A Software Prototype for Object Detection and Recongnition for Unmanned Aerial Vehicles (UAVS) In Nigeria

Iduh Blessing Nwamaka[1]. Prof. Anigbogu Sylvanus. Okwudili[2].
[1] [2] Computer Science Department, Nnamdi Azikiwe University, Awka.

Abstract— This research work was done in order to examine the software processes involved in the creation of Unmanned Aerial Vehicles. As software capabilities advances, it becomes incumbent on software engineers to device systems that are able to respond to complex situations in ways that humans cannot. Using computer vision as source of environmental data, this work tends to show the steps involved in the software workings of UAVs. The objective of this work thus, is to devise a software that can make an unmanned aerial vehicle (UAVs) to function. At the end of the work, the new system was able to identify a target based on some given specifications and it was also able to figure out the target amongst other images surrounding it. The programming languages used includes; Python, Open Computer Vision (OpenCV), Matlab plotting Library (Matplotlib) and cMake. These programming languages were integrated to achieve this work.

Index Terms— Image detection, Object detection, Recognition, Target, Unmanned Aerial Vehicles (UAV).

1 Introduction

The spate of insecurity around the globe has pushed the capabilities and possibilities of technology in the last few decades. The use of information technology to tackle issues of insecurity especially in developing nations has become increasingly inevitable.

The activities of insurgents for instance, has greatly eaten deep into our own very essence of living in Nigeria. According to [1], these groups specializes in attacking churches, schools, large gathering, etc, sending cold fear through the nerves of hapless inhabitants of target communities. Also, the unending attacks on our oil pipelines is another issue of great concern. Pipelines located in interior villages are vandalized on daily bases by so called militants and the government and security agencies seem to lack requisite technology to stem this vice. Surveillance in these affected areas are without doubt grossly inadequate thereby eliciting satanic courage on the vandals to attack with impunity without fear of apprehension by security forces.

Stemming from the above, it has become very necessary to explore this aspect of Information Technology which is Object Detection and Recognition for Unmanned Aerial Vehicles.

Unmanned Aerial Vehicles (UAVs) are also referred to as drones, they are aircraft systems that do not have a human pilot onboard, and its flight is controlled by a remote control or by a computer [2]. They are sent on missions in very high risk areas where humans are at major risk. To survey a camp where terrorist groups are suspected to be using as hideout, for instance, the UAV can be deployed to carry out such seemingly risky detective assignments instead of using army troops with flesh and blood in such potential danger zones.

UAVs have been deployed by over 50 countries, and a lot of them build it themselves. Countries like United States of America, Japan, China, Israel, Russia, Germany, Greece, Iran, Pakistan, amongst others have recorded breakthroughs via UAVs which has become a source of solution to lots of security upheavals even till date[3]. As technology continues to make waves globally, human are becoming less involved, this means that tasks that require mechanical, labour and spontaneous decision making tasks within intelligence gathering are being replaced by software systems[3]. There are different UAVs that perform different tasks in various fields, ranging from the military to transport to oil and gas sectors and many others. The above mentioned countries and many others have achieved tremendous successes in the use of UAVs in various applicable fields.

This study thus, aims at creating a software program that can make an Unmanned Aerial Vehicle to identify a given target based on given specifications, classify the characteristics of such target as specified to isolate the needed target from amongst other images that may be in the same area.

2. LITERATURE REVIEW

[4] Proposed an Automated Detection and Recognition of Wildlife Using Thermal Cameras to contribute to the automated detection and classification of animals in thermal imaging. In their work, the methods and results were based on top-view images taken manually from a lift to motivate work towards unmanned aerial vehicle-based detection and recognition. Hot objects were detected based on a threshold dynamically adjusted to each frame. For the classification of animals, they proposed a novel thermal feature extraction algorithm. A thermal signature was calculated for each detected object, using morphological operations. The thermal signature describes heat characteristics of objects and is partly invariant to translation, rotation, scale and posture.

[3] Further proposed a prototype with the hopes of contributing to an engineering project in the field of unmanned aerial vehicles. Using computer vision as source for environmental data, this project attempts a small prototypical system for identifying objects that it discovers. The system uses OpenCV, Perl scripting, and cURL software libraries.

A HUMAN DETECTION SYSTEM IN UNCLUTTERED ENVIRONMENTS: FROM GROUND TO UAV VIEW WAS PROPOSED BY [5]. IN THEIR WORK, ANHOG DETECTOR WAS ADAPTED FOR UAV USE AND A NEW KIND OF TRAINING DATASET WAS PROPOSED IN ORDER TO INCREASE THE DETECTOR'S ANGULAR ROBUSTNESS. A MORE APPROPRIATE SET OF DETECTION WINDOWS, TOGETHER WITH A NEW DETECTION PIPELINE, WAS ALSO PROPOSED IN ORDER TO REDUCE THE SEARCH SPACE AND CONSEQUENTLY REDUCE THE COMPUTATION TIME. TESTS CONDUCTED USING THE IMPROVED DETECTOR SHOWED SIGNIFICANTLY BETTER RESULTS ON AERIAL IMAGES.

A novel hierarchical moving target detection method based on spatiotemporal saliency was also proposed by [6]. In their work, temporal saliency was used to get a coarse segmentation, and spatial saliency was extracted to obtain the object's appearance details in candidate motion regions.

.

3 MATERIALS AND METHODS

The methodology adopted for this work is the Prototyping Methodology. This decision was reached due to the nature of the system that is being designed. It important to note that when a prototype is properly achieved it can be said that the solution of system is already at sight. [4]

Prototyping being a methodology, has a collection of methods, you can organize the methods into steps, you can write them down in the order in which they should be executed or you can also teach both the steps and the order in which they should be done. Prototyping is done in a systematic way, and that way can be described, taught, scheduled, measured, compared or modified - which are the components of any methodology. In this work, the simulation method of prototyping was adopted.[5][6].

4. SYSTEM DESIGN

The design stage of the system was broken down into several processes, they include; Image Processing, Image Scanning and Sampling, Image Filtering and Edge Detection. These processes are carried out upon definition of the given image. Below are the steps for image processing;

4.1. System Overview and Deliverables

This project is a synthesis of specific software that involves artificial intelligence. The following process is involved in the image processing;

a) Load the specified image.
b) Load the IMU data related to the image.
c) Scan the image for targets, storing those that meet the specifications.
d) Classify the target into a recognized polygon; discard those that do not match.
e) Approximate the dominant colors located within the perimeter of the polygon.
f) Approximate the character located on the target.
h) Archive findings to permanent storage.
j) Repeat.

4.2 Image Processing

Upon successfully importing the image, it is now ready to process. Images provided by the camera are in a JPG format and are stored under the RGB color space. Images that are imported are placed inside OpenCV's Mat object class and have several attributes that are accessed throughout the object's lifespan. The Mat object will also undergo several alterations and operations in the sequence of evaluation. 31 Before the image is examined for targets it undergoes a smoothing filter by using the image pyramid down and pyramid up operations. Imagine the pyramid as a set of layers in which the higher the layer, the smaller the size. If the pyramid down operation is used, pixel values are lost and a transition of moving from a lower level to one above is completed. If the reverse is followed then new pixels are created with their values interpolated from neighbors. Figure 1. displays the concept of image pyramids.

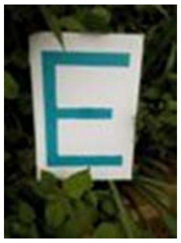


Figure 1. A sample image with no filtering applied.

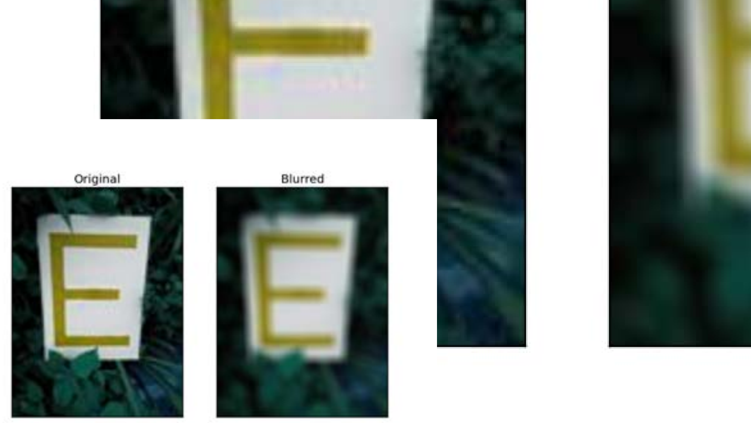


Figure 2. A succession of pyramid down and up sampling.

4.3 Image Dividing.

The next stage is for the image to be devided into colour channels. Now that the image has been smoothed to enhance continuous regions, the program will proceed with shape recognition. The next step taken towards target recognition is to evaluate regions based on color. The color space can maintain values of [0,255] for each of the RGB channels. If the image is left in its current state, there will be a total of 16,777,216 (2553) possible color combinations. This simply is too many colors to match against, and on a computationally modest processor would require a large amount of time to run. Conversely the image could have been converted to grayscale and evaluated for continuous areas based on the gray value. Although quicker, this process does not provide any information about the color relationships and to do so would require the program to refer back to the original image. To get around these obstacles the image is divided into its three color channel independent images.

Figure 3 below shows the color presence from the original source image with its channels separated in *Figure 4*.

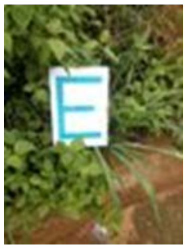


Figure 3. Sample image before being divided in to color channels.

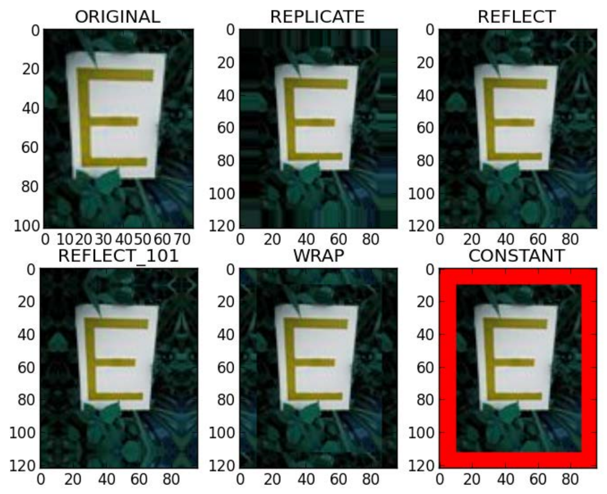


Figure 4. *different channels of the previous image*

Figure 4 above, shows how the system keeps scanning the target to actually place or ascertain if it is really what it's looking for.

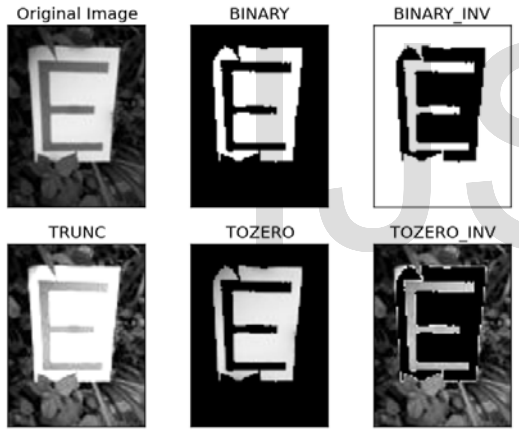


Figure 5. *System scanning image*

Figure 5 shows the original smoothed image divided into its three color channels. These are in fact grayscale but have been colorized for illustrative purposes. The program is ready to search for shapes that meet the requirements of being a target. Each channel undergoes a global threshold (all pixels are evaluated independently against some integer value) for regions of similar value and is bounded by contour edges. Contours represent edges where the contrast is strong enough to represent a break in different regions. These contours are represented as a vector collection of two dimensional points within the image. Points in the vector are arranged by their relationship with its immediate neighbours establishing a sequence of points. Contours that result from this process are shown in *Figure 6*. Before being passed to polygon approximation, each contour will be measured against the surface area, discarding those too small or too large. Also seen in *Figure 5*, there are an unnecessarily large amount of edges that have been detected. Much of the noise reports no important data about the image. *Figure 6* to the right, reveals that there is a triangle in the middle of the image after the edge filtering process.

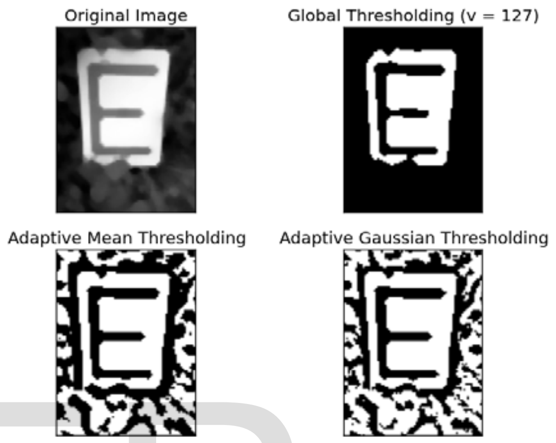

Original Image
Global Thresholding (v = 127)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

Figure 6. *The edges detected in the image with no filtering.*

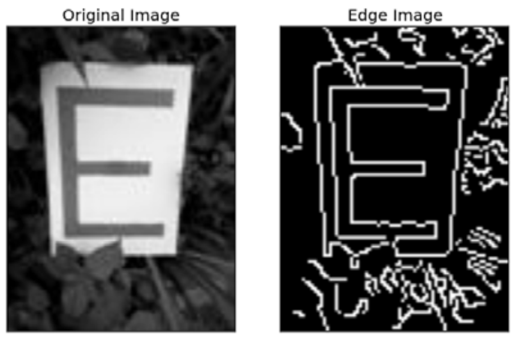


Figure 7. *images unfiltered and filtered*

With the above illustrations, you can now see how the system has been able to analyse an image, and the patterns it used to ascertain that it is actually the specified target.

d. *System flow chart.*

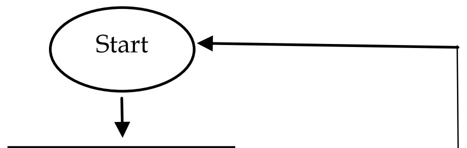
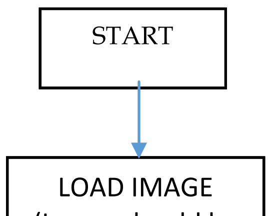


Figure 9. Overview of the Program Module

Figure 8. System Flow Chart

e. *Program Module;*

Several Algorithms were put together to achieve this work



f. *Algorithm for loading the image*

1. Start

```
2.  Import openCV
3.  Declare variables, img, k, s
4.  Import image form system and assign it to 'img'
        img ← cv2.imread('image.jpg',0)
5.  Give OpenCV the command for showing the image
        cv2.imshow('image',img)
6.  Define how much time the system will wait before displaying image
        k ← cv2.waitKey(0)
7.  If k == 27 secs
        wait for ESC key to exit
        Close all windows
    Else if k == ord('s')
        wait for 's' key to save and exit
8.  Close all windows
9.  Stop.
```

g. Algorithm for breaking the image into channels

```
1.  Start
2.  Import OpenCV
3.  Import Numeric Python
4.  Import Matplotlib and extract python Plt from it
5.  Declare variables, img1, replicate, reflect, reflect101, wrap, constant, plt, subplot
6.  Import image form system and assign it to 'img1'
        img1 ← cv2.imread('C:\opencv\projectimage.jpg')
7.  Use OpenCV to define the edges of the image
8.  Use Matplotlib to plot around the image
9.  Show different channels of the plotted image
10. Stop
```

h. Algorithm for Image Scanning

```
1.  Start
2.  Import OpenCV
3.  Import Numeric Python from matplotlib import pyplot as plt
4.  Declare Variables i, img, thresh1, thresh2, thresh3, thresh4, thresh5, titles,
5.  Import image form system and assign it to 'img'
6.  Define the areas to be scanned
7.  Stop
```

i. Algorithm for Image Filtering

```
1.  Start
2.  Import OpenCV
3.  Import Numeric Python
4.  Import Matplotlib and extract python Plt from it
5.  Declare variables, v, img, ret, th1, th2, th3, th4,th5, titles, images
6.  Import image form system and assign it to 'img1'
7.  Give OpenCV the specifications for the Filtering

8.  Give titles to the image specifications
        titles←['Original Image', 'Global Thresholding (v = 127)',
9.  Assign Values to the image
    images ← [img, th1, th2, th3]
10. Give OpenCV the specificatons for ploting
11. Display the filtered images at different stages
12. Stop.
```

j. Algorithm for Image Edge Detection

```
1.  Start
2.  Import OpenCV
3.  Import Numeric Python
4.  Import Matplotlib and extract python Plt from it

5.  Declare variables, img, edges
6.  Import image form system and assign it to 'img'
7.  Use Matplotlib to plot the edges of the image
8.  Display the image
9.  Stop
```

k. Programming Languages used for this system

Due to the complexity of this system, several programming languages where put together to achieve this much. They are;

- Open Computer Vision (OpenCV)
- Python Programming language
- Numpy (Numeric Python) – this handled the numerical aspect of the system program
- Matplotlib (Mathematical Plotting Library) – to handle the graphical aspect of the program.
- cMake Library .

[7][8]

5 Result and Discussion

The result obtained satisfied the slated objectives, which was to simulate a software program that will enable an Unmanned Aerial Vehicle to be able to identify a given target based on given specifications, classify the characteristics of a target as specified and figure out a specified target amongst other images that may be in the same area. The system requires some software and hardware specifications to be able to perform. These are stated below;

5.1 Hardware requirement:

- Hard disk with at least 2GB free space and above
- A1074x768 Screen resolution monitor
- A processor with a least 500MHZ of speed
- A RAM of at least 1Gigabyte
- A mouse and a key board
-

5.2 Software Requirements:

- Windows 7 Operating System or Linus Operating System
- OpenCV image processing library
- MatPlotlib interface must be installed
- The system must be NumPy enables

5.3 Installation

To install a complete Unmanned Aerial Vehicle, there should be a hardware based machine that will be mounted on the target site ie where the object that is to be identified is located. This hardware based device will be the housing for the software system that has just been developed.

The step involved includes:

1. Set the hardware in place
2. Get a microcontroller board
3. Install the software in the micro chip
4. Insert it into the hardware device
5. Power up the switch

This system is expected to be manned, handled and operated by engineering experts; therefore, a special professional training will need to be carried out for the intending operators.

The Parallel Changeover Procedure procedure is recommended for this system since it involves changeover from a manual system of object detection to the use of an Unmanned Aerial Vehicle. In this case, the old and new systems are used concurrently. This will help the users to get used to the new system before a total changeover is done.

The system is highly structured and as such every modification or alterations to its data flow and control needs the assistance of a programmer. Any machine error incurred during execution of the system also calls for special assistance

6 CONCLUSION

Via this research work, we were able to document the steps and software involvement in the execution of a functional UAV system. With the outcome of this work and if implemented in an Unmanned Aerial Vehicle system, the system is expected to function properly.

7. References

Lobel, Mark (2012): "Deadly attack on Nigeria's Bayero university".
BBC. Retrieved 5 May 2012.

2. Tice, Brian P. (1991). "Unmanned Aerial Vehicles –The Force Multiplier of the 1990s". *Airpower Journal*. Retrieved 6 June

3. Malinoski, Phillip. *Object detection and recognition for UAV*. Diss. California State University, Northridge, 2012.

4. Horgen, John (March, 2013) "Unmanned Flight National Geographic.
Retrieved 20-2-2013.

5. *Automated Detection and Recognition of Wildlife Using Thermal Cameras.*
Peter Christiansen , Kim Arild Steen, et al. Department of Engineering, Aarhus University, Finlandsgade 22, Aarhus, Denmark

6. Blondel, Paul, et al. "Human detection in uncluttered environments: From ground to UAV view." *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*. IEEE, 2014.

7. Shen, Hao, et al. "Moving object detection in aerial video based on spatiotemporal saliency." *Chinese Journal of Aeronautics* 26.5 (2013): 1211-1217.

8. Anigbogu, S. O. (1999): *Introduction to Computer Science*
and programming Development *languages.* Christon International Company Ltd. Awka.

9. Osuagwu, O. E. (2008): *Software Engineering, a Pragmatic and Technical Perspective.* Oliv-

erson Industrial Publishing House (OIPH) Ltd, Owerri, Imo State. Pp 546 – 548.

10. Anigbogu, S. O. (2009): *Fundamental Principles of Artificial Intelligence and Expert Systems.* Rex Charles and Patrick Ltd, Nimo, Anambra State.

11. Swaroop C. H. (2013): *A Byte of Python* May, 2013. Pp.1-112. Retrieved from http://files.swaroopch.com/byte_of_python.pdf

12. Parameswaran, Shibin, et al. "*Marine object detection in UAV full-motion video*." *SPIE Defense+ Security*. International Society for Optics and Photonics, 2014.