



Development of a toolkit in Silverlight to detect physiognomies in real time

Marcelo Cabral Ghilardi¹, Vinicius Gadis Ribeiro^{1*}, Jorge Rodolfo Zabadal² and Cristiana Andrade Poffal³

¹Centro Universitário Ritter dos Reis, Faculdade de Informática, Porto Alegre, Rio Grande do Sul, Brazil. ²Departamento de Engenharia Mecânica, Universidade Federal do Rio Grande do Sul, Av. Osvaldo Aranha, 99, 90046-900, Porto Alegre, Rio Grande do Sul, Brazil. ³Departamento de Matemática, Universidade Federal do Rio Grande, Rio Grande, Rio Grande do Sul, Brazil. *Author for correspondence. E-mail: vinicius@uniritter.edu.br

ABSTRACT. Security demands, forensic practices and the identification of criminals require the detection and recognition of iris and fingerprints and of faces in videos and photographs. Moreover, there is an increasing need for multiple forms of human-machine interaction. Control devices by body stimulus are a need and a trend. For example, currently some computers, laptops, phones and video games provide interaction from their cameras, not only for face detection but also for body movements and the detection of objects. Most devices are Internet accessed, which creates an even greater range of possibilities. These technological trends have prompted the development a toolkit for detecting faces in real time. The choice of Silverlight framework for the development of this toolkit provides these applications with instruments that could be implemented in a web browser. This toolkit may be used for other purposes, such as face and iris recognition, body movement and the monitoring of premises. An application was developed as example and proof of concept.

Keywords: technology, face detection, skin color, Silverlight, framework.

Desenvolvimento de um *toolkit* em Silverlight para detecção de faces em tempo real

RESUMO. A demanda por segurança em todos os setores, as práticas forenses e a identificação de criminosos fazem com que a detecção e o reconhecimento de faces em vídeos e fotografias, reconhecimento de íris e digitais, tornem-se um estudo importante. Além disso, há cada vez mais necessidade de múltiplas formas de interação homem/máquina. Controlar dispositivos por estímulos corporais é uma necessidade e tendência. Atualmente, por exemplo, alguns computadores, notebooks, telefones e videogames permitem interação a partir de suas câmeras, não apenas pela detecção de face, mas movimento corporal e detecção de objetos. A maioria destes dispositivos, além de câmera, possui acesso à Internet, o que cria uma gama de possibilidades ainda maior. Estas tendências tecnológicas justificam o desenvolvimento de um *toolkit* para a detecção de faces em tempo real. A escolha do *framework* Silverlight para o desenvolvimento deste *toolkit* proporciona que as aplicações que o utilizarem possam ser executadas em um navegador de internet. Este *toolkit* pode ser aprimorado para outras áreas de detecção e reconhecimento, por exemplo, reconhecimento de faces, de íris, movimento corporal e monitoramento de ambientes. Como prova de conceito, foi desenvolvida uma aplicação-exemplo.

Palavras-chave: tecnologia, detecção de faces, cor de pele, Silverlight, framework.

Introduction

Automatic recognition systems for credit cards, bank cards and driver's license cards are examples in which two images are compared, or rather, a target image obtained at the recognition instance is compared to the image on the card, or to an image bank in which the target image must have an equivalent. Forensic practice and the identification of criminals use the same system in which an image taken from a security camera or from photographs is researched in police data banks. The same occurs in the recognition of people in data bases retrieved

from videos installed in streets, in shopping malls and others. Several authentication systems are currently validated through the recognition of faces, irises, fingerprints and voices.

Further, several forms of interactions between people and machines are constantly in demand. A trend exists to control devices by body stimuli. New videogames, such as Wii, Playstation and Xbox have interaction modules with the games activated by body movements. Handicapped accessible computers may be controlled by body movements and even by brain stimuli. Another trend is

enhanced reality, or rather, it is the integration of the real with the virtual world in real time and in three dimensions (AZUMA, 1993), in which objects in videos are detected and overlaid by other objects, animations or information (LIN; FAN, 2001).

Possibilities are also very high for the invention of other applications to detect objects, faces and people (COSTA et al., 2005; JENG et al., 1998; LOPES, 2005; MORAES, 2006; VEZHNEVETS, 2002). In fact, they were the possibilities that stimulated the project under analysis. It focuses on the invention of a toolkit that would help in the development of Web applications for the detection of faces in real time. Face detection method used in the development of the toolkit comprises the color spectrum of the human skin (CAI; GOSHTASBY, 1999; LOW, 2001; SUNG; POGGIO, 1998; ZAIDAN et al., 2010; LEE, et al., 2007; VIJAYANANDH; BALAKRISHNAN, 2012). In spite of different intensities, human epidermis has slight chrominance variations (DAI; NAKANO, 1996; VEZHNEVETS, 2002). The following section deals with some techniques for face detection, coupled to its advantages and disadvantages.

The framework Silverlight was developed since the toolkit should make it possible that applications are performed by web browsers and accessed by the client's machine webcam. The framework has been developed by Microsoft and made possible the creation of Web and desktop applications (BORCK, 2010; SCHULTE, 2011). Adobe Flash is another available and competing tool. Although the two technologies are highly similar, research by Borck (2010) describes the superiority of Silverlight when compared to the Adobe Flash. In current assay Silverlight has been selected for analysis.

An application-example explaining all the stages for face detection was developed for the validation of the toolkit. Several skin hues and backgrounds have been tested to arrive at a skin color interval that applied to most people and backgrounds. The toolkit contains parameters to frame new backgrounds and several types of skin color. It may be improved so that new methods in the detection and recognition of faces, irises, body movements or monitoring of premises could be developed.

The main objective of this paper is to present, as a proof of concept, the development of an application for face recognition.

Current work consists of the following sections: material and methods, showing the basic method employed to development of software; next section discusses results obtained; last section deals with

conclusions. The Appendix demonstrates results and configurations for several situations (which in some cases are far from being ideal) for face recognition.

Material and methods

The basic method

The method, which foregrounded current analysis, was developed from analysis methods in skin texture. This option was taken due to the high speed in processing, its use in real time by webcams and the low complexity of the algorithm in the Silverlight development.

However, face detection by this method is not easy since it is affected by such factors as skin hues, different visualization points of view, facial illumination and the webcam's image quality.

Keeping in mind the above difficulties and taking into consideration processing in real time and webcam use, its architecture will be detailed in the section below.

Use of color YCbCr space

Even before processing, images are received in pixel matrices when Silverlight technology is employed. Each pixel contains 32 bits representing alpha, red, green and blue, or rather, in ARGB – RGBA (Red Green Blue Alpha) format. Although sometimes described as a color space, in fact it is an RGB color model with additional information, the Alpha channel.

The RGB color space – the color space used by the monitor of cathode ray tubes – has several important disadvantages when dealing with the filtering of bands of a certain color. In other words, if it is required that filtering be accomplished by skin texture, a space with only slight light influence is required. A greater volume of color spaces is required in the case of RGB space, with a consequent impossibility for defining a simple color thresholding. There are other color spaces without the above-mentioned problem: for instance, HSV hue space – the system of colors formed by the components *hue* (matrix), *saturation* and *value* – defines the color in its three components *Hue*, *Saturation* and *Value* (brightness), in which true color (*Hue*) is represented by a 0-360° circle and brightness is the height of the cylinder.

The space is highly used in algorithms for skin detection since computer costs for RGB conversion into HSV are high and a complicating factor for the face detection project in real time. YCbCr colors space – color space characterized by a non-linear transformation of space RGB, greatly used in TV signal standardization in Europe – also allows the limitation

of space for skin texture and its computer costs are lower than those of HSV space. The latter space stores the brightness of Y and the color of Cb (blue) and Cr (red). The conversion of RGB into YCbCr may be accomplished by simple addition and multiplication.

Image processing for face detection is composed of four phases. The first phase consists of filtering the original image with the selection of pixels with values contained in the pre-established skin color thresholds and the generation of a binary image from the filter results. In the second phase, the image is generated and noises filtered, or pixels that do not belong to a face owing to their distance from other pixels, are deleted to generate a binary image once more. The third phase consists of the expansion of remaining pixels, since important pixels may have been lost during noise filtering during the second phase. In the fourth phase, a quantitative, vertical and horizontal histogram is produced so that the face area could be determined by these values. Results of such procedure are coordinates x_1 , y_1 , x_2 , y_2 of the probable face area in the original image (SINGH et al., 2003).

Algorithm of filtering by skin texture

Algorithm development required the definition of the spectrum or thresholds of the skin color – defined from the 13-image tests. Two properties, which define the maximum and minimum standard thresholds, changeable when required, are used for the performance. This is required since camera quality and background illumination may interfere in the thresholds.

```
SkinLowerThreshold = new YCbCr(0.10, -0.25, 0.05);
```

```
SkinUpperThreshold = new YCbCr(1.00, 0.05, 0.20);
```

In the algorithm below, the original image (1) is received and a resulting image (empty) of the same size as the original one is produced (2). Henceforth, all the pixels of the original image (3) are surveyed and RGB pixels are converted into YCbCr (4) since the skin spectrum lies in YCbCr. Verification occurs to see whether the pixel is contained in the established thresholds for the skin (5): if positive, a value that defines that the pixel is a skin pixel (6) is attributed to the pixel of the same place, albeit in the resulting image. The other pixels of the resulting image remain empty. A binary image is thus produced in which marked pixels represent pixels that may be skin pixels (7).

```
1      public      WriteableBitmap
ToSkin(WriteableBitmap imagem) {
2      WriteableBitmap result = new
WriteableBitmap(imagem.PixelWidth,
```

```
imagem.PixelHeight);
    var p = image.Pixels;
    var rp = result.Pixels;
3    for (int i = 0; i < p.Length; i++) {
4        var ybcr = YCbCr.FromARGB(p[i]);
5        if (ybcr.Y >= SkinLowerThreshold.Y &&
ybcr.Y <= SkinUpperThreshold.Y
&& ybcr.Cb >= SkinLowerThreshold.Cb &&
ybcr.Cb <= SkinUpperThreshold.Cb
&& ybcr.Cr >= SkinLowerThreshold.Cr &&
ybcr.Cr <= SkinUpperThreshold.Cr) {
6            rp[i] = ColorSkin;        }        }
7        return result;    }
```

Figure 1 shows that the original image is a colored one. After algorithm processing, the resulting image is a binary one that represent pixels that may be skin pixels.

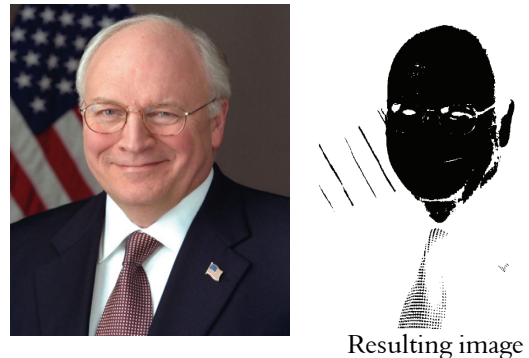


Figure 1. Images of skin filtering.

Source: original image of Dick Cheney, <http://www.wikimedia.org>.

Algorithm for noise reduction

The algorithm runs through all the pixels of the image produced by the previous one: when it is a skin pixel, it searches all the neighboring pixels at a distance of three pixels in all directions. If any neighboring pixel is found empty, it cleans the pixel – henceforth it is not a skin pixel anymore – and thus the image is freed from the noise. Thirteen images were tested to cover a 3-pixel distance by employing distances between 1 and 5 pixels.

The codification of the algorithm is given below. It first receives the resulting image of the previous algorithm (1) and creates a new resulting image (empty) of the same size as the original one (2). In the PosPix matrix, all adjacent pixels which will be visited are informed (3), starting from the pixel-position (0, 0), the pixel to the right (0, 1) and so on and so forth, up to the more distant pixel (-3, 3). All the image pixels are visited, line by line (4). The adjacent pixels are run in each skin pixel (5): if no adjacent empty pixel is found (6), the corresponding pixel is

marked in a new resulting image as if it were the skin pixel (7). Consequently, a new binary image is obtained in which all image noises are deleted (8). A function which verified whether an adjacent pixel is either empty or a skin pixel is thus accomplished (9)

```

1 public WriteableBitmap SkinFiltro(Writeable
  Bitmap image) {
    var p = image.Pixels;
    var w = image.PixelWidth;
    var h = image.PixelHeight;
2 var result = new WriteableBitmap(w, h);
    var rp = result.Pixels;
    int i = 0;
    int qtdPosPix = 49;
3 int[,] PosPix = { { +0,+0},{+1,+0},{+1,
+1},{+0,+1},{-1,+1},{-1,+0},{-1,-1},
    {+0,-1},{+1,-1},{+2,-
1},{+2,+0},{+2,+1},{+2,+2},{+1,+2},
    {+0,+2},{-1,+2},{-2,+2},{-
2,+1},{-2,+0},{-2,-1},{-2,-2},
    {-1,-2},{+0,-2},{+1,-
2},{+2,-2},{+3,-2},{+3,-1},{+3,+0},
    {+3,+1},{+3,+2},{+3,+3},
+2,+3},{+1,+3},{+0,+3},{-1,+3},
    {-2,+3},{-3,+3},{-3,+2},{-
3,+1},{-3,+0},{-3,-1},{-3,-2},
    {-3,-3},{-2,-3},{-1,-3},{+0,-
3},{+1,-3},{+2,-3},{+3,-3}};

4 for (int y = 0; y < h; y++) {
    for (int x = 0; x < w; x++, i++) {
        int iPosPix = 0;
        bool bContinue = true;
5 while (bContinue && iPosPix < qtdPos
  Pix) {
6 if (PixelIsEmpty(x+PosPix[iPosPix, 0],
  y + PosPix[iPosPix, 1], p, w, h))
            { bContinue = false; break; }
            iPosPix++;
        if (bContinue)
7 rp[i] = ColorSkin;
    }
    return result;
}

9 private bool PixelIsEmpty(int x, int y, int[]
  p, int w, int h) {
    bool isEmpty = false;
    if (x > 0 && x < w && y > 0 && y < h) {
        isEmpty = (p[y * w + x] ==
  ARGB.Empty);
    }
    return isEmpty;
}

```

As may be seen in Figure 2, the original image consists of an image with noises. After algorithm processing, the resulting image is a noiseless image.

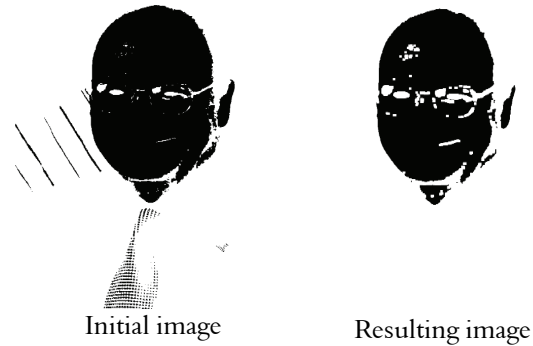


Figure 2. Images by noise filter.

Algorithm for pixel expansion

The previous process by which noises were eliminated may also delete important pixels from a face. An algorithm for the expansion of the remaining pixels is required and thus eventual blank spaces in the face, probably caused by excessive brightness, hair, spectacles or other factors, may be improved.

When the expansion algorithm receives the image resulting from the noise algorithm, it runs through all the pixels and verifies the neighborhood of each empty pixel at a distance up to three pixels in all directions. If it finds any adjacent skin pixel, it marks the pixel as a skin pixel. All existing skin pixels are thus expanded.

The implementation of the algorithm's code follows. The resulting image of the previous algorithm is received (1) and a new resulting image (empty) of the same size as the original image is created (2). The position of adjacent pixels that will be visited are informed in the PosPix matrix (3) – although differently from that in the previous algorithm, the result is the same – verifying from the nearest adjacent pixels to the most distant ones, up to a distance of three pixels. All the image's pixels are visited, line by line (4), while the adjacent pixels are run for each empty pixel (5). If an adjacent skin pixel is found (6), the corresponding pixel is marked in the new resulting image as if it were a skin pixel (7). A new binary image is found in which all the pixels of the received image (8) are expanded. A function is thus accomplished which verified whether an adjacent pixel is either a skin pixel or an empty one (9)

```

1 public WriteableBitmap SkinDilate
  (WriteableBitmap image) {
    var p = image.Pixels;
    var w = image.PixelWidth;
    var h = image.PixelHeight;
2 var result = new WriteableBitmap(w, h);

```

```

        var rp = result.Pixels;
        int i = 0;
3    int[] PosPix = { 0, -1, +1, -2, +2, -3, +3 };
        int ixPosPix;
        int iyPosPix;
        bool bContinue;
4    for (int y = 0; y < h; y++) {
        for (int x = 0; x < w; x++, i++) {
            ixPosPix = 0;
            iyPosPix = 0;
            bContinue = true;
5        while (bContinue && ixPosPix < Pos
Pix.Length) {
            while (bContinue && iyPosPix <
PosPix.Length) {
6                if (PixelIsNotEmpty(x-PosPix[ixPos
Pix],
y+PosPix[iyPosPix], p, w, h))
7                    { rp[i] = ColorSkin; bContinue =
false; continue; }
                        iyPosPix++; }
                        ixPosPix++; } }
8    return result; }

9    private bool PixelIsNotEmpty(int x, int y,
int[] p, int w, int h) {
        bool isNotEmpty = false;
        if (x > 0 && x < w && y > 0 && y < h) {
            isNotEmpty = (p[y * w + x] !=
ARGB.Empty); }
        return isNotEmpty; }

```

Figure 3 reveals that the original image is a noise-less image, with some spaces open in the face. After the processing of the algorithm, the face in the resulting image has less open spaces.



Initial image



Resulting Image

Figure 3. Images from expansion algorithm.

Calculation of the area and position of the face

Since the image is the result of an expansion algorithm, a matrix is produced with a certain quantity of skin pixels per line and column, or rather, the image's X and Y. Taking these values into

consideration, the first and last positions with matrix X value are respectively positions X1 and X2. Similarly, in matrix Y, the first and last positions with matrix Y value are respectively positions Y1 and Y2.

So that the filter could be amplified, the positions with lower values than LimitMinimumX and LimitMinimumY are discarded. After several tests with different sized images, the calculation below defined the minimum limits of values for X and Y.

MaximumLimitX = Image Width/20 and MinimumLimitY = ImageHeight / 20

Two functions were produced from the above calculation: the first function - GetHistogram() – produces two matrices: matrix histX for the lines (1) and matrix histY for the columns (2) of the image, where the number of pixels of each line and column are respectively stored. It also stores most of the pixels found in each X and Y matrix (3). So that all matrices would be complete, all the pixels of the image are visited (4); each color skin pixel found is added (5) and the quantity of each position of each matrix is compared to find the greatest number of pixels (6).

```

public void GetHistogram(WritableBitmap image) {
    var p = image.Pixels;
    var w = image.PixelWidth;
    var h = image.PixelHeight;
1    histX = new int[w];
2    histY = new int[h];
3    histMaxX = 0; histMaxY = 0;
4    for (int y = 0; y < h; y++) {
        for (int x = 0; x < w; x++) {
5            if (p[y * w + x] == ColorSkin) {
                histX[x]++; histY[y]++;
6            if (histX[x] > histMaxX) {
histMaxX = histX[x]; }
                if (histY[y] > histMaxY)
                    {histMaxY = histY[y]; } } } }

```

The second function, GetLimits(), employs the matrices of the previous function to calculate the face's position by running through the items of each matrix and by discarding all values less than the pre-defined ones (1). In the case of X matrix, the first rated position marks X1 (2) and the last rated position marks X2 (3). The same occurs for matrix Y in which the first rated position marks Y1 (4) and the last rated position marks Y2 (5).

```

public void GetLimits(WritableBitmap image) {
    var p = image.Pixels;

```



```

var w = image.PixelWidth;
var h = image.PixelHeight;
X1 = w - 1; Y1 = h - 1; X2 = 0; Y2 = 0;
1 int limitMinimumX = w / 20;
  int limitMinimumY = h / 20;
  for (int x = 0; x < w; x++) {
    if (histX[x] > limitMinimumX) {
2      if (x < X1) { X1 = x; }
3      if (x > X2) { X2 = x; } } }
  for (int y = 0; y < h; y++) {
    if (histY[y] > limitMinimumY) {
4      if (y < Y1) { Y1 = y; }
5      if (y > Y2) { Y2 = y; } } }
  if (X1 == (w - 1) || Y1 == (h - 1)) {
    X1 = 0;
    Y1 = 0; } }

```

Implementation of toolkit

The toolkit was developed from the Integrated Development Environment (IDE) Visual Studio 2010 in a notebook with webcam and Windows 7 system. Visual Studio 2010 is a Microsoft manufactured IDE which may be used to develop a great variety of applications for several ends. A command console for programming and graphic tools is available. The application-example created to demonstrate the use of the toolkit was developed on the toolkit's IDE. Authorization and tests were performed on Internet Explorer 8 browsers and on Google Chrome 10 on the same notebook and on a desktop with webcam 2.0 and Windows 7.

Toolkit was developed on platform .NET and written in C# programming language so that it may be used in applications developed on framework Silverlight. C# is a strongly-typed programming language specific to objects and developed by Microsoft as part of platform .NET. Its syntax was based on C++ and influenced by Java

Framework Silverlight belongs to the platform .NET and allows the creation of Rich Internet Applications (RIA) applications. It is an alternative for Adobe Flash and Adobe Flex development tools with similar characteristics. The possibility of accessing the client's machine equipments, such as webcam, printer, microphone and others was incorporated as from the Silverlight 4.0 version. Microsoft-developed Silverlight currently performs on Windows and Mac, coupled to an implementation of Silverlight, called Moonlight, funded by Microsoft and Novel, which performs on Linux operational system.

RIA are Web applications with the characteristics of Desktop applications. Microsoft *Silverlight*, Adobe

Flash, HTML 5.0 are examples of tools for the development of RIA applications.

Further, .NET is a platform for the development and performance of systems and applications. Codes generated for .NET may be performed on any other device with framework .NET. Although currently it is used for Windows operational systems, parallel performances exist for the Linux and MacOS operational systems (MSDN, 2012).

Toolkit classes

The toolkit was developed in four classes. The three classes ARGB, RGB and YCbCr are tools for color spaces, and one class, FaceDetect, detects faces. Class ARGB has color constants in ARGB format so that conversion costs from RGB to ARGB could be avoided.

Class RGB stores colors in RGB color and has conversion methods for the ARGB format. Class YCbCr stores colors in YCbCr format with conversion methods for RGB format.

Class FaceDetect contains face detection algorithms implemented for this job, as shown at the beginning of the essay. There are other extra methods - such as ToGray and ToLine - that are study methods in the development of current analysis and have been maintained in the class for further implementation.

The following section will demonstrate and discuss the results obtained.

Results and discussion

The application is equipped with a screen that shows all the stages for face detection performed in current analysis (Figure 4).

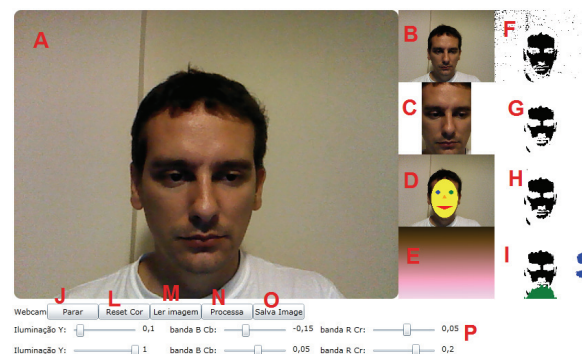


Figure 4. Screen of the application-example (Webcam image).

Area A equals area B – area A may receive any image from areas B to I: user must click on the desired area.

A B contains the image received from the webcam or read on the user's computer.

Area C contains only the face detected on the image.

Area D contains the main image with a SMILE on the detected face.

Area E represents the defined color spectrum of the skin.

Area F is the binary image after the skin filter.

Area G is the binary image after noise filtering.

Area H is the binary image after the expansion of the remaining pixels.

Area I is the graphic representation of the histogram.

Area J is the key to start or stop the webcam.

Area L is the key to re-establish standard configurations of area P on the color spectrum.

Area M is the key to read the image on the user's computer.

Area N is the key to process the image read on the computer.

Area O is the key to save the image on area D and to add a text on the image with information on the employed configurations used in the color spectrum.

Area P consists of the configurations of the color spectrum.

Libraries used: ImageTools and WriteableBitmap

Friendly access and image recording PNG and JPG in the client's machine were performed in the application-example by means of the library ImageTools available at <http://imagetools.codeplex.com>. Library ImageTools for Silverlight was employed to simplify image access. The latter is an open-code library that provides easy downloading and handling of images in different formats.

Images in format WriteableBitmap were handled in the toolkit and in the application. Since the original class provided in Silverlight does not have all the required methods for the implementation of current analysis but only simple functions to access image pixels, the open code library WriteableBitmapEx had to be employed. In fact, the latter is a collection of extension methods for Silverlight WriteableBitmap and is available at <http://writeablebitmapex.codeplex.com/>.

Color spectrum of the skin

A set of 13 images of people with different skin color shades (European, Asiatic and African) and in different backgrounds or complexes was used to obtain maximum and minimum value of YCbCr color space. The job was performed manually and individually for each image. The appendix show the result of the tests in all images used. It should be emphasized that as from image 06 color parameters in some images failed to fit in the predetermined

values – alterations in the red standard were detected. However, as previously remarked, detection may be efficient if adjustments in parameters are provided.

Analysis of images in adverse situations: complex background, small face and different illumination

Figures 5, 6 and 7 show that the complex background interfered in the noise filter when the image was processed with the pre-defined values. As a result, the binary image contains large areas which do not consist of faces.



Figure 5. Image with complex background.

Source: original image of Jui Huang, <http://www.mdemulher.abril.com.br>.



Figure 6. Image with complex background.

Source: original image of Ronaldinho Gaúcho, <http://www.extra.globo.com>.

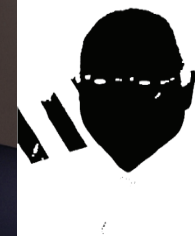


Figure 7. Image with complex background.

Source: original image of Dick Cheney, <http://www.wikimedia.org>.

Figures 8, 9 and 10 exhibit small-sized faces when compared with the entire image, the background is complex and the algorithms used are not adequate for this image type.



Figure 8. Image with a small face and complex background.
Source: original image of a woman wearing winter clothes, <http://www.topmulher.com>.



Figure 9. Image with a small face and complex background.
Source: original image of Demi Lovato, <http://www.demibrasil.com>.



Figure 10. Image with a small face and complex background.
Source: original image of Miley Cyrus, <http://www.mileycyrusworld.org>.

Figure 11 shows a face with good front illumination and scanty illumination in the background. The background is not considered complex owing to the scanty illumination of the background. Face detection is easy.

Figure 12 shows a face with good front illumination; the background also exhibits good

illumination – background colors are emphasized and the background becomes somewhat more complex.

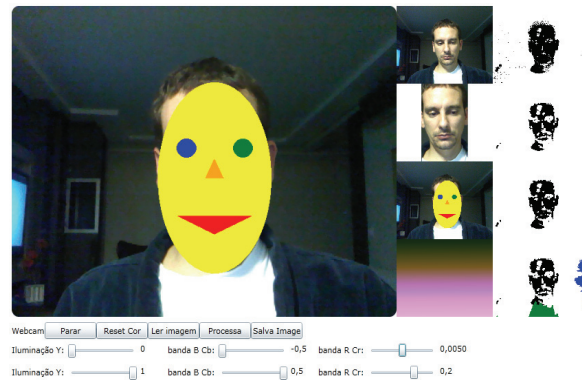


Figure 11. Image with front illumination (Webcam image).

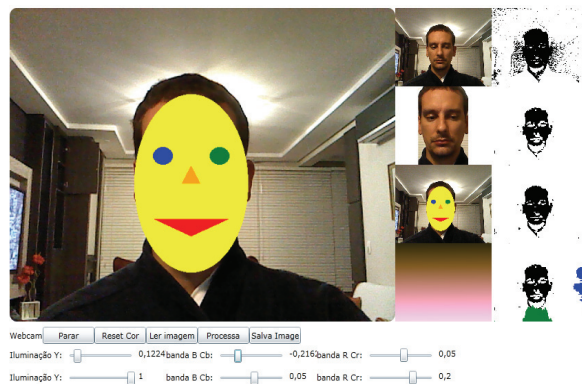


Figure 12. Image with total illumination (Webcam image).

Figure 13 shows a face with scanty front illumination although the background has good illumination. The face becomes a shade and face detection is possible by adjustments of parameters.

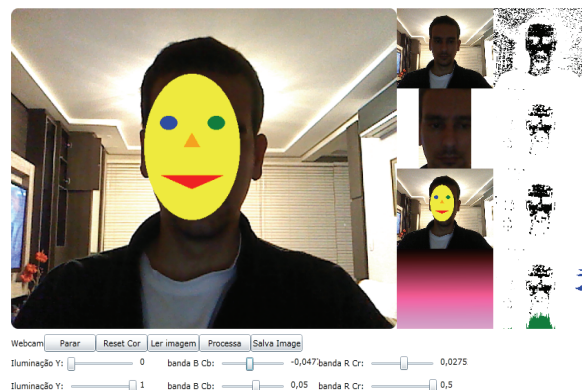


Figure 13. Image without front illumination (Webcam image).

Figure 14 exhibits face and background with almost no illumination; a small focus of light exists on the left side of the image; the focus of light is sufficient for face detection.



Figure 14. Image with low side illumination (Webcam image).

Conclusion

The toolkit for face detection was developed to show its functionality and an application-example that explored all its functionality and performance was provided.

Some considerations are made from the results of current experiment and research. Even when a small image base is used, the algorithm for people with different skin colors, ranging from white to black, was efficient. Complex backgrounds are the main difficulties involving the algorithm. Moreover, it was not applied to find more than one face per image. Image processing time was almost imperceptible in the tested machines and might be classified as a real time system.

Further experiments with the toolkit requires the addition of algorithms to separate the areas of possible faces in the image and face validation, elimination of complex background issues and the problem of more than one face in the same image.

Algorithms for face detection may be put into operation for images and for videos and may be used in log-ins for sites or for access to equipments, search for people in image libraries and for the recognition of people in video records at real time.

References

- AZUMA, R. Tracking requirements for augmented reality. **Communications of the ACM**, v. 36, n. 7, p. 50-51, 1993.
- BORCK, J. R. **Microsoft Silverlight 4 vs. Adobe Flash 10.1. InfoWorld**. Available from: <<http://www.infoworld.com/d/developer-world/infoworld-review-microsoft-silverlight-4-vs-adobe-flash-101-260>>. Access on: Oct. 18, 2010.
- CAI, J.; GOSHTASBY, A. A. Detecting human faces in color images. **Image and Vision Computing**, v. 18, n. 1, p. 63-75, 1999.
- COSTA, A.; RODRÍGUEZ, A. G.; SIMAS, E. P. L.; ARAÚJO, R. S. **Lógica fuzzy: conceitos e aplicações**. Relatório. São Leopoldo: Universidade do Vale do Rio dos Sinos, 2005.
- DAI, Y.; NAKANO, Y. Face-texture model based on SGLD and its application in face detection in a color scene. **Pattern Recognition**, v. 29, n. 6, p. 1007-1017, 1996.
- JENG, S.-H.; HONG, Y. L.; CHIN, C. H.; MING, Y. C.; YAO, T. L. Facial feature detection using geometrical face model: An ancient approach. **Pattern Recognition**, v. 31, n. 3, p. 273-82, 1998.
- LEE, J.; KUO, Y.; CHUNG, P.; CHEN, E. Naked image detection based on adaptive and extensible skin color model. **Journal of Pattern Recognition Society**, v. 40, n. 8, p. 2261-2270, 2007.
- LIN, C.; FAN, K. C. Triangle-based approach to the detection of human face. **Pattern Recognition**, v. 34, n. 6, p. 1271-1284, 2001.
- LOPES, E. C. **Deteção de faces e características faciais**. Porto Alegre: PUCRS, 2005.
- LOW, E. H.; KEE, B. Face detection: a survey. **Computer Vision and Image Understanding**, v. 83, n. 3, p. 236-274, 2001.
- MORAES, D. A. O. **Algoritmo para suavizamento de imagens digitais por filtragem Gaussiana em connection machines**. Porto Alegre: UFRGS, 2006.
- MSDN-Microsoft Developer Network. **Informações sobre .NET**. Available from: <<http://www.microsoft.com/net, 2012>>. Access on: Oct. 12, 2010.
- SCHULTE, R. **FaceLight – Silverlight 4: Real-Time face detection**. Available from: <<http://www.rene-schulte.info/>>. Access on: Mar. 9, 2011.
- SINGH, S. K.; CHAUHAN, D. S.; VATSA, M.; SINGH, R. A robust skin color based face detection algorithm. **Tamkang Journal of Science and Engineering**, v. 6, n. 4, p. 227-234, 2003.
- SUNG, K. K.; POGGIO, T. Example-based learning for viewbased human face detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 20, n. 1, p. 39-51, 1998.
- VEZHNEVETS, V. **Face and facial features tracking for natural human-computer interface**. Available from: http://www.graphicon.ru/2002/pdf/Vezhnevets_En_Re.pdf>. Access on: Nov. 14, 2010.
- VIJAYANANDH, R.; BALAKRISHNAN G. Hillclimbing segmentation with fuzzy C-means based human skin region detection using Bayes rule. **European Journal of Scientific Research**, v. 76, n. 1, p. 95-107, 2012.
- ZAIDAN, A. A.; ABDUL KARIM, H.; AHMAD, N. N.; MAHABUBUL ALAM, G.; Z Aidan, B. B. A new hybrid module for skin detector using fuzzy inference system structure and explicit rules. **International Journal of the Physical Sciences**, v. 5, n. 13, p. 2084-2097, 2010.

Received on June 14, 2011.






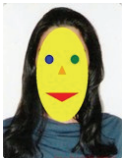






Accepted on May 29, 2012.

License information: This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Appendix 1












Obtained results

Table shows that most faces are detected by the following color spectrum:
Data:
Minimum value: Y = 0.10 Cb= -0.25; Cr = 0.05 and
Maximum value: Y = 0.10 Cb= -0.25; Cr = 0.05 and

	Photo	Detection configuration	Configuração					
			Y		Cb		Cr	
			Min	Max	Min	Max	Min	Max
1	 Imagem de Amori http://www.noticiashospitales.com.br		0.1	1	-0.25	0.05	0.05	0.2
2	 Carla Martins http://www.ofthalmocenter.med.br		0.1	1	-0.25	0.05	0.05	0.2
3	 Foto 3X4 http://www.realmadridwallpapers.com		0.1	1	-0.25	0.05	0.05	0.2
4	 George W. Bush http://www.georgewbushlibrary.gov		0.1	1	-0.25	0.05	0.05	0.2
5	 Ivan http://www.cidadesaopaulo.olx.com.br		0.1	1	-0.25	0.05	0.05	0.2
6	 Jui Huang http://www.mdemulher.abril.com.br		0.1	1	-0.25	0.05	0.09	0.2
7	 Indio da Costa http://www.blogdolobo.com.br		0.1	1	-0.25	0.05	0.05	0.2

Continue...

...continuation

	Photo	Detection configuration	Configuração					
			Y		Cb		Cr	
			Min	Max	Min	Max	Min	Max
8	 Elbridge Thomas Gerry http://www.answers.com		0.1	1	-0.25	0.05	0.05	0.2
9	 Ronaldinho Gaúcho http://www.extra.globo.com		0.1	1	-0.25	0.05	0.09	0.26
10	 Vestido de inverno http://www.topmulher.com		0.15	1	-0.25	0.05	0.02	0.18
11	 Demi Lovato http://www.demibrasil.com		0.35	0.75	-0.25	0.05	0.16	0.18
12	 Miley Cyrus http://www.mileycyrusworld.org		0.1	0.62	-0.25	0.05	0.05	0.18
13	 Dick Cheney http://www.wikimedia.org		0.41	1	-0.25	0.05	0.05	0.18