Public Interactive Display

Ke He, Richard Green
Department of Computer Science and Software Engineering, University of Canterbury

This paper presents a way of improving the coffee waiting experiment by playing a game of snake with a public display. The input of the game consists of a hand detector built with Single Shot Detector(SSD) and a face detector using Histograms of Gradient(HOG) algorithm. We provide empirical evidence to show that both detectors are able to detect their respective objects reliably with high precision and recall, and efficiently with approximately 5 frames per second with little to no GPU acceleration. This paper has laid a solid foundation for any further work to be done with public interactive displays.

***Index Terms*—hand tracking, public display, deep learning, face detection**

I. Introduction

WAITING for a cup of coffee to be made is a boring process, especially if you are buying one from the Reboot Cafe with not many seats available. The only entertainment you are likely to get is from the TVs above the cafe, but it just shows some tedious advertisements. To fully utilize the TVs and make the coffee waiting experience more enjoyable, this paper presents a way for coffee buyers to play a classic game of snake with the public displays above the cafe. The snake game consists of a hand detector and a face detector to detect positions of user's hand and face, and a naive association algorithm is used to associate user's hand with the corresponding face.

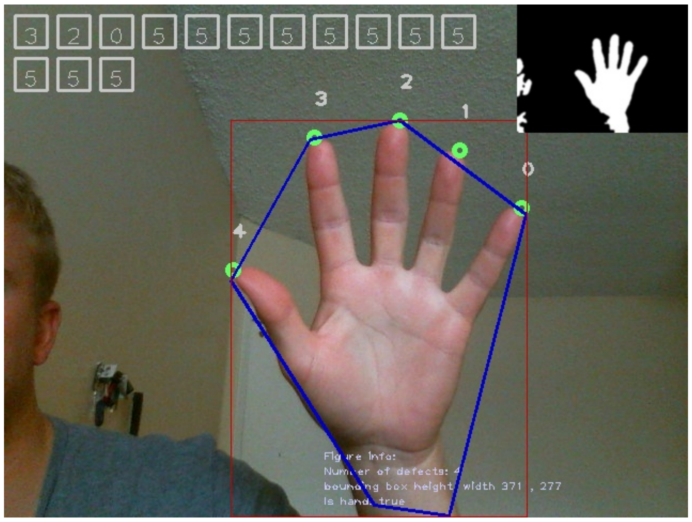
Both the hand and face detection algorithm must be capable of detecting multiple hands and faces in a public environment with a web camera. The algorithms also have to reliably detect a users hand or face from a distance away while being efficient enough to process multiple frames per second. Previous hand and face detection algorithms are examined and a possible solution proposed with extensive experiments done to test the accuracy and efficiency of the proposed solution.

II. Background and Related Work

A. Hand Detection

Wilson *et al.* [1] have examined the hand gesture recognition algorithm proposed by Andresen [2] using hand color sampling in RGB color space. The algorithm is initialized by sampling colors from multiple regions of the hand and is stored as a color profile. For each sampled color, the image is filtered using the sampled color as threshold into binary images. The resulting filtered image is combined to give a complete image of the hand with background extracted. Next, median blur is applied to smooth and eliminate noise in the binary image. The hand is then processed by applying a convex contour onto the binary image and irrelevant defects filtered out. This algorithm is able to detect the hand gesture and finger position accurately as shown in figure 1.

Ghotkar *et al.* [3] proposed three hand gesture recognition systems with color sampling and evaluated the success of each system. The most successful method proposed in the paper involves separating hand color from the background in HSV color space. Canny edge detection algorithm is used to detect edges of the filtered image and apply morphological closing to fill gaps and holes in the image. Finally a edge traversal algorithm is applied to eliminate background edges detected. The resulted contour of the hand is then matched against a set of templates. This paper shows that the choice of color space is a critical factor in any color-based hand segmentation algorithm, and HSV color space yields the best performance.



```
Figure Info:
Number of defects: 4
bounding box height width 371 , 277
is hand: true
```

Fig. 1. Result of the hand detection algorithm [2]

Other hand detection algorithms require additional hardware. Wang *et al.* [4] presented a way of real-time hand tracking and detection using a color glove. Participants must put on a color glove in order for the program to recognize hand position and gesture. This method provides a quick and accurate tracking of the hand, but the hardware limitations make it impractical.

Mitra *et al.* [5] have surveyed a wide range of gesture recognition techniques. In general, gesture recognition algorithms can be categorized into three categories: statistical modeling methods, computer vision methods, and connectionist(artificial neural network) methods.

Statistical methods such as hidden Markov model(HMM) [6], particle filtering and condensation algorithm, and finite state machine(FSM) are used frequently in dynamic gesture recognition, where features from a time-sequential image are extracted and matched with a known template using relevant statistical methods.

Typical Computer vision algorithms consist of two stages, feature extraction, and classification or clustering. First, features, for example, shape, texture, and color, are extracted manually by experts, then the extracted features are classified or clustered by machine learning algorithms such as support vector machine, k-nearest-neighbor or naive Bayesian classifier. Due to the lack of time-sequential processing, computer vision algorithms are often used for static gesture recognition [5].

Connectionist or artificial neural network(ANN) methods also have a two-stage process: feature extraction, and classification or clustering. Unlike computer vision algorithms where features are manually extracted, ANN is able to learn the features themselves by simulating how the human brain functions. However, ANNs are extremely computationally expensive and hard to train.

All methods mentioned above have disadvantages. Color sampling algorithms [1][3] are extremely sensitive to background lighting and occlusions. Hardware-assisted algorithms [4] is impractical in a public environment. Statistical methods such as HMM, particle filtering, and FSM are computationally heavy. Computer vision algorithms require a feature extraction process that is extremely labor consuming. Connectionist approaches are computationally expensive and difficult to train.

B. Face Detection

Viola *et al.* [7] developed a face detection system capable of rapidly processing images with high accuracy. Unlike previous attempts which focus on pixels, this framework focuses on features. It is able to do so by using a new way of representing image called integral image. The integral image represents a pixel located at (x, y) by sum all the pixels above and to the left of (x, y), this representation allows whole features to be classified by an AdaBoost classifier. The AdaBoost classifier was trained to detect a small number of important features to speed up the classification process. Moreover, the framework is able to cascade multiple complex classifiers in order to focus on more promising areas of the image, which further boosts the speed of the face detection. The idea of classifying features and cascading feature structure is very similar to the current state-of-the-art object classification algorithm using deep convolutional nets, which will be discussed later.

A reliable and lightweight algorithm, histograms of oriented gradient(HOG) was proposed by Dalal *et al.* [8]. The algorithm first converts an image into a grey-scale image, then replaces each pixel in the gray-scale image with a gradient vector that points to darker areas. By considering gradient vectors instead of pixel values, the effects of lighting and skin color on detection can be greatly reduced, since the gradient vectors are invariant to color. Using a vector to represent each pixel contains too much detail and tends to overfit the data. Therefore the image is divided into 16x16 pixel blocks and each

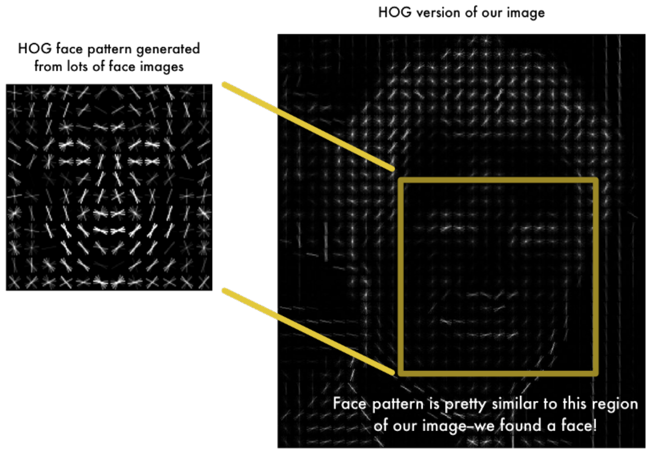

HOG face pattern generated from lots of face images
HOG version of our image
Face pattern is pretty similar to this region of our image–we found a face!

Fig. 2. Left: sum of HOG of multiple faces. Right: HOG of the input face [9]

block is represented by the most frequent gradient direction. Finally, the image is compared with a template HOG consists of multiple faces to determine if there are any faces in the image(see figure 2).

C. Object Detection

Deep learning has gained huge popularity in recent years, and have made major break-throughs in problem-solving from multiple domains ranging from image-processing to natural language processing. Object detection is no exception.

1) Neural Networks

Neural networks were originally inspired by modeling the biological neural system, hoping that computers can "learn" features themselves. Surprisingly, have been in the literature for a long time. The first working multi-layer perceptron was developed by Ivakhnenko *et al.* [10] in 1965. However, neural networks have been ignored by the scientific community for many years due to a large number of computations and memory are required by neural networks(a neural network can easily have millions of neurons). With growing CPU and GPU power and advances in training techniques, a set of more efficient and accurate methods have been developed, known as deep learning.

2) Deep Learning

Deep learning is a subset of representation learning where the algorithm is capable to automatically discover features or representation that is needed for detection and classification directly from raw input data. Deep learning methods are able to do so by having multiple levels of representations. Each representation level is obtained by transforming the representation of the previous level into a slightly more abstract representation. Multi-layer representation with parameter sharing has greatly reduced memory requirements in building and training a deep neural network [11].

Deep learning is by far the most popular and accurate method used in object detection. Convolutional Neural Network(CNN) is a class of deep learning and is often used in visual object detection. Multiple large-scale image classification and object detection competitions are held each year to showcase state-of-the-art CNN architectures and training techniques. For example, ImageNet Large Scale Visual Recognition Competition [12] is widely used as a benchmark in object classification. Competitors are faced with challenges to categorize millions of pictures into hundreds of categories. Winners of each year's competition present new architectures and training techniques to improve the speed and accuracy of object classification. Notable winners include AlexNet [13], VGG net [14] and ResNet [15].

III. Proposed Solution

A. Hand Detection

Traditional hand detection algorithms are extremely limited. Statistical methods using HMM, particle filtering and FSM are computationally heavy; color sampling methods are extremely sensitive to background lighting and occlusions, and require a hand sampling phase[17]; hardware assisted methods are infeasible in a public environment. Due to such limitations, we will use convolutional neural networks to detect hands.

Conventional convolutional neural networks architectures used in object detection, for example, Region-CNN(R-CNN) [18] is extremely computationally intensive for real-time applications [16]. The reason behind it is that R-CNN has to propose all regions of interest in the image with a neural network and then apply classification algorithms with another neural network. The two-step process greatly increases the time required to detect an object. Single-Shot Multi-box Detector(SSD) [16] solves the above problem by combining region proposal and detection in one step. SSD have fixed bounding boxes of various resolution at various locations that are manually but intelligently chosen. During a forward pass of the network, the last convolutional layer is able to show dominant features of the image allowing predefined bounding boxes to be drawn. Figure 3 shows the structure of SSD [16].

As with any other convolutional neural networks, SSDs can take a long time to train. To shorten the training time, we will apply transfer learning on to a pre-trained SSD network. The dataset used to train our network is the egohands dataset [19].

B. Face Detection

As mentioned earlier, using deep learning techniques for object detection is computationally heavy, using another pre-trained SSD for face detection will significantly decrease the frame rate. Therefore the lightweight HOG algorithm is used for face detection instead. The tradeoff is that HOG does not work well when there are shadows or occlusions on user's face or the face is slanted/tilted too much, but it is reasonable to assume that there will sufficient lighting in Reboot cafe and the user will always face the camera.

C. Game Play

The game to be played is a simple single player game of snake. The game can be separated into two stages, initialization stage and playing stage.

The initialization stage requires the participating user to raise their hand above their head for 3 seconds to start. In a public environment there will be multiple hands and multiple faces in the same frame, therefore it is important to associate the hand detected with the correct face detected. This is done by limiting the hand positions to be within a certain range of its head, depending on the head size. In case of multiple hands above the face, the correct hand is chosen to be the one with the largest height. A few assumptions have to be made in order for the initialization stage to work: the user is not obstructed by any other user and user have their own "personal bubble" where other users are unlikely to trespass. A pseudo code of the initialization algorithm is listed in algorithm 1.

Algorithm 1: Initialization algorithm

```
1  Find all faces in the frame, sorted by size of bounding
     box;
2  Find all hands in the frame, sorted by height of bounding
     box;
3  for each face detected do
4      for each hand detected do
5          if $face_{left} < hand_x < face_{right}$ then
               /* use < since lower y values means
                  higher in screen coordinates        */
6              if $hand_y < face_{top}$ then
7                  found the right hand;
8              end
9          end
10     end
11 end
```

Once the initialization stage is finished, the user's face is cropped from the screen and stored for identification. A smaller control rectangle is made with user's hand located at the center. The control rectangle has the same ratio as the screen, and user's hand's relative location in the control rectangle is projected on to the screen. The purpose of the control rectangle is so that the user can move the snake across the screen with minimal hand movement so that others are not disturbed by the playing user. Once the control rectangle is set, there will be no need to detect the face as only the control box is of interest. In case of multiple hands detected in the control box, the hand with the largest height in the rectangle is processed based on the assumption that there are no other customers in front of the user.

The goal of the snake game is to eat as much food as possible in a time period(60 seconds). The food position will be generated randomly, and limited to a maximum of 5 food spots per frame. The user can control the snake by moving their hand inside the control rectangle. The snake will move directly towards the projected hand location on the screen. Once the game finishes, the score is compared with previous high scores and stores the highest score with user's picture. Figure 4 shows both initialization interface and game-play interface.

IV. Results

A. Experiment setup

This setup used to conduct the following experiments are done under a 64 bit Linux mint 18.3 laptop with a quad-core

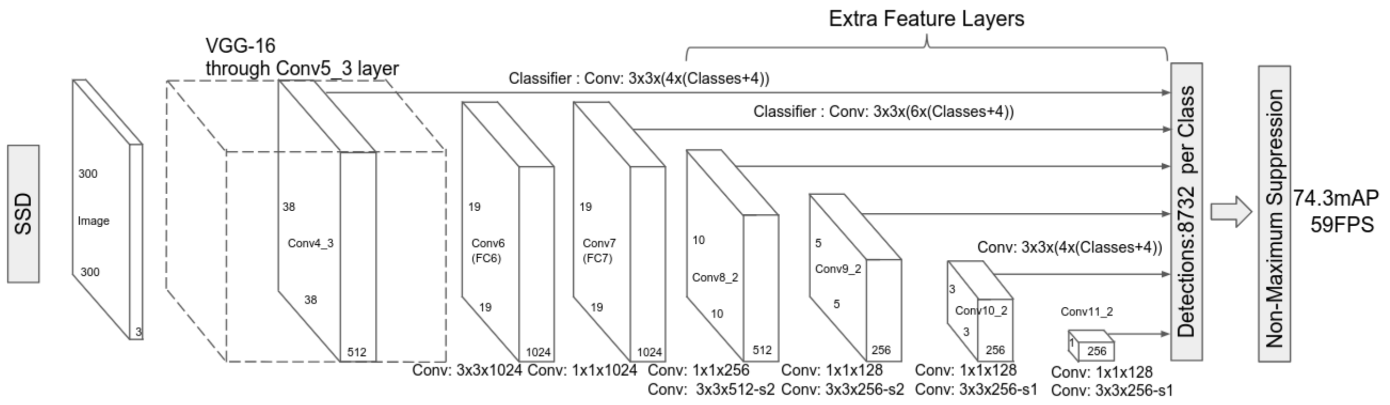


Fig. 3. Structure of SSD [16], the network is based on VGG because of its efficiency and ability to apply transfer learning. The extra feature layers predicts the offsets to the fixed bounding boxes.

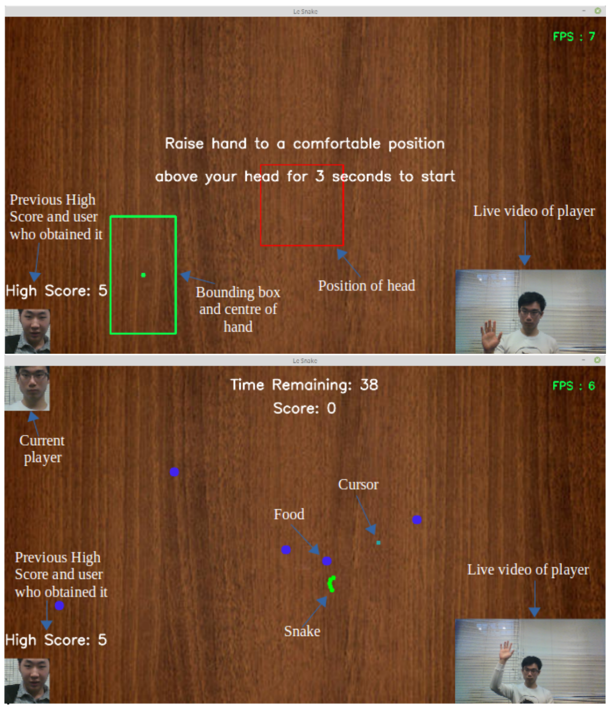

FPS : 7
Raise hand to a comfortable position
above your head for 3 seconds to start
Previous High Score and user who obtained it
High Score: 5
Bounding box and centre of hand
Position of head
Live video of player
Time Remaining: 38
Score: 0
FPS : 6
Current player
Cursor
Food
Previous High Score and user who obtained it
High Score: 5
Snake
Live video of player

Fig. 4. Top: The initialisation screen. Bottom: The gameplay screen

Intel Core i7-4700MQ at 2.4GHz, 8GB RAM and integrated graphics card. The software is written in Python 3.5 with OpenCV 3.2 used as the primary computer vision library. For hand detection, the SSD is implemented with Tensorflow library and the detector is trained with EgoHands dataset [19]. Face detection is done using dlib library. The camera used is integrated HP TrueVision HD with a resolution of 1280x720.

B. *Object Detection Metric*

To test the accuracy of the face and hand detector, we are going to calculate the precision and recall values at 0.5 intersection over union(IoU) value. Precision represents the proportion of the detected object that is correct, while recall represents the proportion of correct values that are detected.

IoU is often used in popular object detection challenges such as Pascal VOC [20], and it calculates the ratio between area of overlap and area of union of the ground truth label and predicted label. A high IoU means our predicted label is very close to the ground truth label. For object detection tasks, an IoU of over 0.5 is considered good enough, since the exact boundary of a face or hand is hard to determine. By definition, a true positive occurs when the detected object has an IoU of over 0.5, else it is a false positive. Therefore the precision and recall can be defined as follows.

$$precision = \frac{true\_positive}{true\_positive + false\_positive}$$

$$recall = \frac{true\_positive}{true\_positive + false\_negative}$$

We will use our own set of labelled images rather than the benchmark FDDB [21] face detection or the Oxford [22] hand data set. This is because benchmark datasets contain a variety of images with different levels of shadows, occlusions, and orientation, which is unlikely to happen under our circumstances. Our dataset consists of photos that are taken from actual gameplay as well as photos from the internet. We wish to obtain a face precision and recall of greater than 0.9 and hand precision and recall of greater than 0.8. Hand detection are expected to be lower because of a large number of hand gestures and orientations available which is difficult to detect accurately.

C. *Object Detection Performance*

The results of object detection is summarized in table I.

Face Precision@0.5IoU	0.909
Face Recall@0.5IoU	0.909
Hand Precision@0.5IoU	0.821
Hand Recall@0.5IoU	0.852

TABLE I
RESULTS OF HAND AND FACE DETECTION

Both face and hand detection have achieved acceptable precision and recall. Upon inspecting the images that are detected

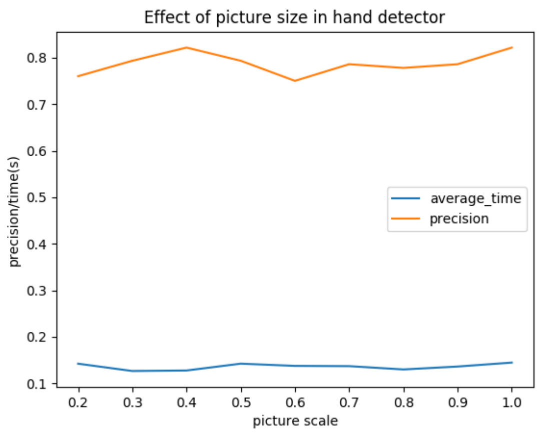


Fig. 5. The effect of picture size against precision and time in hand detector

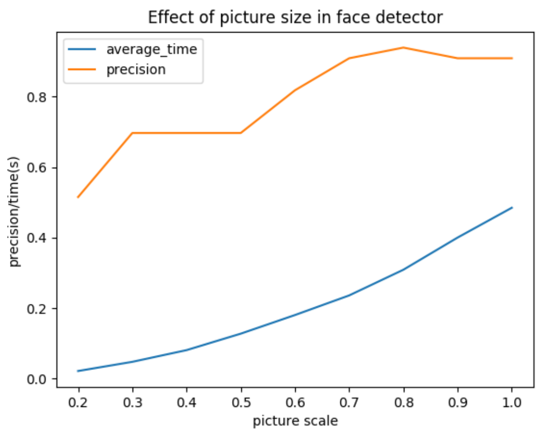


Fig. 6. The effect of picture size against precision and time in face detector

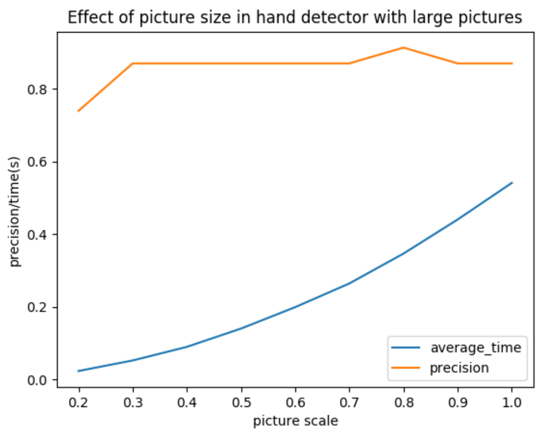


Fig. 7. The effect of picture size against precision and time in face detector with small images filtered out

incorrectly, the incorrect face pictures are often slanted in various directions, which is unlikely to happen under practical circumstances. Erroneous hand pictures mostly have hand placed beside a face, making the bounding box larger than it should have been. Fortunately, our system only considers the center of the hand and a larger bounding box does not alter the center of the hand much.

D. *Performance Analysis*

The proposed approach must run in real time, therefore it is crucial to examine the speed of the two algorithms and find ways to optimize them if they are too slow. One of the easiest ways of optimization is to reduce the image size so that it can be processed faster. However, this will negatively affect the accuracy of the detector. We wish to find the optimal size to reduce the frame while maintaining a high accuracy. The following experiments are done with sample pictures scaled from 0.2 to 1 in increments of 0.1 and the processing speed and accuracy recorded.

Figure 5 shows that the time taken for the hand detector is invariant to picture size and the precision does not vary significantly.

Figure 6 shows that the time took decreases as picture size decreases, but so does the accuracy. The reason for the decrease in accuracy was not due to the algorithm, but due to the inconsistency of the resolutions in the test image. Some images are already small, reducing its size to 0.2 are deemed to fail. To confirm our hypothesis, a follow-up experiment was done by filtering out image resolutions that are less than 700x400(approximately 70KB). The result of the experiment is shown is figure 7. The follow-up experiment confirms our hypothesis and proves that reducing image size is an effective way of improving the speed of HOG algorithm with no huge loss in accuracy. The ”sweet spot” for reducing image size is around 0.3.

We have taken advantage of this property by reducing the image input size for processing. By reducing image width and length to a quarter (0.25), and processing every second frame of the video input, the snake game is able to run at 9 10 frames per second in the described environment.

E. *Limitations*

1) *Hand and Face Detection*

The HOG algorithm is based on gradient vectors, which makes it very sensitive to shadows or occlusions that partially changes the pixel values of the face, which alters the gradient vectors. The hand detection algorithm fails to detect the hand if it is tilted horizontally more than 30 degrees or the user holds up a fist instead of a palm. This is likely due to the egohands dataset [19] used for training is from a egocentric view, where inner fists and tilted hands are rarely shown. This can be improved by training the neural network on a better set of test data obtained via real gameplay.

The proposed system has not been tested in a public area due to time constraints, although the system works well in single-player mode, its performance in a real setting is still unknown. Thus more testing needs to be done in a real setting.

2) *Game*

Currently the game is very boring to play. To make the game more interesting, obstacles/traps and power ups such as speed boost and score multiplier can be added. The GUI for the game is the default OpenCV GUI, which is not aesthetically pleasing nor supports float point pixel locations. It is possible to improve the GUI by using a proper graphical module such at tkinter to improve appearance or even a real game engine.

V. Conclusion

This paper presents a way of improving the coffee waiting experiment by playing a game of snake with a public display, and have laid a solid foundation for any further work to be done. Face detection was done using Histograms of Gradient(HOG), which achieved a precision of 0.909and recall of 0.909at 0.5 IoU using our own sample data. Hand detection using Single Shot Detector(SSD) achieved a precision of 0.821and recall of 0.852at 0.5 IoU. We are able to achieve 9 10 frame rates per second by reducing the width and height of the video input by a quarter and processing every second frame.

The major contribution of this paper is the SSD implemented that is able to solve previous hand detection problems such as sensitive to color, background, occlusions, and shadows while being fast enough to run in real time.

The proposed system is far from perfect: the face detector fails to detect masked or partly covered face and faces that are slanted or tilted; the hand detector fails to detect certain orientations and gestures of the hand or when the hand is in front of a face. The game being played is also dull

References

[1] R. G. Aaron Wilson, "Interactive public display: Upper body pose detection of multiple subjects," Tech Report, 2017.

[2] M. S. Simen Andresen Vegar Osthus. (2013). Hand tracking and recognition with opencv, [Online]. Available: http://simena86.github.io/blog/2013/08/12/hand-tracking-and-recognition-with-opencv/.

[3] A. S. Ghotkar and G. K. Kharate, "Hand segmentation techniques to hand gesture recognition for natural human computer interaction," *International Journal of Human Computer Interaction (IJHCI)*, vol. 3, no. 1, p. 15, 2012.

[4] R. Y. Wang and J. Popovic, "Real-time hand-tracking with a color glove," in *ACM transactions on graphics (TOG)*, ACM, vol. 28, 2009, p. 63.

[5] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311–324, May 2007, ISSN: 1094-6977. DOI: 10.1109/TSMCC.2007.893280.

[6] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 1992, pp. 379–385. DOI: 10.1109/CVPR.1992.223161.

[7] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004, ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000013087.49260.fb. [Online]. Available: https://doi.org/10.1023/B:VISI.0000013087.49260.fb.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, IEEE, vol. 1, 2005, pp. 886–893.

[9] A. Geitgey, *Machine learning is fun! part 4: Modern face recognition with deep learning*, https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78, 2016.

[10] A. Ivakhnenko and V. Lapa, *Cybernetic Predicting Devices*, ser. Jprs report. CCM Information Corporation, 1973. [Online]. Available: https://books.google.co.nz/books?id=FhwVNQAACAAJ.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014. arXiv: 1409.0575. [Online]. Available: http://arxiv.org/abs/1409.0575.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, Springer, 2016, pp. 21–37.

[17] D. Victor, *Real-time hand tracking using ssd on tensorflow*, https://github.com/victordibia/handtracking, 2017.

[18] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. arXiv: 1311.2524. [Online]. Available: http://arxiv.org/abs/1311.2524.

[19] S. Bambach, S. Lee, D. J. Crandall, and C. Yu, "Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions," in *Computer Vision (ICCV), 2015 IEEE International Conference on*, IEEE, 2015, pp. 1949–1957.

[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, *The PASCAL Visual Object Classes Challenge*

2012 (VOC2012) Results, http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[21] V. Jain and E. Learned-Miller, “Fddb: A benchmark for face detection in unconstrained settings,” University of Massachusetts, Amherst, Tech. Rep. UM-CS-2010-009, 2010.

[22] A. Mittal, A. Zisserman, and P. H. S. Torr, “Hand detection using multiple proposals,” in *British Machine Vision Conference*, 2011.