AUTOMATIC FACE ROTATION: IMPROVED DISPLAY ORIENTATION FOR HANDHELD DEVICES USING CONVOLUTIONAL NEURAL NETWORKS

**Riley Tallman**

Arizona State University

rtallman@asu.edu

ABSTRACT

We propose an improved method of rotating smartphones, tablets, and other device displays with Automatic Face Rotation using a convolutional neural network (CNN). Many users of smartphones and tablets experience a problem with their device screen being oriented the wrong way during use. This paper introduces a new algorithm to fix that issue by correctly orienting the screen relative to the user’s face using a CNN. The CNN model is trained to predict the rotation of faces in a variety of environments. The algorithm uses a confidence threshold and analyzes multiple images to be robust. Our solution solves the existing rotation problem, is battery and CPU efficient, and causes no noticeable lag to the user during use.

1 INTRODUCTION

The current basic and nearly universal method to orient handheld displays like on smartphones and tablets is to use gravity. Unfortunately, this method fails when a user lies down to one side, or when the device is flat as figure 1(a) illustrates. In a survey of 513 smartphone and tablet users from 2012, 91% reported that they have experienced incorrect autorotation with their devices, and 42% answered that they experience it several times a week or more [11]. To combat this issue, devices commonly have a native software feature to lock the current rotation state. For example, the original Apple iPad (released in 2010) had a hardware switch to lock and unlock the gravity-based rotation. This solution requires user input, and 58% of users forget to unlock the rotation [11]. The study validates a need for a solution that fixes the problems with gravity-based rotation.

We propose Automatic Face Rotation (Auto-Face Rotation, or Auto-Face) which correctly orients a display relative to the user’s face requiring no user input. Auto-Face introduces a new robust algorithm which analyzes multiple images from the front-facing camera using a VGG-style convolutional neural network. It solves the problem with the current solution using convolutional neural networks (CNNs). It can confidently detect the rotation that the user is using the device without introducing noticeable lag when rotating.

2 RELATED WORK

Our work is related to research in computer vision on mobile devices. It is most similar to the work of [11] who presented iRotate: an automatic screen rotation method by augmenting the gravity-based solution: when a rotation change is detected, run a facial detection algorithm and orient the screen based on the face, but fall back on gravity rotation if it fails. They did not create a custom CNN, and used a pre-built facial feature detection API from iPhone Operating System (iOS) 5. The API was not suited for the application, and the authors deemed it infeasible to use outside of informal experimentation. We propose a custom CNN and introduce a novel algorithm for analyzing multiple images.

Another camera-based approach attempted to rotate a display by tracking the head tilt of user’s faces [6]. Unfortunately, this suffered from a narrow range of degree tilt (about 47 degrees away from *portrait*) [7].

Gavity-based solutions are popular, and various types of sensors have been used including mercury switches [1] and one or more accelerometers [2, 3, 8]. These give incorrect rotations when a user lies down or when the device is flat, but offer a manual toggle to correct it. Our solution fixes these problems, and requires no user input.

Further methods use the way a user holds their phone to distinguish orientations. One solution explores the position of thumbs on the edges of the front screen [9], and another senses an entire grasp from the placement of fingers around the edges and the palm on the back [10].

3 AUTOMATIC FACE ROTATION

We present our automatic iRotate 2.0 algorithm which first uses gravity to detect rotation changes, then uses a CNN to verify correct orientation. It is possible to constantly use a CNN without gravity-based input, but it is drastically more efficient to use the two together as table 1 depicts.

Once a change is detected, the CNN begins analyzing images from the front-facing camera. After analysis, the CNN will either confirm that the gravity rotation is correct, or will override the gravity orientation (aka, perform a “gravity override”). The verification step is performed with a CNN that detects the user’s face and outputs what orientation the screen should be, relative to the user. If the CNN output is the same orientation that gravity outputs, then the program rotates to that orientation. If, however the CNN output does not match the gravity output, the CNN will override gravity and rotate to the correct orientation that is relative to the user if there is enough confidence in the CNN’s output. This case is depicted in the first sample output of Table 1.

**Confidence.** Misclassification may occur if an image is terribly blurry, dark, or bright, or if the user’s face is obstructed. To combat this, we define a confidence threshold to control gravity overrides, and analyze multiple images during analysis. In order for the CNN model to override the gravity orientation, it needs to reach a confidence greater than some threshold on two consecutive images that it analyzes. If the first two images fail to meet the confidence threshold, then the program will continue to analyze images from the front camera up to images.

**ImageQueue.** Auto-Face Rotation can analyze multiple images (the *imageQueue*) to ensure correctness. This technique is analogous to using multiple keys to authorize a military strike: all keys need to work in order to execute the operation. The parameter specifies how many images, or keys, to use in order to override gravity. The loop on line 5 verifies that all confidence outputs are above the threshold and all predictions are consistent (all the same orientation, i.e., *portrait* & *portrait*) within the *imageQueue*. Setting greater than 1 helps prevent misclassifications.

**Multiple Images.** To be robust, the algorithm can analyze more images than are in the *imageSet*. When the CNN is not confident or if the predictions are inconsistent, the process if repeated with one new image, and the first in the *imageQueue* is discarded. This creates a sliding window that shifts over one image in the camera feed up to times. Once a total of images are analyzed, Auto-Face returns “Low Confidence” and the CNN predictions are discarded. When this happens, whatever orientation that gravity has predicted will be used to rotate the display.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Class Confidence | | | | Result |
| portrait | upsidedown | landscapeLeft | landscapeRight |
| Gravity |  |  |  | 1 | OVERRIDE:  **portrait** |
| CNN | **0.93** | 0.05 | 0.01 | 0.01 |
| CNN | **0.93** | 0.06 | 0.01 | 0 |
| Gravity |  |  |  | **1** | Low Confidence:  **landscapeRight** |
| CNN | 0.54 | 0.23 | 0.11 | 0.12 |
| CNN | 0.56 | 0.20 | 0.01 | 0.13 |
| CNN | … |  |  |  |  |

Table 1: Two sample outputs and their resulting orientation. CNN [1] is the first analyzed image, and CNN (2) is the second. The top sample output is oriented to portrait because both images are above a confidence threshold of . All analyzed images in the second sample are below , so the output from gravity is used.

|  |
| --- |
| **Algorithm 1** CNN with Confidence Threshold and Multiple Image Testing |
| 1: *imageQueue =* analyzeNextImages  2: **do**: |
| 3: *imageQueue* = analyzeNextImages  4: *override* = true |
| 5: **for** *image* **in** *imageSet*:  6: **if** *image*.confidence < **and** isInconsistent(*image*.prediction): |
| 7: *override* = false  8: **if** *override*:  9: return ‘OVERRIDE’  10: *imageQueue*.pop() |
| 11: **while**  12: return ‘Low Confidence’ |

**Final Check.** Not show in the algorithm. Only two orientations are possible – gravity and previous. In the cases where the CNN has confidence but is incorrect but does not rotate correctly, we can avoid those cases by adding only 2 possible orientations.

4 CUSTOM CNN

We’ve created and trained a custom convolutional neural network to identify the face of the user and determine the correct orientation. We formulate the problem as a 4-class classification problem and train our model to classify the four possible orientations: *portrait*, *upsidedown*, *landscapeLeft*, and *landscapeRight*. Using a single input image, the CNN model outputs four numbers that sum to 1, each representing the confidence of the image belonging to its class (See Table 1).

4.1 ARCHITECTURE DESIGN

The CNN architecture is most important for engineering efficient image analysis models. The size and efficiency of CNNs are characterized by the number of parameters they have, which depends on the architecture. The number of parameters directly influence the number of calculations that are executed during image analysis, and of course, more calculations take more time. A CNN that is too big (has too many parameters) will not be able to analyze images quickly. On the other hand, a CNN that is too small will not learn very well and will have many misclassifications.

As mentioned before, our architecture needs to be small enough and quickly analyze images so that the user does not notice lag before rotation. Our model uses a VGG style network shown in figure 4. The architecture is composed of 4 nodes which each represent two convolutional layers followed by a single maxpooling and dropout layer. All convolutions use kernels with a stride of (1 in both and direction). In our testing, adding nodes helped our model’s accuracy.

**Preprocessing.** To help the CNN discover relevant features that generalize to the problem, we use a of Laplacian edge detection filter before feeding it into the first convolutional layer. Auto-Face is also trained using normalized values.

4.3 TRAINING

We selected the Helen dataset because the majority of images had faces that were looking at the camera lens, which is like how a user would be looking at a smartphone or tablet’s camera while using it. Every image was flipped horizontally, then rotated three times to create 7 new images for each image. Finally, the images were scaled down to 128x128 pixels. During training, additional data augmentation was used like small random rotations and brightness alterations. The images were also linearly blurred to simulate the user shaking their device while rotating. These techniques create a robust training set which allows our model to correctly classify many general cases.

5 RESULTS

The accuracy of our CNN classifier represents how well the model will perform. Accuracy is defined in Eq. (1), and it is a number between 0 and 1.

Our goal is to correctly orient a device screen relative to the user using the front-facing camera while remaining extremely battery and CPU efficient. Another goal is to be fast enough so that user cannot notice any additional lag compared to gravity. Our algorithm builds on top of the ideas of fully convolutional neural networks and deeply-supervised nets.

6 DISCUSSION

IR camera on iPhones can be used [4, 5]. Flat rotation can be improved.

7 REFERENCES

1. Schmidt, Albrecht, Michael Beigl, and Hans-W. Gellersen. "There is more to context than location." *Computers & Graphics* 23, no. 6 (1999): 893-901.
2. Bartlett, Joel F. "Rock'n'Scroll is here to stay [user interface]." *IEEE Computer Graphics and Applications* 20, no. 3 (2000): 40-45.
3. Hinckley, Ken, Jeff Pierce, Mike Sinclair, and Eric Horvitz. "Sensing techniques for mobile interaction." In *UIST*, vol. 2000, pp. 91-100. 2000.
4. Shimada, Hiroyuki, Yasuyuki Nakamura, and Yuichi Kageyama. "Infrared face authenticating apparatus, and portable terminal and security apparatus including the same." U.S. Patent Application 11/674,622, filed August 16, 2007.
5. Sung, Younghun, and Euihyeon Hwang. "Apparatus and method for human distinction using infrared light." U.S. Patent 7,536,037, issued May 19, 2009.
6. Blaskó, Gábor, William Beaver, Maryam Kamvar, and Steven Feiner. "Workplane-orientation sensing techniques for tablet PCs." In *Proceedings of the 17th Annual ACM symposium on User Interface Software and Technology*. 2004.
7. Bradski, Gary R. "Computer vision face tracking for use in a perceptual user interface." (1998).
8. Forstall, Scott, and Chris Blumenberg. "Portrait-landscape rotation heuristics for a portable multifunction device." U.S. Patent 7,978,176, issued July 12, 2011.
9. Ording, Bas, Marcel Van Os, and Imran Chaudhri. "Screen rotation gestures on a portable multifunction device." U.S. Patent 7,978,182, issued July 12, 2011.
10. Cheng, Lung-Pan, Meng Han Lee, Che-Yang Wu, Fang-I. Hsiao, Yen-Ting Liu, Hsiang-Sheng Liang, Yi-Ching Chiu, Ming-Sui Lee, and Mike Y. Chen. "iRotateGrasp: automatic screen rotation based on grasp of mobile devices." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3051-3054. ACM, 2013.
11. Cheng, Lung-Pan, Fang-I. Hsiao, Yen-Ting Liu, and Mike Y. Chen. "iRotate: automatic screen rotation based on face orientation." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2203-2210. ACM, 2012.
12. Kazemi, Vahid, and Josephine Sullivan. "One millisecond face alignment with an ensemble of regression trees." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1867-1874. 2014.

<https://permadi.com/2011/12/facial-recognition-in-ios-5/>

VGG: [https://arxiv.org/pdf/1409.1556.pdf%20http://arxiv.org/abs/1409.1556.pdf](https://arxiv.org/pdf/1409.1556.pdf%20http:/arxiv.org/abs/1409.1556.pdf)

Laplacian edge detection