Image Abstraction through Overlapping Region Growth

Abstract
We propose a region-based abstraction of a photograph, where the image plane is covered by overlapping irregularly shaped regions that approximate the image content. We segment regions using a novel region growth algorithm intended to produce highly irregular regions that still respect image edges, different from conventional segmentation methods that encourage compact regions.

The final result has reduced detail, befitting abstraction, but still contains some small structures such as highlights; thin features and crooked boundaries are retained, while interior details are softened, yielding a painting-like abstraction effect.

Keywords: Non-photorealistic rendering, Image stylization, Segmentation, Abstraction

CCS Concepts
• Computing methodologies → Non-photorealistic rendering;

1. Introduction
Image stylization has seen a tension between abstraction, which removes detail from input images, and media emulation, which often adds detail such as watercolor texture or brushstroke direction. However, while the resulting images can contain fine-scale details such as paint texture, such details are often dissociated from any fine-scale detail that had been present in the input. The resulting images often show uniformity in shape and size of the abstraction primitives. This issue appears both in filter-based and region-based methods. Abstractions by filter-based methods may blur edges. Region-based abstractions often generate regular size regions which cannot capture small-scale structures and textures. We present an automatic oversegmentation algorithm that can create regions with highly irregular shapes and sizes; the resulting regions can then be used as stylization primitives. Sample results from our approach appear in Figure 1.

Our approach uses region-based abstraction to modify an input photograph, where the image plane is covered by overlapping irregularly shaped regions that approximate the image content. Unlike traditional segmentation methods, or superpixel methods such as SLIC [ASS10], we deliberately create non-compact regions, with the jaggedness and complexity of region shapes creating significant visual interest. Figure 2 shows some sample region shapes generated with our method. Image detail is preserved because the region boundaries conform to image edges, while variation in final region size allows very small details to be kept. We do not add any detail based on simulated media, though that could certainly be added if desired. Our regions have the following properties:
• Irregularity: generated regions have high irregularity in shape and size.
• Detail preserving: capture the small details of the image, like textures or thin features.
• Structure preserving: can conform to edges within the image, which enables capturing the structures.

Our process is entirely automatic. From the initial image, we compute a detail map by subtracting a smoothed version. Connected components of a thresholded detail map generate seeds for
a region growth process. Seeds are ordered based on the residual values, and seeds with smaller residual values are expanded first. When all regions have been found, we assign a color to the visible portion of that region.

This paper makes the following contributions:

- Overall, we propose a mechanism for abstracting images through highly irregular region shapes that nonetheless conform to image content. Our approach can be seen as a hybrid of stroke-based and region-based stylization methods.
- We introduce a region growing algorithm that generates intricate region shapes that conform to local detail. Unlike traditional oversegmentation methods, we emphasize the irregularity of region shape even in uniform areas.
- We perform a preliminary investigation of stylizations that involve manipulation of the regions’ colors or boundary shapes. While this approach is in its infancy, the early results appear promising.

The remainder of the paper is organized as follows. In the following section, we discuss related work. We then describe our method in Section 3 and present results from our abstraction process in Section 4, followed by discussions of the results in Section 5. We close in Section 6 and suggest directions for future work.

2. Previous Work

In the last two decades, there has been considerable progress in the science of generating stylized images. The immensity of the literature makes it difficult to discuss all related research in this section. However, we attempt to cover the work that is most related to ours in both results and approach.

Stroke-Based methods In stroke-based stylization, a common methodology is iteratively placing brush strokes with different color, size and orientation \[\text{[Hae90, Her98, CH02, ZZZZ09, LSF10, ZMGS17]}\]. Various artistic styles could be simulated based on this method. Haeberli \[\text{[Hae90]}\] placed brush strokes in a stochastic distribution in the neighborhood of a manually driven cursor. Hertzmann \[\text{[Her98]}\] used a random order to render the strokes to prevent an undesirable appearance of regularity in the brush strokes. Recently, Zheng et al. \[\text{[ZMGS17]}\] applied strokes in the context of 3D painting and animation where paint strokes change shape and length by movements of the camera. Their results benefit from modeling continuous and long brush strokes with variable size and shape.

Region-based methods Previous researchers have introduced several region-based rendering algorithms for non-photorealistic rendering. DeCarlo and Santella \[\text{[DS02]}\] performed segmentation \[\text{[Ahu96]}\] at different scales and used an overlap rule to infer the hierarchy.

Earlier segmentation methods have been applied on images to generate abstract images. Later, other researchers tried to mimic a style rather than a simplified segmentation of the image. Mould introduced a stained glass filter \[\text{[Mou03]}\] that mimics the medieval stained glass windows. The regions produced by an initial segmentation like meanshift \[\text{[CM02]}\] and then the regions boundaries manipulated by morphological operators.

Wen et al. \[\text{[WLL06]}\] applied an interactive segmentation. The user needs to manually specify which regions are background/foreground. Arty shapes \[\text{[SRHC08]}\] used simple shapes fitted to regions. More recent work by Faraj et al. \[\text{[FXDG17]}\] uses topographical map to generate stylized images from simplified shapes organized in a tree structure. They use a dictionary to extract shapes and simplify the abstraction by removing small shapes based on a threshold value.

Filter-Based methods Several filter-based methods in the literature produce abstract and artform images \[\text{[WOG06, PPC07, OBBT07, CSR10, KS10, KKD09, Mou12, SLKD15]}\]. Papari et al. \[\text{[PPC07]}\] introduce a painterly filter based on the Kawahara filter \[\text{[KHEK76]}\]. They employed different weighting functions to preserve corners and edges. However, their method fails to capture directional features and results in clustering artifacts. Later, Kyprianidis et al. \[\text{[KKD09]}\] present a generalization of the Kawahara filter that preserves the local structure and directional image features, providing better content preservation. Winnemöller et al. \[\text{[WOG06]}\] present an automatic, real-time video and image abstraction, by approximation to anisotropic diffusion. Orzan et al. \[\text{[OBBT07]}\] generated abstracted images by performing a Poisson reconstruction on extracted edges of the image. Their technique also produce different stylizations such as drawing and watercolor style from abstracted images.

Smoothed local histogram filters introduced by Kass and Solomon \[\text{[KS10]}\], computes derivatives and integrals of locally-weighted histograms over large neighborhoods. The histogram can tell about the neighbour population of a pixel and identify the number of modes, their values and the percentage of the population contained within each mode. Identifying the modes can be useful for stylization, segmentation and noise reduction purposes.

The geodesic filter \[\text{[CSR10]}\] and its variant, cumulative range geodesic filter \[\text{[Mou12]}\] smooths the image, while keeping edges sharp. Textured regions abstracted through CRGF, but the irregular shapes and ragged edges are kept. Since we are interested in capturing the irregularity of the textures, we will use CRGF filter to smooth our images.

Semmo et al. \[\text{[SLKD15]}\] proposed an oil painting filtering. They quantize the input image by using color palettes. To mimic the paint texture on canvas, they extract local orientations by means of smoothed structure tensor ciftensor and line integral convolution \[\text{[CL93]}\].

Reduced color palettes Black and white \[\text{[XK08, MG08, LM15]}\] images are simplified into only two colors. Some groups reduced
color palettes to more than just two colors. For example, Rosin and Lai [RL13], proposed an algorithm to add spot colour to grayscale or monochromatic images. Some factors such as layer region shape and salience considered for applying the spot colors. Different techniques on reduced color palette have been explained in a survey by Lai and Rosin [LR13].

Capturing textures of the image was one of our initial idea for region generation. Meanwhile, some learning based method [EM17] in NPR criteria, generates abstract images by transferring textures from one to another image. We are looking forward to investigate a hybrid of learning-base techniques on our regions.

3. Proposed method

Our method is a region growth approach guided by local colors, where regions are the key elements of the abstraction. Given an input image $I$, we want to generate regions $\{R_i\}$ that capture the important structures of the image. We identify the structural components that each corresponds to a region. Highly textured regions are presented by the irregular shape of the regions.

Our method has three main steps:

- **Seed placement**: we identify a starting point for each region; this is done by finding connected components from a thresholded detail layer. Each connected component above a minimum size generates one seed.
- **Growth mechanism**: a region grows until a stopping criterion is reached. We halt expansion locally when the incremental cost becomes too large; when expansion is not possible in any direction, the region is finished.
- **Rendering**: we place regions in order of size: on the bottom, the largest regions cover the background, and smaller ones come on top providing detail. The final color of each region is the median color of its visible pixels.

Figure 3 provides a visual depiction of these stages. In the following, we explain each step of the algorithm in more detail.

3.1. Seed Placement

In an effort to preserve image details, we used the image residuals to guide seed placement. We blur the input image and subtract the original, producing a detail layer. The detail layer is separated into negative and positive portions and thresholded, leaving us with three possible labels for every pixel: high positive value; high negative value; and neutral value.

We then find all connected components with positive and negative labels, discarding components that are too small (less than 10 pixels in size). We compute one region per connected component, starting the region growth at the component’s median pixel position.

**Growth Mechanism**

The seeds are now treated individually, with a region grown around each one, following a best-first order in an 8-connected graph around the seed. We continue adding pixels to the region, ordered by increasing cumulative range geodesic distance [Mou12], until a halting condition is met. Effectively, this process involves computing shortest paths where the edge weight to a node $p$ is the color distance to the seed, $d(p - s)$. In our process, we used the squared color distance in Lab space; we found that squaring the distance led to greater ability to distinguish fine detail.

Each seed is assigned a unique region ID, used to label each pixel in its region. A given pixel can belong to many regions simultaneously. In the end, we sort the regions in order of decreasing size and assign each pixel the ID of its smallest region.

**Region growth termination**

Region growth can be viewed as a single-source shortest path problem, where we compute range geodesic distances to a tree of nodes surrounding the seed. We expand the region by recursively adding the neighbours of each node, tracking the total path cost of each node from the seed. When the incremental cost along the growth path exceeds a per-region parameter $T$, we halt further growth along that path. When all growth paths have been halted, the region is considered complete. In our implementation, we do not explicitly terminate growth through a node; rather, the boundary nodes that we are considering adding are stored in a priority queue, and we implicitly halt progression by not adding potential successor nodes into the queue.

We decide whether or not to locally halt progression based on whether or not the cost of going in the current direction is considered to be large in the context of the current region. We operationalize “large” as follows. The first time a node added to a region is $h$ edges from the seed, we store its accumulated cost. This cost provides a baseline value, say $T$; we increase the baseline by a factor $k$, using $k = 100$ in our examples. Subsequently, for every pixel we consider adding to the region, we compare the total weight of its most recent $h$ edges, say $W_h$, with the baseline $T \times k$. When $W_h > T \times k$, we terminate progression. In other words, any increase beyond a factor of $k$ in the rolling-average path cost halts region growth in that direction. Algorithm 1 shows the steps.

3.2. Rendering

Once we have assigned preliminary region IDs to all pixels, we finalize the IDs by performing a connected component analysis of the ID map, a process we refer to as flattening. Doing so gives us different labels for regions that have been separated by later regions. For each of the final flattened regions, we compute the mode of its color histogram and apply this color to its constituent pixels. The final result is an over-segmented abstraction of the image. Figure 4 shows region contours resulting from rendering the connected components. Note that our approach is able to represent the textured areas of the image such as the grass and tree trunk by irregular-shaped regions. Often, the parts of the image that have little detail do not generate any seeds, and consequently, these areas may not belong to any region after the region growing process completes. Applying the connected component analysis will assure us that all pixels will be given a region ID.

However, uniform areas of the image may be covered by large regions with little interior detail. For many applications, this may be desirable. For cases where it is not, we suggest covering uniform
**Figure 3:** Region-based abstraction pipeline. We compute a detail layer from input image and its filtered one. Then, we generate initial seeds for each region from the connected components of this detail map. For each seed, we expand a region. When all regions have been found, we color the output by assigning to each region the average of the pixel values in the visible portion of that region.

**Algorithm 1 Region Growth**

**Input:** $G$: A graph on input image, $s$: seed, $k$: termination parameter

**Output:** $R$: Generated region

1: **procedure** GENERATE REGION($G, s$)
2: Initialize pixel costs to infinity
3: Set seed cost to 0
4: Push the current seed to an empty priority queue $Q_{reg}$
5: **while** $Q_{reg}$ not empty **do**
6: Pop the top pixel $u$ and
7: Set the total cost $C$ to $u$.cost
8: Push $u$ to $R$
9: **for each** neighbor $v$ of $u$ **do**
10: $\text{alt} := C + D(s, v)^2$
11: **if** $\text{alt} < v$.cost **then**
12: $v$.cost := alt
13: **if** $u$ passed $h$ hops for the first time **then**
14: $T := v$.cost
15: set $A$ to the $h$’th ancestor of $u$
16: $W := v$.cost - $A$.cost
17: **if** $|W| < T 	imes k$ **then**
18: $v$.parent := $u$
19: Push $v$ to $Q_{reg}$
20: **return** $R$

portions of the image by a template of textured regions from another image. In Figure 5, we show an example of a textured image and the corresponding regions, suitable for use as template regions.
4. Region Manipulation for Stylization

From our region-based approach, we can have different options to create variety of styles. Here, we suggest different art styles that could be created by manipulation of region colors and shapes.

Contour simplification

Drastic shape simplification has been used previously in non-photorealistic rendering [SRHC08, FXDG17], where often very simplified shapes used to fit each region. We simplified the region boundaries based on the Ramer-Douglas-Peucker Algorithm [DP73], which simplifies a piecewise linear curve by removing vertices based on a measure of error. This method would keep the shape of the structures very close to the original boundary; because we began with elaborate shapes, we retain a certain level of complexity even after simplification. A user can decide the level of simplification by adjusting the threshold $\varepsilon$ for computing the error. In Figure 6, $\varepsilon$ is set to 4 for both images. Higher thresholds create more drastic shape simplifications.

![Figure 6: Abstractions after boundary simplification by Ramer-Douglas-Peucker Algorithm ($\varepsilon = 4$), with initially generated regions (left) and with flattened regions (right).](image)

Boundary smoothing

We smoothed the region boundaries in frequency space, computing Fourier shape descriptors [BB93, KG82] and then truncating the coefficient sequence, thus producing a high level of abstraction. Some results appear in Figure 7, where we retained the first 7 coefficients.

![Figure 7: Boundary smoothing with Fourier shape descriptors.](image)

Reduced color palettes

We modified the regions’ colors, restricting them to small color palettes. The smallest possible color palette is a black and white palette; Figure 8 illustrates two results in black and white, where we threshold each region separately. Despite the simplicity of the approach, the oversegmentation lets us capture small details such as the fur in the cat image and thin structures in the port image.

![Figure 8: Black and white images: cat and port.](image)

Choosing a color palette and automatic assigning the colors to a region-based abstraction is not trivial. We recolored the region-based abstraction framework with a color palette extracted from an art Deco poster.

![Figure 9: Recoloring using a color palette from Art Deco Poster. Left to right: Art Deco poster and extracted color palette, recolored old man and parrot.](image)

5. Results and Discussion

We display more examples of our region-based abstraction in Figure 12. From left to right we illustrate original images, abstractions with flattened regions and overlapped regions with boundary simplification by Ramer-Douglas-Peucker Algorithm [DP73] with an epsilon value of 4.0. The original images for other examples presented in this paper can be seen in Figure 13.
We encourage readers to zoom in to better observe the abstraction effects. We wanted thin features and textures to stay visible after abstraction. In the market image, the texture of the bale of hay has been captured, and boundaries of the wheat sheaves kept their jagged shapes. Some small objects on the back shelves and hanging objects are abstracted. Our abstraction preserved strong edges such as the bars of the booth. On the right, the simplified version of the abstraction by means of overlapped regions created more artistic results.

The Chimney image shows how our abstraction mechanism can keep the isotropic structures and anisotropic textures at the same time. The abstraction process retained irregular textures in the background. The bricks have low contrast textures which are not prominent but are still captured by the regions. On the right, the abstraction with the simplified boundaries deforms the regular structure of the bricks, which might be undesirable.

Images of nature such as the Autumn image often contain irregular features like masses of leaves and tree trunk textures. Our abstraction mechanism can keep the complexity of these features. The abstraction with simplified region boundaries gives a more severe stylization that nonetheless conveys the content of the image. Unlike the chimney image, where directionality of features generated a slight undesirable effect, the less structured textures of Autumn remain comprehensible.

In our examples, the regions were colored using original image colors. Modification of colors can generate artistic images as well. Some abstraction methods like artistic thresholding [XK08] used meanShift [CM02] to segment the image, then assigned black and white colors to each region. With highly irregular shapes of our generated regions, we assigned the black and white colors with a simple thresholding mechanism. Since this naive method does not considered the connectivity of the regions, the effectiveness of the result is based strictly on the complex shapes of the regions we computed.

In terms of overlapping the regions, our abstraction method has some similarities with existing hierarchical approaches such as DeCarlo and Santella [DS02] or Faraj et al. [FXDG17] methods. Both approaches used the inclusion of regions, while in our algorithm design, inclusion was not a limitation. Our overlapped regions assisted in generating of the non-compact regions and conformed to the image structures. We allowed our regions to overlap each other, which they sometimes did to a great extent. This allowed us to extract small structures with elements of complex regions that represented thin or small features; such details often vanish through region-based abstractions.

5.1. Comparison with over-segmentation methods

Our priority-based overlapping region creation allows us to preserve small structures. We briefly compare our segmentation with those created by SLIC and QuickShift. SLIC was specifically intended to create compact, similar-sized segments; QuickShift has more variation in region size and its segments are not necessarily compact, but it still struggles to capture thin structures. Figure 10 illustrates oversegmented regions produced by our method, QuickShift and SLIC. Our method is capable of keeping the directional patterns. The regions on the flower petals have been oriented parallel to the gradient directions. In the SLIC and QuickShift methods, the vertical boundaries produced cuts on the petals.

5.2. Performance

We did not focus on performance and our current implementation is not optimized. Our algorithm implemented in C++ and uses OpenCV. On an Intel(R) Core(TM) i7-6700 with a 3.4 GHz CPU and 16.0 GB of RAM, it takes approximately 4 minutes to process a 1024 × 682 input image with about 4200 initial seeds, while the processing time for an image of size 480 × 320 with 780 seeds is about 40 seconds.

5.3. Limitations

We intended our algorithm to create highly complex regions, making use of local image detail to guide region shapes. However, it works best when there is significant local color variation. Blurred and smooth images are not good candidates for our algorithm since the lack of enough image details can lead to more uniform and circular shape regions. For example, Figure 11 shows an example with blur background in toque image and smooth surfaces on tomatoes.

6. Conclusions and Future work

In this paper, we presented an image stylization technique that creates intricate spatial primitives using a region growth mechanism, then covers the image plane with these primitives, customized to the image content. The results strike a balance between detail removal and preservation. Further abstraction is possible through simplifying the region boundaries and recoloring the regions.

There remain several directions for further development. The initial seed placement could be adjusted; probably it is possible to obtain equally complex results with fewer seeds, saving some computation time. We would like to experiment with different distance metrics for the initial region calculation, using not only color in the incremental distance but also other image properties, including larger-scale features such as histograms of gradients. Applying the method to video, computing regions in 3D rather than 2D, is a natural extension.

We have obtained appealing abstractions using the cumulative range geodesic, and presented some preliminary results of manipulating these boundaries and coloring the regions. We would like to explore more sophisticated methods to adjust boundaries and color the region interiors.

Acknowledgements

Many images were provided by photographers who made their work available through Flickr. Thanks to the following contributors: Theen Moy (cat), sicknotepix (girl), Greg Myers (tomatoes), David Taylor (port), Nicholas A. Tonelli (Autumn), Mark Lesley (owl), Carl Mueller (Machinery) Orzan (flower), author’s copyright (market). Other images (old man, starfish, cheminey, boat), came from The Berkeley Segmentation Dataset and Benchmark [MFTM01].
References


[DP73] Douglas D. H., Pinker T. K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. 5, 8


Figure 12: Image abstraction results. Left to right: the original images, abstractions with flattened regions and regions with boundary simplification by RDP algorithm [DP73]. Top to bottom, market, chimney and Autumn.


submitted to EUROGRAPHICS Workshop on ... (200x)
Figure 13: Original images of stump, port, flower, owl, parrot, cat, desert, tomatoes, toque and machinery.


