How to Get the Best Group Photo

Changyeon Park
Dept. of Electrical and Computer Engineering
Seoul National University
blackco@snu.ac.kr

Chaehyun Jeong
Dept. of Computer Science and Engineering
Seoul National University
chjeong9727@snu.ac.kr

Yuri Jeon
Dept. of Computer Science and Engineering
Seoul National University
slsl3650@snu.ac.kr

Daniel Chew
Dept. of Computer Science and Engineering
National University of Singapore,
daniel.chew@u.nus.edu

Abstract

We present a group photo auto selection system. Our system consists of 3 modules; detecting movements, detecting eye-blinking, detecting hand poses. We receive 5+ continuously taken photos as input and rank them by our criteria. Detecting movements is achieved by dense optical flow, detecting eye-blinking is achieved by HoG face detection, face landmark detection and EAR(Eye Aspect Ratio) algorithm. Detecting hand poses is achieved by object localization and classification using deep learning. Detecting movements showed 90% accuracy, detecting eye-blinks showed 83% accuracy. In hand pose detection, hand localization showed 77% accuracy and pose detection showed 45% in average accuracy.

1. Introduction

When we take group photos, we often see someone who blinked his/her eyes, someone who is blurred because they move at that moment or someone who didn't do the promised motion (e.g. making a V sign). These types of people spoil a nice group photo. However, it is bothersome to find the best group photo by checking such people manually among the series of photos. The purpose of our project is to find the best group photo among these series of photos. We will first apply optical flow to detect any movements in the photo, followed by applying eye blink and face detection, followed by hand localization, and finally, hand classification. The algorithm then excludes the photos which has blinking or blurred image and make a sensible recommendation based on the resulting photos.

2. Criteria for Best Group Photo and Input Assumption

2.1 Three Criteria for Best Group Photo

Before we can apply these algorithms to the series of photos, we must first define what are the attributes that makes a good group photo a good group photo. The three criteria that we came up with after extensive research is as follows.

(i) Everyone in the group photo are stationary.
(ii) Everyone in the group photo are non-blinking.
(iii) Everyone in the group photo take the same motion.

2.2 Input Assumption

Assume that enough input images are given, and the images are sequential group photos.

3. Background and related works

3.1 Optical Flow

Optical flow is a classical computer vision method for detecting movements. Open source in OpenCV [1] will be referenced. This source implements dense optical flow using the Gunnar Farneback's algorithm to find the velocity vectors of each pixel in series of images. The reason why we chose to use dense optical flow algorithm is because it gives us the flow all over the image while sparse optical flow (for e.g. the Lucas-Kanade algorithm) computes optical flow for a sparse feature set.

3.2 Eye Blink Detection

3.2.1 HoG Based Face Detection

HoG(Histogram of Oriented Gradients) is a feature based descriptor which is used for detecting objects. HoG Face Detection[2] is a face detection model based on HoG feature. In HoG version of an image, it finds the portion corresponding to the HOG face pattern generated from training lots of face image dataset, and the detected position is regarded as face.

3.2.2 Face Landmarks Detection of Dlib

Face landmarks detection of Dlib[3] produces 68 (x, y)-coordinates which represent key features of human face like eyes, eyebrows, nose, mouse and jaw. These landmarks are depicted in Figure 1. It was pre-trained using iBUG 300-W dataset.

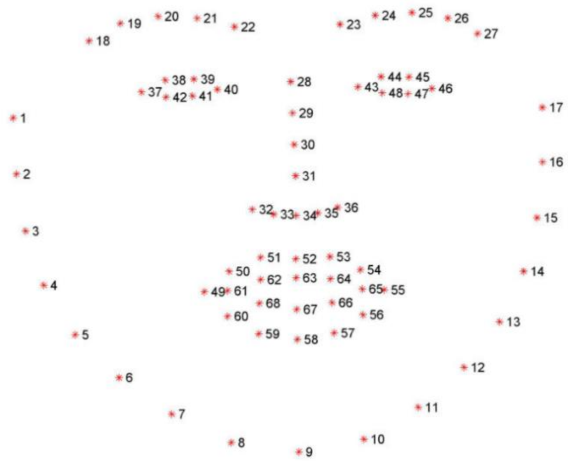
Figure 1 Face landmarks detection of Dlib produces 68 (x, y)-coordinates which represent key features of human face like eyes, eyebrows, nose, mouse and jaw.

3.2.3 EAR(Eye Aspect Ratio) Algorithm

We already know 6 points that represent each eye by using facial landmarks estimation. Using those points, we can calculate EAR(Eye Aspect Ratio)[4] value. It can be computed by dividing eye height by eye width as below equation. EAR can be used in expressing eye blink numerically.

$$EAR = \frac{|p_2 - p_6| + |p_3 - p_5|}{2|p_1 - p_4|}$$

3.4. Hand Pose Detection

Before the rise of deep learning, the approaches of hand detection could be classified into three categories.[5] First one is based on local appearance such as skin color or joints. The advent of depth camera became a help of this approach. By combining color information and the depth information that depth camera offers, researchers tried to reconstruct the skeleton of hands. The second approach relied on global appearance, such as using global hand template. The last one was motion-based detection, which used ego-motion of the camera and assuming hands and the background have different motion.

As deep learning is in the limelight, the major stream of hand detection changed. Exploiting depth map became mainstream of hand pose estimation with the deep learning flow. One of the most popular methods is to directly map the depth images to 3D pose parameters such as joint angles or 3D coordinates. Though many researchers are devoting on the task of hand pose estimation, this is regarded as very challenging area.

4. Methods and Experiment Results

4.1 Detecting Stationary Photos

By applying the dense optical flow algorithm to the series of group photos, we are able to calculate the maximum velocity vectors of each images, which we will then recommend the image with the minimum maximum velocity vectors among the series of photos as the chosen image shows relatively less movement compared to other images. For example, if we use the first image as a reference image, the minimum maximum velocity vector among these series of photos is Figure 2 as it has a maximum velocity vector of 28.2. We tested on 10 sequences of continuously taken photos. All the sequences outputted rank 1 as what we originally handpicked most stationary picture except one sequence. However, this sequence had our handpicked picture in rank 2.

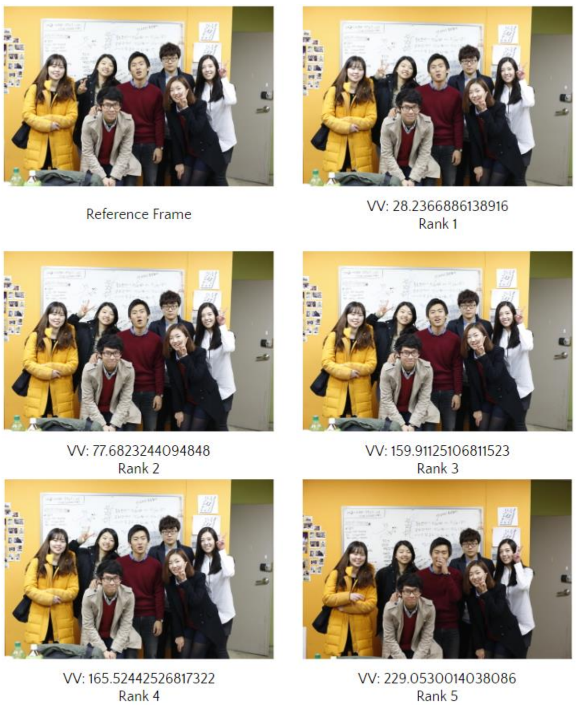

Figure 2 Detecting movements and ranking pictures with optical flow

4.2 Eye Blink Detection

Detecting eye blinks consists of 3 steps. First, detecting faces in an image, second, detecting landmarks of eyes in the face and lastly calculating EAR to find if the person is blinking or not.

4.2.1 Detect Faces in a Photo

Using HOG face detection, we detected faces that appeared in the photos. We choose HoG face detection because it is the fastest method on CPU among the face detections. We pass the detected faces to next step.

4.2.2 Detect Landmarks of Eyes

We applied facial landmark detection to localize the region which we are interested in. In this experiment, we only need information of the eyes, so we extracted 6 points among 68 facial landmark points. Therefore, we find out 6 landmarks per each eye in detected faces. We pass the (x,y)-coordinates of eye landmark points to the next steps.

4.2.3 Calculating EAR of Both Eyes

Using 6 landmarks of eye, we calculate EAR values. There are two eyes in each face, so we calculate the average of the EAR value of left eye and the EAR value of right eye. And then, we pass those values to the next step.

4.2.4 Set Proper Threshold

We determine that the eyes blinked if EAR value is below the threshold, which we arbitrarily set. What is the best threshold to detect blinked eyes? We tried some different threshold values to find best one. Below graph(Figure 3) is the result. As you can see, the best result is when the threshold is 0.21. Therefore, we set EAR threshold to 0.21 and continued our experiments.

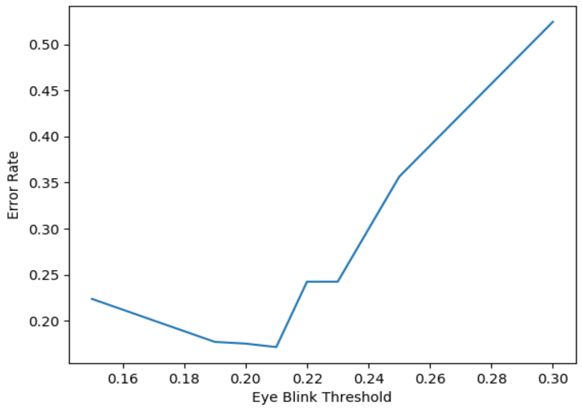

Figure 3 Error rate according to the eye blink threshold. We can find that error rate is minimum at 0.21.

4.3 Hand Pose Detection

4.3.1 Hand Localization

First, we used ssd-mobilenet object detection deep learning model to localize hand in a photo. We used SSD based MobileNet because it is famous for real-time evaluating and its capability to run on non-gpu conditions. We used pre-trained parameters from Tensorflow detection model zoo[6] to shorten our train time and trained them on EgoHands dataset.[7](Figure 4)

In 100 photos each containing 4 people and 8 hands, we localized total of 618 hands. Giving approximately 77.2% accuracy in localizing hands. After testing on group photos, we realized that group photos had hands slightly different from the training set which was EgoHands dataset. This might be the cause of low accuracy in localizing hands. We can expect better result if we use hand data from actual group photos.

Figure 4 EgoHands dataset

4.3.2 Building Learning Model

Among many state-of-the-art techniques detecting hand poses, we came up with transfer learning since hand image data in group photos are small and various according to hand shape, pose, and angle so we decided to use the advantage of transfer learning.

The majority of our model is inspired from MrEliptik's github repository[8]. We first preprocess the input image as 28x28x1 in gray scale. There are two convolutional layers and we chose its kernel size as 3, followed by a 2x2 max pooling. The activation function is chosen to be with ReLu. Finally, there comes a single 128 dense layer followed by a softmax layer to give the overall probability of classification.

4.3.3 Generating Dataset

Instead of using already existing dataset such as NYU hand pose dataset, we tried to generate our own. This is because we do not need very detailed information about hand for our task. The major feature in our raw input, group photo is that the hand image will be very small. It would be hard to extract some specific and detailed features such as location of finger joint from group photo as other datasets suggest. With regard of this, we determined to produce our own dataset and recorded short videos emphasizing hands of people with various age and gender. The models of the video twisted his hands little by little, in order to generate dataset from various angle and positions. We cropped the hand image from the video using our own hand localizer. Finally, we manually deleted some wrong cropped images in order to exclude the malfunctioning effect of our localizer. From this method, we could generate about 30 qualified hand images per 10 seconds-long short videos.

4.3.3 Evaluations and Issues

In order to evaluate the pure performance of classifier excluding from localizer, we cropped hand image from various group photo by hands. We used 20 sets of group photo with 20 serial images each. We applied our classifier on each of these cropped hand images and if one of the softmax probability is over threshold(0.7), the image is evaluated to take the specific pose.

Graph below(Figure 5) is the evaluation result of our classifier. You can denote that the prediction accuracy between poses are very different. It relatively works well with palm – its accuracy is close to 70%. The problem is in V pose. The performance of classifier even got worse when the number of training set increases. Moreover, the accuracy of rock pose is only about 50% - not high as our expectation. We found out that in many rock cases, the classifier does not give any answer over the threshold value – it produces similar values among all poses although it is definitely a rock. Another problem is that the graph of other pose shows no linear tendency with the training. In other words, even when people do not take any pose, the classifier tries to predict it and it gives wrong answer.

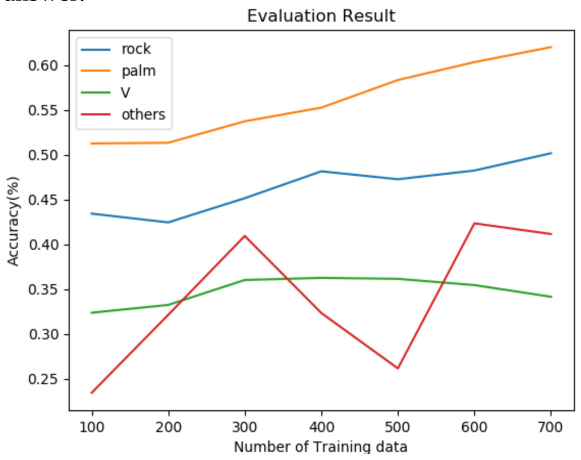

Figure 5 Accuracy test with number of training data. Test data is 100 images per pose

To resolve those problems, we first tried to control those cases by adjusting threshold value, but this did not work since increasing threshold generates more and more 'no answer' state. The solution we had come up with was to categorize garbage poses into another single class. When people do not take any pose, they usually put their hands by their side. We found out that those poses were most of the garbage poses since other unprepared images such as bent-fingered V are successfully removed at first stage when we detect optical flow. Thus, almost all annoying garbage poses look similar and it is possible to categorize them and train it as a single class.

When inspecting the pictures that our classifier gave wrong prediction, those were the pictures with colorful background. Therefore, we guessed that removing background of images before classifying would boost performance. We tried to segment the image to distinguish hand from background by applying watershed algorithm. What we attempted was 'marker-based watershed', which labels where the region is foreground or background, and gives different labels for the known objects. We tried to label hand as definite foreground object.[9]

Table 1 Accuracy test of 4 classes of poses with training variations

Pose Accuracy (%)	Original	Adding Garbage	Applying watershed
rock	50.14	54.73	34.71
palm	63.98	61.44	44.10
V	34.20	42.30	31.14
others	41.12	68.08	43.19

Adding garbage class helped the accuracy of rock, V pose and others incredibly. However, segmentation made our classifier even worse from first try. This is because segmentation does not work well when the hand is too close with the body or the color of background is very similar with skin color. Those are the examples of hand segmentation. The right one looks plausible while the left one (rock pose sticks with face) produces awkward segmentation.

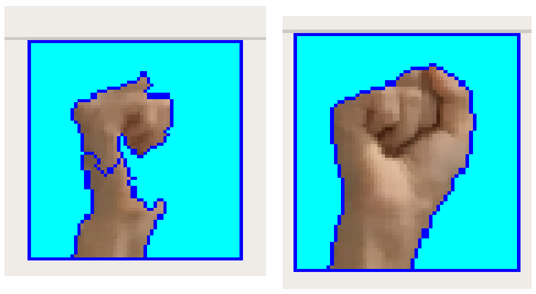

Figure 6 Applying skin hand segmentation by skin color

5. Conclusion

Our goal was to find the best group photo among the given images. With 3 methods we proposed, detecting movement in an image, detecting eye blinks, detecting specific hand pose, we could find the best group photo. However, there are room for improvements. For example, for eye blink detection, adjusting EAR algorithm to fit for different eye shapes and improve error rate. For hand localization, training actual hand data from group photos will improve accuracy in localizing hands in an image. For hand classification, collecting more hand pose data could improve accuracy. Also, hand localization and hand classification can be merged into one object detection algorithm.

6. References

[1] OpenCV Dense Optical Flow https://docs.opencv.org/3.4/dc/d6b/group__video__track.html#ga5d10ebbd59fe09c5f650289ec0ece5af

[2] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, 2005.

[3] Kazemi and Sullivan, One millisecond face alignment with an ensemble of regression trees, 2014.

[4] Tereza Soukupov´a and JanˇCech, Real-Time Eye Blink Detection using Facial Landmarks, 2016.

[5] C. Li and K. M. Kitani. Pixel-level hand detection in egocentric videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3570– 3577, 2013.

[6] Tensorflow detection model zoo https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

[7] EgoHands dataset http://vision.soic.indiana.edu/projects/egohands/

[8] MrEliptik's github repository https://github.com/MrEliptik/HandPose

[9] https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Watershed_Algorithm_Marker_Based_Segmentation.php