IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2024–05–14.
Submitted on 2023–06–28, accepted on 2024–04–18.
ISSN 2105–1232 © 2024 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2024.495

A Brief Analysis of the Generic Framework for the Structured Abstraction of Images

Noura Faraj[2], Lucía Bouza[1], Julie Delon[1]

[1]Université Paris Cité, CNRS, MAP5, France
[2]Université de Montpellier, LIRMM, France
(noura.faraj@umontpellier.fr, luciabouzah@gmail.com, julie.delon@u-paris.fr)

Communicated by Bruno Galerne and Luis Álvarez *Demo edited by* Lucía Bouza

Abstract

In this study, we present a simplified implementation of a versatile framework for structured image abstraction, as outlined in [Faraj et al., A Generic Framework for the Structured Abstraction of Images, International Symposium on Non-Photorealistic Animation and Rendering, 2017]. The framework relies on the topographic map, a hierarchical and geometric representation composed of all shapes of an image, organized in a tree structure. Within this framework, abstract renderings of digital photographs can be generated by iteratively applying simple local operations like replacement, removal, or rotation of shapes. These operations give rise to a diverse spectrum of renderings, spanning from geometrical abstraction and painting-like effects to style transfer. We perform a brief analysis of the results produced by the method, highlighting its quality and limitations.

Source Code

The reviewed source code and documentation for this implementation are available from the web page of this article[1]. Compilation and usage instructions are included in the README.md file of the archive.

Keywords: image processing; non-photorealistic image rendering; topographic map; image abstraction; image generation; style transfer

[1]https://doi.org/10.5201/ipol.2024.495

1 Introduction

Within the domain of non-photorealistic image rendering, image abstraction describes the procedure of producing simplified and stylized versions of photographs [3, 7]. In this article, we analyse and describe the generic framework for the structured abstraction of images proposed in [4]. This framework relies on the topographic map of images [2]. The topographic map is a hierarchical organization of all shapes of an image, forming a tree structure that captures inclusion properties. These shapes can be modified or manipulated in order to edit the input image. The original paper [4] suggests various shape modifications, such as displacements, scaling, blur, or substitution of each shape by another one taken from a set of reference shapes, giving rise to a diverse spectrum of image renderings, spanning from geometrical abstraction and painting-like effects to style transfer, all within the same framework. We refer to [4] for a more general introduction to non-photorealistic image rendering and description of other state-of-the-art methods.

In the following, we briefly describe the method and then perform an analysis of its results with some experiments. We use here a simplified version of the original implementation, with some additional functionalities.

2 Framework for Image Abstraction

The framework for the structured abstraction of images can be decomposed in three steps, as illustrated by Figure 1. First, the original image is decomposed in a colored tree of shapes. Then, all shapes undergo different geometric or color transformations. Finally, the final image is reconstructed from the set of modified shapes. These steps are described in detail in [4] and are only briefly summarized in the following. The use of the topographic map is the key idea of this framework, since it allows to modify shapes while naturally respecting the inclusion relations between objects in the image.

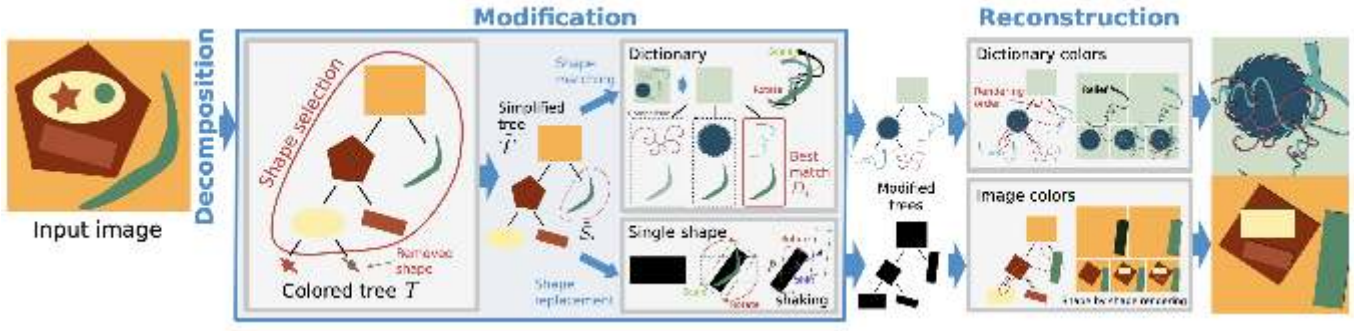

Figure 1: Different steps of the image abstraction framework. Initially, the framework decomposes the input image and computes its colored tree representation. Subsequently, the tree undergoes modification: the framework selects shapes and removes those that are either too small or too large based on predefined thresholds. The remaining shapes are then replaced, either by shapes extracted from a dictionary (top) or by geometric shapes (bottom). Additional adjustments can be applied, such as random displacement, smoothing, or transparency. The final step is image reconstruction, where the shapes are rendered onto the resulting image. This rendering can be done using either the color of the new shape (top) or the color of the original shape (bottom). Image taken from [4].

2.1 Decomposition

During this stage, the topographic map of the luminance channel (in an HSV color space) of the original image is computed using the Fast Level Set Transform [6]. This results in a tree structure

of all image shapes. For each shape **s**, the average color (in the RGB space) of all pixels included in **s** is computed and kept along with **s**.

In the demo, the user is given the option to draw a mask on the input image. If the option *use mask to simplify the input image* is chosen, the input image is segmented using [5], and all segments overlapping with the drawn mask are transformed into an average color of the image. This option is useful for instance to remove spurious texture or shapes in the background of an image, in order to highlight a foreground object. If the option is not chosen, the mask can also be drawn and used later in the modification stage, in order to apply two different abstraction styles to different regions of the input image.

2.2 Modification

During this phase, the input shapes undergo different transformations. A first optional but important step consists in keeping only meaningful boundaries [1] in the tree, which roughly reduces the list of shapes to the ones corresponding to contrasted edges in the image. Advanced parameters can also be modified if required, such as the minimum and maximum areas of the shapes in the tree (other shapes are removed), see Section 4 below. Once this selection has been made, shapes are modified depending on the desired task, as described below. Observe that several different shape modifications can be combined in practice (for instance abstraction with shaking).

Shape Abstraction. For this rendering, each remaining shape in the tree is replaced by a geometric primitive chosen by the user: a rectangle, a disk, or an ellipse. Original shapes can be preserved if desired.

Watercolor. To achieve this effect, each shape in the color tree of the input image is smoothed by a simple median filter. As a consequence, the smoothing is more pronounced at points with strong curvature, which results in a painting-like effect.

Shape Shaking. Each shape in the color tree of the input image is shifted by a random vector. This results in oscillating boundaries, as may be encountered in oil-paintings.

Shape Filtering. Small shapes are eliminated, which removes partly the textures of the original image, resulting in a more abstract rendering.

Style Transfer. For this rendering, the shapes of a chosen style image are computed and considered as a dictionary. Each shape **s** of the original image is then associated to its nearest shape in the dictionary. The notion of distance between shapes depends on several geometric or color properties as described in [4] in detail. Finally, the texture and (optionally) color of the dictionary shape is transferred to **s**.

For each rendering, a set of optional parameters permit to control the intensity of these modifications. For some renderings, such as shape abstraction, mixing two models in two different image regions is possible, based on the mask drawn by the user. For instance, shapes can be replaced by rectangles in the selected region, while in the rest of the image shapes are replaced by disks.

Finally, additional effects such as transparency (as an option) and blur (to avoid aliasing) are added. Observe that the set of possible parameters in the demo is simplified compared to the one available in the original code [4].

2.3 Reconstruction

In the last phase, an image is reconstructed from the tree of modified shapes. Observe that once modified, the shapes do not necessarily satisfy the same inclusion properties as before in the topographic map. As a consequence, it is necessary to choose a rendering order to reconstruct an image. The demo gives two possibilities for this order, based on the shape size (from largest to smallest), or on their depth in the tree.

Algorithm 1 describes the general structure of the code, implemented in the function "TreeOfShapes::render".

3 Experiments

3.1 Simple Renderings

Figure 2 shows the behavior of the method for image abstraction, using different models: using the original shapes, or transforming them into ellipses, rectangles or disks. In all the experiments shown here, the default parameters of the algorithm for image abstraction were used. Observe how a large number of different abstractions can be applied within the same framework.

Table 1 shows the default parameters of each task. Some experiments carried out use the default parameters, and others show the effect of modifying some of these parameters.

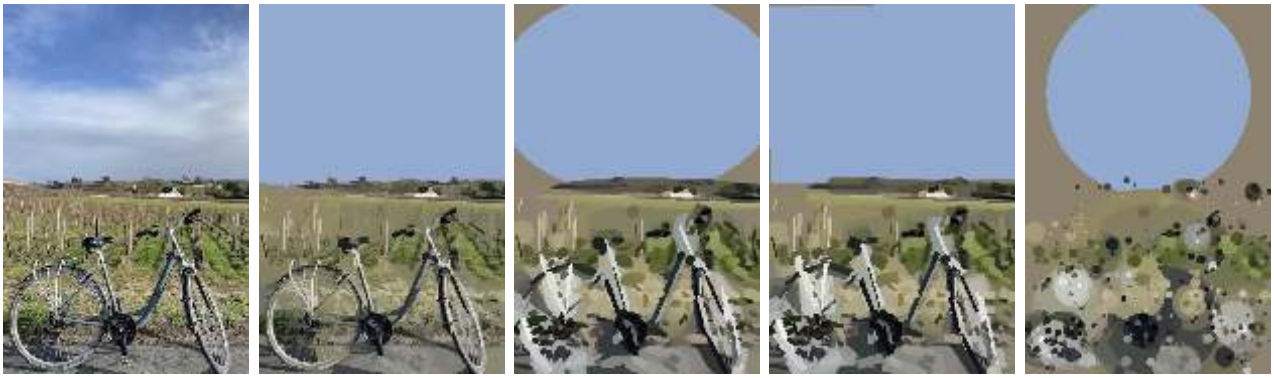

Figure 2: Abstraction task. From left to right: Original image, Abstraction with original shapes, ellipses, rectangles and circles.

Figure 3 shows an example of image abstraction using ellipses, with thresholds on minimal and maximal areas for the selection of the shapes. The minimal area parameter permits to remove small shapes and results in a more abstract representation. The maximal area parameter removes large shapes, such as the sky background.

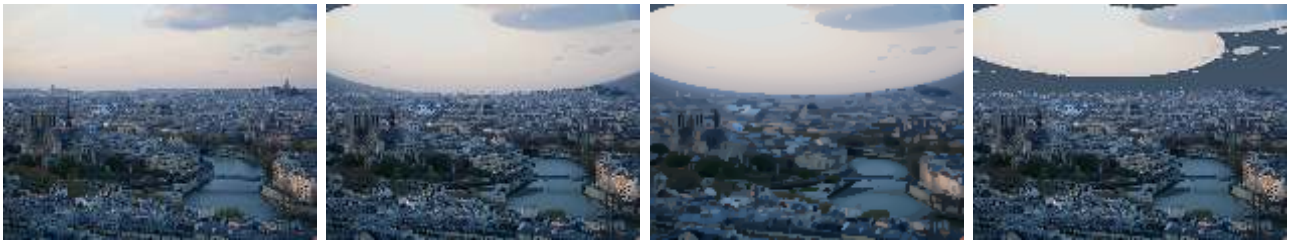

Figure 3: Abstraction with ellipses. From left to right: Original image, Abstraction with selection of shapes with (minimum area, maximum area) = (0%,100%); (0.01%, 100%); (0%,20%).

Figure 4 provides examples of similar interesting abstractions for a portrait image. In all the experiments shown here, the default parameters of the algorithm for image abstraction were used.

Algorithm 1: Pseudocode for the structured abstraction of images

Data: An initialized tree of shapes (ToS) with image, $ToS\_Parameters$, $Mask$ for model mixing, $Alternative\_Model$ if model mixing, ToS_Dict initialized with Style image and tree computed, $ToS\_Dict\_Parameters$.

Result: image abstraction

```
imgSynt ← newImage();
//DECOMPOSITION
T ← Compute_tree(ToS_Parameters);
//MODIFICATION: SHAPE SELECTION
//Filter shapes according to shape area and color
C ← Get_Filter_Condition(ToS_Parameters);
for shape in T do
    if shape satisfies C then
        shape.removed ← True;
//MODIFICATION AND RECONSTRUCTION
//Sort shapes according to rendering order
render_order ← Select_rendering_order(T, ToS_Parameters);
for shapes in T according to render_order do
    if shape is first shape then
        //shape is background
        if Style Transfer then
            //Color from Content or Style image
            color ← get_origin_of_color(ToS_Dict_Parameters);
        //Transform shape and update result image
        imgSynt ← synshape(shape, Rectangle, average(color), ToS_Parameters);
    else
        //select model to use
        if maskintersectsshape and alternative_model then
            model ← alternative_model;
        else
            model ← ToS_Parameters.model;
        if model ≠ Dict then
            //Transform shape and update result image
            imgSynt ← synshape(shape, model, ToS_Parameters);
        else
            //Select which shape from the Style image should be used to transform the shape
            Shape_Dict ← get_correspondence_shape(shape, ToS_Dict);
            //Transform shape and update result image
            imgSynt ← synShapeDict(shape, model, Shape_Dict, ToS_Dict_Parameters);
return(imgSynt)
```

Parameter	Shape abstraction	Watercolor	Shaking	Filtering	Style transfer
Min area	0.01	0	0	0.1	0
Max area	100	100	100	100	100
Scale ratio	3	3	3	3	3
Threshold color filtering	0.5	0.6	0.6	0.8	0
Epsilon	0	0	0	0	0
Render order	Large-small	Top-down	Top-down	Top-down	Large-small
Transparency	0	0	0	0	0.2
Keep meaningful boundaries	1	0	0	0	0
Dictionary model	-	-	-	-	2
Dictionary colors	-	-	-	-	From dictionary
Average color background	-	-	-	-	From dictionary
Shape x/y scaling	-	-	-	-	Different
Compactness	-	-	-	-	0

Table 1: Default parameters for each task.

Figure 4: From left to right: Original image, shape abstraction using original shapes, disks and rectangles respectively. For all the images, the default parameters of the algorithm for image abstraction were used.

Figure 5 shows different renderings on the same image, combining different shape models and shape modifications: Watercolor and Shaking are performed with the original shapes and ellipses, and shape filtering with the original shapes. In all the experiments shown in this figure, the default parameters of the algorithm for each task are used. Other model combinations are possible to create different abstractions. Other interesting renderings can also be achieved by modifying the advanced parameters (see Section 4).

Figure 5: From left to right: Water color using original shapes, Water color using ellipses, Shaking using original shapes, Shaking using ellipses, Shape filtering using original shapes.

Figures 6 and 7 present two examples of style transfer with the default parameters. It is generally easier to obtain satisfying results with images not containing large and flat regions, which are likely to be filled up with several spurious shapes from the dictionary. Better results can also be obtained by playing with the advanced parameters of the method, as shown in Figure 8. In this specific case, the largest shapes of the image have been ignored by selecting a threshold on the maximum area of the shapes.

Figure 6: From left to right: Style image, Content Image, Result Image of style transfer. This is a successful case of Style transfer with default parameters, probably because the image does not have large homogeneous areas.

3.2 Image Simplification

The framework allows removing chosen areas of the original image, prior to the application of the abstraction. First, the user draws a mask on the original image, then a segmentation of the image is carried out and the segmented areas that intersect with the drawn mask are transformed into an average color of the image. In the demo, the result of this procedure can be seen in the output section called "Segmented image".

Figure 9 shows the application of this process prior to applying the shaking task. The figure shows the original image, the image with the drawn mask, and the segmented image, which corresponds to

Figure 7: From left to right: Style image, Content Image, Result Image of style transfer. This example shows the limitations of the method when used with images with large homogeneous areas. The homogeneous sky is filled up with many different shapes of the style image.

Figure 8: Image abstraction with rectangles. From left to right: Style image, Content Image, Style transfer with default configuration and Style transfer using advanced configuration: maximal area of 2% and filter shapes based on contrast ($keepmeaningfulboundaries$ checked). This is a successful case of the application of the style transfer framework, even though the image has a large homogeneous area.

the elimination of the segments that intersect with the mask. The result of the shaking task on that image is shown on the right. A minimal area has been used in order to remove the small remaining background segments.

Figure 9: From left to right: Original image, Masked image for segmentation, Segmented image and Result Image of shaking with minimal area 0.08%.

Figure 10 shows another example using this segmentation step, this time with an abstraction rendering using disks.

3.3 Mixed Models

If the user selects a region but does not choose to apply image simplification, the selection allows to mix two models: all shapes intersecting the mask are modified using one model (called the alternative model), while the remaining shapes are modified using the main model. This alternative model can be used in all tasks, even in style transfer.

Figure 10: From left to right: Original image, Masked image for segmentation, Segmented image, and abstraction with circles.

In Figure 11, disks (in the selected area) and original shapes (in the remaining area) are mixed for an abstraction rendering.

Figure 11: From left to right: Original image, Masked image, Result abstract rendering, mixing two models for shape abstraction: original shapes and disks (for the masked zone).

Figure 12 shows an example of style transfer, using once again two models: the original shapes are used in the small masked area, while shapes from the dictionary are chosen for the rest of the image.

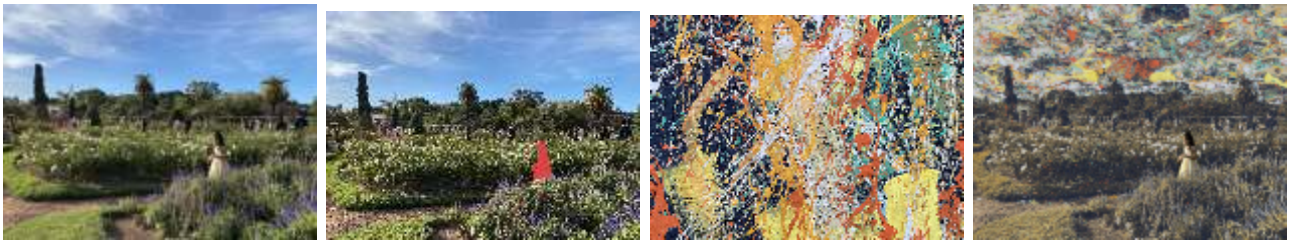

Figure 12: From left to right: Original image, Masked image, Style image, Image obtained by mixing two models for style transfer: dictionary shapes and original shapes (for the masked zone).

4 Demo

In order to use the demo provided on the IPOL website, it is required to upload a content image or to choose one of the available ones. A style image is also needed for Style transfer.

Two models can be mixed. Using the mask available in the demo, sections of an image can be selected. The main model to be used to transform the shapes will be the one chosen in the *Model* parameter. For shapes that intersect the drawn mask, the *Alternative Model* will be used.

Beware that the drawn mask can also be used to segment the input image, in order to simplify backgrounds for instance. If a mask is used to segment the image, it cannot be used to mix models.

We provide below a short description of all the main parameters available in the demo.

- Task: the available tasks are abstraction, watercolor, shaking, shape filtering or style transfer.

- Model: the type of shapes chosen to replace the ones of the input image. These shapes can be the original shapes, rectangles, ellipses, or disks.
- Use mask to simplify the input image: a flag to use segmentation in order to remove parts of the input image that intersect with the mask.
- Alternative model: geometric transformation used to replace the shapes of the content image that intersect with the drawn mask. It can be selected only if the mask is not already used to segment the image.
- Shape selection: keep only meaningful boundaries [1]. The option ε in the "advanced option" section permits to choose the level of meaningfulness of the selected shapes.

If necessary, some more advanced parameters can be modified in the demo. This set of advanced options is a simplified list of what is made possible by the full code provided with this paper, which implements all the options described in [4].

- Shape selection: Minimal area (in percentage of the input image size). Remove shapes with an area smaller than this value.
- Shape selection: Maximal area (in percentage of the input image size). Remove shapes with an area larger than this value.
- Shape selection: epsilon. Appears only when "Keep meaningful boundaries" is selected. Controls the level of meaningfulness of the selected level lines.
- Render order. Final rendering order of the shapes.
- Transparency: Alpha for transparency on the resulting image.
- Dictionary options: Model. Model used to select dictionary shapes.
- Dictionary options: Use Color. Source of color (input image or style image).
- Dictionary options: Use average color background. Source of color for the background.

Image Credits

 L.Bouza CC-BY 1.0.

 C.Launay CC-BY 1.0.

 J.Delon CC-BY 1.0.

 M.Muniz CC-BY 1.0.

 Vadim Onishenko CC-BY 1.0.

 M.Heguerte CC-BY 1.0.

 Starry Night - Vincent Van Gogh (Public Domain)

 Noura Faraj, inspired in Mondrian CC-BY 1.0.

References

[1] F. Cao, P. Musé, and F. Sur, *Extracting Meaningful Curves from Images*, Journal of Mathematical Imaging and Vision, 22 (2005), pp. 159–181, https://doi.org/10.1007/s10851-005-4888-0.

[2] V. Caselles, B. Coll, and J.-M. Morel, *Topographic Maps and Local Contrast Changes in Natural Images*, International Journal of Computer Vision, 33 (1999), pp. 5–27, https://doi.org/https://doi.org/10.1023/A:1008144113494.

[3] D. DeCarlo and A. Santella, *Stylization and Abstraction of Photographs*, ACM Transactions on Graphics, 21 (2002), p. 769–776, https://doi.org/10.1145/566654.566650.

[4] N. Faraj, G.-S. Xia, J. Delon, and Y. Gousseau, *A Generic Framework for the Structured Abstraction of Images*, in International Symposium on Non-Photorealistic Animation and Rendering (NPAR), 2017, https://doi.org/10.1145/3092919.3092930.

[5] P. F. Felzenszwalb and D. P. Huttenlocher, *Efficient Graph-Based Image Segmentation*, International Journal of Computer Vision, 59 (2004), pp. 167–181, https://doi.org/10.1023/B:VISI.0000022288.19776.77.

[6] P. Monasse and F. Guichard, *Fast Computation of a Contrast-Invariant Image Representation*, IEEE Transactions on Image Processing, 9 (2000), pp. 860–872, https://doi.org/10.1109/83.841532.

[7] A. Orzan, A. Bousseau, P. Barla, and J. Thollot, *Structure-Preserving Manipulation of Photographs*, in International Symposium on Non-Photorealistic Animation and Rendering (NPAR), 2007, https://doi.org/10.1145/1274871.1274888.