Fast Leak-Resistant Segmentation for Anime Line Art

Benjamin Allen
OLM Digital, Inc.
Japan
bn-allen@olm.co.jp

Akinobu Maejima
OLM Digital, Inc.
Japan
IMAGICA GROUP Inc.
Japan
akinobu.maejima@olm.co.jp

Ken Anjyo
OLM Digital, Inc.
Japan
IMAGICA GROUP Inc.
Japan
anjyo@acm.org

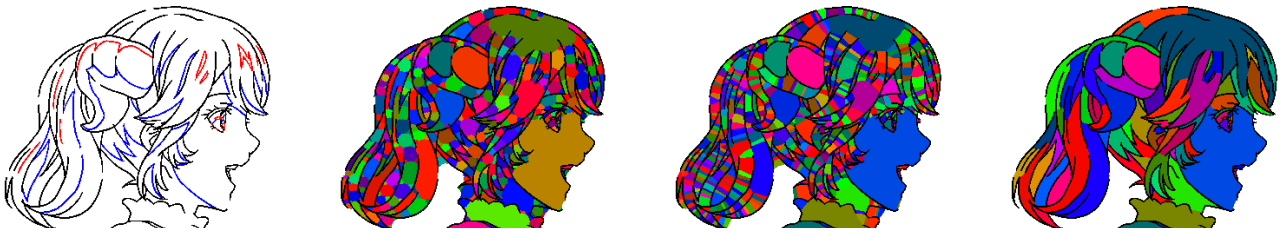

Figure 1: From left: input, existing trapped-ball method, ours, ours with region merging (8px ball radius, random colourization). © Junpei Inuzuka, IMAGICA INFOS/Restaurant to Another World 2 Project.

Abstract

We propose a fast leak-resistant automatic segmentation method for line art with gaps. Using the existing *trapped-ball* idea, we develop a fair, prioritized flood-fill that avoids artifacts, augment it with a heuristic region merging strategy and obtain robust results on anime line art minimizing over-segmentation, with consistent performance ~5–10 times faster than the baseline method.

CCS Concepts

• **Computing methodologies** → **Image manipulation**.

Keywords

line art, segmentation, trapped-ball, distance field

ACM Reference Format:
Benjamin Allen, Akinobu Maejima, and Ken Anjyo. 2024. Fast Leak-Resistant Segmentation for Anime Line Art. In *SIGGRAPH Asia 2024 Technical Communications (SA Technical Communications '24), December 03–06, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3681758.3698003

1 Introduction

Line art is a key component of Japanese anime. These drawings often have small unintentional gaps that break the watertightness of the lines, occuring in production as a result of cleanup or rasterization prior to colourization. These gaps yield incorrect segmentation and degradation of region-based methods for colourization [Hensman and Aizawa 2017], tracking [Zhu et al. 2016], labelling [Siyao et al. 2022], and vectorization [Favreau et al. 2016]. Leak-resistant segmentation is thus crucial to these applications, and the *trapped-ball* method [Zhang et al. 2009] is a popular solution.

We propose a fast, leak-resistant, automatic segmentation method for line art based on the existing trapped-ball method. We present a fair, prioritized flood-fill algorithm that resolves undesired segmentation when applying the existing method to monochrome line art [Hensman and Aizawa 2017], avoids ball-shape artifacts, and achieves consistently good performance at any ball radius. We also present a heuristic region merging algorithm to address the over-segmentation of narrow areas inherent to trapped-ball methods. We demonstrate that our method delivers robust results, ~5–10 times faster than the baseline method, in anime production scenarios.

2 Related Work

Zhang et al. [2009] proposed the *trapped-ball* leak-resistant segmentation method for edge masks derived from colour images. The method uses the idea of moving balls of successively smaller radii within the confines of the mask to produce distinct regions, then growing regions by colour similarity.

Hensman and Aizawa [2017] modified this method for monochrome line drawings by processing all ball radii then growing all regions to fill remaining pixels. Corners not reached by a larger ball may be filled by a smaller ball before region growing, producing a mottled segmentation and incentivizing use of a smaller radius, limiting leak-resistance. Larger regions may also be undesirably grown into areas too narrow to have been ball-filled.

Parakkat et al. [2021; 2022] propose various algorithms based on Delaunay triangulation, primarily for user-guided colourization and cleaning. They also explore automatic segmentation [2022] using skeleton gap completion to inform their Delaunay grouping.

Fourey et al. [2018] close gaps by local geometry estimation with a resulting automatic segmentation, however unintended junctions

SA Technical Communications '24, December 03–06, 2024, Tokyo, Japan

ACM ISBN 979-8-4007-1140-4/24/12
https://doi.org/10.1145/3681758.3698003

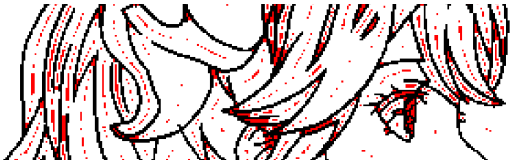

Figure 2: Distance field local maxima (red). © Junpei Inuzuka, IMAGICA INFOS/Restaurant to Another World 2 Project.

can result. Yin et al. [2022] identify viewer-perceived junctions of vector inputs with deep learning trained on human-annotated sketches, closing gaps and deriving an automatic segmentation. Scrivener et al. [2024] perform automatic segmentation of vector inputs based on a feature space of winding numbers.

Our method mitigates drawbacks of existing trapped-ball methods while delivering greater performance. Unlike methods using higher-level representations, we work with pixels, trivially preserving the exact input-output correspondence required in anime production. Our method helps to improve the quality and performance of region-based applications in anime and beyond.

3 Trapped-Ball Segmentation

Before describing our method in detail, we first review the existing trapped-ball segmentation method for line art as outlined by Hensman and Aizawa [2017] based on the approach of Zhang et al. [2009], which we use as a comparison baseline. The method uses repeated morphological operations on a binary mask of unfilled areas to simulate the trapped ball. The high level process is:

(1) Erode unfilled mask by ball radius
(2) Select a seed pixel from the eroded mask
(3) Flood fill from the seed to select one eroded component
(4) Dilate the selected area by the ball radius
(5) Assign these pixels a region ID and remove from the mask
(6) Repeat from (1) until no more valid seed pixels
(7) Decrement ball radius
(8) Repeat from (1) if ball is larger than a single pixel
(9) Expand filled regions to any reachable pixels
(10) Flood-fill remaining areas with new region IDs

The main bottleneck is the erosion in step 1 to select a seed, which processes the entire image every time. Seed selection cannot be trivially extracted from the loop in step 6; such seeds could be subsumed by dilation of others. The bounds from step 3 can be used to accelerate subsequent steps, notably step 4. The overall runtime increases with the ball radius due to the loop in step 8.

4 Fast Leak-Resistant Segmentation

4.1 Prioritized Flood Fill

Rather than improve the method in section 3, we propose a fast, alternative approach using the trapped-ball idea that processes all ball radii simultaneously and performs fair dilation of regions.

We compute a distance field on the binarized input using jump flooding [Rong and Tan 2006] (JFA+2). We define the edge distance $d(p)$ of a pixel p from a set E representing the line art as:

$$d(p) = \min_{p_e \in E} \|p - p_e\| \tag{1}$$

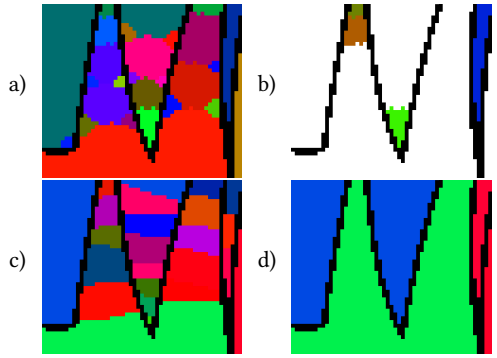

Figure 3: Baseline (a, radius 8) (b, radius 3, leaking) vs. ours (c, radius 8) with merging (d), from figure 8.

We consider that edge distance d will admit a ball of radius $\leq d$. To match the expected behaviour, we bias the initial radius r_b by +0.5 over the user-specified value. We work with d^2 to avoid square roots. Local maxima of the distance field $p_s \in S$ (figure 2) serve as seeds from which to fill, using 8 neighbours (if in bounds) N_8:

$$S = \{p_s \mid d(p_s) \geq d(p_n), \text{ for } \forall\, p_n \in N_8(p_s)\}\,. \tag{2}$$

We grow seeds to their erosion limit and then dilate them with a single queue-based prioritized N_8 flood-fill operation. Erosion is handled by filling from a seed while $d >= r_{b,s}$, where $r_{b,s} = \min(r_b, r_s)$ and $r_s = d(p_s)$. Dilation is handled by measuring the distance r_d between a pixel and its most recent queue ancestor where $r_{b,s} \leq d \leq r_s$ and whose parent was not dilating.

The fill queue is initialized with all seeds. Each seed is lazily assigned a region ID g when popped ($g = 0$ before then), or discarded if that pixel is already claimed. To each queue element we associate g and the priority 4-tuple $q = \langle q_1, q_2, q_3, q_4 \rangle$. The queue is processed greatest priority first, compared lexicographically, and dilation is continued until all remaining pixels are filled. As an optimization, we pre-fill seeds where $r_s > r_b$ beforehand, largest r_s first. The pre-fill uses a simple breadth-first flood-fill up to the erosion limit.

$$q_1 = \begin{cases} \max\left(0, r_{b,s}^2 - r_d^2 + 1\right) & \text{if } r_d > 0 \\ r_{b,s}^2 & \text{if } r_s \geq 2 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$q_2 = \begin{cases} -g & \text{if } r_d = 0 \wedge g > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$q_3 = \begin{cases} r_{b,s}^2 & \text{if } 0 < r_d < r_{b,s} \\ d^2 & \text{otherwise} \end{cases} \tag{5}$$

$$q_4 = \begin{cases} 1/2 & \text{if } p \in S \\ -\frac{\Delta d^2}{\|\Delta p\|} & \text{otherwise} \end{cases} \tag{6}$$

Priority q_1 processes biggest seeds (by $r_{b,s}$) first before dilation, and then largest remaining dilation first. The delay of 'tiny' seeds ($r_s < 2$) and the +1 bias serve to resolve contour-chasing edge cases. Priority q_2 grows the oldest planted seed to its erosion limit before other seeds are planted. Priority q_3 causes most operations to be ordered by d, where the $r_{b,s}$ term serves as a tiebreak at equal q_1.

Figure 4: Fair dilation.Baseline (left) with ball shapes vs. ours (right) with straight boundaries (ball radius 16).

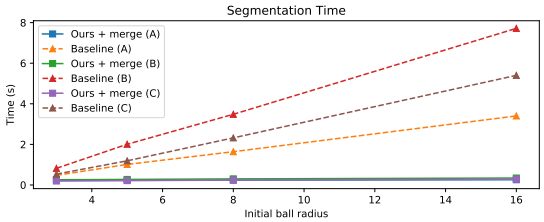

Figure 5: Performance comparison; inputs in figure 8.

Priority q_4 prefers filling downslope. This prevents tiny seeds from growing along edges because extended dilation of normal seeds will run downslope before tiny seeds are planted.

Our prioritization, especially q_1, results in 'fair' dilation of all regions from all ball radii. Unlike the baseline, this allows us to avoid ball shapes and produce straight boundaries (figure 4).

4.2 Region Merging

Our initial segmentation, like the baseline, produces many small regions in areas tighter than the initial ball. To address this, we propose a fast, heuristic region merging method.

For regions i and j, neighbouring pixels $\{p_i, p_j\} \in P_{ij}$ form a 'neck' of radius r_n, the largest d at the boundary. If $P_{ij} \neq \emptyset$, a merge candidate $m = \{i, j\} \in M$ is recorded. We evaluate the undirected viability v_+ of each m, and the viable merge arity l_s of each region, using two passes ($l = 0$ initially, m removed from M if $\neg v_+$). Let $r_{m,i} = \max_{k \in i} r_{s,k}$, and $l_{m,i} = \max_{k \in i} l_{s,k}$.

$$l_{s,i} = |\{k \mid \exists\ \{i,k\} \in M \wedge v_{i+k}\}| \tag{7}$$

$$r_n = \max_{\{p_i,p_j\} \in P_{ij}} \min\left(d(p_i), d(p_j)\right) \cdot \left[\left\|p_i - p_j\right\| = 1\right] \tag{8}$$

$$f_{i \to j} = \left\langle r_{m,i}, l_{m,i} \right\rangle \leq \left\langle r_{m,j}, l_{m,j} \right\rangle \tag{9}$$

$$w\,(x, \gamma) = \left(x^2 + \epsilon\right) \leq \gamma \left(r_n^2 + \epsilon\right) \tag{10}$$

$$v_{i \to j} = \begin{cases} w(r_{m,i}, \gamma_1) & \text{if } l_{m,i} \leq 2 \\ w(r_{m,i}, \gamma_2) \wedge w(r_{b,s,j}, \gamma_2) & \text{otherwise} \end{cases} \tag{11}$$

$$v_{i+j} = v_{j+i} = \left(v_{i \to j} \wedge f_{i \to j}\right) \vee \left(v_{j \to i} \wedge f_{j \to i}\right) \tag{12}$$

The parameters ϵ and γ take usual values $\epsilon = 2$, $\gamma_1 = 2$ and $\gamma_2 = 2.5$. Parameter ϵ makes the behaviour more stable at small values of d, while γ is a threshold for the effective 'difficulty' of pushing a region through the available neck.

The elements of M are converted to a priority queue, each assigned the priority 4-tuple u. M is updated to track remaining candidates. Regions are merged lowest arity first to allow high arities to reduce leaks, then largest first, ensuring the largest relevant r_s is compared to any neck. Candidates $m = \{i, j\}$ (such that $v_{i \to j}$) are popped and merged. When $l_{m,i} > 2$, if $\exists\ k \mid \{i,k\} \in M \wedge v_{i \to k} \wedge \neg v_{k \to i}$, this ambiguous merge is cancelled and no future merges $\{i,k\}$ where $f_{i \to k}$ are attempted. When m refers to an already-merged region, it is updated and, if still v_+, re-queued.

$$u = \left\langle -\max\left(l_{m,i}, 2\right), r_{m,i}^2, -r_{b,s,j}^2, r_n \right\rangle \tag{13}$$

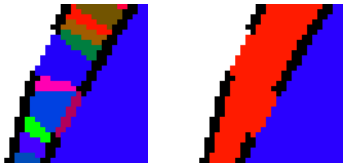

Figure 6: Leak resistant region merging.

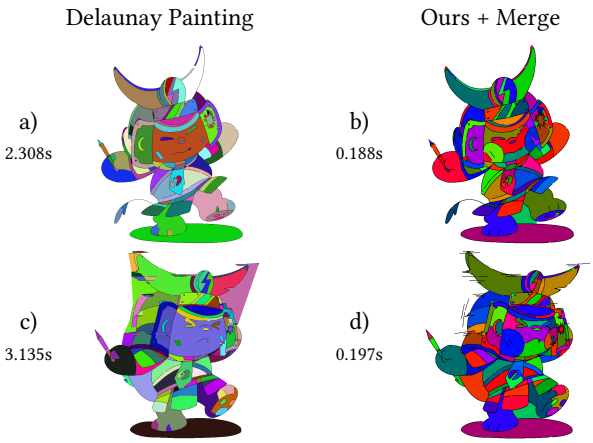

Figure 7: Comparison with automatic random-colour *Delaunay Painting* [Parakkat et al. 2022] (ours with radius 8). Actual resolution 1920×1080; BG trimmed for display. © OLM Asia SDN BHD.

5 Evaluation

We evaluate our proposed method comparing the segmentation and performance to the baseline. We implement both methods in C++. We use OpenCV for the morphological operations in the baseline, with the bounds optimization in section 3. We parallelize our jump flood with OpenMP (noting that a GPU implementation would trivially eliminate the cost of this step), and use a few SSE instructions for the comparison of 4-tuples q in section 4.1. Our results are obtained using an AMD Ryzen 5950x with 64GB RAM.

We compare results across inputs from production sources, showing the segmentation at a representative radius of 8px in figures 1 and 8, and the performance in figure 5 at various radii. We add artificial gaps $\leq$ 10px to leak-free base images, and compare to ground-truth segmentation to make leakage maps (figure 8e) for our method with merging. The pixel redness indicates the area ratio of other truth regions within that region of our result.

As noted in section 3, the runtime of the baseline scales quickly with increasing ball radius. In contrast, our runtime increases much more slowly (and asymptotically, as the pre-fill becomes less effective) with a gain of ~5–10 times at practical radii of 5–8px.

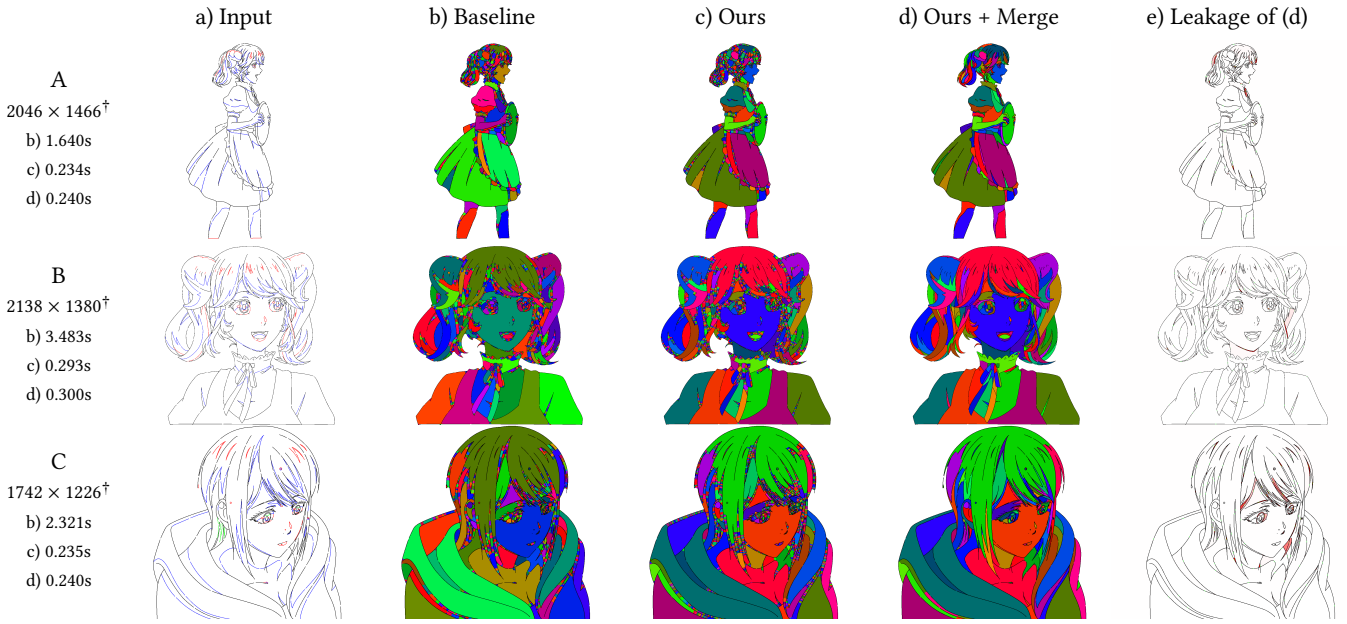

Figure 8: Segmentation at ball radius 8, random colours. Leakage (e) shows artificial gaps in green, leakage in red. Some 'leaks' from N_4 ground-truth segmentation. †Actual resolution; BG trimmed for display. Please zoom in for details.

The baseline (figure 3) shows tight corners filled by progressively smaller balls. Lowering the radius (figure 3b) partially mitigates this at the risk of leaks. Ours produces straight boundaries instead of circles, while merging (figure 3d) eliminates the corner problem and the oversegmentation of narrow areas. See also figures 1 and 8.

Figure 7 shows a comparison with automatic *Delaunay Painting* [Parakkat et al. 2022], after thinning the input. They produce broadly good results in ~10 times our runtime (results gathered separately on an Apple M2 Max). Their skeleton closure produced some undesirable boundaries (figure 7c), and we also observe that it may cause leakage with incorrect closure (figure 7a, top-right of helmet). Some unmerged very small (few pixel) regions also remain.

Our results demonstrate good resistance to leaks (figure 8e), with clean boundaries at gaps (figure 6). Our merging is robust in practice, particularly to gaps that are small compared to neighbouring regions, with relatively few leakages. Indeed, leaks primarily occur where r_n is near r_s on both sides of a gap. Some thin regions also end up leaking; we have not yet been able to satisfactorily prevent this without over-segmentation. A current limitation of the distance field is that it cannot distinguish between gaps of 1 and 2px (etc), reducing precision in narrow areas.

6 Conclusion

We have presented a fast, leak-resistant, automatic segmentation method for line art using a fairer version of the popular trapped-ball idea, augmented with heuristic region merging. These eliminate ball-shape artifacts and most over-segmentation, delivering robust results for anime production with consistent, fast performance.

In future we will improve the merge heuristics, to better handle thin regions and further reject ambiguity, for our production needs.

Acknowledgments

We would like to thank Junpei Inuzuka and IMAGICA INFOS for their permission to use images from *Restaurant to Another World 2*, and also Dr. Parakkat for running *Delaunay Painting* on our data.

References

Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. 2016. Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.

Sébastien Fourey, David Tschumperlé, and David Revoy. 2018. A fast and efficient semi-guided algorithm for flat coloring line-arts. In *International Symposium on Vision, Modeling and Visualization.*

Paulina Hensman and Kiyoharu Aizawa. 2017. cGAN-based Manga Colorization Using a Single Training Image. In *Proc. ICDAR*, Vol. 3. 72–77.

Amal Dev Parakkat, Marie-Paule R Cani, and Karan Singh. 2021. Color by numbers: Interactive structuring and vectorization of sketch imagery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–11.

Amal Dev Parakkat, Pooran Memari, and Marie-Paule Cani. 2022. Delaunay Painting: Perceptual Image Colouring from Raster Contours with Gaps. *Computer Graphics Forum* 41, 6 (2022), 166–181. https://doi.org/10.1111/cgf.14517 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14517

Guodong Rong and Tiow-Seng Tan. 2006. Jump flooding in GPU with applications to Voronoi diagram and distance transform. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games.* 109–116.

Daniel Scrivener, Ellis Coldren, and Edward Chien. 2024. Winding Number Features for Vector Sketch Colorization. In *COMPUTER GRAPHICS forum*, Vol. 43.

Li Siyao, Yuhang Li, Bo Li, Chao Dong, Ziwei Liu, and Chen Change Loy. 2022. Animerun: 2d animation visual correspondence from open source 3d movies. *Advances in Neural Information Processing Systems* 35 (2022), 18996–19007.

Jerry Yin, Chenxi Liu, Rebecca Lin, Nicholas Vining, Helge Rhodin, and Alla Sheffer. 2022. Detecting viewer-perceived intended vector sketch connectivity. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–11.

S. Zhang, T. Chen, Y. Zhang, S. Hu, and R. R. Martin. 2009. Vectorizing Cartoon Animations. *IEEE Transactions on Visualization and Computer Graphics* 15, 4 (2009), 618–629.

Haichao Zhu, Xueting Liu, Tien-Tsin Wong, and Pheng-Ann Heng. 2016. Globally Optimal Toon Tracking. *ACM Transactions on Graphics (SIGGRAPH 2016 issue)* 35, 4 (July 2016), 75:1–75:10.