Layer Opacity Optimization

- Both **palette** and **palette order** are fixed now.
- We solve for layers' opacity values (**find a unique compositing path**)
- We minimize an energy **E** =

Layer Opacity Optimization

- Both **palette** and **palette order** are fixed now.
- We solve for layers' opacity values (**find a unique compositing path**)
- We minimize an energy **E** =

$$\|\text{original} - \text{reconstructed image}\|^2 \text{ (}\textbf{polynomial}\text{)}$$

Layer Opacity Optimization

- Both **palette** and **palette order** are fixed now.
- We solve for layers' opacity values (**find a unique compositing path**)
- We minimize an energy **E** =

$$\|\text{original} - \text{reconstructed image}\|^2 \text{ (\textbf{polynomial})}$$

$$+$$

$$\text{Per pixel opacity sparsity} \quad \sum -(1-\alpha_i)^2$$

Layer Opacity Optimization

- Both **palette** and **palette order** are fixed now.
- We solve for layers' opacity values (**find a unique compositing path**)
- We minimize an energy **E** =

$\|\text{original} - \text{reconstructed image}\|^2$ (**polynomial**)

+

Per pixel opacity sparsity $\sum -(1-\alpha_i)^2$

+

Opacity spatial smoothness (**Laplacian**)

Layer Opacity Optimization

- Both **palette** and **palette order** are fixed now.
- We solve for layers' opacity values (**find a unique compositing path**)
- We minimize an energy **E** =

Shrink compositing paths' solution space

$$\|\text{original} - \text{reconstructed image}\|^2 \text{ (\textbf{polynomial})}$$

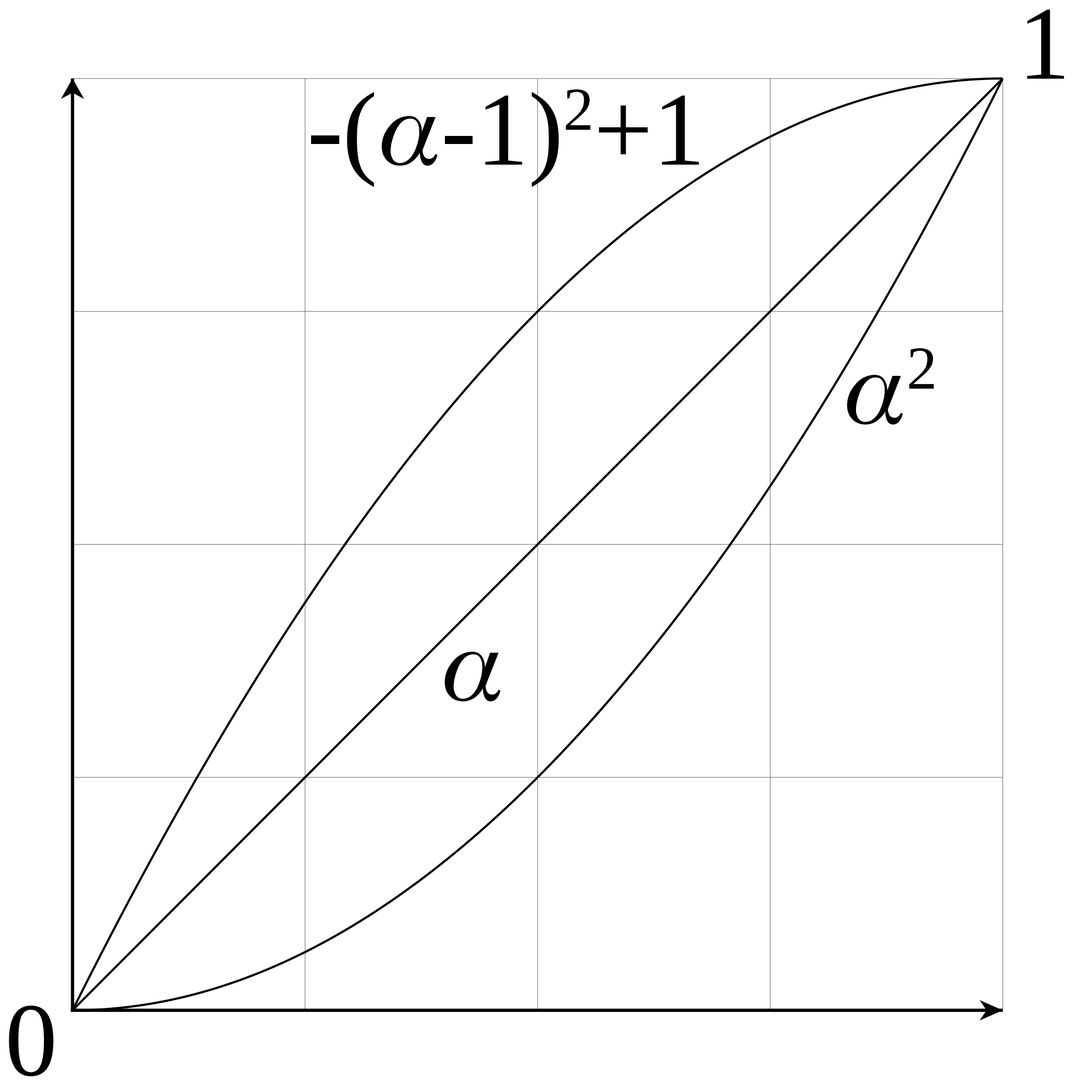
$$+$$

$$\text{Per pixel opacity sparsity} \quad \sum -(1-\alpha_i)^2$$

$$+$$

$$\text{Opacity spatial smoothness (\textbf{Laplacian})}$$

Our sparsity regularization term



		opacity optimization			
image	width × height	runtime (seconds)	RMSE	median error	max error
apple	500 × 453	330.3	1.8	0.0	32.3
bird	640 × 360	500.4	3.7	2.2	60.1
rowboat	589 × 393	981.4	3.3	2.4	19.2
buildings	589 × 393	317.9	2.3	1.4	32.2
cup	400 × 400	40.9	3.0	0.0	34.3
fruit	650 × 414	212.1	1.9	0.0	22.6
girls	589 × 393	152.7	2.4	1.4	29.1
hoover	500 × 500	47.1	3.9	1.0	39.2
light	504 × 538	101.6	2.4	1.0	25.5
robot	450 × 600	519.8	3.5	2.8	14.5
scrooge	410 × 542	97.6	2.6	1.4	15.7
Figure 4	500 × 500	434.3	0.7	0.0	3.3
trees	606 × 404	80.2	5.9	4.2	35.0
turtle	525 × 250	19.5	2.8	1.0	45.4
boat	480 × 600	520.0	4.2	2.2	40.2
castle	747 × 344	280.0	3.7	2.2	45.6
turquoise	480 × 585	498.7	2.0	1.4	17.8
moth	650 × 390	675.1	3.1	1.7	25.7

		opacity optimization			
image	width × height	runtime (seconds)	RMSE	median error	max error
apple	500 × 453	330.3	1.8	0.0	32.3
bird	640 × 360	500.4	3.7	2.2	60.1
rowboat	589 × 393	981.4	3.3	2.4	19.2
buildings	589 × 393	317.9	2.3	1.4	32.2
cup	400 × 400	40.9	3.0	0.0	34.3
fruit	650 × 414	212.1	1.9	0.0	22.6
girls	589 × 393	152.7	2.4	1.4	29.1
hoover	500 × 500	47.1	3.9	1.0	39.2
light	504 × 538	101.6	2.4	1.0	25.5
robot	450 × 600	519.8	3.5	2.8	14.5
scrooge	410 × 542	97.6	2.6	1.4	15.7
Figure 4	500 × 500	434.3	0.7	0.0	3.3
trees	606 × 404	80.2	5.9	4.2	35.0
turtle	525 × 250	19.5	2.8	1.0	45.4
boat	480 × 600	520.0	4.2	2.2	40.2
castle	747 × 344	280.0	3.7	2.2	45.6
turquoise	480 × 585	498.7	2.0	1.4	17.8
moth	650 × 390	675.1	3.1	1.7	25.7

Increase solver's tolerance values

Runtime	RMSE
39	1.9
68	3.8
140	4.2
54	2.3
21	2.8
38	2.0
49	2.7
29	3.9
46	2.3
50	3.7
38	2.6
37	1.5
53	5.8
15	2.8
105	2.2
66	3.6
129	2.7
67	3.3

		opacity optimization			
image	width × height	runtime (seconds)	RMSE	median error	max error
apple	500 × 453	330.3	1.8	0.0	32.3
bird	640 × 360	500.4	3.7	2.2	60.1
rowboat	589 × 393	981.4	3.3	2.4	19.2
buildings	589 × 393	317.9	2.3	1.4	32.2
cup	400 × 400	40.9	3.0	0.0	34.3
fruit	650 × 414	212.1	1.9	0.0	22.6
girls	589 × 393	152.7	2.4	1.4	29.1
hoover	500 × 500	47.1	3.9	1.0	39.2
light	504 × 538	101.6	2.4	1.0	25.5
robot	450 × 600	519.8	3.5	2.8	14.5
scrooge	410 × 542	97.6	2.6	1.4	15.7
Figure 4	500 × 500	434.3	0.7	0.0	3.3
trees	606 × 404	80.2	5.9	4.2	35.0
turtle	525 × 250	19.5	2.8	1.0	45.4
boat	480 × 600	520.0	4.2	2.2	40.2
castle	747 × 344	280.0	3.7	2.2	45.6
turquoise	480 × 585	498.7	2.0	1.4	17.8
moth	650 × 390	675.1	3.1	1.7	25.7

Increase solver's tolerance values

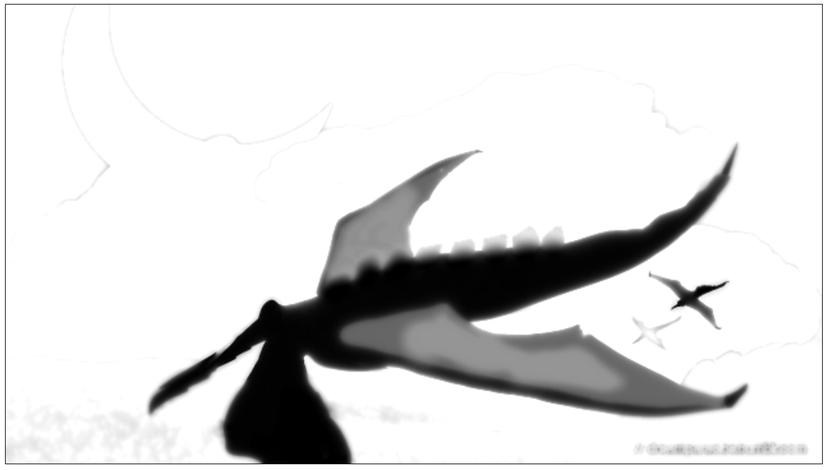
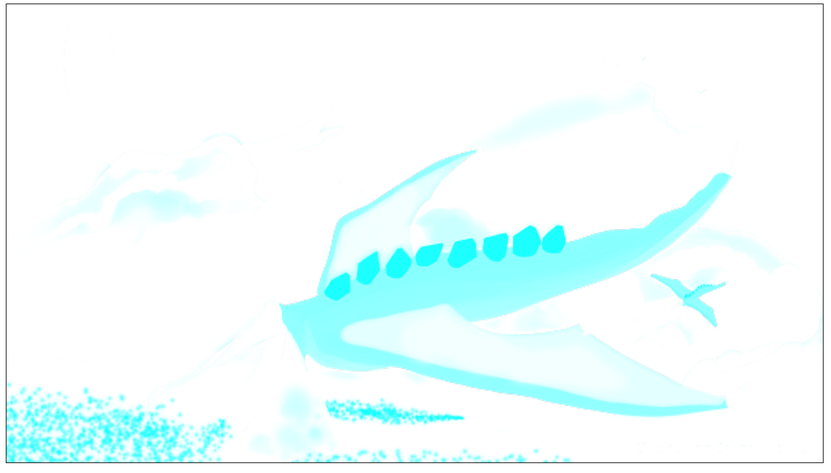
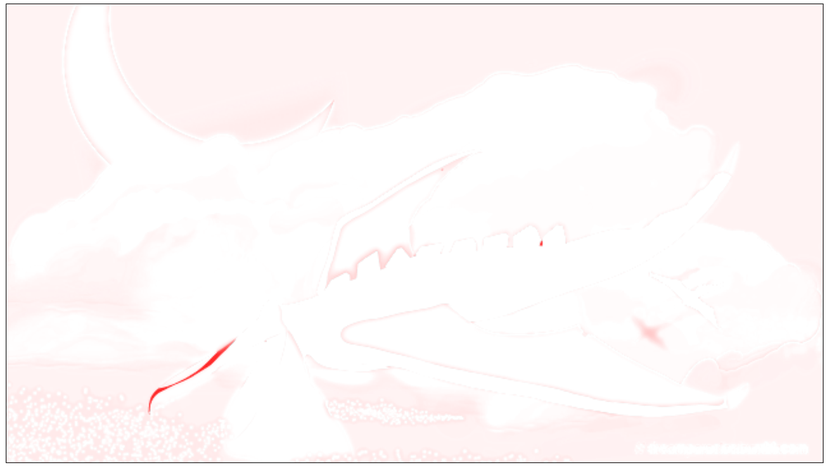
Runtime	RMSE
39	1.9
68	3.8
140	4.2
54	2.3
21	2.8
38	2.0
49	2.7
29	3.9
46	2.3
50	3.7
38	2.6
37	1.5
53	5.8
15	2.8
105	2.2
66	3.6
129	2.7
67	3.3

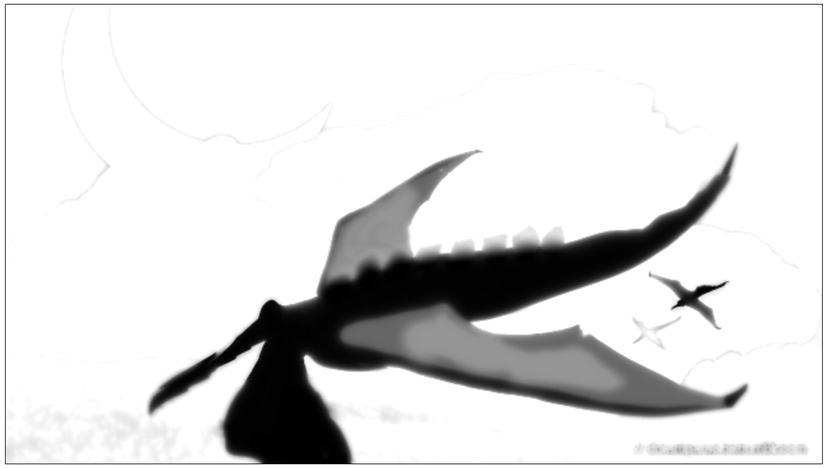
Much faster!

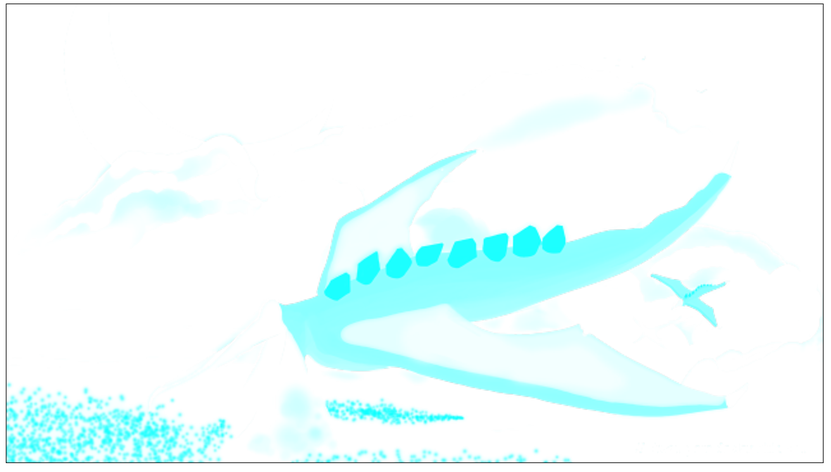
Results

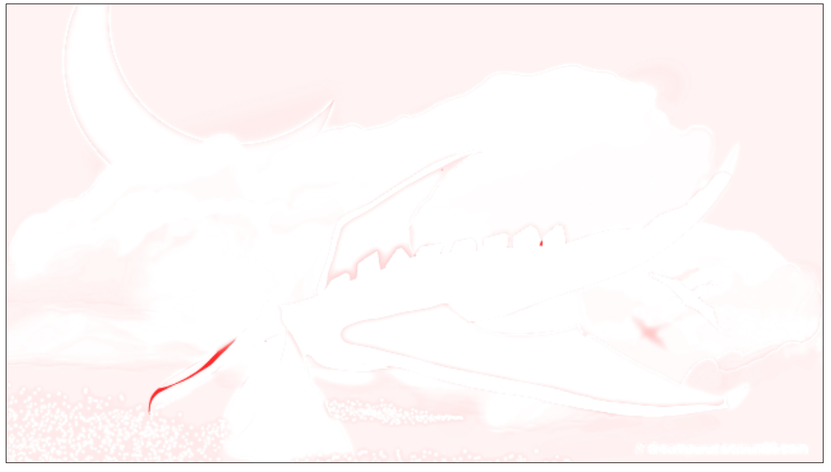

// dreamparacite.tumblr.com

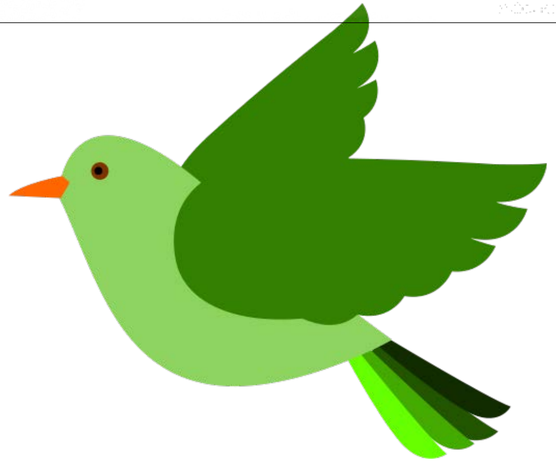


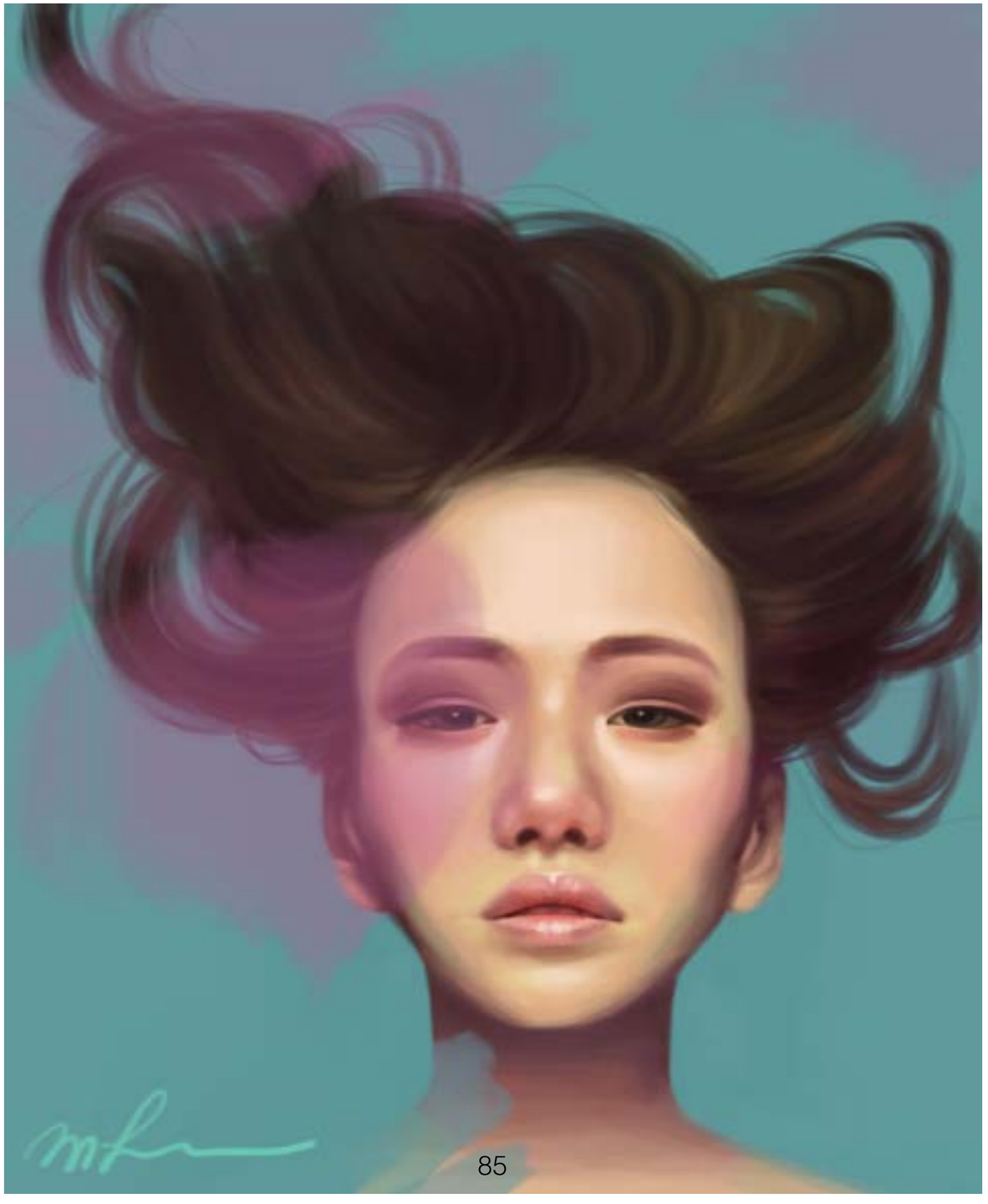










// dreamparacite.tumblr.com

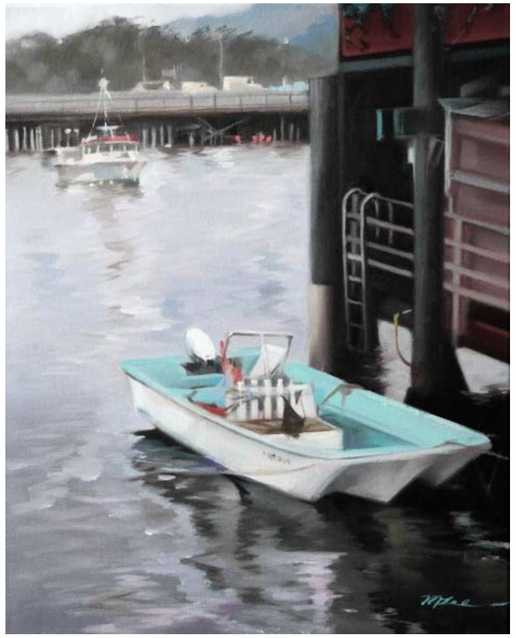


http://cargocollective.com/michellelee/

http://cargocollective.com/michellelee/

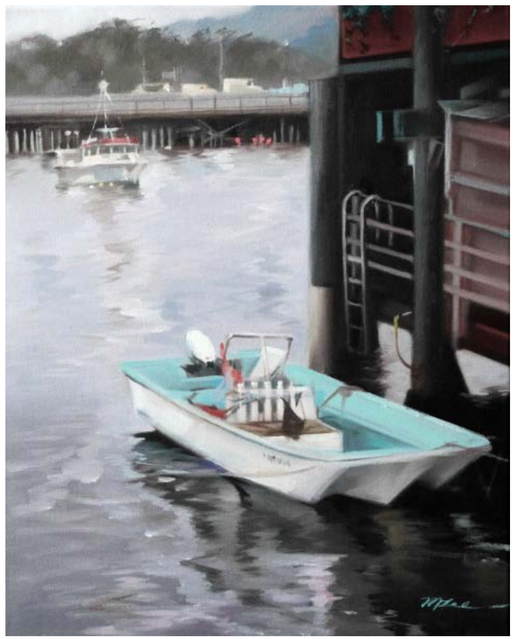
Local Recoloring

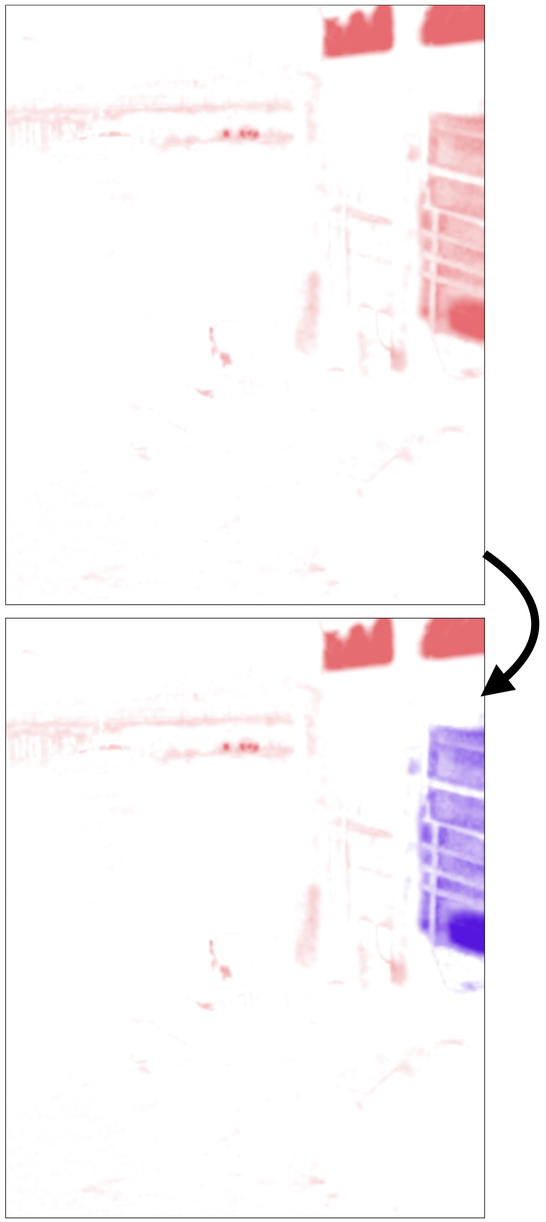
Original



Local Recoloring

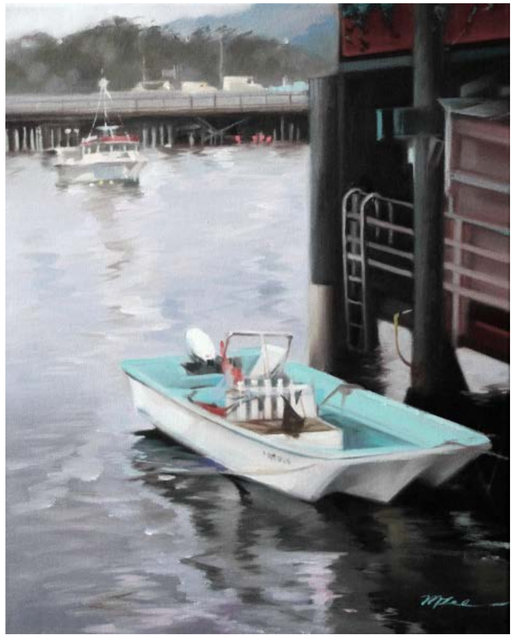
Original

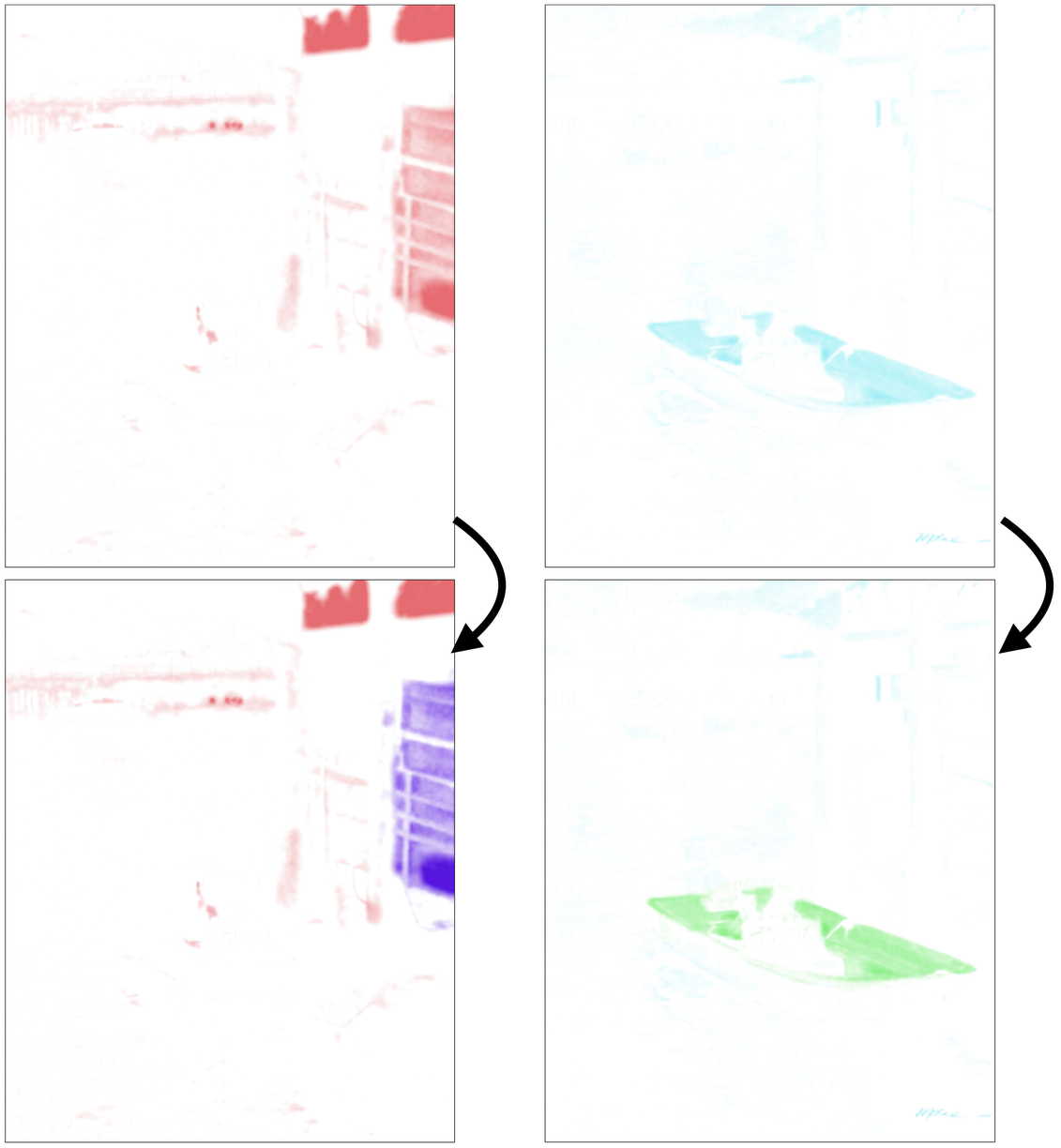




Local Recoloring

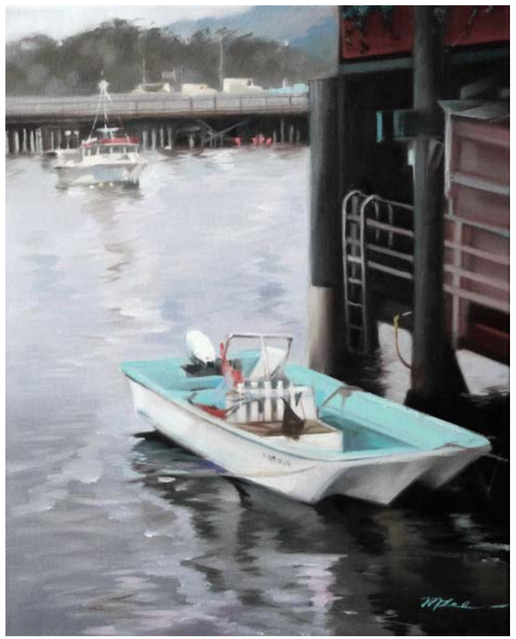
Original

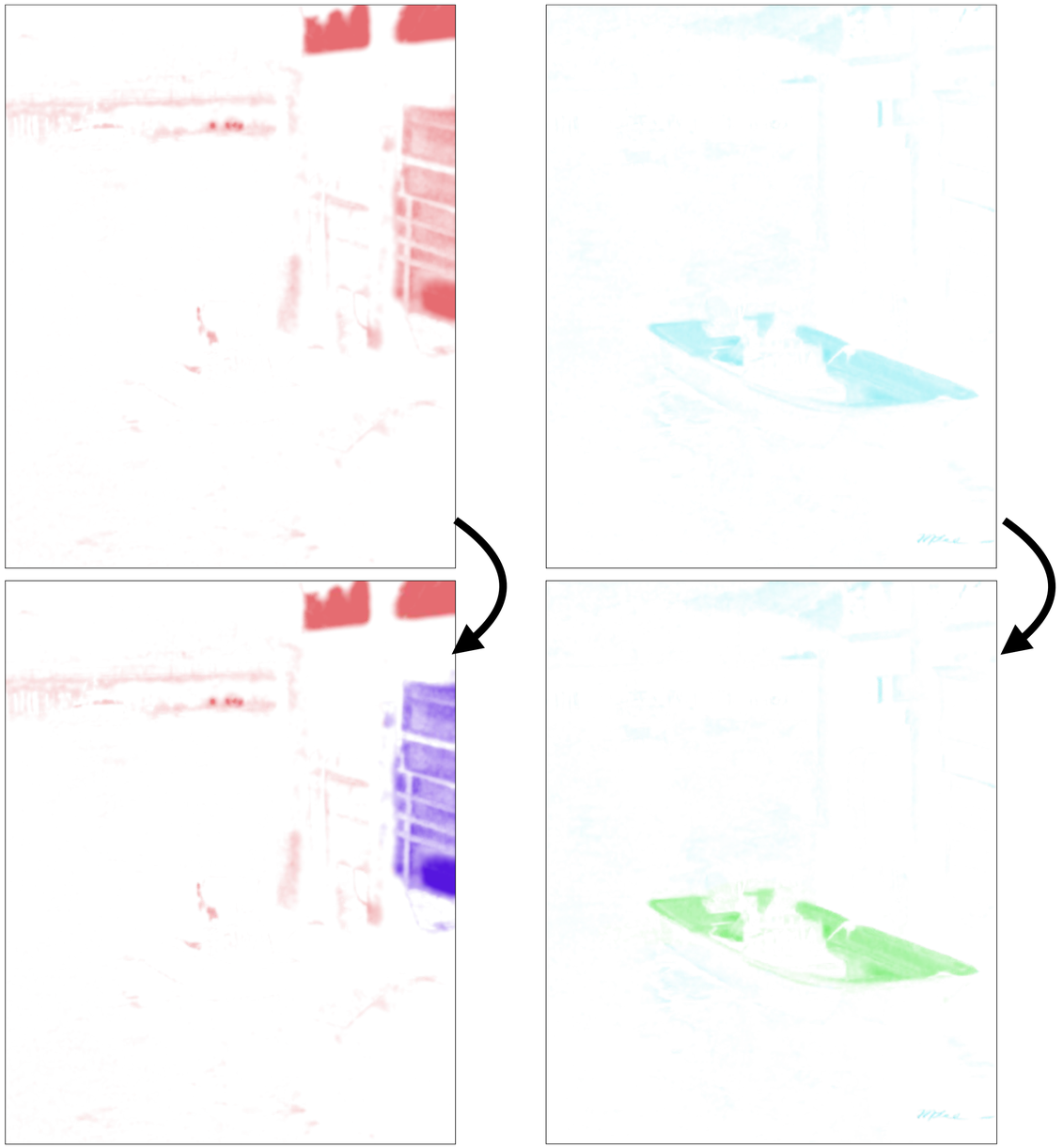




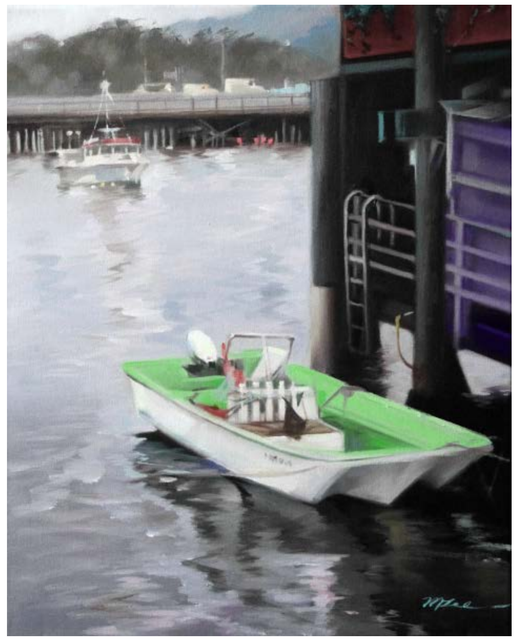
Local Recoloring

Original





Modified

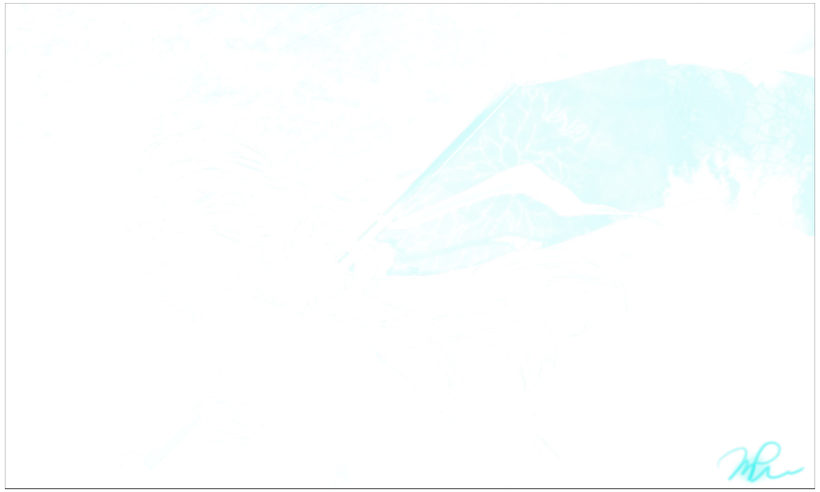


Local Recoloring

Original

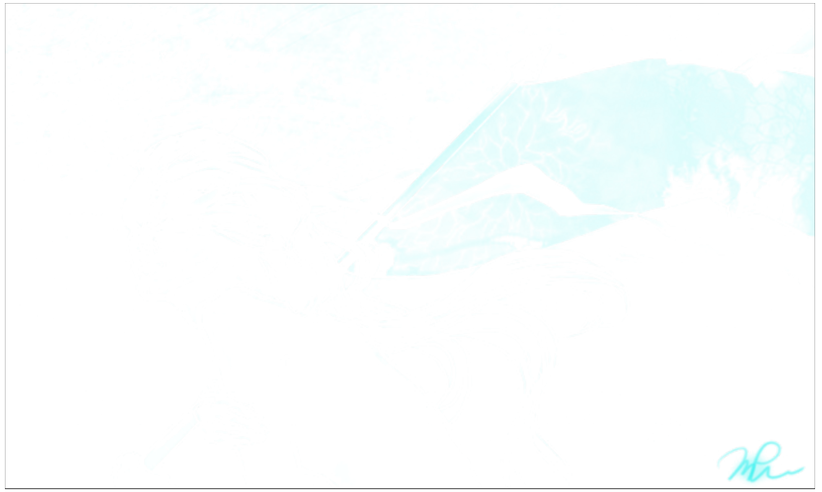
Local Recoloring

Original



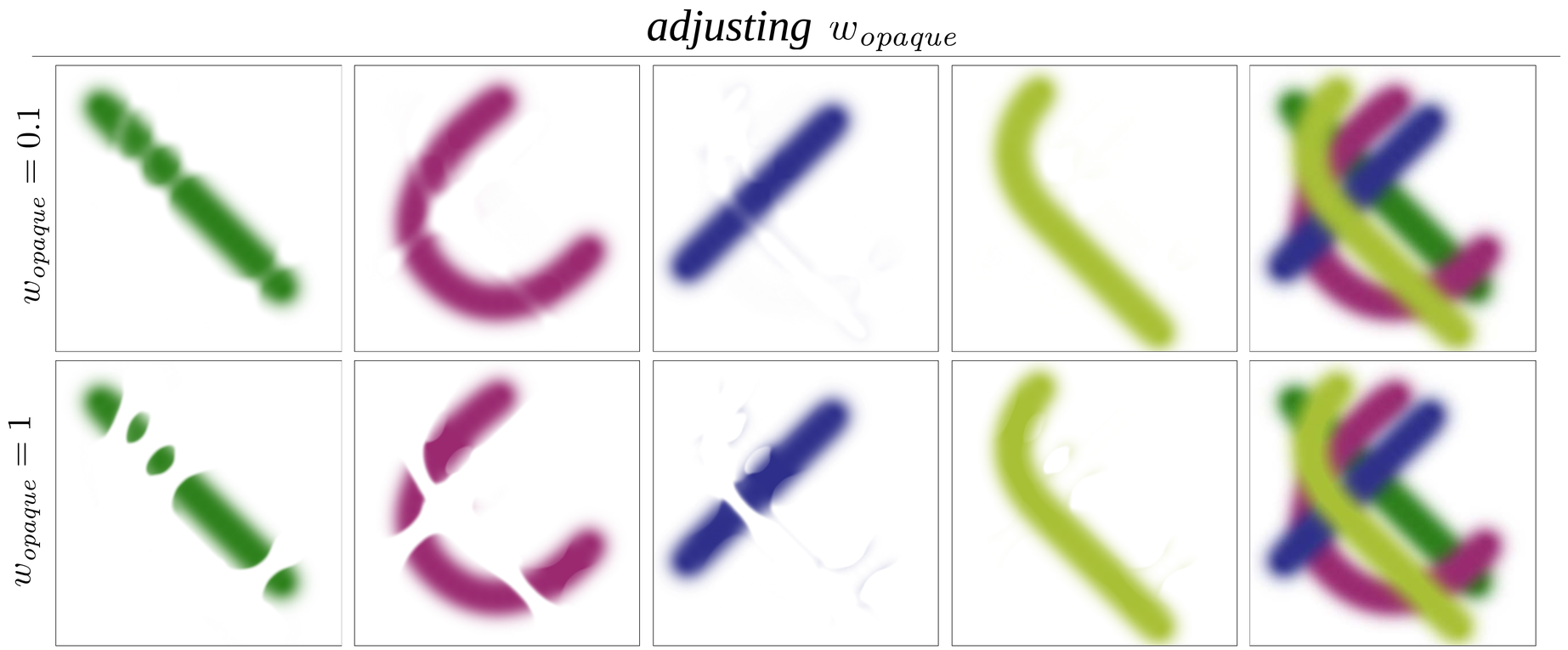
Local Recoloring

Original

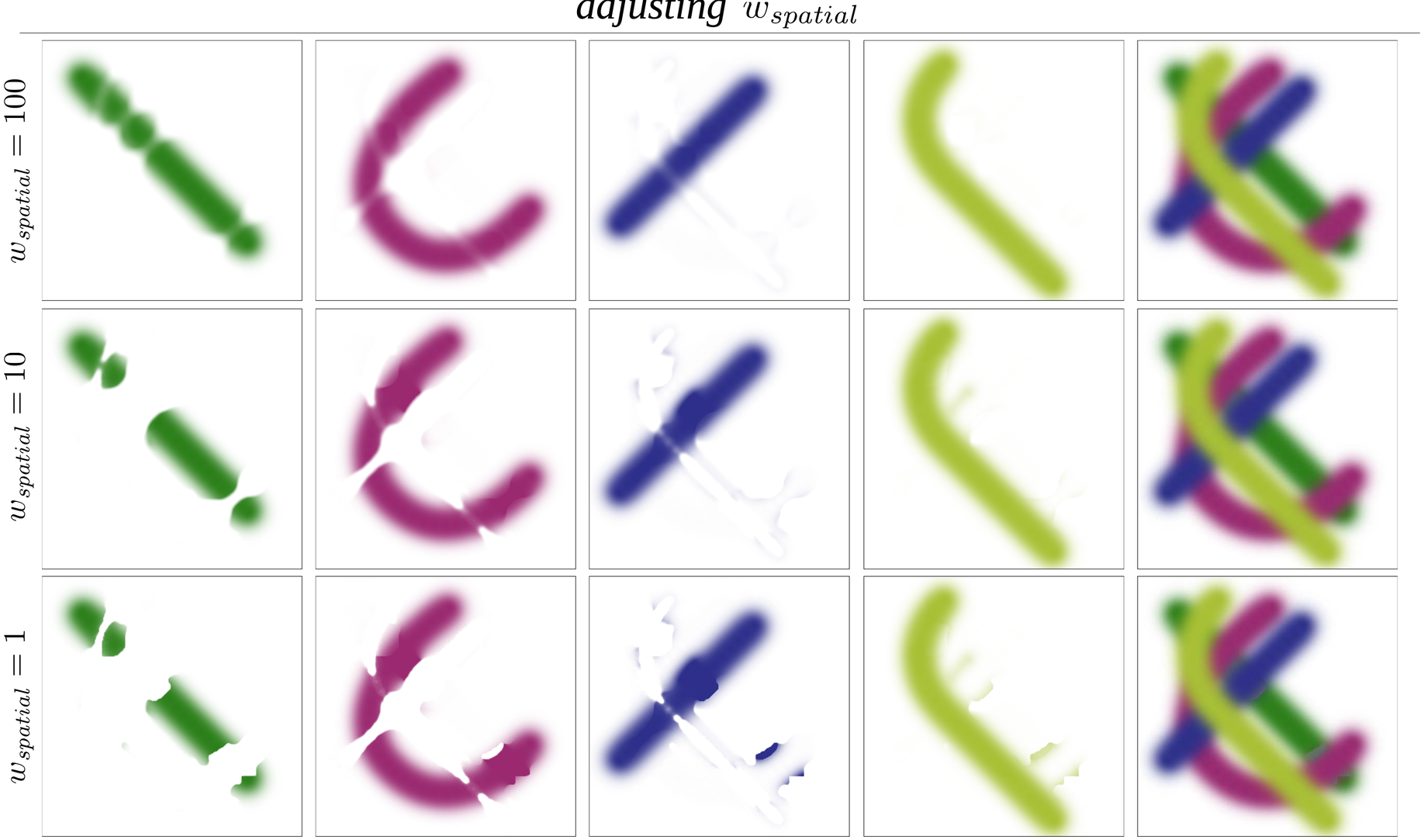


Modified

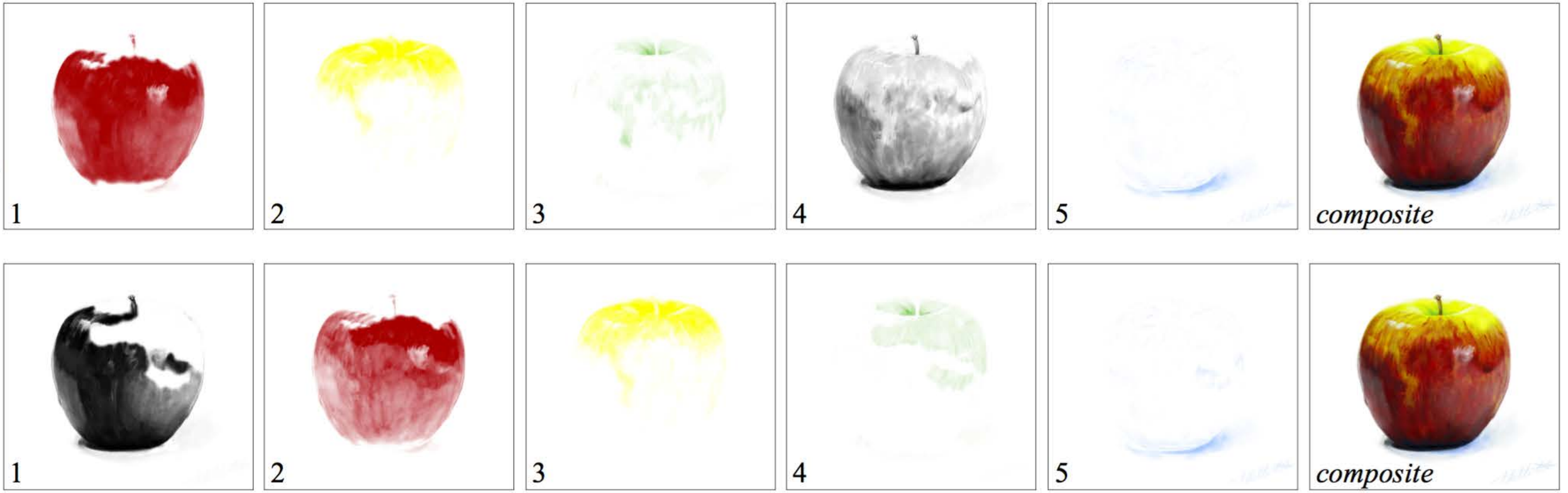
Optimization parameters influence



Optimization parameters influence



Layer order influence



Generalized Barycentric Coordinates

$$p = \sum w_i c_i$$

Generalized Barycentric Coordinates

$$p = \sum w_i c_i$$ linear mixing weights

Generalized Barycentric Coordinates

$$p = \sum w_i c_i$$ linear mixing weights

$$p = c_n + \sum_{i=1}^{n} \left[(c_{i-1} - c_i) \prod_{j=i}^{n} (1 - \alpha_j) \right]$$

Alpha Compositing

Generalized Barycentric Coordinates

$$p = \sum w_i c_i$$ linear mixing weights

$$p = c_n + \sum_{i=1}^{n} \left[(c_{i-1} - c_i) \prod_{j=i}^{n} (1 - \alpha_j) \right]$$ layer opacities

Alpha Compositing

Generalized Barycentric Coordinates

$$p = \sum w_i c_i$$

linear mixing weights

Unique

$$p = c_n + \sum_{i=1}^{n} \left[(c_{i-1} - c_i) \prod_{j=i}^{n} (1 - \alpha_j) \right]$$

layer opacities

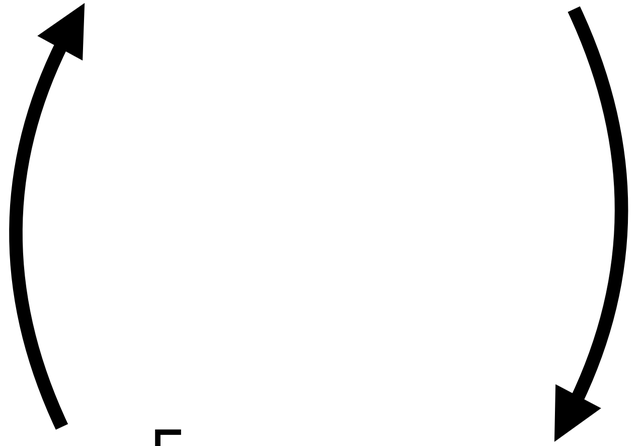
Alpha Compositing

Generalized Barycentric Coordinates

$$p = \sum w_i c_i$$

linear mixing weights

Unique



Ambiguous

$$p = c_n + \sum_{i=1}^{n} \left[(c_{i-1} - c_i) \prod_{j=i}^{n} (1 - \alpha_j) \right]$$

layer opacities

Alpha Compositing

GBC and Opacity conversions

GBC and Opacity conversions

Opacity to GBC (Unique)

$$w_i = \begin{cases} \prod_{j=i+1}^{n}(1-\alpha_j) & \text{if } i = 0 \\ \left(\prod_{j=i+1}^{n}(1-\alpha_j)\right) - \left(\prod_{j=i}^{n}(1-\alpha_j)\right) & \text{if } 0 < i < n \\ 1 - \prod_{j=i}^{n}(1-\alpha_j) = \alpha_i & \text{if } i = n \end{cases}$$

GBC and Opacity conversions

Opacity to GBC (Unique)

$$w_i = \begin{cases} \prod_{j=i+1}^{n}(1-\alpha_j) & \text{if } i = 0 \\ \left(\prod_{j=i+1}^{n}(1-\alpha_j)\right) - \left(\prod_{j=i}^{n}(1-\alpha_j)\right) & \text{if } 0 < i < n \\ 1 - \prod_{j=i}^{n}(1-\alpha_j) = \alpha_i & \text{if } i = n \end{cases}$$

GBC to Opacity (Ambiguous)

$$\alpha_i = \begin{cases} 1 - \frac{\sum_{j=0}^{i-1} w_j}{\sum_{j=0}^{i} w_j} & \text{if } \sum_{j=0}^{i} w_j \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

GBC and Opacity conversions

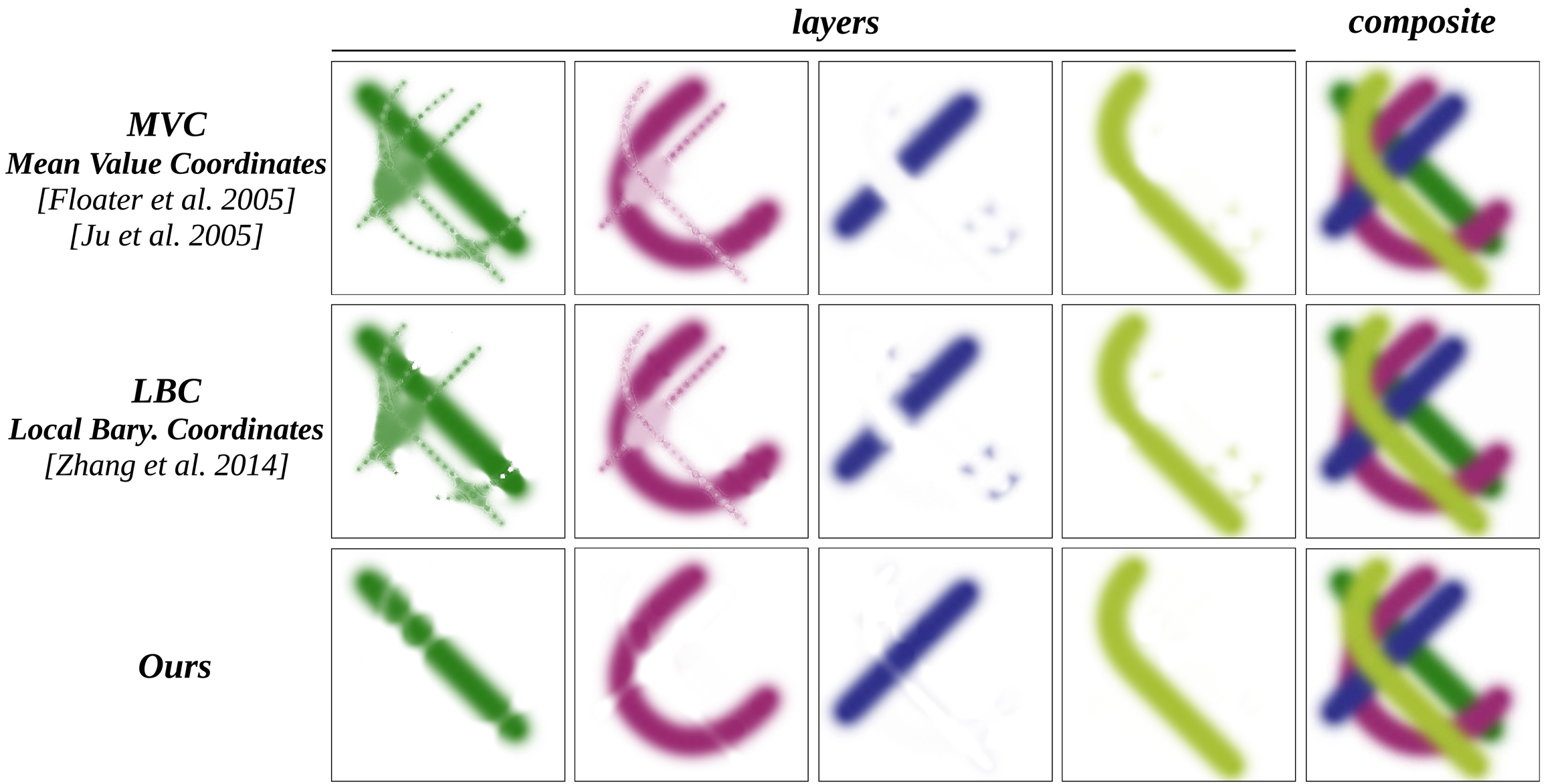
Opacity to GBC (Unique)

$$w_i = \begin{cases} \prod_{j=i+1}^{n}(1-\alpha_j) & \text{if } i = 0 \\ \left(\prod_{j=i+1}^{n}(1-\alpha_j)\right) - \left(\prod_{j=i}^{n}(1-\alpha_j)\right) & \text{if } 0 < i < n \\ 1 - \prod_{j=i}^{n}(1-\alpha_j) = \alpha_i & \text{if } i = n \end{cases}$$

GBC to Opacity (Ambiguous)

$$\alpha_i = \begin{cases} 1 - \frac{\sum_{j=0}^{i-1} w_j}{\sum_{j=0}^{i} w_j} & \text{if } \sum_{j=0}^{i} w_j \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Generalized Barycentric Coordinates



Demo

Our layer opacities can also be uniquely converted into Generalized Barycentric Coordinates.

Actually, we now get additive mixing layers, which is layer order independent.

Global Recoloring

Visualize the colors of an image as a 3D RGB point cloud.

apple.png

width: 500, height: 453

total pixels: 226500

unique pixels: 226500

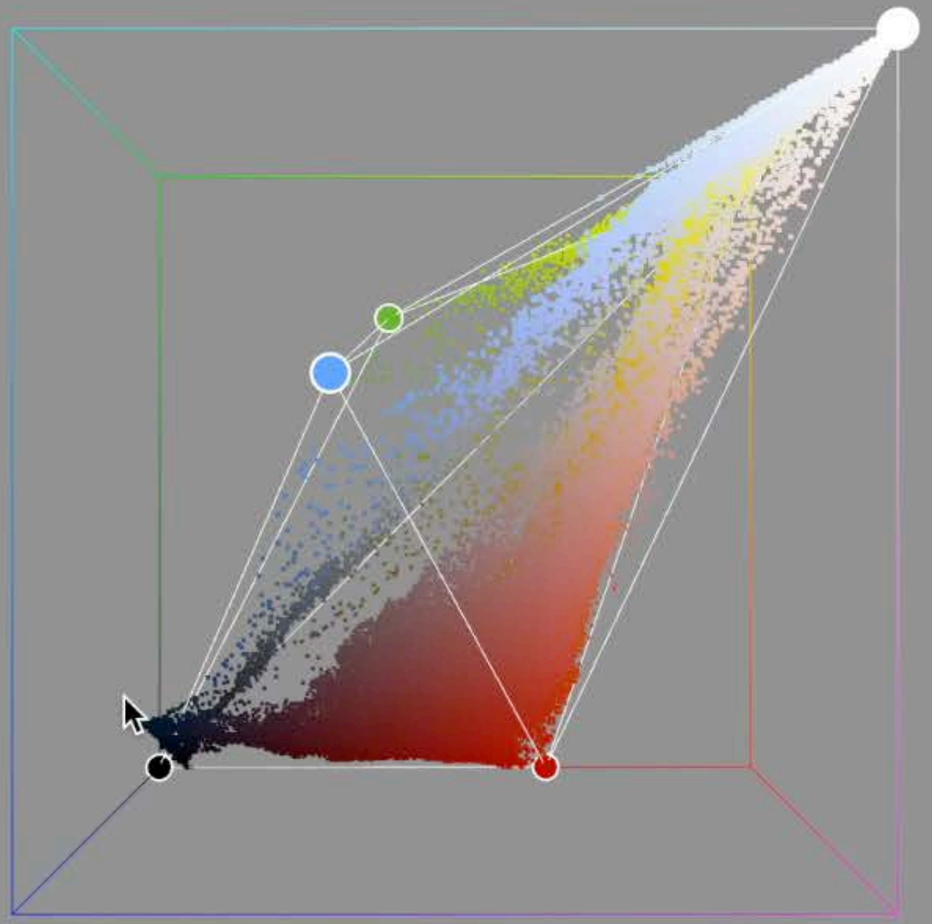
Choose File No file chosen

Rotation has intertia:

Look from white

Save Everything

Save Camera Only



Visualize the colors of an image as a 3D RGB point cloud.

apple.png

width: 500, height: 453

total pixels: 226500

unique pixels: 226500

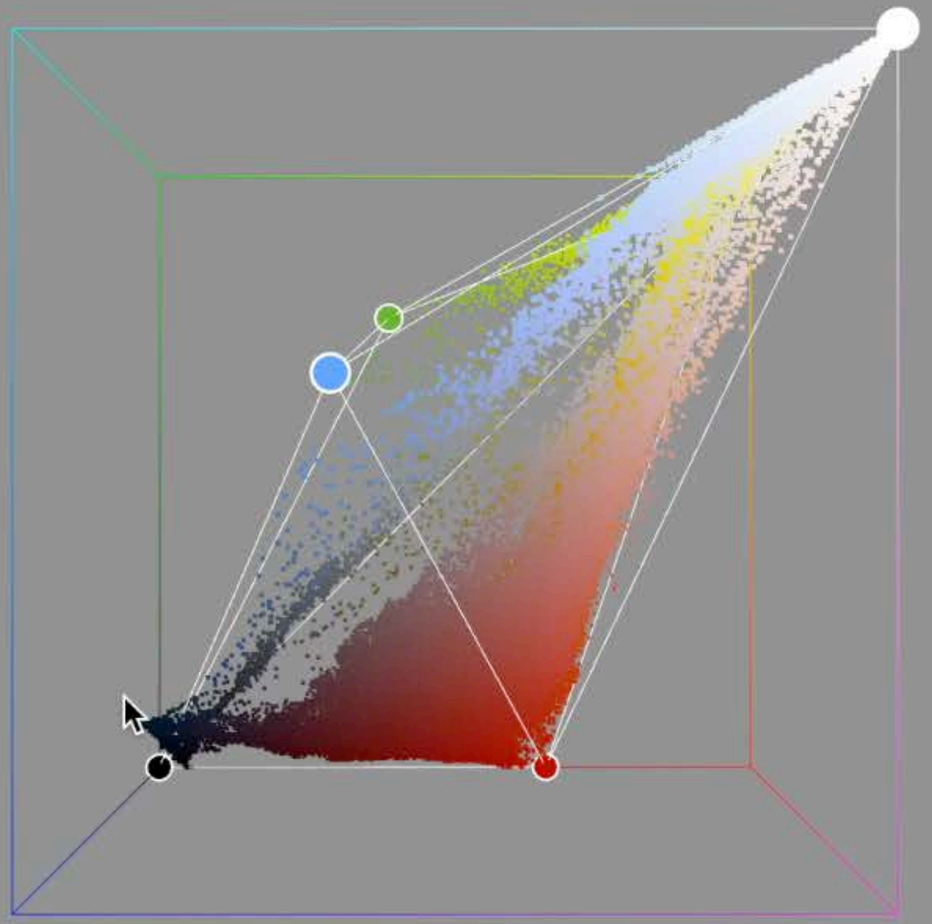
Choose File No file chosen

Rotation has intertia:

Look from white

Save Everything

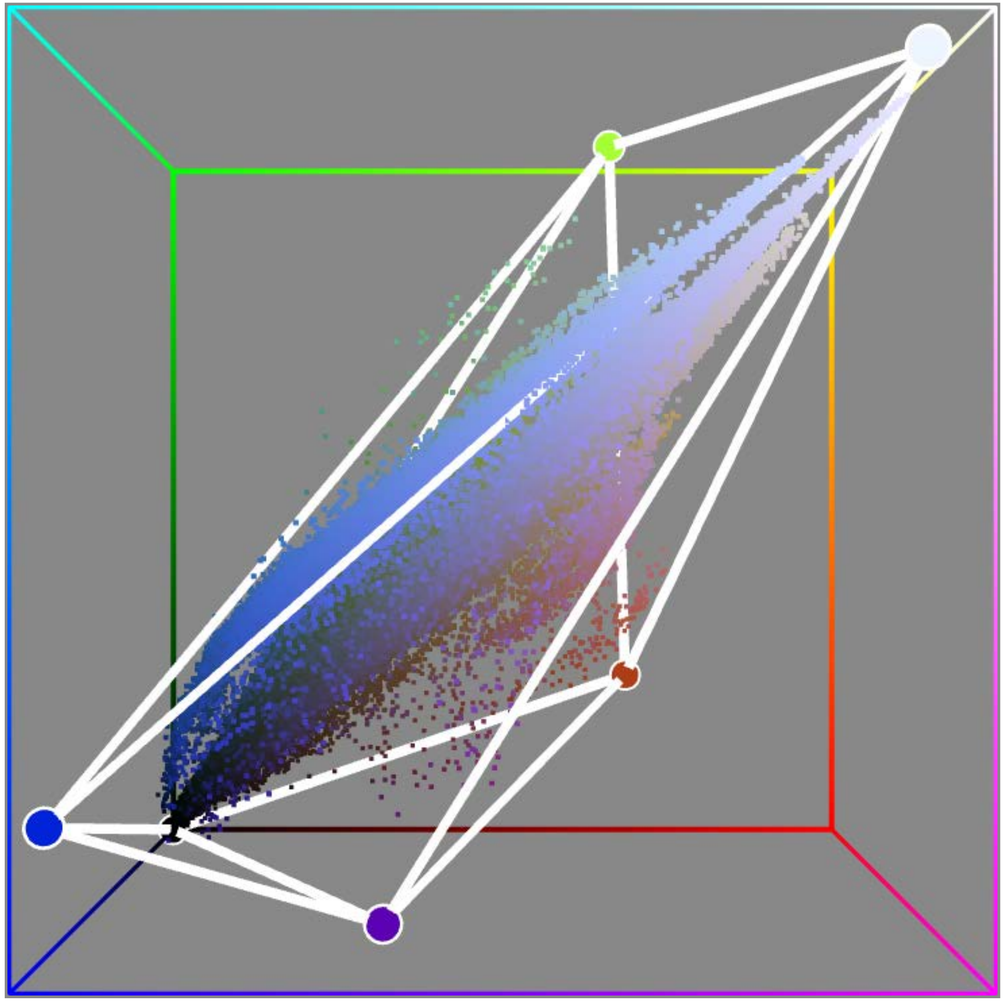
Save Camera Only



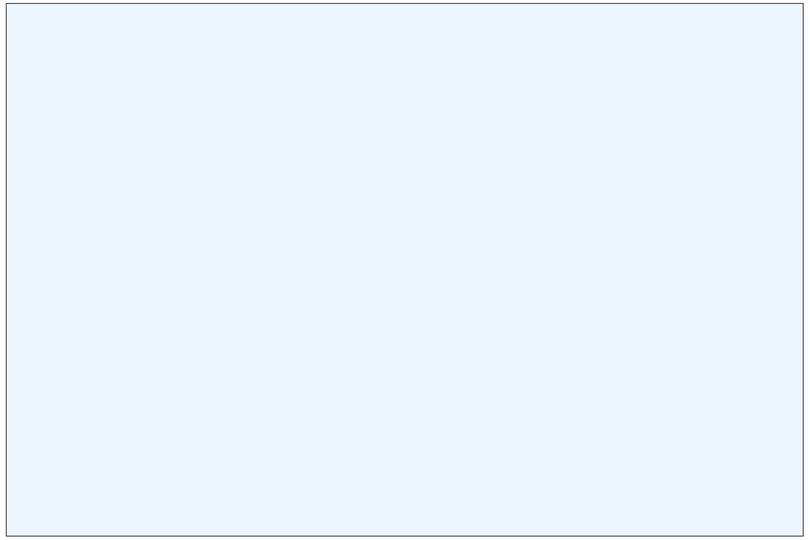
Natural Images

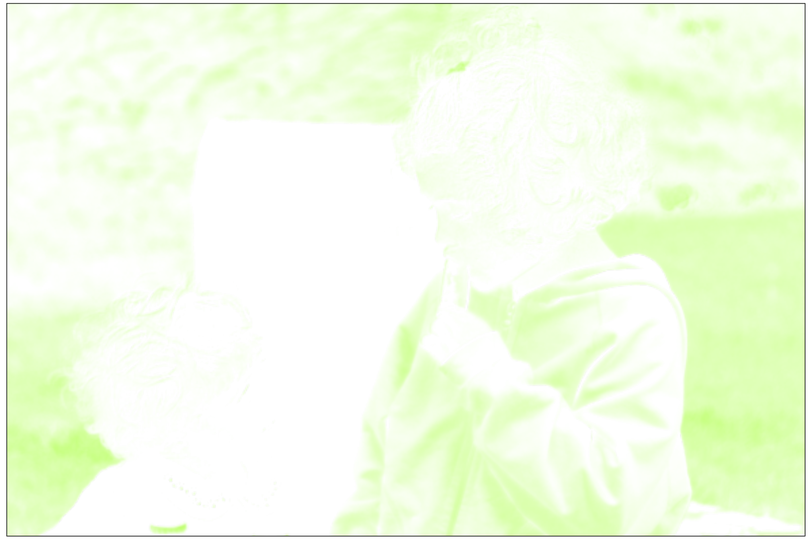
Original

Simplified hull



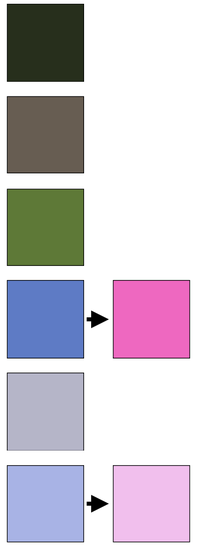
layers



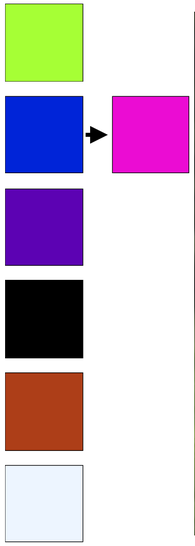


Global Recoloring Comparison

Original



Chang et al. 2015



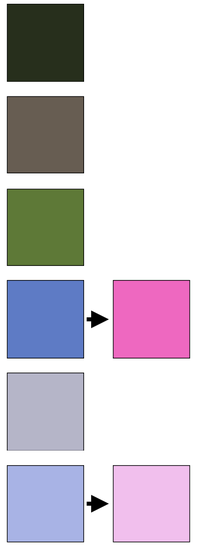
Ours

MVC

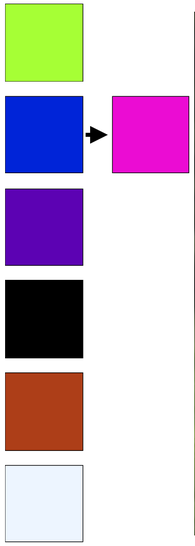
LBC

Global Recoloring Comparison

Original



Chang et al. 2015



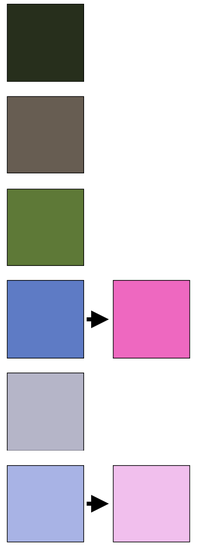
Ours

MVC

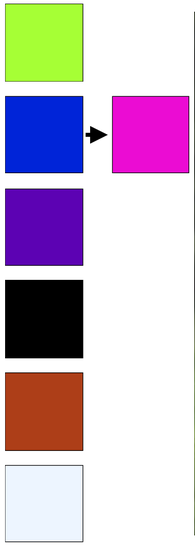
LBC

Global Recoloring Comparison

Original



Chang et al. 2015



Ours

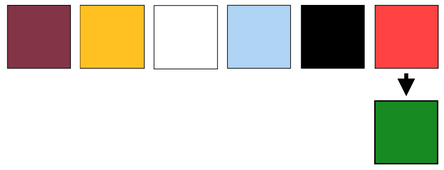
MVC

LBC

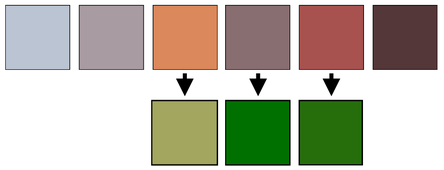
Global Recoloring Comparison

Original

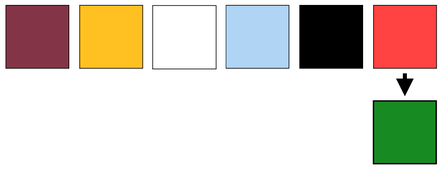
Ours



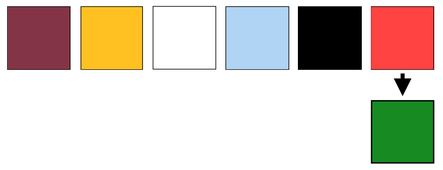
Chang et al. 2015



MVC



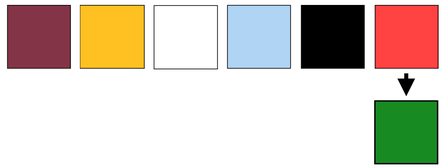
LBC



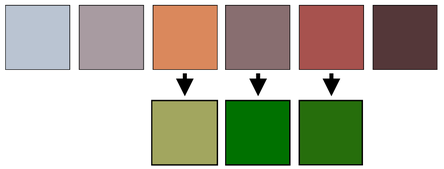
Global Recoloring Comparison

Original

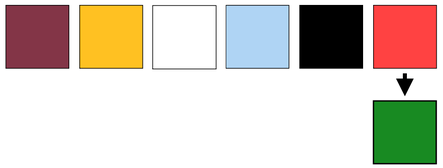
Ours



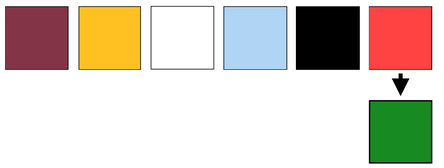
Chang et al. 2015



MVC



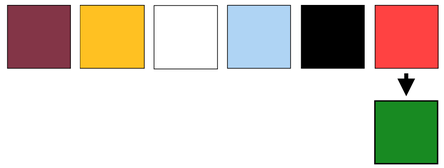
LBC



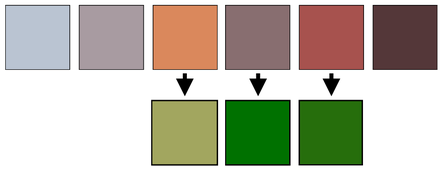
Global Recoloring Comparison

Original

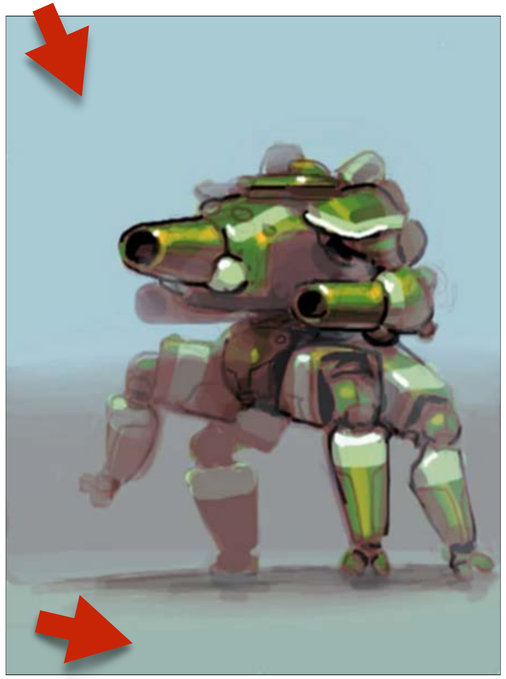
Ours

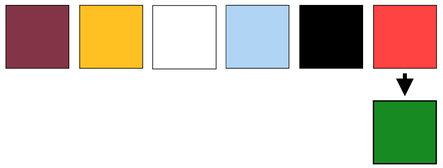


Chang et al. 2015

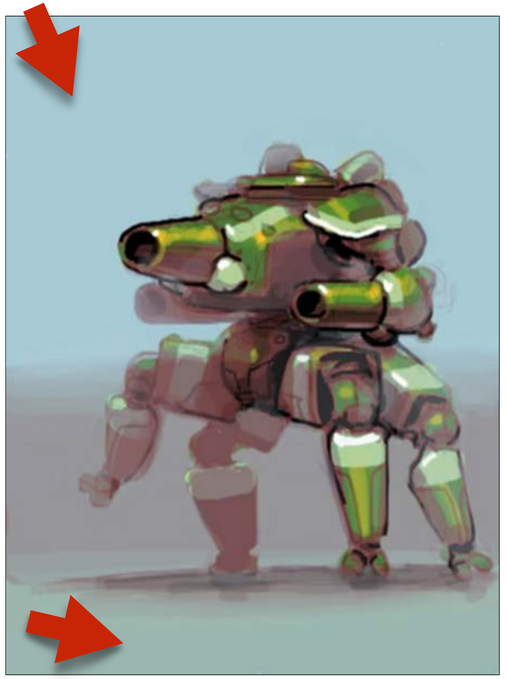


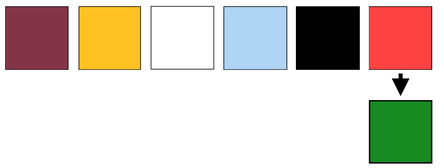
MVC





LBC





Weights Sparsity Comparison

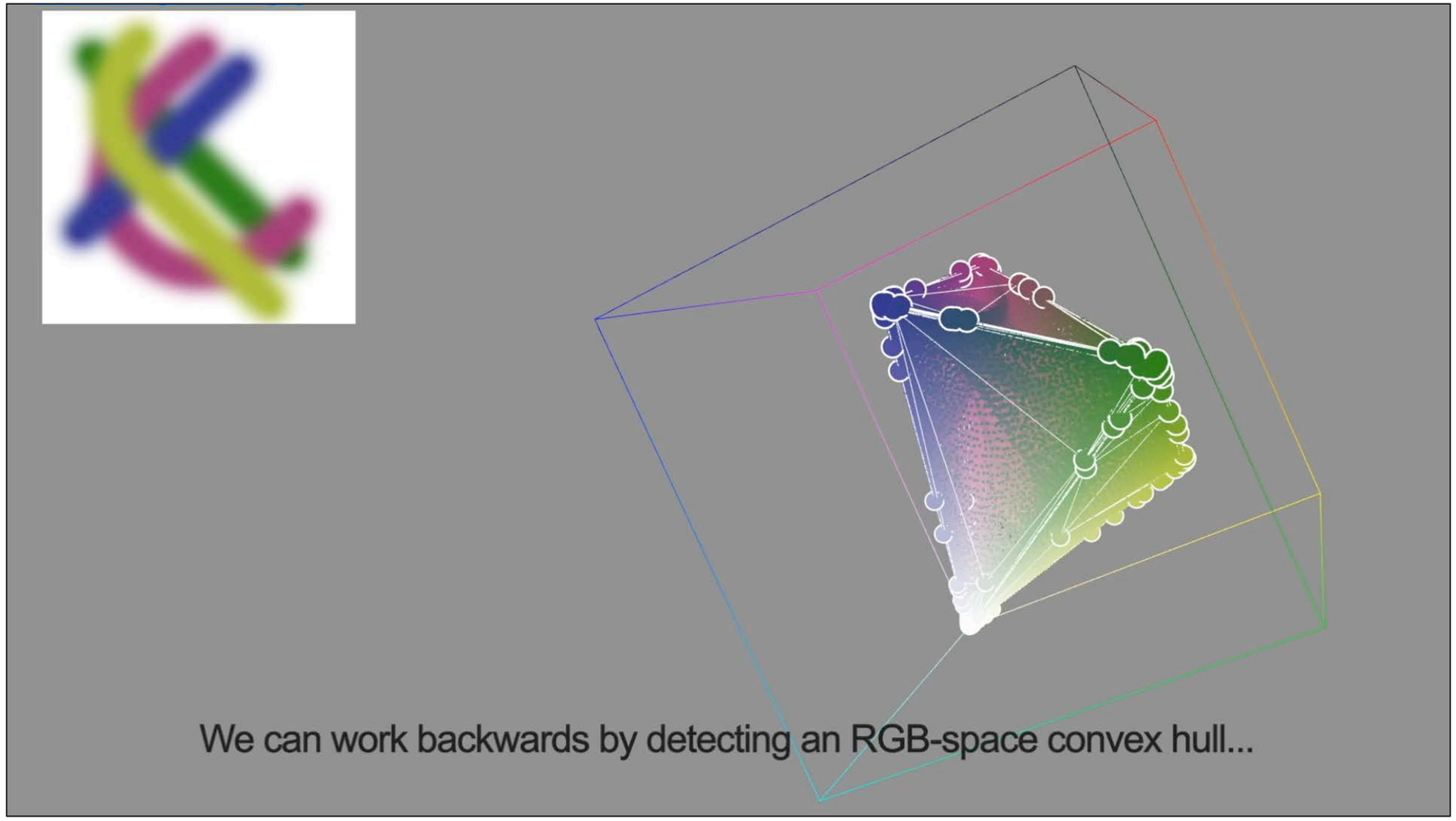
image	MVC	LBC	Our optimization
apple	0.54	0.59	0.62
rowboat	0.11	0.34	0.44
girls	0.10	0.22	0.36
robot	0.01	0.14	0.47
scrooge	0.30	0.38	0.51
Figure 4	0.33	0.38	0.41
boat	0.08	0.24	0.49

Sparsity is computed as the fraction of near-zero values in weights matrix.

$$\epsilon = \frac{1/512}{\#\text{vertices}}$$

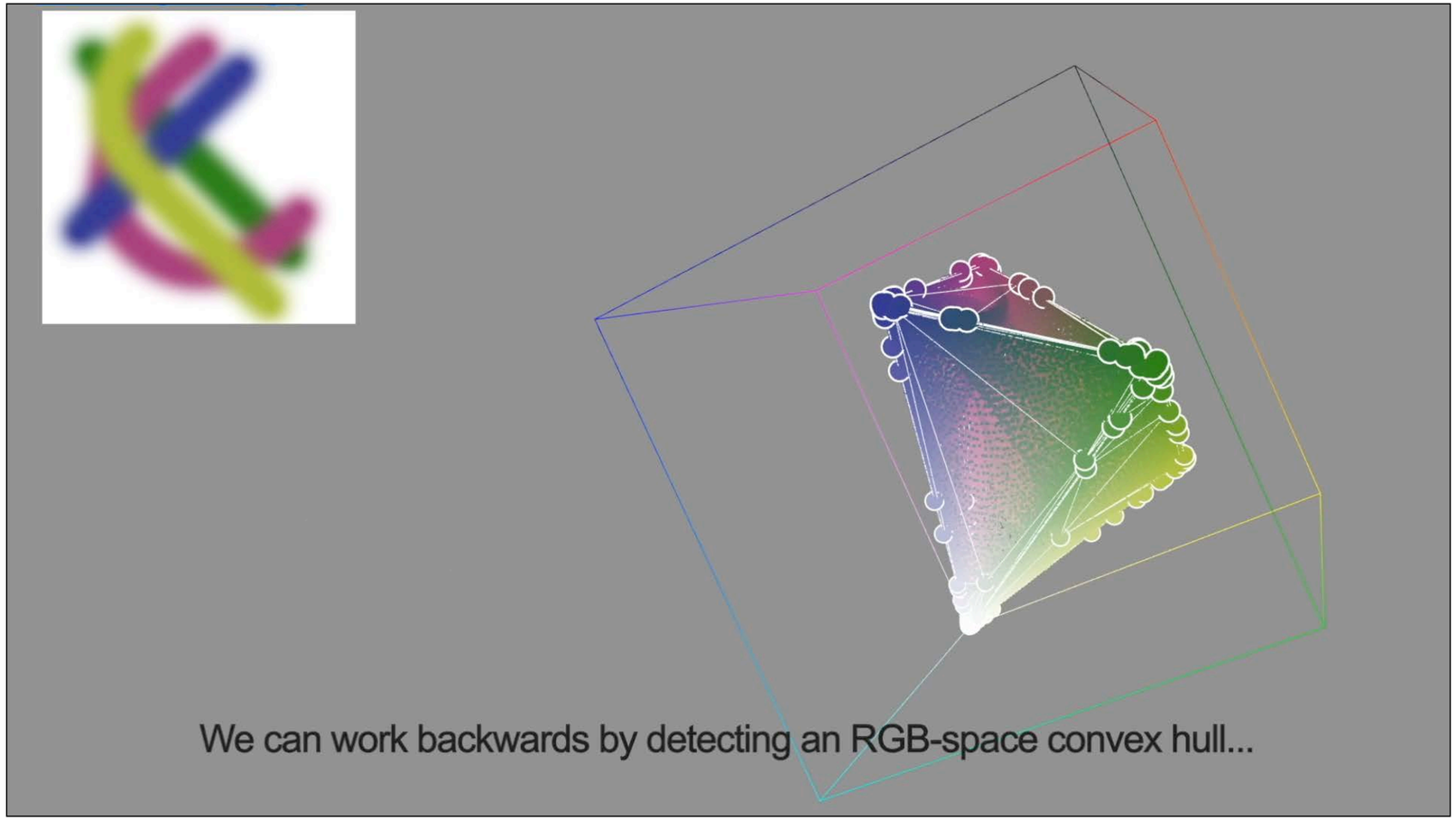
Summary

- The RGB-space geometry of an image contains a hidden geometric structure.



Summary

- The RGB-space geometry of an image contains a hidden geometric structure.



Summary

- We regularize the under-constrained layer opacity problem by balancing sparsity and smoothness.

Original

We then solve an optimization problem to extract translucent layers.

Summary

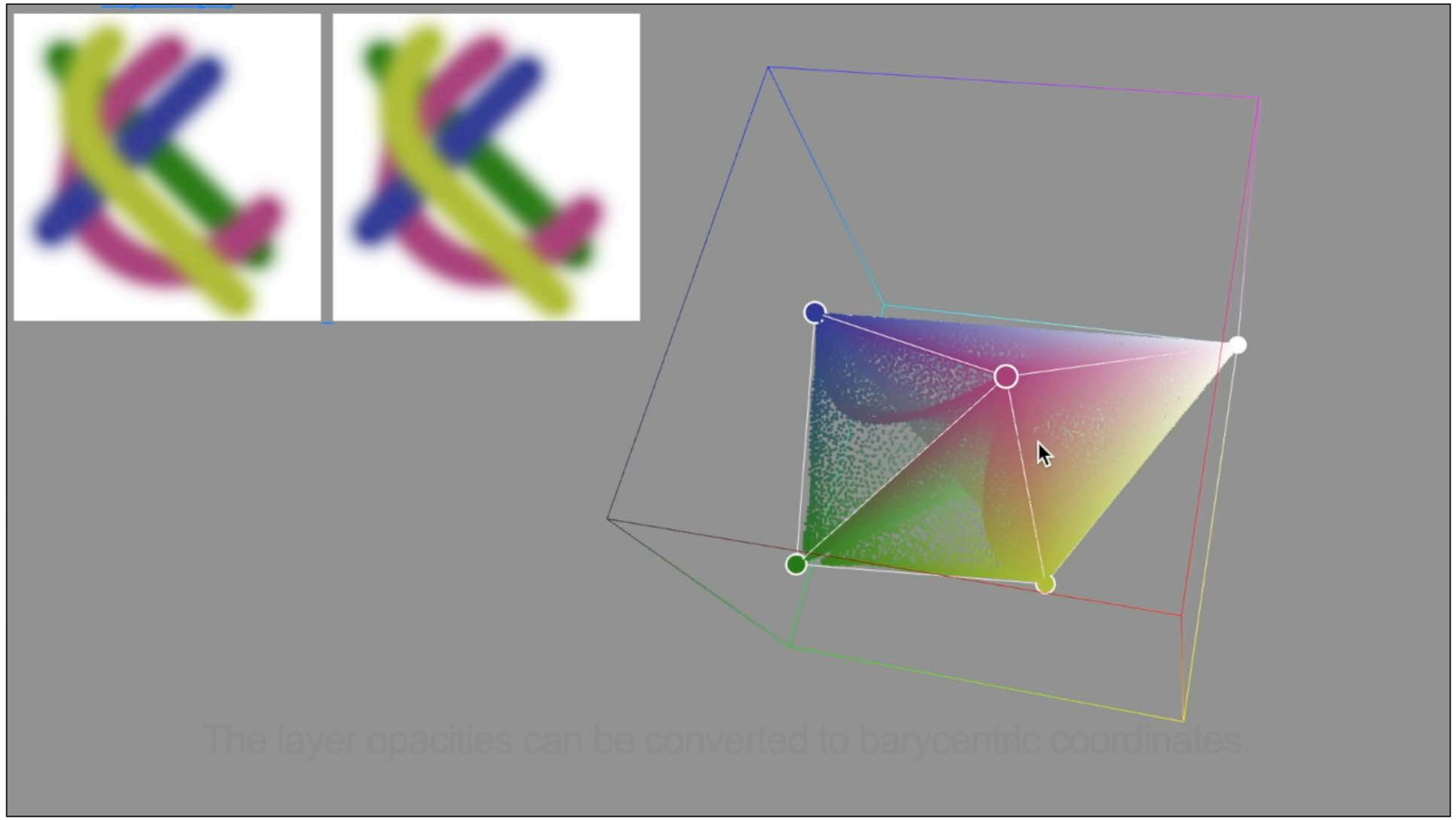
- We regularize the under-constrained layer opacity problem by balancing sparsity and smoothness.

Original

We then solve an optimization problem to extract translucent layers.

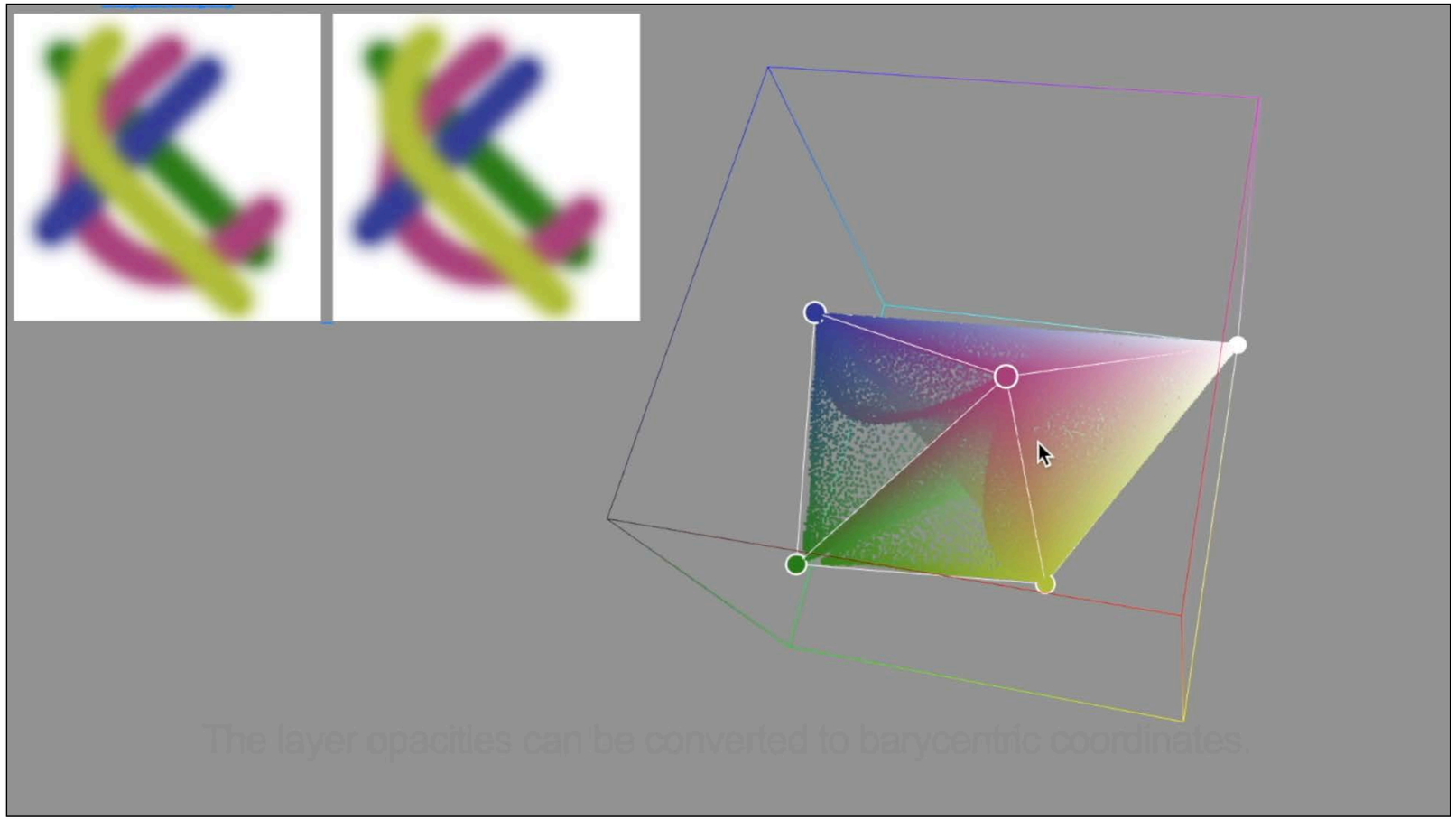
Summary

- The layers can be edited or converted to generalized barycentric coordinates



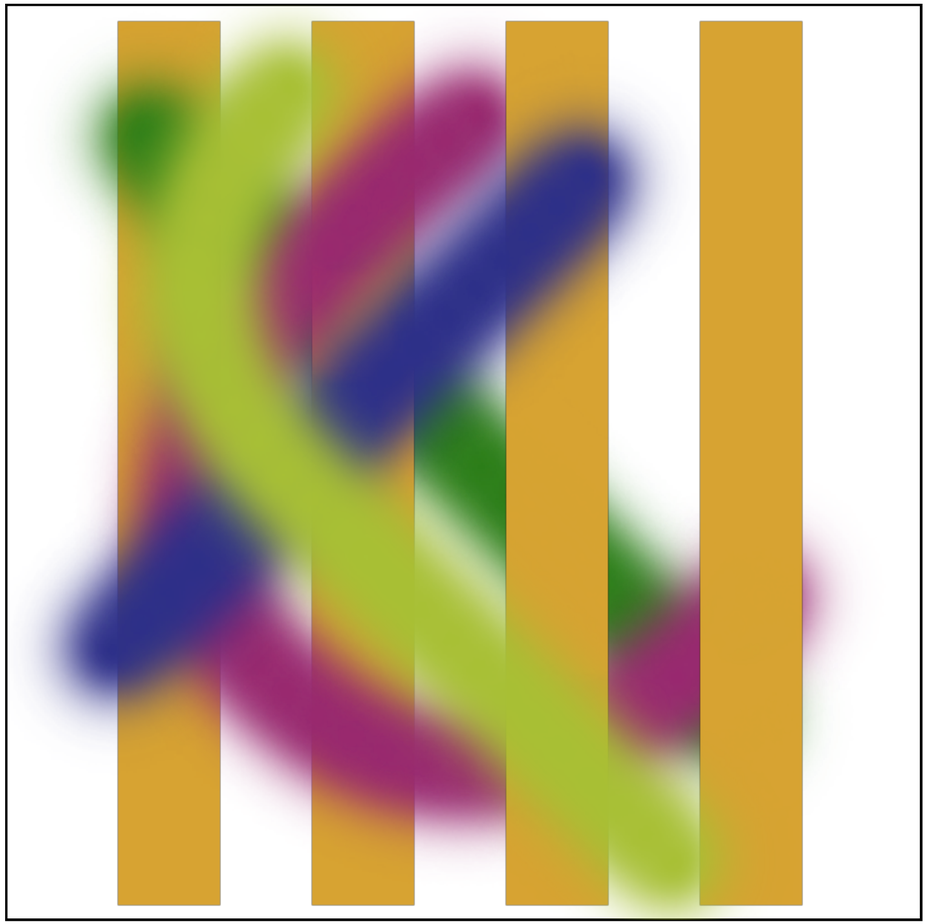
Summary

- The layers can be edited or converted to generalized barycentric coordinates



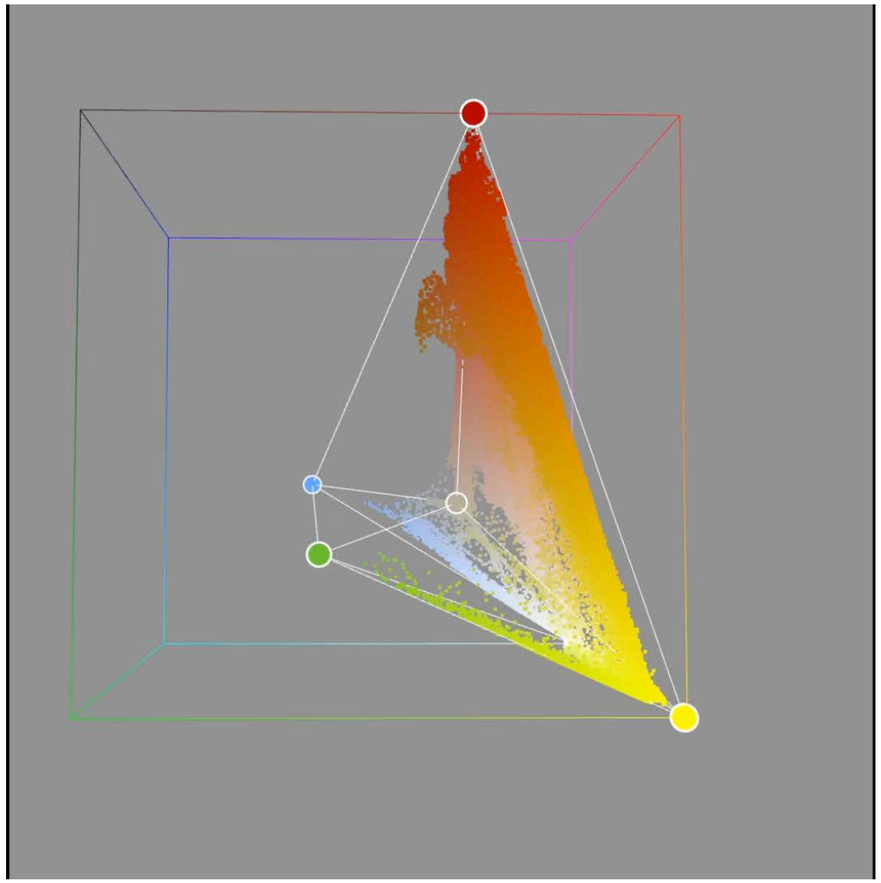
Limitation

- We use a global order for layers, which may not match true editing history.



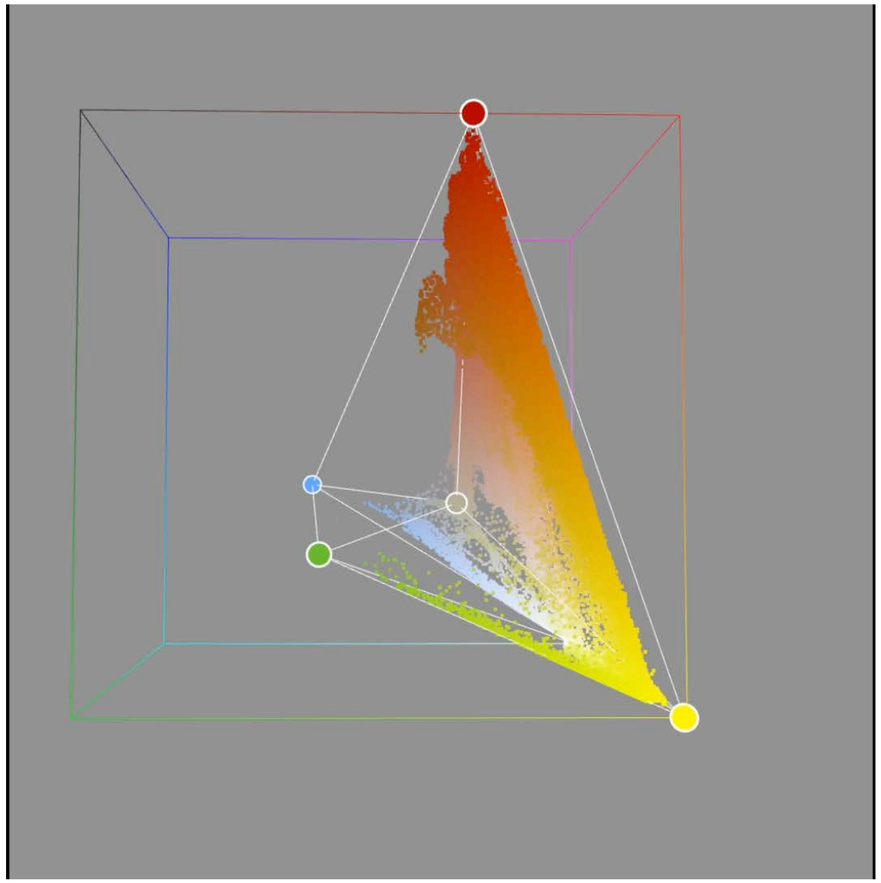
Limitation

- Pigment colors that lie within the convex hull cannot be detected.



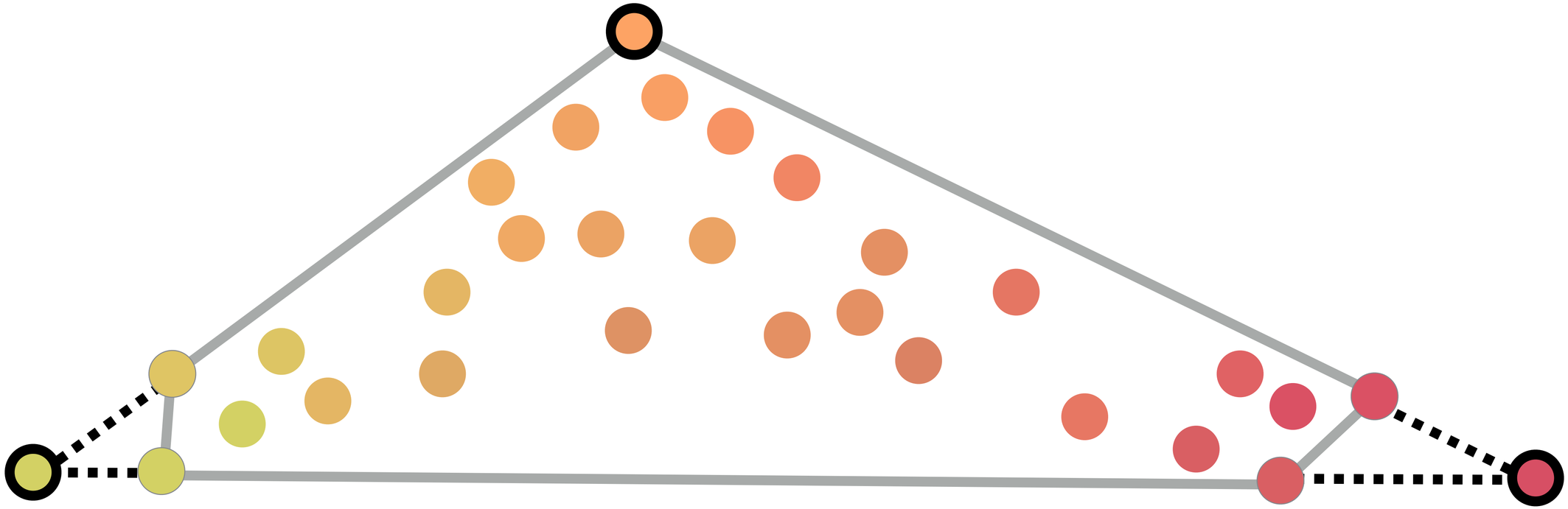
Limitation

- Pigment colors that lie within the convex hull cannot be detected.



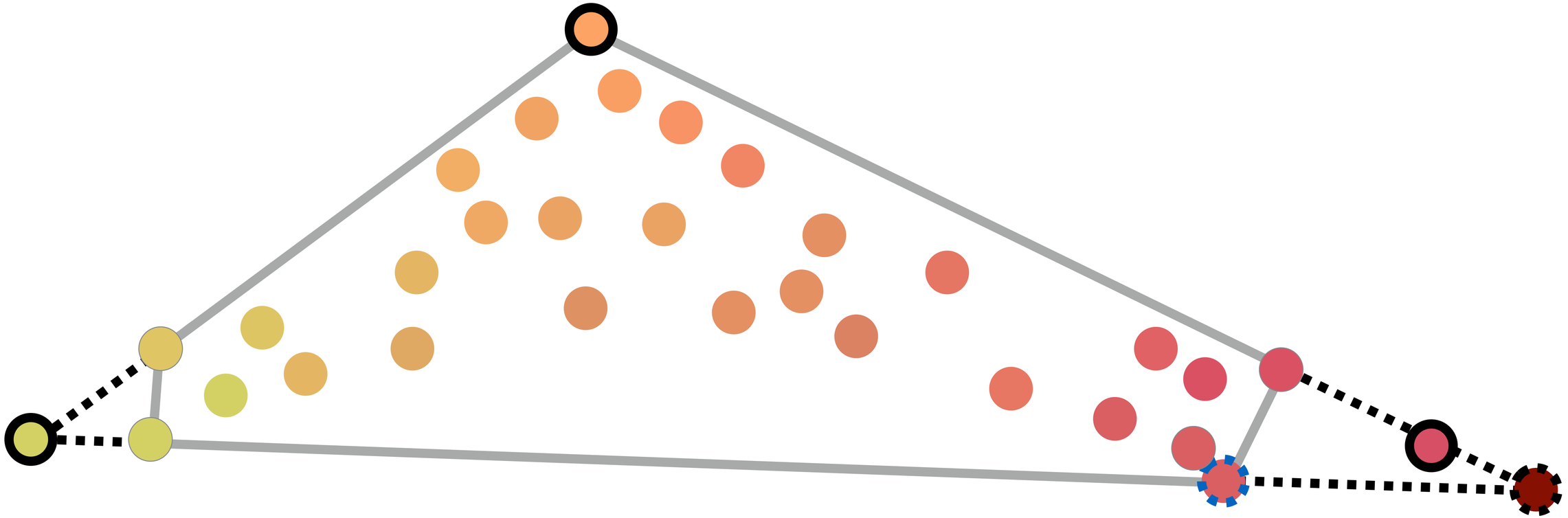
Limitation

- Outlier colors can influence the convex hull used in palette selection.



Limitation

- Outlier colors can influence the convex hull used in palette selection.



Future Work

Future Work

- Automatically determine palette size and layer order.

Future Work

- Automatically determine palette size and layer order.
- More semantic palette extraction.

Future Work

- Automatically determine palette size and layer order.
- More semantic palette extraction.
- Physically-inspired blending models (e.g. Kubelka-Munk).

Future Work

- Automatically determine palette size and layer order.
- More semantic palette extraction.
- Physically-inspired blending models (e.g. Kubelka-Munk).
- Additive mixing layers (works well, similar optimization but quadratic).

Future Work

- Automatically determine palette size and layer order.
- More semantic palette extraction.
- Physically-inspired blending models (e.g. Kubelka-Munk).
- Additive mixing layers (works well, similar optimization but quadratic).

$$\mathbf{E} = \|\text{original} - \text{reconstructed image}\|^2 \quad \sum ||P_i - w_{ij}C_j||^2$$

$$+$$

$$\text{Per pixel mixing weights sparsity} \quad \sum -(1 - w_{ij})^2$$

$$+$$

$$\text{Mixing weights spatial smoothness} \quad (\textbf{Laplacian})$$

Thank You!

- Contact Information
  - Jianchao Tan: jtan8@gmu.edu
  - Jyh-Ming Lien jmlien@gmu.edu
  - Yotam Gingold: ygingold@gmu.edu
- Project Website (GUI, code, data): https://cragl.cs.gmu.edu/singleimage/
- Artists: Adelle Chudleigh; Dani Jones; Karl Northfell; Michelle Lee; Adam Saltsman; Yotam Gingold.
- Sponsors:
  - United States National Science Foundation, Google.

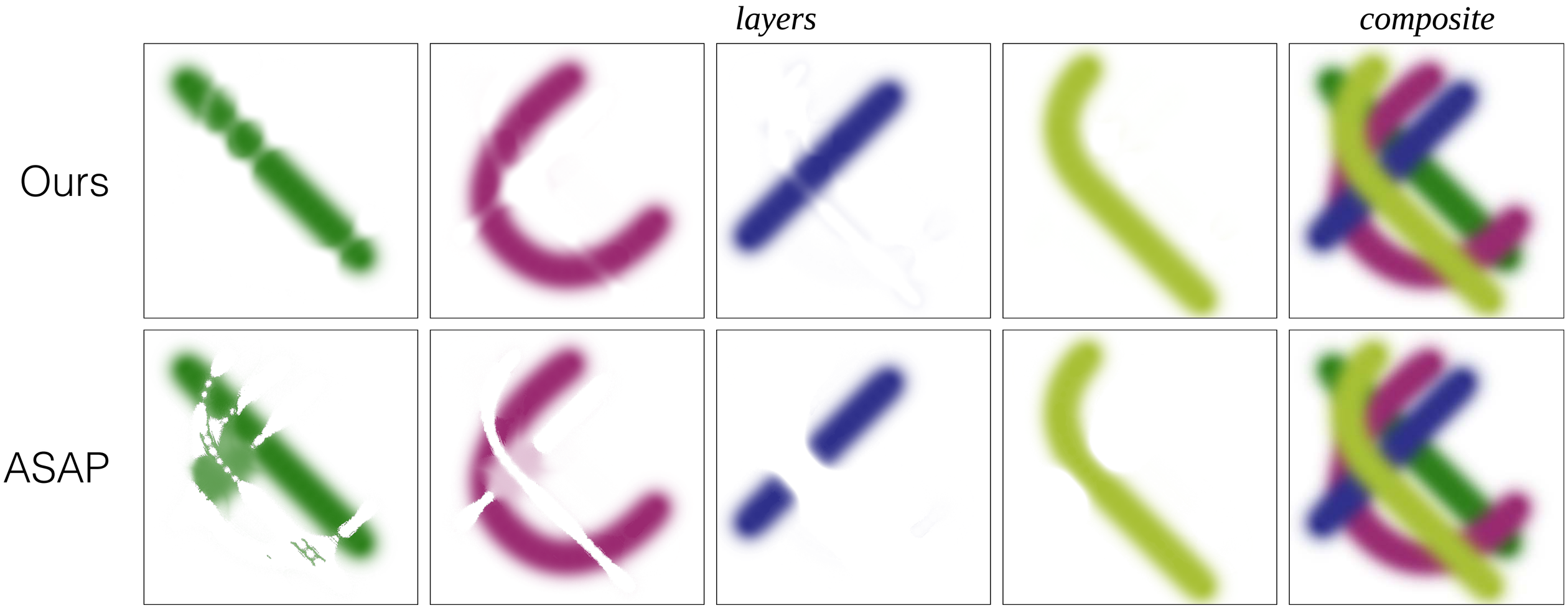
Extra Slides

As-sparse-as-possible(ASAP)

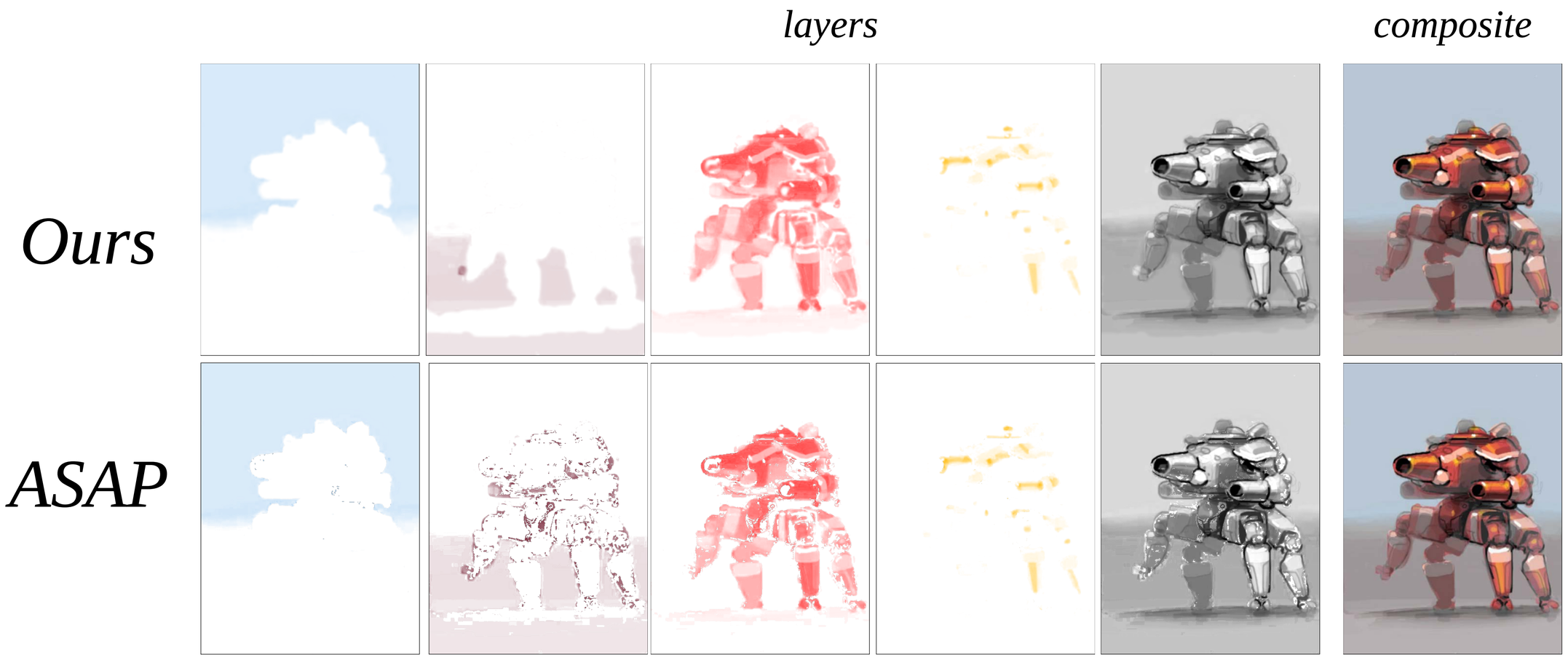
The sparsest possible weights have at most four non-zero weight values. In other words, a pixel can always be expressed as the weighted average of four colors. This corresponds to the well-defined, non-generalized barycentric coordinates of any tetrahedron enclosing color p whose vertices are a subset of the palette colors.

If the user has identified an background color as the bottom-most layer, the ASAP weights will assume that all pixels are the mixture of at most three non-background colors.

As-sparse-as-possible(ASAP)

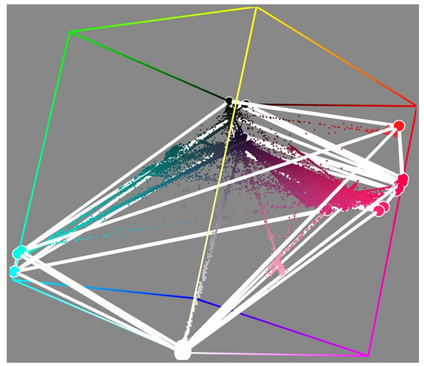


As-sparse-as-possible(ASAP)

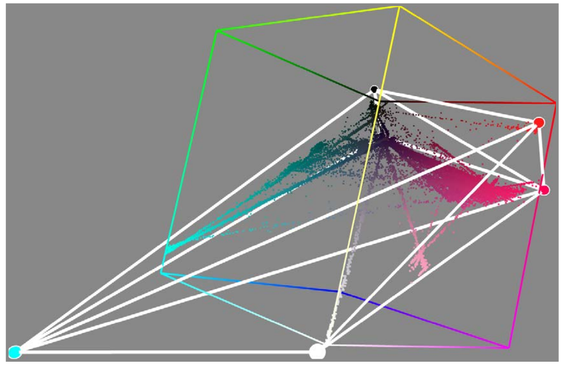


Post vertex brute force optimization led to an improvement in vertex positions.

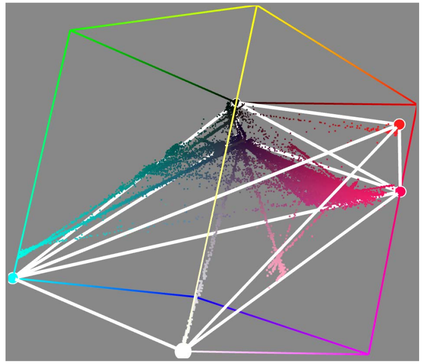
input image



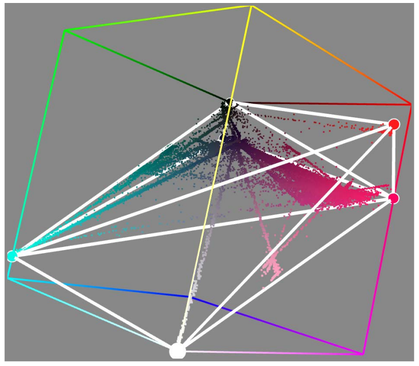
convex hull



simplified hull with invalid colors



projected hull



optimized hull

Spectral Matting [Levin et al. 2008]

input

9 components

1

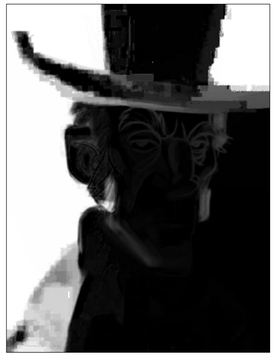
2

3

4

5

6



7

8

9

k-means

best alpha matte