Volume 3, Issue 11, November 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Face Detection using Digital Image Processing

Ridhi Jindal, Anuj Gupta
M.tech (C.S.E)
Bahra University,Shimla, India

Dr.Sonia Vatta
HOD (C.S.E)
Bahra University,Shimla, India

Abstract: ***This paper presents a technique for automatically detecting human faces in digital color images. This is two-step process which first detects regions contain human skin in the color image and then extracts information from these regions which might indicate the location of a face in the image. The skin detection is performed using a skin filter which relies on color and texture information. The face detection is performed on a grayscale image containing only the detected skin areas. A combination of thresh holding and mathematical morphology are used to extract object features that would indicate the presence of a face. The face detection process works predictably and fairly reliably, as test results show.***

I. Introduction

Designing a system for automatic image content recognition is a non-trivial task that has been studied for a variety of applications. Computer recognition of specific objects in digital images has been put to use in manufacturing industries, intelligence and surveillance, and image database cataloging to name a few.[1] In this project, an algorithm for automating the detection of human faces in digital images was developed and can serve as an introduction for future work in detecting people in videos[2].

Several systems designed for the purpose of finding people or faces in images have already been proposed by numerous research groups. Some of these programs, such as the Rowley, Baluja, and Kanade system developed at Carnegie Mellon, rely on training of a neural network and computing distance measures between training sets to detect a face[3]. Other software packages exist which can recognize facial features in pictures known to contain a human face somewhere in the image. This project focused on face detection in arbitrary color images and differs from the first type of system in that it relies on a combination of color and grayscale information[2]. Additionally, it does not require the time consuming process of training a neural net or computing distance measures between every possible region in the image[4]. The developed system also differs from those software packages that recognize facial features because, in this scenario, the task is to detect a facial region in an arbitrary image, and not to analyze images known to contain a face.

The process for detection of faces in this project was based on a two-step approach. First, the image is filtered so that only regions likely to contain human skin are marked. This filter was designed using basic mathematical and image processing functions in MATLAB. The second stage involves taking the marked skin regions and removing the darkest and brightest regions from the map [7]. The removed regions have been shown through empirical tests to correspond to those regions in faces which are usually the eyes and eyebrows, nostrils, and mouth. By performing several basic image analysis techniques, the regions with "holes" created by the threshholding can be considered likely to be faces[11]. The entire system was entirely automated and required no user intervention save for indicating the correct file names to be processed at each stage. While not implemented in this project, a more advanced program could implement a third step to discriminate between hole sizes and spatial relationships to make an even more robust detection system.

II. METHODOLOGY

i. Skin Filter

In this project, the filter was built in MATLAB. Several of the low level images processing functions are already built into the MATLAB environment and this project required time to be invested in building a working filter algorithm, not writing code for low level functions.

The input color image should be in RGB format with color intensity values ranging from 0 to 255. Due to restrictions on speed and performance, this project used images smaller than 250x250 in area[9]. The RGB matrices are "zeroed" [10]to supposedly prevent desaturation when the image is converted from RGB color space to IRgBy color space. The smallest intensity value greater than 10 pixels from any edge in any of the three color planes is set as the zero-response of the image. This value is subtracted from all three color planes.

The RGB image is transformed to log-opponent (IRgBy) values and from these values the texture amplitude, hue, and saturation are computed. The conversion from RGB to log-opponent is calculated according to a variation on the formula given by Fleck& amp Forsyth:

$$I = [L(R)+L(B)+L(G)]/3$$
$$Rg = L(R)-L(G)$$
$$By = L(B)-[L(G)+L(R)]/2$$

The L(x) operation is defined as L(x)=105*log10(x+1). The Rg and By matrices are then filtered with a windowing median filter of with sides of length 4*SCALE. The SCALE value is calculated as being the closest integer value to (height+width)/320. The m edian filtering is the rate limiting step throughout the skin detection process, and could be improved by implementing an approximation of a windowing median filter as suggested by Fleck's multi-ring operator.

A texture amplitude map is used to find regions of low texture information. Skin in images tends to have very smooth texture and so one of the constraints on detecting skin regions is to select only those regions with little texture. The texture map is generated from the matrix I by the following steps:

1. Median filter I with a window of length 8*SCALE on a side
2. Subtract the filtered image from the original I matrix
3. Take the absolute value of the difference and median filter the result with a window of length 12*SCALE on a side.

Hue and saturation are used to select those regions whose color matches that of skin. The conversion from log opponent to hue is hue = ($atan^2$(Rg,By)), where the resulting value is in degrees. The conversion from log opponent to saturatio n is saturation = sqrt(Rg^2+By^2). Using constraints on texture amplitude, hue, and saturation, regions of skin can be marked.

If a pixel falls into either of two ranges it is marked as being skin in a binary skin map array where 1 corresponds to the coordinates being a skin pixel in the original image and 0 corresponds to a non-skin pixel. The allowed ranges are either :

(1) texture<4.5, 120<160, 10<60
(2) texture<4.5, 150<180, 20<80

The skin map array can be considered as a black and white binary image with skin regions (value 1) appearing as white. The binary skin map regions are expanded using a dilation operator and a disc structuring element[7]. This helps to enlarge the skin map regions to include skin/background border pixels, regions near hair or other features, or de-saturated areas. The dilation adds 8-connected pixels to the edges of objects. In this implementation, the dilation was performed recursively five times for best results. The expanded map regions are then checked against a lenient constraint on hue and saturation values, independent of texture. If a point marked in the skin map corresponds to a pixel with 110<=hue<=180 and 0<=saturation<=130, the value remains 1 in the map.

The skin filter is not perfect, either due to coding errors or improper constraints, because there is a tendency for highly saturated reds and yellows to be detected as skin. Often this causes problems in the face detection when a large red or yellow p atterned object is present in the image[8]. See the results in Appendix B for several examples of the skin filter output and cases in which the skin filter marked highly saturated red and yellow as skin.

ii. Face Detection from Skin Regions

The binary skin map and the original image together are used to detect faces in the image. The technique relies on threshholding the skin regions properly so that holes in face regions will appear at the eyebrows, eyes, mouth, or nose. Theoretically, all other regions of skin will have little or no features and no holes will be created except for at the desired facial features[5]. This method seems to be an oversimplification of the problem, but with some additional constraints on hole sizes or spatial relationships, could prove to be a powerful, fast, and simple alternative to neural network processes.

Detection of face regions was broken into two parts, the first using the Khoros visual programming application and the second part using a MATLAB program. The two Khoros workspaces are shown in Appendix A along with the MATLAB code. All of the functions used are standard image analysis techniques (hole filling algorithms, threshholding, connected components labeling, etc.) that should be straightforward, though perhaps tedious, to build in any programming language.

The first step is to ensure that the binary skin map is made up of solid regions (i.e. no holes). Closing holes in the skin map is important because later the program assumes that the only holes are those generated after the threshholding operation[7]. A hole closing is performed on the skin map image with a 3x3 disc structuring element and then this image is multiplied by a grayscale conversion of the original image. The result is a grayscale intensity image showing only the parts of the image containing skin.

To improve contrast, a histogram stretch is performed on the resulting grayscale image. This helps to make the dark and light regions fall into more predictable intensity ranges and compensates somewhat for effects of illumination in the image. The image can now be threshholded to remove the darkest and lightest pixels. Experimentation showed that an acceptable threshold was to set all pixels with values between 95 and 240 equal to 1 and those pixels above and below the cutoff equal to 0. For most test images, this threshold work quite well[10]. The binary image created by the threshold is then passed through a connected components labeling to generate a "positive" image showing distinct skin regions.

A negative image is next generated that will show only holes as objects. Hole closing of the binary image generated by the threshold operation is performed with a 4x4 disc structuring element. The result is subtracted from the original binary image and the difference shows only hole objects.

The negative hole image and the positive labeled image are then used together to find which objects in the image might be faces. First those holes in the negative image which are only 1 pixel in size are removed because these tend to represent anomalous holes[11]. An even better technique might be to remove all but the three largest hole objects from the negative image. The hole objects are expanded using a dilation and this binary image is then multiplied by the positive labeled image. The product is an image where only the pixels surrounding a hole are present. Because the positive image was labeled, the program can easily determine which objects have holes, and which do not. A simple function computes which integers appear in the hole adjacency image and then generates an output image containing the labeled connected components that have this value.

Because this process relies only on finding holes in threshholded objects, there is a greater chance of finding faces regardless of the perspective. A drawback is that there is also a greater risk of detecting non-face objects. The test results show very good performance when a face occupies a large portion of the image, and reasonable performance on those images depicting people as part of a larger scene. To make the program more robust, detected face objects could be rejected if they don't occupy a significant area in the image. Another drawback of this process is that images in which people appear partially clothed tend will result in a very large skin map. The result is often a labeling of the entire head, arms, and torso as a single object. Thus the face finding is an overestimate of potential skin objects[6]. All things considered, this technique developed over a period of ten weeks shows promise and with some "intelligence" added to the algorithm, could generate very reliable results on a wide variety of images.

III. CONCLUSION

In this project, the goal of implementing a hardware system to detect and track human faces in real time was achieved. A software implementation of the algorithm was examined in MATLAB to verify its accuracy. Although the transition from software to hardware required some modification to the original algorithm, the initial goal was still accomplished. The face detection algorithm was derived from a skin detection method. Face tracking was achieved by computing the centroid of each detected region, although it only worked in the presence of at most two people. The system was proved to work in real time with no lagging and under varying conditions of facial expressions, skin tones, and lighting.

REFERENCES

[1] A. Ajit.(2008). \FACE RECOGNITION TECHNOLOGY".COCHIN UNIVERSITY OF SCIENCE & TECHNOLOGY,KOCHI - 682022. Accessed on 10th March, 2013.
[2] Animetrics.(2008). \Biometrics and Facial Recognition".Animetrics. : http://www.animetrics.com/technology/frapplications.html
[3] Animated.(2008). \Biometrics and Facial Recognition".Animated.: http://www.animetrics.com/technology/frapplications.html
[4] A Model-Based Gaze Tracking System by Rainer Stiefelhagen, Jie Yang, Alex Waibel
[5] Digital Image Processing Using MATLAB by Gonzalez, Woods &Eddins,Prentice
[6] Frontal-view face detection and facial feature extraction using color and morphological operations by Jian-Gang Wang, Eric Sung *
[7] H. Veronica.(2001) \Biometrics: Face Recognition Technology".GIAC, SANS Institute. pp.2-3. Accessed at March 7th, 2011: http://www.giac.org/download.php?p=gsec 627&c=6203efa1e18401f74c8870e2f54fbb3b
[8] Images taken from : www.faceresearch.org
[9] L.-F. Chen, H.-Y. Liao, J.-C. Lin, and C.-C. Han. Why recognition in a statistics-based face recognition system should be based on the pure face portion: a probabilistic decision-based proof. Pattern Recognition,34(5):1393–1403, 2001.
[10] M. Ooi, "Hardware Implementation for Face Detection on Xilinx Virtex-II FPGA Using the Reversible Component Transformation Color Space," in Third IEEE International Workshop on Electronic Design, Test and Applications, Washington, DC, 2006.
[11] M. Bartlett, J. Movellan, and T. Sejnowski. Face recognition by independent component analysis. IEEE Trans. on Neural Networks, 13(6):1450–1464, November 2002.

