Reforestration Using Drones and Deep Learning Techniques

GANDHAM VENKATA SAI LOHIT

Abstract— Deforestation is proving to be a major contributor towards climate change. The earth houses around 3 trillion trees (30 % of the planet) and according to a report by TIME, around 15 billion trees and being cut down each year [1]. It is reported that there has been a 46% reduction in the number of trees since the start of human civilization and only 400 billion trees exist in today's world, according to Thomas Crowther, Yale University. At. this rate it is estimated that the rainforests may disappear in around the next 100 years. Due to reduction of trees the earth undergoes the phenomenon of 'Green House Effect', wherein the greenhouse gases majorly carbon dioxide ,methane, nitrogen dioxide and chlorofloro carbons, traps the sun's light (IR range) in the atmosphere more than required to maintain an equilibrium, which cause the earth to heat up, proving fatal for life forms on earth. When trees are cut down the carbon dioxide which they store to maintain an equilibrium will now be released in to the earth atmosphere, which adds to the cause. The main contributors towards the greenhouse gases are the agriculture and forestry activities. These include forest fires, tree chopping for agriculture and husbandry, urban sprawling and logging of trees. Due, to these activities the mother nature is slowly losing its capacity to reforest and existing tools and nursey supply chains are largely inadequate to fill the gap, and this is the challenge we would like to address using our project "Reforestation using drone and deep learning". We study about how drones and be used for reforestation and making a working prototype of the same. Using drone reforestation, we can tackle a multitude of problems. It is 9 times faster than other human planting systems, staying airborne allows efficient and quick travel which saves time and covers more area in less time with more efficiency. I have also devised a method where deep learning techniques can be used efficiently for tracking deforested land by a drone and sowing the required seeds at that exact location by hovering to that point or dropping the seed airborne.

Index Terms— Agriculture, Deep Learning, Deforestation, Drones, Forestry, Greenhouse effect, Reforestation.

1. Introduction

In today's world deforestation poses a very serious forest, and contributes to reduced tress, loss of habitat and more devasting effects. Taking the example of the amazon rainforests which cover almost 50 % of south America, loses almost 0.5 % of its area due to deforestation [2]. Also, one of the major contributors towards loss of tress are forest fires and flash fires. According to the world resources institute the tropics lost almost 12 million hectares of tress natural tree cover in 2019. Recently, Bolivia also experienced a record-breaking loss of tress due to deforestation [3]. Therefore, to especially reduce the loss of trees due to forest fires, I have come up with a solution that may reduce this problem to a major extent. I proposed a solution, wherein we can use drones to tackle this problem. I have identified few classes which can help in identifying different types land in a region, and plant seeds in that particular location. The drone and hover on the deforested land effected by deforestation and then shoot a seed at that particular location, which can help in regeneration of trees at that particular location. Also, with advances in satellite imagery, the detection of such lands is becoming more and more simple. To, do this I have assembled parts to make the drone, which includes parts like ESC's, brushless motors, carbon frame, 2200mah battery, RC controller etc. Also, to implement the deep learning algorithm to identify our required lands is done on python, and real time implementation and transmission of data will take place using a raspberry pi model B+ module. The major objective of this project is to make a working prototype of the drone using KK 2.1.5/APM 2.8 flight controller and a deep learning algorithm with a raspberry pi serving as its brain. These are the objectives we are intending to achieve is conducting a stable flight of our drone using a APM 2.8/KK 2.1.5 flight controller, and analyzing its stability, mounting the raspberry pi and camera onto the drone to obtain stable image of a targeted 100 X 100 m region and making a comparatively accurate deep learning classifier using classifies a set of images into our required classes, so that we identify our region of interest. In doing so I would like to make sure that the following boxes are ticked which are, reducing deforestation, minimizing the use of manual labor in plantation, encouraging the use of AI to solve real time problems and remote sensing using drones. I am going to use the MobileNet V2 architecture, DenseNet121 and Resnet152 to make a comparative analysis of their accuracies.

- *GANDHAM VENKATA SAI LOHIT is currently pursuing Bachelor's degree program in Electronics and Communication engineering in Amity University, Noida, U.P, India. PH-+918448700459. E-mail: lohit0399@gmail.com*

2. Literature Review

Cohen et al shown and proved that forest imagery can show the pacific northwest. From then on, many works from Hansen et el. [4] and Popatov et al. [5] have shown data Landsat data and can used to estimate any important features. It is only in the recent years that people started working on images from drones or Unmanned Aerial Vehicles (UAV), and work in this fielding have been growing at a rapid pace. In this paper I used recent deep

learning algorithms like the Mobile Net, DenseNet121 and Resnet 152 for classification of our required data. Also, the drone is made on the APM 2.8 flight controller and was stabilized on various parameters like roll, pitch and yaw which stabilize the drone. The drone is run in loiter mode when it is airborne. Also step like ESC calibration and Controller binding also was performed to maintain and smooth transmission and reception.

2.1 MOBILE NET V2

This algorithm was used mainly due its ease of memory occupation and also lesser number of computations. It uses something called the Depth wise Separable Convolutions Which reduces the number of computations and cost drastically, also it reduces the machine's computational power which increases the speed. In this architecture we have two types of blocks namely the, Residual Block with stride 1 and Residual block with stride 2 for downsizing parameters. Also, this type of architecture consists of 3 types of layers namely, 1X1 convolution with ReLU acting as it's activation function, depth-wise convolution layer and 1X1 convolution with no non linearity. The unique feature in this architecture is the depth wise separable convolutions which uses basically two steps they are Depth wise convolutions and Point wise convolutions.

The depth wise convolutions (difference is shown in figure 1) involve convolution if each channel, unlike the traditional convolution wherein all the channels are convolved in a single go [5]. The output obtained from stacking such outputs will be fed into the second stage namely the point wise convolution stage. Here, a filter namely KPConv, which are 1x1 convolution block of the required depth are used to convolve over the entire block to give the output of our required depth. This method reduces the number of steps drastically. The architecture of MobileNetV2 is shown in figure 2.[6]

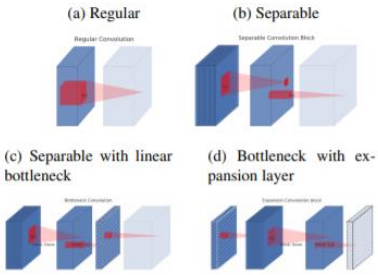


Figure 1.This figure shows the Regular and Seperable Convolutions

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

MobileNetV2 Overall Architecture

Figure 2. The Overall Architecture Used.

2.2 DENSE NET 121

Dense 121 is again a next step to counter the problems that arises as the depth of the convolutional layers increases. The motivation to develop this architecture is that, as the convolutional layers become deeper the way from the input to output become quite big, and the information by the time it gets to the end, tends to get vanished. Therefore, to counter this problem, the dense networks uses: -

a) Highway Networks
b) Residual Network
c) Fractal Networks

Also, to ensure that the output does not degrade, all the layers are simply directly connected to each other, while doing so redundant feature are not learned again and again, and hence number of parameters reduces drastically. (See figure 3). In, this project we use these existing states of art model and modify them to our requirements using something called transfer learning. Here we prune the models trained on the ImageNet dataset and reduces the number of output layers to our requirement. [7]

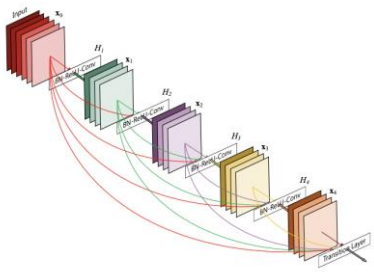


Figure3. The DenseNet block

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6	[1 × 1 conv, 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12	[1 × 1 conv, 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv, 3 × 3 conv] × 24	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48	[1 × 1 conv, 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv, 3 × 3 conv] × 16	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 32	[1 × 1 conv, 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 4.DenseNet Architectures

2.3 RESNET 152

The ResNet152 stands for residual networks consisting of 152 layers. The motivation which led to this architecture is that, as the number of layers increases, it was found that the output degrades drastically and even the performance drop below that of the initial image. Therefore, the ResNet 152 came up with two special features in its architecture: -

a) Skip connections.
b) Heavy batch normalization.

The advantages of having these features is that the skip connections tend to stop the problem of vanishing gradient which tends to occur if the layers become deep and also, it retains the input, hence it does allow the output to degrade beyond the threshold set by the input image itself. Also, the heavy batch normalization step helps to avoid the problem of overfitting in our architecture. The skip connections are made directly from the input to the output. [8]
Also, this design uses the bottleneck layer (1x1 convolutions), as the layers are deep and hence this leads to more time complexity. A bottleneck layer tends to remove this disadvantage. This layer is added in the start and end of a residual block, this is inspired from the GoogLeNet architecture.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

The overall architecture for all network

Figure 5. The ResNet Architectures

2.4 DRONE PARAEMETERS

Now, coming to some literature about drones, drones are stabilized by a lot of parameters and, all these parameters and variables must work in tandem in order for the drone to work. Out of these the most important are roll, pitch and yaw parameters.[9]
Yaw: This direction refers to the direction in the front, this rotation takes places on the plane in which the drone exists. It can rotate either in the clockwise or anti clockwise direction.
Pitch: This dimension helps the drone move, the drone tilts forward or backward based on the orientation of it's front. I drone tilts forward it moves in the forward direction and if

the drone tilts backwards it moves in the backward direction.
Roll: This helps the drone move sideways, helping it to 'roll'. This does not vary the drone's height or altitude but helps the drone move left or right. [10] (see figure 6).

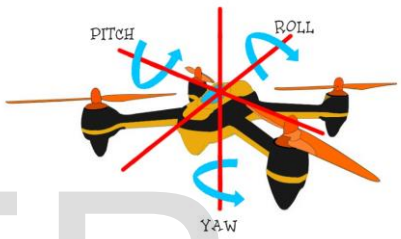


Figure 6. The visual representation of the Drone Parameters

3. DATASET

The dataset which I have used was a custom dataset obtained from the following sources: -

1.Aerial Drone Dataset – Kaggle

2.UAV Dataset – Kaggle

3.Locality Park – Images from the drone

4.Open source photos

There were a total of 1574 images segregated into 5 classes as mentioned before. The data was split into a training to testing ratio of 70:30

4. PROPOSED COMPONENTS AND MODELS

4.1 THE SOFTWARE COMPONENT

I worked on the 3 deep learning algorithms, as elaborated earlier in the literature survey section, they are: -

1. MobileNetV2
2. ResNet152
3. DenseNet121

Using these algorithms, we are going to classify our custom-made dataset into the following classes: -

1. Road
2. Deforested Land
3. Forested Land
4. Rocky Terrain

Figure 7. The classes in which I classified

5. Buildings

We applied transfer learning on each these models and pruned the last layer to our required number of classes (in this case 5).
Based on the accuracies obtained from these algorithms I selected the best algorithm, to work on to make my prototype.

4.2 The Hardware Component

In the hardware-based work we have the construction of the drone and stabilizing its variables and parameters which were elaborated in the literature review section. The following components have been used in construction of my drone: -

1. Raspberry pi, Arduino and Raspberry pi camera module for image processing and deep learning implementation.
2. Python 3.7
3. 4 X 30 A Brushless ESC (Electronic Speed Control) Motors.
4. APM 2.8 flight controller.
5. F450 Quadrotor Frame with integrated PCB.
6. 4 X RC Brushless Motors (1000A)
7. 2 X 1045 Propeller Pair. (CW & ACW).
8. RC Transmitter and Receiver.
9. GPS Module (external)

Once, I assembled the drone the too stabilize it I have used the mission planner software to calibrate the settings and stabilize the drone. The three important parameters mentioned earlier, namely the roll, pitch and yaw are wrapped in a device called the gyroscope, this device is used to calibrate these parameters. It measures the rate of rotation, keeping the drone stabilized or balanced, even in poor weather conditions [11]. No matter where you orient your drone the gyroscope will always maintain its stability.

Also, I have the flight controller i.e. the APM 2.8, comes with geofencing to create a virtual boundary that triggers responses, when the drone comes and goes in and out of a particular area. This feature can be enabled to make the drone fly in only a restricted area, which will be decided by us. Also, once assembled the drone must be calibrated to the Radio controller and also the GPS coordinates must be set, by the 4 geo-stationary satellites which exists above the earth's atmosphere. Also, the compass but be calibrated so that the internal compass in the controller will have a sense of direction.

Figure 8. Few of these components are shown

5. Budget

The budget for making the drone and procuring the raspberry pi module costed around 10,000 to 11,000 rupees. I wanted to keep the budget, as minimum as mostly, and hence APM 2.8 and raspberry pi were used. To get much supreme industrial grades performances we can use a Pixhawk for the flight controller, and much better frames, to make the drone crash resistant.

6. Results

6.1 The Software Component

The following results attached below are result shown from the architectures which have been implemented using python.
Language Used: python
Libraries Used: TensorFlow, Keras, NumPy and Matplotlib.
Also I have used the "Categorical Cross Entropy" as my loss function and the Adam optimizer for all my architectures, to maintain homogeneity. The equation for categorical cross entropy is shown below

$$CCE(p,t) = -\sum_{c=1}^{C} t_{o,c} \log\left(p_{o,c}\right)$$

In my case I have 5 classes, as mentioned earlier.

I have obtained the following results for 50 epochs for the 3 architectures, which were already mentioned: -

The results are culminated in the figure below

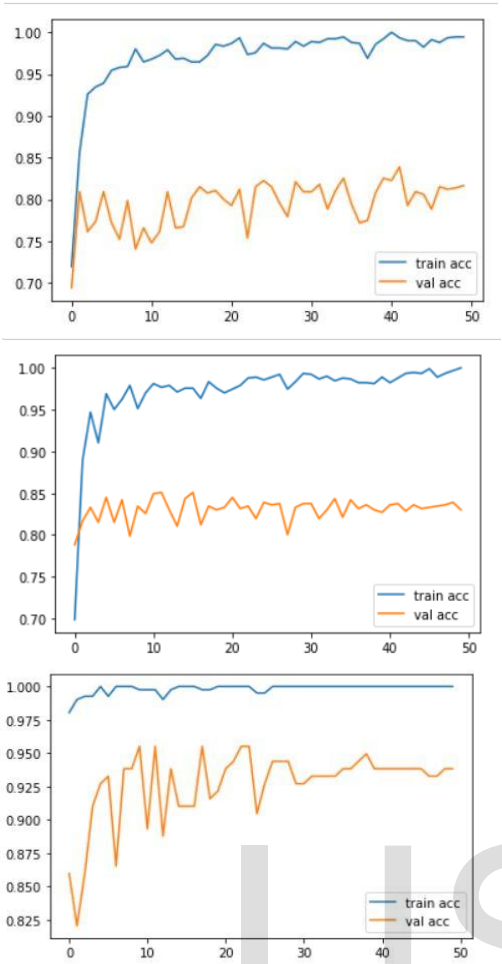


Figure 9. Results obtained for MobilenetV2, DenseNet121 and ResNet153 architectures respectively.

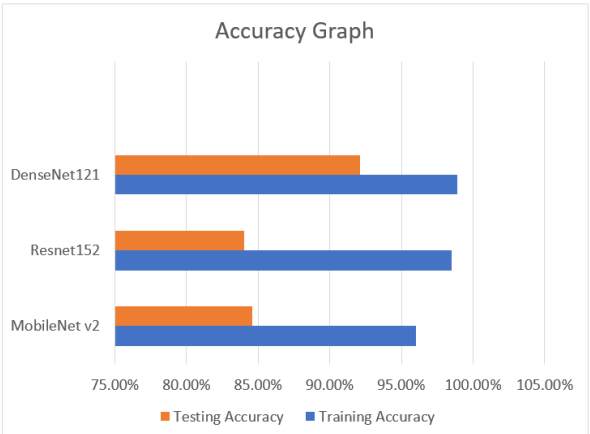


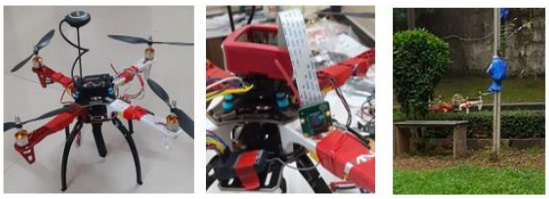
Figure 10. Accuracy Graph obtained from the results

Fields of comparison	Mobile Net v2	Dense Net 121	Resnet152
Size on Disk	14 MB	33 MB	232 MB
Number of parameters	4.2M	8M	60M

Figure 11. The Table shows the on-disk space and parameters involoved in these architectures.

6.2 THE HARDWARE COMPONENT

The drone which I assembled and the raspberry pi module is shown below. Also, few images of me flying the flight is shown below.



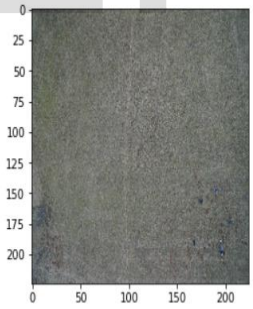
(a) (b) (c)

Figure 12.
(a) The fully operational drone
(b) Show my camera model on the Raspberry pi Model B+
(c) Shows my drone flying and cpaturing images for my Dataset

Therefore, from the above observations I found that DenseNet121 gave the best accuracy that is 93.1 %. Few predictions made by the DenseNet121 are shown.

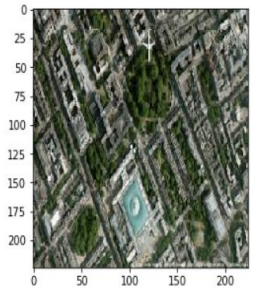
CLASS	DESCRIPITION
CLASS 0	Buildings
CLASS 1	Deforested Land
CLASS 2	Forested Land
CLASS 3	Road
CLASS 4	Rocky Terrain

Figure 13. The reference chart.



```
[[2.2752209e-18 1.0000000e+00 2.4401340e-20 1.3920327e-22 5.8696349e-19]]
```

Figure 14. This shows class 1, i.e. Forested Land (as the second element has the maximum value)



```
[[1.0000000e+00 6.0561943e-17 5.1460914e-14 5.1527572e-19 0.0000000e+00]]
```

Figure 15. This shows class 0, i.e. Buildings (as the first element has the maximum value)

7. FUTURE SCOPE

In the future, I intend to getting all the different components which I made here in this project, into one **prototype.**

- I also intent to work on the plant sowing mechanisms which I can inculcate into the drone.
- I also want to include a IoT based system design to make the drone 'smarter'.
- I also, want to include a gimbal and a robotic arm to make the drone more robust and multitasking.
- I also want to try running or burning the algorithms which, I have made in this project onto my raspberry pi, and see how the project works real-time.

ACKNOWLEDGMENT

I would like to thank my professor and project guide Mr. Haneet Rana, who helped me complete my project successfully. Also, I would like to thank the people who directly or indirectly contributed towards success of the project.

REFERENCES

[1] Max Roser (2017) - "Forests". Published online at OurWorldInData.org. Retrieved from: 'https://ourworldindata.org/forests'

[2] Aaron Loh, Kenneth Soo for "Amazing Amazon: Detecting Deforestration in our Largest rainforest", MS CD'17, Standford University,2017.

[3] P. Rizzoli, J. L. Bueso Bello, A. Pulella, F. Sica and M. Zink, "A Novel Approach to Monitor Deforestation in the Amazon Rainforest by Means of Sentinel-1 and Tandem-X Data," IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, 2018, pp. 192-195, doi: 10.1109/IGARSS.2018.8518483.

[4] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pages 1135–1143, 2015.

[5] Soravit Changpinyo, Mark Sandler, and Andrey Zhmoginov. The power of sparsity in convolutional neural networks. CoRR, abs/1702.06257, 2017.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[7] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 2261-2269, doi: 10.1109/CVPR.2017.243.

[8] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.

[9] T. C. Mallick, M. A. I. Bhuyan and M. S. Munna, "Design & implementation of an UAV (Drone) with flight data record," 2016 International Conference on Innovations in Science, Engineering and Technology (ICISET), Dhaka, 2016, pp. 1-6, doi: 10.1109/ICISET.2016.7856519.

[10] R. Labayrade and D. Aubert, "A single framework for vehicle roll, pitch, yaw estimation and obstacles detection by stereovision," IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683), Columbus, OH, USA, 2003, pp. 31-36, doi:

[11] G. A. Venkatesh, P. Sumanth and K. R. Jansi, "Fully Autonomous UAV," 2017 International Conference on Technical Advancements in Computers and Communications (ICTACC), Melmaurvathur, 2017, pp. 41-44, doi: 10.1109/ICTACC.2017.20.