Disaster Impact Information Retrieval Using Deep Learning Object Detection in Crowdsourced Drone Footage

Yalong Pi, Nipun D. Nath, Amir H. Behzadan
Texas A&M University, USA
abezhadan@tamu.edu

Abstract. Collecting and sharing timely and reliable post-disaster data is of utmost importance in disaster mitigation. Current methods of aerial search for post-disaster reconnaissance rely on human involvement, making them expensive, subjective, and slow. We propose to equip ordinary drones with convolutional neural networks (CNN) for fast object detection and viewpoint projection to assist in evacuation, wayfinding, resource allocation, and damage assessment. In this research, an in-house hurricane video dataset and the YOLO object detection algorithm are used. Testing the trained CNN model on unseen drone footage yields an overall accuracy of 74.48%. Next, GPS-free geometrical projection is used to autonomously transform the pixel coordinates of detected objects from the drone’s perspective view to the world coordinates in an orthogonal map. When tested on classes undamaged roof and car, a high accuracy (~5% error) is achieved, demonstrating the robustness of this method for real-time mapping of disaster footage.

1. Introduction

Natural disasters impact societies, economies, and the built infrastructure. According to U.N. Office for Disaster Risk Reduction (2019), between 2004 and 2014, 1.7 billion people were affected by natural disasters globally, resulting in 700,000 deaths and $1.4 trillion in damages. Disaster management (a process that involves preparedness, mitigation, response, and recovery) requires timely access to accurate information describing disaster impact (Guha-Sapir and Lechat, 1986). Aerial reconnaissance is commonly used for locating victims and assessing damage, particularly in large-scale events such as hurricanes, tsunamis, and wildfires. While traditionally, helicopters and low-altitude flying aircrafts are deployed for data collection, the rapidly growing drone technology provides a more flexible, scalable, safer, and lower-cost alternative (Adams and Friedland, 2011). It is estimated that by the year 2022, there will be more than 2.4 million UAVs in the U.S. (Federal Aviation Administration, 2018), which can lead to opportunities for crowdsourced data collection and exchange in disaster events. In addition to collecting large and diverse disaster data, engaging people and communities in sharing volunteered geographic information (VGI) (Haworth and Bruce, 2015) can also lead to more transparency and trust in technology. The current practice of processing disaster footage, however, requires intensive manual data curation (data cleaning, formatting, analyzing, synthesizing, and delivering). Considering the lack of skilled personnel, and limited support and computing resources in the disaster’s aftermath, this can hinder the full utilization of data by users, e.g., response teams, aid agencies, volunteer groups, and the general public.

This research aims to investigate whether aerial footage captured by drone-mounted red green blue (RGB) cameras can be processed using deep learning (DL) methods to automatically extract critical disaster-related information. Particularly, a convolutional neural network (CNN) algorithm, namely you-only-look-once (YOLO v2) (Redmon and Farhadi, 2017), is employed to detect and localize key ground objects in drone footage, at fast speed, and with high accuracy. In computer vision, the task of identifying the class labels and pixel positions of objects in an image is termed object detection. The pixel coordinate of each detected object (herein referred to as target of interest or ToI) is then transformed into real-world positions to create geocoded maps of disaster impact. The designed methodology is validated using a video dataset.
2. Literature Review

Previous research has mined disaster-related VGI. For example, Craglia et al. (2012) used Facebook and Twitter text data to map disaster impact. Kim and Hastak (2018) used social media data to map the damage after the 2016 Louisiana flood. Faxi et al. (2017) utilized machine learning to extract infrastructure damage from tweets in Fort McMurray during a wildfire. Drones have been also adopted for ground inspection and information extraction. For example, Radovic et al. (2017) used YOLO (Redmon and Farhadi, 2017) to identify airplanes, buses, and cars on the ground. Han et al. (2012) integrated region-based CNN (R-CNN) and kernelized correlation filter (KCF) (Henriques et al., 2014) to track humans on the ground. Baker et al. (2016) implemented a decentralized Monte Carlo tree search algorithm to find survivors in a simulated drone path planning scenario. They conducted tests on data from the 2010 Haiti earthquake and reported consistent performance gains of up to 18% over a discretized algorithm in the number of located survivors. However, there is a dearth of work in the area of large-scale, multi-class disaster object detection and mapping from drone footage.

Object detection has evolved rapidly in recent years along with advancement of graphic processing unit (GPU), which in part led Girshick et al. (2014) to design R-CNN and Fast R-CNN (Girshick, 2015) that replaced traditional sliding box methods with region of interest (RoI) to improve processing speed. Later, Girshick (2015) proposed Faster R-CNN with region proposal network (RPN) that achieved 73.2% mean average precision (mAP) on VOC dataset (Everingham et al., 2015). Redmon et al. (2016) introduced YOLO which takes input image grids and outputs classification and position at once. Later, YOLO v2 (Redmon and Farhadi, 2017) was introduced with a better performance. Also, single shot detector (SSD) algorithm, presented by Liu et al. (2016), predefines anchor boxes and feature maps, and achieves 76.8% mAP on VOC dataset (Everingham et al., 2015). RetinaNet, proposed by Lin et al. (2017b), uses a focal loss function that puts more weight on rare training samples and yields 37.8% average precision (AP) on COCO dataset (Lin et al., 2014). As listed in Table 1, there is a tradeoff between detection accuracy and speed. For example, YOLO v2 can achieve a speed of 40 frames per second (FPS) which is faster than typical video frame rate (i.e., 30 PFS). In comparison, RetinaNet processes images at 5.81 FPS, but achieves 16.2% higher accuracy.

Table 1: Speed and accuracy comparison among CNN models (Lin et al., 2017b).

<table>
<thead>
<tr>
<th>CNN architecture</th>
<th>COCO-AP (%)</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOLOv2 (Redmon and Farhadi, 2017)</td>
<td>21.6</td>
<td>40.00</td>
</tr>
<tr>
<td>SSD321 (Liu et al., 2016)</td>
<td>28.0</td>
<td>16.39</td>
</tr>
<tr>
<td>R-FCN (Dai et al., 2016)</td>
<td>29.9</td>
<td>11.76</td>
</tr>
<tr>
<td>DSSD513 (Liu et al., 2016)</td>
<td>33.2</td>
<td>13.70</td>
</tr>
<tr>
<td>FPN FRCN (Lin et al., 2017a)</td>
<td>36.2</td>
<td>6.41</td>
</tr>
<tr>
<td>RetinaNet-101-800 (Lin et al., 2017b)</td>
<td>37.8</td>
<td>5.81</td>
</tr>
</tbody>
</table>

3. Methodology

3.1 Problem Statement

The 2017 U.S. hurricane season caused a staggering $125 billion in damages (Benfield, 2018) from hurricanes Harvey, Maria, Irma, and other named storms. With sea temperatures on the
rise, it is expected that more water-related disasters will strike coastal communities around the world every year (Trenberth et al., 2018). In this research, crowdsourced drone videos (from YouTube) capturing the aftermath of hurricanes are annotated with multiple class labels (i.e., flooded area, building roofs, car, debris, vegetation) to enable the detection of ToIs, listed in Table 2. These class labels are chosen for their value to disaster response. For instance, mapping flooded areas helps first responders in wayfinding, search and rescue, and evacuation planning. Over time, these fine-grained flood maps will be instrumental for flood map assessment by showing how floodwaters move and which downstream neighborhoods are more prone to flood damage. In this paper, two experiments are conducted to test the performance of trained CNN models and mapping technique.

<table>
<thead>
<tr>
<th>ToI</th>
<th>Potential application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flooded area</td>
<td>Rescue planning, resource deployment, wayfinding, storm surge mapping, aid delivery, flood plan improvement, public education</td>
</tr>
<tr>
<td>Undamaged roof</td>
<td>Damage information map, insurance claims, mapping reference points</td>
</tr>
<tr>
<td>Damaged roof</td>
<td>Rescue, damage information map, insurance claims, construction repair, debris removal</td>
</tr>
<tr>
<td>Car</td>
<td>Rescue, insurance claims</td>
</tr>
<tr>
<td>Debris</td>
<td>Clean-up, damage information map, construction repair, rescue</td>
</tr>
<tr>
<td>Vegetation</td>
<td>Clean-up, public education</td>
</tr>
</tbody>
</table>

### 3.2 Model Architecture, Training, and Deployment

For ToI detection from drone footage, a CNN architecture trained on an in-house experimental dataset is used. Given speed and accuracy tradeoff (Table 1), and considering that the ultimate goal of this work is to equip drone with onboard capability for real-time image processing and situational awareness, YOLO v2 (Redmon and Farhadi, 2017) is selected as an ideal CNN candidate. Similar to other CNN models, YOLO v2 takes the RGB values of an image (in form of a matrix) and outputs the class labels of ToIs and their pixel coordinates. As shown in Figure 1, there are 23 linked layers in this architecture including convolutional and max-pooling layers. Each layer computes its input and feeds into the next layer. A max-pooling layer selects the maximum value from boxes that scan the entire input with a stride, and then combines the maximum values to form the output. As a result, max-pooling downsizes the input while preserving the representative features. On the other hand, convolutional layer applies a kernel i.e., matrix with weights ($w_1, w_2, ..., w_n$), on the input to extract features (e.g., color and shape). With multiple kernels, a convolutional layer outputs a denser layer. Altogether, YOLO v2 takes an image divided into a 13×13 grid (each grid predicts 5 anchor boxes), and outputs a multidimensional matrix that describes the predictions. For each prediction, the output contains X- and Y- pixel coordinates, width, height, confidence (likelihood that the box contains a ToI), and probability for each class. The key to achieving accurate predictions is the weights of the kernels in convolutional layers, i.e., models with optimum weights yielding precise predictions.

The process of obtaining the optimum weights is referred to as model training (LeCun et al., 1998). During training, randomly initialized weights are updated by feeding the model with the input (images) and annotated output (class labels and pixel coordinates). Weights that lead to wrong predictions are penalized while those leading to correct predictions are rewarded. Eventually, optimized weights are used in the trained model. The trained model is then tested on unseen images and predictions (class labels and pixel coordinates) are recorded.
As shown in Figure 2, in this work, the CNN model is trained using transfer learning, i.e., pre-trained on COCO dataset followed by retraining on Volan2018, an in-house dataset of annotated hurricane footage. Transfer learning takes advantage of pre-trained weight on datasets such as COCO and VOC, which are later updated using an in-domain dataset (Oquab et al., 2014), thus leading to better predictions. The trained model is tested on unseen hurricane footage to retrieve disaster impact information.

### 3.3 Projection from the Drone’s Perspective View to an Orthogonal Map

The fully trained CNN model can detect ToI classes and coordinates of pixel boxes in hurricane footages. However, detected bounding boxes cannot be readily used in mapping applications that utilize grid systems such as the universal transverse Mercator (UTM) coordinate system or the United States national grid (USNG). Since these mapping systems are widely used in disaster response and coordination, projecting the output of CNN detection from the perspective view into an orthogonal grid system is desired. This mathematical transformation can be done knowing the pixel positions (in drone’s local view) of four reference points (with any three not being collinear) and their corresponding real-world positions. For pixel coordinates, with the uppermost left pixel of the image serving as the origin, rows parallel to the X-axis, and columns parallel to the Y-axis, the pixel coordinates of the centroid of the detected bounding box is determined. The real-world position of the object, on the other hand, refers to the location in the Cartesian grid system (global coordinates on Earth). Using these conventions, the goal of
projection is to transform detected pixel coordinates of any ToI with class and size information onto the grid system that is independent of video properties such as camera position, viewpoint, or zoom factor in any particular frame. Figure 3 shows an example in which \((x_1, y_1), (x_2, y_2), (x_3, y_3), \) and \((x_4, y_4)\) are four reference points in drone view, with their corresponding real-world positions marked as \((x'_1, y'_1), (x'_2, y'_2), (x'_3, y'_3), \) and \((x'_4, y'_4)\) in an orthogonal map.

**Figure 3: Perspective to Orthogonal Projection of drone’s Viewpoint**

From these reference points, transformation matrix \(M = B.A^{-1}\) is calculated using Equations 1 through 4. Next, for any new point in the drone’s perspective view, denoted by pixel coordinates \((x_5, y_5)\), the corresponding real-world position \((x'_5, y'_5)\) can be obtained using Equation 5, in which \((x''_5, y''_5, w)\) is the homogenous coordinates of the ToI, i.e., \(w = 0\) means that the point is at infinite distance from the camera. For reference point selection, different types of visible ground objects (e.g., landmarks, buildings, parking lots, road intersections) can be used as long as their real-world positions are known or can be extracted.

\[
\begin{bmatrix}
a_1 \\
a_2 \\
a_3 \\
\end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_4 \\ y_4 \\ 1 \end{bmatrix}
\]  
\(1\)

\[
A = \begin{bmatrix} a_1.x_1 & a_2.x_2 & a_3.x_3 \\ a_1.y_1 & a_2.y_2 & a_3.y_3 \\ a_1 & a_2 & a_3 \end{bmatrix}
\]  
\(2\)

\[
\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \end{bmatrix} = \begin{bmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x'_4 \\ y'_4 \\ 1 \end{bmatrix}
\]  
\(3\)

\[
B = \begin{bmatrix} b_1.x'_1 & b_2.x'_2 & b_3.x'_3 \\ b_1.y'_1 & b_2.y'_2 & b_3.y'_3 \\ b_1 & b_2 & b_3 \end{bmatrix}
\]  
\(4\)

\[
\begin{bmatrix} x''_5 \\ y''_5 \\ w \end{bmatrix} = M.\begin{bmatrix} x_5 \\ y_5 \\ 1 \end{bmatrix} \Rightarrow x'_5 = \frac{x''_5}{w}, \ y'_5 = \frac{y''_5}{w}
\]  
\(5\)
4. Experiment and Results

4.1 Data Collection and Description

The dataset used in this research, Volan2018, is created using web-mined videos from YouTube, and contains eight videos from four different hurricanes striking 8 different U.S. coastal regions during the 2018-2019 hurricane season (including hurricanes Harvey, Maria, Irma, and Michael), captured by drones and helicopters. These videos are extracted frame by frame, and ToIs in each frame are annotated with the classes and pixel coordinates using DarkLabel (2017). In this paper, only three of these videos (Volan1, 2, and 3) are used since they are captured by drone camera. All three videos are captured from the aftermath of hurricane Harvey in 2017, in and around Houston, Texas. The duration of these videos is 84, 72, and 1,333 seconds, for Volan1, 2, and 3 respectively. The total number of instances per ToI for each video is summarized in Table 3. Annotation is done frame by frame by drawing one bounding box covering each ToI (e.g., one car or one damaged roof). For example, the video frame presented in Figure 4 contains 4 instances of undamaged roof, and 1 instance of damaged roof, 4 instances of debris, and 3 instances of flooded area.

Table 3: Statistics of Volan2018 videos used in this study

<table>
<thead>
<tr>
<th>TOI class/Video number</th>
<th>Volan1</th>
<th>Volan2</th>
<th>Volan3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flooded area</td>
<td>1,015</td>
<td>1,572</td>
<td>38,480</td>
</tr>
<tr>
<td>Undamaged roof</td>
<td>1,814</td>
<td>1,174</td>
<td>37,661</td>
</tr>
<tr>
<td>Damaged roof</td>
<td>1,457</td>
<td>871</td>
<td>296</td>
</tr>
<tr>
<td>Car</td>
<td>1,046</td>
<td>612</td>
<td>24,710</td>
</tr>
<tr>
<td>Debris</td>
<td>2,678</td>
<td>1,653</td>
<td>0</td>
</tr>
<tr>
<td>Vegetation</td>
<td>123</td>
<td>0</td>
<td>38,976</td>
</tr>
</tbody>
</table>

![Figure 4: Annotation Example from Volan2018 Dataset](image)

4.2 CNN Model Training and Testing

All frames in Volan1, 2 and 3 videos are combined and then split into three portions: training (60%; 34,791 frames), validation (20%; 8,258 frames), and testing (20%; 8,264 frames). At the start of training, the first 22 layers of the model are frozen, and only the last layer is trained (for 25 epochs at learning rate $10^{-3}$). After unfreezing the first 22 layers, all layers are trained simultaneously (at learning rate $10^{-4}$), and if loss does not drop after 3 epochs, the learning rate is reduced by half, and the process iterates. Training is terminated if the validation loss does not
decrease in 10 consecutive epochs. In the example shown in Figure 5(a), several instances of vegetation, flooded area, and undamaged roof are detected, but no debris or damaged roofs are found. Each detection box is labeled with the predicted class and a confidence index. The trained CNN model is further tested to measure detection performance for each class, as well as the overall mAP. As shown in Figure 5(b), classes debris and flooded area have the highest precision of 90.27% and 86.16%, respectively, while class car achieves the lowest precision of only 45.67%. The overall mAP (for detecting all ToIs) is 74.48%.

Figure 5: (a) Detection Example from Volan2018 Dataset; (b) Model Performance for All Classes

4.3 Projection Error Measurement and Results

Two example classes are used to demonstrate the technique for projecting detected ToIs on a 2D orthogonal map without reliance on drone’s position information, as described in Section 3.3. Four building roofs are marked as reference points with their real-world positions obtained from Google Earth, and all other detected building roofs and two cars are projected from drone’s perspective view to a 2D map following the UTM coordinate system. The input drone video (Volan3) is from Houston, TX, and covers zone 15R of the UTM system. Two example frames of this video in which detected undamaged roofs (including reference points), flooded areas, and cars are marked is shown in Figure 6. For example, for frame #12210 shown in Figure 6(a), the pixel positions of all detections and the real-world positions of reference points 1-4 are used to project all other detections in the real-world coordinate system.

Figure 6: Pixel Coordinates (Box Centroids) of Reference Points and Detected ToIs in Two Frames
In Figure 7(a), these four points are marked with numbered circles, and all other detected undamaged roofs are marked with rectangles numbered 5-14. Real-world positions of these undamaged roofs serve as ground truth to measure the projection error. Figure 7(b) shows the frame projection on UTM system. For each instance of undamaged roof, the Euclidean distance between the ground truth and projected positions is defined as error. The frame-level projection error is calculated by Equation 7, in which $n$ is the total number of detected ToIs in that frame.

$$frame\ error = \frac{1}{n} \cdot \sum_{i=1}^{n} error_i$$  \hspace{1cm} (7)

![Real-World Positions of ToIs on Google Earth (left), and Projection Errors (right)](image)

Projecting undamaged roofs in a 1-second long segment between frames #12210 and #12240 (drone moving to the left in 30 consecutive frames) of Volan3 video yields a frame error of 7.18 meters. Considering the size of the visible area by the drone in the frame ($173.95 \times 130.53$ m²), this projection error translates to approximately 5%, indicating the high accuracy of the designed technique for transforming perspective drone views to orthogonal maps. As shown in Figure 6(b), two car instances are additionally detected in frame #12374 of Volan3 video, which are also projected along with building roofs, as shown in Figures 7(c) and 7(d). In this projection, detected roofs 1, 2, 4, and 5 are selected as reference points, yielding a 7.79-meter (or 5%) frame error considering both classes. It must be noted that the change of viewpoint (as a result of drone moving in the scene) may cause any or all of reference points (i.e., building roofs) in one frame to fall out of sight, in which case, new reference points are needed to have an uninterrupted projection. For instance, comparing Figures 7(a) and 7(c), reference point 3 is
replaced with a new reference point 5, using an ad-hoc reference point tracking and selection. A description of this process, however, is outside the scope of this paper.

5. Conclusion and Discussion

This research demonstrated the capability of using CNN models (trained on disaster footage) and viewpoint transformation to detect ToIs in drone-captured perspective views and project them onto orthogonal maps. The need for this operation was justified through describing practical examples of how timely and reliable collection and delivery of disaster impact information could add value to disaster management. Experiments were conducted to demonstrate the process of disaster-related ToI detection and projection, and measure performance. Results indicated that the model trained on an in-house dataset, Volan2018, could successfully detect several ToI classes (flooded areas, damaged and undamaged roofs, cars, debris, vegetation) from drone-mounted RGB cameras with an mAP of 74.48. Furthermore, as a proof-of-concept experiment for viewpoint transformation based on the real-world positions of four reference points on the ground was conducted on two classes, and an average error of ~5% was achieved, demonstrating the robustness of the method in processing drone videos for geocoded mapping of disaster footage with limited reliance on GPS information. Beyond the immediate application domain described in this paper, the developed GPS-free projection technique can be extended to other applications that require simultaneous localization and mapping (SLAM) using existing (natural or manmade) landmarks and reference points. Examples include construction robotics, traffic management, agriculture and forestry, and marine research. Future research will also enable the ad-hoc selection of projection reference points with less reliance on prior knowledge. Another direction of future work will be to perform ToI detection and mapping beyond hurricane footage, by training and testing CNN models on other types of natural disasters such as earthquakes and wildfires.

Acknowledgments

Model training was performed on Texas A&M University’s High Performance Research Computing (HPRC) clusters. Any opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not represent those of the HPRC.

References


