Projects

I am Irfan, currently (2022) working as an AI Engineer in Japan. I always seek to learn something new, enthusiastic about - Artificial Intelligence, Computer Vision, Machine Learning and Robotics etc. Some of my works in these fields are organized in this page. Python, C++, little bit of React (js) is mosly used in these works. The detailed algorithms code and results for each project is demonstrated below in respective sections. I have designed the flowcharts to precisely demonstrate how each of these algorithms work. Some people may find it redundant but it helps me to dump my knowledge this way. The page has many graphical contents, please be patient while loading. E-mail : irfanhasib.me@gmail.com

Table of Content

**Note : '(FS)' means implementation From Scratch.*

- *Remote rescue robot*
- *GPS, IMU data Fusion for precision velocity*
- *Desktop CNC Machine(2014)*
- *GUI software for Robotic ARM(2014)*
- *Visually Instructed Robotic ARM(2013)*

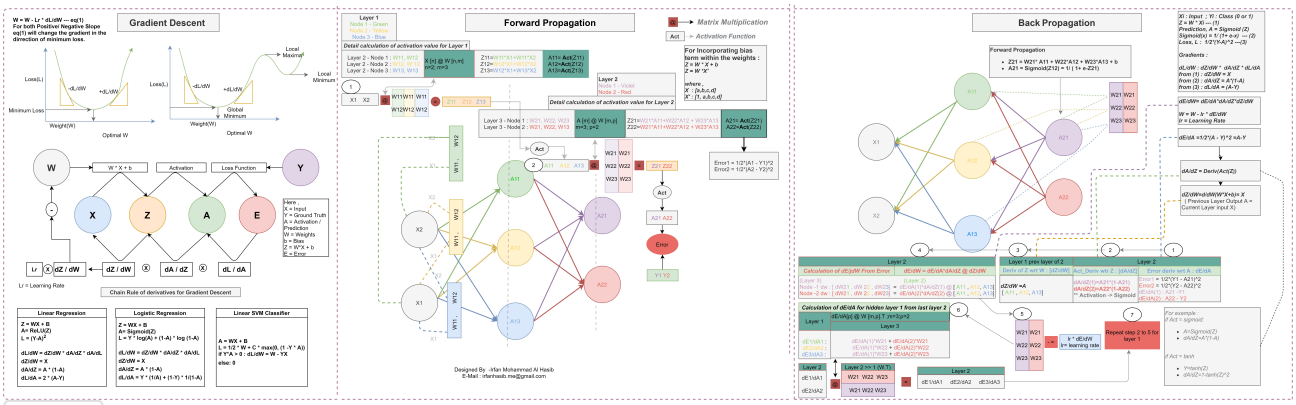
Neural Network Implementation from scratch

Go Back to Table of Content

1. Overview :

A simple multi layer neural network is implemented here from scratch, using raw Python programming language. I preffered OOP for encapsulation and pproper organization. Layer object are the backbone of this implementation while Network object used to organize it all together. A stack of Layer objects can be used to build a Network. Network object has the following fucntions(1) Layer :: "forward_propagation", (2) Layer :: "backward_propagation", (3) Layer :: "update_weights" . I have tested it with Banknote dataset. It was fitting good. Result can be found in Results section

2. Flow Chart (ANN) : (Open Image in new tab for full resolution)



☆ Star

3. Implementation Code :

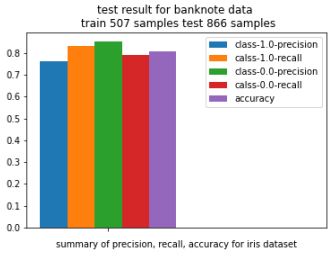
GitHub : NN Implementation from scratch - Notebook

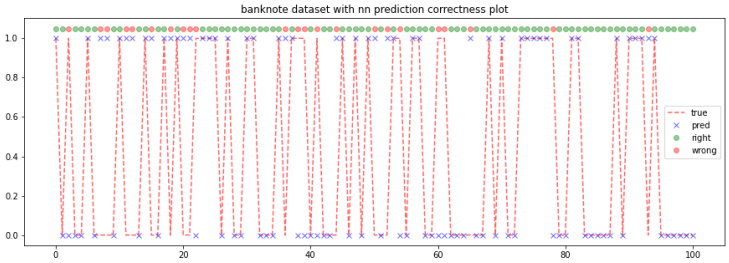
ANN Implementation From Scratch

In [1]:
```
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, RobustScaler
```

In [2]:
```
dataset='banknote'
df=pd.read_csv(f'Dataset/{dataset}.csv').sample(frac=1.0)
N=len(df)

df=df.fillna(0)
feats=df.columns.values[:-1]
target=df.columns.values[-1]
```

4. Result of ANN implementation for Bank Note data- a.Precision, Recall and Accuracy and b.True Label vs Prediction





Descision Tree Implementation from scratch

Go Back to Table of Content

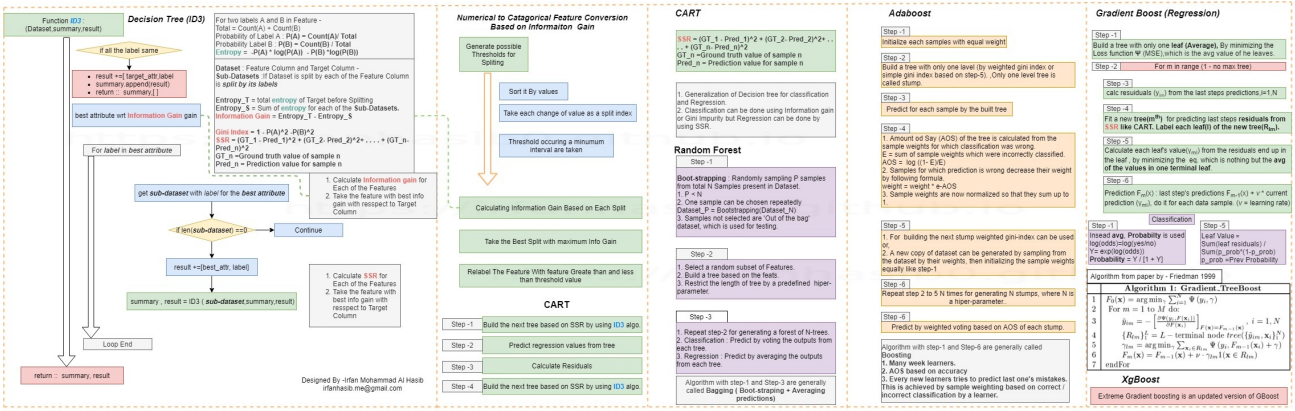
- ***Section 1: Summary***
- ***Section 2: Flowchart***
- ***section 3: Code***
- ***Section 4:Result (Results can be found after code demonstration for each algorithm)***

1. Overview

A fundamental descision tree based classifier is implemented here using ID3 algorithm with Python from scratch. Titanic and irish dataset was used for testing the algorithm. Function for (1) Continuous data spliting based on information gain, (2) Information Gain Calculation, (3) ID3 Algorithm (4) Prediction from the built tree, are main part of the implementation. For preventing overfitting reduced error pruining was applied. Accuracy,Precision,Recall is reported in Result section.

2. ID3 Flow Chart of Implementation (Open Image in new tab for full resolution)

☆ Star



3. ID3 Implementation Code

GitHub : ID3 Implementation from scratch - Notebook

ID3 Implemented from scratch with python

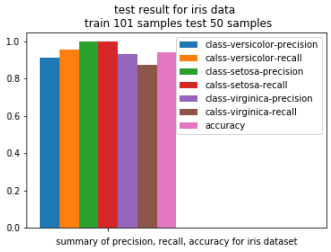
** some basic pandas and nupmpy utility functions are used

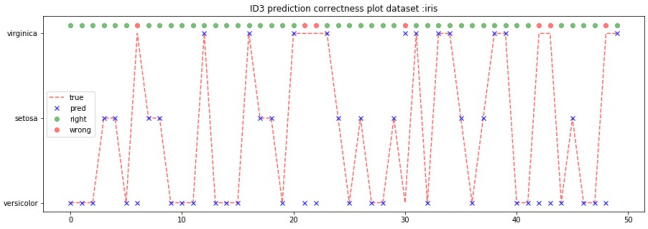
1. Continuout to Discrete Convertion

Utility functions

Here I tried to avoid pandas utility functions and implemented many of them from scracth.

4. Result of ID3 implementation for iris data - a.Precision, Recall and Accuracy and b.True Label vs Prediction -





SVM, Logistic Regression Implementation from scratch

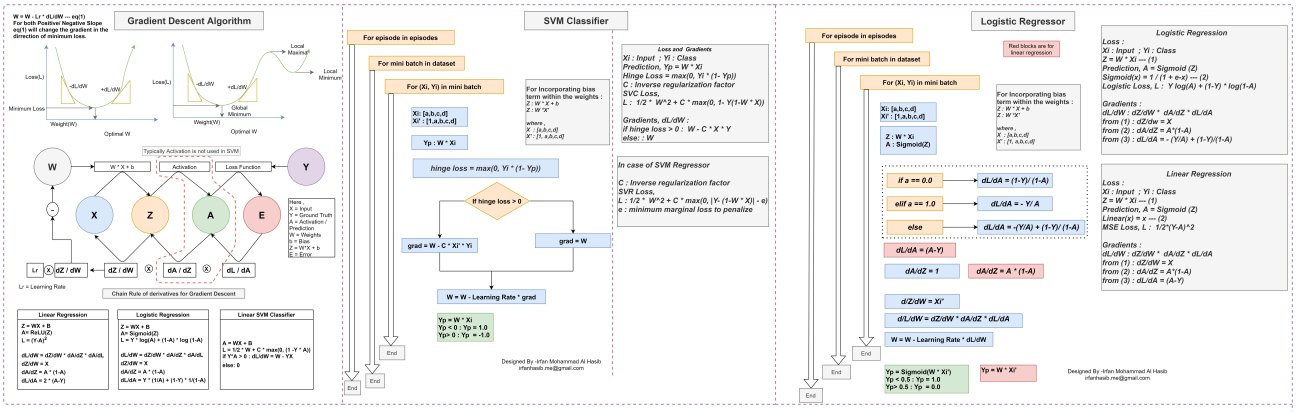
Go Back to Table of Content

- ***Section 1: Summary***
- ***Section 2: Flowchart (SVM & Logistic Regression)***
- ***section 3: Code (3.1 SVM, 3.2 Logistic Regression)***
- ***Section 4: Result(4.1 SVM ,4.2 Logistic Regression)***

1. Overview

SVM and Logisitic Regression code from scratch is presented below. SvmC object with the functions (a) SvnC.predict (b) SvmC.calc_gradients (c) SvmC.update_weights is implemented. LogisticRegression object also have the same functions. Banknote dataset is used for testing both of the algorithms. SVM and LogisticRegression Class is implemented accordingly for the respective algorithms. For each of the 2 class predict, calc loss, calc grads, update weights functions are implemented. Both of the algorithms showed good accuracy for banknote dataset after minimal parameter tuning. Results can be found in respective results section.

2.0 SVM(Left), Logistic Regression(Right) Flow Chart of Implementation (Open Image in new tab for full resolution)



3.1 Implementation Code SVM

(a) GitHub : SVM Implementation from scratch - Notebook

SVM Implementation From Scratch

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, RobustSc
aler
dataset='banknote'
df=pd.read_csv('Dataset/banknote.csv').sample(frac=1.0).
reset_index(drop=True)
N=len(df)

df=df.fillna(0)
feats=df.columns.values[:-1]
```

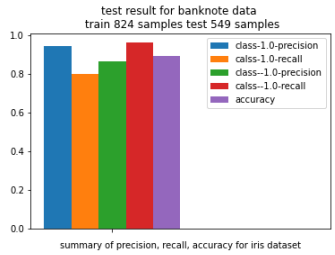
3.1 Implementation Code Logistic Regression

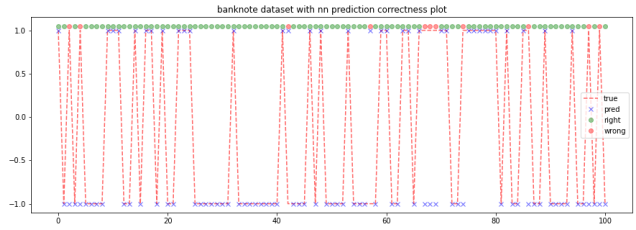
(b) GitHub : Logistic Regression Implementation from scratch - Notebook

Logistic Regression Implementation From Scratch

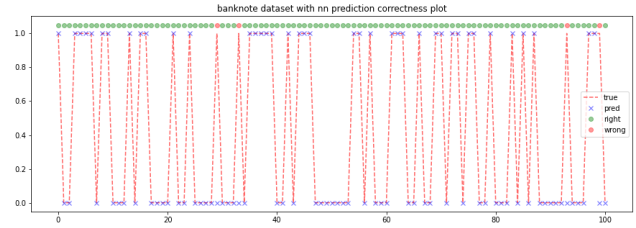
In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, RobustSc
aler
dataset='banknote'
df=pd.read_csv('Dataset/banknote.csv').sample(frac=1.0).
reset_index(drop=True)
N=len(df)

df=df.fillna(0)
feats=df.columns.values[:-1]
```

4.1 Result of SVM implementation - a.Precision, Recall and Accuracy and b.True Label vs Prediction





4.2 Result of Logitic Regression implementation - a.Precision, Recall and Accuracy and b.True Label vs Prediction



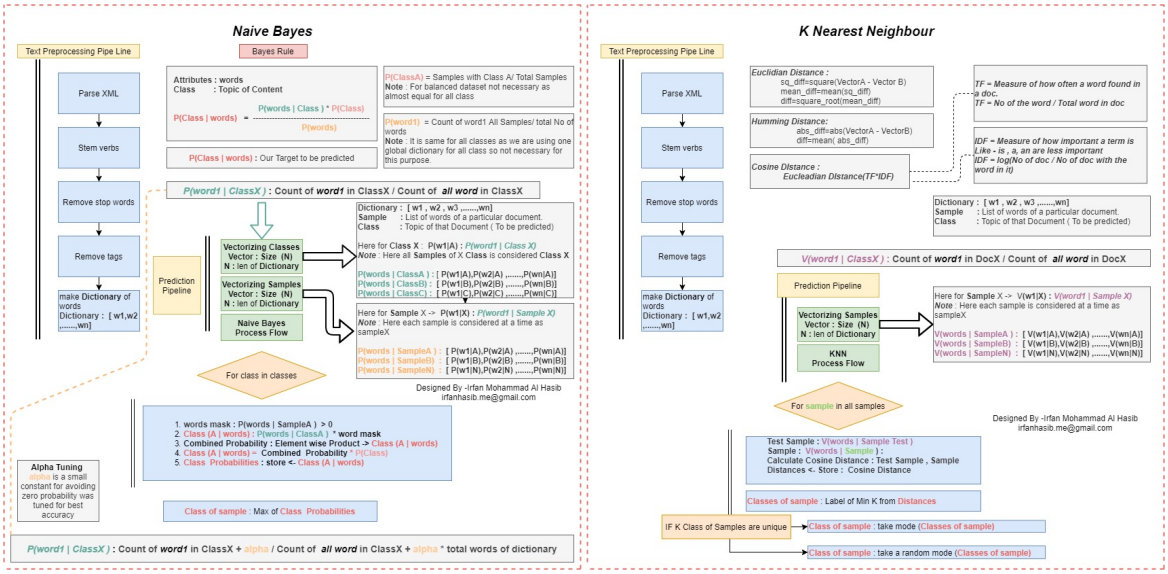
Naive Bayes and KNN Implementation for text classification

Go Back to Table of Content

1. Overview

The steps for Naive Bias algorithm are as follows - (1) Calculating Attribute(word) Probabilities given each class. (2) Calculating Attribute(word) Probabilities given each Samples. (3) Applying Bayes Theorem for getting class probalities. (4) Max class probality is the predicted label. Algorithm Steps for KNN are - (1) Calculating Attribute Probability vector for each sample (2) Calculating Cosine/Eucleidian/Humming Distances from test sample to each of train sample (3) Taking labels of K min distance samples from train data. Taking mode of K labels as prediction (tie breaks by random choise). The model was Tuned for best alpha values. Archived data from stack exchange is used for testing the implementations. Accuracy,Precision and Recall are reported.

2. (a)Naive Bayes (b)K Nearest Neighbour algorithm Flow Chart



3.1 Implementation Code Naive Bias

(a) GitHub : Naive Bayes Implementation from scratch - Notebook

Naive Bias Implementation From Scratch

In [1]:
```
from xml.dom import minidom
import re
import pandas as pd
import sys
import pickle
import numpy as np
#from nltk.stem import PorterStemmer
#from nltk.tokenize import sent_tokenize, word_tokenize
#ps = PorterStemmer()
```

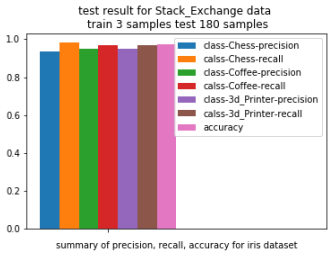
3.2 Implementation Code K Nearest Neighbour

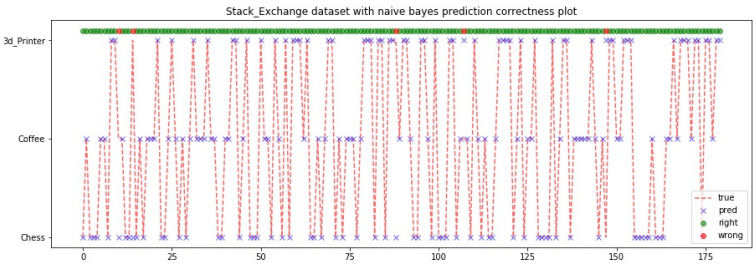
(b) GitHub : KNN Implementation from scratch - Notebook

KNN Implementation from Scratch

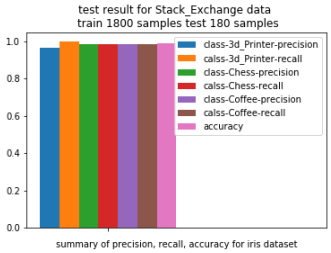
In [1]:
```
from xml.dom import minidom
import re
import pandas as pd
import sys
import pickle
import numpy as np
import time
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
ps = PorterStemmer()
sys.__stdout__=sys.stdout
```

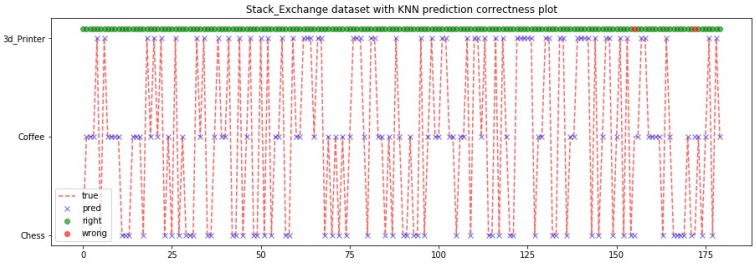
4.1 Result of Naive Bayes implementation - a.Precision Recall and Accuracy b.True Label vs Prediction





4.2 Result of KNN implementation - a.Precision Recall and Accuracy b.True Label vs Prediction





Go Back to Table of Content

Yolo-V2.0 with KERAS and Tensorflow for Object detection

Go Back to Table of Content

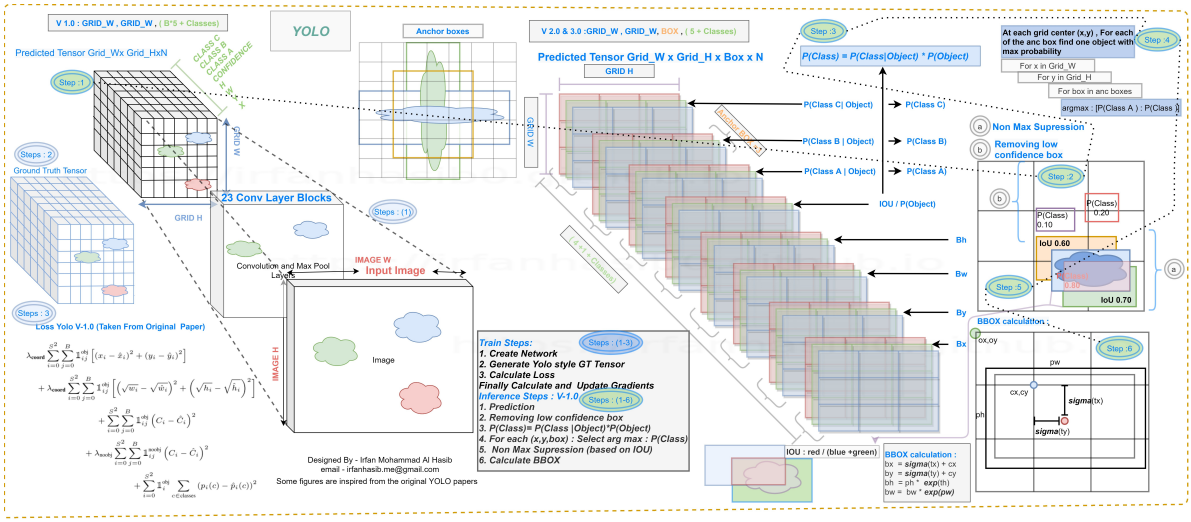
- ***Section 1: Overview***
- ***Section 2. Flowchart***
- ***Section 3. Code***

- ***Section 4. Result***

1. Overview

Yolo V-2.0 is implemented here and tested with COCO dataset. It is one of the SOTA algorithm for object detection. The algorithm is showed in the figure in algorithm section. Python code can be found in code section.

2. Flowchart



2. Code YOLO-V2.0 for COCO dataset

GitHub Link : Yolo-V2 ☆ Star

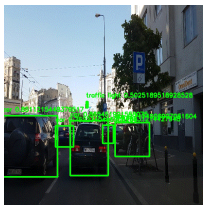
YOLO -V2.0 Implementation

1. Implementated from scratch using tensorflow
2. Tested for COCO dataset

In [1]:
```
#!pip install tensorflow==1.12.0
```

In [2]:
```
# Initialization
colab_run=False
anc_box= True
_grid_offset=True
train=False
```

3. Results:



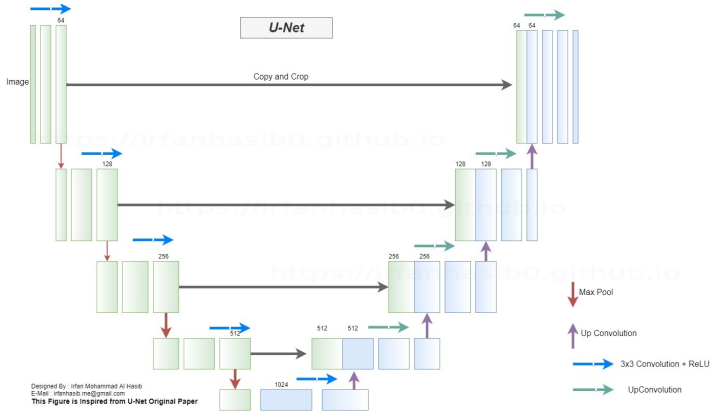
U-Net with KERAS for City Space Dataset

Go Back to Table of Content

1. Overview

U-Net is implemented here and tested with CityScapes dataset. It is one of the SOTA algorithm for semantic segmentation. The network architecture is showed in the figure of Network Architecture section. Python code can be found in code section.

2. Network Architecture



3. Code of U-Net for CityScapes dataset

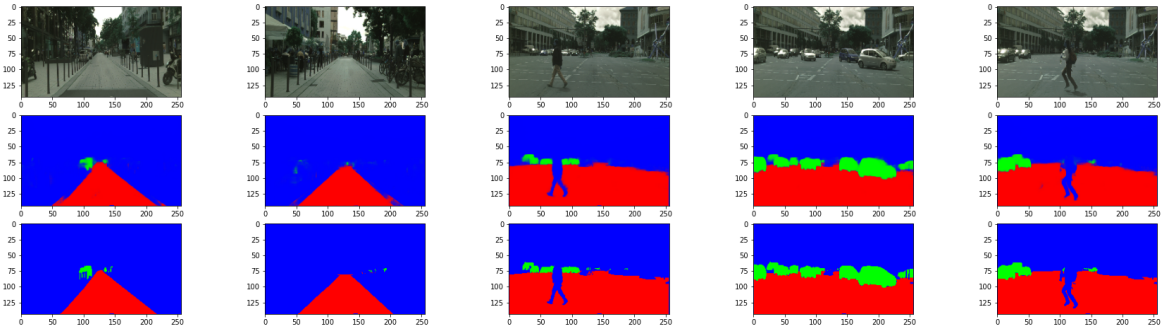
GitHub :Unet for segmenting City Space Dataset

U-Net Implementation

1. Implemented with Tensorflow and Keras
2. Trained and tested on CityScapes Dataset

In [2]:
```
seed=1
import os
os.environ['PYTHONHASHSEED']=str(0)
import random
random.seed(seed)
import numpy as np
np.random.seed(seed)
```

4. Result



Flow-Net with KERAS and Tensorflow

Go Back to Table of Content

1. Overview

FlowNet is one of the most popular architecture for optical flow prediction. Here the implementation code and results on KITTI Flow Dataset is presented below

2. Code

GitHub Link :FlowNet for Optical Flow Kitti Dataset

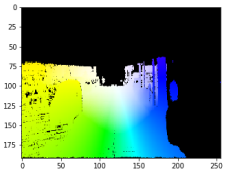
Flow-NetS Implementation

1. Using Tensorflow
2. tested on KITTI Flow Dataset

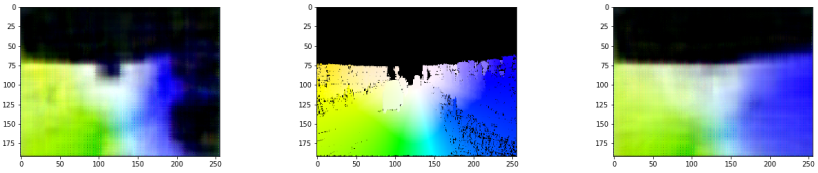
In [1]:
```
train= False
_gen_flow=True
_compress=False
epochs=2000
```

In [2]:
```
exp_name='exp-F3\\'
data_root='D:\\KITTI\\'
```

3. Result



Train GT, Train prediction, Test GT, Test Prediction



Monocular Disparity using KERAS and Tensorflow

Go Back to Table of Content

- ***Section 1: Overview***
- ***Section 3. Code***
- ***Section 4. Result***

1. Overview

Slightly modified Architecture of FlowNet is used for monocular depth estimation. It still performed satisfactorily. The implementation code and results on KITTI Flow Dataset(on disp ground truth) is presented below

Code

GitHub Link : Single View Depth Estimation

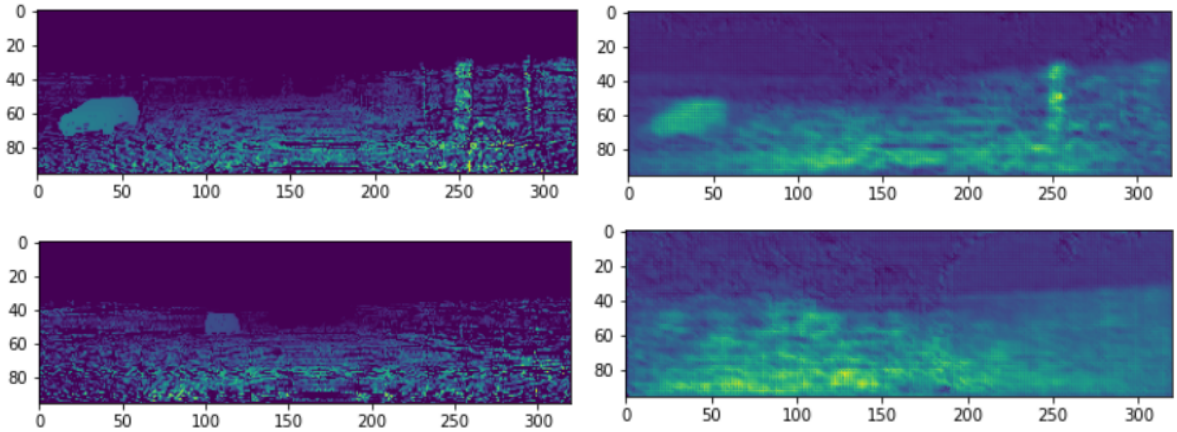
Monocular Depth Estimation Implementation

```
1. FlowNetS architecture was used for depth estimation.
2. Implemented using Tensorflow
3. Tested on KITTI Flow Dataset (KITTI FLow dataset have
disparity ground truth also).
```

```
In [1]: train= False
        _gen_disp=True
        epochs=2500
```

Result

Top row : Train ; Bottom Row : Validation ; Left : Ground Truth ; Right : Prediction



DQN , DDPG, PPO A2C Implementation from scratch

Go Back to Table of Content

1. GitHub Code Links :

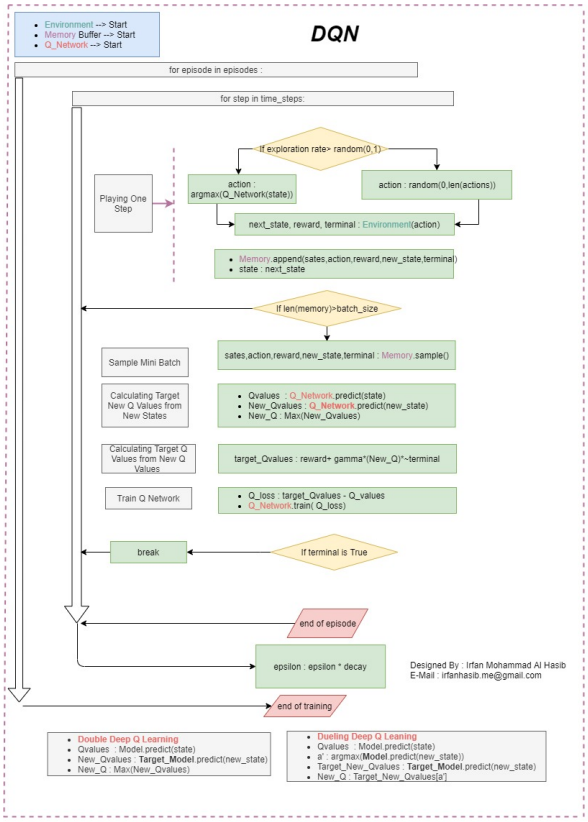
- *Multi Arm Bandit - Notebook*
- *Value Iteration - Notebook*
- *DQN for Mountain Car - Notebook*
- *DDPG for Pendulum - Notebook*
- *PPO for Bipedal Walker - Notebook*
- *PPO for Lunar Lander - Notebook*

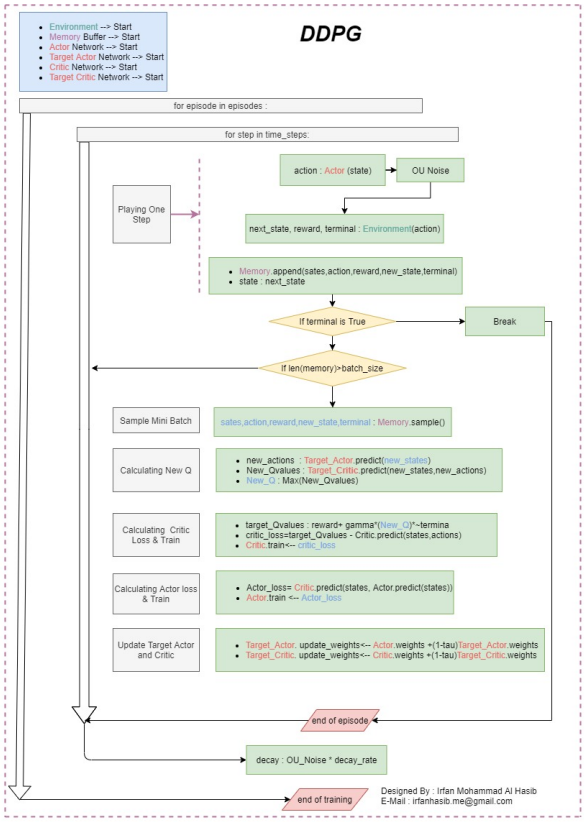
2. Overview

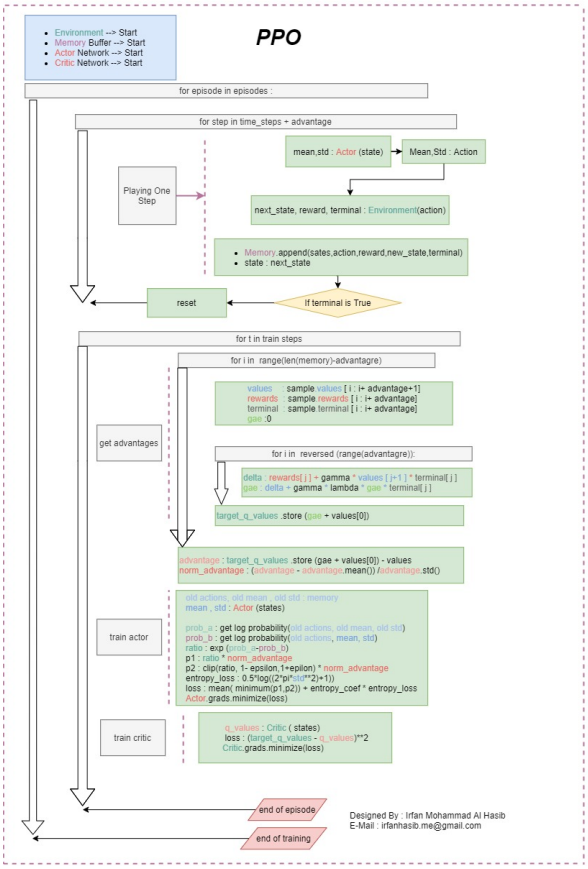
DQN implemented and tested on OpenAI gym Mountain Car ENvironment, DDPG implemented and tested on OpenAI gym Pendulumn Environment, PPO implemented and tested on OpenAI gym Bipedal Walker and Lunar Lander Environment. All these algorithms are implemnted from scratch using TensorFlow, Numpy and Python.

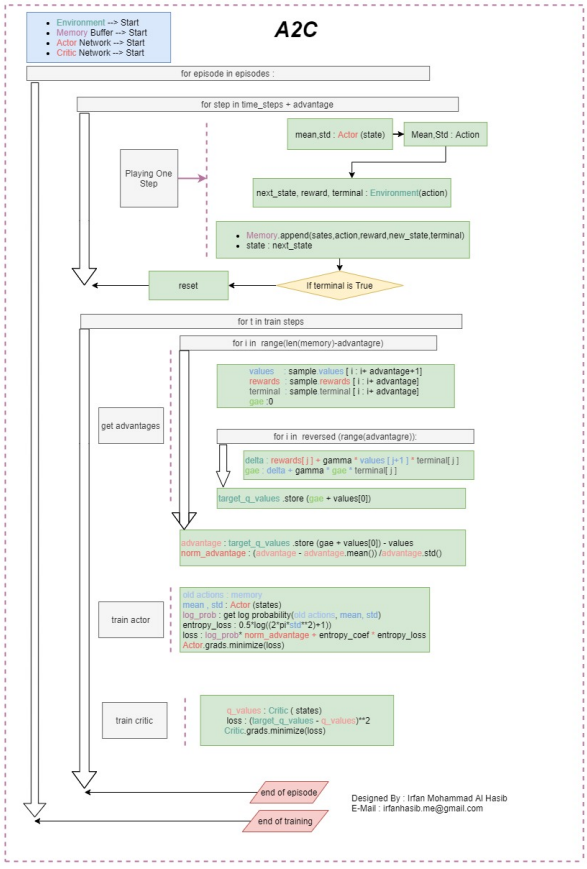
3. Detailed Flow Chart for DQN and DDPG : (Please Zoom or Open in New tab for proper resolution)

4. Code PPO for bipedal environment


DQN
Environment --> Start
Memory Buffer --> Start
Q_Network --> Start
for episode in episodes :
for step in time_steps:
If exploration rate> random(0,1)
Playing One Step
action : argmax(Q_Network(state))
action : random(0,len(actions))
next_state, reward, terminal : Environment(action)
Memory.append(sates,action,reward,new_state,terminal)
state : next_state
If len(memory)>batch_size
Sample Mini Batch
sates,action,reward,new_state,terminal : Memory.sample()
Calculating Target New Q Values from New States
Qvalues : Q_Network.predict(state)
New_Qvalues : Q_Network.predict(new_state)
New_Q : Max(New_Qvalues)
Calculating Target Q Values from New Q Values
target_Qvalues : reward+ gamma*(New_Q)*-terminal
Train Q Network
Q_loss : target_Qvalues - Q_values
Q_Network.train(Q_loss)
break
If terminal is True
end of episode
epsilon : epsilon * decay
end of training
Designed By : Irfan Mohammad Al Hasib
E-Mail : irfanhasib.me@gmail.com
Double Deep Q Learning
Qvalues : Model.predict(state)
New_Qvalues : Target_Model.predict(new_state)
New_Q : Max(New_Qvalues)
Dueling Deep Q Leaning
Qvalues : Model.predict(state)
a' : argmax(Model.predict(new_state))
Target_New_Qvalues : Target_Model.predict(new_state)
New_Q : Target_New_Qvalues[a']


DDPG
Environment --> Start
Memory Buffer --> Start
Actor Network --> Start
Target Actor Network --> Start
Critic Network --> Start
Target Critic Network --> Start
for episode in episodes :
for step in time_steps:
Playing One Step
action : Actor (state)
OU Noise
next_state, reward, terminal : Environment(action)
Memory.append(sates,action,reward,new_state,terminal)
state : next_state
If terminal is True
Break
If len(memory)>batch_size
Sample Mini Batch
sates,action,reward,new_state,terminal : Memory.sample()
Calculating New Q
new_actions : Target_Actor.predict(new_states)
New_Qvalues : Target_Critic.predict(new_states,new_actions)
New_Q : Max(New_Qvalues)
Calculating Critic Loss & Train
target_Qvalues : reward+ gamma*(New_Q)*-termina
critic_loss=target_Qvalues - Critic.predict(states,actions)
Critic.train<-- critic_loss
Calculating Actor loss & Train
Actor_loss= Critic.predict(states, Actor.predict(states))
Actor.train <-- Actor_loss
Update Target Actor and Critic
Target_Actor. update_weights<-- Actor.weights +(1-tau)Target_Actor.weights
Target_Critic. update_weights<-- Critic.weights +(1-tau)Target_Critic.weights
end of episode
decay : OU_Noise * decay_rate
end of training
Designed By : Irfan Mohammad Al Hasib
E-Mail : irfanhasib.me@gmail.com


PPO
Environment --> Start
Memory Buffer --> Start
Actor Network --> Start
Critic Network --> Start
for episode in episodes :
for step in time_steps + advantage
Playing One Step
mean,std : Actor (state)
Mean,Std : Action
next_state, reward, terminal : Environment(action)
Memory.append(sates,action,reward,new_state,terminal)
state : next_state
reset
If terminal is True
for t in train steps
for i in range(len(memory)-advantagre)
values : sample.values [i : i+ advantage+1]
rewards : sample.rewards [i : i+ advantage]
terminal : sample.terminal [i : i+ advantage]
gae :0
get advantages
for i in reversed (range(advantagre)):
delta : rewards[j] + gamma * values [j+1] * terminal[j]
gae : delta + gamma * lambda * gae * terminal[j]
target_q_values .store (gae + values[0])
advantage : target_q_values .store (gae + values[0]) - values
norm_advantage : (advantage - advantage.mean()) /advantage.std()
train actor
old actions , old mean , old std : memory
mean , std : Actor (states)
prob_a : get log probability(old actions, old mean, old std)
prob_b : get log probability(old actions, mean, std)
ratio : exp (prob_a-prob_b)
p1 : ratio * norm_advantage
p2 : clip(ratio, 1- epsilon,1+epsilon) * norm_advantage
entropy_loss : 0.5*log((2*pi*std**2)+1))
loss : mean(minimum(p1,p2)) + entropy_coef * entropy_loss
Actor.grads.minimize(loss)
train critic
q_values : Critic (states)
loss : (target_q_values - q_values)**2
Critic.grads.minimize(loss)
end of episode
end of training
Designed By : Irfan Mohammad Al Hasib
E-Mail : irfanhasib.me@gmail.com


A2C
Environment --> Start
Memory Buffer --> Start
Actor Network --> Start
Critic Network --> Start
for episode in episodes :
for step in time_steps + advantage
Playing One Step
mean,std : Actor (state)
Mean,Std : Action
next_state, reward, terminal : Environment(action)
Memory.append(sates,action,reward,new_state,terminal)
state : next_state
reset
If terminal is True
for t in train steps
for i in range(len(memory)-advantagre)
values : sample.values [i : i+ advantage+1]
rewards : sample.rewards [i : i+ advantage]
terminal : sample.terminal [i : i+ advantage]
gae :0
get advantages
for i in reversed (range(advantagre)):
delta : rewards[j] + gamma * values [j+1] * terminal[j]
gae : delta + gamma * gae * terminal[j]
target_q_values .store (gae + values[0])
advantage : target_q_values .store (gae + values[0]) - values
norm_advantage : (advantage - advantage.mean()) /advantage.std()
train actor
old actions : memory
mean , std : Actor (states)
log_prob : get log probability(old actions, mean, std)
entropy_loss : 0.5*log((2*pi*std**2)+1))
loss : log_prob* norm_advantage + entropy_coef * entropy_loss
Actor.grads.minimize(loss)
train critic
q_values : Critic (states)
loss : (target_q_values - q_values)**2
Critic.grads.minimize(loss)
end of episode
end of training
Designed By : Irfan Mohammad Al Hasib
E-Mail : irfanhasib.me@gmail.com

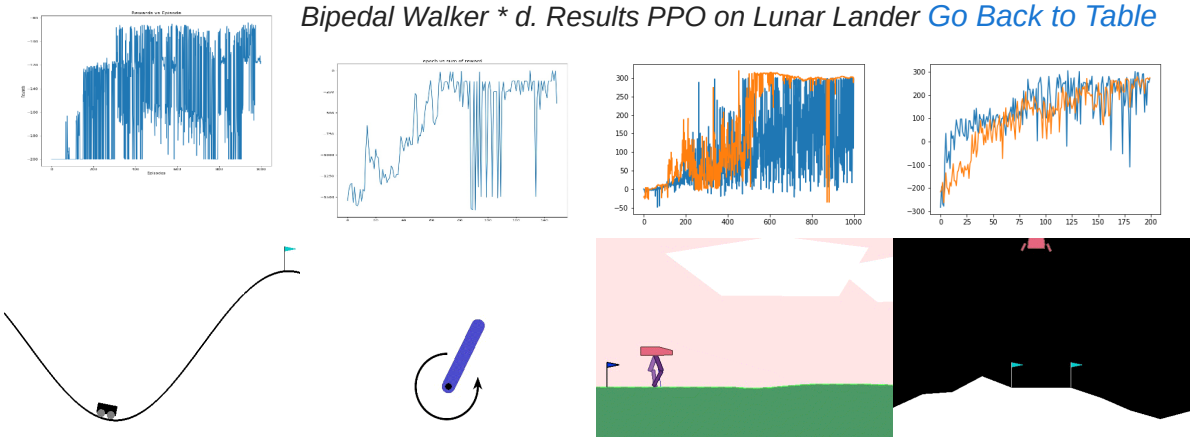
Proxilmal Policy Optimization

1. Impletation from scratch with tensorflow
2. Tested for Bipedal walker environment of OpenAI

In [1]:
```
seed=1
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
os.environ['PYTHONHASHSEED']=str(seed)
import numpy as np
np.random.seed(seed)
import tensorflow as tf
tf.random.set_seed(seed)
```

5. Results

** a. Results DQN on Mountain Car * b. Results DDPG on Pendulum * c. Results PPO on Bipedal Walker * d. Results PPO on Lunar Lander* *Go Back to Table of Content*



House Price Prediction :: Data Pre-Processing and Hiper-Parameter Tuning

Go Back to Table of Content

- ***Section 1: Overview***
- ***Section 2: Demonstration of Project Code***
- ***Section 3: Flow Chart***

1. Overview :

- ***House Price Dataset from kaggle.com***
- ***Data Preprocessing***

- ***ANN Class with tensorflow low level API***
- ***Hiperparameter Tuning***
- ***All the graphs of Data preprocessing and Hiperparameter Tuning can be found in Notebook***

▶ ***Click to expand***

Notebook : House Price Prediction - Notebook (Project Presentation and Code Link)

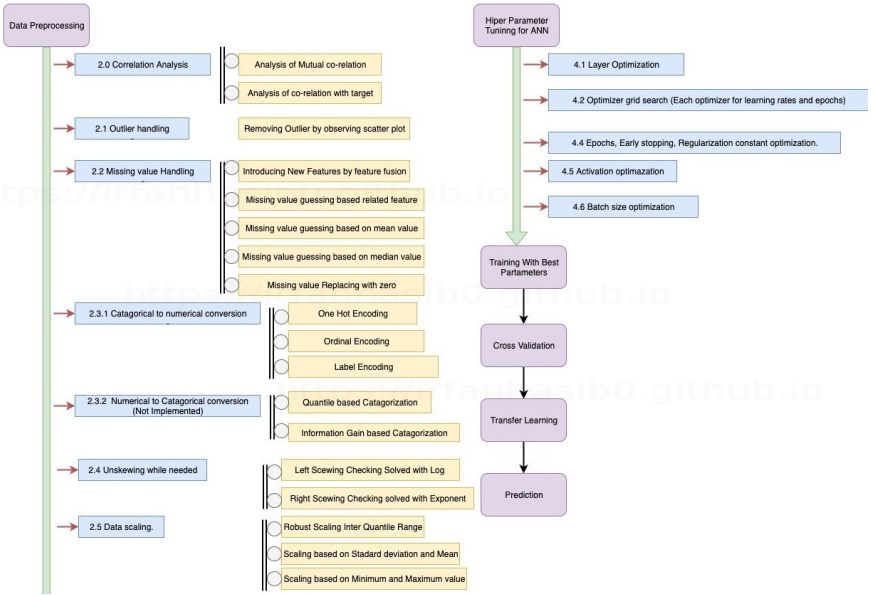
☆ *Star*

House Prices: Advanced Regression Techniques

Summary :

```
House price is a kaggle competition which is based on pr
edicting house price from
different features of house using machine learning techn
iques. The following processes
are done for solving the house price prediction problem.
```

2. Project Flow Chart :



Japanese Job Entrance Problem :: Shakura Bloom Prediction

Go Back to Table of Content

- ***Section 1: Overview***
- ***Section 2: Demonstration of Project Code***
- ***Section 3: Flow Chart***

1. Overview

- ***Weather data from japanese meteorological agencym***
- ***Feature Extraction and Data Preprocessing***
- ***ANN Class with tensorflow low level API***
- ***Hiperparameter Tuning***
- ***All the graphs of Data preprocessing and Hiperparameter Tuning can be found in Notebook***

▶ ***Click to expand***

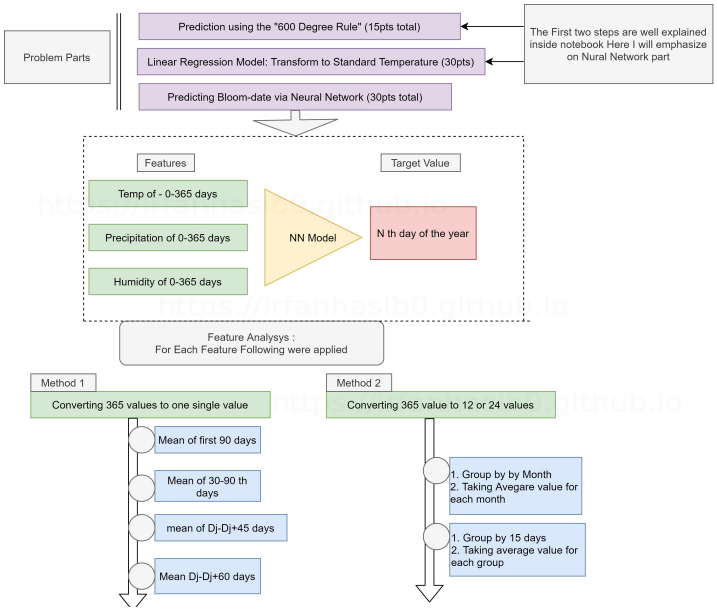
Notebook : Shakura Bloom Prediction - Notebook (Project Presentation and Code Link)

Entrance Challenge: When Will the Sakura Bloom?

Basics of the Sakura Bloom-cycle (5pts total)

In a year, sakura trees basically go through 4 phases: energy production, hibernation, growth, and of course flowering. These phases roughly follow the seasons, but not

2. Project Flow Chart :



ILQR and MPC :: Implementation from scratch for self driving car simulator

Go Back to Table of Content

GitHub links

- *GitHub : ILQR Implementation - Notebook*
- *GitHub : MPC Implementation - Notebook*

1. Overview :

- ***Simulation Platform :***
  - *AIRSIM by Microsoft Inc.*
  - *OpenAI GYM - Car Environment was tested*
- ***IO :***

  Input --> Map Points , Output --> Steering Angle, Acclelation, Brake
- ***MAP to Trajectory : Steps :***
  - *Environment Module*
  - *Map Tracker Module*
  - *Data Preprcessor Module*
- ***Optimaization Algorithm : Trajectory to Optimal Steering,Accleration,Brake :***
  - *iLQR using Raw Python and Numpy*
  - *MPC using Python , Numpy and CVXPY*

▶ ***Click to expand***

MPC modeling

$$State\ Space : z = [x, y, v, \phi] \quad where, x : x - position, y : y - position, v : velocity$$
$$Action\ Space : u = [a, \delta] \quad where, a : accellation, \delta :$$

Cost and Constraints :

Cost :

$$min\ Q_f(z_{T,ref} - z_T)^2 + Q\Sigma(z_{t,ref} - z_t)^2 + R\Sigma u_t{}^2 + R_d\Sigma(u_{t+1} - u_t)^2$$

z_{ref} : *target states*

Constraints :

$$z_{t+1} = Az_t + Bu + C$$
$$Maximum\ steering\ speed = abs[u_{t+1} - u_t] < du_{max}$$
$$Maximum\ steering\ angle = u_t < u_{max}$$
$$Initial\ state = z_0 = z_{0,ob}$$
$$Maximum\ and\ minimum\ speed = v_{min} < v_t < v_{max}$$
$$Maximum\ andminimum\ input = u_{min} < u_t < u_{max}$$

State Space model for Car system

$$z_{t+1} = Az_t + Bu + C\ where,\ A = \begin{bmatrix} 1 & 0 & cos(\bar{\phi})dt & -\bar{v}sin(\bar{\phi})dt \\ 0 & 1 & sin(\bar{\phi})dt & \bar{v}cos(\bar{\phi})dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{tan(\delta)}{L}dt & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ dt & 0 \\ 0 & \frac{\bar{v}}{Lcos^2(\delta)}dt \end{bmatrix} \quad C = \begin{bmatrix} \bar{v}sin(\bar{\phi})\bar{\phi}dt \\ -\bar{v}cos(\bar{\phi})\bar{\phi}dt \\ 0 \\ -\frac{v\delta}{Lcos^2(\delta)}dt \end{bmatrix}$$

▶

Expand to see derivation

Iterative Linear Quadratic Regulator :

1. U as a function of Previous U , X and Previous X :

$$u'(i) = u(i) + k(i) + K(i)(x'(i) - x(i))$$

▶

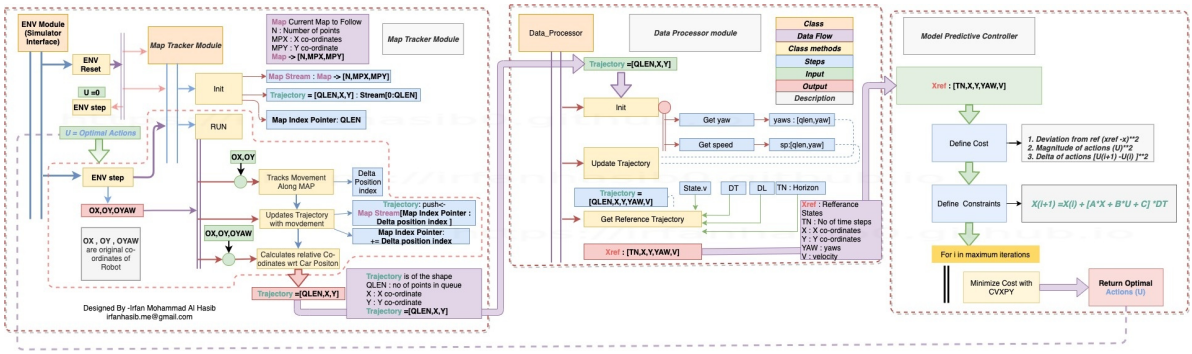
2. Calcualting K and k : (Expand to see)

4. For detail derivation please see the paper :

Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization by - Yuval Tassa

citations(369) according to February 27 2020, Published on - 2012

2. Project Flow Chart :

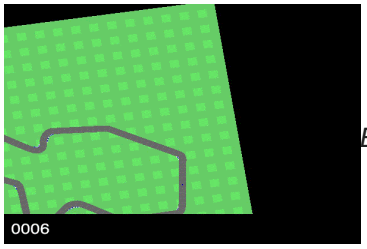


▶ ***Click to expand***

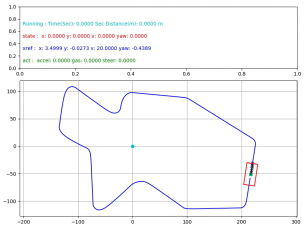
☆ ***Star***

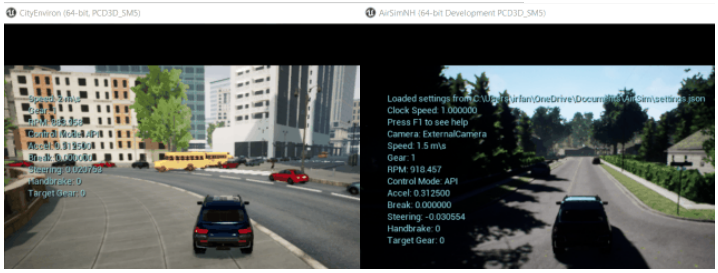
3. Results (ILQR) :

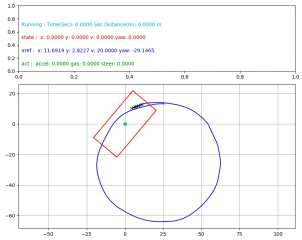
** OpenAI Gym Car Environment * Airsim City Space Environment * Airsim Neighbourhood*

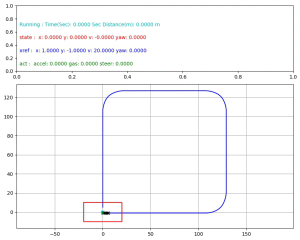


Environment









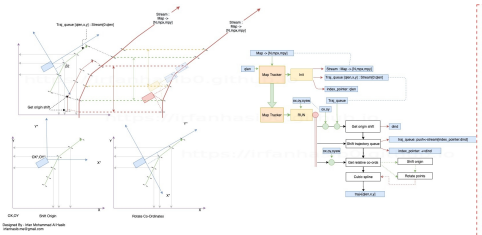
Inspired from -

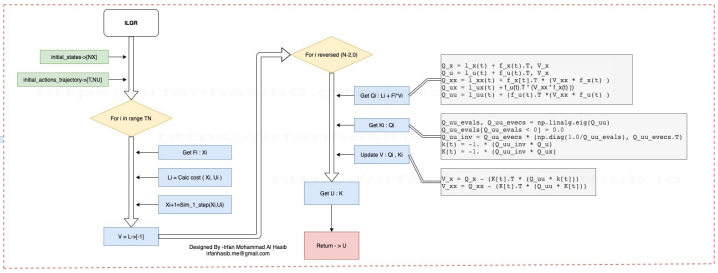
AtsushiSakai/PythonRobotics

Reference

- *AtsushiSakai/PythonRobotics*
- *Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization by -Yuval Tassa*

Appendix : Map Tracker and iLQR





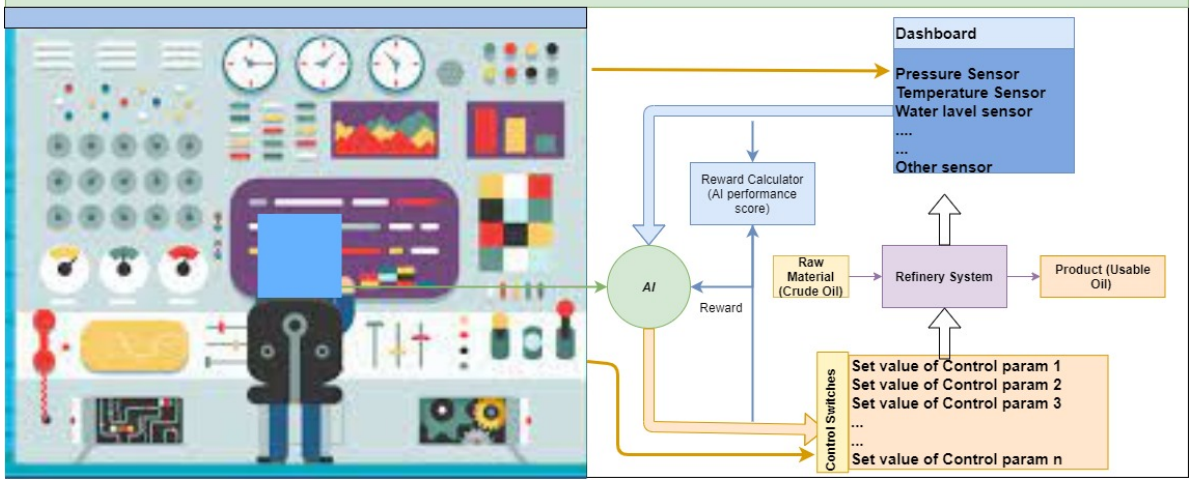
Go Back to Table of Content

Professional AI Solutions

Go Back to Table of Content

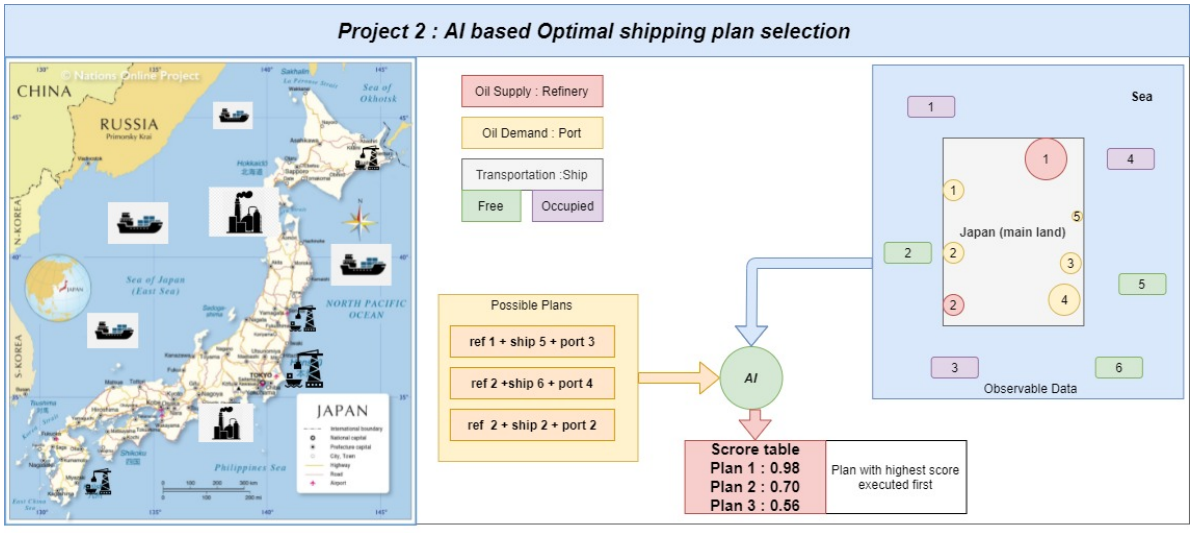
These are some of my profesional AI solutions. Just the outline of the projects (without any technical information) is shared for confidentiality reasons.

Project 1 : AI based process control automation

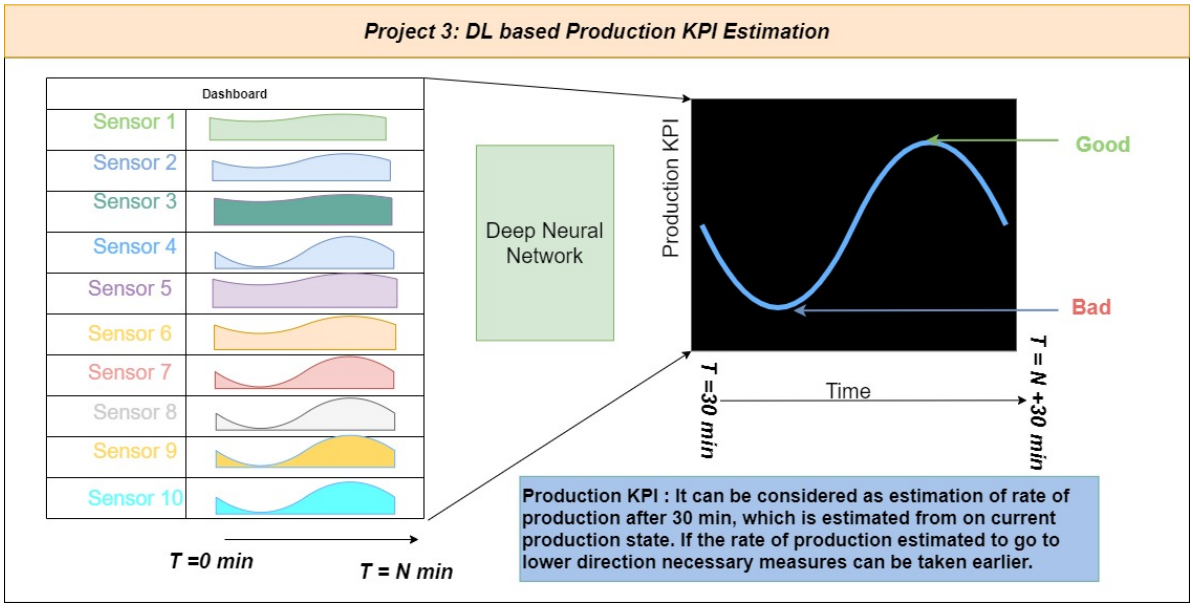


Go Back to Table of Content

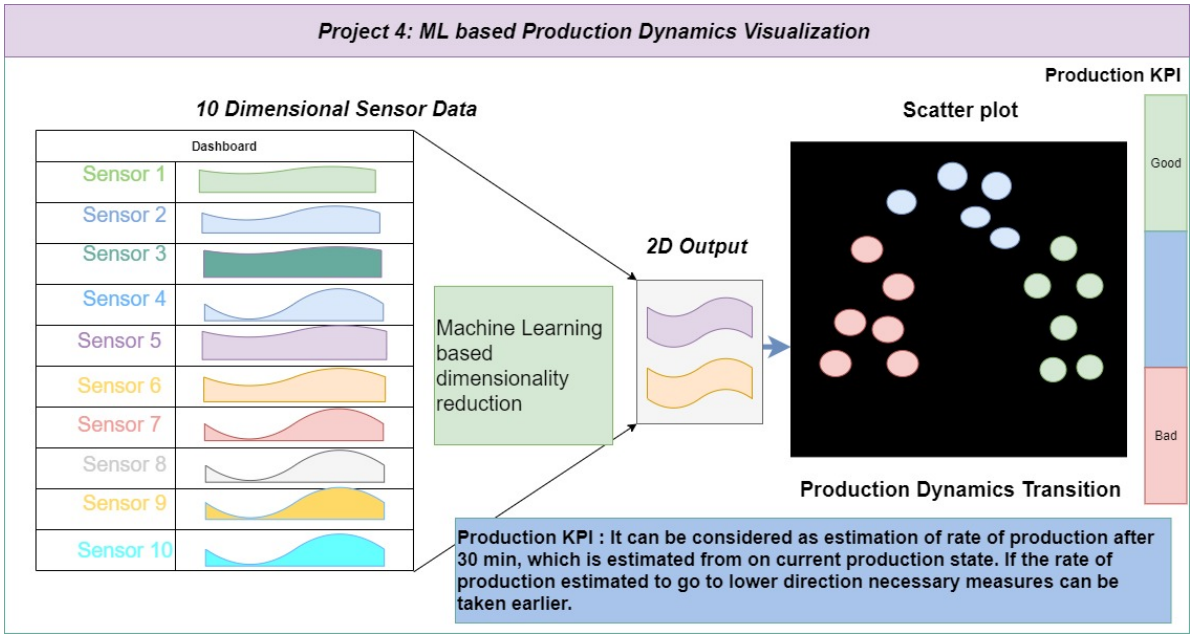
Project 2 : AI based Optimal shipping plan selection



Go Back to Table of Content



Go Back to Table of Content



Go Back to Table of Content

Professional Embedded System Projects

Go Back to Table of Content

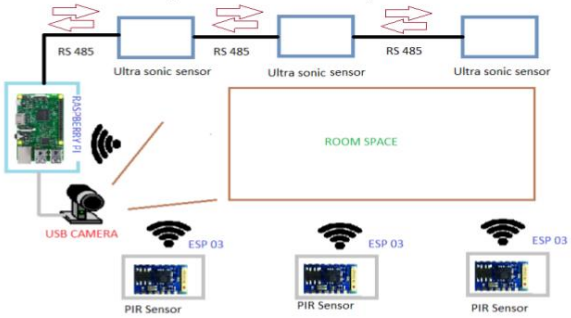
All these projects I did as an employee of Pi Labs BD Ltd. www.pilabsbd.com

VAULT SECURITY SYSTEM - PI Labs BD Ltd.

I was co-developer of vault security system v-1.0

I have developed -

- The wireless PIR (passive infrared sensor) sensor nodes with ESP 03 (ESP 8266 based Wi-Fi transceiver module) for sending data to raspberry pi server.
- Motion detection with USB camera on raspberry pi
- Sending Ultra sonic sensor data to raspberry pi via RS485 protocol.
- Worked on low power modes of operation of ESP 03.

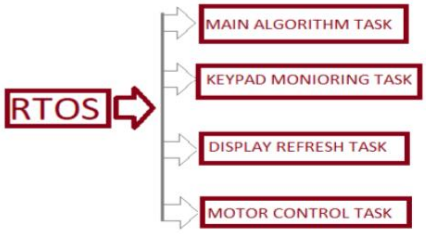


Go Back to Table of Content

SYRINGE INFUSION PUMP Pi Labs Bangladesh Ltd.

I was main developer of syringe infusion pump.

- The pump was programmable for pumping liquid at desired rate for desired time period.
- The system was designed on AVR platform using RTOS (Real time operating system).
- Keypad and Display for programming the pump.
- Stepper motor controlling with TB6560 driver with 1/32 micro step controlling.
- 3 modes of operation. They are – Automatic Mode, Manual Mode and Refill Mode.
- Pumping volume precision up to 0.01 cc/s



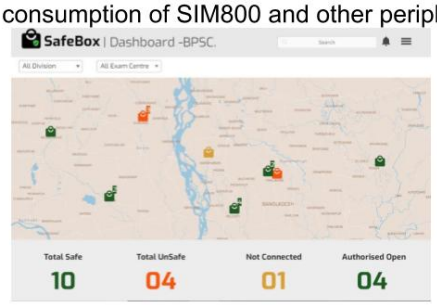
6

Go Back to Table of Content

SafeBox – Box tracking Device for Pi Labs BD Ltd.

I was co-developer of vault Safe Box v-1.0 a box tracking and box safety status reporting device.

- I have worked on sending data packet to the server at regular interval via GSM network with SIM800 with least data loss.
- I have developed an algorithm for keeping back up of box status data on SD card (specially when GSM network strength is low) with FATFS on AVR platform by SPI interfacing with SD Card and reporting backup to server when network is back again.
- I have also worked on the battery backup time enhancement and researched on power consumption of SIM800 and other peripherals.

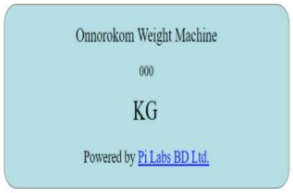


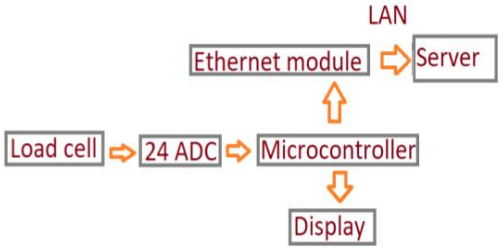
5

Go Back to Table of Content

Online weight machine for Pi Labs Bangladesh Ltd.

- The weight machine was developed on AVR microcontroller platform.
- The weight machine capable of showing up to +/- 3 g precision using 24 bit ADC(Analog to digital conversion).
- The weight machine by e capable of sending measured weight directly to server by using LAN connection.
- I have also developed prototype of version two with native raspberry pi server with python-flask and Wi-Fi support.





7

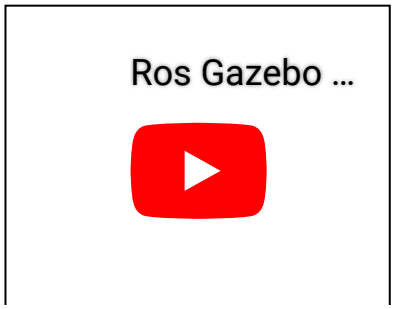
Go Back to Table of Content

ROS : Custom 2 Link robot robot inspired from rrbot

Go Back to Table of Content

I have developed a simple 2 link robot inspired from rrbot .It is capable of obtaining Laser Scan values from laser sensor and Camera feed from both overhead camera and camera attached to arm front. I have provided the URDF and Gazebo config files and python driver code for publishing driving command and listening to sensor values.

- *Simple 2 Link robot : Driver (Python3)*
- *Simple 2 Link robot : description file (URDF) Link*
- *Simple 2 Link robot : Controller Code(Launch file) Link*
- *Simple 2 Link robot : Gazebo config Link*
- *Simple 2 Link robot : YouTube Link*

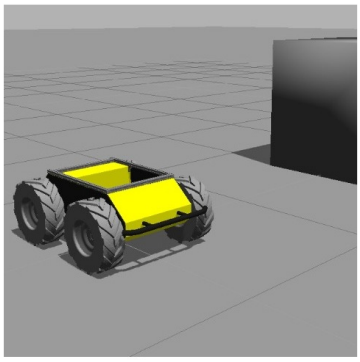


ROS : Husky, URbot driver in Python

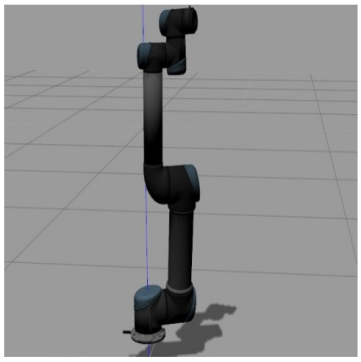
Go Back to Table of Content

Husky is a terrain travelling robot, developed by Clearpath Robotics. URbot(UR-5) is one of the robotic arms developed by Universal-Robots. Below are the GitHub link of the basic driver code I have developed for these two robots. The driver software is capable of publishing driving commands and listening to sensor valued using ROS.

- *Husky Controller(Lunch File)*
- *Husky Driver (Python3)*
- *Universal Robots (UR5) Driver (Python3)*



Husky Robot

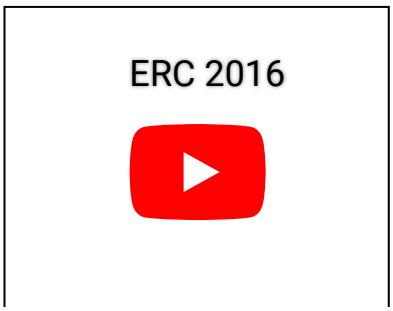


UR-5 Robot

University Rover Challenge - 2016

Go Back to Table of Content

Along with my team, Interplaneter I have participated in University Rover challenge, 2016 at Utah, USA. Our team attained 5th position in Phobos final. I was in charge of Robotic Arm Design and deployment. The Competition is organized by Mars Society, USA annually for college students world wide. URC 2016 Result, video link YouTube



Academic Project and Thesis:

Go Back to Table of Content

Thesis Book (loading can be slow, In that case download will be faster)

- *My undergrad project of intrumentation and measurement course.*
- *My undergrad thesis.*

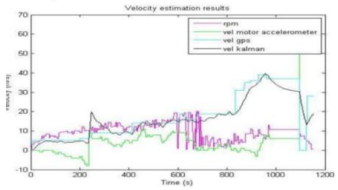
Academic Project and Thesis

Level-3 Term-1 project-Remote Controlled Surveillance Robot:

- Development of a Remote c ontrol Surveillance bot Hardware and Software Platform for soil sample collection and data acquisition and transmission.

Level 4 -Thesis

- Development of a precision vehicle speed measurement and data logging system for making a Drive Cycle of Bangladesh. (A study of precision velocity measurement system including Inertial Measurement system (IMU), GPS and sensor fusion with kalman filter.)



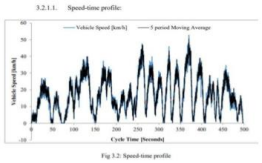


Fig 3.2: Speed-time profile

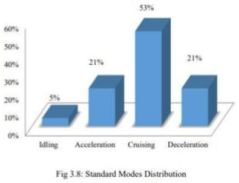


Fig 3.8: Standard Modes Distribution

8

Go Back to Table of Content

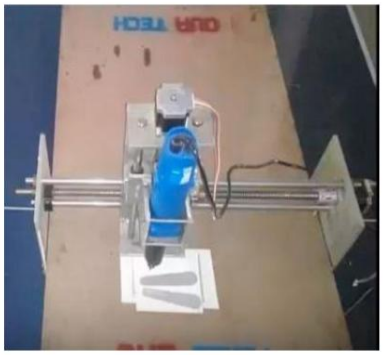
Embedded System & Robotics Projects Personal (Undergrad) :

Go Back to Table of Content

- *Hobby CNC machine 'Ourtech v 2.0' and 'Ourtech v1.0', Desktop CNC Machine.Video Link*
- *Interfacing ov7670 camera sensor with atmega 32 and using object tracking algorithm on AVR platform (Feb-May - 2013)Video Link*
- *Software platform for controlling Robotic arm with openCV, python and Raspberry pi (Apr-Sep,2014)Video Link*

Desktop CNC Machines

- I have developed 2 Desktop CNC machines as my hobby project.
- Both were capable of cutting 2D designs from G-Code from Computer via Serial.
- I have used GRBL Firmware(Open source) with AVR microcontroller.
- I have also designed H-brige drivers for stepper motors with mosfet.



11

Go Back to Table of Content

OBJECT TRACKING WITH ROBOTOC ARM ON AVR PLATFORM

- Object Tracking with minimal hardware. i.e. 8 bit microcontroller and camera sensor.
- Interfacing OV-7670 Camera sensor with Atmega-32 microcontroller using SCCB (Serial Camera Control Bus).
- Implementing Sobel filter on AVR platform.
- Note : Object tracking can perform far better on computer platform or raspberry pi but I wanted to build the system with 8 bit microcontroller and camera sensor.

9

Go Back to Table of Content

Software platform for controlling robotic ARM

- Developed a User Interface for Controlling a robotic arm with – Joystick / Mouse/ Keyboard.
- UART to USB support was added for connecting PC with microcontroller that is controlling the servo motors.
- Developed Inverse Kinematics calculations for robotic arm for converting xy co-ordinates to servo angles.
- Added robot pose visualization support.
- The GUI was developed with Tkinter and OpenCV.

10

Go Back to Table of Content

In […