Low complexity orientation detection algorithm for real-time implementation

Vikram V. Appia[a], and Rajesh Narasimha[b]

[a]Georgia Institute of Technology, Atlanta, GA, USA.
[b]Texas Instruments, Dallas, TX, USA.

ABSTRACT

In this paper we describe a low complexity image orientation detection algorithm which can be implemented in real-time on embedded devices such as low-cost digital cameras, mobile phone cameras and video surveillance cameras. Providing orientation information to tamper detection algorithm in surveillance cameras, color enhancement algorithm and various scene classifiers can help improve their performances. Various image orientation detection algorithms have been developed in the last few years for image management systems, as a post processing tool. But, these techniques use certain high-level features and object classification to detect the orientation, thus they are not suitable for implementation on a capturing device in real-time. Our algorithm uses low-level features such as texture, lines and source of illumination to detect orientation. We implemented the algorithm on a mobile phone camera device with a 180 MHz, ARM926 processor. The orientation detection takes 10 ms for each frame which makes it suitable to use in image capture as well as video mode. It can be used efficiently in parallel with the other processes in the imaging pipeline of the device. On hardware, the algorithm achieved an accuracy of 92% with a rejection rate of 4% and a false detection rate of 8% on outdoor images.

Keywords: Image orientation detection, scene classifier, real-time implementation.

1. INTRODUCTION

Image orientation detection has emerged as a new research area with the rapid advance and widespread use of digital cameras. Orientation information can be used as an aid to enhance the performance of other associated image processing tasks such as scene classification and color enhancement for digital cameras as well as various detection/classification based applications in video surveillance and automotive vision. The high-end digital cameras have a built-in mechanical sensor to detect orientation. But, many low-end mobile phone/digital cameras and video surveillance equipments do not have mechanical sensors and they require a software based solution for orientation detection.

The biggest challenge in developing real-time algorithms for low-cost equipment is the limited availability of resources (computational ability, memory *etc.*). Although various orientation detection algorithms have been developed in the past, they have all been developed for image management systems, as a post processing tool. These techniques use high-level features and object classification, thus they are not suitable for real-time implementation on low-cost devices. Some previous work in this field use sky detection and other scene classifiers to improve the performance of the image orientation detector. Authors in[9] use color based features in a Bayesian framework and authors in[3,10] use color histograms, edge histograms and color moments as features for classification. Probabilistic models to determine orientation have been explored in[4] and in[11] the authors use low-level visual content with Support Vector Machines to detect orientation. Zhang *et al.* in[8] noticed that the accuracy decreased in indoor images when they used the same algorithm for all the images, thus they developed an indoor/outdoor classifier to improve the accuracy of the orientation detection algorithm. Recently, Ciocca *et al.* in[2] proposed using a face detector to boost the performance of their classifier.

As in all previous literature on this topic, we also make the assumption that the correct image orientation is restricted to four possible rotations of the image in multiples of $90°$. Figure 1 shows an image rotated in multiples of $90°$ indicating the four possible orientations: North, East, South, West. With this framework we can pose the problem as a four-class classifier problem. All the existing methods[1,2,7,9,11] train a detector on huge database of images which have been significantly

Further author information: (Send correspondence to Vikram V. Appia, Rajesh Narasimha)
Vikram V. Appia: E-mail: vikram.appia@gatech.edu
Rajesh Narasimha: E-mail:rajeshn@ti.com)

modified (digitally) with various enhancement and detection/classification algorithms. Thus, the images on which the algorithms are trained are very different from the raw images that will be captured by the sensor. Besides, the raw images (unenhanced images) obtained by capturing the same scene under the exact same conditions using different sensors may differ significantly without these enhancements. In real-time applications the image available to the orientation detector is the unenhanced image obtained from the sensor. Thus, the performance of such training based classifiers will vary based on the image sensor.

In this paper we describe a low complexity orientation detection algorithm that uses the image available directly from the sensor, without resorting to learning based techniques, making it ideal for real-time implementation on low-cost camera equipment. For this task we use only the low-level features in the image. Wang $et\ al.$ in[10] illustrate that using only the image information along the periphery can improve the classifier performance due to boundary symmetry. Luo $et\ al.$[6] present an in-depth analysis of human perception of image orientation, and they suggest that human observers classify the orientation based on source of illumination and the amount of scene activity along each image boundary. We thus make the following assumptions regarding image orientation: (i) useful features for orientation detection lie in the periphery of the image, (ii) smoother regions indicate North, (iii) source of illumination generally points North and (iv) higher scene activity indicates South.

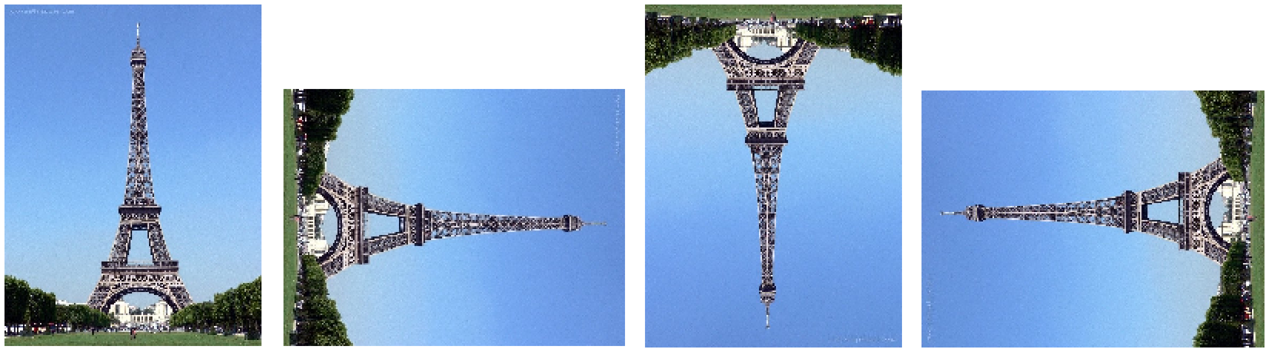

(a) North (b) East (c) South (d) West

Figure 1. Four possible orientations of an image.

The paper is organized as follows: In the Section 2 we will describe our proposed orientation detection algorithm in detail. In Section 3 we present some numbers to illustrate the accuracy and also the computation times on a hardware platform which meet the real-time implementation criteria. And finally, we will conclude our paper in Section 4.

2. PROPOSED METHOD

As mentioned in the introduction we rely on certain low-level features to detect the orientation of the given image. Our goal is to develop an algorithm which can detect the orientation of an image obtained directly from the sensor, which has not been enhanced (color enhancement) based on the the classification of scene type (indoor/outdoor or sky/foliage/faces). Using color based features in a Bayesian framework as suggested in[9] or color histograms and color moments as suggested in[5,10] will lead to unreliable detection due to possible irregularities caused by the camera sensor. Hence, we have developed an algorithm which uses the input gray scale image.

Luo $et\ al.$[6] suggest that an image of QQVGA (160x120) resolution is sufficient to successfully detect orientation based on low level features. Thus the first step in the algorithm would be to down-sample the images to QQVGA resolution. Also, using a low resolution image will reduce the computational complexity and make it suitable for real-time implementation. We now divide the image into 20x20 sub-blocks. This simplifies the task of comparing the various peripheral boundaries as we will see later in this section.

According to assumption (i) we only need to use the information from the peripheral sub-blocks. We generate a smoothness map of the image based on the image gradient. We calculate the image gradient using Sobel Operator. Once, we calculate the gradient of the image, we threshold the gradient image to obtain the smoothness map as shown in the Figure2. This threshold will vary based on the sensitivity of the sensor to noise. And the final smoothness map is a binary

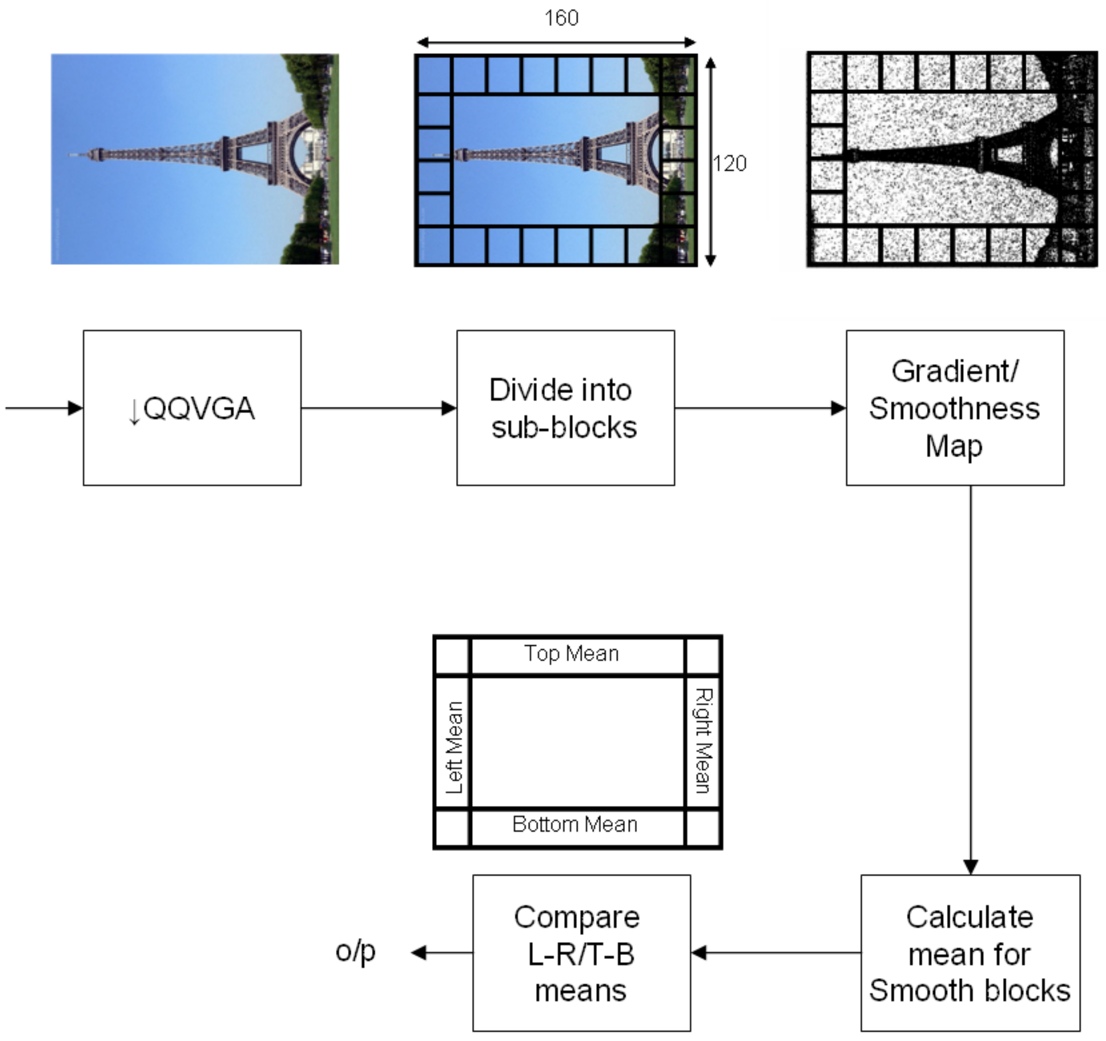


Figure 2. Block Diagram explaining the proposed method

map which assigns '1' to pixels lying in smooth regions. Thus, we can associate a smoothness score with each block which is proportional to the total number of smooth pixels. According to assumption (ii), smoother regions along a periphery of the image indicate higher probability of that boundary indicating North. This suggests that the peripheral sub-blocks which have a larger proportion of smooth regions (higher smoothness score) will indicate North. We use a Smoothness Threshold, α to generate a binary decision indicating the possibility of each sub-block belonging to the North of the image. The chosen blocks are the 'smooth' blocks and we will use only these for further calculations.

Our third assumption states the source of illumination points North, thus the blocks with higher mean intensity indicate North. So, we calculate the means in the peripheral regions of the image considering the values within the selected 'smooth' sub-blocks. We then compare the difference in the means along the opposite sides (peripheries) of the image to decide the orientation of the image. Using the comparison below, we decide the orientation.

$$|\text{Top Mean - Bottom Mean}| > |\text{ Left Mean - Right Mean }| + \text{Rejection Threshold} \Rightarrow \text{North},$$
$$|\text{ Left Mean - Right Mean }| > |\text{Top Mean - Bottom Mean}| + \text{Rejection Threshold} \Rightarrow \text{West},$$

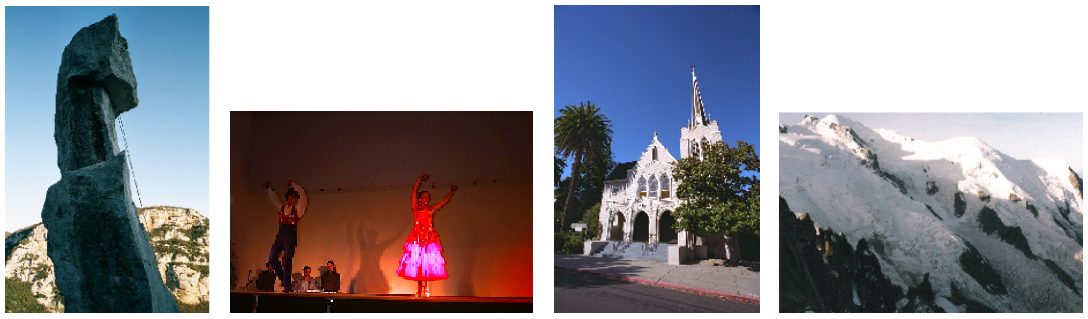
$$|\text{Top Mean - Bottom Mean}| < |\text{ Left Mean - Right Mean }| + \text{Rejection Threshold} \Rightarrow \text{South},$$
$$|\text{ Left Mean - Right Mean }| < |\text{Top Mean - Bottom Mean}| + \text{Rejection Threshold} \Rightarrow \text{East}.$$

Since, we compare the relative means from the opposite sides, we also take into consideration the fourth assumption that the periphery with higher gradient (scene activity) indicates South. The edge with the highest relative score indicates the correct orientation of the image. We also add a Rejection Threshold to this comparison. This makes our algorithm robust against ambiguous cases which may increase the rate of false detection. In the next section we will present some results which illustrate the benefits of using the Rejection Threshold.

Figure 2 illustrates a block diagram of the various steps of the process. The performance of the algorithm is mainly governed by two parameters, the Smoothness Threshold (α) and the Rejection Threshold. Both these parameters can be varied suitably for each sensor type and one can use regression analysis decide these values.

3. EXPERIMENTAL RESULTS

There is a large variation in the accuracy of the methods proposed in previous literature. This is likely due to the differences in the test databases used in each approach. Luo $et\ al.$[6] present the accuracy of orientation detected by human observers and they argue that any automated orientation detection algorithm can not match the sophistication of human brain. Thus, they suggest that accuracy achieved by human observers can be treated as the upper bound on accuracy. To measure accuracy of orientation detection by human observers using only low-level cues, as is the case in our approach, they presented the subjects with down-sampled test images. In such down-sampled images the observers have to rely on features like source of illumination and scene activity to identify the correct orientation. In this experiment they report an average accuracy of 84%, which we will use as the upper bound for orientation detection for our algorithm.



(a) (b) (c) (d)

Figure 3. Images whose orientation was detected correctly.

3.1 Offline Validation

Although the developed algorithm is designed to use the sensor image in real-time, we did some offline testing of our algorithm for validation. For this purpose we tested the accuracy of our detector on 200 randomly chosen consumer images. These images included both indoor and outdoor images, as well as portrait/landscape images in varying conditions. In Figure 3 we present a few images where we were able to detect the orientation correctly. Figure 4 samples a few images which were rejected by our algorithm due to the Rejection Threshold. Overall we achieved an accuracy of 86% with a rejection rate of 14% and a false detection rate of 14%. In Figure 5, we present some examples of false detected images, along with the correct orientations. It can be seen that the false detections occurred in images where one or more of the four assumptions were not satisfied. Without the Rejection Threshold our accuracy drops to 74%, which is still at par with some of the existing orientation detection algorithms.

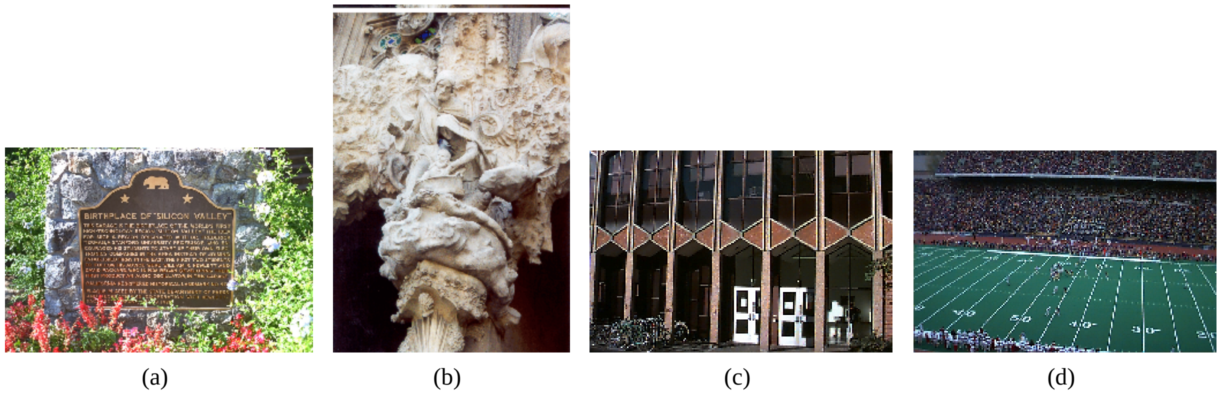


Figure 4. Rejected image by our algorithm due to Rejection Threshold.

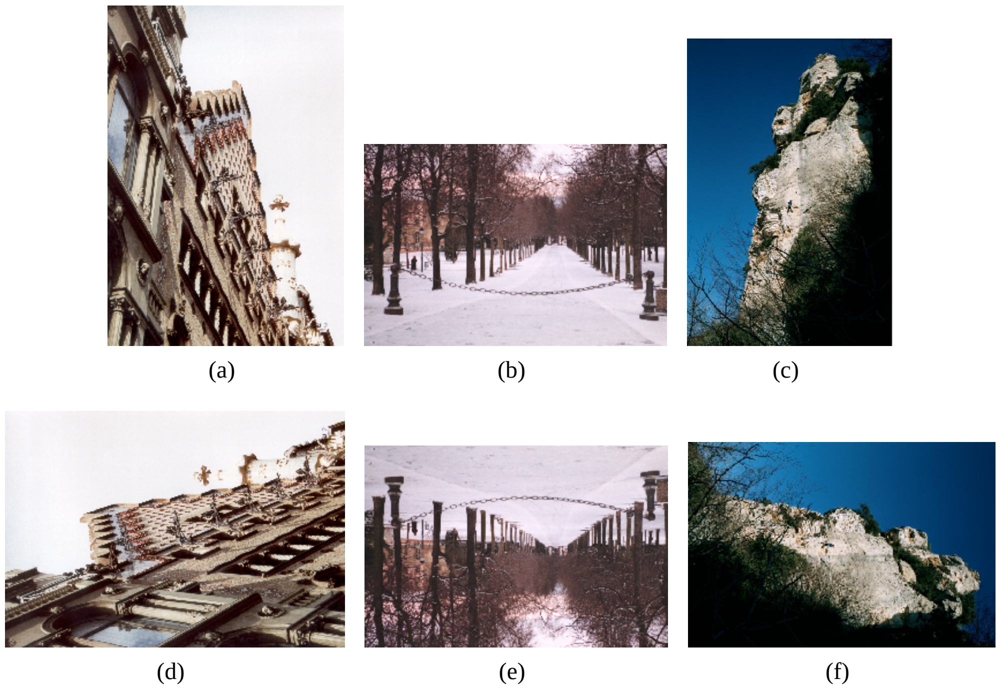


Figure 5. Images with False detection. Top Row: Input orientation. Bottom Row: Detected orientation.

3.2 Validation for Hardware Implementation

In order to evaluate the performance of our algorithm on hardware in real-time, we tested the performance of our algorithm on a mobile phone camera device with a 180 MHz, ARM926 processor. To test the accuracy we used professional prints of some of the outdoor images from our dataset. Since, ambient lighting may significantly vary the image captured by the sensor, we placed these images in front of the camera inside a light box which simulated broad daylight. Under these conditions, for outdoor images, we achieved an accuracy of 92% with a rejection rate of 4% and a false detection rate of 8%. We summarize the results in Table 1.

3.2.1 Computation time

We also measured the computation times on hardware for the various processing steps involved in our algorithm. To achieve real-time implementation for image as well as video capture, the algorithm must process each image frame in less than 33ms. Our measurements suggest that the Sobel operator used to calculate the gradient of the image, consumes the bulk of the processing time. For a QQVGA frame it take approximately 6ms to calculate the gradient image. The

Table 1. Performance comparison with and without rejection threshold for offline testing and hardware implementation

	Accuracy with Rejection Threshold	Rejection Rate	False Detection	Accuracy without Rejection Threshold
Offline Validation	86%	14%	14%	74%
Hardware Implementation	92%	4%	8%	88%

other tasks mainly involve thresholding, computing mean and comparing relative means. These calculations are performed on-the-fly with a single pass over the entire image to suit real-time video capture and the combined time required for these operations is about 4 ms. We are thus, able to detect the orientation of each frame in 10ms, and this information can be used by other algorithms in the image processing pipeline as auxiliary information.

3.2.2 Memory Requirements

Since the target application of our algorithm is low-cost camera devices, we also have to adhere to stringent memory requirements. The input image, a QQVGA image, requires 19,200 bytes of memory. The Sobel operation on this image generates two images of the same size for the x- and y- gradient images. Finally, the smoothness map generated from these gradient images also requires 19,200 bytes of memory. All the other variables required consume a few bytes of memory. Thus, overall our algorithm uses approximately 77 Kb of available memory.

4. CONCLUSION

We have developed an orientation detection algorithm which performs reliably across all scene types and it can also be used efficiently in parallel with the other processes in the imaging pipeline of the device. The algorithm achieved an accuracy of 86% with a rejection rate of 14% and a false detection rate of 14% in offline testing spanning a wide range of consumer images. On hardware the algorithm achieved an accuracy of 92% with a rejection rate of 4% and a false detection rate of 8% on outdoor images. It can be implemented even in devices with minimal availbale memory and it is suitable for real-time image and video capture.

REFERENCES

[1] Shumeet Baluja. Automated image-orientation detection: a scalable boosting approach. *Pattern Anal. Appl.*, 10(3):247–263, 2007.

[2] Gianluigi Ciocca, Claudio Cusano, and Raimondo Schettini. Image orientation detection using low-level features and faces. In *Digital Photography*, page 75370, 2010.

[3] Alessandra Lumini and Loris Nanni. Detector of image orientation based on borda count. *Pattern Recogn. Lett.*, 27:180–186, February 2006.

[4] Jiebo Luo and Matthew Boutell. A probabilistic approach to image orientation detection via confidence-based integration of low-level and semantic cues. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop Volume 9*, CVPRW '04, pages 141–, Washington, DC, USA, 2004. IEEE Computer Society.

[5] Jiebo Luo and Matthew Boutell. Automatic image orientation detection via confidence-based integration of low-level and semantic cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:715–726, 2005.

[6] Jiebo Luo, David Crandall, Amit Singhal, Matthew Boutell, and Robert Gray. Psychophysical study of image orientation perception.

[7] Siwei Lyu. Automatic image orientation determination with natural image statistics. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 491–494, New York, NY, USA, 2005. ACM.

[8] Lei Zhang Mingjing. Boosting image orientation detection with indoor vs. outdoor classification, 2002.

[9] Aditya Vailaya, Hongjiang Zhang, Senior Member, Changjiang Yang, Feng-I Liu, and Anil K. Jain. Automatic image orientation detection. In *IEEE Transactions on Image Processing*, pages 600–604, 2002.

[10] Yongmei Wang and Hongjiang Zhang. Content-based image orientation detection with support vector machines. 2001.

[11] Yongmei Michelle Wang and Hongjiang Zhang. Detecting image orientation based on low-level visual content. *Comput. Vis. Image Underst.*, 93:328–346, March 2004.