Open UAS Repository

Jacob Longar David Yang
Department of Computer Science,
College of Engineering and Mining
University of North Dakota
Email: {jacob.longar, david.yang}@und.edu

ABSTRACT
The Open UAS Repository Project is made up of the UAV Repository and the Xcel UAV Projects and focuses on giving users the ability to upload image mosaics taken from *Unmanned Aerial Vehicle* (UAVs) to a site for further sharing and modifying. Users can currently organize, share, draw shapes, and filter uploaded images into different channels. The UAV Repository Project focuses on giving users extra options for viewing UAV imageries through different filters for easier viewing and recognition of image contents. The Xcel UAV Project deals with creating a *Convolutional Neural Network* (CNN). The CNN is created and trained to find downed power lines through UAV imagery. The CNN will need to use some of the filtering options created by the UAV Repository Project in order to recognize downed power lines more efficiently.

I. Introduction

The Open UAS Repository Project is made up of two individual projects: the UAV Repository Project and Xcel UAV Project. The two separate projects were combined when Dr. Travis Desell created the Open UAS Repository website to safely store and transport large image files, up to hundreds of Gigabytes, instead of using up memory space on PCs and thumb drives.

As mentioned above, the Open UAS Repository Project is made up of two individual projects. The UAV Repository Project, which was initially called "the Wildlife@home Project" and worked by Jacob Longar, focused primarily on improving some of the functionalities of the website such as adding a color palette and an option to provide additional channels to view the mosaics. As for the Xcel UAV Project, which is worked by David Yang, it focuses on establishing a link between a Convolutional Neural Network (CNN) with the website's database in order to extract data to build training images to identify certain objects within the uploaded mosaics. Much of the report will be split between the work, results, and implementation methods done by the UAV Repository and the Xcel UAV Projects.

A. UAV Repository Project Overview

The UAV Repository project aims to create a user friendly site that allows for the uploading and management of large UAV / mosaic imagery. A large portion of the project has been created and updated by Dr. Desell. Jacob's version of this site is a copy of Dr. Desell's site with extra features the site didn't originally contain.

B. Xcel UAV Project Overview

The Xcel UAV project aimed to identify the power lines of utility poles within a UAV mosaic image, constructed from multiple UAV imageries. A CNN was used to identify the power lines. In general, a CNN identifies an object by searching through a fixed size block of pixels from one end of the mosaic to the other end. The selected block is filtered with three different kinds of layers: convolutional layers, pooling layers, and fully connected layers. For each selected block of pixel, the CNN performs the same operations of intermixing convolutional and pooling layers, creating a 2D activation map each. Near the end, the FC layers are finally used to process the multiple 2D activation maps. Two CNNs have been created by Dr. Desell: the First CNN will search for the power lines if start and end values of the power lines are given as part of the inputs, and the Second CNN searches for the tops of the utility poles.

The current milestone is to create a way for the CNNs to retrieve data from the database associated with the Open UAS Repository and use the retrieved data to create training data. Originally, the CNNs read given inputs from binary files with training images already extracted from larger mosaics and cut down to the acceptable sizes. If the CNNs can retrieve data directly from the database and use that data in place of the binary files and extracted images, then the CNNs can work directly on the original mosaics. For example, the process for creating training images is manual and tedious, also taking up additional memory. Thus, without having to manually extract training images of set sizes and store them in another directory can save time and memory if this milestone is successfully implemented.

C. Scientific Merit

UAVs are being used more frequently and becoming more important in other fields of study. Exploration in the many possible usages for UAVs continues to grow, but the storage and transfer of UAV imageries will become major issues for many individuals and institutions that rely heavily on UAV imageries for scientific research and analysis. The Open UAS Repository has an easy-to-use interface for marking up and labeling mosaics, and it also provides a more efficient

method of safely storing and transferring UAV imageries. The additional functionalities from the UAV Repository Project can increase the role of UAV imageries for other purposes, and the integration of a CNN with the Open UAS Repository database from the Xcel UAV Project can help researchers in other fields of study to search, locate, or count for specific objects within the UAV imagery. The Open UAS Repository can help popularize institutions around the United States and other countries to create their own repositories for a variety of research topics.

D. Overall Result

This is nothing worth mentioning yet.

II. Related Work

File sharing or the storage of large data inside databases is nothing new and has already been implemented in the past. It is how old solutions are being adapted and used to solve emerging problems in the present that actually matters. Below are research studies and experimental projects conducted in the past that the UAV Repository and Xcel UAV Projects have used as references for ideas or suggestions.

A. UAV Repository

A project led by John A. Tsanakas, aimed to detect a number of specific animals in an area using infrared and near-infrared imagery. The study focused on counting the number of hippopotami, an endangered animal, through usage of UAVs, thermal cameras and software [1]. Individual thermal/infrared images were taken in Garamba National Park (Democratic Republic of Congo) and used as data for an algorithm that would detect the hippopotomi. The following process was used in order to detect these animals: First, an open source software known as *Quantam Geographical Information System Software* (*QGIS*) was used to enter relevant data to each image tested by the algorithm. The data consisted of the height of the UAV at the time the image was taken as well as the polygon counts of the resulting image. The polygon count was used to find differences in heat signatures as a single number value. This process was essentially used to ”digitize” each animal seen by the UAV's thermal camera, creating the image as shown in Figure 1. After the images were formed, they were further processed by adding points to the resulting images to indicate local maxima of each stack of polygons. This allowed each animal to be identified and easily counted. The study concluded that, while UAV imagery was very useful in detecting hippopotomi, a large amount of data and technology was required in order to create their results.

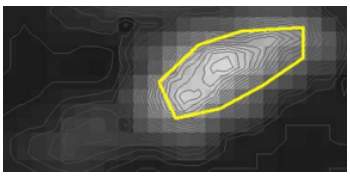

Fig. 1. mapping relevant polygons

A study conducted by Gomez-Candon, A. I. De Castro, and F. Lopez-Granados focused on how much geological accuracy could be obtained by taking images from a UAV at differentiating heights [2]. In order to do this, the same number of Ground Control Points (GCPs) had to be obtained per height difference in order to maintain accuracy while creating the image mosaic. Images of the same landscape (grass / wheat landscapes), were taken by a UAV at height levels of 30, 60, and 100 m. The project concluded with a result in which there wasn't a significant differences in heights between 30 and 100 m above ground level as long as the same number of GCPs are used for each mosaic.

B. Xcel UAV

In 2010, a team, Krizhevsky et al., created a CNN for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2010 competition. Their CNN needed to classify 1.2 million images into 1000 different classes [3]. In response, their CNN used a total of 8 layers and ran on two different GPUs to handle the large number of images as well as shorten their CNN's training time. The team scored very high for the competition with a very small marginal error, compared to the other algorithms used in the ILSVRC by other teams. Unlike the Krizhevsky et al., the two CNNs created by Dr. Desell are searching for multiple instances of a single object within a mosaic, whether this object at this location is a set of power lines or utility poles or not.

Another deep learning algorithm was the VGG 16, introduced at the ILSVRC 2014 by the VGG Team, consisting of Simonyan and Zisserman [4]. It needed to classify 14 million images into 1000 categories. The VGG 16 achieved a very high accuracy, but it was very hard to train and consumed a lot of resources like memory. Despite that, its major contribution was determining that the depth of a CNN, meaning the number of convol, pooling, and FC layers, does impact the overall performances.

There was a project that tried to incorporate GPS coordinates with their CNN. The project was developed by a team from Stanford University and Facebook AI Research, Tang et al. [5]. The groups goal was to predict the location class labels if given an image with an associated GPS coordinate. They first created a rectangular grid representing the US and have indicators for each image to show what grid cell the image would be in based on its GPS coordinates. They also incorporated a huge amount of other information like jobs, wealth, and education to create a background to increase the CNNs knowledge. For example, if the CNN was looking at an image of a rich neighborhood, then the image should likely contain these kinds of objects for the CNN to pick up. In the end, the team found out that location-specific classes gave the best performances since they werent mixed up with other locations of similar background.

III. Methodology

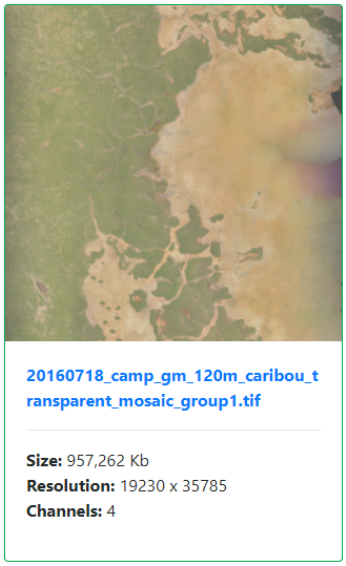

Fig. 2. Example of HTML template: Mosaici Card

The Open UAS Repository site is built from a variety of languages and tools. The following is an overview of how the website was built from the ground up. The HTML was built using Twitter's bootstrap for style and functionality. The *Mustache* template system was also used in the PHP code in order to generate portions of HTML code. For example, all of the mosaic cards were generated using mustache and HTML templates. An example of this can be seen in Figure 2.

PHP was also used to perform back-end functions like querying the database and tiling mosaics as they were uploading. Both ImageMagick and ImageSlicer were called in the PHP code in order to assist in splitting the mosaic into slices. The mosaics are split into pieces so the tool we used for viewing *OpenSeaDragon* can quickly zoom in and out of the image, even if the size of the original image uploaded was very large.

Javascript was then used to add functionality to the HTML created earlier. Javascript was also used to perform complex functions in creating labels and drawing points on the mosaics. Most of the code dealing with drawing points on the mosaics comes from *index.js*. This file also handles any functions that need to be executed on the main page including; sharing mosaics, organizing mosaics, uploading mosaics, uploading rgb files, and performing any additional actions not handled directly by the HTML or PHP code.

The PHP code also handled querying the database for all information required by the user. The tables were used for a multitude of things including but not limited to; storing mosaic files and all of their metadata, storing all data points for rgb files, storing user information and which mosaics they have access to, and storing points and lines drawn on mosaics.

IV. Implementation

A. UAV Repository

The site built by Jacob Longar, being a modified version of the site built by Dr. Desell, expanding on the code and ideas of the original site. Jacob Longar created code that adds more functionality to the original website. He used the following languages to add to the original code: PHP, Javascript, AJAX, and HTML. A couple tools were also used to build the original website, but were also used in the creation of the modified website. The Mustache engine was used in order to aid in the creation of HTML templates. OpenSeaDragon was used for the viewing of images created by the UAVs, and the filtering of those images so users could more easily see specific details of each image. Twitter's Bootstrap was used to create the look and feel of the website. Google authentication was used for users to log in and create separate accounts with the UAV repository website. MagicSlicer was used to split each original image into multiple smaller images in order for OpenSeaDragon to display UAV images. Finally, ImageMagick was used for additional filtering options on top of OpenSeaDragon.

Jacob's portion of the project focused on creating additional filtering options both by using .rgb files downloaded from the NCL color table gallery website[6], and creating a pallet of colors for the user to choose from. This was done by using Javascript, PHP, HTML, and AJAX. Like many portions of the Open UAS site, a HTML template was used to create an upload button. A javascript action was called to perform the following AJAX script when the button was clicked:

```
$.ajax({
  url : 'rgb_script.php',
   dataType: 'text',
   cache: false,
   type : 'POST',
   data : formData,
   processData: false, // tell jQuery not to
       process the data
   contentType: false, // tell jQuery not to
       set contentType
   success : function(data) {
     console.log(data);
      alert(data);
   }
});
```

This AJAX script then passes the values of the .rgb file uploaded from the user into a PHP file called `rgb_script.php`. This file does the work of extracting data from the .rgb file and inserting the data into the database table created for the .rgb script data. The top section of the following code snippet shows the data input, the bottom portion found beneath the "..." represents the addition of this data to the database table.

```
if
    (is_uploaded_file($_FILES['file']['tmp_name']))
   {
     echo "File ". $_FILES['file']['name'] ."
         uploaded successfully.\n";
}
else {
  echo "The following file was not uploaded
      correctly";
```

```
  echo "filename '".
      $_FILES['userfile']['tmp_name'] . "'.";

...

$query = "INSERT INTO rgb_files SET owner_id
    = '2', name = '$jsFileLoc'";
error_log($query);
query_our_db($query);
```

B. Xcel UAV

The Xcel UAV Project uses C, C++, MySQL queries, cmake, and the already-built directory "exact," which contains the majority of the codes to implement the CNNs; and "exact" stands for "the Evolutionary Exploration of Augmenting Convolutional Topologies." The programs in the "exact" directory are written in a mixture of C and C++. Thus, any programming or modification from David switches back and forth between C and C++ when applicable in order to be consistent. MySQL queries have been added to enable "exact" to connect to the Open UAS Repository database. So far, MySQL query test runs in retrieving specific data from multiple data tables have shown to be successful, consistent, and fast. Lastly, cmake is used to combine and compile all the required programs necessary to run an executed version.

Amongst the many data tables stored in the Open UAS Repository database, only two data tables will be used frequently for data retrieval. They are the "Mosaics" table, which contains data on the uploaded mosaics such as filename, number of channels, etc.; and the "Lines" table, which contains the starting and ending points of the power lines. Remember, the First CNN, responsible for searching for power lines, requires the starting and ending points of the power lines in order to validate those locations. The "Points" table, containing the coordinates of the utility poles, may also be included for the Second CNN, which is responsible for searching for the tops of utility poles. Using the retrieved coordinates from the Points table may resolve the Second CNN's confusion with a utility pole's shadow. Once the required data are retrieved, they are placed inside vectors of vectors, and a function will open up a connection to a shared "Xcel Energy" folder, containing all of the mosaics for the projects, to find the mosaics that match the filenames retrieved from the Mosaics table.

To create the training images for the CNNs, the vector with data from the Lines table will be used as inputs. However, the CNNs only process mosaics of a set size, and the original mosaics are too large to be processed. As a result, for each mosaic being processed, a smaller function steps in to temporarily extract the fixed size to be processed. Once the CNNs finish processing, the temporary extract is discarded, and the function moves onto the next pixel to start the next temporary extraction to be processed. This operation will repeat almost continuously until the end of the mosaic is reached. Then, the CNNs will move onto the next mosaic in line and continue until all the mosaics have been processed. An example is shown in the image below, where we can see a larger image of N-unit by M-unit is being cut down into many small pieces of n-unit by m-unit. Just as explained, the smaller function is extracting a smaller image size from the larger image.

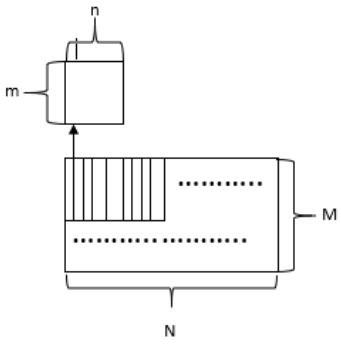

Fig. 3. Repetitive Image Extraction Function

This repetition of operations will become the training data for the CNNs. Since each image extracted by the smaller function exists temporarily, there is no need to consume memory to store the those smaller images in another directory, which can amount of thousands of images.

V. Results

This will contain our individual project results.

VI. Future Work

This will contain any possibilities of future work for us or others down the line.

VII. Conclusion

THE END.

REFERENCES

[1] Lhoest, S., et al. *How many hippos (HOMHIP): Algorithm for automatic counts of animals with infra-red thermal imagery from UAV*. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Copernicus GmbH, 19 Aug. 2015/2016, https://www.int-arch-photogramm-remote-sens-spatial- inf-sci.net/XL-3-W3/355/2015/isprsarchives-XL-3-W3-355-2015.pdf

[2] Gomez-Candon, A. I. De Castro, F. Lopez-Granados *Assessing the accuracy of mosaics from unmanned aerial vehicle (UAV) imagery for precision agriculture purposes in wheat*. SpringerLink, Springer US, 8 Nov. 2013, link.springer.com/article/10.1007/s11119-013-9335-4.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. Advances in Neural Information Processing Systems 25, Neural Information Processing Systems Foundation, Inc., 2012.

[4] Saxena, Prakarsh. *VGG 16 - An Advanced Approach Towards Accurate Large Scale Image Recognition*. TechLeer Media Pvt Ltd, Sept. 6, 2017, 11:14 AM. https://www.techleer.com/articles/305-vgg-16-an-advanced-approach-towards-accurate-large-scale-image-recognition/

[5] Kevin Tang, Manohar Paluri, Li Fei-Fei, Rob Fergus, and Lubomir Bourdev. *Improving Image Classification with Location Context*. Collaboration: Stanford University and Facebook AI Research. http://vision.stanford.edu/pdf/tang2015iccv.pdf

[6] http://www.ncl.ucar.edu/Document/Graphics/color_table_gallery.shtml