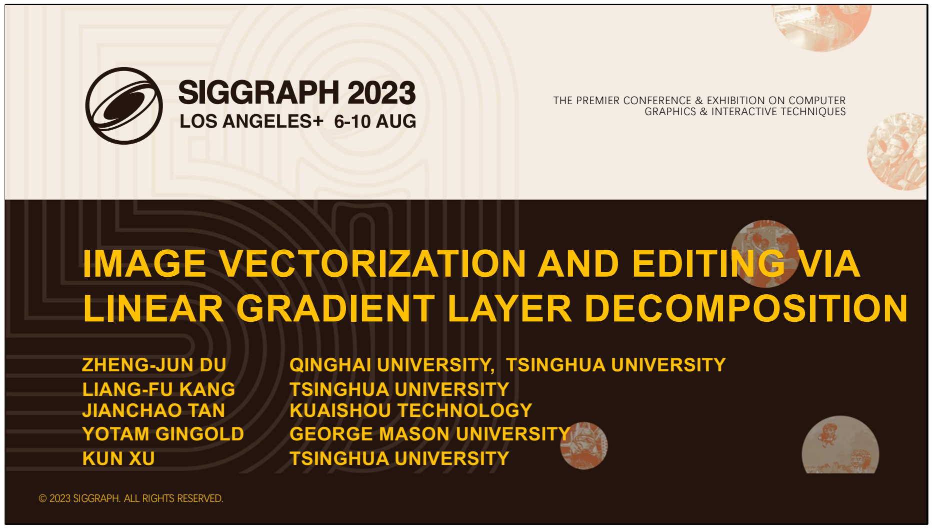


SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

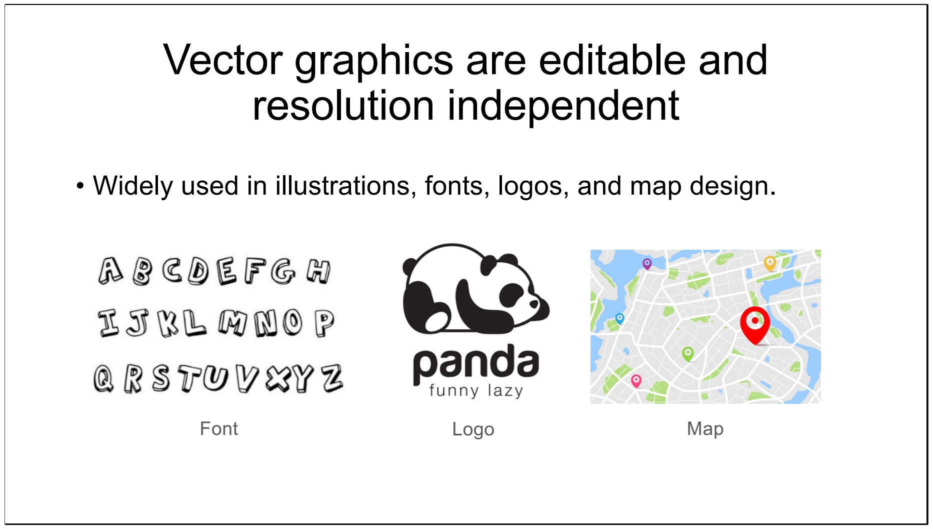
THE PREMIER CONFERENCE & EXHIBITION ON COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES

IMAGE VECTORIZATION AND EDITING VIA LINEAR GRADIENT LAYER DECOMPOSITION

ZHENG-JUN DU	QINGHAI UNIVERSITY, TSINGHUA UNIVERSITY
LIANG-FU KANG	TSINGHUA UNIVERSITY
JIANCHAO TAN	KUAISHOU TECHNOLOGY
YOTAM GINGOLD	GEORGE MASON UNIVERSITY
KUN XU	TSINGHUA UNIVERSITY

© 2023 SIGGRAPH. ALL RIGHTS RESERVED.

Hello everyone, I am Yotam Gingold from George Mason University, it's my honor to present our work 'Image vectorization and editing via linear gradient layer decomposition'.

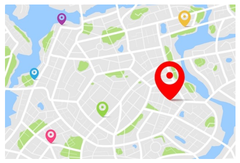


Vector graphics are editable and resolution independent

- Widely used in illustrations, fonts, logos, and map design.

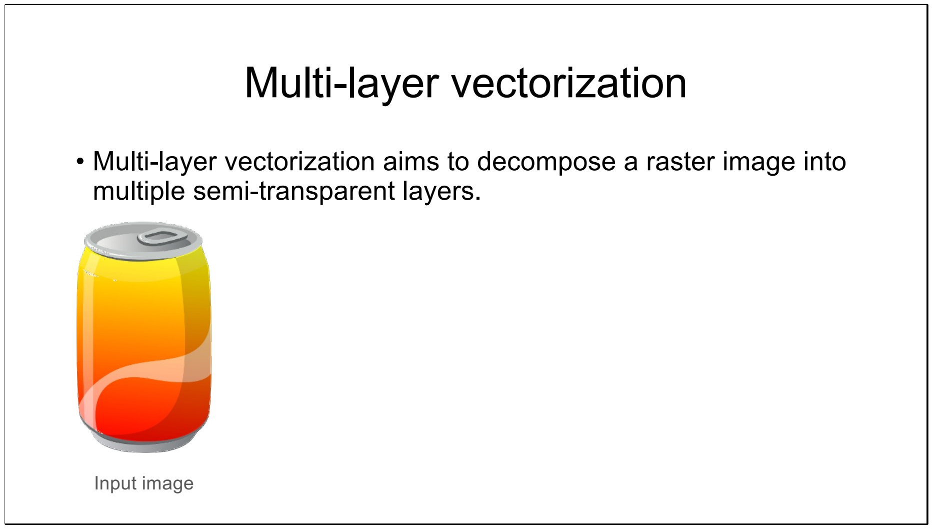
Font

Logo



Map

A key advantage of vector graphics over raster graphics is their editability and resolution independence. They are widely used in font, logo, map design, etc.



Multi-layer vectorization

- Multi-layer vectorization aims to decompose a raster image into multiple semi-transparent layers.

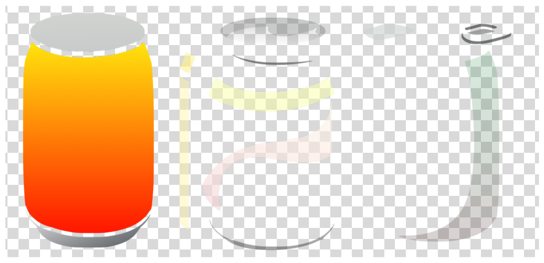
Input image

In this work, we focus on a specific type of vectorization i.e., multi-layer vectorization. For a given raster image, multi-layer vectorization decompose it into a set of ordered semi-transparent layers.

Multi-layer vectorization

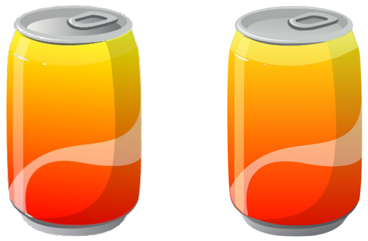
- Multi-layer vectorization aims to decompose a raster image into multiple semi-transparent layers.

Input image

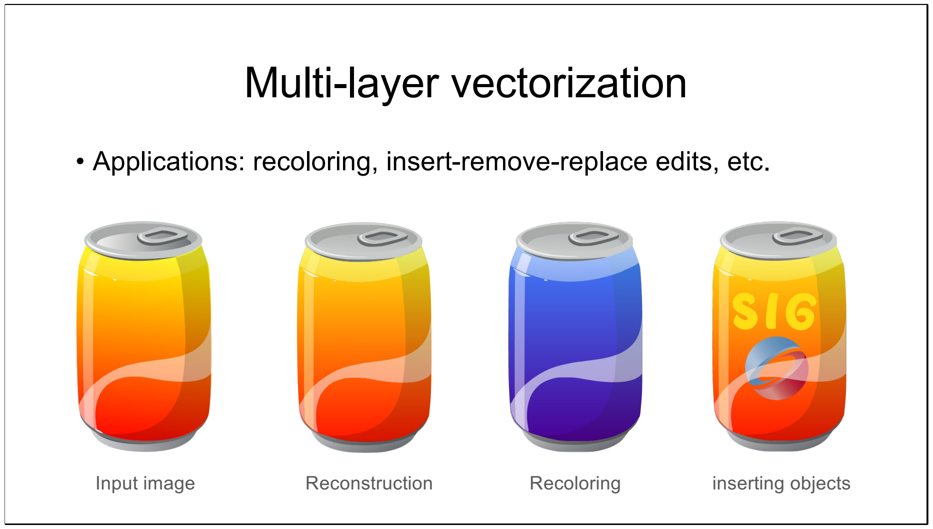

Decomposed semi-transparent layers

Multi-layer vectorization

- The decomposed layers reconstruct the input raster image by alpha blending.

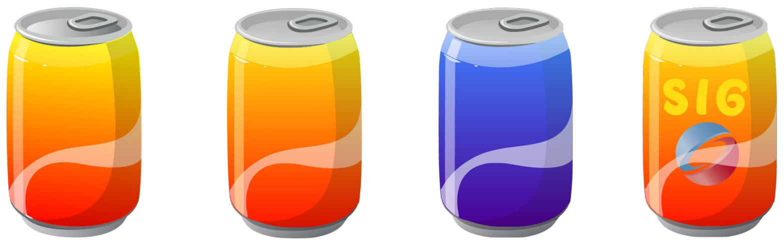

Input image Reconstruction

Meanwhile, these decomposed layers can accurately reconstruct the input by alpha blending.



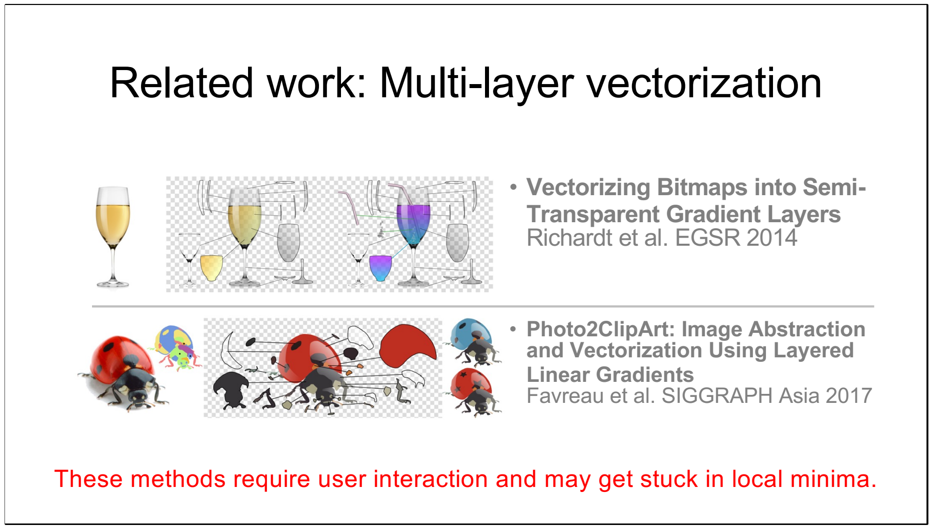
Multi-layer vectorization

- Applications: recoloring, insert-remove-replace edits, etc.



Input image Reconstruction Recoloring inserting objects

After decomposition, we can perform recoloring or object insert-remove-replace edits using these linear gradient layers in illustrator software.



Related work: Multi-layer vectorization

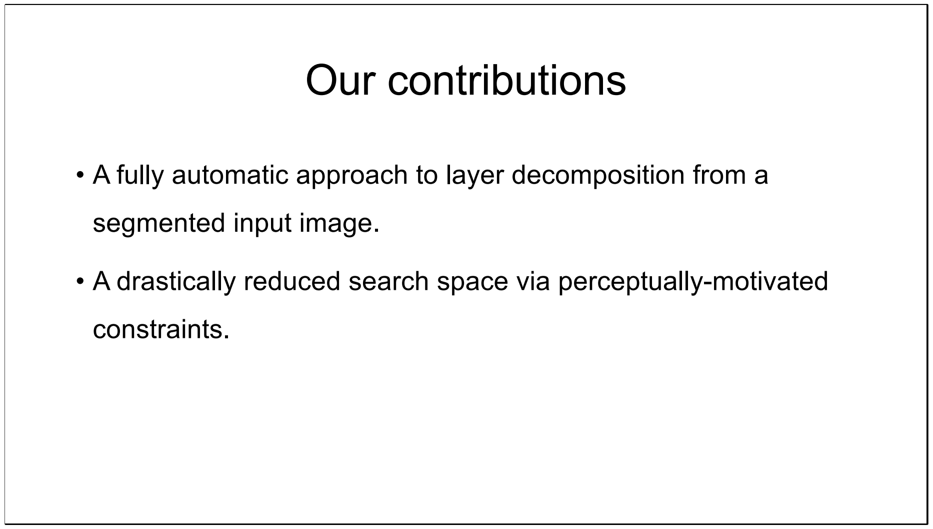
- **Vectorizing Bitmaps into Semi-Transparent Gradient Layers**
  Richardt et al. EGSR 2014

- **Photo2ClipArt: Image Abstraction and Vectorization Using Layered Linear Gradients**
  Favreau et al. SIGGRAPH Asia 2017

These methods require user interaction and may get stuck in local minima.

The most similar works to ours are [Richardt et al. 2014] and [Favreau et al. 2017]. Richardt et al's work proposed to have users select image regions one-at-a-time to decompose into linear or radial gradient layers. Photo2ClipArt proposed a Monte Carlo Tree Search algorithm for multi-layer decomposition.

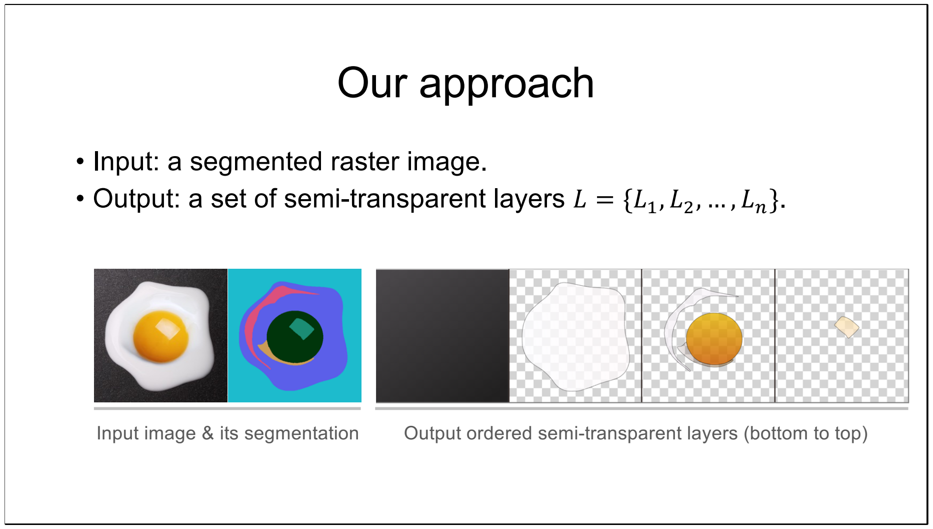
Both methods require more or less user interaction. Photo2ClipArt may get stuck in local minima.



Our contributions

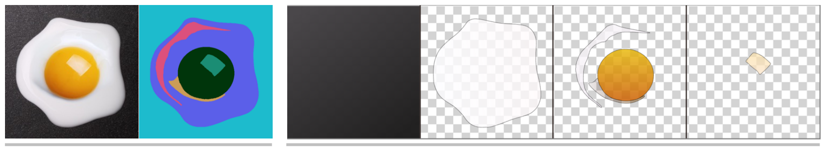
- A fully automatic approach to layer decomposition from a segmented input image.
- A drastically reduced search space via perceptually-motivated constraints.

To address this problem, we propose a fully automatic approach to gradient layer decomposition that finds globally optimal layers given a segmented input image. We drastically reduce the search space with perceptually-motivated constraints, resulting in a small space that can be exhaustively enumerated quickly.



Our approach

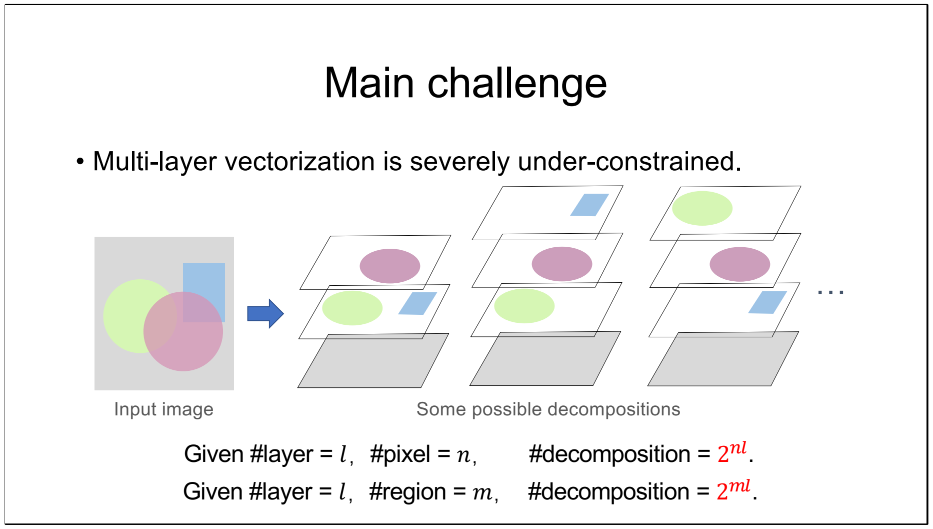
- Input: a segmented raster image.
- Output: a set of semi-transparent layers $L = \{L_1, L_2, \ldots, L_n\}$.



Input image & its segmentation

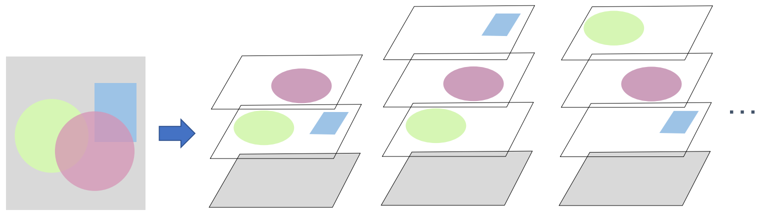
Output ordered semi-transparent layers (bottom to top)

Our method takes a raster image and its segmentation mask as input, outputs a set of ordered semi-transparent layers.



Main challenge

- Multi-layer vectorization is severely under-constrained.



Input image

Some possible decompositions

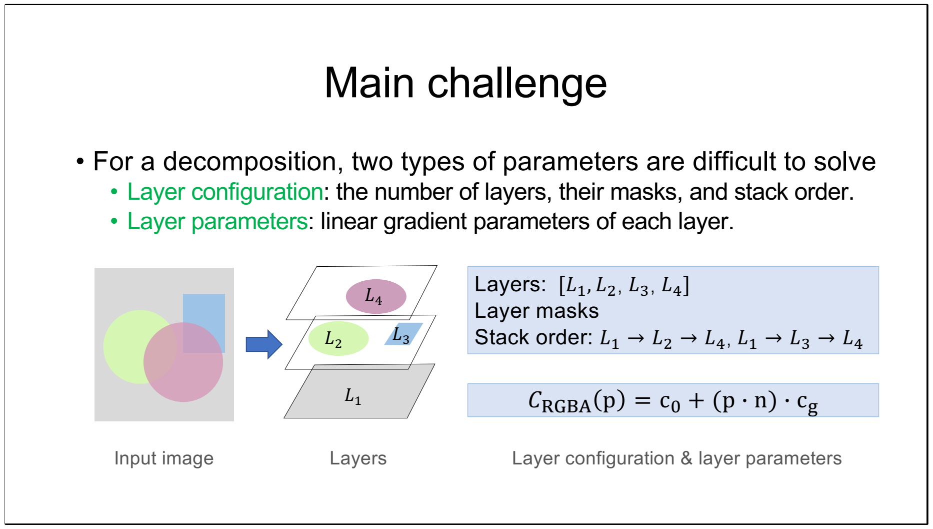
Given #layer = l, #pixel = n, #decomposition = 2^{nl}.

Given #layer = l, #region = m, #decomposition = 2^{ml}.

Multi-layer vectorization is a severely under-constrained problem, for a given raster image, there are infinite number of possibilities.

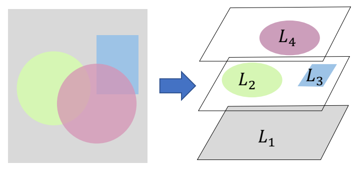
Assuming there are l layers and all n pixels in the input image are independent, so there are 2^{nl} possible decompositions.

If we segment the input into m regions and force pixels in a region to share the same layers, then there are still 2^{ml} possible decompostions.



Main challenge

- For a decomposition, two types of parameters are difficult to solve
  - Layer configuration: the number of layers, their masks, and stack order.
  - Layer parameters: linear gradient parameters of each layer.



Input image

Layers

Layers: $[L_1, L_2, L_3, L_4]$
Layer masks
Stack order: $L_1 \to L_2 \to L_4,\ L_1 \to L_3 \to L_4$

$$C_{\text{RGBA}}(\mathrm{p}) = \mathrm{c}_0 + (\mathrm{p} \cdot \mathrm{n}) \cdot \mathrm{c}_\mathrm{g}$$

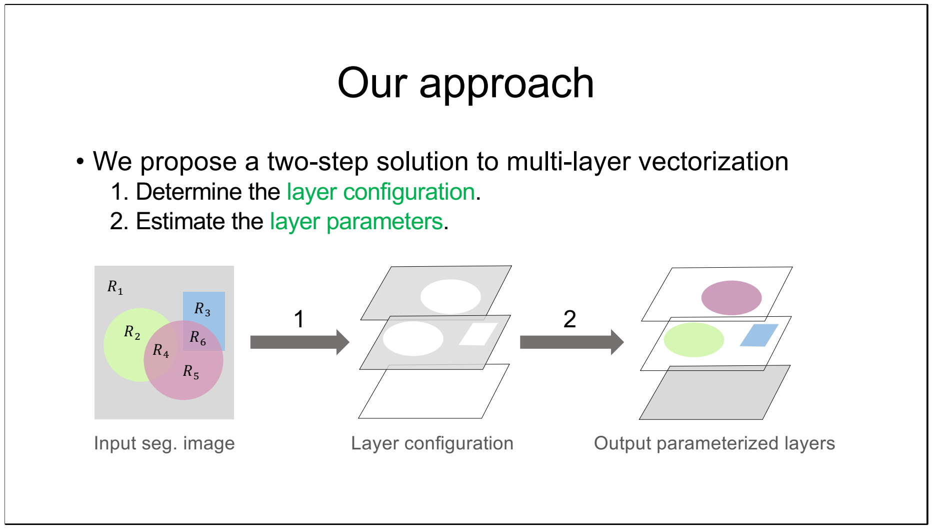
Layer configuration & layer parameters

For a specific decomposition, two types of parameters need to be determined.

One type is the discrete parameters: the number of layers, their masks, and stack order, which we refer to as the layer configuration.

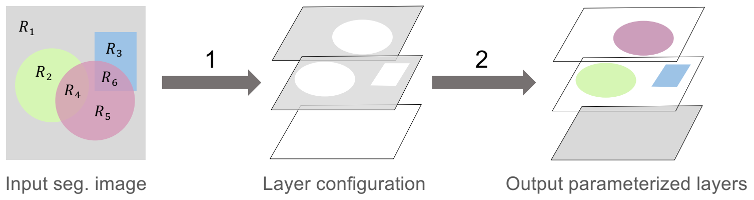
The other type is the continuous parameters: the linear gradient parameters for each layer, which we refer to as the layer parameters.

It is difficult to optimize both types of parameters simultaneously.



Our approach

- We propose a two-step solution to multi-layer vectorization
  1. Determine the layer configuration.
  2. Estimate the layer parameters.

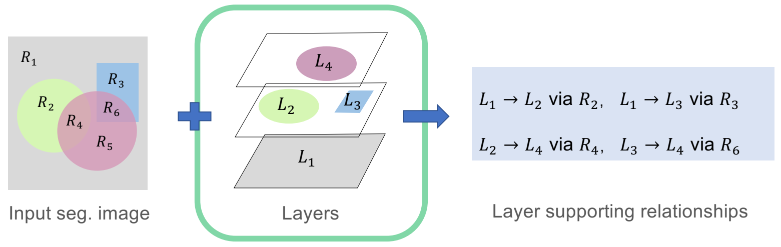


In general, our proposed method contains two steps. Firstly, we determine the layer configuration from the input image and its segmentation. Secondly, we optimize the layer parameters from the layer configuration.

Next, before introducing our method, we first give two basic concepts used throughout our paper.

Basic concepts

- Layer support: A lower layer L_i supports ($\rightarrow$) a higher layer L_j if
  1) L_i and L_j overlap.
  2) No regions in their overlap are covered by a layer in-between.



$L_1 \rightarrow L_2$ via R_2, $L_1 \rightarrow L_3$ via R_3

$L_2 \rightarrow L_4$ via R_4, $L_3 \rightarrow L_4$ via R_6

The first concept is layer support.

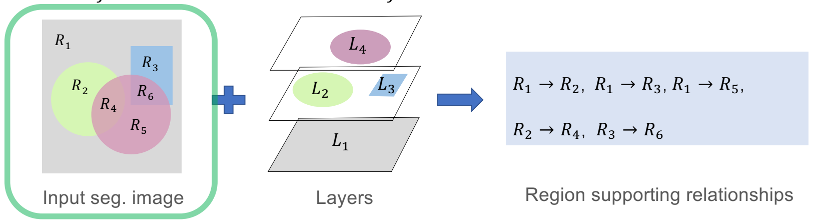
We say that a lower layer L_i supports a higher layer L_j if they satisfy two conditions:
1) the two layers overlap;
2) one or more overlapping regions are not covered by any in-between layer

The overlapping regions satisfying condition 2) are referred to as the supporting region(s)

In this example, $L_1 \rightarrow L_2$ via supporting region R_2 , $L_1 \rightarrow L_3$ via supporting region R_3.

Basic concepts

- Region support: Region R_i supports ($\rightarrow$) region R_j if
  1) R_i and R_j are adjacent.
  2) R_i's top layer L_i supports R_j's top layer L_j.
  3) R_j is in the overlap of L_i and L_j.



$R_1 \rightarrow R_2,\ R_1 \rightarrow R_3,\ R_1 \rightarrow R_5,$

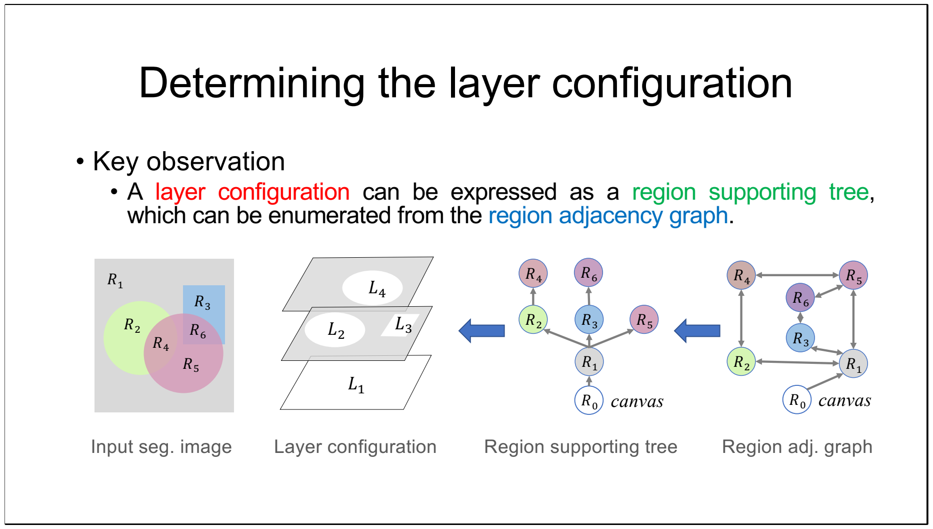
$R_2 \rightarrow R_4,\ R_3 \rightarrow R_6$

The second concept is the region supporting.

We say that region R_i supports region R_j if
1) they are adjacent regions.
2) R_i's top (covered) layer L_i supports R_j's top (covered) layer L_j.
3) R_j is one of the supporting regions from L_i to L_j.

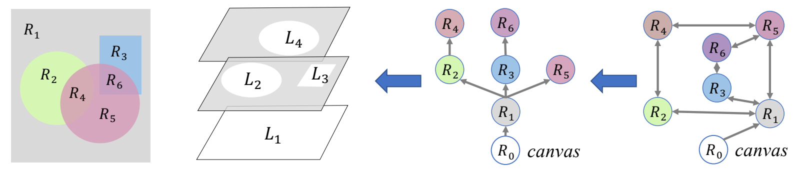
In this example, $R_1 \rightarrow R_2$, $R_2 \rightarrow R_4$

Is 3 the same as "R_j is in the overlap of L_i and L_j"?



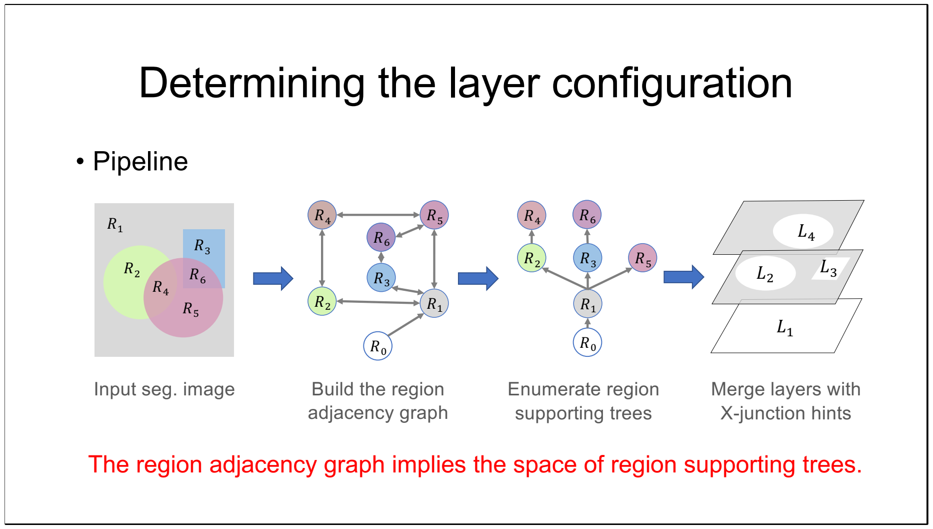
Determining the layer configuration

- Key observation
  - A layer configuration can be expressed as a region supporting tree, which can be enumerated from the region adjacency graph.



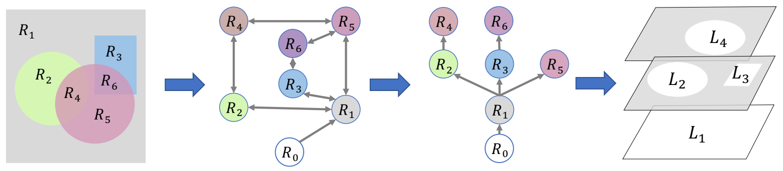
Input seg. image Layer configuration Region supporting tree Region adj. graph

With the concepts, we find that the region supporting relationships in a layer configuration can be expressed as a tree, we call it region supporting tree, and which can be enumerated from a region adjacency graph.
The region adjacency graph has edges for every possible valid supporting relationship.



Determining the layer configuration

- Pipeline



Input seg. image

Build the region adjacency graph

Enumerate region supporting trees

Merge layers with X-junction hints

The region adjacency graph implies the space of region supporting trees.

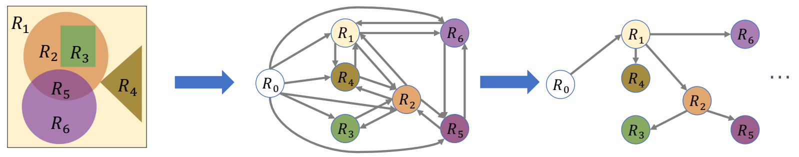
So we propose a graph-based method to determine the layer configuration, specifically our pipeline contains three steps:

1) Building the region adjacency graph from the input, which essentially defines the space of all region supporting trees
2) Enumerating the spanning tree from the region adjacency graph as the region supporting tree
3) Merging layers with X-junction hints and derive the layer configuration

Next, we first introduce the region adjacency graph construction.

Building the region adjacency graph

- Directly enumerating over all region supporting trees from the initial region adjacency graph is rather expensive.

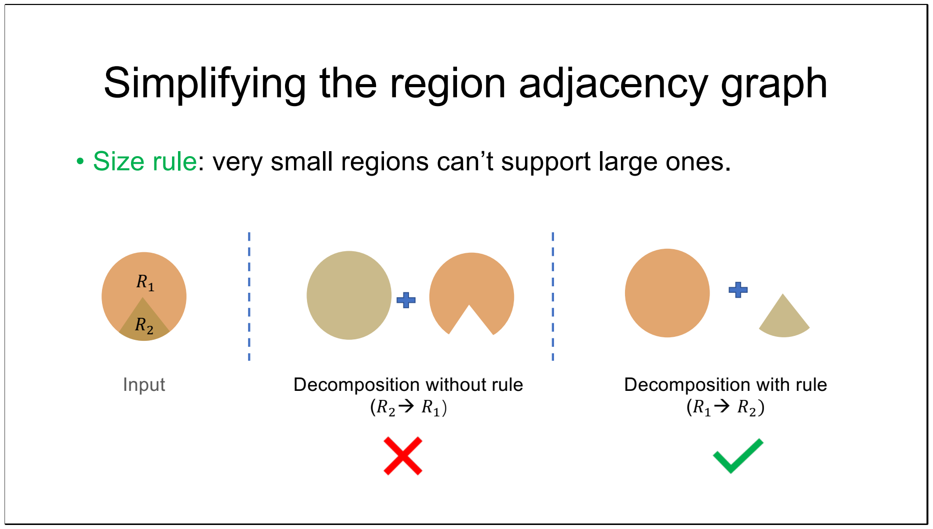


Input seg. image   Initial region adj. graph   Many region supporting trees

How to reduce the search space of region supporting trees?

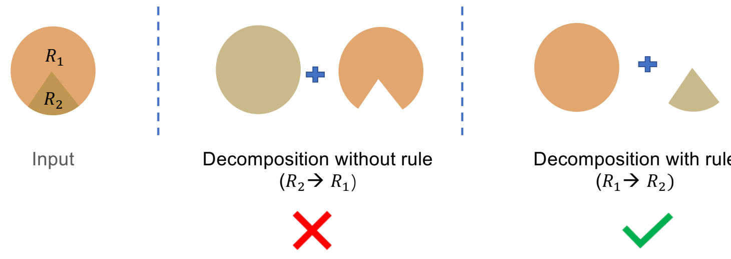
In a region adjacency graph, a node corresponds to a region in the input segmentation, a directed edge corresponds to a region supporting relationship. We add a virtual node R_0 as the default canvas. We add directed edges from it to all other regions.

Directly enumerating over all region supporting trees from the initial region adjacency graph is rather expensive. The region adjacency graph implies the space of region supporting trees. Motivated by figure ground and part perception of shapes , we propose three rules to remove very unlikely supporting edges from the graph to reduce the search space.



Simplifying the region adjacency graph

- Size rule: very small regions can't support large ones.



Input

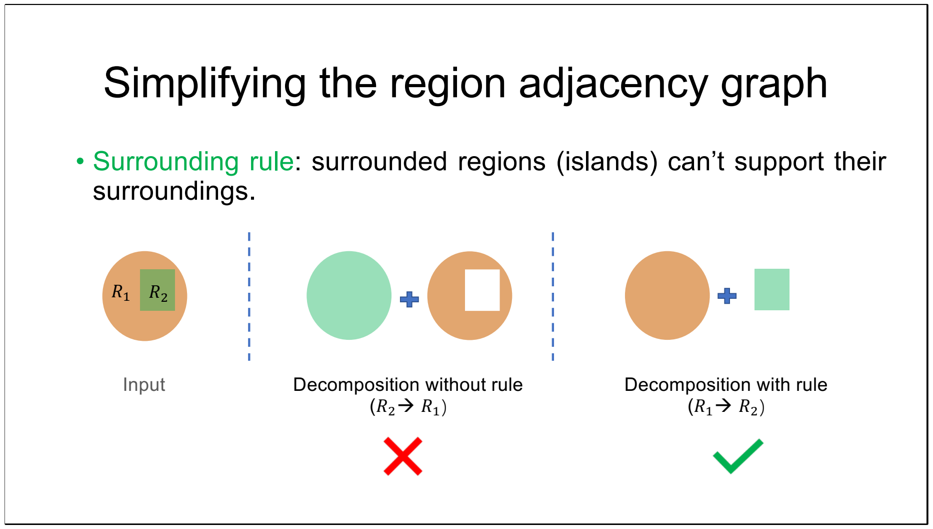
Decomposition without rule
$(R_2 \rightarrow R_1)$

Decomposition with rule
$(R_1 \rightarrow R_2)$

The first one is the size rule. For two adjacent regions, if one's size is much larger another one, then the small-sized region are not allowed to support another one.

As shown in the figure, the decomposition results violating and obeying this rule are shown in the middle and right, respectively.

In contrast, the decomposition result on the right is more consistent with the figure ground perception.



Simplifying the region adjacency graph

- Surrounding rule: surrounded regions (islands) can't support their surroundings.

Input

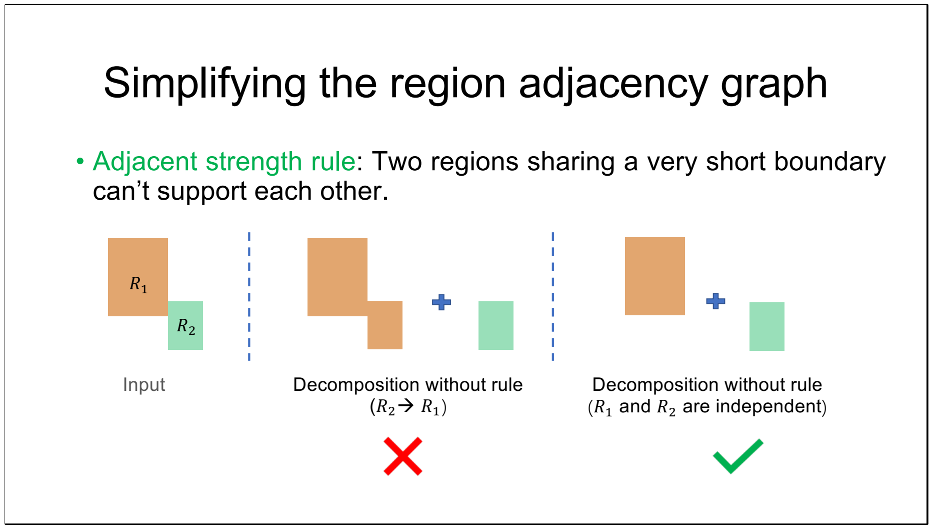
Decomposition without rule
($R_2 \rightarrow R_1$)

Decomposition with rule
($R_1 \rightarrow R_2$)

The second one is the surrounding rule. If one region is surrounded by another one, then the surrounded region is not allowed to support the surrounding one.

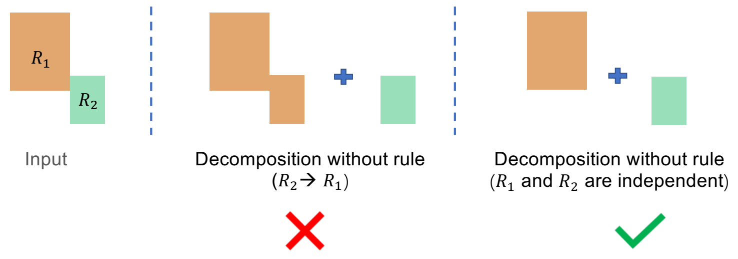
The decomposition results violating and obeying this rule are shown in the middle and right, respectively.

We can see that the right one is more conform to the figure ground perception, while the middle one is less intuitive because some layer contain hole.



Simplifying the region adjacency graph

- Adjacent strength rule: Two regions sharing a very short boundary can't support each other.

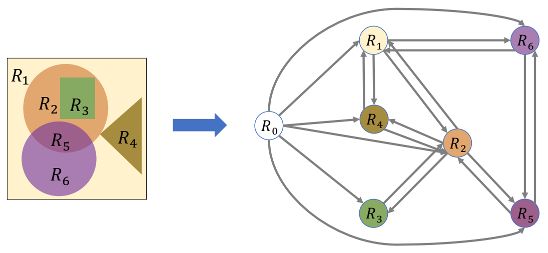

R_1
R_2
Input
Decomposition without rule
($R_2 \rightarrow R_1$)
Decomposition without rule
(R_1 and R_2 are independent)

The last one is the adjacent strength rule. If two regions share very short boundary, then they are not allowed to support each other.

As shown in the figure, the decomposition results violating and obeying this rule are shown in the middle and right, respectively.

Compared to the middle one, the right one that decompose the input into two isolated rectangular layers is more consistent with the part perception of shapes.

An example

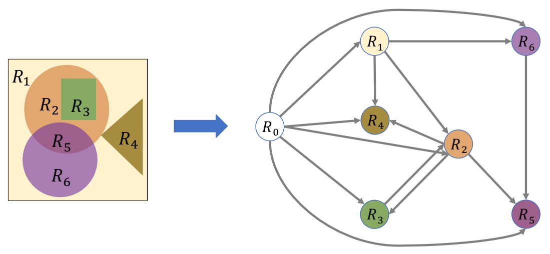


Input seg. image   Initial region adjacency graph

Here we provide an example to show the these three rules in region adjacency graph simplification.

For this example, the initial region adjacency graph contains 6 nodes and 20 directed edges. Next, we simplify it with these aforementioned 3 rules.

An example

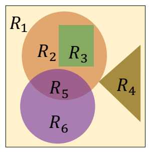


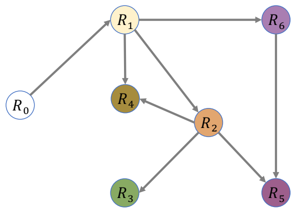
Input seg. image

Simplified with the size rule

Firstly, we remove edges that violate the size rule.

An example



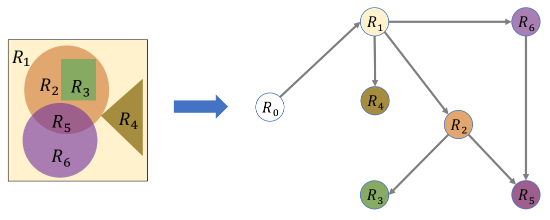


Input seg. image

Simplified with the size rule
surrounding rule

secondly, we remove edges that violate the surrounding rule.

An example



Input seg. image

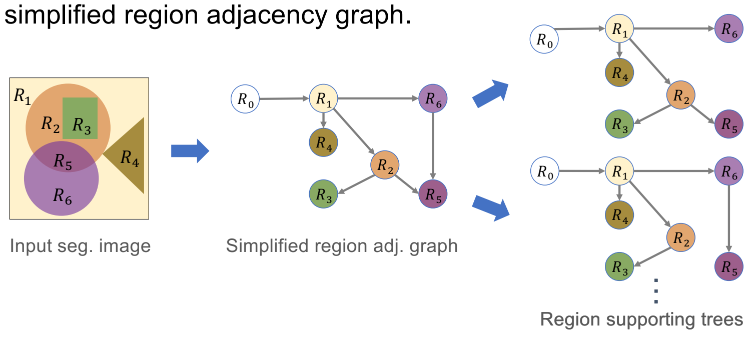
Simplified with the size rule
surrounding rule
adjacent strength rule

At last, we remove edges that violate the adjacent strength rule.

As a result, there are 7 edges in the simplified region adjacency graph, which greatly reduce the search space.

Enumerating region support trees

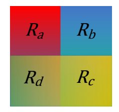
- Enumerating the spanning trees (region support trees) from the simplified region adjacency graph.


R_1 R_2 R_3 R_4 R_5 R_6
Input seg. image
R_0 R_1 R_2 R_3 R_4 R_5 R_6
Simplified region adj. graph
R_0 R_1 R_2 R_3 R_4 R_5 R_6
Region supporting trees

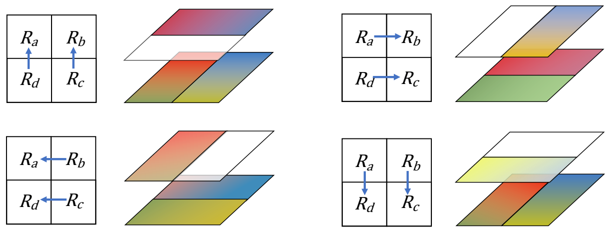
After simplifying the region adjacency graph, next we enumerate all spanning trees from the region adjacency graph and regard them as the region supporting trees.

Merging layers with X-junction hints

- An X-junction occurs when a semi-transparent layer runs across 2 other layers [Meteli 1974, 1985].



An X-junction



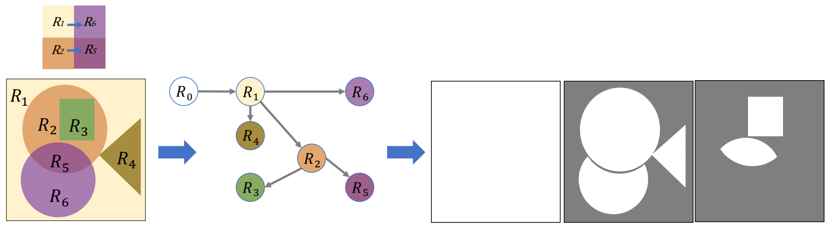
Regions' supporting relationships and layer configurations

For each region supporting tree, we first convert it into a layer representation and then try to merge some of them according to the X-junction assumption.

As shown in the figure, an X-junction denotes 4 regions with a 2 × 2 grid-like layout, an x-junction occurs when a semi-transparent layer runs across 2 other layers.

According to the X-junction assumption, there are only 4 allowed configurations of supporting relationships as shown on the right.

Merging layers with X-junction hints

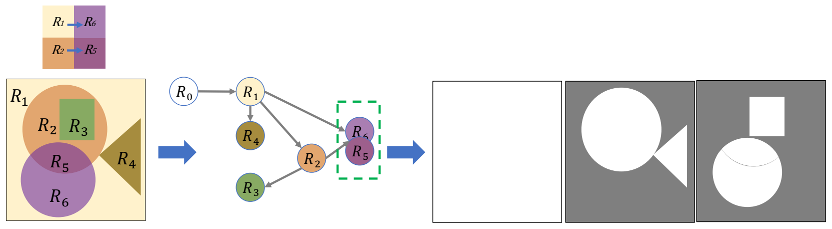


Input seg. image & an X-junction A region supporting tree Layer configuration before layer merging

Here for the region supporting tree, the derived layer configuration as shown in the right.

As we can see, the purple circular layer covering R_5 and R_6 splits into two layers. Next we merge the top layers of R_5 and R_6 according to the x-junction assumption.

Merging layers with X-junction hints



Input seg. image & an X-junction

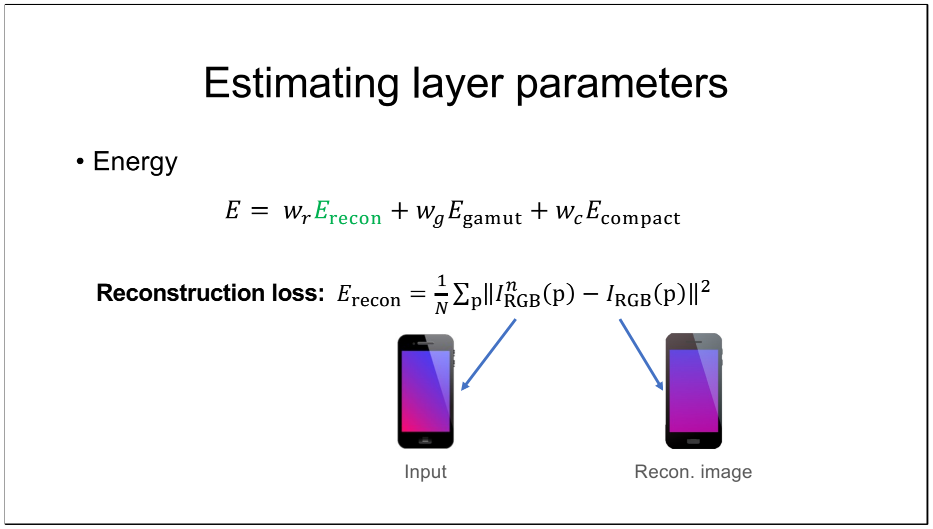
Merging the top layers of R_5 & R_6

Layer configuration after layer merging

After layer merging, we obtain a complete circular purple layer on the top.

As a result, we get 5 layers, the layer configuration is shown on the right.

Once we have obtained a layer configuration, the next step is to estimate the layer parameters.



Estimating layer parameters

- Energy

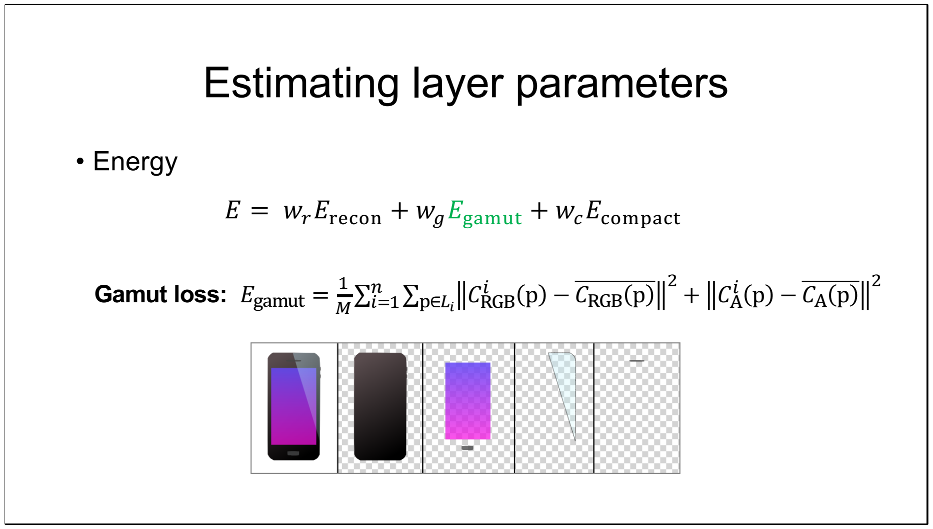
$$E = w_r E_{\mathrm{recon}} + w_g E_{\mathrm{gamut}} + w_c E_{\mathrm{compact}}$$

Reconstruction loss: $E_{\mathrm{recon}} = \frac{1}{N} \sum_{\mathrm{p}} \| I^{n}_{\mathrm{RGB}}(\mathrm{p}) - I_{\mathrm{RGB}}(\mathrm{p}) \|^2$

Input
Recon. image

We define an energy function to assess the layer quality. Our energy function is defined as the weighted sum of a reconstruction term, a gamut term, and a compactness term.

The reconstruction term prefers the reconstructed image to be as close to the input image as possible.

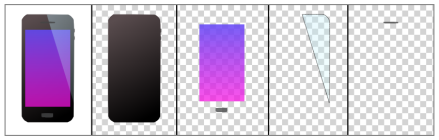


Estimating layer parameters

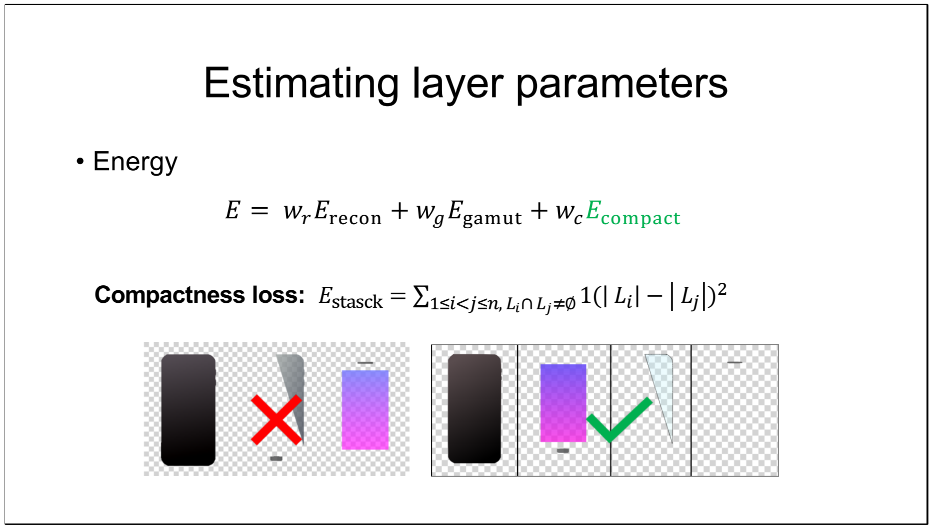
- Energy

$$E = w_r E_{\text{recon}} + w_g E_{\text{gamut}} + w_c E_{\text{compact}}$$

Gamut loss: $E_{\text{gamut}} = \frac{1}{M} \sum_{i=1}^{n} \sum_{\mathrm{p} \in L_i} \left\| C_{\text{RGB}}^{i}(\mathrm{p}) - \overline{C_{\text{RGB}}(\mathrm{p})} \right\|^2 + \left\| C_{\text{A}}^{i}(\mathrm{p}) - \overline{C_{\text{A}}(\mathrm{p})} \right\|^2$



We define a gamut term to penalize layers with color outside the RGB cube or large opacity . Its purpose is to generate layers with valid color and opacity values, and encourage more semi-transparent layers to achieve good editability

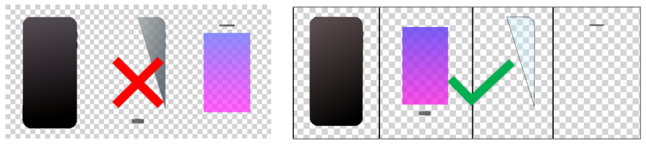


Estimating layer parameters

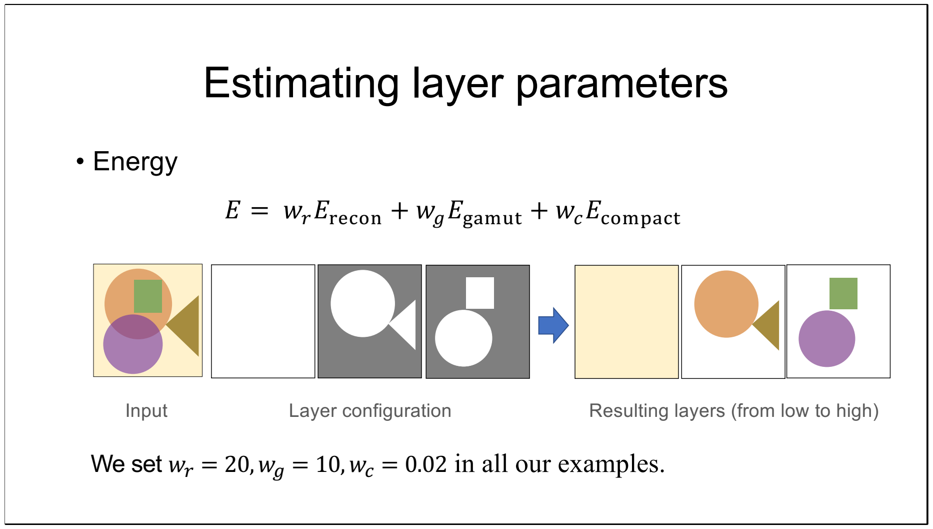
- Energy

$$E = w_r E_{\text{recon}} + w_g E_{\text{gamut}} + w_c E_{\text{compact}}$$

Compactness loss: $E_{\text{stasck}} = \sum_{1 \le i < j \le n,\, L_i \cap L_j \neq \emptyset} 1(|L_i| - |L_j|)^2$



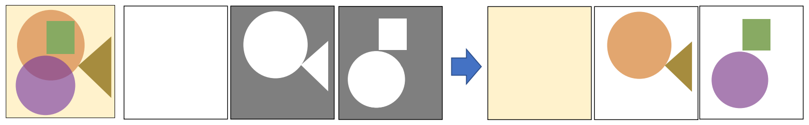
We define a compactness term to encourages smaller-sized layers to be on top of larger-size layers.



Estimating layer parameters

- Energy

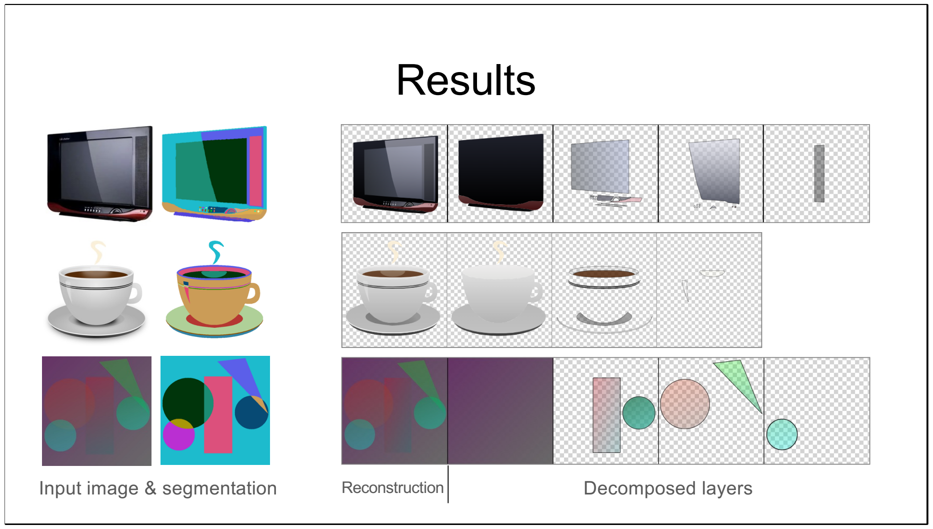
$$E = w_r E_{\text{recon}} + w_g E_{\text{gamut}} + w_c E_{\text{compact}}$$



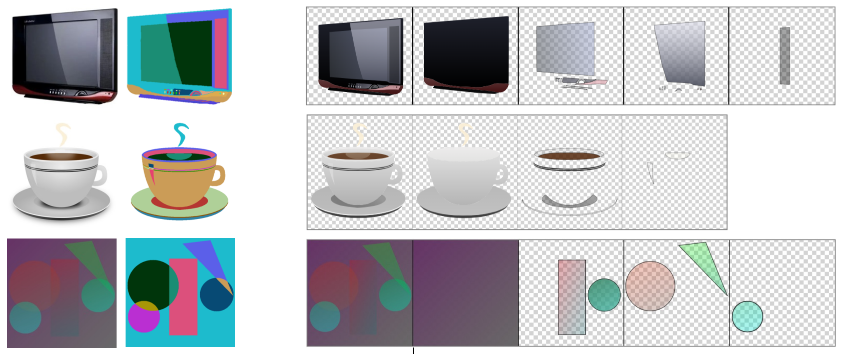
We set $w_r = 20, w_g = 10, w_c = 0.02$ in all our examples.

For each layer configuration, we employ the L-BFGS-B algorithm via the NLopt library to optimize the layer parameters. In all of our examples, we set $w_r = 20, w_g = 10, w_c = 0.02$ to achieve better results.

Finally, when all these candidate layer configurations along with layer parameters are obtained, we select the one with minimal energy loss to generate the resulting vector graph with the Potrace library.



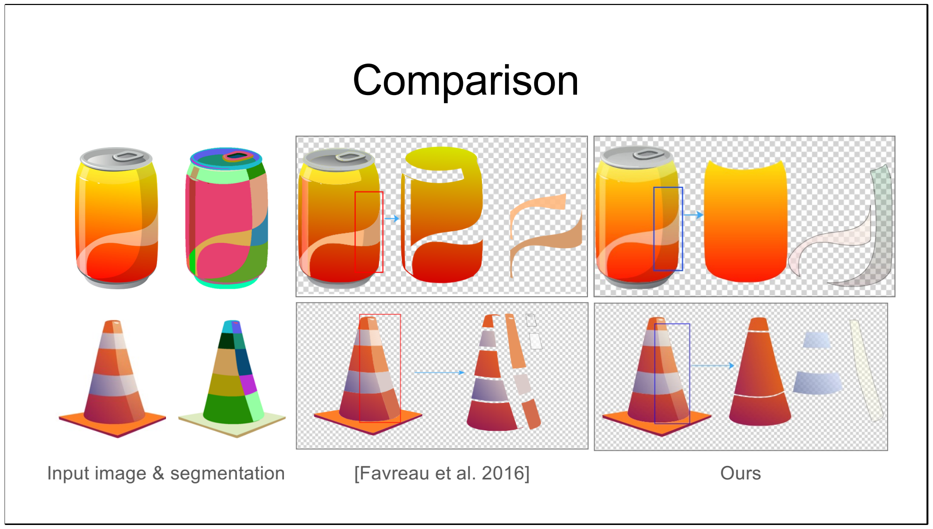
Results



Here we provide 3 decomposition results. We have consistently achieved high-quality decomposition results in all examples.

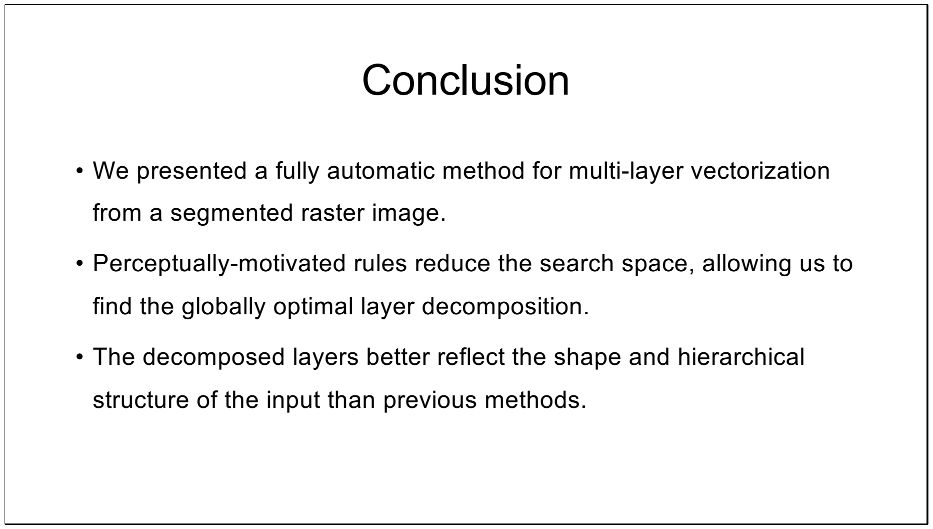
Notice how the highlight regions in TV and coffee cup are all successfully extracted as transparent layers.

Besides, all these 3 examples have x-junctions, our method perfectly recovers the layer stack and achieves reconstruction results with high accuracy.



Comparison

Here we provide 2 results compared to Photo2ClipArt. In general, our method successfully recovers the layer order at each X-junction, and the layers do not contain holes. In contrast, Photo2ClipArt fails to obtain complete semi-transparent layers at x-junction and tends to generate some unnatural layers with holes.



Conclusion

- We presented a fully automatic method for multi-layer vectorization from a segmented raster image.
- Perceptually-motivated rules reduce the search space, allowing us to find the globally optimal layer decomposition.
- The decomposed layers better reflect the shape and hierarchical structure of the input than previous methods.

In conclusion, we proposed a fully automatic approach to multi-layer vectorization without any user interaction, and we introduced perceptually-motivated rules to drastically reduce the search space, allowing us to find globally optimal layer decompositions. For the decomposition results, the layers better reflect the shape and hierarchical structure of the input than SOAT methods.

THE PREMIER CONFERENCE & EXHIBITION ON COMPUTER GRAPHICS & INTERACTIVE TECHNIQUES

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

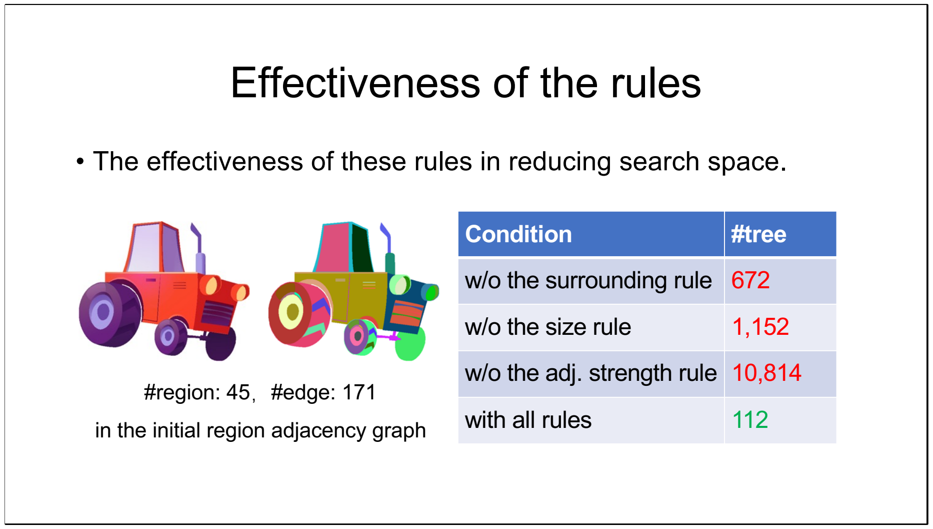
THANK YOU!

- **Contact Information:**
  - Zheng-Jun Du: dzj@qhu.edu.cn
  - Liang-fu Kang: kanglf20@mails.tsinghua.edu.cn
  - Jianchao Tan: tanjianchaoustc@gmail.com
  - Yotam Gingold: ygingold@gmu.edu
  - Kun Xu: xukun@tsinghua.edu.cn

Supplementary

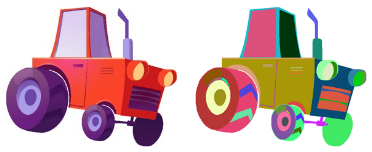
© 2023 SIGGRAPH. ALL RIGHTS RESERVED.

Thanks for you attention.
Please feel free to contact us if you have any questions.



Effectiveness of the rules

- The effectiveness of these rules in reducing search space.



#region: 45, #edge: 171

in the initial region adjacency graph

Condition	#tree
w/o the surrounding rule	672
w/o the size rule	1,152
w/o the adj. strength rule	10,814
with all rules	112

Here we provide an example to illustrate the effectiveness of the perceptually-motivate rules in region adjacency graph simplification.

In the truck example, there are 45 nodes and 171 directed edges in the initial region adjacency graph.

It generates 672, 1,152 and 10,814 region supporting trees when the surrounding rule, the size rule or the adjacent strength rule is disabled.

And it generates only 112 region supporting trees when all rules are enabled.

Notice that in all our examples, the result quality (i.e., the energy loss E) is almost unchanged after enabling the rules.