AI Documentation

image_scraping.py

- This code is for scraping the satellite image from google maps as soon as the farmer enters a field and clicks the button to send gps coordinates to the interface. After receiving the coordinates which is in degree, minute, second format is now converted to degrees after passing the coordinates to another code implemented in ***convertor.py***. Now ***convertor.py*** returns the coordinates and that coordinate's image is scraped from the google maps with a pause of 2 seconds (It scrapes piece by piece because scraping entire area altogether is not possible. For that purpose, we have used tile stitching technique i.e. image is segmented into a fixed areas and each area is scraped order-wise and stitched together to form an entire image. 2 second pause is there because we are scraping pieces continuously and google doesn't allow repetitive requests operating too frequently).

convertor.py

- Simply converts coordinates from degree, minute, second format to degrees.

road_seg.py

- This code is used to segment the roads from other areas. Other areas are suppressed and the roads identified are marked with thick black lines. This is necessary because our model requires setting up of warehouses for stubble collection and for optimal places, it is better that the warehouses should be set alongside roads so that it would be convenient for trucks, tractors, etc. to carry stubble from these warehouses to their destinations. This is now incomplete as we have to do a little research on how to prevent our model to detect the optimal points for setting up the warehouses at places where there are already houses, fields, or any other installations. We need empty land for setting up of warehouses so that it doesn't coincide with any other installations.

detect_edge.py

- In this code, we import the coordinates and mark it accordingly on the satellite image procured. Prior to marking, we have applied a **Holistic-nested edge detection (HED)** to identify the boundaries in the image. Now we have filtered image, and on this image, we will perform contour detection to mark the boundaries identified in the real image. For the demo purpose and also because we had less time before prelims, we have used classical method of contour detection which performs well enough but not up to our expectations. Also, HED filter was also not up to the mark because we used the dataset BSD5000 and took a pretrained caffe model (hed_pretrained_bsds.caffemodel). After training this model, we detected boundaries of the fields. Then we marked the coordinates on the image with a blue circle. Now

we have to check which field or boundary contains this coordinate and according to that it will identify the contour. Because contours were not smooth, we applied convex hull to take average of the points in the boundary and according to the farthest point, it should draw the boundary. With this method, we were able to contain our field into the contour. After that, area was calculated of the field which came out to be 5058 m^2 at the time of demo/ presentation. When we cross-checked with the google earth imagery, the actual area was 5024 m^2 which was nearly perfect.

But what about the problem of the HED filter and contour detection method which were not perfect?

Solution : For HED filter, we have planned to make our own dataset. We will take few images and mark the boundaries. After that, we will create a Generative Adversarial Networks (GANs) using Pytorch and train it after feeding these images. In this way, we will get some more dataset, as GANs will generate fake images with many variations. In this way we will be able to create a huge dataset for training. Then we will create HED architecture in Pytorch and train with these dataset. In this way, we will get a perfect filter.

For Contour Detection, we will implement it with a new algorithm known as **globalized probability of boundary**. Using this we will be able to get rid of irrelevant contours and obtained perfect closed smooth boundaries.