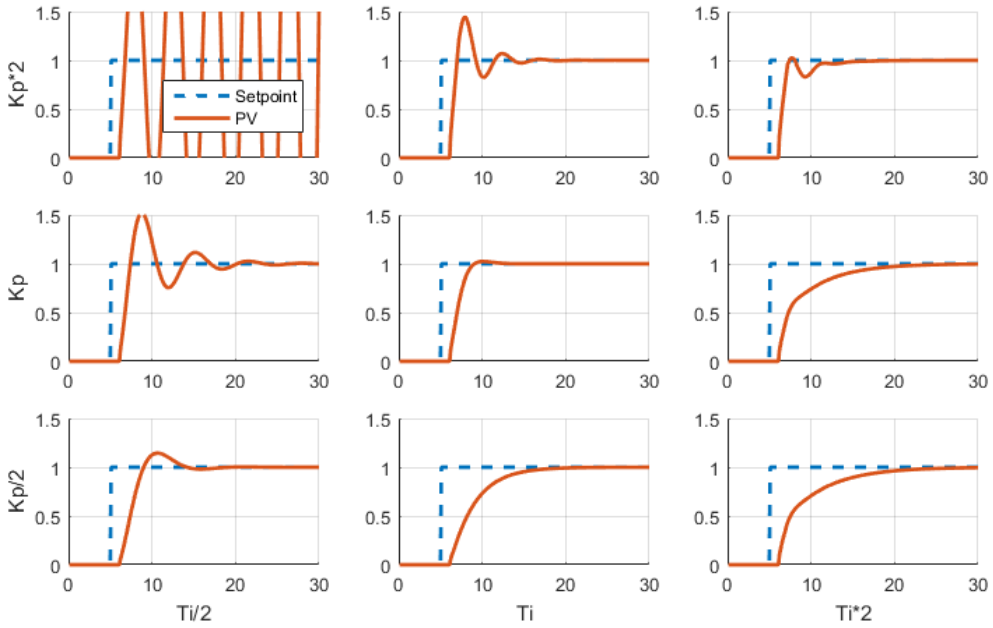
To find the error we can apply **set_velocity - current_velocity**, where the set_velocity is the velocity that we want to maintain.

After finding the error, we introduced the feedback control loop in which we used the error calculated above and put the values we got in the formula. This is to be done for all the wheels and apply PID control to all the wheels.
P= Kp*error
I=Ki*(sum of all errors)*d(t)
D=Kd*(error-prev_error)/d(t)
PID = P+I+D
By using the above mathematical forms, the values of PID are published to the topic
/fb_model/r_con_position_controller/command
/fb_model/l_con_position_controller/command
As we have seperated the left and right wheels' velocity, there are only two topics as we are taking the velocity of back and front wheels same.

This has to take place in a continuous manner until the set_velocity is changed and to change the set_velocity with ease, we decided to go with the "teleop_twist_keyboard" package.
For that we have to clone it using git clone https://github.com/ros-teleop/teleop_twist_keyboard.git

PID tuning:
To achieve higher accuracy, we have to tune the values of Kp, Ki, Kd. For that we would have to construct graphs for Velocity/time and check the error. This would be done with the help of rqt_plot and the topic name(this topic should have msg type of Int or Float only) to be given.
Initially put the values of Ki and Kd as zero and try to set Kp at a set_velocity so that the bot is controllable and also the graph will be obtained as shown in the 1st graph in the following image.

- Nextly Ki has to be set and Kd has to be set to zero at this phase, the plot will be as 2nd row in 1st column.



Lastly we have to tune Kd and it is totally related to the overshooting and undershooting. To lower the overshooting, we will increase Kd and tune it to obtain the graph at (2,2) in the picture shown above.

7. Mapviz(Map Visualization):

Mapviz is a tool in ROS used to visualise the model containing a map in the background. It contains some plugins which can be used depending upon the file we want to visualize.

steps:-

1. To visualize the model we need to run docker (it downloads the latest available satellite map from the web).
2. Add the required plugins like "/navsat/fix", tile_map, etc.
3. Run the bag files using command "rosbag play <name_of_bag_file>"

Installing Docker:

https://docs.docker.com/engine/install/ubuntu/ follow this link and the steps given.

To run docker, run this command in terminal :sudo docker run -p 8080:8080 -d -t -v ~/mapproxy:/mapproxy daniel snider/mapproxy

Plugins:

Before adding plugins make sure that you've added all the plugins in the order you want them to be operated, as the order in which the plugins are added plays a vital role in mapviz.

1. Tile_map:

Set source to "custom WMTS".

Put this link in Base URL:

http://localhost:8080/wmts/gm_layer/gm_grid/{level}/{x}/{y}.png

2. Navsat:

Put the topic name as "/navsat/fix".

And select the color as you wish.

Also add this line to your mapviz.launch file.

"<remap from="fix" to="/navsat/fix"/>"

NXP-AIM CAR DESIGN CHALLENGE

The objective of this challenge was to design and develop the CAR model as per guidelines, develop algorithms using AI/ML to handle obstacles on the track provided by NXP AIM INDIA.

To challenge involved the aid of the following simulation and control tools :

- ✓ Ubuntu Development Environment
- ✓ Gazebo Simulator
- ✓ ROS (Robotic Operating System)
- ✓ PX4
- ✓ rQT Image Viewer
- ✓ Sample Track

INSTALLING ROS2

We had run the script - **foxy_install_aim.sh** to begin the ROS2 installation process. This script can be accessed via the following link:
https://firebasestorage.googleapis.com/v0/b/gitbook-28427.appspot.com/o/assets%2F-MWeiLnwrIKZJWLFsXhf%2F-MXcfaeFblmznqth_0fk%2F-MXcgSRrBPzq5-K6O-C5%2Ffoxy_install_aim.sh?alt=media&token=2adf3a55-8463-4ff1-890e-67b6d32fe747

Now, we will need to run the foxy_install_aim.sh file to set up our software. To start, run the following commands in ubuntu terminal:

```
$ cd ~/Downloads
$ chmod a+x foxy_install_aim.sh
$ ./foxy_install_aim.sh
```

And then source your .bashrc by running the following command:

```
$ source ~/.bashrc
```

INSTALLING GAZEBO

Cloning and running sim_gazebo_bringup
First, we need to create a ROS2 workspace directory to store the installation files. Go to your home folder and create a new folder called "ros2ws", and a folder inside of that called "src":

```
$ cd ~
$ mkdir -p ros2ws/src && cd ros2ws/src
```

Once you are in the directory ~/ros2ws/src, it's time to clone the bringup repo. The bringup repo contains code that sets up the workspace for you automatically. To clone it, run the following command:

```
$ git clone git@github.com:rudislabs/sim_gazebo_bringup.git -b aim
```

When git prompts you to continue connecting with your RSA fingerprint, type yes:

Next, we are going to run sim_gazebo_bringup. To do this, run the following commands:

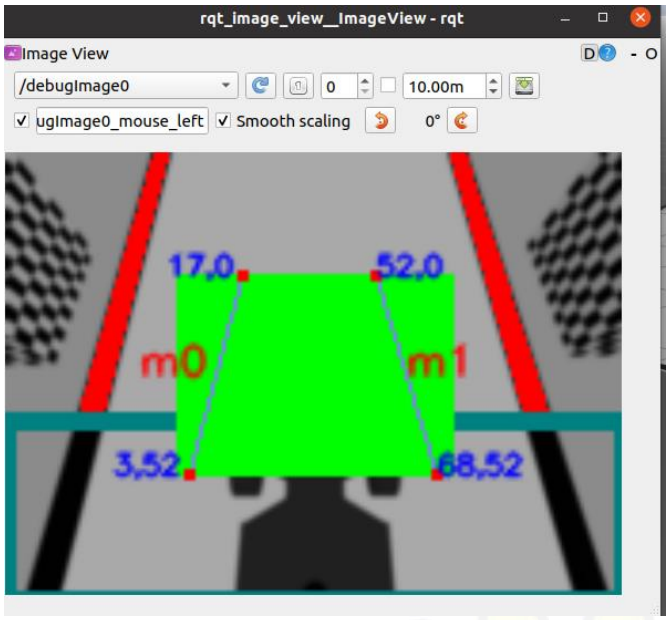
```
$ cd ~/ros2ws
$ colcon build --packages-select sim_gazebo_bringup --symlink-install
$ echo "source /home/$USER/ros2ws/install/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

Now that we have the bringup package set up, we can start installing all of the NXP Gazebo packages. Run the following command:

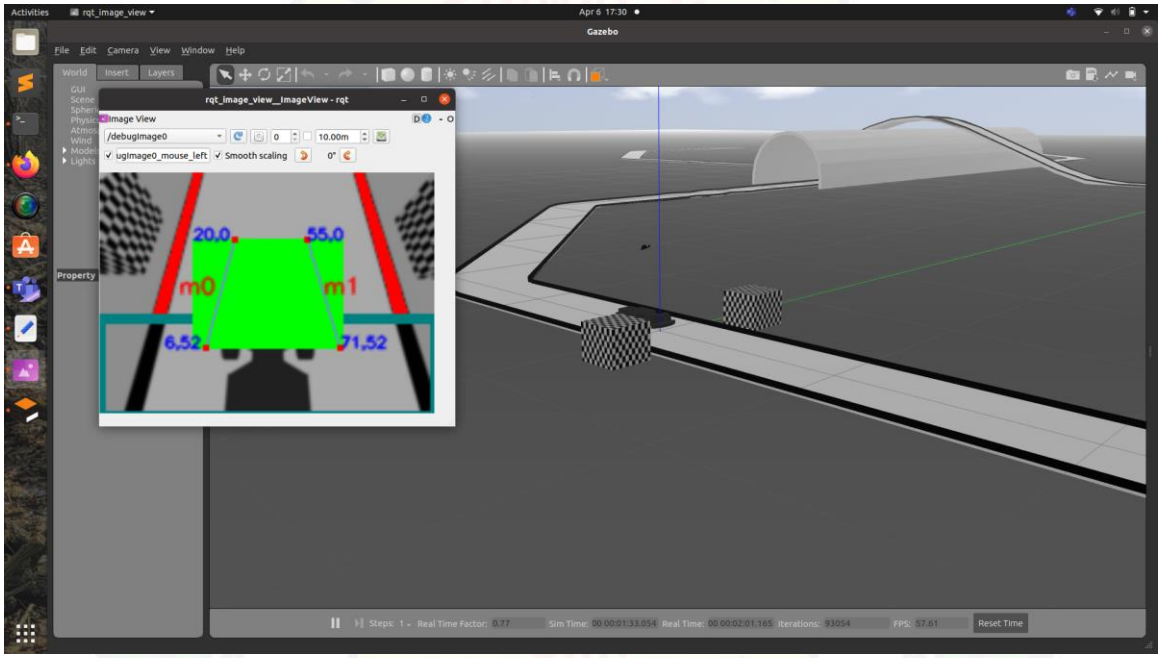
```
$ ros2 launch sim_gazebo_bringup sim_gazebo.launch.py
```

Visualizing Components:

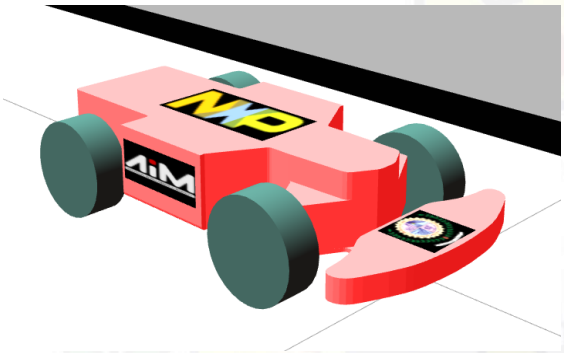
- Rqt Image Viewer



- Gazebo simulation of track

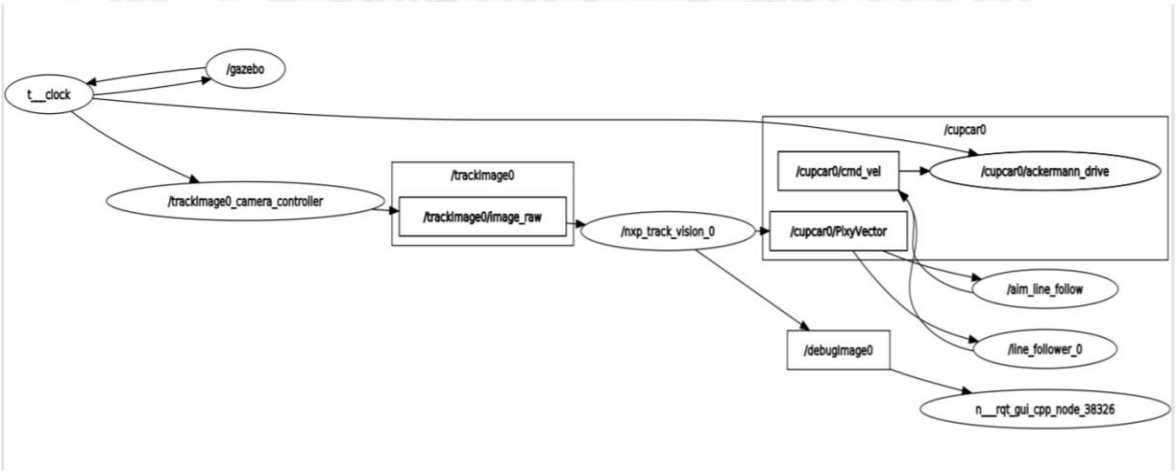


- Model



Methodology :

Using the given aim_line_follow algorithm, we completed the track at a very low linear_velocity that consumed three minutes fifty seconds. To optimize this we used PID controllers between vectors and the centre of the track, After tuning the PID we found out that we were getting high velocity while descending through the bridge that led to overshooting of the model. So we used the time library and decreased the linear velocities at the time of descent and made it high elsewhere. Given below is an architectural diagram that showcases the workflow of the topics that are being published and subscribed to give a better understanding of working of the model.



LOGICS BEHIND THE ALGORITHMS:

aim_line_follow.py : The above task was achieved by manipulating the aim_line_follow.py file in which the values of linear_velocity, angular_velocity and single_line_steer_scale which are scaled primarily and are used later. It is subscribing to a topic on message type defined in PixyVector, and being published by nxp_track_vision.py.

If both the vectors are visible: When the cupcar is moving in a straight path or taking mild turns, Both the vectors are visible. Here as per the code we are comparing the value of the centre of the frame width with the average of vectors and getting the value of steer velocity by that.

If only one of the vectors is visible: Whenever the cupcar is taking steep turns or moving a little out of the path, only one of the vectors would be visible. The remaining vector's slope is taken and the steer velocity can be taken out. Also the linear speed is decreased linearly as per time.

If no vectors are visible: When the cupcar moves out of the track completely, no vectors would be visible. The car gradually slows down, and after a decided time its speed will become zero and the car will stop.

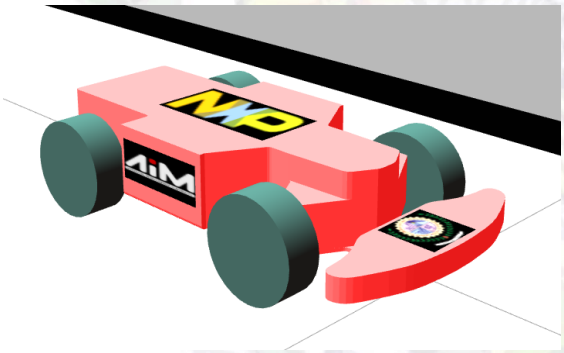
We used this code and tuned the linear_velocity, angular_velocity and single_line_steer_scale so that the cupcar was completing the complete track without intersecting the edges of the path. Though the time taken to complete the track was about 3 Mins and 50 seconds. In order to reduce the timings we used the several methods given below:

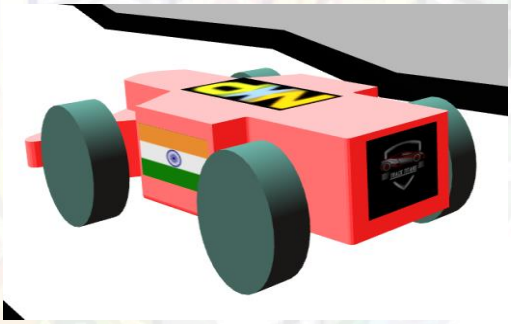
- **PID controller:** It is used for minimizing the error between the middle of track and average of the vectors when both the vectors are visible, by this we could increase our linear_velocity still there were problems with descending through the bridge and also the steep right turns.

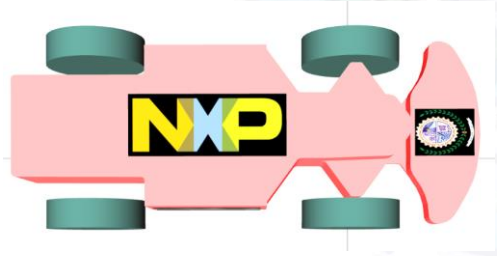
- **Working with datetime library:** To overcome the problem we decreased the linear_speed at the time while descending the bridge and taking right turns, also when only one of the vectors is visible the linear_speed is decreased again making steep turning possible. Completing all the steep turns its linear_speed is increased again so that the time taken to complete the track would be even lesser.Using all above methods we were able to complete the track in about 145 seconds.

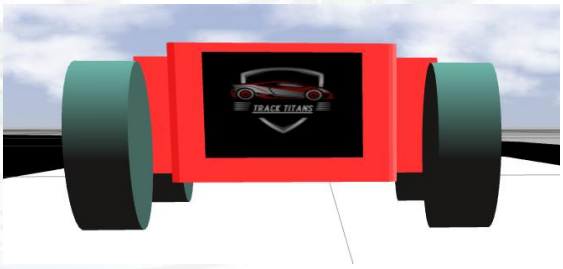
- **Addition of Logos :** The surfaces of the car model had several partitions or lines on the .stl files where the logos were supposed to be added. These obstructed the addition of textures on the chassis by direct changes in the script of CupCarCollisionBody.stl. So, we decided to place 2D planes in the 3D dimensional space which lie exactly at the parts of the chassis where we needed to add logos. To do so, we created separate visuals for every plane with corresponding size ratio as that of the logo file. Then we added the logos as scripts using the uri tag. Also we changed the materials file by adding all the five logos along with their names and sizes.

Following are the images of our car model:









14 References:

- Python:
  https://www.w3schools.com/python/default.asp

- OpenCV:
  https://drive.google.com/drive/folders/1aoMNZr66dvexoF9pBmBc5HGaiBNr87sl

- Linear Regression:
  https://medium.com/@shuklapratik22/linear-regression-from-scratch-a3d21eff4e7c

- Logistic Regression:
  https://medium.com/@martinpella/logistic-regression-from-scratch-in-python-124c5636b8ac

- Deep Neural Network Course:
  https://www.youtube.com/playlist?list=PLpFsSf5Dm-pd5d3rjNtIXUHT-v7bdaEIe

- Keras-tensorflow:
  https://keras.io/api/

- Tensorflow:
  https://www.tensorflow.org/guide

- Previously used Datasets:
  1. https://github.com/cwfid/dataset
  2. colab.research.google.com/notebooks/intro.ipynb#recent=true

- Image classification using CNN:
  https://www.geeksforgeeks.org/image-classifier-using-cnn/

- Dropouts and Batch Normalization in CNN:
  https://analyticsindiamag.com/everything-you-should-know-about-dropouts-and-batchnormalization-in-cnn/

- Video Classification using keras and Deep Learning:

https://www.pyimagesearch.com/2019/07/15/video-classification-with-keras-and-deep-learning/

- Pretrained CNN Models: https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/

- Final Dataset used for training : https://drive.google.com/drive/u/0/folders/1CQ101m-jyAFwTE4xfh1tKZyTistlQBoM

- Video used in testing: https://drive.google.com/drive/u/0/folders/1fqBOpZfmxrASNA_bTMHEa7zPWtAa1Fil
- Mapviz https://github.com/danielsnider/MapViz-Tile-Map-Google-Maps-Satellite