Silhouette-based Pose Estimation driven by Adaptive Hand Segmentation

Author:
Liam Carroll

Abstract

By continuously sampling the skin tone of the face we develop a calibration range that prepares our application for diverse environments. Applying different approximation and maximization functions we can obtain a segmentation mask of the hand which parallels the input format of our convolutional neural net (CNN). We used a fairly simple learning model to reduce processing speeds, though deploying a more complex model may result in higher accuracy. Adjusting sensitivity thresholds and continuously sampling allows our application to remain versatile, enabling further implementation with personal devices.

Introduction

Throughout the last decade, the intimacy in which we connect with the digital world has continued to accelerate. While technology as a whole has offered a new pinnacle of convenience it falls short in accessibility. Learning-enhanced gesture estimation offers a hands-free interface with a device or smart home. Regardless of whether you predominantly speak through sign language–or don't want to get off the couch the field of pose/gesture estimation and identification offers newfound conveniences to a wide range of communities. Deployment of gesture estimation means users will not be directly tethered to a device and are free to move around their environment.

Using samples of skin from the face assists us in identifying the subtle or dramatic ranges in hue, saturation, and value(HSV). We can use this information to segment hands from the background (and the rest of the objects) in the image. As

deployed by Mo and Neumann we used a fairly simple pose learning model to reduce processing speeds. By building out simple applications now and making them accessible – we open the door for a new wave of quality-of-life improvements. Without adaptive applications, all but those in perfect conditions are left without the ability to implement this technology. In such a collaborative and open-source field this case would be considered a serious net loss.

Related Work

We used a hand pose dataset consisting of ten classes: [call *me, rock* on, fingers crossed, okay, paper, peace, rock, scissor, thumbs, up] each consisting of about 500 images [1]. While the NUS and NYU datasets are more comprehensive both require better video capture equipment in order to input efficiently.

Mittal Zisserman and Torr also identified hands in accordance with the face but did not directly use skin tone and instead used the face to reduce the region of interest, similar to many eye recognition techniques [4].

Methodology

Segmentation and Preprocessing

We wagered the open-source computer vision library, OpenCV, to preprocess our frames in order to best align them with the input of the model [3]. We define two skin tones, an upper and lower bound as defined by our faceToSkinTone(FTST) method. The FTST can be used simultaneously with the pose prediction model or separately(recommended) to predefine the user's average skin tone in a given environment.

Using the Haar cascades face identification model(HCF) offered by OpenCV we approximate a region of interest around the face [2]. Two adjustable parameters are offered to further improve this region's accuracy. Through averaging the pixels in the adjusted region we deduce a mean value of the user's skin tone. Once we have identified the mean skin tone we create the upper and lower bounds by increasing or reducing its

HSV values by a saturation, hue, and value threshold. While predefined these thresholds can be further tuned to fit environments (ex. hard shadows in the environment – increase *value* threshold).

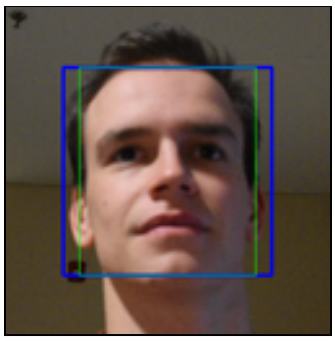


Figure 1. Blue: face region from pre-trained HCF
Green: custom parameters to better fit

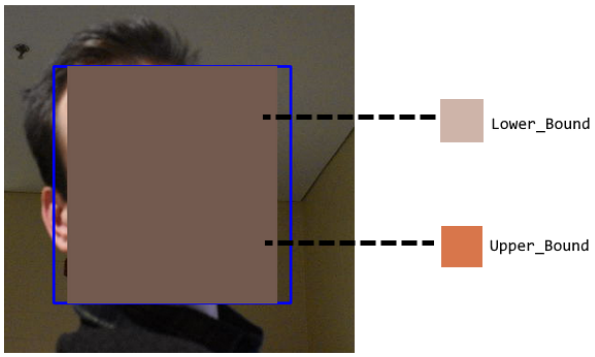


Figure 2. Tone bounds as defined by region average

Using the skin tone range defined in the FTST method we segment the hand from a new region of interest. We accomplish this by first masking all pixels within our defined upper and lower bounds. As depicted in figure 3 this initial mask is still fairly noisy. In order to reduce the noise, we apply several functions. First, we identify the contours around each white segment of the mask.

Figure 3. Open CV Contours

Using these contours we can deduce the collection of points that are connected around each edge. With this information, we can treat the points as boundaries and compare the inner area as demonstrated in the red of figure 3. As the hand will be closest to the camera and have the highest representation of the skin color we return the contour (collection of points) with the highest area.

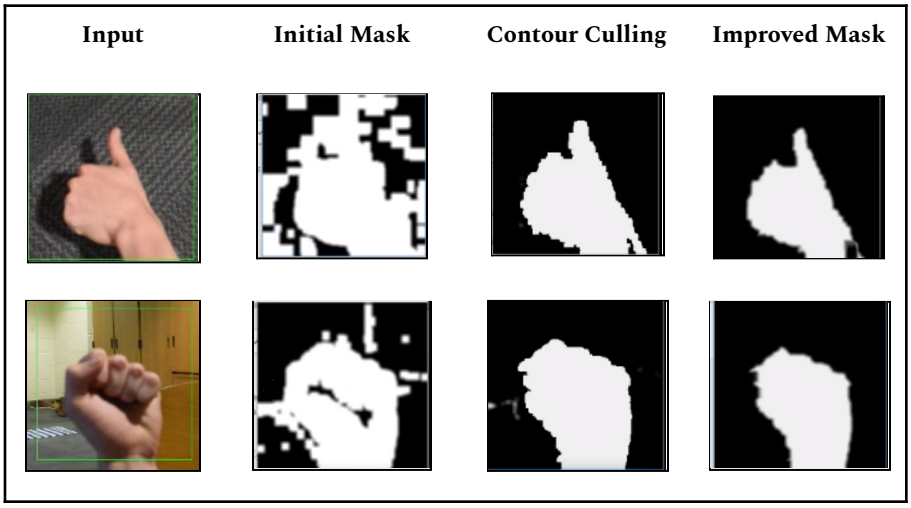


Figure 4. Steps of hand segmentation

We then apply a gaussian blur on our current approximation and erode the output. Blurring helps remove additional artifacts by reducing their pixel value as surrounding pixels are black. The erosion serves two purposes: eliminate the blurred noise and counteract the dilatation necessary to properly approximate the contour. As a result, we have an improved mask that can be fed into our model for prediction.

Convulutional Neural Net (CNN)

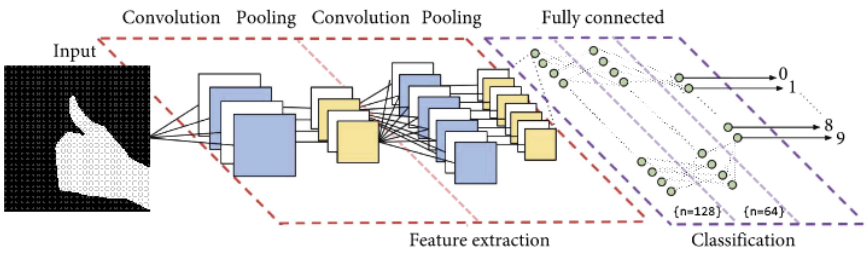


Figure 5. CNN and its layers. Image inspired and adapted from Vargas et al. [7]

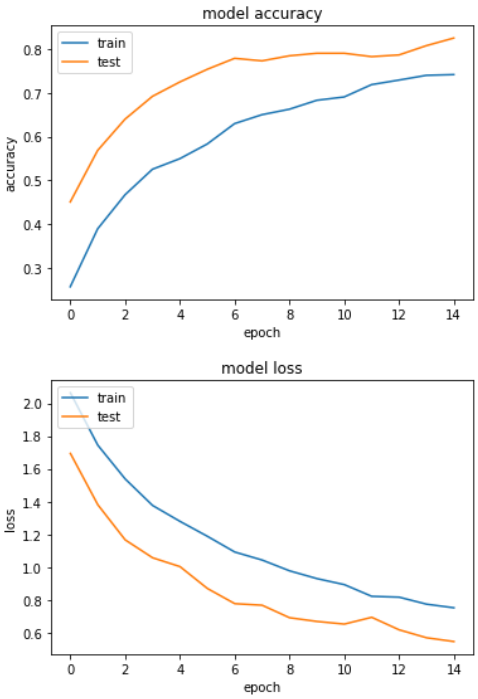
Our CNN model architecture is defined by an initial two convolution layers with max-pooling layers with a kernel size of five. Pooling downsamples the input image in order to keep only the significant details and remove noise. These are then flattened and passed to two fully connected layers, each with thirty percent dropout layers that are tasked with classification. These dropout layers help prevent overfitting by randomly setting a portion of the values of the nodes to zero. The final dropout layer is connected to the output layer utilizing a softmax activation function. The basic structure can be

visualized in Figure 5. The hidden layers are of dimensions 128 and 64 respectively. Optimization was done using the Adam optimizer, with a 1e-4 learning rate.

Experimentation and Results

After 15 epochs our model reached 83.3% testing accuracy across ten classes. The model's historical accuracy and loss are displayed in figure 6. Note, the train set is underperforming compared to the test set. This is not a failure in the model but instead an expected behavior as we implemented two-thirty percent dropout layers. These dropout layers, keeping the overall layer and hidden dimension complexity low all showed to significantly improve the model's overall accuracy in real-world tests.

Figure 6. Model accuracy & loss plotted against epochs



Since we trained our model on two-dimensional images we run predictions every frame. In order to ensure we meet our confidence threshold, we sample the highest confidence over a customizable number of frames (default of thirty). If the average confidence of the most predicted class is above our confidence threshold we count that as a success and display it to the user.

Pose Label	Video Capture, approx. 7200 samples per (%)
Call me	88.21
Fingers crossed	76.46
Okay	68.97
Peace	85.25
Paper	91.74
Rock	90.56
Scissors	88.42
Thumbs up	82.02
Up	90.52
Rock on	82.67

Figure 7. Testing with video capture implementation

Our dataset was a collection of binary images. Without a predefined validation or test set, we had to create one. To do this we excluded twenty percent of each class of the dataset and composed the validation set from our main dataset which drew twenty percent out of the main set. To further improve the variance in our sets we added a random possibility of shearing at twenty percent (about x or y-axis), zooming twenty percent, rotating 30°, and horizontally flipping the image. All images in the set have

between zero(unchanged) and all degrees of alteration within these predefined thresholds.

As indicated by figure 7 testing our model against video capture proved very successful. We tested in a dimly lit environment after calibrating and logged predictions per thirty samples, thirty minutes of predictions were sampled.

Limitations and Future Work

Initially approached by training the head of the re-trained SqueezeNet model as suggested by Forresst et al. we sought to increase the deployability of our application. Unfortunately, it resulted in overfitting with high confidence [6]. We also encountered overfitting in several other datasets.

In order to truly create an adaptive application fixed regions of interest are non-viable solutions. The major limitation of this work is that we used a static hand region of interest. In future works, we will likely deploy a pre-trained model to first predict the location of the hand, then run through our segmentation process and predict its posture.

Figure 8: Sobel contour edge detection.

We also limit faces in frames to one. This means multiple *accurate* skin samplings are impossible. Deploying the aforementioned approach to identifying dynamic hand regions of interest would mean that we could run multiple posture predictions at the same time but without accurate skin tone bounds, the earlier stage of segmentation fails. While currently based on user parameters the adaptability of the tighter face region could also be improved through Sobel edge detection. Using the edges identified we could improve our precision in defining the boundaries of the skin sample area as depicted in figure 9. To further improve skin tone accuracy we could weight samples from the region depending on their likelihood to contain skin. With a weighted average akin to figure 8 our bound tightness may increase, further reducing noise in our segmentation mask.

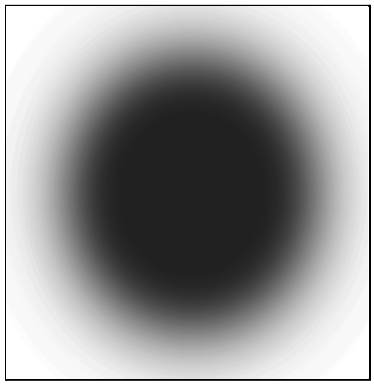


Figure 9: pixel weights by distance from nose.

We also hope to continue this work in a temporally dynamic space. Capturing motion and posture would open up possibilities of customizable gestures through the combination of a recognizable pose and a movement.

Brief Demo: https://i.gyazo.com/fea2777d14d9244f11b8c0f6a6ad3072.mp4

Works Cited

1. Rooban Sappani. “Hand Gesture Recognition” *Kaggle.* Retrieved December 2021 From https://www.kaggle.com/roobansappani/hand-gesture-recognition. 2021.

2. Bradski, G. “The OpenCV Library”. *Dr. Dobb's Journal of Software Tools*. 2000.

3. Paul Viola. Rapid Object Detection using a Boosted Cascade of Simple Features. ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION 2001. 2001

4. Mittal, Arpit, Andrew Zisserman, and Philip HS Torr. "Hand detection using multiple proposals." *Bmvc*. Vol. 2. November 3 2011.

5. Mo, Zhenyao, and Ulrich Neumann. "Real-time hand pose recognition using low-resolution depth images." *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE, 2006.

6. Forrest N. Iandola, Song Han, Matthew W. Moskewicz, et all. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size in Cornell Computer Vision and Pattern Recognition”, 24 February 2016.

7. Jianhong Chang, Jinzhuang Xiao, Jin Chai and Zhen Zhou. “An Improved Faster R-CNN Algorithm for Gesture Recognition in Human-Robot Interaction” *Conference: 2019 Chinese Automation Congress (CAC).* 2019.

8. A. C. G. Vargas, A. M. P. Carvalho, and C. N. Vasconcelos, “Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres,” in *Proceedings of the SIBGRAPI—Conference on Graphics, Patterns and Images*, Sao Paulo, Brazil, October 2016.