Composite Edge Detection in Convolutional Neural Networks

Porter Libby
Earlham College
Department of Computer Science
pelibby16@earlham.edu

ABSTRACT

Feature detection with neural networks is a fast growing field, with big improvements to speed and capability being made every year. One of the most popular ways to perform such detection is with a CNN (Convolutional Neural Network), which is a type of deep learning network particularly adept at detecting features in images. This kind of artificial intelligence has become important, alongside a number of applications which can generate huge amounts of image data. The abundance of data in the modern world, coupled with the increasing accessibility of machine learning means that new methods must be explored for ways to efficiently process data. This paper proposes a method for testing a range of operations, comparing their outputs, and determining a set of algorithms which are most effective for this process. The goal of this work is to provide insight into various edge detection operations, and how they can be applied to a CNN to increase accuracy and efficiency without requiring additional financial or technical resources.

KEYWORDS

machine learning, artificial intelligence, convolutional neural network, shape detection, edge detection, feature detection, drone imagery, image processing

1 INTRODUCTION

The motivation for this work is to create a more accessible method for performing high-level feature detection in images. Modern surveying and data collection can produce thousands of images, and the tools that are needed to process such a volume of imagery are complex and frequently expensive. The goal is to provide insight into potential methods for producing high accuracy feature detection without the need for costly hardware, or cluster computing. Not everyone has access to industry-grade software and hardware, but this should never prevent science from being done

Machine Learning is a quickly expanding field in computer science, and there are many open-source tools available for setting up neural networks. Tools like Image AI, FastAI, PyCNN, and many more, can even be set up to run on a personal computer, utilizing a CPU, instead of a GPGPU [10] [6] [1] [16]. These libraries provide new levels of accuracy and flexibility to users.

In this paper, we propose a solution that uses modern edge detection methods and multiple types of surveying imagery, to provide the maximum amount of detail into the subterranean features of the area. The preprocessing techniques described have a cost in the overall processing time of data, but it does so in order to increase accuracy. This could be vital if it is not possible to collect more data, and the best possible results need to be achieved, even at the cost of processing time. The aim of this solution is to make use of the variety of methods available for the detection of edges and processing of images, as well as the robust information provided by drone surveying, and create a data model with the most detail possible. The proposed solution will make use of machine learning to sort and optimize the output of the collected details.

2 RELATED WORK

2.1 Applications of Edge Detection

A big part of this project is the idea that data from different edge detection algorithms run over the same images will extrapolate different details. One example is the Canny edge detection method, which is used by Soman and Kurnia et al. in their respective research. Both conclude that the algorithm is both fast and effective at identifying objects in noisy or distorted environments[13][7].

Another method, which makes use of Bayes' Theorem, is based on measuring the variance of the directions of the gradient of brightness. The probability of the event that a point belongs to the approximation of a straight segment of the isoline of brightness passing through the point being tested is computed using the technique of Bayesian estimations and used as a weight [12]. At the time this method was proposed (2002), this method had a similar time complexity as other methods such as Beaudet, Deriche–Giraudon, Harris–Stephens, and others, and is much more accurate according to test results from Sojka [12].

The Sobel operator performs the measurement of 2-D spatial gradient on an image and it emphasizes high spatial frequency regions that correspond to edges. This operator,

which is based on the local maxima and minima of derivatives, can be very efficient, and in one implementation by Wang et al. , this method is used for real-time video edge detection. Part of the efficiency in this particular implementation comes from the use of FPGA (Field Programmable Gate Array) processors because FPGAs have parallel and high computational density [15].

One more modern algorithm involves using particles for the detection. The basic idea is that at the corner the bubble has less options to move, compared to the flat edge or an edgeless window. The particles follow the edges of a shape until they reach the other side, recording their status [8]. The authors conclude that this novel method is more effective than standard methods in terms of accuracy. It remains a drawback, however, that this method is extremely new and relatively untested.

Another more modern tool for corner detection is Artificial Eye Tremors. This is a very unique method, as it is inspired by the way our eyes involuntarily move and detect shapes. It involves simulating an eye tremor by shifting the origin of the address scheme over each initial layer one location [4]. The authors conclude that while their bio-inspired algorithm is not as accurate as some other Tradition Image Processing (TIP) methods, however, it is much faster when used over large pieces of data, and still provides a reasonable estimation of corners.

Algorithms such as these are frequently used for processing surveying images, as can be seen in Soman's work, where image processing is used to detect edges in images of building rooftops, and identify the boundaries of buildings, even under cover of trees and other potential false positives [13]. Soman's implementation used the Canny edge detection method, which is an algorithm that we plan to use in our proposed solution. Another aspect of Soman's work that relates very directly is the use of filters like NDVI (normalized difference vegetation index), which allow more detail to be extracted from an aerial image.

In Teng and Xue's work, the Sobel operator is used to reduce the blurring around shapes in underwater imagery [14]. While this work does not make use of drone imagery, the Sobel operator is an important image processing tool, and might also work to reduce noise in out implementation.

3 DESIGN AND IMPLEMENTATION

3.1 Hypothesis

Our goal for this project is to show if combining multiple layers of edge detection into each image before training anAI with them can yield a higher accuracy than using the images with no preprocessing. We hypothesize that a small percentage of accuracy can be gained with our initial experiments. Such a result would also indicate that more time, research, and testing might be able to yield a more substantial increase in model accuracy.

3.2 Datasets

For this project, we hope to test multiple sets of data. Just as different algorithms can be used together to create a composite result, the project as a whole will be tested and compared with different combinations of data. Most of the data for this project came from Kaggle. The sets chosen from here were picked because they represent very different applications, which can all make use of a similar machine learning approach. Some of the data was chosen because it is binary; it detects whether an image does, or does not, contain a specific thing.

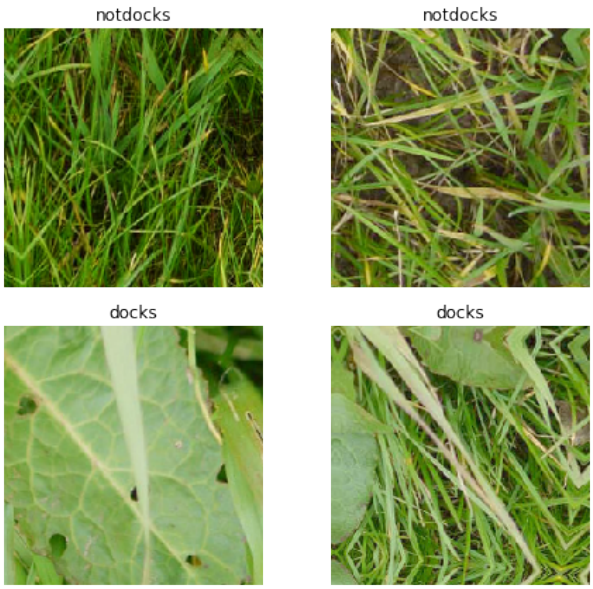

Figure 1: Detecting Docks Data Sample

3.2.1 Detecting Docks. One set used in this project is a training set for the detection of a plant called docks [2]. The set contains about 2000 images of the plant we are trying to detect and about 4000 images of other plants. The purpose of this set is to be able to detect a specific plant from a set of similar-looking images, such as those captured with a drone. Since these images will all be some a similar angle and are all images of plant matter, they can be much harder to classify than something with more distinct features. This is part of the reason this set was chosen; it represents a classification obscured by similar negative and positive cases.

3.2.2 Detecting Cats vs. Dogs. This set is another simple binary training set, with one big folder of cat images, and one big folder of dog images [11]. This set was included

because unlike the docks set, which is relatively stable in terms of color and orientation, this set is very chaotic. Its images vary in perspective, the color of the subject, surroundings, lighting, and many other parameters. This provides an opportunity to test which kind of data provides better results

Figure 2: Detecting Cats vs. Dogs Data Sample

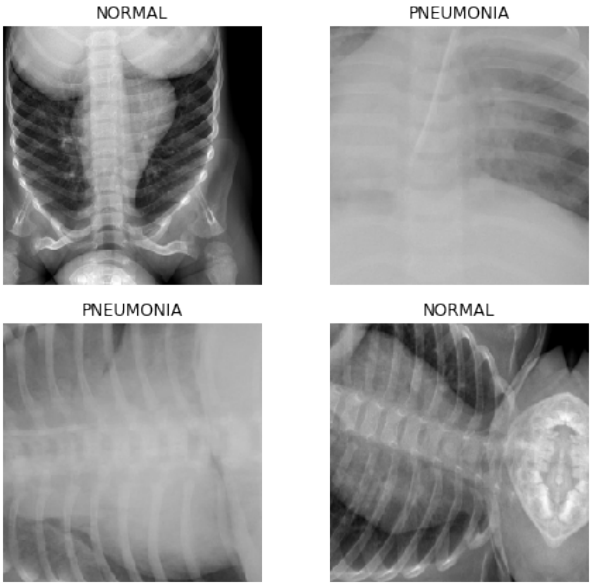

Figure 3: Detecting Pneumonia Data Sample

3.2.3 Detecting Pneumonia. Another binary data set used in this project is for training an AI to recognize an x-ray of a set of lungs which have pneumonia [9]. This set contains examples of healthy lung x-rays, as well as those diagnosed with pneumonia. This set was chosen because it is a media other than visible light imagery (the light spectrum that people see), and it is represented in grayscale. These two differences make it possible that the set will produce different results than the others, making it a good addition to our testing data.

3.2.4 Detecting Natural Features. This set, created by Intel, provides six labels; buildings, forest, glacier, mountain, sea, and street [3]. This set was included because it provides a lot of image data for each category (similar to the amount provided in other sets with only two labels), and has a small amount of well-defined categories.

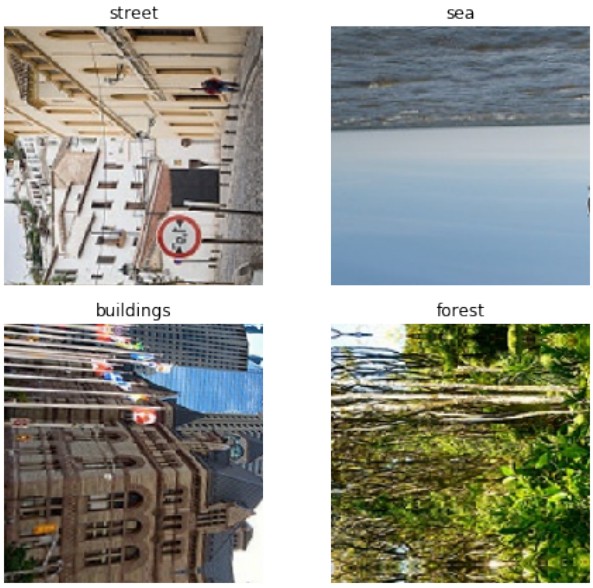

Figure 4: Detecting Natural Features Data Sample

3.2.5 Detecting Fruit Types. This set has 15 different labels, each a type of fruit, and also includes enough data to make each label comparable to those in the two-label sets [5]. This set was included in order to test an outlier to the rest, which all have relatively small numbers of labels. Our hypothesis is that sets with more labels will achieve lower accuracy than those with less.

3.3 Edge Detection

For our implementation of edge detections, we make use of two popular libraries; SciPy and OpenCV-Python. These libraries each include a handful of algorithms for edge detections, and between the two of them, a reasonably large

amount of combinations are possible. The SciPy library includes an implementation for the Prewitt edge detector (Shown in Fig 6).

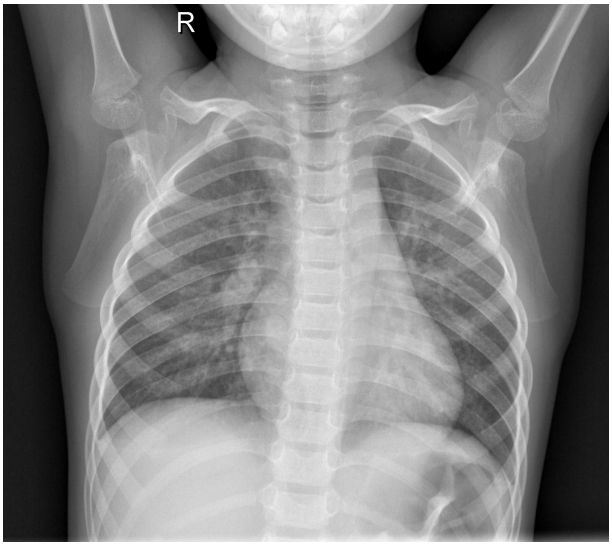

Figure 5: Sample Original Image

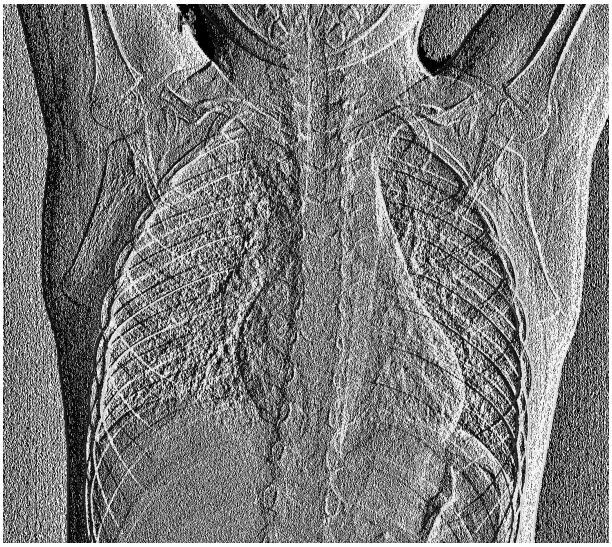

Figure 6: Prewitt Output of Fig 5

Using OpenCV, we can create simple, efficient implementations of Canny edge detection without losing any robustness or configurability. The library also provides algorithms such as the Sobel x and y operators (Shown in Fig 7 and 8) and Laplacian Edge detection. The provided implementations are fast and effective but must be tuned carefully in order to produce a meaningful result.

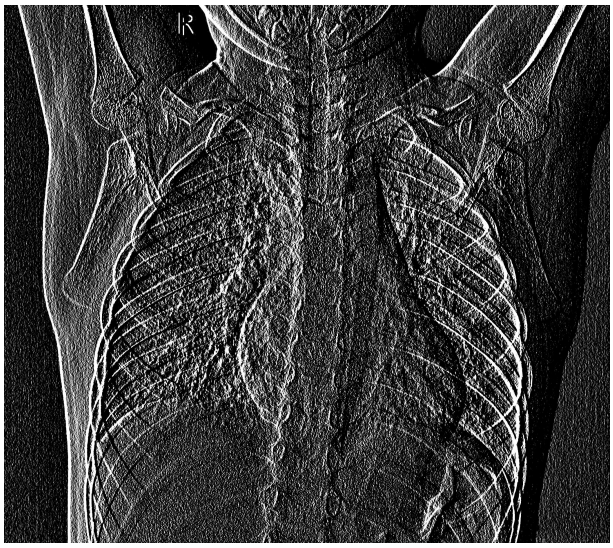

Figure 7: Sobel X Output of Fig 5

Figure 8: Sobel Y Output of Fig 5

These algorithms on their own provide a useful insight into the detectable edges of an image, but a composite result from multiple detections will provide much more depth of information. In order to combine the outputs of these detections, we make use of the Pillow image manipulation library. This allows images to be constructed from individual color channels. By assigning one to three edge detections to one channel each, a three-colored composite image can be formed. This composite will serve as the input to the neural network (Shown in Fig 9).

Figure 9: Composite of Fig 5, Fig 8, and Fig 6

3.4 Neural Network

This project also makes use of a neural network through the library fastai. Fastai is a relatively new machine learning library for Python, built on top of the torch library. Its expressed goal is to provide an industry level machine learning experience that is accessible to anyone. For this project, we use a CNN (Convolutional Neural Network), which is a kind of neural network specifically well suited to image feature detection.

The fastai library provides access to multiple kinds of learning model, which is a parameter that greatly affects the results of machine learning. For our purposes, we will use the Resnet18, Resnet 34, and Resnet50 models.

3.5 DataGlob

In order to tie all these things together, our implementation uses a python data strucutre, called DataGlob. This structure was created to account for all the data and variables associated with creating a model using the edge detection process described above. The main advantage to organizing everything into a data structure is that it can be used easily in different formats, such as terminals, Jupyter notebooks, or in a GUI.

The DataGlob keeps track of which algorithms should be used to process images, and allows the user to configure up to three to be used together. It also keeps track of where data will be found, and where temporary data and output data can be stored. once things are set up, data will be copied from the source directory, processed in the temporary directory, and placed in a structure identical to what they came from in the output directory, this way there will be one edge detected version of each image in the source directory in the same place in the output directory.

Once the data has been processed, the data structure provides easy methods for using the same data to create a model. Also stored in the Dataglob are the configurations for all of the CNN settings, such as the size each image should be, and what kind of resnet model to use. After being trained, the model will be exported into the output directory, alongside the processed data set.

4 RESULTS

Over the course of this project, we have run hundreds of hours of machine learning epochs. Most of the data that is relevant has been recorded and is kept in a detailed document which can be seen in parallel with the source code for the project.

4.1 First Tests

	Chest X-ray	Cats Dogs	Intel Images	Plant Id
Control	92.4%	**97.2%**	**87.4%**	**92%**
Canny Tight	92.4%	94.8%	78.7%	91.7%
Canny Auto	92.6%	91.8%	79.6%	88.8%
Canny Wide	93.1%	91%	77.8%	88.5%
Laplacian	93.4%	94.1%	79.9%	92.2%
Sobel X	92.3%	92.4%	78.2%	90.3%
Sobel Y	**94.7%**	90.9%	78.5%	90.5%
Prewitt	94.3%	88.3%	76.6%	89.8%

Figure 10: Single Algorithm Accuracy

For our first phase of testing, a model was created using each algorithm on its own. Canny edge detection, Laplacian operator, Sobel operator, and Prewitt edge detection were tested for this (shown in Fig 10). From this initial test, we notice a trend in the amount of difference and the ability to pre-process to positively affect results. It is detrimental in the intel images set, where there are multiple different labels for sets of various diverse images. Plant id and cats-dogs are both binary labels (they only have two states, such as true and false), and show a much closer accuracy. The chest x-ray set makes use of pre-labeled x-ray data which is all in roughly the same form and pattern, with the same colors and orientation. We hypothesize from this trend that sets with many different labels or sets with varying, chaotic images may be negatively affected by preprocessing, which those with a more standard arrangement or orientation will benefit from the edge processors' ability to filter data.

4.2 Follow Up Tests

We wanted to start testing combinations of algorithms, to see if any set of two achieves higher accuracy than the single-algorithm tests. In order to do this, we used the same set from Fig 10 which yielded positive results, and started focusing in on what worked. Since the Sobel y-axis test had the highest accuracy, the logical starting place was to test that algorithm combined with each other algorithm in pairs of two (Shown in Fig 11). Fig 9 is an example of the images generated by this preprocessing setup.

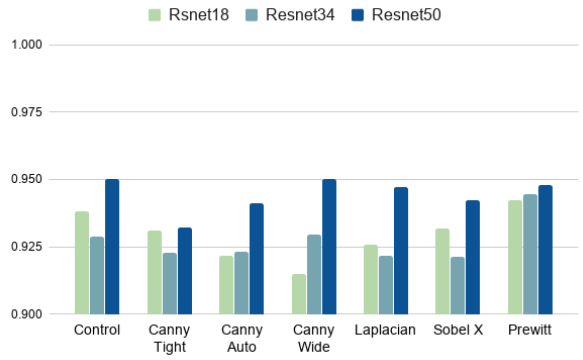

Figure 11: Sobel Y + Single Algorithm Accuracy

4.3 Final Tests

To wrap up the tests performed over the course of this project, the final tests was designed to compare a full preprocessed compression, with three different layers, each processed using a different edge detection algorithm. This kind of usage was what was intended for the project, and using results from Fig. 10 and Fig. 11, we were able to focus in on the combinations most likely to yield positive results. Just like the second set of tests, we use the Chest X-ray set, which has been the most responsive to preprocessing so far.

Fig. 12 shows the benefits of this changing scope and specificity between our tests. The blue control line represents the accuracy over training of an unaltered set of data, using no preprocessing. The orange line represents a combination of the Prewitt and Sobel Y preprocessing algorithms (the third layer in the JPEG compression will just be a copy of the original). The green line shows a new test, using the most effective combination from our previous testing (Prewitt+Sobel Y), and adding in the most effective third algorithm, Canny Wide. The result is a higher accuracy after 5 epochs than either the previous test, or the control test is able to achieve.

4.4 Applications

The applications we foresee for this work are mostly related to low-budget science. More effective data can be captured

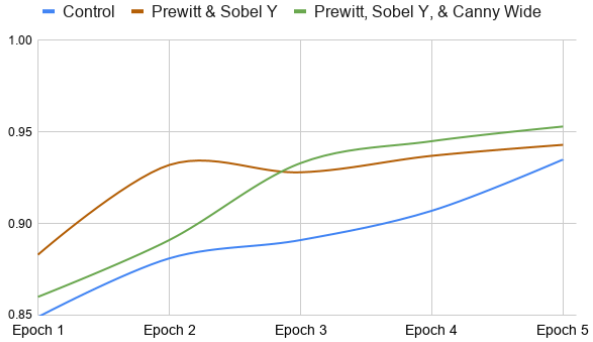

Figure 12: Sobel Y + Prewitt + Canny Accuracy

with costly hardware such as specialized lenses and cameras, and more efficient models can be created with the use of cluster computing and GPGPUs. This project, however, provides a unique level of user-friendliness to the problem of increasing accuracy without new data. We provide a free, easy to use option for increasing accuracy in some data with very low overhead compared to the time it takes to train the model itself.

Our processes described in this paper does not affect all data in the same way. With certain sets of data, an increase in accuracy may not be possible, however, with some, feature detection can be improved by several percents. The specifics that determine whether data will be positively, neutrally, or negatively affected by these preprocessing steps are not concrete, but positive results seem to correlate to binary data sets containing very similar images. For this reason, a more specific application for this work might be to use it alongside drone collected data, because of its homogenous nature, or MRI imagery, which tends to look similar from image to image. These applications have the most to gain from this kind of preprocessing method.

5 CONCLUSION

We conclude that this project has the potential to benefit some Convolutional Neural Networks, at the cost of some additional time for preprocessing. With the work that has been done so far, it is clear that not all sets of data can benefit from the kind of layered detail provided by this preprocessing method. Some sets are too diverse, or too chaotic to have emergent patterns from this processing. Other sets, however, can benefit from it; particularly those sets with a predictable orientation and noise level, such as imagery captured with a drone.

5.1 Future Work

The next step for this project would be to conduct more testing. Creating models with each permutation of a set of

algorithms is a time consuming process, and if the knobs and settings of a CNN model are also considered as mutable aspects of the project, the number of test cases increase exponentially.

Our preliminary work outlined in this project is just a starting point for refining this process, meant to demonstrate the possibility for improvement to model accuracy.

Another addition to this work would be to test more data sets. Our initial idea for the project involved collecting data with drones to test preprocessing with. This idea was scraped based on the time-frame of the project, however, we feel that, given more time, this would be the best way to refine the process and implementation described here.

ACKNOWLEDGEMENT

Thanks to Dr. Charles Peck, Dr. Igor Minevich, Dr. David Barbella, Dr. Thomas Hamm, Dr. Cynthia Fadem, Dr. Xunfei Jiang, Greg Vaughn, Mary Bogue, and the Earlham Icelandic Field Studies program.

REFERENCES

[1] Ankit Aggarwal. 2014–. Image Processing with Cellular Neural Networks in Python. https://github.com/ankitaggarwal011/PyCNN

[2] Gavin Armstrong. 2018. Open Sprayer Images. https://www.kaggle.com/gavinarmstrong/open-sprayer-images.

[3] Puneet Bansal. 2019. Intel Image Classification. https://www.kaggle.com/puneet6060/intel-image-classification.

[4] John Fegan, Sonya Coleman, Dermott Kerr, and Bryan Scotney. 2018. Fast, Biologically Inspired Corner Detection Using a Square Spiral Address Scheme and Artificial Eye Tremor. In *Proceedings of the 2018 International Conference on Artificial Intelligence and Pattern Recognition (AIPR 2018)*. ACM, New York, NY, USA, 61–65. https://doi.org/10.1145/3268866.3268883

[5] Chris Gorgolewski. 2020. Fruit Recognition. https://www.kaggle.com/chrisfilo/fruit-recognition.

[6] Jeremy Howard et al. 2018. fastai. https://github.com/fastai/fastai.

[7] Rahmadi Kurnia, Melia Asmita, and Ikhwana Elfitri. 2017. Object Detection on Hindered Condition by Using Chain Code-based Angle Detection. In *Proceedings of the 2017 International Conference on Telecommunications and Communication Engineering (ICTCE '17)*. ACM, New York, NY, USA, 50–56. https://doi.org/10.1145/3145777.3145780 event-place: Osaka, Japan.

[8] Pasindu Kuruppuarachchi and Stanislav S. Makhanov. 2018. Corner Detection via Walking Particles. In *Proceedings of the 2018 International Conference on Communication Engineering and Technology (ICCET '18)*. ACM, New York, NY, USA, 15–17. https://doi.org/10.1145/3194244.3194252

[9] Paul Mooney. 2018. Chest X-Ray Images (Pneumonia). https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia.

[10] Moses and John Olafenwa. 2018–. ImageAI, an open source python library built to empower developers to build applications and systems with self-contained Computer Vision capabilities. https://github.com/OlafenwaMoses/ImageAI

[11] Chetanim Ravan. 2018. Dogs Cats Images. https://www.kaggle.com/chetankv/dogs-cats-images.

[12] Eduard Sojka. 2002. A New Algorithm for Detecting Corners in Digital Images. In *Proceedings of the 18th Spring Conference on Computer Graphics (SCCG '02)*. ACM, New York, NY, USA, 55–62. https://doi.org/10.1145/584458.584469

[13] Kritik Soman. 2019. Rooftop Detection Using Aerial Drone Imagery. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data (CoDS-COMAD '19)*. ACM, New York, NY, USA, 281–284. https://doi.org/10.1145/3297001.3297041

[14] Lu Teng and Feng Xue. 2018. Underwater Image Processing: MSRCR Algorithm Based on the Guided Filter of Sobel Operator. In *Proceedings of the Thirteenth ACM International Conference on Underwater Networks & Systems (WUWNet '18)*. ACM, New York, NY, USA, Article 22, 2 pages. https://doi.org/10.1145/3291940.3291959

[15] Jin Wang, Honggang Wang, Shaoen Wu, Xiaodong Lin, and Qing Yang. 2015. Design and Implementation of Real-time Sobel Edge Detection on FPGA for Mobile Device Applications. In *Proceedings of the ACM International Workshop on Mobility and MiddleWare Management in HetNets (MobiMWareHN '15)*. ACM, New York, NY, USA, 9–14. https://doi.org/10.1145/2757757.2757759

[16] Xin Zhao "Yupei Wang and Kaiqi Huang". 2017. Deep Crisp Boundaries. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*.