Abstract

In this work, we present a novel non-photorealistic rendering method which produces good quality stylization results for color images. The procedure is driven by saliency measure in the foreground and the background region. We start with generating saliency map and simple thresholding based segmentation to get rough estimation of the foreground-background mask. We improve this mask by using a scribble-based method where the scribbles for foreground-background regions are automatically generated from the previous rough estimation. Followed by the mask generation, we proceed with an iterative abstraction process which involves edge-preserving blurring and edge detection. The number of iterations of the abstraction process to be performed in the foreground and background regions are decided by tracking the changes in saliency measure in the foreground and the background regions. Performing unequal number of iterations helps to improve the average saliency measure in more salient region (foreground) while decreasing the average saliency measure in the non-salient region (background). Implementation results of our method shows the merits of this approach with other competing methods.

CCS Concepts

• Computing methodologies → Non-photorealistic rendering; Image-based rendering; Image processing;

Keywords

Non-photorealistic rendering; Saliency; Image Abstraction; Guided Filter

1. Introduction

The human visual system (HVS) is selectively sensitive to certain features of the incident light. Expressive rendering deals with emphasizing the features of light to which the HVS is sensitive to. The most important aspect is abstraction which is the process of removing not so important details while retaining the important ones. Edges, which include silhouettes and contours, play an important role in the abstraction process [16]. HVS draws most of the information from the more sensitive regions called salient regions in an image. Hence while doing abstraction on images, it is essential to retain more information in these salient regions by performing lesser abstraction and retaining lesser information in other regions by doing more abstraction. Our image abstraction framework is based on this idea. Consider a scene with distinct foreground (FG) and background (BG) regions. The first step in our approach is finding saliency in different regions followed by simple thresholding based segmentation to get a rough idea about the Foreground-Background (FG-BG) mask. Among the many saliency methods proposed in literature, the method proposed in [20] provides a good distinction between FG-BG saliency values and thus helpful in thresholding based segmentation. The poor segmentation achieved by simple thresholding on the saliency map is improved by using one-cut algorithm [29]. This algorithm is based on the grabcut algorithm which requires scribbles in the FG-BG regions as constrains [2]. To avoid user interaction for providing scribbles, we use simple morphological operations to generate FG-BG scribbles automatically from the previous rough estimation of FG-BG mask. These entire efforts provide an improved FG-BG mask. This FG-BG mask is used to perform non-uniform abstraction over the image and retain more details (by performing lesser abstraction) in FG region (which is more salient in most of the cases) than the BG region. Like many other abstraction approaches, our image abstraction method is also iterative in nature and iterations are performed using guided filter which performs edge preserving blurring [14]. The number of iterations to be performed is decided automatically by how the average saliency values are changing in the FG-BG regions rather than deciding based on subjective quality only. While performing iterations we keep track of changes in the saliency measures in both the FG and BG regions. Finally to stylize the abstraction result, we add the edges back to abstracted image using the contour detection algorithm proposed in [8]. We have tested our method on MSRA10K benchmark dataset images and for most of the images good stylization results are obtained within few iterations [5]. The proposed algorithm is well suited for stylizing images containing distinct FG-BG regions or salient object(s).

Our key contributions are:

1. Automatic generation of FG-BG scribbles from the saliency map which are required in segmentation.
2. Non-uniform abstraction over FG-BG regions using guided filter as edge-preserving blurring filter.

3. Saliency based stopping criteria for iterative abstraction process.

The organization of paper is as follows. In the related work section, we provide the literature survey of relevant works performed prior to our work. In the proposed framework section, we present our algorithm for automatic segmentation, abstraction and stylization followed by the details of implementation in the implementation section. Finally, we discuss the results achieved and comparative study which includes visual observations along with subjective analysis with other stylization methods in the results and discussion section. This section is followed by conclusion section.

2. RELATED WORK

Many non-photorealistic effects come under the category of image stylization. The main aim of any stylization algorithm is to give artistic effect to a photograph. Most recent work in this domain is by Deep Dream Generator which use artificial intelligence for stylization [6]. It also can transform relevant effects from one image to other. Another state-of-the art work published by Prisma Labs which also uses artificial intelligence to modify predefined effects to best match the query images [25]. In ([31], [27]) authors have worked to generate pen-and-ink illustration from photographs. Authors in ([28], [10]) observed the interaction of pencil and paper to generate pencil drawings. Another popular effect that comes under the category of stylization is painting illustration and emphasising on prominent edges are in-general steps followed by results and supporting discussion in section. Each block in the figure 1 is explained in detail

Some of the earlier works related to image stylization used the bilateral filter as edge preserving filter along with suitable edge detection algorithm [30]. Winnemoller et.al. used bilateral filter for modeling and photo editing applications [22]. Orzan et al. used edge based approach and gradient domain image processing techniques to manipulate photographs in non-realistic manner [23]. Kang et al. describe the flow of salient features in an image based on shape/color filtering guided by vector field [17]. An inherent problem with bilateral filter is reversal of gradient. Since in many approaches ultimately edge responses are multiplied with abstraction output (without edges), the areas where gradient changes overlap with edge responses, gradient reversal does not become visible but they do show up if such overlaps are missing in some regions. Besides bilateral filter, abstraction based on Kuwahara filter, morphological filters and partial differentiation based methods are also used ([24], [26]). Gerstner et al. performed abstraction on images by pixalating the original image [11]. Another approach is the adaptive image abstraction algorithm where over-segmentation is performed followed by saliency driven adaptive smoothing [21]. We would like to provide an alternate solution to this image stylization problem deriving the necessity of such a solution from these earlier works.

3. PROPOSED FRAMEWORK

Figure 1 shows the flow diagram of our work. The figure shows the key steps in the stylization process and how they are achieved. Each block in the figure 1 is explained in detail in successive subsections. Section 3.1 explains how we obtain the rough nature of FG-BG mask. Section 3.2 explains how we automatically generate scribbles for the FG-BG to get improved FG-BG mask. Followed by this, section 3.3 gives an in-depth idea about how we perform abstraction with guided filter and how number of iterations are automatically decided based on the saliency measure changes. Further in section 4, we provide required implementation details followed by results and supporting discussion in section 5. Each section is supported with necessary intermediate results to illustrate the significance of different steps involved in the stylization process.

3.1 Rough FG-BG mask: Saliency based thresholding

The first step in the process is to get saliency map. Diff-
different saliency methods were explored to test which method best suits for the application ([1], [3], [34], [5]). In [5] authors have proposed their own saliency approach and compared their method with many existing state-of-art saliency algorithms. By comparing the results of different saliency methods we finalized to use method proposed in [20]. The saliency algorithm helps to get a rough nature of the FG-BG mask via simple thresholding as shown in the equation:

\[ S_{th}(i,j) = \begin{cases} 1 & S(i,j) \geq th \\ 0 & S(i,j) < th \end{cases} \]

where \( S(i,j) \) is saliency value at pixel \((i,j)\) in the saliency map and \( S_{th} \) represents thresholded map. Here threshold \( th \) is chosen as the average value of saliency in saliency map over complete image. Figure 3 shows these steps for a sample image from MSRA10K benchmark dataset [5].

### 3.2 Improved FG-BG Mask with one-cut algorithm

![Figure 3: Getting improved FG-BG mask: (a) Thresholded saliency map, (b) FG scribbles, (c) BG scribbles, (d) Output of one-cut, and (e) Improved FG-BG mask.](image)

One-cut algorithm proposed by Tange et al. requires user interaction to provide FG scribbles and BG scribbles as input along with original image to produce FG-BG mask([29], [2]). These scribbles are generated automatically from the thresholded saliency map(figure 3a). The automatic scribble generation works because saliency technique proposed in [20] produces more saliency values in most of the FG region and some surrounding region. In order to omit the surrounding portion of FG, thresholded saliency map is eroded by some amount to generate FG scribbles (figure 3b). To get the BG scribbles, we dilate the thresholded saliency map and take logical negation over complete matrix (figure 3c). For all the images tested with our algorithm, we used disk as the structuring element of size 25 for erosion and dilation. The output of this implementation is shown in figure 3d. To further improve this process, we reject small mask elements and fill the holes to get continuous mask (figure 3e).

### 3.3 Abstraction with guided filter

![Figure 4: Image abstraction flow diagram.](image)

Figure 4 shows the different steps carried out to perform abstraction. Red dashed line in the figure 4 denotes single iteration of the proposed abstraction process.

#### 3.3.1 Initial iterations

Abstraction process starts with converting RGB image to Lab space. We use guided filter on L channel [14]. Guided filter requires an input image whose properties will be changed according to another input called as guided image. When both the input image and the guided image are same, guided filter makes edge preserving blurring on an image [14]. Colors are added back to the output of the guided filter. In the abstraction process, edges play an important role. Also the position of edges, nature of edges (continues/broken and thick/thin) affect the quality of the final output. To detect edges, we use the contour detection algorithm proposed in ([8], [9], [35]). The algorithm gives nice continuous edges when there are depth changes, occlusions and avoids complete gradient dependency which is the case with popular edge detectors such as DoG, Canny, etc. We initially perform three iterations over the complete image. The further iterations are performed according to saliency changes.

#### 3.3.2 Iterations based on saliency changes

Initial iterations make uniform abstraction over the entire image. After each successive iteration, performed after initial iteration, we calculate the average saliency measure in FG and BG region using masks found previously (figure 3e). Since FG is of more interest in most of the images, we perform lesser number of iterations over FG and more in BG region. By tracking changes in the average saliency values in the FG and the BG regions, we stop the iterations:

- over the FG region when the average saliency in the FG region attains local maximum.
- over the BG region when the average saliency in the BG region attains local minimum and we have already achieved local maximum in the FG region.

The main advantage of non-uniform abstraction is that more details are retained in the more salient region i.e., the FG
region as compared to the BG region. The abstraction results are shown in figure 5 and the changes in saliency map are shown in figure 6. Also, the figure 7 shows how average saliency values in the FG-BG regions change with iterations after the first three iterations on both the FG and the BG regions. Red squares show the stopping values.

Figure 5: Abstraction result: (a) Original image, (b) After minimum iterations, and (c) Final abstraction.

Figure 6: Abstraction result: (a) Original saliency map, (b) Saliency map after minimum iterations, and (c) Final saliency map.

Figure 7: Variations in average saliency values: (a) Foreground region and (b) Background region.

4. IMPLEMENTATION

The proposed algorithm is implemented and tested on MATLAB version-2015a running on Windows 10 with Intel core i3 processor clocked at 2.49 GHz. Here, we present rough idea about the timing details of proposed algorithm on a color image of size 400×351. In the algorithm, the most time consuming portion is calculating saliency of the image which we have to do initially to obtain FG-BG mask as well as in each iteration of abstraction. It takes about 5 sec to the calculate saliency measure. Detecting edges using contour detection algorithm with already trained model takes about 3 sec which we have to do only once on the original image. Abstraction with guided filter takes very less time as compared to other portions. For an image under consideration, all 11 iterations (3 initial iterations plus 7 saliency based iterations) of guided filter took about 2 sec to complete the task. For a typical image of mentioned size, it takes about 41 sec including all the steps and the storage of intermediate and final results on hard drive. The computation time of algorithm is higher as compared to the implementations of algorithms proposed in [17], [33] but it is comparable with the algorithm proposed in [23].

Though the algorithm is slower to work as compared some of the state-of-art abstraction methods, it is fully automatic and requires very few parameters to vary in order to obtain good abstraction results for different images. Different parameter settings are discussed below.

4.1 Guided filter parameters

Guided filter implementation requires an input image and a guide image (which is the same as the input image in our case), neighborhood size $r$ and $\epsilon$ [14].

\[
I_{GF} = aI + b; a = \frac{\sigma^2}{\sigma^2 + \epsilon}; b = (1 - a)\mu
\]

In the above expressions, $I_{GF}$ represents filtered image for input image $I$ and the filtering is controlled by parameters $a$ and $b$. $\sigma$ represents variance and $\mu$ represents mean over a chosen neighborhood size $r$. From the expressions -

- higher the value of $\epsilon$, lower the value of $a$, lower will be the contribution in $I_{GF}$ from the original pixel values in $I$.
- lower the value of $a$, higher will be the contribution of $b$ i.e., mean of pixel values in the neighborhood of size $r$, more will be the blurring.

In our implementation, for all the images, we used $r = 3$ and $\epsilon = 5^2$.

4.2 Contour detection parameters

Contour detection algorithm provides final output in range of 0-1 [8]. Here, we need to set the threshold value to get only required amount of edges. A threshold value between 0.7-0.85 provides good result in our implementation. This is the only parameter which needs to be changed depending on the image under consideration.

5. RESULTS AND DISCUSSION

In this section, we show the comparison results of our method with the other state-of-art methods proposed in [33], [17] and [23]. Figure 8 and figure 9 show these comparisons. Here the comparison is done using 12 images. Methods proposed in [33] and [17] uses a color quantization which makes
the abstraction for some images (such as figure 8-1st and 3rd image and figure 9-2nd and 4th image) producing slightly better output than our method but still the outputs are comparable. Color quantization improves overall look of the above mentioned images but it creates unnecessary artifacts (especially in the BG region) which can be observed in the remaining abstraction results. Moreover the edges are better in the result using our method compared to the other three methods which can be prominently observed in the case of figure 8-5th and 6th image and figure 9-2nd and 4th image. The edge detection method used in our stylization framework enhances the quality of abstraction as well as edges in the FG region help to improve saliency measure. In very rare cases, the proposed framework produces small undesirable edges which can be seen in the case of figure 8-2nd image.

Along with this, we carried out a subjective analysis to compare our method with the above mentioned three methods. In the subjective analysis, every time a user performing the analysis is asked to rate all the four abstraction results between 1 to 10 (10 being the highest). While doing this user is asked to rate each abstraction result individually. Since the user can give same rating to two or more results, the user is also asked to rank each stylization result from rank 1 to rank 4 (rank 1 being the best out of all) for each image. For every new user, images appear in random order as well as for each image, the order in which abstraction result with particular method appear is also random. This avoids any discrepancy due to the same ordering of result images. For all the users, conditions such as monitor/display and light conditions are kept exactly same so that each result appear same to all the users. The study is carried out with 17 users same to all the users. The study is carried out with 17 users.

In the final stylization result. However, it is not very common to perform abstraction on such images according to the literature on image stylization.

### 6. LIMITATIONS

Here, we discuss few limitations which were realized during the implementation of the proposed approach. While deciding the number of iterations of abstraction process, we rely on the local maximum and the local minimum of average saliency values which might not be the global optima. Still the algorithm produces good quality stylization results for most of the images. Another limitation of the algorithm is when an input image does not have a distinct FG and BG, unequal abstraction may produce noticeable artifacts in the final stylization result. However, it is not very common to perform abstraction on such images according to the literature on image stylization.

### 7. CONCLUSION

We have presented a novel approach for image abstraction based on saliency measure. By combining the saliency based segmentation with one-cut segmentation algorithm and automatic scribble generation for the foreground and the background using simple morphological operations, we have successfully achieved good FG-BG segmentation. This mask is used to calculate saliency measure in the FG and the BG regions separately in each iteration of the guided filter. According to changes in average saliency value in these regions the number iterations of guided filter in the FG and the BG regions are decided. More ever we always perform the number of iterations in the BG region more than or equal to the number of iterations in the FG region. This helps to keep more details in more salient FG region and perform more abstraction in lesser salient BG region. Also edges obtained in our implementation are more prominent which emphasizes the overall stylization result. Good stylization results are obtained usually with just a few iterations.

We would like to improve the method by devising good optimization strategy which is able to provide the best saliency measure to stop the iterations in the foreground and the background regions. We would also like to evaluate the results obtained through this strategy on more number of subjects. We would like to speed up the proposed approach using fast saliency detection algorithms such as the one proposed in [34].

### 8. REFERENCES


**Table 1: Results of subjective analysis.**

<table>
<thead>
<tr>
<th>Method</th>
<th>No. of times rank 1 given</th>
<th>Average user rating</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method in [33]</td>
<td>48</td>
<td>6.43</td>
<td>2.001</td>
</tr>
<tr>
<td>Method in [17]</td>
<td>50</td>
<td>6.48</td>
<td>1.751</td>
</tr>
<tr>
<td>Method in [23]</td>
<td>60</td>
<td>6.58</td>
<td>1.783</td>
</tr>
<tr>
<td>Our method</td>
<td>78</td>
<td>6.7</td>
<td>1.763</td>
</tr>
</tbody>
</table>

In table 1, it can be seen that in both the cases, rating and ranking, the method proposed in the paper performs better than the other three methods. It can be observed that the proposed method gets the best rating 78 times which is significantly better than that of the other methods. The average rating is 6.7 which is also higher compared to that of the other methods. The third column in table 1, standard deviation, states the consistency of user rating. The lower the value of standard deviation more the consistent a particular method is. In consistency measure, the proposed method stands at the second position. In all the three measures the proposed method either performs best or is very near to the one which performs the best. The subjective analysis results shown in the table 1 as well as comparative results shown in the figure 8 and the figure 9 clearly shows that even though the proposed method does not out perform the other state-of-art methods by very large margins, the method is observed to produce visually appealing stylized images.
Figure 8: Comparison with different methods: First column—original image, Second column—method proposed in [33], Third column—method proposed in [17], Fourth column—method proposed in [23], and Fifth column—Our method.
Figure 9: More comparison results with different methods: First column-original image, Second column-method proposed in [33], Third column-method proposed in [17], Fourth column-method proposed in [23], and Fifth column-Our method.


