Dynamic Gesture Recognition in Real-time Based on Generated Datasets

Anonymous CVPR submission

Paper ID 1917

Abstract

Dynamic gesture recognition is promising for Human-Machine Interaction. Yet how to detect the gestures with various appearances in different light conditions and complex backgrounds remains a problem. This paper tackled the problem by generating gesture datasets artificially and developing the deep neural network based system. Firstly, we realized that traditional datasets of real-world photos had several drawbacks, such as hard to collect and lack of variety; thus we proposed a brand new method to generate datasets artificially by augmenting and synthesizing a basic dataset with various background images. It not only guaranteed the quality and quantity of datasets, but greatly saved time on collecting and labeling images compared to traditional ways. Then, we developed our real-time dynamic gesture recognition system based on modified YOLO and discrete-HMM and trained it with our generated datasets. Finally, the system was tested in an objective and pragmatic way with recorded videos, which verified its robustness in real-world detection.

1. Introduction

Gesture is one of the basic ways for us to communicate with each other. Recently, it becomes an alternative approach for human-machine interaction. Actually, gesture recognition can be used in many applications: interacting with computers, playing games, communicating with robots, interpreting and translating sign language, augmented reality and virtual reality. However, hand gesture recognition is still challenging, since gestures have various appearances in different light conditions and backgrounds. Recently, many approaches [13, 21, 9] with depth sensors turned out to be effective. In comparison, human does not need these supplementary devices. We can merely take a glance and tell where and what the gestures are. Therefore we developed a real-time dynamic gesture recognition system based on video streams without depth information to handle the problem in a more natural way. Firstly, as shown in Figure 1, we modified the YOLO [15] model for gesture detection. The model ran in real-time and had such high accuracy both in localization and classification that the trajectories of gestures in the frame sequence can be easily obtained from its output bounding boxes. Then, an algorithm for trajectory classification derived from Hidden Markov Model (HMM) was designed for dynamic gesture recognition, so that the trajectories of moving hands in the video can be distinguished into basic shapes.

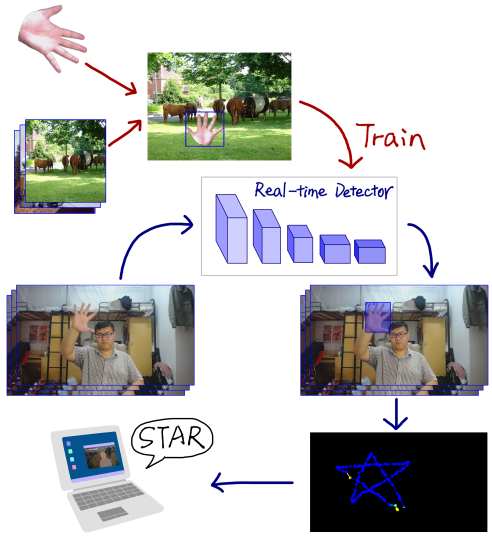

Figure 1. **Overview of our system.** A Real-time Detector derived from YOLO can locate and classify static gestures accurately in real-time. It was trained by the gesture dataset which was generated using our method of augmenting the basic gesture models (e.g. the palm) and synthesizing them with multiple backgrounds. With the trajectories formed by detected gestures in the frame sequence, the HMM can figure out the meaning of dynamic gestures (e.g. the STAR shape).

Meanwhile, though novel designs of systems may lead to better performance, further improvement can be achieved by improving the quality of training data, where ImageNet [17] was a representative practice. As for the datasets of hands and gestures, it was so hard to find a suitable one. The early datasets such as Jochen Triesch Static Hand Posture Databases [20, 19] and Sebastien Marcel Dynamic Hand

Posture Database [7] had very limited samples per gesture, and their backgrounds were either single-colored or monotonous. The Oxford's Hand Dataset [3] contained a variety of real-scene images, including manually labeled hands. Sadly, the dataset only aimed at hand detection and the hands were not classified. VIVA challenge dataset [10] took videos with depth channel into consideration for gesture recognition.

However, it is unavoidable that the appearances of gestures are always changing according to the specific person and environment. Collecting and labeling gesture images will be a great workload. To overcome these problems, we came up with the idea of artificially generating datasets. We firstly collected several basic images of target objects as basic models, and augmented them with multiple transformations. Then, by randomly synthesizing basic models and complex backgrounds, the quantity and quality of the dataset were both guaranteed and the work of labeling was eliminated, see Figure 1.

The main contributions of this paper are: We developed a real-time dynamic gesture recognition system, which performed robustly in varying light conditions and complex backgrounds. Meanwhile, we proposed a method to generate datasets artificially by augmenting and synthesizing, and validated its applicability. Finally, we tested our system with datasets which were totally different from training set, and demonstrated that this test method was more objective and pragmatic to evaluate the systems aiming at detecting gestures in reality.

2. Related Works

Gesture detection and dynamic gesture recognition. In early years, features like color, edge and motion were frequently used for hand detection and segmentation [14, 5]. With the development of machine learning, many studies [11, 18] proved the potential to use CNNs for gesture recognition, but more or less, they evaded the problems of light conditions, backgrounds, rotation, scale, etc. Meanwhile, some supplementary devices were introduced, such as Kinect, which can provide RGBD images or videos containing depth information. Molchanov *et al*. [9] developed CNN networks to make use of the depth information and worked out well on VIVA challenge. Qian *et al*. [13] presented a real-time system for hand tracking utilizing a depth sensor. Althof *et al*. [1] described a context-specific approach for dynamic hand and head gesture analysis based on infrared cameras.

Data augmentation and synthesizing. Training data is an important part for putting neural networks in use. Sometimes millions of pictures were needed to achieve high accuracy [8]. Many authors have proposed data augmentation strategies. Krizhevsky [6] used translation, mirroring and RGB jittering to augment training and testing images for image classification. Simonyan and Zisserman [2] used similar augmentation on video frames to train CNNs for human activity recognition. For video augmentation, Pigou *et al*. [12] translated the video frames temporally to apply spatial transformations. Molchanov *et al*. [9] additionally transformed the frames temporally by displacing frames.

As for the synthetic datasets, methods were proposed for different purposes. Rozantsev *et al*. [16] proposed an approach to synthesize similar and realistic images given corresponding 3D models and background images. Masi *et al*. [8] enriched existing datasets by changing the appearances of faces using 3D face models, which were used for face recognition. These methods were either domain specific or lacking universality. We proposed a more general approach which was easy to implement and able to apply in different areas. In a broader sense, the method can be used to synthesize some datasets related to special themes which were difficult to obtain, such as military objects.

3. Dataset

We aimed at training detectors based on deep neural networks that can recognize various gestures accurately with complex backgrounds and varying light conditions. With existing elegant network architectures, an appropriate dataset is rather important. And as discussed in Sec. 1, the current situation left us no way but to make our own dataset.

Sec. 3.1 firstly introduced the preprocessing method to obtain basic models. Then Sec. 3.2 described how to augment basic models and how to synthesize the augmented models with the background set to get the generated dataset. Finally, Sec. 3.3 discussed the advantages of our generating method and its promising application for the advancement of deep learning.

3.1. Preprocessing

Obtaining the basic models of target objects is the precondition of augmenting and synthesizing datasets. Taking our gesture recognition dataset as an example, the basic model set is a set of gesture images captured from different perspectives. In our gesture recognition dataset, we defined five basic gestures: palm, flat, fist, one, two, as depicted in Figure 2. Totally 150 photos were processed and turned into basic models.

Figure 2. **Basic Gestures.** The figure shows five samples of basic models in each class.

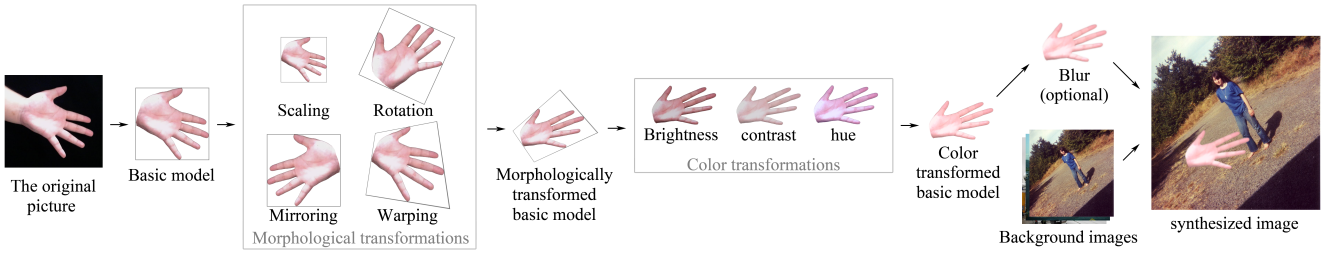

Figure 3. **Steps of generating datasets.** Firstly, we took photos for gestures and segmented them out as basic models. Then, various augmentation methods were applied on them to obtain the augmented models. Finally, we synthesized them with background images.

Firstly, for each gesture, we took exactly 30 photos from different angles, ensuring the hand in each photo was clear and complete. The hands were against a black board so that we could easily remove the background afterwards. We also slightly changed hand appearances (like angles between fingers) during the process to make the models more general.

Next, we extracted the hand of each captured image, by segmenting the hand (without wrist) from the black background, see Figure 3. This can be done with some very simple tools, such as the magic wand tool in Photoshop; and it was the last part that needed human participation.

3.2. Augmenting and synthesizing

As shown in Figure 3, the augmenting process is to largely expand the basic model set, in order to diversify the morphology and color of the basic models. It is committed to representing hands of different individuals in different conditions as many as possible. The synthesizing process is to select an appropriate background set, which is large enough and has various scenes, and synthesize the augmented models with the background set to obtain the generated dataset.

The augmented models are obtained in accordance with the following rules:

- Morphological transformations of the basic models, including appropriate scaling, mirroring, rotation, and further warping, that is, randomly adjusting the four corner points of the model image in a given range, so that the model looks distorted (see Figure 3).
- Color transformations of the basic models, including brightness, contrast and hue adjustment, blur processing.

Then, select an appropriate background set, and synthesize it with augmented models. The following are rules:

- The background set should contain large quantities of images, covering a variety of scenes. We chose the training set of VOC2012 as background set, which has 17,125 images. Notice that we only use the images of the VOC2012 training set.
- For each image in the background set, select at least one model in the augmented models and randomly place it (them) on the image, ensuring the models do not overlap or cross the image border. The number of placed models is randomly decided, and the upper bound depends on the size of the image.
- The ground truth labels and bounding boxes of the target objects are determined during the synthesizing process. As shown below, L_i represents the category of the selected model and G_i is the overlapped area between the model and the background image.

$$\begin{cases} Label = L_i \\ x_{min} = \min\limits_{(x,y)\in G_i} x \\ x_{max} = \max\limits_{(x,y)\in G_i} x \\ y_{min} = \min\limits_{(x,y)\in G_i} y \\ y_{max} = \max\limits_{(x,y)\in G_i} y \end{cases} \tag{1}$$

As a result, the number of models had greatly increased. These steps were proved to be necessary in the subsequent experiments. Some essential issues should be noticed in the augmenting and synthesizing procedure. One is warping and blur. Lack of warping would lead to overfitting of some models, such as Faster R-CNN. And the method of blur can help avoid overfitting, especially when the objects are not clear in the testing images. Another issue is multiple-hand problem. The models trained with images each containing just one hand worked terrible when facing multiple-hand images. Two hands were detected to be one when they were close. Hands within the same image also affected each other's classification result greatly.

The augmenting and synthesizing process can be done several times on the same basic models and background set to generate more training data.Based on 150 images, we repeated the procedure 6 times to generate 102,750 images containing about 154,000 hands. The scale is 685 times larger than the basic model dataset, and 21 times larger than

Figure 4. **Samples of generated dataset.** The augmented gesture models were randomly placed on background images. Some images contained more than one gesture.

the Oxford's hand dataset. Some samples are shown in Figure 4.

3.3. Rethinking generated dataset

Our dataset generation method has three advantages. First, the diversity of both objects and backgrounds is guaranteed, which is critical for training applicable models that can be used in our real life. Besides, the augmentation of basic models enriches the variation of target objects and prevents overfitting. Second, compared to human labeling, the ground truth labels and bounding boxes are expected to be more accurate, for most people do not follow a strict standard when labeling. Third, it eliminates the need of manual labeling, which means it may save a lot of time and money. Many large scale datasets are collected commercially and take a few months to complete , yet we artificially generated our 102,750-picture training dataset in a single afternoon.

Furthermore, according to the current trend, technology in the future will greatly depend on deep learning, which means that complicated neural networks are already built and proved to be effective, but we still have to provide enough data to train them according to specific requirements. If all datasets rely on collecting real photos and manual labeling, it would be inefficient and unrealistic. As we recall the situations when parents teach their children to recognize objects, the material provided may not be real examples. They may give their children a model, like a standard picture, or a toy. We believe that the training of deep neural networks can imitate this idea, to use generated datasets, which provide core characteristics of the target objects, as training sources. The innovations in this field will greatly advance deep learning and broaden its applications.

4. System

To achieve dynamic gesture recognition, we designed a system mainly made up with two parts. Sec. 4.1 introduces the gesture detection model based on YOLO, which is the first part of our system; and explains how we modified and trained it. Sec. 4.2 describes the second part, which extracts hand trajectories and classifies dynamic gestures. Once a specific trajectory is extracted, it goes to a classification model which can handle spatio-temporal series. We left this task to a discrete HMM.

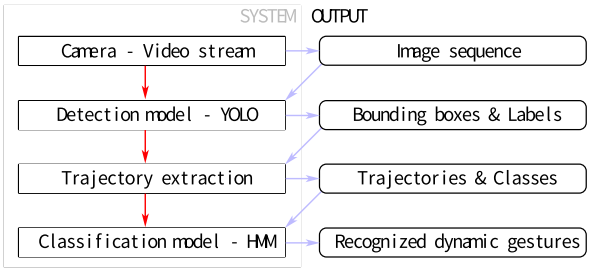

Figure 5. **Overview of dynamic gesture recognition system.** The detection model can locate and classify the gestures in each frame of the image sequence. And the gesture location sequences formed the trajectories. They were put into the classification model to recognize dynamic gestures.

4.1. Gesture detection

Detection model. We implemented our gesture detection model based on YOLO. As shown in Figure 5, it served for detecting gestures in images accurately and in real-time. The essential idea of YOLO is to use the entire image as network input and directly regress the positions and categories of bounding boxes in the output layer. Rather than region proposal, it divides the image into a grid, and each cell predicts its corresponding area. Such structure is faster than other structures for object detection, like Faster R-CNN, and can meet the requirement of real-time detection based on video stream. But this method trades accuracy for time, which means it may be unable to predict positions accurately, especially for small objects. Considering that hands are relatively small in the images, we modified the original loss function of YOLO to make it tend to predict small bounding boxes, which can reduce localization error for gestures:

$$
\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left[(\omega_i - \hat{\omega}_i)^2 + \left(h_i - \hat{h}_i \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 \\
&+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{I}_{ij}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \qquad (2)
\end{aligned}
$$

Training. We used our generated gesture dataset for training, which was described explicitly in Sec. 3. Our YOLO model used 77 grid with each cell providing 3 boxes.

4.2. Dynamic gesture recognition

Trajectory extraction. When applying the previous model in successive frames, the output bounding box and category sequence can be obtained. We used 20 FPS video frames as input, and the output came at 21.83 FPS in average. The processing speed is limited by our GPU, yet the present speed is fine. Then hand trajectories were extracted frame by frame. First, make valid bounding boxes in the first frame into disjoint sets.

$$D_i = \{B_i\}, \text{if Confidence}(B_i) > 0.1 \tag{3}$$

Merge sets D_m and D_n if

$$\begin{aligned} &B_i \in D_m, B_j \in D_n \\ &IoU(B_i, B_j) < 0.1 \end{aligned} \tag{4}$$

After each merging, eliminate $B_i \in D_m$ where there is

$$\begin{aligned} &B_j \in D_m, i \neq j, Frame(B_i) = Frame(B_j) \\ &\max_{B_k \in D_m, k \neq i,j} IoU(B_i, B_k) \leqslant \max_{B_k \in D_m, k \neq i,j} IoU(B_j, B_k) \end{aligned} \tag{5}$$

Then, every time when the model finishes processing the next frame, make output bounding boxes disjoint sets according to (3), and merge sets including previous ones according to (4) and

$$|Frame(B_i) - Frame(B_j)| < 10 \tag{6}$$

After each merging, eliminate boxes according to (5). Notice that after processing each frame, the remaining disjoint sets suffice

$$\begin{aligned} &\text{if } B_i \in D_m, B_j \in D_m, i \neq j \\ &\text{then } Frame(B_i) \neq Frame(B_j) \\ &\text{or } Frame(B_i) = Frame(B_j) = \min_{B_k \in D_m} Frame(B_k) \end{aligned} \tag{7}$$

consider D_m as a valid hand trajectory if

$$\max_{B_k \in D_m} Frame(B_k) - \min_{B_k \in D_m} Frame(B_k) > 20 \tag{8}$$

For each detected trajectory, the final output is two sequences, one containing hand moving directions (discretized into 16 directions) and the other containing corresponding categories. The sequence should be arranged in chronological order.

Trajectory classification model. As for the trajectory recognition problem, we used a discrete Hidden Markov Model (HMM) [4]. HMM is a statistical model used to describe a Markov process with implicit unknown parameters. The moving direction sequences were put into our HMM and classified into basic shapes. Together with category sequences, which can be processed with simple statistical methods, it is easy to classify dynamic gestures.

Training. Recalling the aforementioned example of parents teaching children, our training set for HMM is just like parents drawing pictures to show their children the specific objects. We developed a program to capture cursor movements on the computer screen, and we then used our mouse to draw circles, squares, and so on, to create the training dataset, as shown in Figure 6. The collected cursor tracks were used to train the HMM, which was defined to have 16 observations (moving directions) and 64 states to get good results. What's more, the testing gestures in our experiments are shown in Figure 7. Double-handed gestures can be represented as two basic shapes or simple lines.

5. Experiments

In the experimental section, we verified the methods and system presented above. First, we used Oxford's hand dataset and our generated gesture dataset to train the CNNs. Since the Oxford's dataset was not classified, we used it to

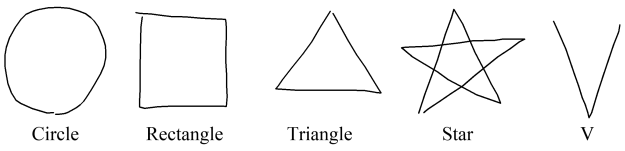

Figure 6. **Basic shapes for training.** We drew more random versions of them by mouse and used them for HMM training.

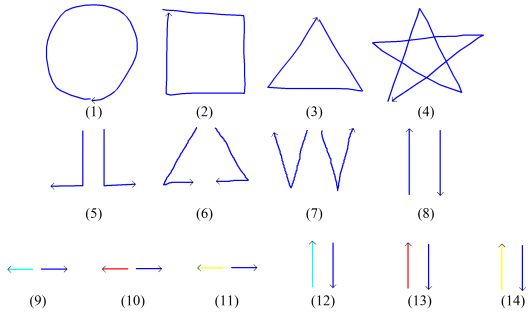

Figure 7. **Dynamic gestures for training.** Moving directions are marked with arrows; in most cases, directions do not matter. The first row contains single-handed gestures; the second row contains double-handed gestures; the third row contains double-handed gestures with different static gestures, where Palm is marked blue; Flat is cyan; One is red; Two is yellow.

Figure 8. **Testing sets.** The Low (Low Quality) testing set images were captured by camera with resolution 640×360. The images have more noise and the objects have no clear boundaries. Images of other sets were captured by camera with resolution 1280×720. Indoor and Low had a lamp as light source, yet light in Restaurant was dim and yellowish.

train a hand detector. Our dataset had all categories considered as one class to fit the experiment. Then, based on our training set, we compared our selected gesture detection models. We trained several state-of-the-art models and evaluated their performance by comparing their Average Precision (AP) and mapping the Precision-Recall Curve (PRC). Lastly, the dynamic gesture recognition system was tested by a number of videos recorded in different scenes, each containing all predefined dynamic gestures. The tests were all performed on an NVidia Tesla K80 GPU.

5.1. Comparison of different training sets

We used datasets different from the training set to do the test, as shown in Figure 8. The images of our testing set came from videos recorded in real-world scenarios, with complex backgrounds and varying light conditions.

Some of the previous works had training and testing done on the similar dataset. It was not convincing for the model could overfit and become useless when being applied in reality. In contrast, we used our generated dataset for training, and recorded real-world videos for testing. If the model has a good performance in this training-testing mode, we can confirm that it is robust, able to be applied into a wide range of recognition applications. Specifically, there were four testing sets, as shown in Figure 8. Each testing set contains more than 400 images, sampled evenly from the corresponding video. Actions in each video contained the five gestures and their combinations. For each gesture, the angle of hands and the distance towards camera were in great diversity.

Then we tested our own generated dataset and Oxford's dataset. They were used to train the original YOLO and faster R-CNN for hand detection task, which means that there are only two classes (hand and background) in the models and they aimed at detecting the hands in images.

We mainly used the Average Precision (AP) to evaluate the models. According to the Table 1, we can find that, the Oxford's hand dataset can train the models capable of detecting hands as it was described in [3]. In comparison, the overall performance of models trained by our generated gesture dataset turned out to be better. It indicated that the generated gesture dataset worked out well for gesture detection and when handling real-life conditions, it was more effective and useful for the training of models.

	Train	Low	Outdoor
YOLO	Oxford	81.6	72.8
	Generated	97.6	97.1
Faster R-CNN ZF	Oxford	76.4	76.1
	Generated	96.4	96.8
Faster R-CNN VGG-M	Oxford	68.4	62.2
	Generated	96.3	95.1

Table 1. **Evaluation on different datasets.** The AP for hand detection of each test set was listed. And the overall results indicated that models trained by the generated gesture dataset worked out well and more effective for gesture detection in real-life conditions.

5.2. Comparison of different gesture recognition models

After verifying the reliability of our own dataset, we undertook experiments to compare our modified YOLO model with other state-of-the-art models in gesture detection. They were all trained with our generated dataset.

Table 2 shows the mAP, AP_1 and FPS of each model for different testing sets. By comparing the mAP, we could find that YOLO is a suitable model for gesture recognition. It performed well on Indoor, Low and Outdoor, but exposed drawbacks on Restaurant. However, our modified YOLO showed overall high performance on the four testing sets. Compared to original YOLO, it was improved on Low and Restaurant while maintaining almost the same mAP on Indoor and Outdoor. Its improvement on Restaurant exceeded 10 percentage. SSD500 was the best in AP_1 on Restaurant, yet its accuracy or speed cannot outperform Faster R-CNN. Both Faster R-CNN and SSD500 were not real-time detectors when evaluating the FPS. Due to its simplicity, YOLO detected at a real-time speed; and it had such high mAP, which is amazing.

We also compared the mAP with the AP_1 of each model. The AP_1 was always adequately high, and higher than the mAP, especially on Restaurant testing set, which indicated that the models can perform well when detecting gestures and providing bounding boxes, but sometimes had trouble in classifying gestures properly. Actually our system can tolerate a lower mAP if the AP_1 is high enough, because when extracting the trajectories of dynamic gestures, the localization precision of gestures would greatly influence the trajectory, but several misclassified gestures did not bother much.

Furthermore, we plotted the Precision-Recall Curve (PRC) of YOLO and YOLO Modified for Restaurant testing set in detecting the five different gestures, as shown in Figure 9. Restaurant is almost the hardest testing set for

	Indoor		Low		Outdoor		Restaurant		
	mAP	AP_1	mAP	AP_1	mAP	AP_1	mAP	AP_1	FPS
YOLO	96.2	96.6	92.4	96.2	97.3	97.9	77.4	86.4	21.83
YOLO Modified	95.7	97.3	96.8	97.8	98.5	99.0	88.4	94.9	21.83
Faster R-CNN ZF	91.2	93.4	85.4	89.7	98.0	97.6	81.8	93.9	11.9
Faster R-CNN VGG-M	92.2	94.4	82.2	89.3	96.5	97.1	85.0	94.0	10.2
SSD500	89.5	90.7	81.0	86.3	95.8	93.9	84.1	97.4	6.9

Table 2. **Performance of different gesture detection models.** The mAP is mean of AP for each class. And for AP_1, it represented the AP for the combined class in which we regarded the five gesture classes as one class (thus classification error is ignored). We can find that YOLO and Faster R-CNN perform better than SSD500, but Faster R-CNN was not real-time detector. And AP_1 was always high, which means the localization precision of models were pretty good when ignoring the task classification.

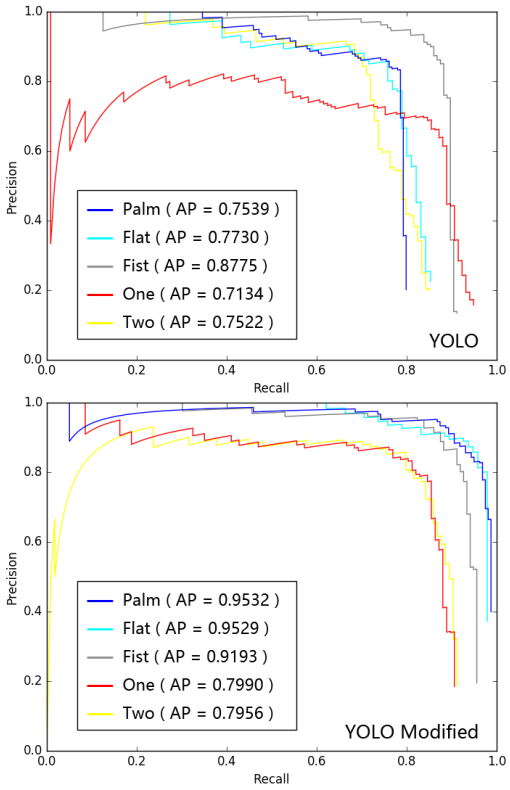

Figure 9. **Precision-Recall Curve (PRC) of YOLO for Indoor test set.** The improvement of YOLO Modified is obvious. The mAP for Palm and Flat increases dramatically, and the mAP for the other three classes also improves.

every model, for its dim, yellowish light and severe motion blur. The curves clearly revealed the differences among different gestures and the improvement of YOLO's performance. The One and Two class were relatively difficult for each model. YOLO Modified already had adequate mAP of around or more than 80 percentage. Meanwhile, It can be observed that AP, as a single number, could not tell the whole story. If we only pay attention to the part of curves in a relatively high recall rate (higher than 0.8), the precision of detecting One is better than Palm, while the Palm's AP was greater than One's.

5.3. The performance of dynamic gesture recognition system

To test our system, multiple videos were recorded, which contained all dynamic gestures we designed. In these videos, we defined fist to be the base, which was ignored when detecting hands. The other gestures were called visible gestures. We considered the start of each dynamic gesture to be the time when visible gestures appeared, and the end to be the time when visible gestures vanished.

We compared the outputs of our system and manual annotations, as some examples depicted in Figure 10 and Figure 11. The system was able to track hands accurately and fix minor problems such as losing or wrong bounding boxes in a few frames. Moreover, since we knew exactly where the hands were with the help of our gesture detection model, the system could be easily modified to do more things other than dynamic gesture recognition, such as human-computer interaction. The mean delay of the dynamic gesture recognition process was 2.24ms, which was trivial and did not affect detection FPS, since we applied pipeline.

6. Conclusion

In this paper, we managed to develop a real-time dynamic gesture recognition system based on gesture detection and trajectory recognition. Meanwhile, we proposed a novel method to generate dataset artificially by augmenting and synthesizing methods. We used this method to generate our gesture dataset and train our models. Then we recoded testing sets totally different from the training set to make evaluations. Through many comparing experiments, the generated dataset was validated applicable and our system turned out to work out robustly in complex light conditions and backgrounds.

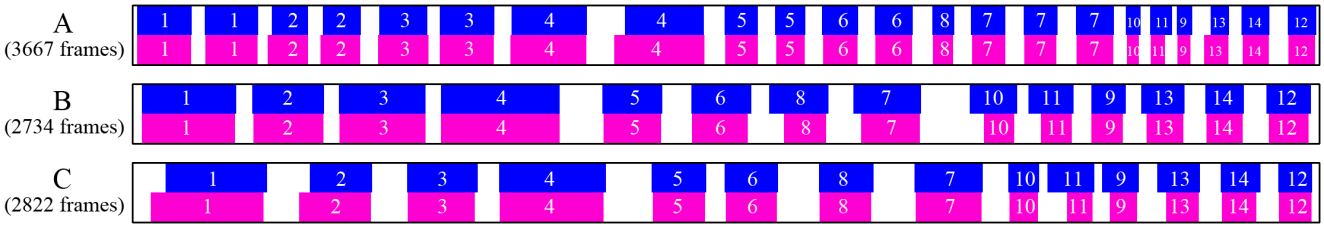

Figure 10. **Comparison examples of dynamic gesture recognition.** Each horizontal tape shows the recognized gestures through frames (time). The upper blue regions are dynamic gestures detected by the system, and the lower purple regions are manually annotated. Numbers in the regions show their category, according to the index in Figure 7. It is evident that our system had a perfect accuracy in classification, while existing errors in start and end frames. Only the end frames matter considering response delays, which our system matched well.

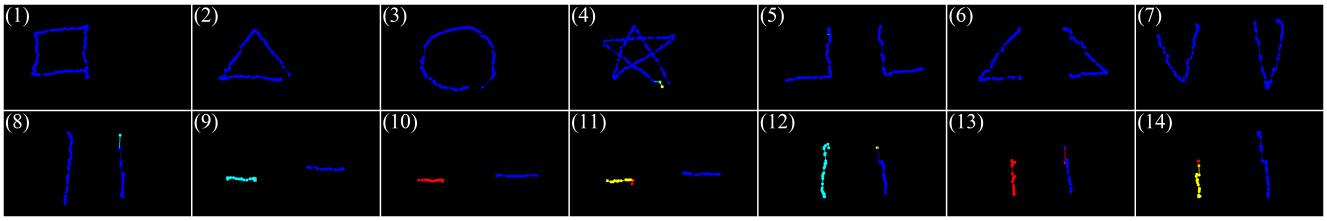

Figure 11. **Trajectories detected by the system.** The meanings of shapes and colors are the same as Figure 7. The trajectories clearly show the gesture and movement of the hands, while existing trivial errors which had not been filtered.

References

[1] F. Althoff, R. Lindl, L. Walchshusl, F. Althoff, and R. Lindl. Robust multimodal hand-and head gesture recognition for controlling automotive infotainment systems. 2008. 2

[2] D. Annane, J. C. Chevrolet, S. Chevret, and J. C. Raphal. Two-stream convolutional networks for action recognition in videos. *Advances in Neural Information Processing Systems*, 1(4):568–576, 2014. 2

[3] O. D. Cortazar, A. Megia-Macias, and A. Vizcaino-De-Julian. Hand detection using multiple proposals. In *British Machine Vision Conference*, pages 75.1–75.11, 2011. 2, 6

[4] B. T. Kanungo. Umdhmm: A hidden markov model toolkit. In *In Proceedings of the*, 2010. 5

[5] R. Z. Khan. Comparative study of hand gesture recognition system. 2(3):203–213, 2012. 2

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25(2):2012, 2012. 2

[7] S. Marcel. Hand posture recognition in a body-face centered space. *Lecture Notes in Computer Science*, 1739:97–100, 1999. 2

[8] I. Masi, A. T. Trn, T. Hassner, J. T. Leksut, and G. Medioni. Do we really need to collect millions of faces for effective face recognition? 2016. 2

[9] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand gesture recognition with 3d convolutional neural networks. In *Computer Vision and Pattern Recognition Workshops*, 2015. 1, 2

[10] E. Ohn-Bar and M. M. Trivedi. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2368–2377, 2014. 2

[11] O. K. Oyedotun and A. Khashman. Deep learning in vision-based static hand gesture recognition. *Neural Computing and Applications*, pages 1–11, 2016. 2

[12] L. Pigou, S. Dieleman, P. J. Kindermans, and B. Schrauwen. *Sign Language Recognition Using Convolutional Neural Networks*. 2015. 2

[13] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106–1113, 2014. 1, 2

[14] A. Raysarkar, G. Sanyal, and S. Majumder. Hand gesture recognition systems: A survey. *International Journal of Computer Applications*, 71(15):25–37, 2013. 2

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *Computer Science*, 2016. 1

[16] A. Rozantsev, V. Lepetit, and P. Fua. On rendering synthetic images for training an object detector. *Computer Vision and Image Understanding*, 137:24–37, 2015. 2

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1

[18] S. Shin and W. Sung. Dynamic hand gesture recognition for wearable devices with low complexity recurrent neural networks. 2016. 2

[19] J. Triesch and V. D. M. Christoph. A system for person-independent hand posture recognition against complex back-

grounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1449–1453, 2001. 1

[20] J. Triesch and C. V. D. Malsburg. Robust classification of hand postures against complex backgrounds. In *International Conference on Automatic Face and Gesture Recognition*, pages 170–175, 1996. 1

[21] P. Trigueiros, F. Ribeiro, and L. P. Reis. *Hand Gesture Recognition System Based in Computer Vision and Machine Learning*. 2015. 1