Analysis of Image Thresholding Algorithms for Automated Machine Learning Training Data Generation

Tristan Creek and Barry E. Mullins
Air Force Institute of Technology, Wright-Patterson AFB, OH, USA
Tristan.Creek@afit.edu
Barry.Mullins@afit.edu

Abstract: Secured compounds often safeguard physical layout details of both internal and external facilities, but these details are at risk due to the growing inclusion of Light Detection and Ranging (LiDAR) sensors in consumer off-the-shelf (COTS) technology such as cell phones. The ability to record detailed distance data with cell phones facilitates the production of high-quality three-dimensional scans in a discrete manner which directly threatens the security of private compounds. Therefore, it behooves the organizations in charge of private compounds to detect LiDAR activity. Many security cameras already detect LiDAR sources as generic light sources in specific conditions, but further analysis must identify these light sources as LiDAR sources in order to alert an organization of a potential security incident. Testing confirms the feasibility of identifying some LiDAR sources based on the color and intensity of light shined directly into a camera sensor, but this analysis proves inadequate for cell phone LiDAR. However, the unique intensity and pattern characteristics of cell phone LiDAR reflected off a surface can potentially be identified by an object identification machine learning model. In order to train a model to identify a LiDAR object, we must first produce a training dataset containing marked and labelled LiDAR objects. To do this, we apply an image thresholding algorithm to isolate the LiDAR object in an image to calculate its bounding box. The image thresholding algorithm directly affects the bounding box accuracy, so we test two different algorithms and find that Otsu's image thresholding algorithm performs best, resulting in 99.5% accurate bounding boxes.

Keywords: image thresholding, machine learning, LiDAR, CMOS

1. Introduction

The use of Light Detection and Ranging (LiDAR) devices spans across decades of history in various implementations for the military, private sector, and even civilian hobbyists (McManamon, Kamerman and Huffaker, 2010; Molebny, Kamerman and Steinvall, 2010). Although the basic principle of LiDAR relies on simply emitting infrared light at a surface or object and capturing what reflects back, the long-term development of LiDAR has produced countless LiDAR enabled technologies that facilitate a range of functions such as personnel detection through foliage (Tether, 2004), ecological measurement (Eitel *et al.*, 2016), self-driving cars (Lin *et al.*, 2020), and digital recreation of 3-dimensional (3D) objects (Raj *et al.*, 2020). However, the technological accomplishments enabled by LiDAR devices often come with a price tag beyond typical consumer budgets due to the specialized nature of LiDAR hardware components and supporting hardware/software (Queralta *et al.*, 2019). Fortunately for consumers, recent developments now provide inexpensive LiDAR hardware to consumers in a familiar portable package: cell phones. Unfortunately for organizations conscious of their security posture, this also now provides the layman with quick, inexpensive access to devices capable of nefarious activities like stealing intellectual property (Noble, 2020) and identifying burglary targets (Sturgeon, 2021).

Many collections of buildings, land, and external or internal facilities (hereby referred to as "compounds") often wish to maintain some degree of security. The actions taken to secure a compound vary, but the primary goal is to maintain some level of privacy whether that mean only allowing certain people access to the compound or simply shrouding internal facilities with external walls to prevent intelligence gathering. Despite best efforts to maintain privacy, no fool-proof method exists to prevent exploitation by potential threats due to the possibility of insider threats, espionage, etc. (Queralta *et al.*, 2019). Therefore, compound defenders must opt to implement the best possible measures to detect threats as early as possible when preventing them is not an option.

The newfound access to LiDAR-enabled cell phones increases the difficulty of a compound's defenders' ability to detect threats as early as possible. Cell phone LiDAR now enables a person to discreetly and openly carry around cell phone, as is common for people to do, and glean information about a compound from the LiDAR

sensor (Hetherington, 2021). Whether that be the width and length of the only access road to a compound or a detailed 3D external scan of a sensitive building, the security posture of a compound suffers from the intelligence that may be quickly obtained with a commercial off-the-shelf (COTS) device. Due to the fact that LiDAR emissions lie in the infrared spectrum invisible to the naked eye, defenders may worry of the requirement for costly, specialized hardware to detect foreign LiDAR emissions. Fortunately, inexpensive COTS devices also exist that aid in the detection of LiDAR.

Despite lying above the spectrum of light visible to the naked eye, some COTS security cameras capture LiDAR light emissions (Akopyan, 2016). Fortunately for defenders, this capability exists in many new security cameras as well as various security cameras manufactured in the past couple of decades, enabling defenders in some cases to leverage cameras already installed on their compounds. These cameras fail to capture the majority of unique LiDAR characteristics, such as light frequency, that differentiate it from visible light, but they do capture the high intensity and unique pattern of light produced by 3D cell phone LiDAR. In the absence of other unique characteristics, we focus on the intensity and pattern alone to identify cell phone LiDAR activity in security camera footage.

Object detection methods identify if certain objects exist within a picture according to a model trained on images with said objects accurately marked and labelled (Aker and Kalkan, 2017). In order to produce a model that accurately identifies 3D cell phone LiDAR in an image, we must first train the model on images that contain accurately marked and labelled 3D cell phone LiDAR emissions. No public collection of such data or any framework for producing such data exists to our knowledge, so this research develops a framework to collect security camera footage of 3D cell phone LiDAR emissions and produce individual frames with each LiDAR object marked and labelled for use in training an object detection machine learning model. Image thresholding algorithms allow us to isolate the LiDAR object automatically, so we investigate which algorithm produces the most accurate bounding boxes.

2. Background and related research

The rear LiDAR module on cell phones commonly emits a 2-dimensional (2D) array of beams (Figure 1) for use in applications such as 3-dimensional (3D) scanning and distance measuring (Raj *et al.*, 2020; Gallagher, 2021). This characteristic exists in multiple LiDAR-enabled cell phones, but this research uses only the rear LiDAR sensor on the iPhone 12 Pro Max (hereby referred to generically as “cell phone”). However, the research can be applied to other cell phones if their rear LiDAR sensor emits a 2D array similar to the iPhone 12 Pro Max.

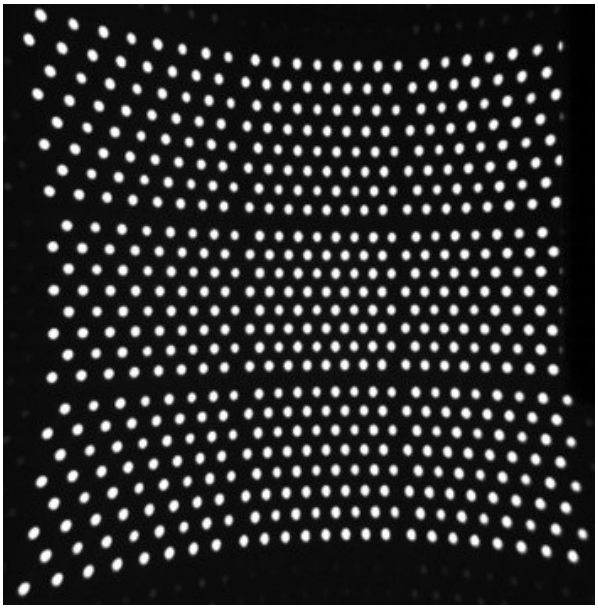

Figure 1: 2D array of LiDAR light points emitted by iPhone 12 Pro Max (Y, 2021)

Most cameras that use a complementary metal–oxide–semiconductor (CMOS) sensor can record a portion of light above the visible spectrum in the near-infrared (NIR) spectrum between 800 nanometers (nm) and 1000 nm (Vollmer, Möllmann and Shaw, 2015; Akopyan, 2016). However, physical NIR filters normally block the CMOS

sensor to prevent NIR light from distorting the image (Vollmer, Möllmann and Shaw, 2015). Cameras utilizing a CMOS sensor, such as the Lorex LNB8921BW-C used in this research, contain a Night Mode which physically moves the NIR filter out of the way of the sensor which enables the capture of NIR light (Lorex, 2021). Many LiDAR sources emit light in the NIR spectrum, enabling cameras with unfiltered CMOS sensors to capture the light (Li *et al.*, 2021).

LiDAR technologies obtain information by reflecting light off a surface and capturing the light that returns, meaning the light only contains meaningful data directly after emission from the LiDAR source and after its first reflection off a surface (Raj *et al.*, 2020). Therefore, cameras must capture LiDAR emissions either directly after they leave the LiDAR source ("direct") or after they reflect off a single surface ("indirect") to identify light characteristics indicative of LiDAR. Four key characteristics of LiDAR emissions help differentiate them from visible light: pulse frequency, color, intensity, and pattern (Marcoe, 2007; Raj *et al.*, 2020). The pulse frequency, color, and intensity differentiate LiDAR light from visible light in direct captures whereas the intensity and pattern differentiate between the two in indirect captures. The described capabilities of CMOS cameras and LiDAR sources indicate that both methods should be possible, but testing indicates that we can only reasonably differentiate between cell phone LiDAR emissions and visible light in indirect light captures.

With the NIR filter blocking NIR light from the CMOS sensor, our Lorex camera captures LiDAR from the Garmin LiDAR Lite 3HP as an intense white dot with purple edges that is roughly the width of a human thumb (1 inch) at a distance of 4 feet (see Figure 2). To differentiate the Garmin LiDAR from a light source in the visible spectrum, we can analyze the intensity and color components of the dot. Furthermore, we can develop a Fourier Transform model of the LiDAR emissions based on the known frequency and pulse width of the Garmin LiDAR source to compare to the Fourier Transform of the dot in the image. However, we do not know the exact frequency or pulse width of the cell phone LiDAR, and testing shows that cell phone LiDAR produces a much smaller dot at a maximum visible distance of 8 inches (see Figure 3). Therefore, analyzing the pulse frequency, intensity, and color of the cell phone LiDAR emitted directly into the camera sensor is not feasible.

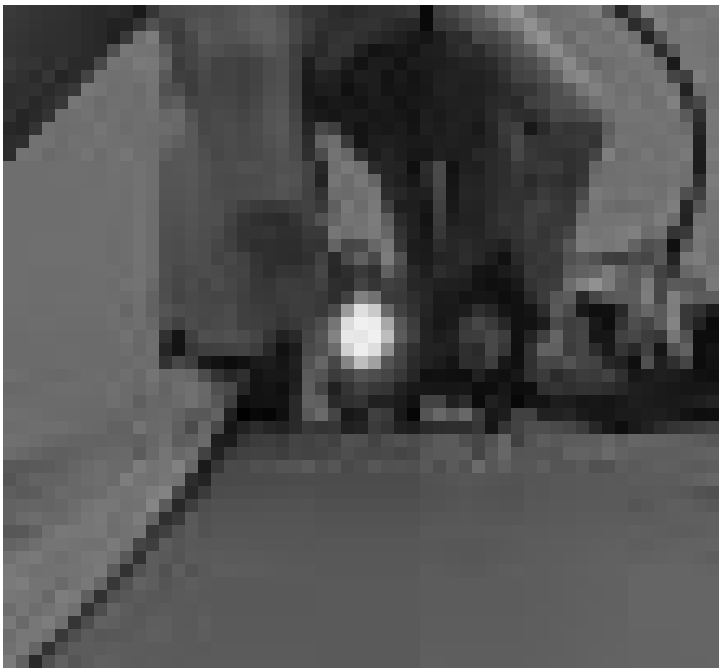

Figure 2: Garmin LiDAR captured by Lorex camera with Night Mode disabled at 4 feet

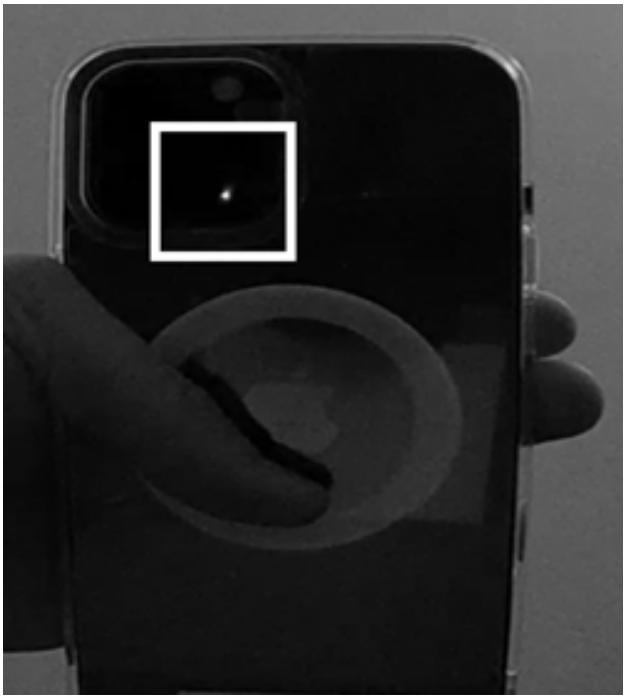

Figure 3: Cell phone LiDAR captured by Lorex camera with Night Mode disabled at 8 inches

Indirect capture provides two benefits: wider angle of collection and pattern recognition. Only a fraction of the LiDAR light reflected off a surface returns to the LiDAR sensor while the rest scatters into the environment (Raj *et al.*, 2020). This allows a secondary sensor (the camera) to capture the reflected light from a more diverse range of angles than is required in direct capture. This also allows the camera to capture all the light points in the 2D matrix emitted by the cell phone in order to identify the unique pattern (see Figure 4). Although the camera captures many of the LiDAR light points while in the presence of other light sources, isolating the light points in an image proves difficult due to the low intensity of the reflected LiDAR. Capturing the indirect LiDAR in a dark room increases the contrast of the light points, making it much easier to isolate them from the background surface. Once isolated, a machine learning model can learn the attributes of the cell phone LiDAR to identify the LiDAR in other images.

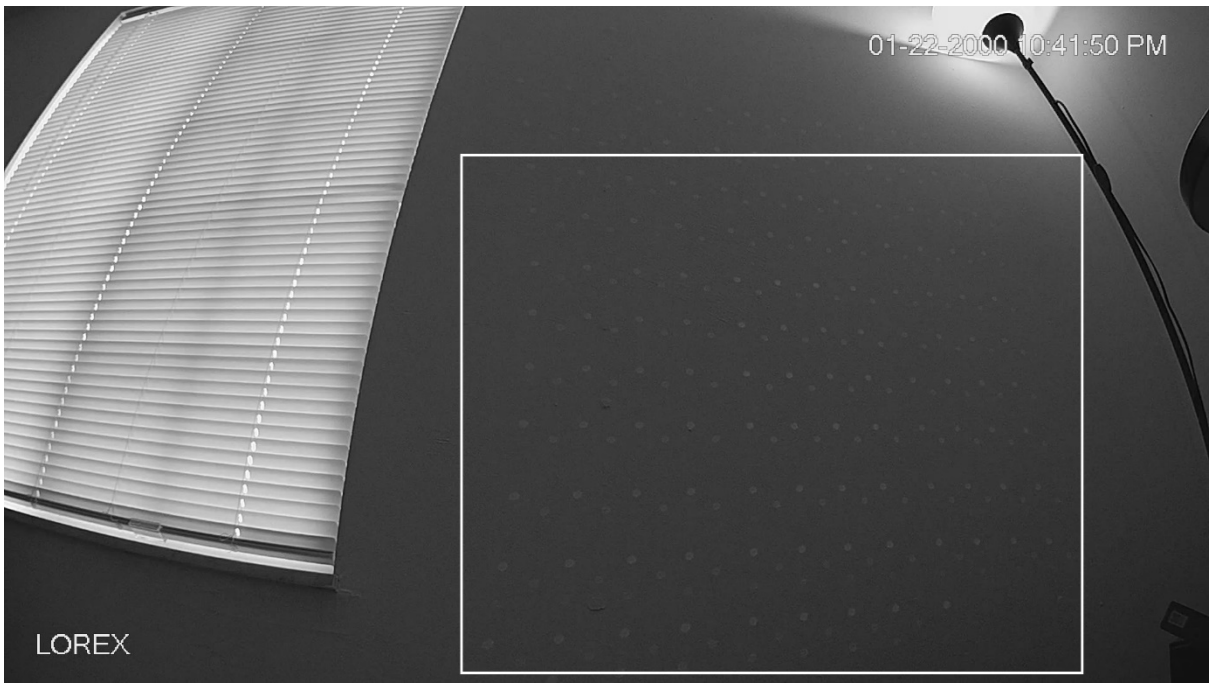

Figure 4: Indirect cell phone LiDAR emitted against a wall at 4 feet

Much research exists on both training and employing machine learning models for purposes such human body posture analysis (Xie and Guo, 2019), common household objects identification (Noman, Stankovic and Tawfik, 2019), license plate detection on moving cars (Peker, 2019), and custom generated traffic sign detection (Kilic and Aydin, 2020). These projects all utilize the TensorFlow machine learning framework which we use in this research (TensorFlow, 2018). Other popular machine learning frameworks such as Caffe, Torch, and Theano exist, but they lack the relative GPU support, ease of use, and customization of TensorFlow (Phadnis, Mishra and Bendale, 2018). TensorFlow provides highly accessible functions thanks to its implementation in Python, a powerful high-level programming language which we use exclusively for this research (Python, 2021). The high customization of TensorFlow allows us to produce a model from a wide range of machine learning algorithms. The exact TensorFlow algorithm used falls outside of the scope of this research, but studies show that TensorFlow provides many efficient algorithms for object detection which makes it a good candidate towards which we develop training data (Phadnis, Mishra and Bendale, 2018; Kilic and Aydin, 2020).

To identify custom objects in an image, we must first train a TensorFlow machine learning model on a dataset containing said objects. Production of a training dataset employs the following generic steps: data collection and cleaning, object marking and labelling, and dataset structure generation (Kilic and Aydin, 2020). We follow this methodology but break it down into further specific steps in Section 3 to generate the training dataset.

Data collection and cleaning consists of collecting the images that contain the objects the machine learning model will learn to detect. “Cleaning” regards ensuring all images possess the same characteristics such as resolution, height and width, color mode, etc (Kilic and Aydin, 2020).

Object marking and labelling, typically the most labor-intensive part, draws a bounding box around each relevant object in an image and labels it with its object name, or class (Kilic and Aydin, 2020). A bounding box, more aptly called a minimum bounding box, describes the smallest rectangle that surrounds the entire desired object. Each bounding box contains a label of the object. A training dataset must contain accurately marked and labelled data, called “truth data”, to ensure the machine learning model learns the relevant information about the object it should detect (Kilic and Aydin, 2020). Due to the large workload required to manually mark and label many images, much research exists on methods to automatically mark and label data (Feng, Jun and Cheng, 2010; Kim, Hong and Han, 2018; Pelkmann, Tharwat and Schenck, 2020; Roh, Heo and Whang, 2021). In the absence of automatic methods, tools such as LabelImg allow us to mark and label each image by hand (tzutalin, 2021).

Calculating the bounding boxes relies on the concept of image thresholding to detect the edges of the light points. Image thresholding algorithms segment images based on detected object edges using various analysis techniques. Two popular algorithms perform such analysis efficiently: Otsu’s algorithm and Yen’s algorithm. Otsu’s image thresholding algorithm minimizes the intra-class variance of pixel color values in a grayscale image to dynamically calculate a threshold value (Otsu, 1979). Yen’s algorithm defines a cost function which it minimizes to calculate a threshold value (Yen, Chang and Chang, 1995). When applying the threshold value as a binary filter on a grayscale image, all pixels with a color value below the threshold turn black (grayscale value 0), while all those equal to or above the threshold turn white (grayscale value 255). Only the high intensity pixels remain, allowing for the trivial calculation of the bounding box from the minimum and maximum X and Y coordinates of all non-black pixels.

The accuracy of a calculated bounding box regards its proximity to the true bounding of the object. Many machine learning applications gauge accuracy with a metric called Intersection over Union (IoU) which compares the intersecting area of two bounding boxes to the total area covered by both boxes. A perfect calculated bounding box has an IoU of 1 (completely intersecting) while a completely inaccurate calculated bounding box has an IoU of 0 (not intersecting at all) (Rezatofighi *et al.*, 2019).

3. Project design

We developed a framework to collect and parse video footage that consists of three distinct parts: the cell phone, the security camera, and the parsing software.

As shown in Figure 5, we place the cell phone and security camera in a dark room approximately 5 feet deep and 4 feet wide. Both the cell phone and camera stand at the same height 4.5 feet off the ground pointed directly horizontal at the back wall. The camera points from the left wall approximately 3 feet from the back wall, while

the phone points from the front wall approximately 5 feet from the back wall. The field of view of both devices intersect, enabling the camera to record the indirect LiDAR emissions from the cell phone.

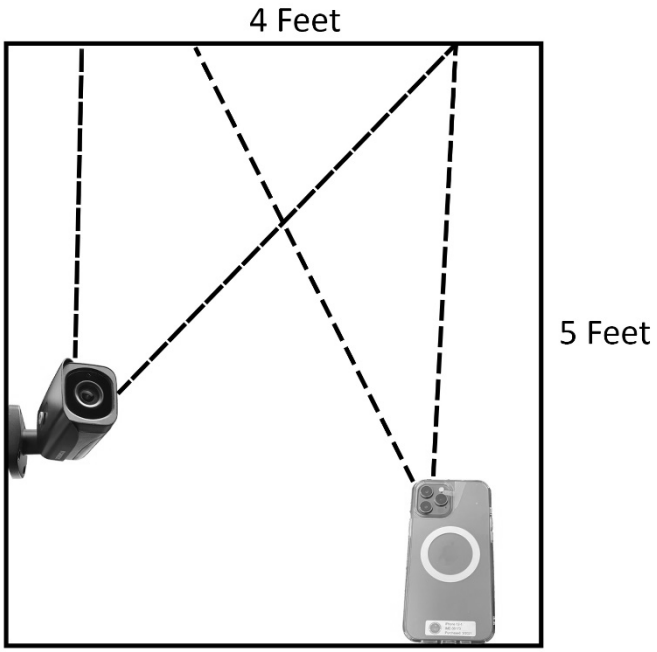

Figure 5: Cell phone and camera set up in dark room

3.1 iPhone 12 pro max

We use the iPhone 12 Pro Max to emit LiDAR light from the rear LiDAR sensor at a wall which reflects off for collection by the security camera. Apple provides a sample project that enables distance data collection with the rear LiDAR module (Apple, 2020). We slightly modify this project for ease of use, but the project as provided by Apple sufficiently emits the 2D LiDAR matrix (Figure 1) for collection by the security camera. We use the software XCode 12 on a 2020 M1 MacBook Pro to deploy the project as an app for the cell phone (Apple, 2021). Once deployed to the phone, we simply open the app and point it at the wall to constantly emit LiDAR light.

3.2 CMOS security camera

We use the Lorex LNB8921-C IP security camera to record video footage of the indirect LiDAR emissions. Preliminary testing shows successful collection of indirect LiDAR emission from the cell phone when we enable Night Mode on the camera. Various shutter speeds accurately record the LiDAR emissions, but testing shows that a shutter speed of 1/30 performs best at collecting video footage to process for training a future machine learning model. We record the video footage with a resolution of 704x480 at a frame rate of 15 frames per second (FPS). All of the aforementioned settings must be set manually in the camera's local web interface before recording footage with the Python script detailed in the following section.

3.3 Parsing software

We implemented the parsing software as a Python 3 script that primarily utilizes two libraries, TensorFlow 2.0 and OpenCV, which henceforth is referred to generically as Python, TensorFlow, and OpenCV respectively. OpenCV provides a wide range of real-time computer vision functions that we leverage for the majority of the image processing (OpenCV, 2021).

The bulk of this research focused on developing Python code to parse the video footage from the Lorex security camera and label the indirect LiDAR object data in each frame according to the standards of the machine learning training requirements for TensorFlow. This results in a dataset containing:

1. Each frame saved as an image
2. Comma Separated Values (CSV) file containing a bounding box and object label for the indirect LiDAR object in each frame
3. Class file containing LiDAR object label
4. Record files for training and evaluating machine learning model

This dataset contains everything needed for training an object identification machine learning model with TensorFlow. The following section details the exact methodology of how we implement the parsing software to generate the dataset.

4. Research methodology

To produce the dataset for training a machine learning model, our Python script collects and parses video footage in the following steps:

1. Record video footage from the security camera
2. Process the footage to generate bounding boxes and object label for indirect LiDAR emissions in each video frame
3. Produce data files required to train machine learning model

We perform step 2 twice on the same footage from step 1: once using Otsu's algorithm and once using Yen's algorithm. We use LabelImg to manually mark and label the bounding boxes for each frame in the video from step 1 which we use as truth data to analyse the performance of each image thresholding image.

4.1 Record video footage

The Lorex security camera readily exports footage via the Real Time Streaming Protocol (RTSP), a standardized protocol for controlling delivery of data with real-time properties such as live video (Schulzrinne, Rao and Lanphier, 1998). This allows OpenCV to interface directly with the camera and record video footage to a computer for processing. With the camera settings configured according to the Background section, the FPS of 15 must also be provided to OpenCV when writing the video file to disk to ensure it matches the frame rate at which the camera recorded it.

4.2 Process the video footage

With the video containing indirect LiDAR emissions recorded to disk, we then process the video footage into the training dataset as follows:

1. Separate the video footage into frames
2. Calculate the threshold value for an image mask to isolate the LiDAR emission points
3. Calculate the bounding box for the light visible in the masked image
4. Write data for training use

4.2.1 Separate the video footage into frames

When reading the video file from disk, we must provide the video height, width, FPS, and encoding format to OpenCV according to the Lorex security camera settings. The Python code splits one video of indefinite length into individual frames, represented as OpenCV images, for which the next step calculates image mask thresholds.

4.2.2 Calculate threshold for image mask

For later use in calculating the bounding box of the LiDAR object in each frame, we use Otsu's and Yen's image thresholding algorithms in separate executions to isolate the LiDAR object from the original image (Figure 6(a)). This produces the image mask seen in 6(b) which we use to calculate the bounding box.

4.2.3 Calculate bounding box

The ideal bounding box encapsulates every white pixel in the image mask without encapsulating any black pixel not required to encapsulate a white pixel. Although this can be manually calculated from the minimum and maximum X and Y coordinates of all white pixels, OpenCV provides much more efficient functions that find the minimum bounding box for the contours (object edges) found in an image. Since we effectively provide the contours in the image mask, OpenCV quickly identifies them and calculates the bounding box for the LiDAR object in the image as seen in 6(c).

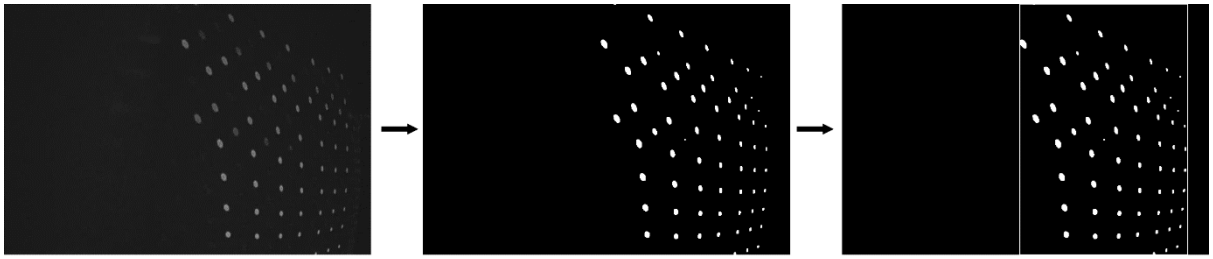

Figure 6: Image thresholding is applied to the original image (a) to derive the image mask (b) and calculate the bounding box (c)

4.2.4 Write data

With the bounding box calculated and object label known, we write the data files described in Section 3.3 for later use in training a machine learning model. We use OpenCV to save the individual frames obtained in Section 4.2.1 as PNG files. The CSV file, class file, and record files must meet specific format requirements, so we use a publicly available Python script to generate them (Meneghetti, 2020). This script formats our single "LiDAR" object label into the proper class file, as well as aggregates each image's file name, width, height, object label, and bounding box parameters into a CSV file which it then combines with the class file to generate the record file.

5. Results and analysis

We calculate the Intersection over Union (IoU) for the bounding boxes resulting from both Otsu's and Yen's image thresholding algorithms as compared to our manually calculated ground truth bounding boxes. The comparison yields 150 IoU measurements, one for each image. The Wilcoxon signed rank test comparing the 150 IoU values from both Otsu's and Yen's algorithm reports a P-value 1.05289e-43 indicating that Otsu's algorithm undoubtedly performs better than Yen's algorithm. Considering the small relative IoU standard deviations for both algorithms, a similar result can be seen intuitively in Table 7. The IoU mean of Otsu's algorithm lies more than 10 standard deviations above the IoU mean of Yen's algorithm, clearly indicating better accuracy. Both algorithms appear to produce highly accurate bounding boxes based on visual analysis, but this statistical analysis proves Otsu's algorithm is the best choice for future use in producing data for training a machine learning model.

Table 7: Intersection over Union results for both image thresholding algorithms

Algorithm	IoU Mean	IoU Standard Deviation
Otsu's	0.995169	0.0162411
Yen's	0.926803	0.0556291

6. Conclusion

The resulting framework allows us to easily record security camera footage that contains indirect cell phone LiDAR emissions and automatically mark and label the LiDAR object in each frame, saving a large amount of time that would otherwise be spent manually producing the same results. The choice of image thresholding algorithm directly influences the accuracy of the bounding boxes which behooves us to choose the best algorithm possible to produce accurate training data for a machine learning model. Testing shows that Otsu's image thresholding algorithm produces training data with 99.5% accurate bounding boxes, making it a better choice than Yen's algorithm. Therefore, we plan to utilize Otsu's image thresholding algorithm in the future to produce a large number of labelled images with which we can train a machine learning model that accurately detects LiDAR objects in a wide range of images.

Acknowledgements

We thank the government employees Ashleigh Coker and Capt Frankie Cruz (AFRL/RYAA) for their aid in hardware acquisition.

References

Aker, C. and Kalkan, S. (2017) 'Using deep networks for drone detection', *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS 2017*. doi: 10.1109/AVSS.2017.8078539.

Akopyan, V. (2016) *A modified DSLR does not make a good multispectral sensor*, *Quickbird*. Available at: https://blog.quickbird.uk/why-a-modified-dslr-does-not-make-a-good-multispectral-sensor-6766a9270820 (Accessed: 8 June 2021).

Apple (2020) *Displaying a Point Cloud Using Scene Depth*. Available at: https://developer.apple.com/documentation/arkit/environmental_analysis/displaying_a_point_cloud_using_scene_depth (Accessed: 10 September 2021).

Apple (2021) *Xcode 13 Overview*. Available at: https://developer.apple.com/xcode/ (Accessed: 10 September 2021).

Eitel, J. U. H. *et al.* (2016) 'Beyond 3-D: The new spectrum of lidar applications for earth and ecological sciences', *Remote Sensing of Environment*. Elsevier Inc., pp. 372–392. doi: 10.1016/j.rse.2016.08.018.

Feng, Q., Jun, H. and Cheng, Z. K. (2010) 'An algorithm for multi-label learning based on minimum threshold', *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, pp. 2793–2797. doi: 10.1109/WCICA.2010.5554938.

Gallagher, W. (2021) *How to use the LiDAR scanner in iPhone 12 Pro*, *Apple Insider*. Available at: https://appleinsider.com/articles/21/03/02/how-to-use-the-lidar-scanner-in-iphone-12-pro (Accessed: 7 September 2021).

Hetherington, E. (2021) *LiDAR on a phone - a new way to capture 3D data - Resource Centre*, *Esri UK & Ireland*. Available at: https://resource.esriuk.com/blog/lidar-on-a-phone-a-new-way-to-capture-3d-data/ (Accessed: 14 September 2021).

Kilic, I. and Aydin, G. (2020) 'Traffic Sign Detection and Recognition Using TensorFlow' s Object Detection API with A New Benchmark Dataset', *2020 International Conference on Electrical Engineering, ICEE 2020*. doi: 10.1109/ICEE49691.2020.9249914.

Kim, K., Hong, Y. G. and Han, Y. H. (2018) 'General labelled data generator framework for network machine learning', *International Conference on Advanced Communication Technology, ICACT*, 2018-Febru, pp. 127–131. doi: 10.23919/ICACT.2018.8323670.

Li, N. *et al.* (2021) 'Spectral imaging and spectral LIDAR systems: moving toward compact nanophotonics-based sensing', *Nanophotonics*, 10(5), pp. 1437–1467. doi: 10.1515/NANOPH-2020-0625.

Lin, G. H. *et al.* (2020) 'Self-driving Deep Learning System based on Depth Image Based Rendering and LiDAR Point Cloud', *2020 IEEE International Conference on Consumer Electronics - Taiwan, ICCE-Taiwan 2020*. doi: 10.1109/ICCE-TAIWAN49838.2020.9258010.

Lorex (2021) *4K Ultra HD Resolution 8MP Outdoor IP Camera, 200ft Night Vision*. Available at: https://www.lorextechnology.com/4k-security-camera/4k-ultra-hd-8mp-noctunnal-ip-camera/LNB8921BW-1-p (Accessed: 10 September 2021).

Marcoe, K. (2007) *LIDAR an Introduction and Overview*. Available at: http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Marcoe_LiDAR.pdf (Accessed: 13 June 2021).

McManamon, P. F., Kamerman, G. and Huffaker, M. (2010) 'A history of laser radar in the United States', in *Laser Radar Technology and Applications XV*. SPIE, p. 76840T. doi: 10.1117/12.862562.

Meneghetti, D. D. R. (2020) *detection_util_scripts/generate_tfrecord.py*, *GitHub*. Available at: https://github.com/douglasrizzo/detection_util_scripts/blob/master/generate_tfrecord.py (Accessed: 8 September 2021).

Molebny, V., Kamerman, G. and Steinvall, O. (2010) 'Laser radar: from early history to new trends', in *Electro-Optical Remote Sensing, Photonic Technologies, and Applications IV*. SPIE, p. 783502. doi: 10.1117/12.867906.

Noble, J. (2020) *FIA bans 3D cameras as part of F1 car copying clampdown*, *Autosport*. Available at: https://www.autosport.com/f1/news/fia-bans-3d-cameras-as-part-of-f1-car-copying-clampdown-4977570/4977570/ (Accessed: 15 September 2021).

Noman, M., Stankovic, V. and Tawfik, A. (2019) 'Object Detection Techniques: Overview and Performance Comparison', *2019 IEEE 19th International Symposium on Signal Processing and Information Technology, ISSPIT 2019*. doi: 10.1109/ISSPIT47144.2019.9001879.

OpenCV (2021) *Home - OpenCV*. Available at: https://opencv.org/ (Accessed: 10 September 2021).

Otsu, N. (1979) 'A Threshold Selection Method from Gray-Level Histograms', *IEEE Trans Syst Man Cybern*, SMC-9(1), pp. 62–66. doi: 10.1109/TSMC.1979.4310076.

Peker, M. (2019) 'Comparison of Tensorflow Object Detection Networks for Licence Plate Localization', *Proceedings - 2019 IEEE 1st Global Power, Energy and Communication Conference, GPECOM 2019*, pp. 101–105. doi: 10.1109/GPECOM.2019.8778602.

Pelkmann, D., Tharwat, A. and Schenck, W. (2020) 'How to Label? Combining Experts' Knowledge for German Text Classification', *Proceedings - 2020 7th Swiss Conference on Data Science, SDS 2020*, pp. 61–62. doi: 10.1109/SDS49233.2020.00023.

Phadnis, R., Mishra, J. and Bendale, S. (2018) 'Objects Talk - Object Detection and Pattern Tracking Using TensorFlow', *Proceedings of the International Conference on Inventive Communication and Computational Technologies, ICICCT 2018*, pp. 1216–1219. doi: 10.1109/ICICCT.2018.8473331.

Python (2021) *Welcome to Python.org*. Available at: https://www.python.org/ (Accessed: 8 September 2021).

Queralta, J. P. *et al.* (2019) 'FPGA-based Architecture for a Low-Cost 3D Lidar Design and Implementation from Multiple Rotating 2D Lidars with ROS', *Proceedings of IEEE Sensors*, 2019-October. doi: 10.1109/SENSORS43011.2019.8956928.

Raj, T. *et al.* (2020) 'A Survey on LiDAR Scanning Mechanisms', *Electronics (Switzerland)*. MDPI AG, p. 741. doi: 10.3390/electronics9050741.

Rezatofighi, H. *et al.* (2019) 'Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression', *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June, pp. 658–666. Available at: https://arxiv.org/abs/1902.09630v2 (Accessed: 15 September 2021).

Roh, Y., Heo, G. and Whang, S. E. (2021) 'A Survey on Data Collection for Machine Learning: A Big Data-AI Integration Perspective', *IEEE Transactions on Knowledge and Data Engineering*, 33(4), pp. 1328–1347. doi: 10.1109/TKDE.2019.2946162.

Schulzrinne, H., Rao, A. and Lanphier, R. (1998) *Real Time Streaming Protocol (RTSP)*, *Internet Engineering Task Force*. Available at: https://datatracker.ietf.org/doc/html/rfc2326 (Accessed: 10 September 2021).

Sturgeon, R. (2021) *The Privacy and Security Threat of the iPhone 12 Pro LIDAR Sensor*, *Mac O'Clock*. Available at: https://medium.com/macoclock/the-privacy-and-security-threat-of-the-iphone-12-pro-lidar-sensor-bb6de2bc4dd9 (Accessed: 15 September 2021).

TensorFlow (2018) *Basics of Machine Learning*, *TensorFlow*. doi: 10.1007/978-3-319-77800-6_8.

Tether, T. (2004) 'Subcommittee on Terrorism, Unconventional Threats and Capabilities, House Armed Services Committee, U.S. House of Representatives', 25 March. Available at: https://www.darpa.mil/attachments/TestimonyArchived(March 25 2004).pdf (Accessed: 14 June 2021).

tzutalin (2021) *LabelImg*, *Github*. Available at: https://github.com/tzutalin/labelImg (Accessed: 15 September 2021).

Vollmer, M., Möllmann, K.-P. and Shaw, J. A. (2015) 'The optics and physics of near infrared imaging', *Education and Training in Optics and Photonics: ETOP 2015*, 9793(8), p. 97930Z. doi: 10.1117/12.2223094.

Xie, L. and Guo, X. (2019) 'Object detection and analysis of human body postures based on TensorFlow', *Proceedings - 2019 IEEE International Conference on Smart Internet of Things, SmartIoT 2019*, pp. 397–401. doi: 10.1109/SMARTIOT.2019.00070.

Y, M. (2021) *Apple LIDAR Demystified: SPAD, VCSEL, and Fusion…*. Available at: https://3da.medium.com/apple-lidar-demystified-spad-vcsel-and-fusion-aa9c3519d4cb (Accessed: 8 June 2021).

Yen, J. C., Chang, F. J. and Chang, S. (1995) 'A New Criterion for Automatic Multilevel Thresholding', *IEEE Transactions on Image Processing*, 4(3), pp. 370–378. doi: 10.1109/83.366472.