Smart Multipurpose Unmanned Aerial Vehicle with 3D Depth Mapping & Spectral Imaging

Dr. Syeda Gauhar Fatima[1], Abdul Aleem[2], Saquib Mohiuddin[3], Abdus Samee[4]
[1]Professor, ECE Department, DCET, Telangana, India
[2]Assistant Professor, ECE Department, DCET, Telangana, India
[3]Student, ECE Department, DCET, Telangana, India
[4]Student, ECE Department, DCET, Telangana, India

Abstract - *Unmanned Aerial Vehicles are becoming more and more popular to meet the demands of increased population and agriculture. It is also expected that drones will take a major role in the connected smart cities of the future. They will be delivering goods, merchandise and serving as mobile hotspots for broadband wireless access and maintaining surveillance and security of smart cities. Drones equipped with appropriate cameras, sensors and integrating modules will help in achieving easy, efficient, precision agriculture. The proposed solutions which includes 3D ToF Camera for superior depth mapping and semantic segmentation using multispectral imaging to aid in SAR Missions. related to these drones, if integrated with various Machine Learning and Internet of Things concepts, can help in increasing the scope of further improvement. Proposed solutions that can be integrated into the drone using various microcontrollers such as raspberry pi.*

***Keywords*: Unmanned Aerial Vehicle, Image Processing, MATLAB, 3D TOF Camera, Semantic Segmentation, Multispectral Imaging, Surveillance, Depth Mapping of Terrains**

1.INTRODUCTION

The project aims at designing a drone which is used for surveillance systems. The UMAV will have two wireless video cameras on it i.e. ToF camera for Depth Mapping and High-Resolution Video Camera for Streaming. The chopper is controlled using RC remote control.

RF Communication ranges in between 30 KHz to 300 GHz. RF communication works by creating electromagnetic waves at a source and being able to pick up those electromagnetic waves at a particular destination. These electromagnetic waves travel through the air at near the speed of light. The wavelength of an electromagnetic signal is inversely proportional to the frequency; the higher the frequency, the shorter the wavelength. The controlling device of the whole system is managed by MCU at control center. The MCU gets the input from RF receiver which receives the data transmitted by pressing control buttons in RF remote. This data is processed and acts accordingly on chopper motors. The captured video images are transmitted through AV transmitter and are received by AV receiver and displayed on PC/ Laptop with the help of TV tuner card. We can also see the video recording in the PC.

This project finds its major applications in military while we are monitoring larger areas like political canvassing, cricket stadiums, international conferences, worship places, banking etc. This project assures us with more reliable and confident security system

1.1 Functional Operation: How Do Drones Work?

One of the most common flying drone designs is the quadcopter, is a type of drone that is lifted and propelled by four rotors. The concept of quadcopter vehicles is not new; manned quadcopters were first experimented with in the 1920s, but their effectiveness was hampered by the technology available at the time.

However, with the advancement of electronic technology in sensors, batteries, cameras and GPS systems, quadcopters have become widely employed over the past decade both recreationally and commercially.

The 4 propellers of a quadcopter are fixed and vertically orientated. Each propeller has a variable and independent speed which allows a full range of movements. Shown below is the different propeller combinations that facilitate different drone movements.

2. BASIC PRINCIPLE

This is unlike conventional helicopters which are controlled by propellers with blades that dynamically pitch around the rotor hub. The components required for blade pitch are expensive which is one of the reasons quadcopters are becoming so common with recreational UAV enthusiasts.

Table -1: Core Components of a Drone

Preparation of Manuscript			
Chassis	1	Battery	1
Propeller's	4/6	Radio Receiver	1

Motors	4/6	RF Video Transmitter	1
Electronic Speed Controllers	1	Main Control Unit	1
Flight Controller	1		

3. METHODOLOGY

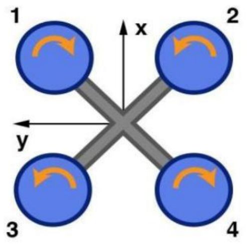
The initial step would be to create a chassis (frame) where raspberry pi, the main component would be mounted. Lithium battery will be used as they have the best ratio of weight to power. Motors would be used to spin the propellers. Using accelerometer and gyroscope, Electronic Speed Controller (ESC) unit will control the four motors and provide stability to the quadcopter. Remote Controller (RC) will provide input to the Quadcopter. Quadcopter flight dynamics for controlling the altitude a remote controller (RC) is used. When the controller of the quadcopter is moved up or down the propeller speed is adjusted causing the quadcopter to gain or lose altitude. Thrust is a type of force. When a system accelerates mass in one direction, the accelerated mass will cause a force of equal magnitude but opposite direction on that system. The force applied on a surface in a perpendicular direction or normal to the surface is called thrust.

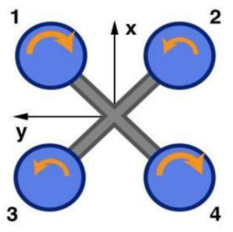
4. FLIGHT CONTROL OF UAVS

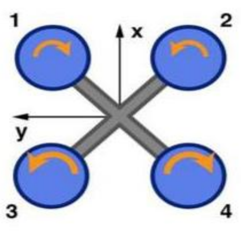
Each rotor produces both a thrust and torque about its center of rotation, as well as a drag force opposite to the vehicle's direction of flight. If all rotors are spinning at the same angular velocity, with rotors one and three rotating clockwise and rotors two and four counterclockwise, the net aerodynamic torque, and hence the angular acceleration about the yaw axis, is exactly zero, which means there is no need for a tail rotor as on conventional helicopters. Yaw is induced by mismatching the balance in aerodynamic torques (i.e., by offsetting the cumulative thrust commands between the counter-rotating blade pairs).

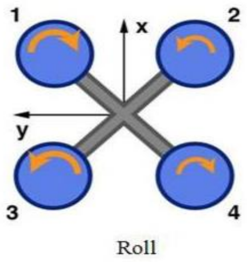
5. QUADROTOR FLIGHT DYNAMICS

Schematic of reaction torques on each motor of a quadcopter aircraft, due to spinning rotors. Rotors 1 and 3 spin in one direction, while rotors 2 and 4 spin in the opposite direction, yielding opposing torques for control.









Flight control of a quadcopter

Hover Yaw Pitch Roll The two pair of propellers as shown in figure1, (1,3) and (2,4) rotates in opposite direction. The pair (1,3) rotates clockwise and remaining pair (2,4) rotates anticlockwise. This combination of rotation produces apposite torque. These results propellers generate vertical lifting force upward which raises quadrotor body in the air and it can move in hover, yaw, pitch, roll, landing and take-off. Pitch and Roll movement can be achieved by altering the speed of any one pair of motor. While other motor pair speeds remain constant. Yaw movement can be achieved by altering the speed of both motor pairs in quadrotor.

a. **Roll:** Rotation around the front-to-back axis is called roll. Roll is controlled with the aileron stick, making it move left of right, if the aileron stick is moved to the left, the quadcopter will fly left, if moved the aileron stick to right, the quadcopter will fly right.

b. **Yaw:** Rotation around the vertical axis is called yaw. Yaw rotates the head of the quadcopter either to left or right, yaw can be controlled through the throttle stick making it to rotate either to the right or left.

c. **Pitch:** Rotation around the side-to-side axis is called pitch. Pitch is the movement of quadcopter either forward or backward. Forward Pitch is achieved by pushing the aileron stick forward, which makes the quadcopter tilt and move forward. Backward pitch is achieved by moving the aileron stick backwards, making the quadcopter, come closer.

6. PROPOSED SYSTEM

When it comes to take on rescue operations, it becomes difficult for the rescue team to go to a specific place and carrying out SAR. To provide a solution to this problem Smart Multipurpose Unmanned Aerial Vehicle with 3D Depth Mapping & Spectral Imaging can be used. The aim is to construct drone equipped with hi-tech 3D ToF and High-resolution cameras which will go to places having unfavorable conditions where a human cannot go in person and provide audio as well as visual aid. This will further facilitate rescue missions held in hostile territories.

7. SYSTEM ARCHITECTURE:

The purpose of the work is to control a quadcopter from a distance by using a computer or RC. The quadcopter movement and control will be carried out by the raspberry pi. The model consists of two blocks which is mentioned below.

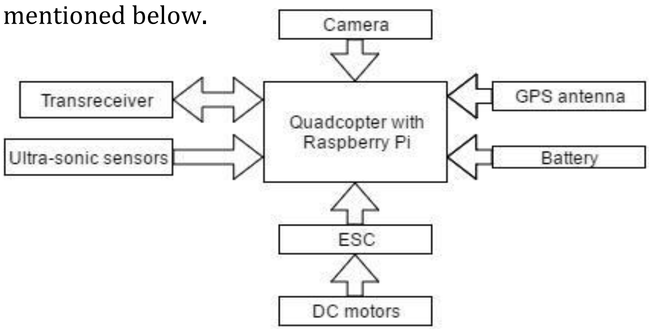


Fig: Block diagram (Quadcopter Side)

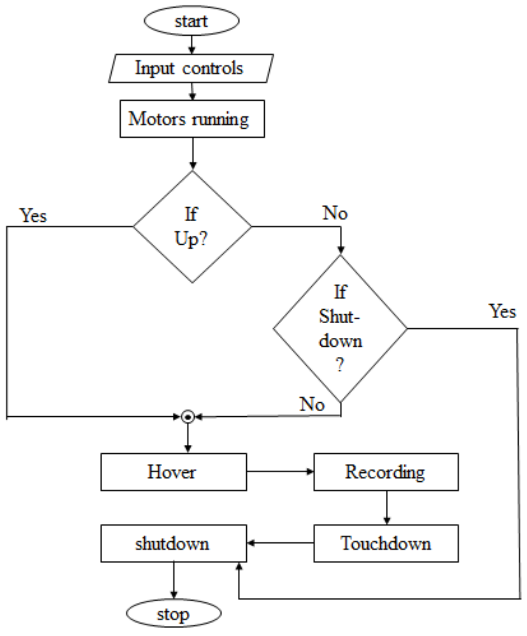


Fig: Operational Flowchart

Fig: Framework of UAV system

The quadcopter is command through the raspberry pi from the computer side block. The command signal will be transmitted wirelessly from the trans-receiver of computer/remote controller side block to the trans-receiver of the quadcopter side and the quadcopter will move accordingly. The quadcopter also includes some features like camera, navigation, etc. All these features are also controlled by the raspberry pi.

8. TIME OF FLIGHT: 3D DEPTH SENSING

A 3D time-of-flight (TOF) camera works by illuminating the scene with a modulated light source, and observing the reflected light. The phase shift between the illumination and the reflection is measured and translated to distance. Figure 1 illustrates the basic TOF concept. Typically, the illumination is from a solid-state laser or a LED operating in the near-infrared range (~850nm) invisible to the human eyes. An imaging sensor designed to respond to the same spectrum receives the light and converts the photonic energy to electrical current. Note that the light entering the sensor has an ambient component and a reflected component. Distance (depth) information is only embedded in the reflected component. Therefore, high ambient component reduces the signal to noise ratio (SNR).

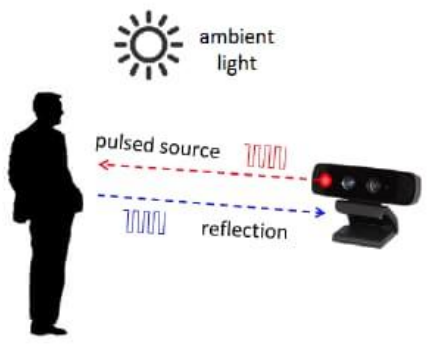


Figure 1: 3D time-of-flight camera operation.

To detect phase shifts between the illumination and the reflection, the light source is pulsed or modulated by a continuous-wave (CW), source, typically a sinusoid or square wave. Square wave modulation is more common because it can be easily realized using digital circuits.

Pulsed modulation can be achieved by integrating photoelectrons from the reflected light, or by starting a fast counter at the first detection of the reflection. The latter requires a fast photo-detector, usually a single-photon avalanche diode (SPAD). This counting approach necessitates fast electronics, since achieving 1-millimeter accuracy requires timing a pulse of 6.6 picoseconds in duration. This level of accuracy is nearly impossible to achieve in silicon at room temperature.

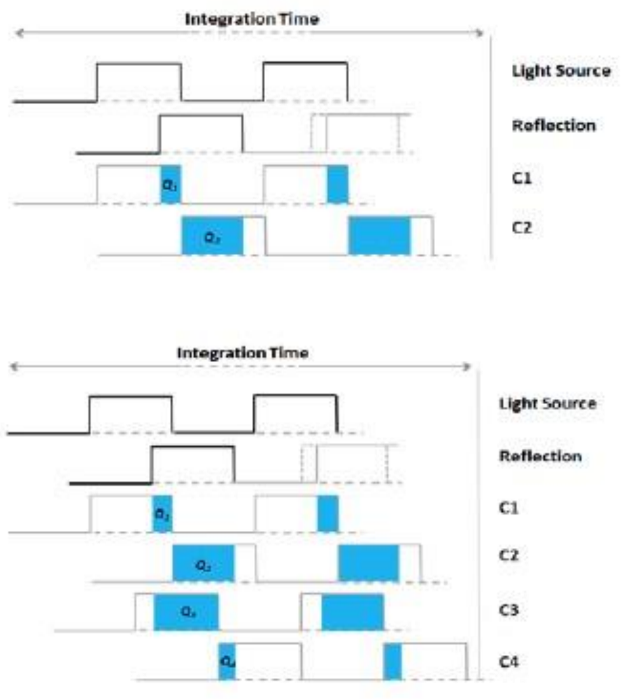


Figure 2: Two time-of-flight methods: pulsed (top) and continuous-wave (bottom).

The pulsed method is straightforward. The light source illuminates for a brief period (âˆ†t), and the reflected energy is sampled at every pixel, in parallel, using two out-of-phase windows, C1 and C2, with the same âˆ†t. Electrical charges accumulated during these samples, Q1 and Q2, are measured and used to compute distance using the formula:

$$d = \frac{1}{2} c\, \Delta t \left(\frac{Q_2}{Q_1+Q_2}\right). \qquad \text{Eq. 1}$$

In contrast, the CW method takes multiple samples per measurement, with each sample phase-stepped by 90 degrees, for a total of four samples. Using this technique, the phase angle between illumination and reflection, Ï†, and the distance, d, can be calculated by

$$\varphi = \arctan\left(\frac{Q_3-Q_4}{Q_1-Q_2}\right), \qquad \text{Eq. 2}$$

$$d = \frac{c}{4\pi f} \varphi. \qquad \text{Eq. 3}$$

It follows that the measured pixel intensity (A) and offset (B) can be computed by:

$$A = \frac{\sqrt{(Q_1-Q_2)^2+(Q_3-Q_4)^2}}{2}, \qquad \text{Eq. 4}$$

$$B = \frac{Q_1+Q_2+Q_3+Q_4}{4}. \qquad \text{Eq. 5}$$

In all of the equations, c is the speed-of-light constant.

At first glance, the complexity of the CW method, as compared to the pulsed method, may seemed unjustified, but a closer look at the CW equations reveals that the terms, (Q3 â€ "Q4) and (Q1 â€ "Q2) reduces the effect of constant offset from the measurements. Furthermore, the quotient in the phase equation reduces the effects of constant gains from the distance measurements, such as system amplification and attenuation, or the reflected intensity. These are desirable properties.

The reflected amplitude (A) and offset (B) do have an impact the depth measurement accuracy. The depth measurement variance can be approximated by:

$$\sigma = \frac{c}{4\sqrt{2}\pi f} \cdot \frac{\sqrt{A+B}}{c_d A} \qquad \text{Eq. 6}$$

The modulation contrast, ð‘ð‘‘, describes how well the TOF sensor separates and collects the photoelectrons. The reflected amplitude, ð´, is a function of the optical power. The offset, ðµ, is a function of the ambient light and residual system offset. One may infer from Equation 6 that high amplitude, high modulation frequency and high modulation contrast will increase accuracy; while high offset can lead to saturation and reduce accuracy.

At high frequency, the modulation contrast can begin to attenuate due to the physical property of the silicon. This puts a practical upper limit on the modulation frequency. TOF sensors with high roll-off frequency generally can deliver higher accuracy.

The fact that the CW measurement is based on phase, which wraps around every 2Ï€, means the distance will also have an aliasing distance. The distance where aliasing occurs is called the ambiguity distance, ô€€famb, and is defined as:

$$d_{amb} = \frac{c}{2f} \quad \text{Eq. 7}$$

Since the distance wraps, ô€€famb is also the maximum measurable distance. If one wishes to extend the measurable distance, one may reduce the modulation frequency, but at the cost of reduced accuracy, as according to Equation 6.

Instead of accepting this compromise, advanced TOF systems deploy multi-frequency techniques to extend the distance without reducing the modulation frequency. Multi-frequency techniques work by adding one or more modulation frequencies to the mix. Each modulation frequency will have a different ambiguity distance, but true location is the one where the different frequencies agree. The frequency of when the two modulations agree, called the beat frequency, is usually lower, and corresponds to a much longer ambiguity distance. The dual-frequency concept is illustrated below.

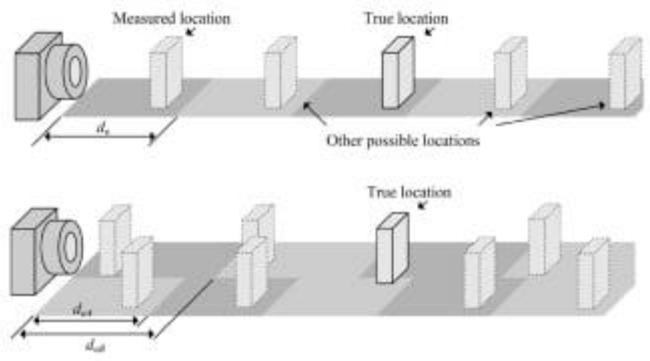


Figure 3: Extending distance using a multi-frequency technique

3. Point Cloud

In TOF sensors, distance is measured for every pixel in a 2D addressable array, resulting in a depth map. A depth map is a collection of 3D points (each point also known as a voxel). As an example, a QVGA sensor will have a depth map of 320 x 240 voxels. 2D representation of a depth map is a gray-scale image, as is illustrated by the soda cans example in Figure 4â€" the brighter the intensity, the closer the voxel. Figure 4 shows the depth map of a group of soda cans.

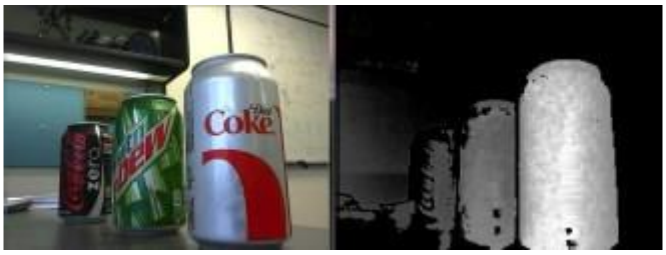


Figure 4: Depth map of soda cans.

Alternatively, a depth map can be rendered in a three-dimensional space as a collection of points, or point-cloud. The 3D points can be mathematically connected to form a mesh onto which a texture surface can be mapped. If the texture is from a real-time color image of the same subject, a life-like 3D rendering of the subject will emerge, as is illustrated by the avatar in Figure 5. One may be able to rotate the avatar to view different perspectives.

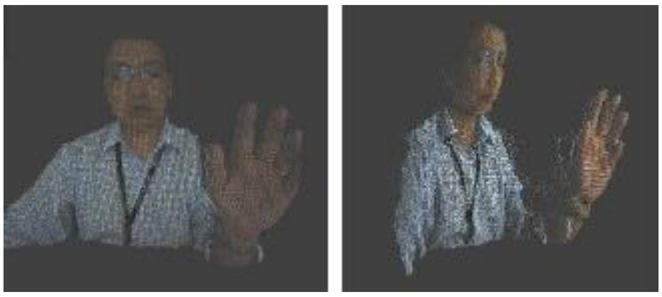


Figure 5: Avatar formed from point-cloud.

Robust 3D vision overcomes many problems of 2D vision, as the depth measurement can be used to easily separate foreground from background. This is particularly useful for scene understanding, where the first step is to segment the subject of interest (foreground) from other parts of the image (background).

Gesture recognition, for example, involves scene understanding. Using distance as a discriminator, a TOF sensor enables separation of the face, hands, and fingers from the rest of the image, so gesture recognition can be achieved with high confidence.

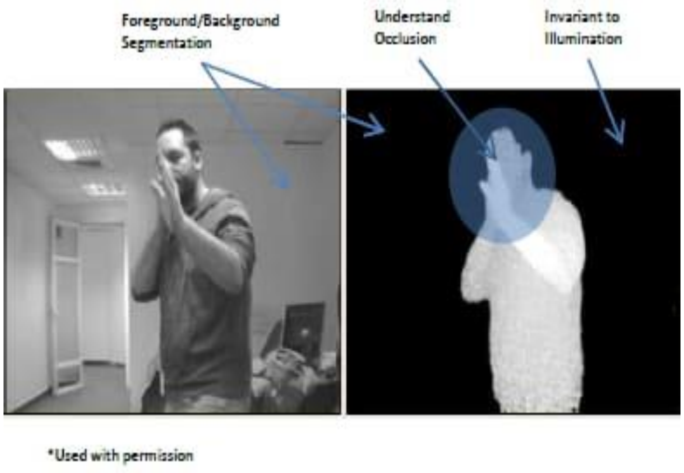


Figure 7: Advantages of 3D vision over 2D.

9. MATLAB CODE FOR IMAGE SEGMENTATION

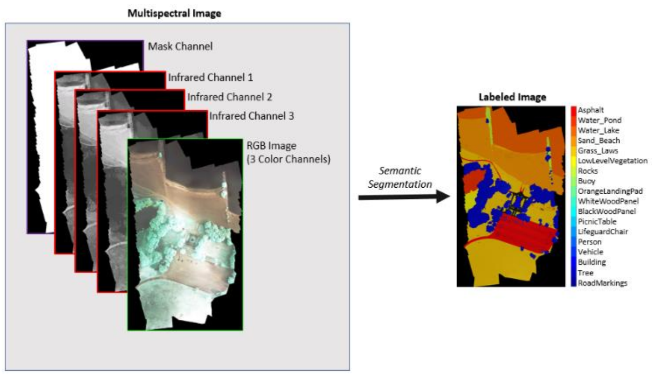
```
% This is a program for extracting objects from
an image. Written for vehicle number plate
segmentation and extraction
% input - give the image file name as input. eg
:- car3.jpg
clc;
clear all;
```

```
k=input('Enter the file name','s'); % input
image; color image
im=imread(k);
im1=rgb2gray(im);
im1=medfilt2(im1,[3 3]); %Median filtering the
image to remove noise%
BW = edge(im1,'sobel'); %finding edges
[imx,imy]=size(BW);
msk=[0 0 0 0 0;
     0 1 1 1 0;
     0 1 1 1 0;
     0 1 1 1 0;
     0 0 0 0 0;];
B=conv2(double(BW),double(msk));    %Smoothing
image to reduce the number of connected
components
L  =  bwlabel(B,8);%  Calculating  connected
components
mx=max(max(L))
% There will be mx connected components. Here U
can give a value between 1 and mx for L or in a
loop you can extract all connected components
% If you are using the attached car image, by
giving 17,18,19,22,27,28 to L you can extract
the number plate completely.
[r,c] = find(L==17);
rc = [r c];
[sx sy]=size(rc);
n1=zeros(imx,imy);
for i=1:sx
    x1=rc(i,1);
    y1=rc(i,2);
    n1(x1,y1)=255;
end % Storing the extracted image in an array
figure,imshow(im);
figure,imshow(im1);
figure,imshow(B);
figure,imshow(n1,[]);
```

10. SEMANTIC SEGMENTATION OF MULTISPECTRAL IMAGES USING DEEP LEARNING

Semantic segmentation involves labelling each pixel in an image with a class. One application of semantic segmentation is tracking deforestation, which is the change in forest cover over time. Environmental agencies track deforestation to assess and quantify the environmental and ecological health of a region.

Deep-learning-based semantic segmentation can yield a precise measurement of vegetation cover from high-resolution aerial photographs. One challenge is differentiating classes with similar visual characteristics, such as trying to classify a green pixel as grass, shrubbery, or tree. To increase classification accuracy, some data sets contain multispectral images that provide additional information about each pixel. For example, the Hamlin Beach State Park data set supplements the colour images with near-infrared channels that provide a clearer separation of the classes.



Inspect Training Data

Load the data set into the workspace.

```
load(fullfile(imageDir,'rit18_data','rit18_data.mat'));
```

Examine the structure of the data.

```
whos train_data val_data test_data
  Name          Size                 Bytes Class    Attributes

  test_data     7x12446x7654         1333663576 uint16
  train_data     7x9393x5642          741934284 uint16
  val_data      7x8833x6918          855493716 uint16
```

The multispectral image data is arranged as *numChannels*-by-*width*-by-*height* arrays. However, in MATLAB®, multichannel images are arranged as *width*-by-*height*-by-*numChannels*arrays. To reshape the data so that the channels are in the third dimension, use the helper function, switchChannelsToThirdPlane. This function is attached to the example as a supporting file.

```
train_data = switchChannelsToThirdPlane(train_data);
val_data  = switchChannelsToThirdPlane(val_data);
test_data  = switchChannelsToThirdPlane(test_data);
```

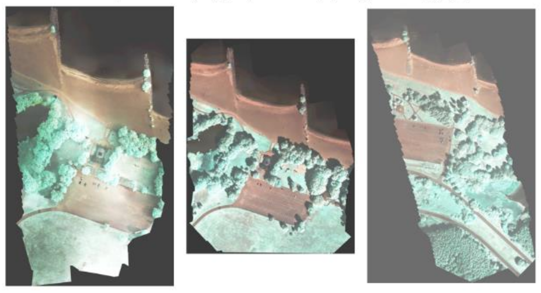
Confirm that the data has the correct structure.

```
whos train_data val_data test_data
  Name            Size                 Bytes  Class    Attributes

  test_data     12446x7654x7         1333663576  uint16
  train_data     9393x5642x7          741934284  uint16
  val_data      8833x6918x7          855493716  uint16
```

The RGB color channels are the 4th, 5th, and 6th image channels. Display the color component of the training, validation, and test images as a montage. To make the images appear brighter on the screen, equalize their histograms by using the histeq function.

```
figure
montage(...
   {histeq(train_data(:,:,4:6)), ...
   histeq(val_data(:,:,4:6)), ...
   histeq(test_data(:,:,4:6))}, ...
   'BorderSize',10,'BackgroundColor','white')
title('RGB Component of Training Image (Left), Validation Image (Center), and Test Image (Right)')
```

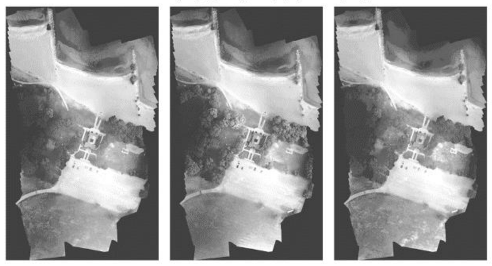
RGB Component of Training Image (Left), Validation Image (Center), and Test Image (Right)



Display the first three histogram-equalized channels of the training data as a montage. These channels correspond to the near-infrared bands and highlight different components of the image based on their heat signatures. For example, the trees near the center of the second channel image show more detail than the trees in the other two channels.

```
figure
montage(...
   {histeq(train_data(:,:,1)), ...
   histeq(train_data(:,:,2)), ...
   histeq(train_data(:,:,3))}, ...
   'BorderSize',10,'BackgroundColor','white')
title('IR Channels 1 (Left), 2, (Center), and 3 (Right) of Training Image')
```

IR Channels 1 (Left), 2, (Center), and 3 (Right) of Training Image



Channel 7 is a mask that indicates the valid segmentation region. Display the mask for the training, validation, and test images.

```
figure
montage(...
   {train_data(:,:,7), ...
   val_data(:,:,7), ...
   test_data(:,:,7)}, ...
   'BorderSize',10,'BackgroundColor','white')
title('Mask of Training Image (Left), Validation Image (Center), and Test Image (Right)')
```

Mask of Training Image (Left), Validation Image (Center), and Test Image (Right)

The labeled images contain the ground truth data for the segmentation, with each pixel assigned to one of the 18 classes. Get a list of the classes with their corresponding IDs.

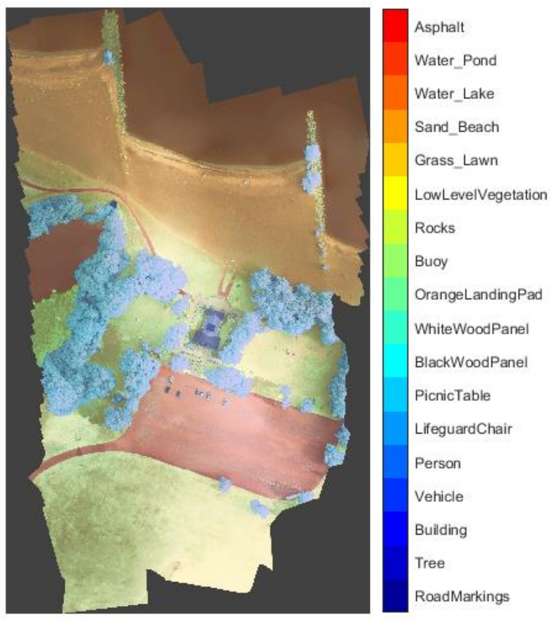
```
disp(classes)
0. Other Class/Image Border
1. Road Markings
2. Tree
3. Building
4. Vehicle (Car, Truck, or Bus)
5. Person
6. Lifeguard Chair
7. Picnic Table
8. Black Wood Panel
9. White Wood Panel
10. Orange Landing Pad
11. Water Buoy
12. Rocks
13. Other Vegetation
14. Grass
15. Sand
16. Water (Lake)
17. Water (Pond)
18. Asphalt (Parking Lot/Walkway)
```

Create a vector of class names.

```
classNames = [
"RoadMarkings","Tree","Building","Vehicle","Person", ...

"LifeguardChair","PicnicTable","BlackWoodPanel",...

"WhiteWoodPanel","OrangeLandingPad","Buoy","Rocks",...

"LowLevelVegetation","Grass_Lawn","Sand_Beach",...
      "Water_Lake","Water_Pond","Asphalt"];
```

Overlay the labels on the histogram-equalized RGB training image. Add a colorbar to the image.

```
cmap = jet(numel(classNames));
B = labeloverlay(histeq(train_data(:,:,4:6)),train_labels,'Transparency',0.8,'Colormap',cmap);

figure
title('Training Labels')
imshow(B)
N = numel(classNames);
ticks = 1/(N*2):1/N:1;
colorbar('TickLabels',cellstr(classNames),'Ticks',ticks,'TickLength',0,'TickLabelInterpreter','none');
colormap(cmap)
```



Save the training data as a MAT file and the training labels as a PNG file.

```
save('train_data.mat','train_data');
imwrite(train_labels,'train_labels.png');
```

Create Random Patch Extraction Datastore for Training

Use a random patch extraction datastore to feed the training data to the network. This datastore extracts multiple corresponding random patches from an image datastore and pixel label datastore that contain ground truth images and pixel label data. Patching is a common technique to prevent running out of memory for large images and to effectively increase the amount of available training data.

Begin by storing the training images from 'train_data.mat' in an imageDatastore. Because the MAT file format is a nonstandard image format, you must use a MAT file reader to enable reading the image data. You can use the helper MAT file reader, matReader, that extracts the first six channels from the training data and omits the last channel containing the mask. This function is attached to the example as a supporting file.

```
imds = imageDatastore('train_data.mat','FileExtensions','.mat','ReadFcn',@matReader);
```

Create a pixelLabelDatastore to store the label patches containing the 18 labeled regions.

```
pixelLabelIds = 1:18;
pxds = pixelLabelDatastore('train_labels.png',classNames,pixelLabelIds);
```

Create a randomPatchExtractionDatastore from the image datastore and the pixel label datastore. Each mini-batch contains 16 patches of size 256-by-256 pixels. One thousand mini-batches are extracted at each iteration of the epoch.

```
dsTrain = randomPatchExtractionDatastore(imds,pxds,[256,256],'PatchesPerImage',16000);
```

The random patch extraction datastore dsTrain provides mini-batches of data to the network at each iteration of the epoch. Preview the datastore to explore the data.

```
inputBatch = preview(dsTrain);
disp(inputBatch)
    InputImage      ResponsePixelLabelImage
  _______________    _____________________

[256×256×6 uint16]  [256×256 categorical]
[256×256×6 uint16]  [256×256 categorical]
[256×256×6 uint16]  [256×256 categorical]
[256×256×6 uint16]  [256×256 categorical]
[256×256×6 uint16]  [256×256 categorical]
[256×256×6 uint16]  [256×256 categorical]
[256×256×6 uint16]  [256×256 categorical]
[256×256×6 uint16]  [256×256 categorical]
```

Create U-Net Network Layers

This example uses a variation of the U-Net network. In U-Net, the initial series of convolutional layers are interspersed with max pooling layers, successively decreasing the resolution of the input image. These layers are followed by a series of convolutional layers interspersed with upsampling operators, successively increasing the resolution of the input image [2]. The name U-Net comes from the fact that the network can be drawn with a symmetric shape like the letter U.

This example modifies the U-Net to use zero-padding in the convolutions, so that the input and the output to the convolutions have the same size. Use the helper function, createUnet, to create a U-Net with a few preselected hyperparameters. This function is attached as a supporting file to the example.

```
inputTileSize = [256,256,6];
lgraph = createUnet(inputTileSize);
disp(lgraph.Layers)
 58x1 Layer array with layers:

   1  'ImageInputLayer'                Image Input
256x256x6 images with 'zerocenter' normalization
   2  'Encoder-Section-1-Conv-1'       Convolution
64 3x3x6 convolutions with stride [1  1] and padding [1  1  1  1]
   3   'Encoder-Section-1-ReLU-1'      ReLU
ReLU
   4  'Encoder-Section-1-Conv-2'       Convolution
64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
   5   'Encoder-Section-1-ReLU-2'      ReLU
ReLU
   6  'Encoder-Section-1-MaxPool'      Max Pooling
2x2 max pooling with stride [2  2] and padding [0  0  0  0]
   7  'Encoder-Section-2-Conv-1'       Convolution
128 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
   8   'Encoder-Section-2-ReLU-1'      ReLU
ReLU
```

9 'Encoder-Section-2-Conv-2' Convolution 128 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]
10 'Encoder-Section-2-ReLU-2' ReLU ReLU
11 'Encoder-Section-2-MaxPool' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0 0 0]
12 'Encoder-Section-3-Conv-1' Convolution 256 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]
13 'Encoder-Section-3-ReLU-1' ReLU ReLU
14 'Encoder-Section-3-Conv-2' Convolution 256 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
15 'Encoder-Section-3-ReLU-2' ReLU ReLU
16 'Encoder-Section-3-MaxPool' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0 0 0]
17 'Encoder-Section-4-Conv-1' Convolution 512 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
18 'Encoder-Section-4-ReLU-1' ReLU ReLU
19 'Encoder-Section-4-Conv-2' Convolution 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]
20 'Encoder-Section-4-ReLU-2' ReLU ReLU
21 'Encoder-Section-4-DropOut' Dropout 50% dropout
22 'Encoder-Section-4-MaxPool' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0 0 0]
23 'Mid-Conv-1' Convolution 1024 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]
24 'Mid-ReLU-1' ReLU ReLU
25 'Mid-Conv-2' Convolution 1024 3x3x1024 convolutions with stride [1 1] and padding [1 1 1 1]
26 'Mid-ReLU-2' ReLU ReLU
27 'Mid-DropOut' Dropout 50% dropout
28 'Decoder-Section-1-UpConv' Transposed Convolution 512 2x2x1024 transposed convolutions with stride [2 2] and cropping [0 0 0 0]
29 'Decoder-Section-1-UpReLU' ReLU ReLU
30 'Decoder-Section-1-DepthConcatenation' Depth concatenation Depth concatenation of 2 inputs
31 'Decoder-Section-1-Conv-1' Convolution 512 3x3x1024 convolutions with stride [1 1] and padding [1 1 1 1]
32 'Decoder-Section-1-ReLU-1' ReLU ReLU
33 'Decoder-Section-1-Conv-2' Convolution 512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]
34 'Decoder-Section-1-ReLU-2' ReLU ReLU
35 'Decoder-Section-2-UpConv' Transposed Convolution 256 2x2x512 transposed convolutions with stride [2 2] and cropping [0 0 0 0]
36 'Decoder-Section-2-UpReLU' ReLU ReLU
37 'Decoder-Section-2-DepthConcatenation' Depth concatenation Depth concatenation of 2 inputs
38 'Decoder-Section-2-Conv-1' Convolution 256 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]
39 'Decoder-Section-2-ReLU-1' ReLU ReLU
40 'Decoder-Section-2-Conv-2' Convolution 256 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
41 'Decoder-Section-2-ReLU-2' ReLU ReLU
42 'Decoder-Section-3-UpConv' Transposed Convolution 128 2x2x256 transposed convolutions with stride [2 2] and cropping [0 0 0 0]
43 'Decoder-Section-3-UpReLU' ReLU ReLU
44 'Decoder-Section-3-DepthConcatenation' Depth concatenation Depth concatenation of 2 inputs
45 'Decoder-Section-3-Conv-1' Convolution 128 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
46 'Decoder-Section-3-ReLU-1' ReLU ReLU
47 'Decoder-Section-3-Conv-2' Convolution 128 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]
48 'Decoder-Section-3-ReLU-2' ReLU ReLU
49 'Decoder-Section-4-UpConv' Transposed Convolution 64 2x2x128 transposed convolutions with stride [2 2] and cropping [0 0 0 0]
50 'Decoder-Section-4-UpReLU' ReLU ReLU
51 'Decoder-Section-4-DepthConcatenation' Depth concatenation Depth concatenation of 2 inputs
52 'Decoder-Section-4-Conv-1' Convolution 64 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]
53 'Decoder-Section-4-ReLU-1' ReLU ReLU
54 'Decoder-Section-4-Conv-2' Convolution 64 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]
55 'Decoder-Section-4-ReLU-2' ReLU ReLU
56 'Final-ConvolutionLayer' Convolution 18 1x1x64 convolutions with stride [1 1] and padding [0 0 0 0]
57 'Softmax-Layer' Softmax softmax
58 'Segmentation-Layer' Pixel Classification Layer Cross-entropy loss

Select Training Options

Train the network using stochastic gradient descent with momentum (SGDM) optimization. Specify the hyperparameter settings for SDGM by using the trainingOptions function.

Training a deep network is time-consuming. Accelerate the training by specifying a high learning rate. However, this can cause the gradients of the network to explode or grow uncontrollably, preventing the network from training successfully. To keep the gradients in a meaningful range, enable gradient clipping by specifying 'GradientThreshold' as 0.05, and specify 'GradientThresholdMethod' to use the L2-norm of the gradients.

```
initialLearningRate = 0.05;
maxEpochs = 150;
minibatchSize = 16;
l2reg = 0.0001;

options = trainingOptions('sgdm',...
   'InitialLearnRate',initialLearningRate, ...
   'Momentum',0.9,...
   'L2Regularization',l2reg,...
   'MaxEpochs',maxEpochs,...
   'MiniBatchSize',minibatchSize,...
   'LearnRateSchedule','piecewise',...
   'Shuffle','every-epoch',...
   'GradientThresholdMethod','l2norm',...
   'GradientThreshold',0.05, ...
   'Plots','training-progress', ...
   'VerboseFrequency',20);
```

Train the Network

After configuring the training options and the random patch extraction datastore, train the U-Net network by using the trainNetwork function. To train the network, set the doTraining parameter in the following code to true. A CUDA-capable NVIDIA™ GPU with compute capability 3.0 or higher is highly recommended for training.

If you keep the doTraining parameter in the following code as false, then the example returns a pretrained U-Net network.

Note: Training takes about 20 hours on an NVIDIA™ Titan X and can take even longer depending on your GPU hardware.

```
doTraining = false;
if doTraining
   modelDateTime = datestr(now,'dd-mmm-yyyy-HH-MM-SS');
   [net,info] = trainNetwork(dsTrain,lgraph,options);
   save(['multispectralUnet-' modelDateTime '-Epoch-' num2str(maxEpochs) '.mat'],'net','options');
else

load(fullfile(imageDir,'trainedUnet','multispectralUnet.mat'));
end
```

You can now use the U-Net to semantically segment the multispectral image.

Predict Results on Test Data

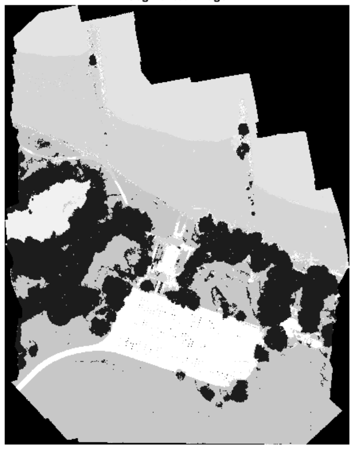
To perform the forward pass on the trained network, use the helper function, segmentImage, with the validation data set. This function is attached to the example as a supporting file. segmentImage performs segmentation on image patches using the semanticseg function.

```
predictPatchSize = [1024 1024];
segmentedImage = segmentImage(val_data,net,predictPatchSize);
```

To extract only the valid portion of the segmentation, multiply the segmented image by the mask channel of the validation data.

```
segmentedImage = uint8(val_data(:,:,7)~=0) .* segmentedImage;

figure
imshow(segmentedImage,[])
title('Segmented Image')
```

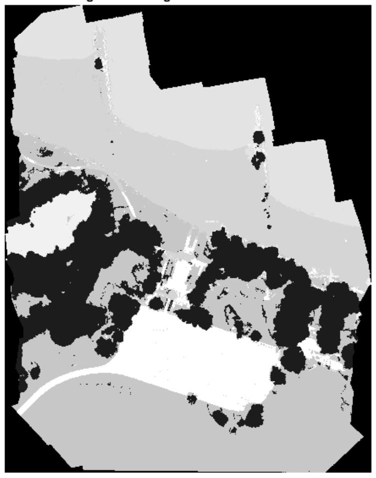
Segmented Image



The output of semantic segmentation is noisy. Perform post image processing to remove noise and stray pixels. Use the medfilt2 function to remove salt-and-pepper noise from the segmentation. Visualize the segmented image with the noise removed.

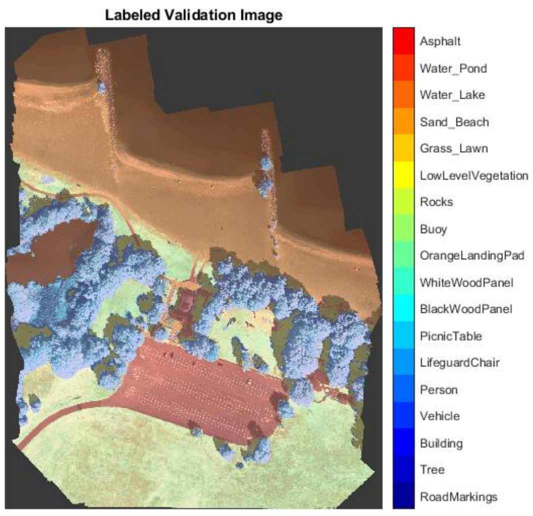
```
segmentedImage = medfilt2(segmentedImage,[7,7]);
imshow(segmentedImage,[]);
title('Segmented Image  with Noise Removed')
```

Segmented Image with Noise Removed



Overlay the segmented image on the histogram-equalized RGB validation image.

```
B = labeloverlay(histeq(val_data(:,:,4:6)),segmentedImage,'Transparency',0.8,'Colormap',cmap);

figure
imshow(B)
title('Labeled Validation Image')
colorbar('TickLabels',cellstr(classNames),'Ticks',ticks,'TickLength',0,'TickLabelInterpreter','none');
colormap(cmap)
```



Save the segmented image and ground truth labels as PNG files. These will be used to compute accuracy metrics.

```
imwrite(segmentedImage,'results.png');
imwrite(val_labels,'gtruth.png');
```

Quantify Segmentation Accuracy

Create a pixelLabelDatastore for the segmentation results and the ground truth labels.

```
pxdsResults = pixelLabelDatastore('results.png',classNames,pixelLabelIds);
pxdsTruth = pixelLabelDatastore('gtruth.png',classNames,pixelLabelIds);
```

Measure the global accuracy of the semantic segmentation by using the evaluateSemanticSegmentation function.

```
ssm = evaluateSemanticSegmentation(pxdsResults,pxdsTruth,'Metrics','global-accuracy');
Evaluating semantic segmentation results
----------------------------------------
* Selected metrics: global accuracy.
* Processing 1 images...
[==================================================] 100%
Elapsed time: 00:00:31
Estimated time remaining: 00:00:00
* Finalizing... Done.
* Data set metrics:
   GlobalAccuracy
     0.90698
```

The global accuracy score indicates that just over 90% of the pixels are classified correctly.

Calculate Extent of Vegetation Cover

The final goal of this example is to calculate the extent of vegetation cover in the multispectral image.

Find the number of pixels labeled vegetation. The label IDs 2 ("Trees"), 13 ("LowLevelVegetation"), and 14 ("Grass_Lawn") are the vegetation classes. Also find the total number of valid pixels by summing the pixels in the ROI of the mask image.

```
vegetationClassIds = uint8([2,13,14]);
vegetationPixels = ismember(segmentedImage(:),vegetationClassIds);
validPixels = (segmentedImage~=0);

numVegetationPixels = sum(vegetationPixels(:));
numValidPixels = sum(validPixels(:));
```

Calculate the percentage of vegetation cover by dividing the number of vegetation pixels by the number of valid pixels.

```
percentVegetationCover = (numVegetationPixels/numValidPixels)*100;
fprintf('The percentage of vegetation cover is %3.2f%%.',percentVegetationCover);
The percentage of vegetation cover is 51.72%.
```

Summary

The above example shows how to create and train a U-Net network for semantic segmentation of a seven-channel multispectral image. These are the steps to train the network:

- Download and reshape the training data.
- Create a randomPatchExtractionDatastore to feed training data to the network.
- Define the layers of the U-Net network.
- Specify training options.
- Train the network using the trainNetwork function.

After training the U-Net network or loading a pretrained U-Net network, the example performs semantic segmentation of the validation data and measures the segmentation accuracy.

11. APPLICATIONS

- Drones with Collision Avoidance Systems
- Indoor navigation
- Gesture recognition
- Object scanning
- Collision avoidance
- Track objects
- Surveillance of a target zone
- Count objects or people
- Fast precise distance-to-target readings
- Augmented reality / Virtual Reality

- Estimate size and shape of objects
- Enhanced 3D photography
- Logistics
- Surveillance & security
- Archaeology
- Environment projects

12. CONCLUSION & FUTURE SCOPE

This paper presents an approach which could be used for developing a Smart Multipurpose Unmanned Aerial Vehicle with 3D Depth Mapping & Spectral Imaging which could aid in carrying out Search & Rescue Missions, 3D Depth Mapping of difficult terrains, Analysis of Location and also provide audio/video aid to the people in distress. It could also be used as a surveillance system to increase the security strength especially in the area where human interference is strictly prohibited. UAV's offer advantages for many applications when comparing with their manned counter parts. They save human pilots from flying in dangerous and (or) Hazardous conditions that can be encountered not only in military applications but also in other scenarios involving operation in bad weather conditions, Hazardous/Toxic environment or near to buildings, trees, civil infrastructures, Difficult Terrains and other obstacles.

13. RESEARCH FOR FUTURE STUDY

- The future research can be carried out to implement and design SWARM technology so that a fleet of drones can be sent that communicate with each other and perform various operations.
- Performance Enhancements can also be made and could be equipped with Combat capability

14. ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mentioning of the people whose constant guidance and encouragement made it possible. We take pleasure in presenting before you, our paper, which is result of studied blend of both research and knowledge.

We (Abdus Samee, Saquib Mohiuddin, Abeer Arif & Sania Fatima) take this opportunity to express our gratitude and sincere thanks to Dr. Mohammed Abdul Nayeem, Professor & Head of Electronics and Communication Engineering Department - DCET, for providing guidance and the necessary facilities for completion of paper. He also provided us with Mentor, Dr. Syeda Gauhar Fatima, Professor ECE, DCET, who organized, coordinated, and mentored us throughout.

15. REFERENCES

I. Kemker, R., C. Salvaggio, and C. Kanan. "High-Resolution Multispectral Dataset for Semantic Segmentation." CoRR, abs/1703.01918. 2017.

II. Ronneberger, O., P. Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." CoRR, abs/1505.04597. 2015.

III. Dhriti Raj Borah et al., "A review on Quadcopter Surveillance and Control", *ADBU-Journal of Engineering Technology*, 4(1), 2016, 116-119

IV. Prof.A.V.Javir, Ketan Pawar, Santosh Dhudum, et al., "Design, Analysis and Fabrication of Quadcopter", *Journal of The International Association of Advanced Technology and Science*, vol. 16, 2015

V. Yiwen Luo, Meng Joo Er, et al., "Intelligent Control and Navigation of an Indoor Quad-copter", *IEEE*, 2014, 1700-1705.

VI. Gordon Ononiwu, Arinze Okoye, et al., "Design and Implementation of a Real Time Wireless Quadcopter for Rescue Operations", *American Journal of Engineering Research*, 5(9), 2016, 130-138.

VII. Prabhjot Singh Sandhu, "DEVELOPMENT OF ISR FOR QUADCOPTER", *International Journal of Research in Engineering and Technology*, 03(4), 2014, 181-189

A. Samba Siva, B. Prudhviraj kumar, et al., "Development of Mini Unmanned Aerial Vehicle*", IOSR Journal of*

VIII. *Mechanical and Civil Engineering,* 12(2), 2015, 16-19

IX. Vimal Raj , Sriram, Ram Mohan , Manoj Austin , "Design and fabrication of inclined arm miniature sized quadcopter UAV", *IOSR Journal of Mechanical and Civil Engineering,*13(5), 2016, 73-76

X. It Nun Thiang, Dr.LuMaw, Hla Myo Tun, "Vision-Based Object Tracking Algorithm With AR. Drone",

XI. *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH*, 5(6), 2016, 135-139

XII. Ramamoorthy Luxman, Xiao Liu, "Implementation of back-stepping integral controller for a gesture driven quad-copter with human detection and auto follow feature", *IEEE*, 2015, 134-138

XIII. Mr. B. Vinoth Kumar, S. Kalaiyarasan, et al., "Quadcopter Based Gas Detection System", *IOSR Journal of Electronics and Communication Engineering*, 11(1), 2016, 64-68

XIV. Alex G. Kendall, Nishaad N. Salvapantula, Karl A. Stol, "On-Board Object Tracking Control of a Quadcopter with

XV. Monocular Vision", *International Conference on Unmanned Aircraft Systems*, 2014, 404-411

XVI. https://www.mathworks.com/help/images/multispectral-semantic-segmentation-using-deep-learning.html

XVII. https://www.allaboutcircuits.com/industry-articles/capturing-3d-images-with-tof-camera-technology/

XVIII. https://www.mouser.in/applications/time-of-flight-robotics/