Highlight Image Filter Significantly Improves Optical Character Recognition on Text Images

Iulia Știrb
Computer and Software Engineering Department, “Politehnica” University of Timișoara, România

Abstract - Image filtering is the change of the appearance of an image by altering the shades and colors of the pixels. Increasing the contrast as well as adding a variety of texture, tones and special effects to images are some of the results of applying filters. In order to obtain a high success rate of OCR (Optical Character Recognition) on images which contain text, the main target of filters is, however, reducing the noise of the image. Within Silicon and Software System Limited (S3Group) Company from Dublin, Ireland, the OCR is part of the process of testing the menu options displayed on the output video of the set-top box under test. The important role played by image filters in improving a subsequent OCR process on text images was the reason I created two new non-linear efficient filters that will be presented in this paper. The first image filter that is named Smart Contrast, increases the contrast of the image in a way depending on the value of each component (Red, Green and Blue) of each pixel in the image. The second image filter, called Highlight, produces an outstanding increase of OCR success rate on the filtered images. As Highlight filter is carried out, the implementation differs from all other known filters, while the visual effect on the filtered images can be described as a combination between increased contrast as said before with other two visual effect: sharpening and highlighting the edges (of the characters). Precisely, combining these specific visual effects on the resulting image makes Highlight filter so powerful in improving OCR on text images.

Keywords: image, filter, highlight, sharpen, contrast, OCR

1 Introduction

Many often, the choice of the image filter is done non-automatically, by humans, as a result of observing the characteristics of the image (color, shape or thickness of text in the image). Researches reveal that specific filters are used for certain images. For instance if the original image is blurred and the expected result is an image with a higher clarity than Sharpen filter can be used and on the other hand, if a less level of details is desired in the resulting image, than Blur filter would be the right choice.

As in [2], the process of selecting the scale of a filter in order to perform edge detection over the image can be automated. Overall, as in [1], researches have been carried out regarding the automated selection of the filters’ parameters. In other words, once the proper filter to apply to the image has been choose by humans, the filter parameter is selected by a computer depending on the desired output image.

However, an automated analysis of the image properties in order to select the proper filters to be applied to it would be a complicated, expensive and time consuming process since the analysis depends on many factors (e.g. noise, clarity or contrast of the input image).

The image filter which I named it “Highlight” is designed to be a universal filter for improving OCR rate of success on a large variety of text images and because of its large applicability it avoids the automated selection of the proper filter to be applied to a specific image.

The paper contains one major section in which Highlight new image filter and some already existing filters are described. Snippets of code from Highlight filter implementation are also shown in this section. Conclusions section presents the major benefit Highlight image filter brings in improving the success rate of OCR in comparison to other filters and describes the visual effect of Highlight filter on images.

1.1 Properties of Highlight image filter

Highlight image filter detects the edges of the features in the image i.e. edges of the characters in a text image, highlights and sharpens the text and increases the contrast of the edges of the characters in a way similar to Smart Contrast image filter.

What Highlight filter brings new regarding the way contrast is done is that it performs a selection between two types of transformation and choses the proper one to be applied. Instead of simply applying the same transformation to all components of each pixel like Contrast filter does, the selection is done for each component (e.g. Red) of each pixel. This improvement made by Highlight image filter regarding the way contrast is increased produces even more contrast between the text in an image and the background in cases when this is needed such as when the colour of the text is close to the colour of the background making the text more visible than it would be by simply applying Contrast filter.

2 Description of Smart Contrast image filter

Smart Contrast filter compares the value of each component (e.g. Red) of each pixel with 127 (255 / 2, note that the range in which the components Red, Green
and Blue vary is 0 - 255) and if the value is less if less than 127 perform a certain transformation to that specific component, if greater perform a different transformation. As a case study, if it is considered an image which contains some text and the color of the text would be (R_G,B) = (126, 126, 125) and the color of the background would be (R_G,B) = (130, 137, 136), then the colors would be pretty similar, so the text would be hard to recognize even for the human eye. In this case, Smart Contrast filter decreases more the color of the text and increases more the color of the background than Contrast filter would do, making the text more visible and more easily to be detected. Thus, Smart Contrast keeps the good work Contrast filter does and, in addition, produces good results for edge cases.

2.1 Important step in creating Smart Contrast filter: knowing how Contrast filter works

Contrast filter is based on the transformation in (1) where contrast is the contrast scale (the degree to which the contrast is increased) and red, green and blue are the values of the components of a pixel.

\[
\begin{align*}
((\text{red}/255.0-0.5)*\text{contrast}+0.5)*255.0 & \quad (1a) \\
((\text{green}/255.0-0.5)*\text{contrast}+0.5)*255.0 & \quad (1b) \\
((\text{blue}/255.0-0.5)*\text{contrast}+0.5)*255.0 & \quad (1c)
\end{align*}
\]

The graphic representation for transformation in (1) is represented in Fig. 1 (blue plot). In the same figure, the identity function it is also drawn (green plot) to spot how pixel components values increase or decrease according to the transformation. If the value of the pixel component is less than zero it is set to zero and if it is greater than 255 it is set to 255.

So far, the same transformation is being applied to all components of each pixel. Thus, this is how Contrast filter performs, however the property is also specific to many other filters (e.g. Invert, Color).

2.2 New image filter: Smart Contrast

Smart Contrast performs two similar transformations depending on the value of the pixel component. For less than 127 (255 / 2) values the formula is illustrated by:

\[
((\text{value}/255.0-0.6)*\text{contrast}+0.6)*255.0. \quad (2)
\]

Equation (3) shows the formula for values of the pixel components greater than 127.

\[
((\text{value}/255.0-0.4)*\text{contrast}+0.4)*255.0 \quad (3)
\]

Value 127 is the threshold for Smart Contrast filter and the value of the threshold is chosen to be the median value from the range 0 to 255 (i.e. 0 is lowest value and 255 the highest value for color intensity).

As a remark, the same pixel could be the result of applying two types of transformations to its components (e.g. Red, Green or Blue). For instance, we could focus on an arbitrary pixel that has the coordinates (x,y) relative to the upper-left corner of the image. By assuming that (2) is applied to the Red component of that pixel and (3) is applied to the Blue component of the same pixel, we are facing a possible situation that could arise in the algorithm. Despite this fact exemplified before, no more than one transformation will be applied to a single component of a pixel (e.g. Red component could not possibly be the result of applying (2) and (3), it will have to be either (2) or (3), but not both).

Furthermore, two different pixels could be the result of applying different transformations to the same component of the two pixels (e.g. the filtered Red component of the first pixel that has the coordinates (x1,y1) could be the result of applying (2) and the filtered Red component of the second pixel that has the coordinates (x2,y2) could be the result of applying (3)). The graphic representation of the two transformations is shown in Fig. 2A, together with the identity function that helps in spotting the way pixel components are increased or decreased. Fig. 2B highlights the difference between Contrast and Smart Contrast algorithms. If the value of the pixel component is less than threshold 127 the blue plot describes the transformation that is applied to that certain component, else the transformation shown in the red plot is the one applied to the component.

Best OCR rate of success for the filtered images using Smart Contrast filter is produced when contrast scale is set to 1.5.

2.3 Visual result of applying Smart Contrast on images

Smart Contrast filter produces the results shown in Fig. 3. The results produced by Contrast filter are also shown in Fig. 3 in order to spot the improvements made by Smart Contrast. The effect of applying Smart Contrast filter would be that, in most of the cases, contrast is increased in the areas of the image where characters appear. Exceptions occur when the color of
the characters is close to the color of the background and both are close to either the lowest color intensity either the highest color intensity. This drawback is solved in Highlight filter.

3 Highlight image filter description

This filter decreases more the values of pixel components (i.e. Red, Green and Blue) that are less than 127 (using (2)) and increases more the values greater than 127 (using (3)) than Contrast filter would do. OCR benefits from Highlight filter’s improved way of contrasting the image (which is similar to the way Smart Contrast filter performs) and from the other two properties that are sharpening and highlighting the edges of the features in the image (e.g. the characters).

Highlight filter detects the areas of rapid intensity change (i.e. edges) like Laplacian of Gaussian filter would do. Once the edges are detected, they are being sharpened, which would produce a visual effect that is similar to what Sharpen filter would do to an image. Still, the implementation of Highlight image filter has no similarities with Sharpen and Laplacian of Gaussian filters’ implementation.

In addition, Highlight filter creates shadows behind characters (the color of the shadows contrasts with the color of the characters).

All this combined properties of contribute to a better success rate of OCR on text images.

3.1 Important step in creating Highlight image filter: understanding the effect of Sharpen and Laplacian of Gaussian filters on images

Smart Contrast produces an effect similar to what Contrast does, that is increasing the contrast of the image. In addition, Smart Contrast decreases more the values of pixel components that are less than 127 and increases more the values greater than 127 than Contrast would do. Highlight filter contrasts the image in a way similar to Smart Contrast.

Sharpen filter accentuates edges, but it does as well with the noise, as in [3], which is undesired and could make the OCR produce worse results than with the unfiltered image. Highlight filter takes the concept of spotting the edges from Sharpen filter, but does not accentuate the noise.

Laplacian of Gaussian combines the effects of Laplacian filter and Gaussian filter (which blurs the images to reduce the sensitivity to noise). While Laplacian detects the regions of rapid intensity change therefore being used in edge detection, Laplacian of Gaussian sharpens edges between two regions of uniform color but different intensities, as in [4].

3.2 New image filter: Highlight

Highlight filter gather together visual effect similar to the ones produced by Contrast, Sharpen and Laplacian of Gaussian filters. The implementation is carried out in an original manner using no template convolution (masks) like the last two filters do.

Highlight filter performs a different contrast increase for each component of each pixel in the image. A component - let’s take as an example the Red component - of the current pixel is increased or decreased depending on the value of the Red component of the current pixel and the two other filled with Red color pixels as in Fig. 4 which shows a small 3 x 3 area within an image. The pixels which are not filled with Red color in the same figure do not contribute to the new value of the Red component of the current pixel.

Pixels which contribute to the new value of the current pixel in Fig. 4 are placed in a diagonal manner. Because of this, both vertical and horizontal edges of the characters are detected.

For each component of each pixel a contrast scale is computed separately. The way contrast scale of each component of the current pixel (x,y) is computed, is emphasized in (4), where r, g and b

Figure 2. A. Smart Contrast transformations (blue and red plot) and the identity function (green plot)  
B. Smart Contrast (blue and red plots) and Contrast transformations (green plot)

Figure 3. The visual effect of Smart Contrast image filter
indicate the Red, Green and Blue components and \( x_1y_1 \), \( x_2y_2 \) indicate the pixels with coordinates \((x - 1, y - 1)\) and \((x - 2, y - 2)\). For instance, \( r_{x_1y_1} \) is the value of the Red component of the pixel with coordinates \((x - 2, y - 2)\).

\[
\begin{align*}
(100 + |r_{x_1y_1} - r_{-x_1y_1}| + |r_{x_2y_2} - r_{-x_1y_1}|)/100 & \quad (4a) \\
(100 + |g_{x_1y_1} - g_{-x_1y_1}| + |g_{x_2y_2} - g_{-x_1y_1}|)/100 & \quad (4b) \\
(100 + |b_{x_1y_1} - b_{-x_1y_1}| + |b_{x_2y_2} - b_{-x_1y_1}|)/100 & \quad (4b)
\end{align*}
\]

Equations \((4a-c)\) produce values in the range from 1.0 to 6.1 and are not applied to the left and top edges of the image. Once the contrast for each component of each individual pixel in the image has been recorded, the algorithm is ready to be applied.

The starting point of Highlight filter algorithm is based on the fact that human eye is sensitive to a difference of at least 30 between the values of at least one of the same component of two adjacent pixel when it comes to perceive and recognize characters.

To be more specific, if we would have to write some characters on a background which is uniformly colored with the intensity \((R_0,G_0,B_0) = (0,0,0)\), the color of the text would have to be \((R_n,G_n,B_n) = (0 + 40,0 + 31,0)\) or \((R_n,G_n,B_n) = (0,0 + 31,0)\) or any other combination that would meet the above request, in order for the human eye to recognize what is written. Fig. 5 proves what is being said before.

A reasonable assumption that is made from the start is that characters that would not be perceived by the human eye, are not expected to be recognized by a machine using OCR, but every character perceive by the human eye must be also recognized by a machine (assuming there is no noise), or at least expect to be recognized. Thus, for the text that could not be perceived by the human eye, the performances of OCR are not improved by filtering first the text image using Highlight filter.

The core of the algorithm is described in the following. Since an example makes the general case more explicit, let’s bring into discussion the Blue component of the pixel having the coordinates \((x, y)\) relative to the upper-left corner of the image.

If the absolute difference between the value of the Blue component of the pixel with coordinates \((x - 1,y - 1)\) and the value of the Blue component of the pixel having the coordinates \((x,y)\) is greater than 15 and the absolute difference between the value of the Blue component of the pixel with coordinates \((x - 2,y - 2)\) and the value of the Blue component of the pixel with coordinates \((x - 1,y - 1)\) is also greater than 15 than a certain transformation will be applied to the Blue component of the pixel having the coordinates \((x - 2,y - 2)\) (for \(x\) and \(y\) greater than 2).

The requests described before and shown in (5) will be as well tested separately for the other two components i.e. Red and Green of each pixel in the image, except for the pixels in the right and bottom edges of the image (those will not be filtered). This remark is valid for (5), but as well for all the following formulas.

\[
\begin{align*}
|r_{x_1y_1} - r_{-x_1y_1}| > 15 \& \& |r_{x_2y_2} - r_{-x_1y_1}| > 15 & \quad (5a) \\
|g_{x_1y_1} - g_{-x_1y_1}| > 15 \& \& |g_{x_2y_2} - g_{-x_1y_1}| > 15 & \quad (5b) \\
|b_{x_1y_1} - b_{-x_1y_1}| > 15 \& \& |b_{x_2y_2} - b_{-x_1y_1}| > 15 & \quad (5c)
\end{align*}
\]

Why 15 \((30/2)\)? Let’s assume that the text in an image has some noise around it and the transition between the color of the text and the color of the background is done through an intermediary pixel which could be called “noise pixel” which has a different color. This situation appears frequently in the real scenarios. Then, the minimum difference of 30 between the values of the color component of the background and the color component of the text could be spread among three adjacent pixels (place as the colored ones in Fig. 4) with three different colors instead of just two adjacent pixels i.e. “text pixel” and “background pixel” as in Fig. 5. For instance, if the intensity of the color of the background would be \((R_0,G_0,B_0) = (0,0,0)\), the intermediate color (noise color) would have to be at least \((R_n,G_n,B_n) = (0,0,16)\) and the text color would have to be at least \((R_n,G_n,B_n) = (0,0,32)\), for the condition in (5c) to be fulfilled.

Because of the diagonal manner in which the pixels which contribute to the new value of the current pixels are placed, meaning that the direction of the gradient is a diagonal direction, all the noise around curve edges that follow this direction is eliminated. Noise around vertical and horizontal edges, which form an angle of \(-45^\circ\) and \(45^\circ\) with the diagonal direction, is as well almost eliminated. Anyway, the noise affects more the curve edges than the horizontal and vertical edges.

Fig. 6 shows the visual representation of all possible cases that fulfill the condition in (5c), which refers to the Blue component. The visual representation for the Red and Green component can be obtained in the same way.
Before getting to the point where a specific transformation is applied, another condition, in addition to the one described in (5c), must be first met. The new condition is described in (6c).

If the conditions in (6c) are fulfilled, there is a diagonal “blue” gradient. If (5c) and (6c) are met then the transformation described in the first paragraph of Section II is applied: to the Blue component of the current pixel so, therefore, the contrast for the Blue component is increased depending on the result of the comparison with 127 (255/2).

\[
|r_{x_2}y_2 - r_{x_1}y_1| > 30 \\
|g_{x_2}y_2 - g_{x_1}y_1| > 30 \\
|b_{x_2}y_2 - b_{x_1}y_1| > 30
\]

(6a)  
(6b)  
(6c)

Fig. 7 spots the two of the six possible cases shown in Fig. 6, showing the visual representation of the situations when condition in (6c) is met.

If the condition in (6c) is not fulfilled then the intensity of the Blue component will be increased or decreased depending on whether the condition in (7c) described below is met or not. To be more specific, if the condition in (7c) is fulfilled, the intensity of the Blue component is increased using a transformation in (2) and in the opposite case the intensity is increased using transformation in (3). Both transformations are applied, this time, regardless of the value of the Blue component.

\[
|r_{x_2}y_2 - r_{x_1}y_1| > 0 \\
|g_{x_2}y_2 - g_{x_1}y_1| > 0 \\
|b_{x_2}y_2 - b_{x_1}y_1| > 0
\]

(7a)  
(7b)  
(7c)

Transformations in (2) and (3) are applied to the value of each component i.e. value variable and the result depends on the contrast scale i.e. contrast variable.

Fig. 8 spots the two of the 4 left possible cases shown in Fig. 6 (the first two of them were already discussed and matched with the condition in (7c)). More exactly, Fig. 8 shows the visual representation of the cases which fulfill the condition in (7c).

In case the condition in (7c) is not fulfilled, the two left cases that were not discussed before, out of six spotted in Fig. 6, are shown in Fig. 9.

It was never said before what happens if the condition (5c) is not met and this will be the appropriate time to be speaking about this. Well, if the condition is not fulfilled another condition is being tested and shown in (8c).

If neither (8c) is fulfilled no transformation will be applied to the Blue Component of the pixel with coordinates (x-2,y-2). If (8c) is met, the condition in (7c) will be tested again. The visual representation of the cases that meet the condition in (8c) is shown in Fig. 10.

3.3 Visual results of applying Highlight filter on images

Highlight image filter produces the results in Fig. 11. It can be seen how this filter detects the edges of the characters and sharpens them (the best example would be “ALL CHANNELS” image) and how it creates contrasting shadows behind the characters (the black shadows can be best seen on white colored “Golf: Women’s British Open” text in the image).

Because of the shadows behind the sharpened edges of the characters and because of the increased contrast in the edges, the characters appear to be highlighted in the filtered image, which is the main visual effect of Highlight image filter.

3.4 Highlight image filter program code and visual representation

Part of the C# code that corresponds to Highlight image filter is being listed in Code 1. buffer array stores the image representation, more exactly the Blue, Green, Red and Alpha components in this order for the first pixel, then the components for the second pixel in the same order and so on for the rest of the pixels in the image.

Code 1. A part of the Highlight image filter algorithm

```csharp
public void EdgesIntensityChange(byte[] buffer, double[] contrastBuffer, int Stride, int k)
{
    int diff01 = buffer[k - Stride * 2 - 8] - buffer[k - Stride * 4];
    int diff02 = buffer[k - Stride * 4] - buffer[k];
    int diff03 = buffer[k - Stride * 2 - 8] - buffer[k];
    // ...
}
```
The visual representation of the algorithm is shown in Fig. 12. The six cases next to the first if are the visual representation of the condition. In other words, if the condition in (5c) is met we will found ourselves in one of the six possible cases. So far for the rest of the figure, the possible cases that meet the conditions are shown next to the if-s and else-s. like it is also in the case of the second if, for which its condition is represented visually by two out of six possible cases.

4 Conclusions

I recommend the usage of Highlight filter rather than other filters in situations when the image contains areas of narrow text (but sure you can successfully use it also when the text in the image is wide). After applying this filter in the situation mentioned above, the success rate of OCR on the filtered image is considerably increased. Probably the most important thing to mention regarding Highlight image filter is that it eliminates the noise in the image, more exactly, it focuses on eliminating the noise located all around the edges of characters in the text image. Beside the other actions (sharpen, contrast, highlight) of this filter, the diagonal gradient direction also contributes to removing the noise from the filtered image. The image could be also a bit blurred (not too much) and still, the OCR is improved.

In few words, Highlight filter determines outstanding OCR results on text images in which:

- Text is narrow
- Noise is present (could be around characters)
- Any other situation (e.g. lack of contrast, too much blurring)

The effect of Highlight image filter is detecting the edges and once detected it sharpens them and increases their contrast. As a result, it highlights the edges. The visual effect on characters which are present in the image would be sharpening them and increasing their contrast, creating shadows (behind them) that contrast with their color and obviously highlighting them.

In conclusion, the visual result of Highlight filter on characters is briefly described as an appropriate combination of the following visual effects, which contribute together to OCR increased performances:

- **Sharpen**
- **Contrast**
- **Highlight**

Many techniques, such as adaptive restoring of text image, have been tried, as in [5]. Image filtering has also made an improvement in important areas, such as medicine, as described in [6], but lately,
improving OCR performance using filtering has become and will be a great challenge. When developing Smart Contrast, which is a non-linear image filter, I had as a starting point the Contrast filter that was already implemented within Silicon & Software Systems Limited (S3Group) Company. Beside Smart Contrast’s incontestable performances in improving OCR, this filter was actually just the triggering point for my following creation, namely Highlight image filter, as well non-linear, which overcomes the challenge of performing OCR with a very high success rate on text images.

5 References


[5] P. Stubberud1, J. Kanai, V. Kalluri, Adaptive image restoration of text images that contain touching or broken characters, 1Nevada University, Las Vegas, United States, 1995