Decomposing Images into Layers via RGB-space Geometry

Jianchao Tan **Jyh-Ming Lien** **Yotam Gingold**

George Mason University

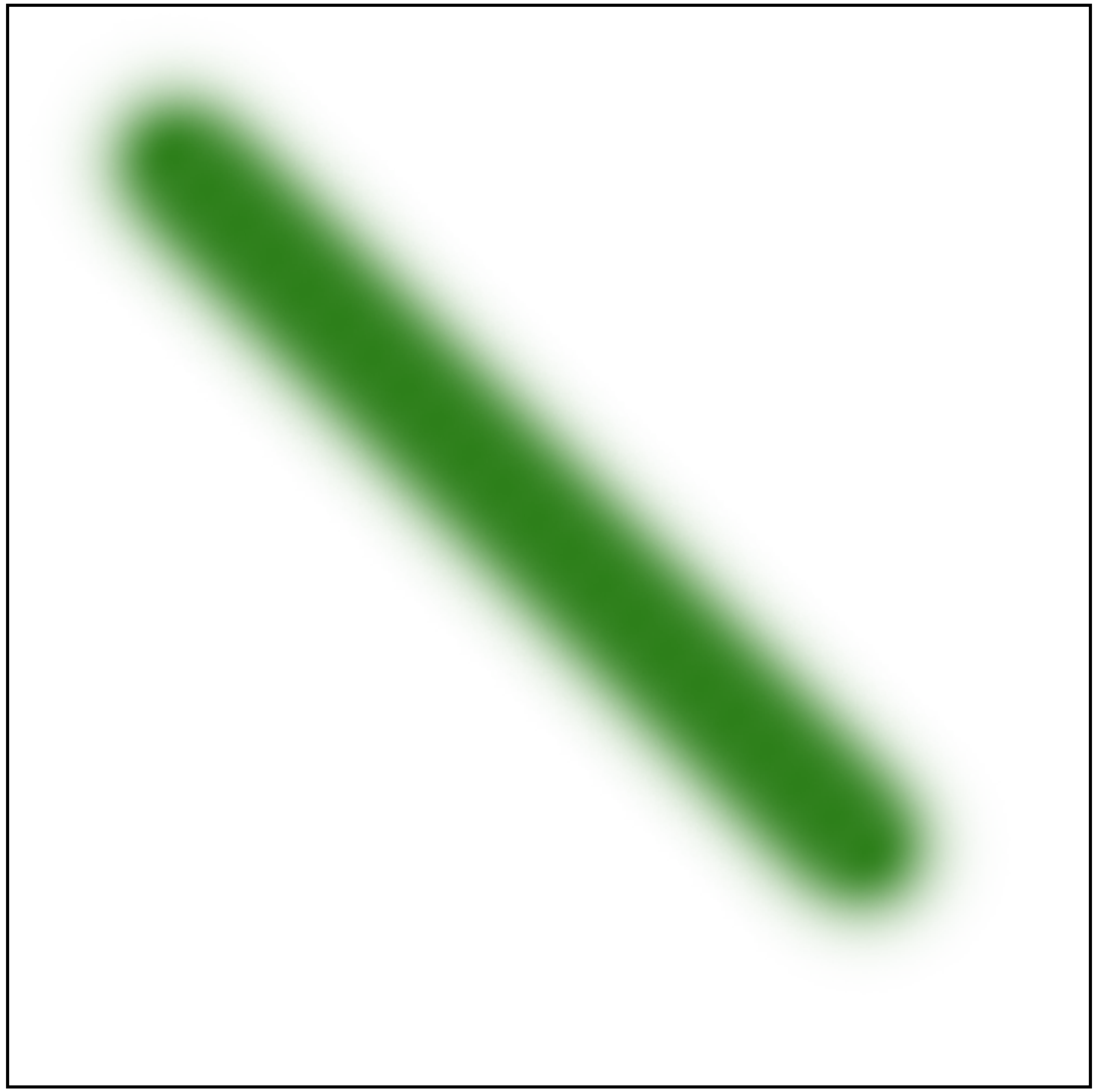
CraGL
Creativity and Graphics Lab

George Mason University

MASC
Motion and Shape Computing

Background: Digital Painting

Background: Digital Painting

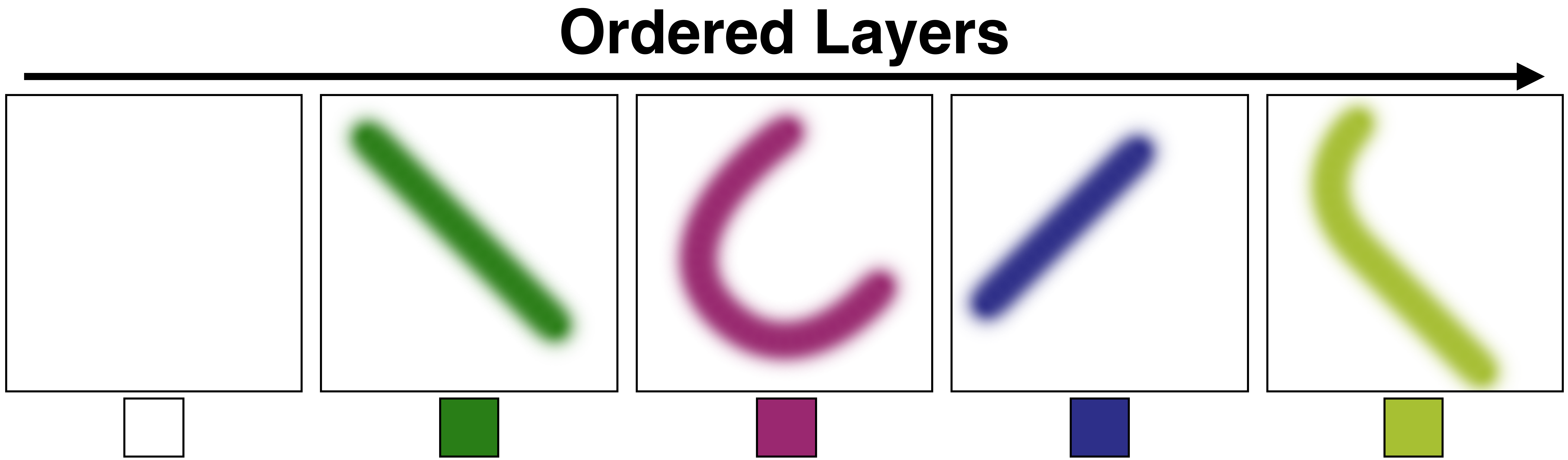


Background: Digital Painting

Background: Digital Painting

Background: Digital Painting

Background: Digital Painting

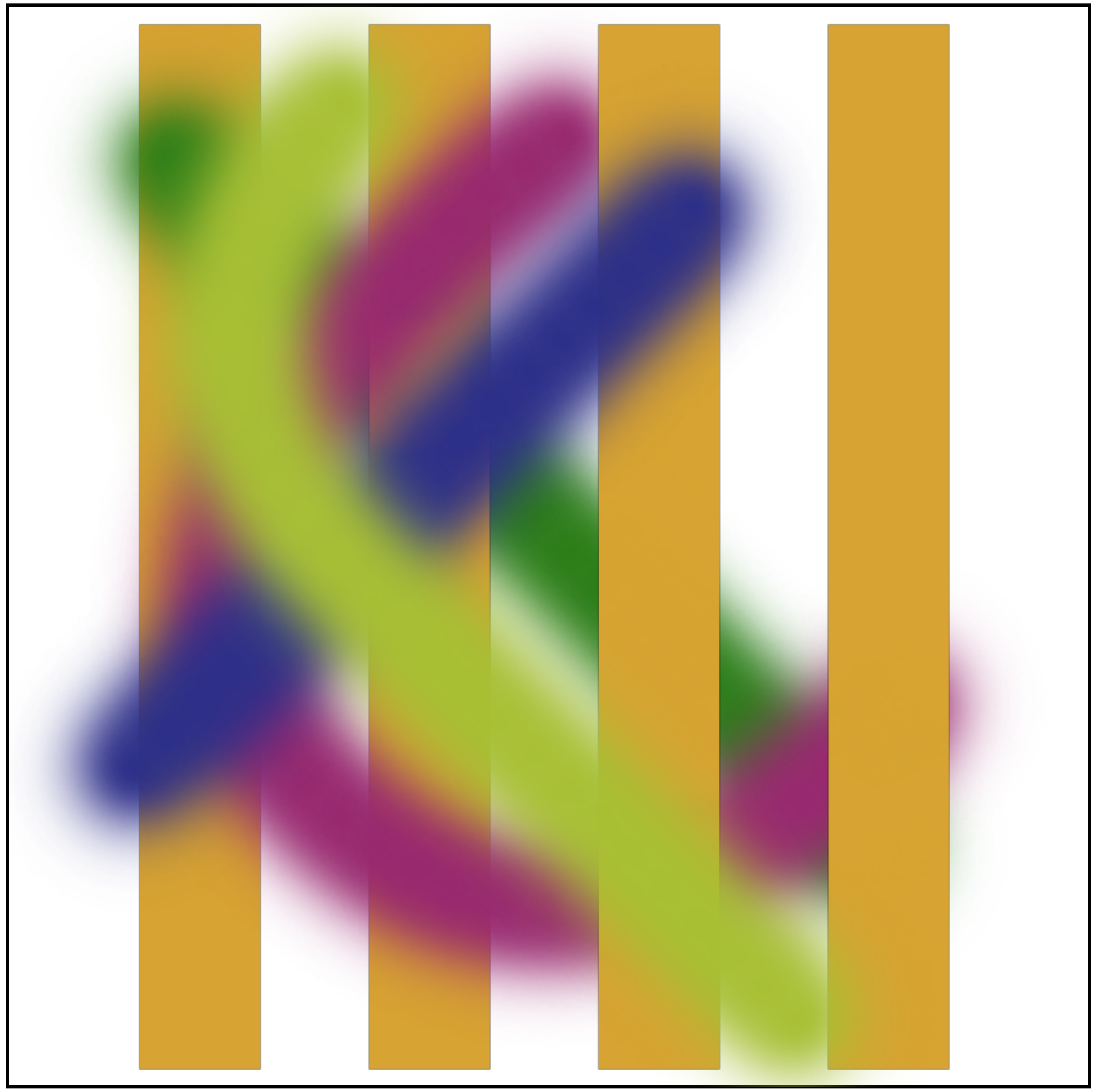


Background: Digital Painting

Motivation: Layers Organize Images

Motivation: Layers Organize Images

Motivation: Layers Organize Images



Images in the wild don’t have layers

Can we decompose them into layers?

That reproduce the original image?

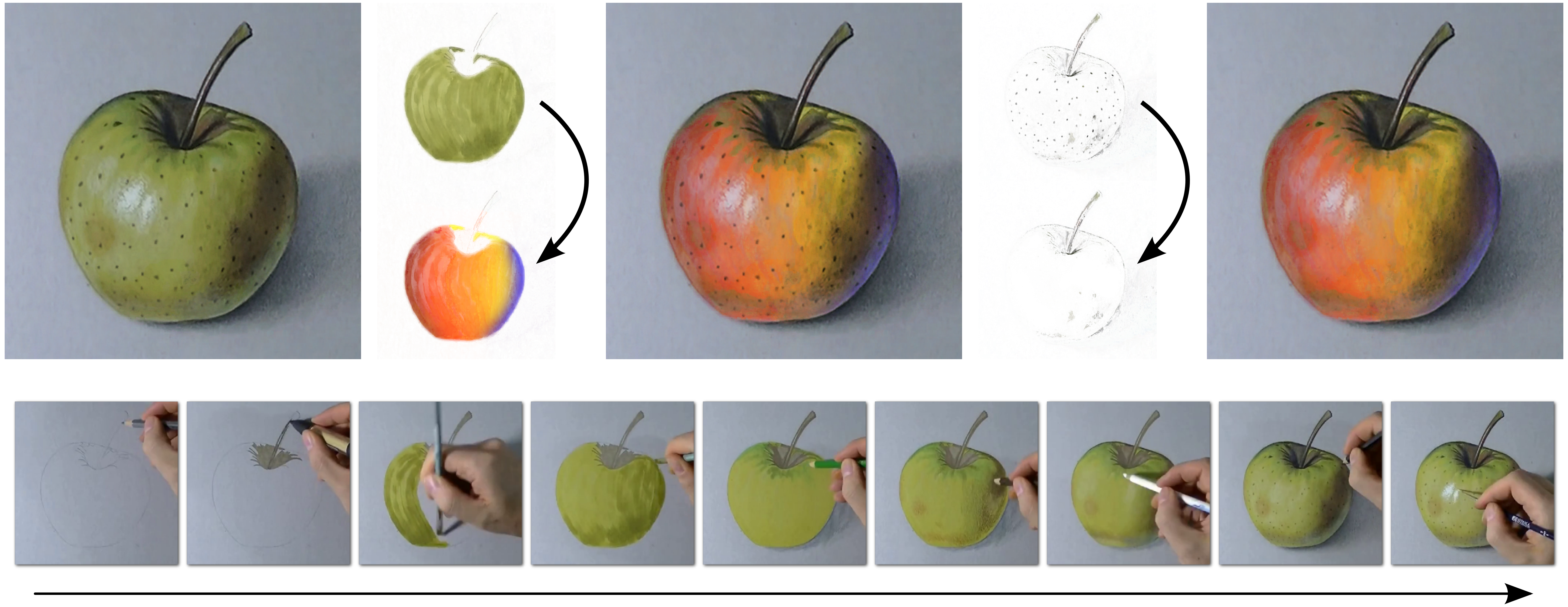
Two Subproblems

Layer Colors (Coats of Paint)

Layer Opacity

Related Work

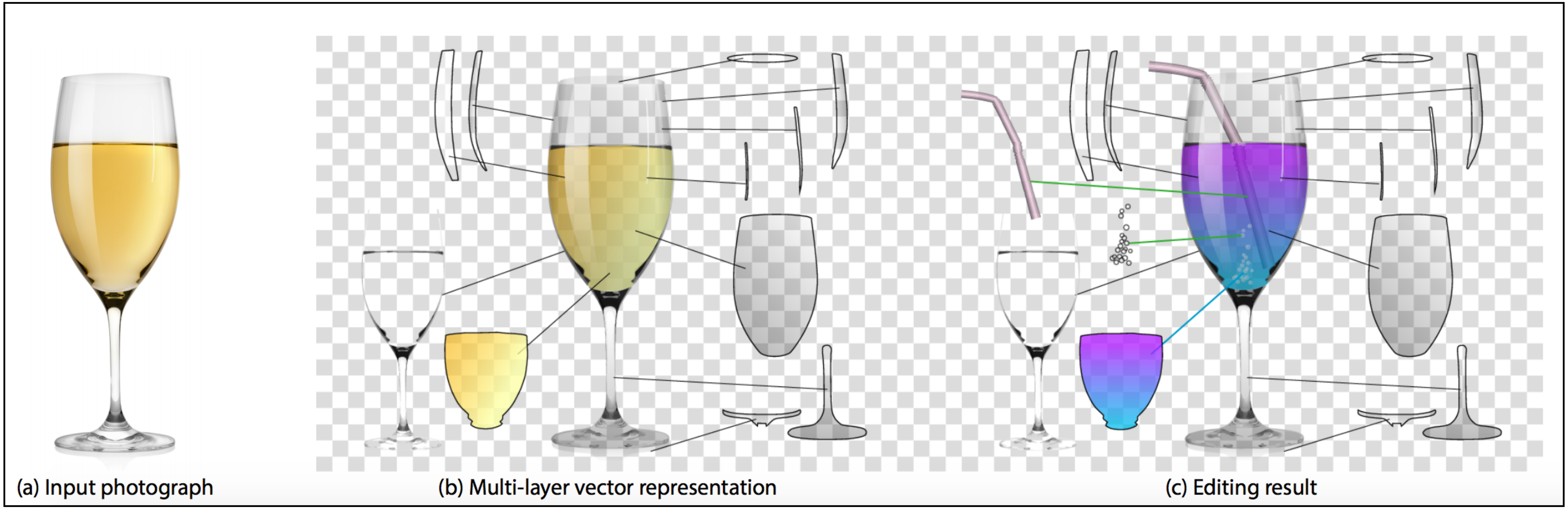
- Interacting with editing history
  - Su et al. [2009], VisTrails [2009], McCann and Pollard [2009; 2012], Grossman et al. [2010], Noris et al. [2012], Denning and Pellacini [2013] , Chen et al. [2014], Matzen and Snavely [2014], Karsch et al. [2014]. Amati and Brostow [2010], Hu et al. [2013]. Tan et al. [2015].



Decomposing time-lapse paintings into layers [Tan et al. 2015]

Related Work

- Decomposing edits
  - Xu et al. [2006], Amati and Brostow [2010], Fu et al. [2011], Hu et al. [2013], Richardt et al. [2014].



Vectorising bitmaps into semi-transparent gradient layers [Richardt et al. 2014]

Related Work

- Image matting
  - Smith and Blinn [1996], Zongker et al. [1999], Farid and Adelson [1999], Szeliski et al. [2000], Levin et al. [2006; 2008] and so on.

Blue screen matting [Smith and Blinn 1996]

Related Work

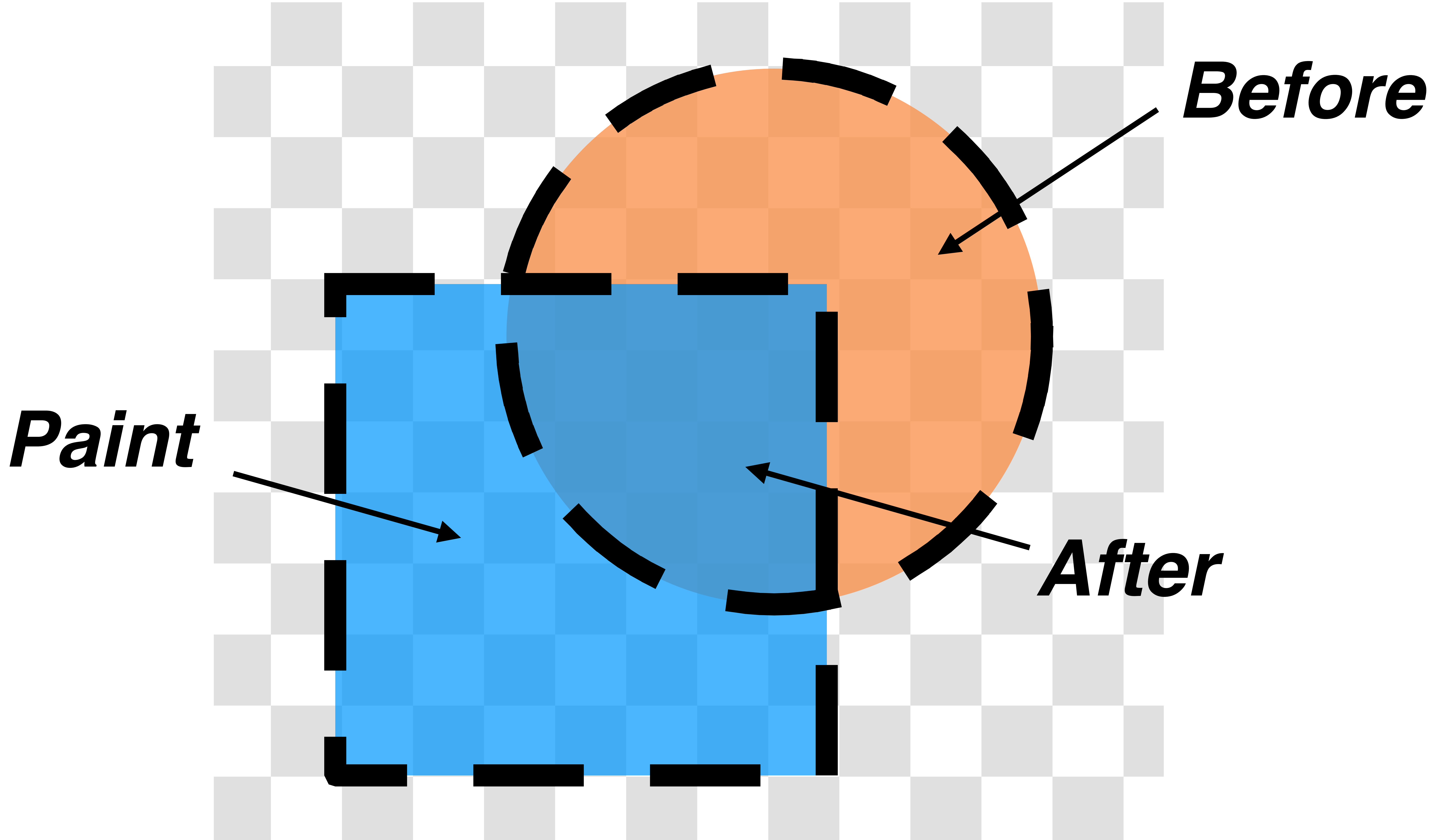
- Palette Selection
  - Shapira et al. [2009], O'Donovan et al. [2011], Lin et al. [2013], Gerstner et al. [2013], Chang et al. [2015].

Palette based photo recoloring [Chang et al. 2015]

Geometry of Compositing

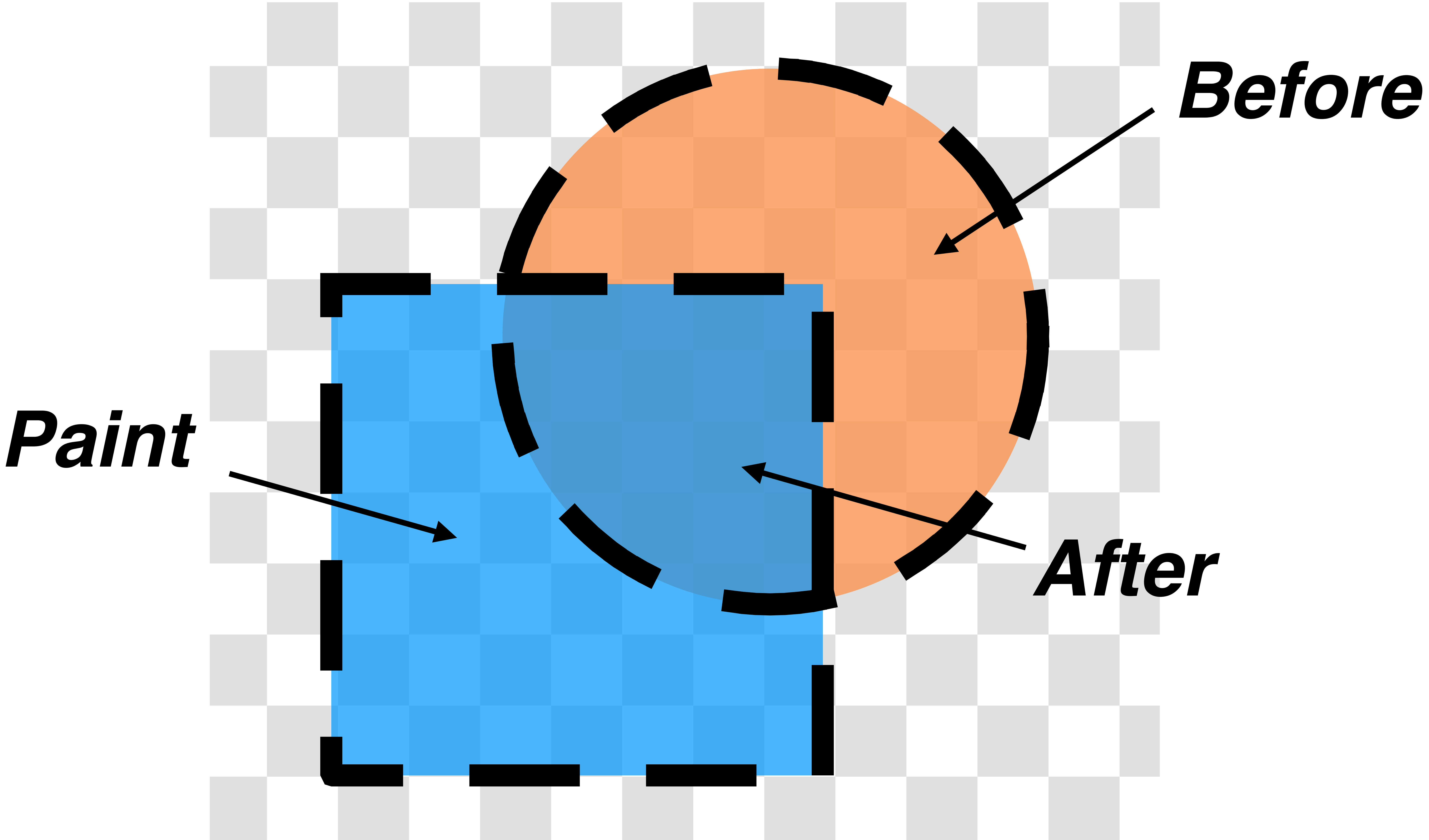
Porter-Duff "Over" Color Compositing

$$After = Before \cdot (1 - \alpha) + Paint \cdot \alpha$$

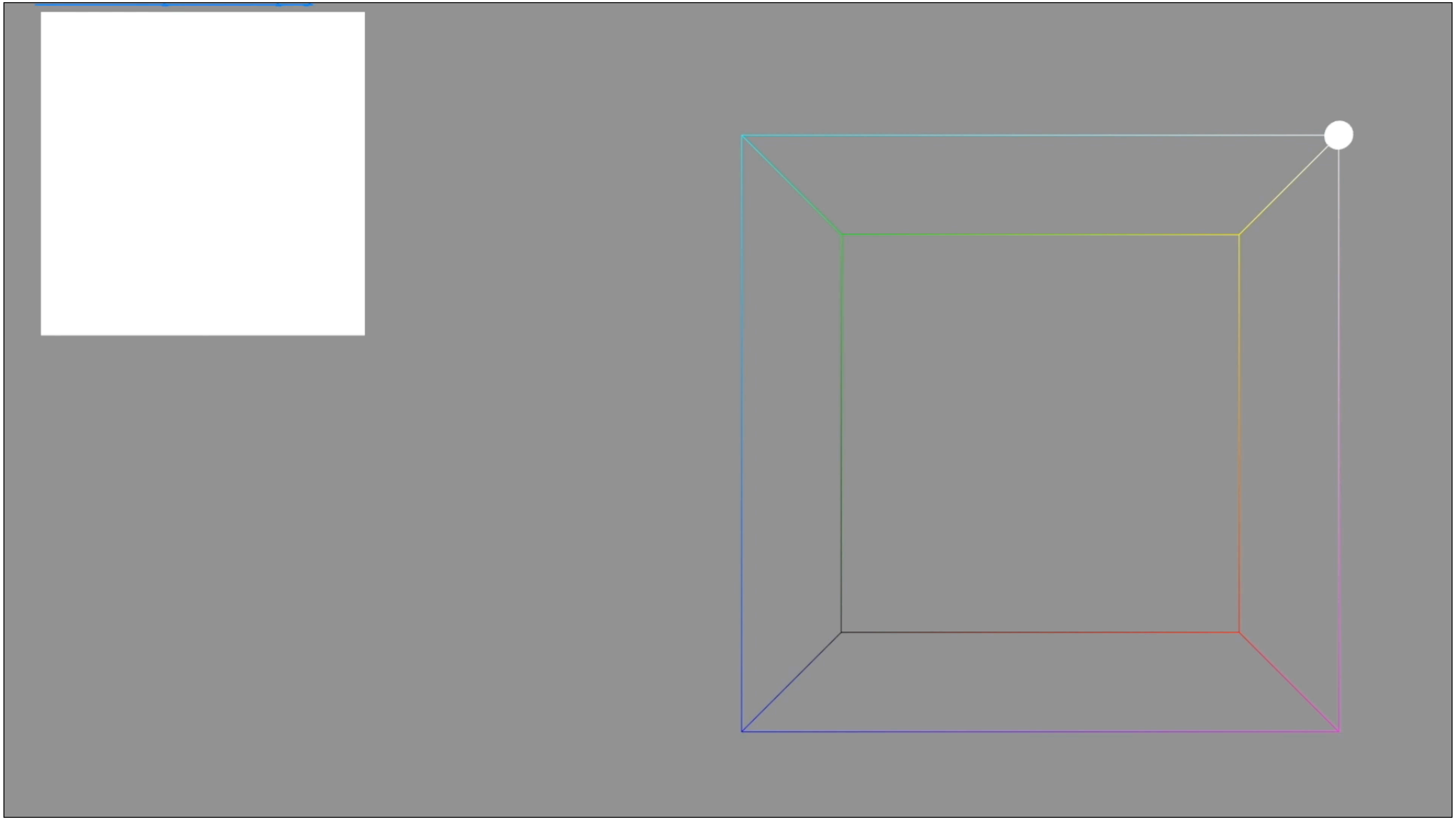


Porter-Duff "Over" Color Compositing

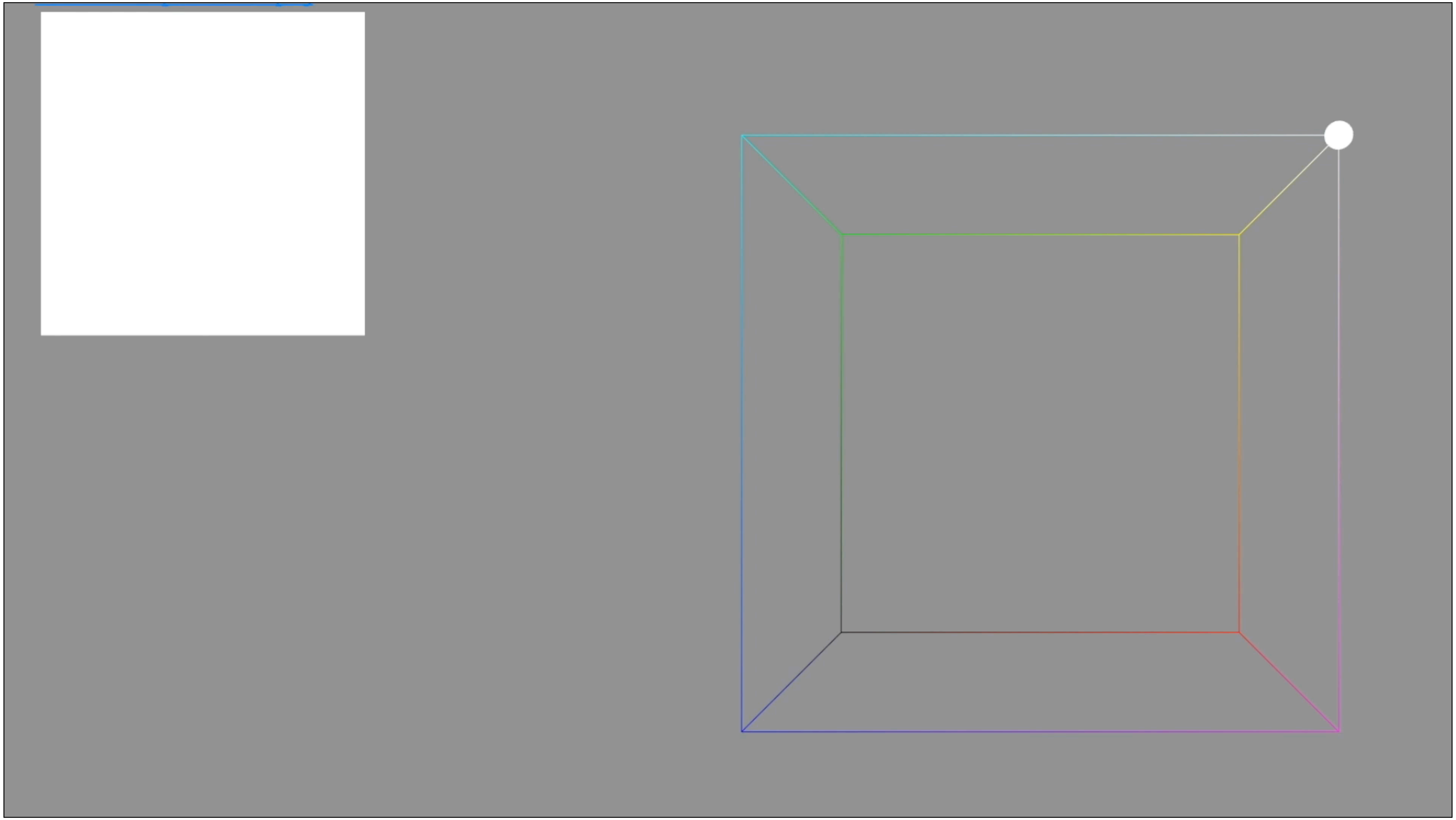
$$After = Before \cdot (1 - \boxed{\alpha}) + Paint \cdot \boxed{\alpha}$$



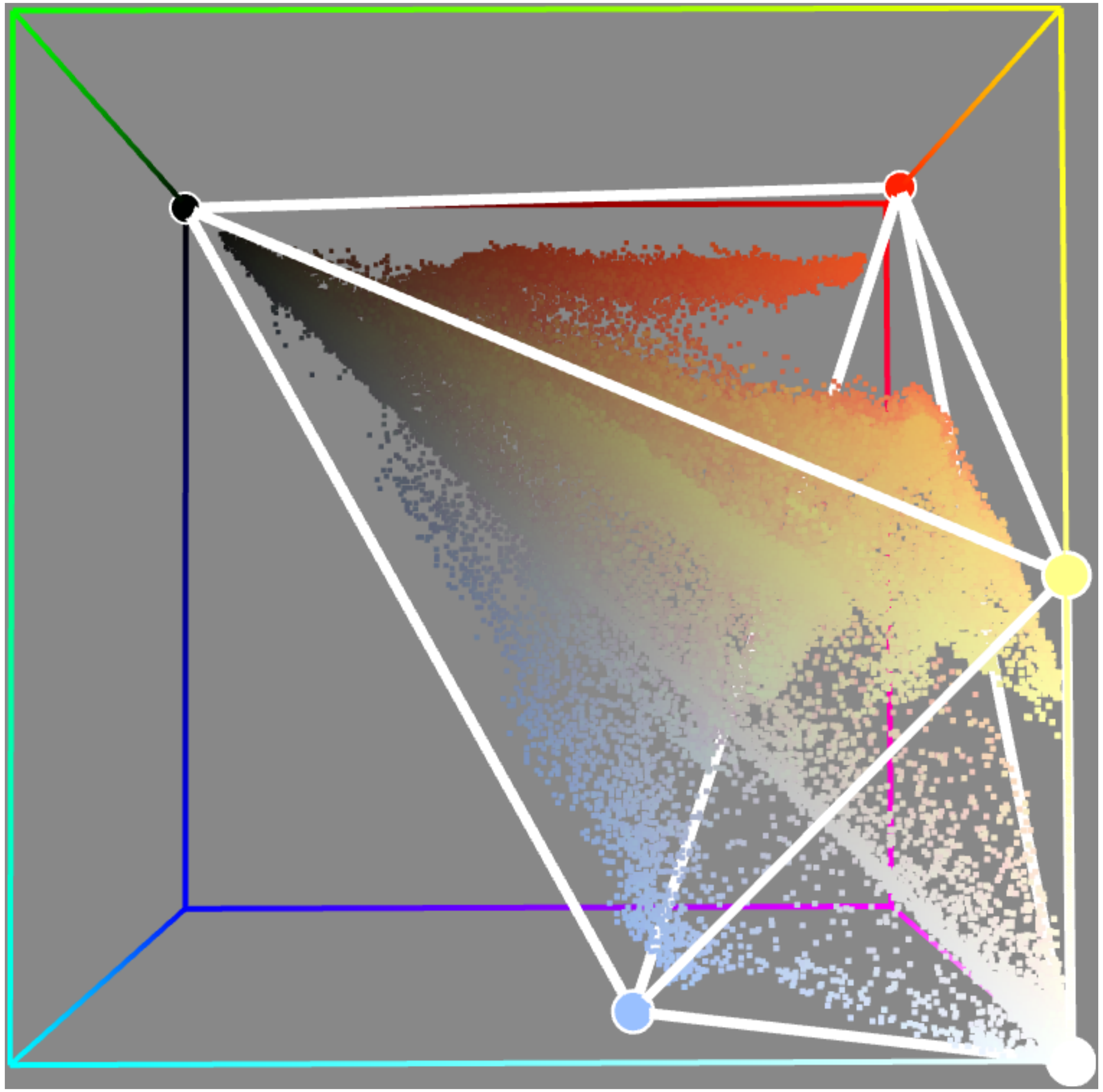
Coats of Paint follow a convex structure in RGB space



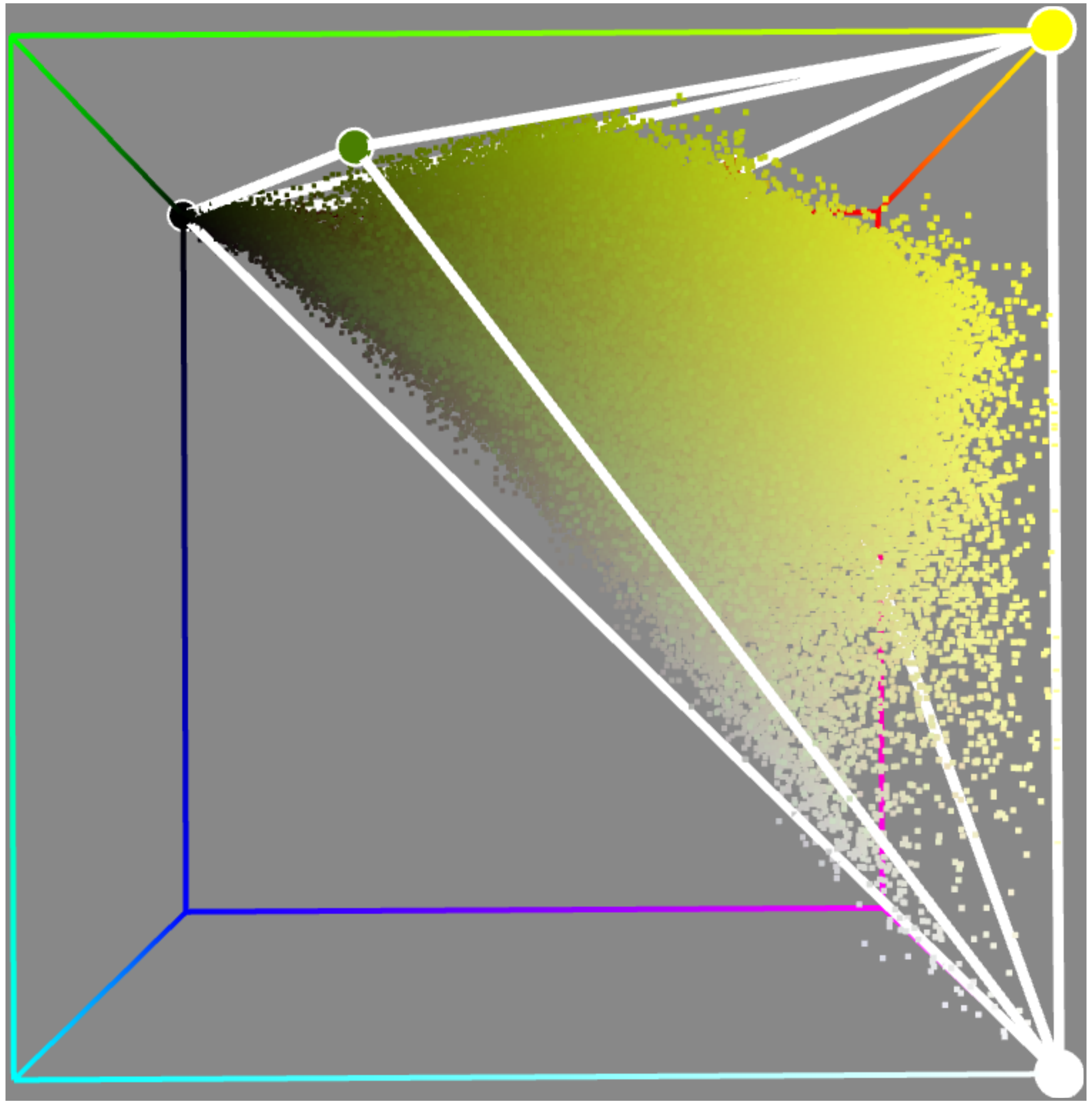
Coats of Paint follow a convex structure in RGB space



More examples

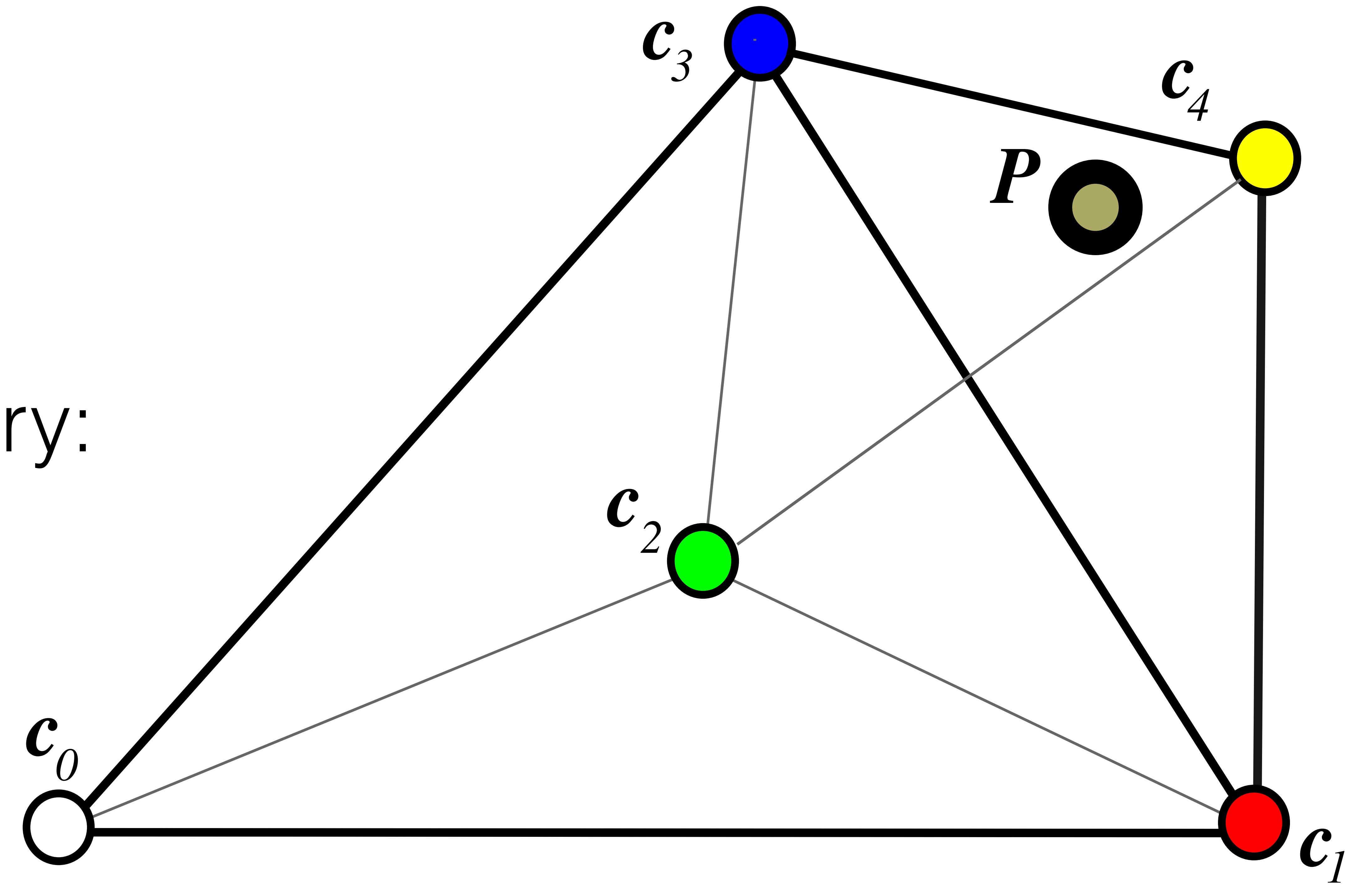


More examples



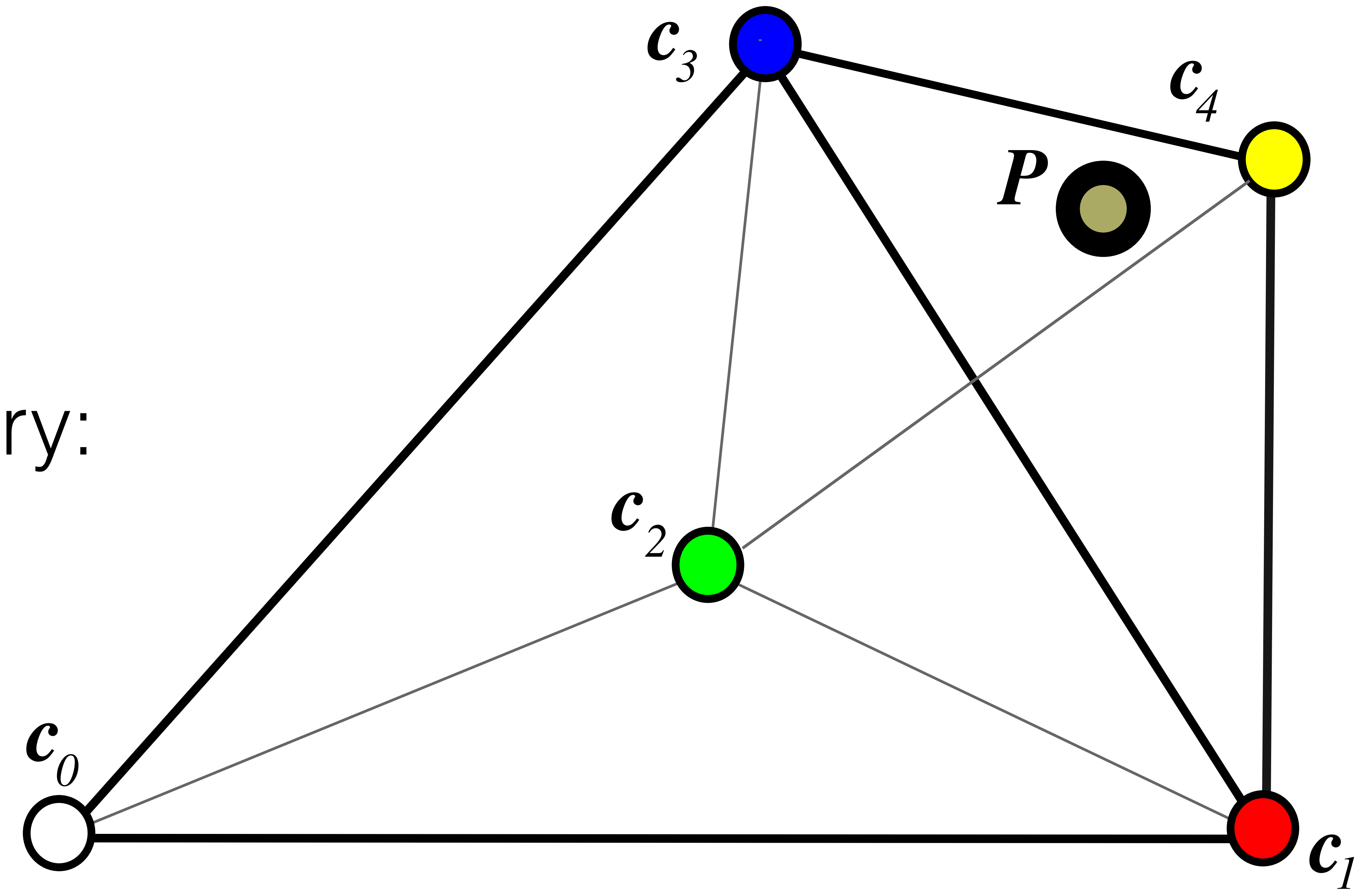
Geometric interpretation of 'over' compositing equation

Geometry:



Geometric interpretation of 'over' compositing equation

Geometry:

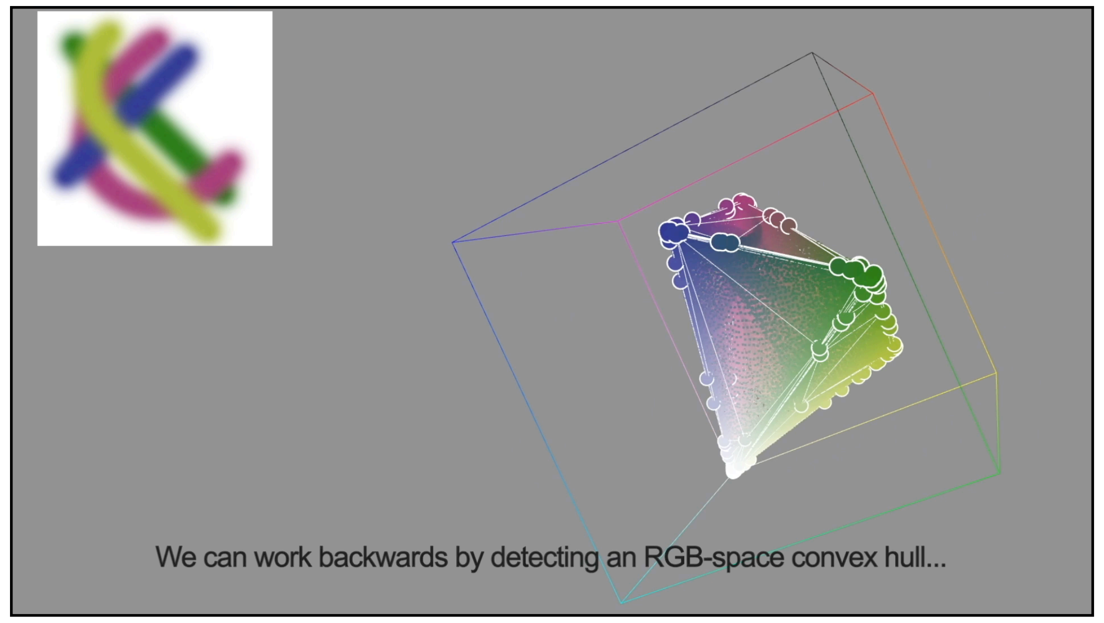


Algebra:

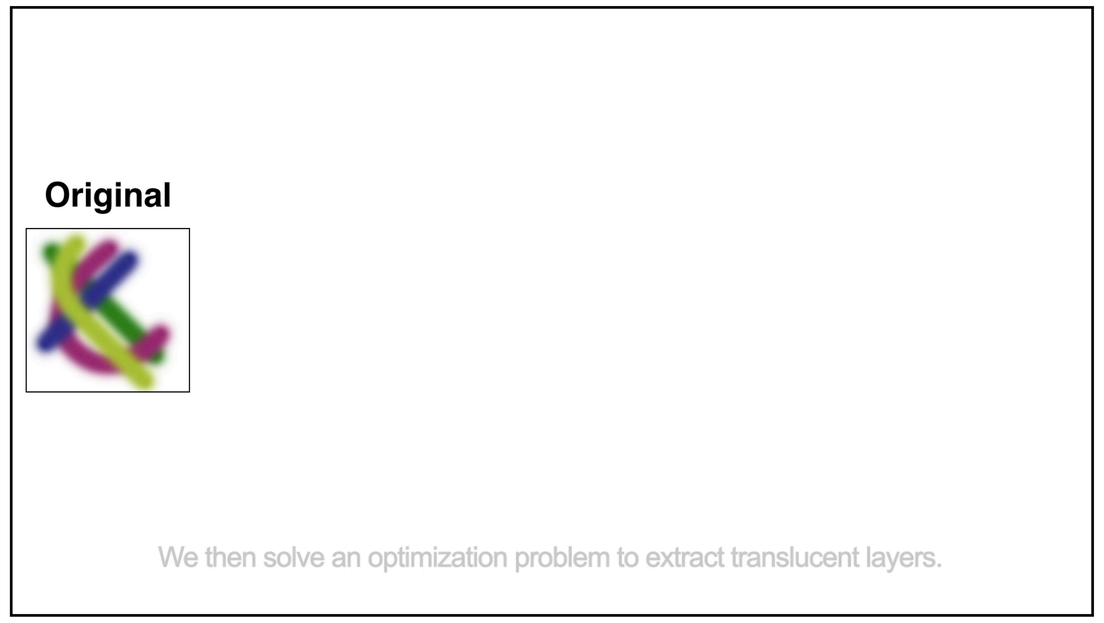
$$\mathbf{p} = \mathbf{c}_n + \sum_{i=1}^{n} \left[(\mathbf{c}_{i-1} - \mathbf{c}_i) \prod_{j=i}^{n} (1 - \alpha_j) \right]$$

Our Pipeline

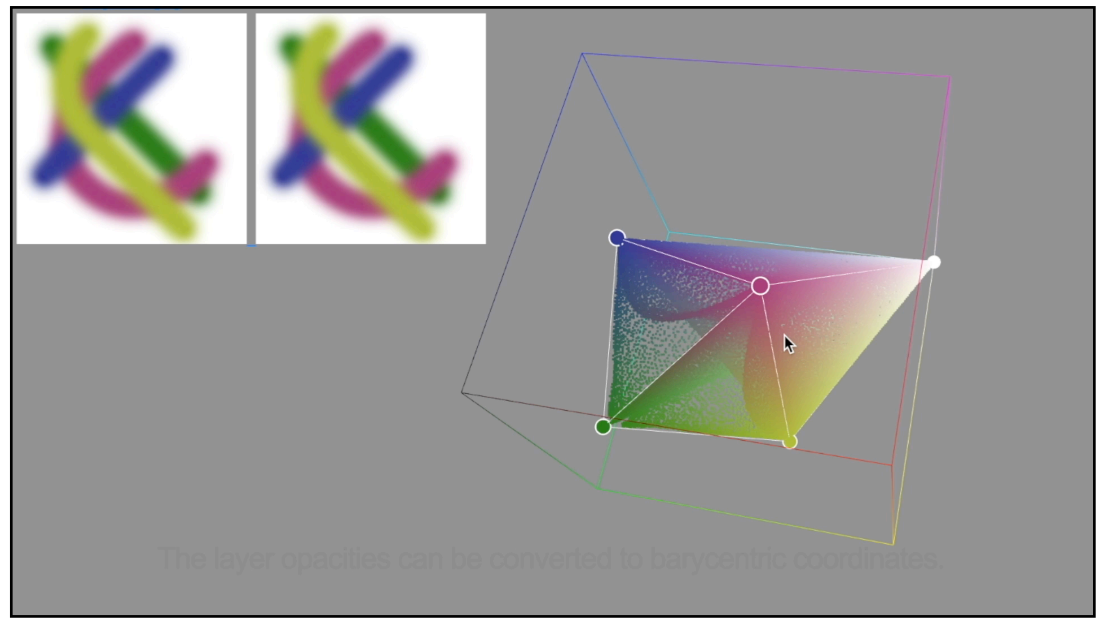
Input



Palette selection

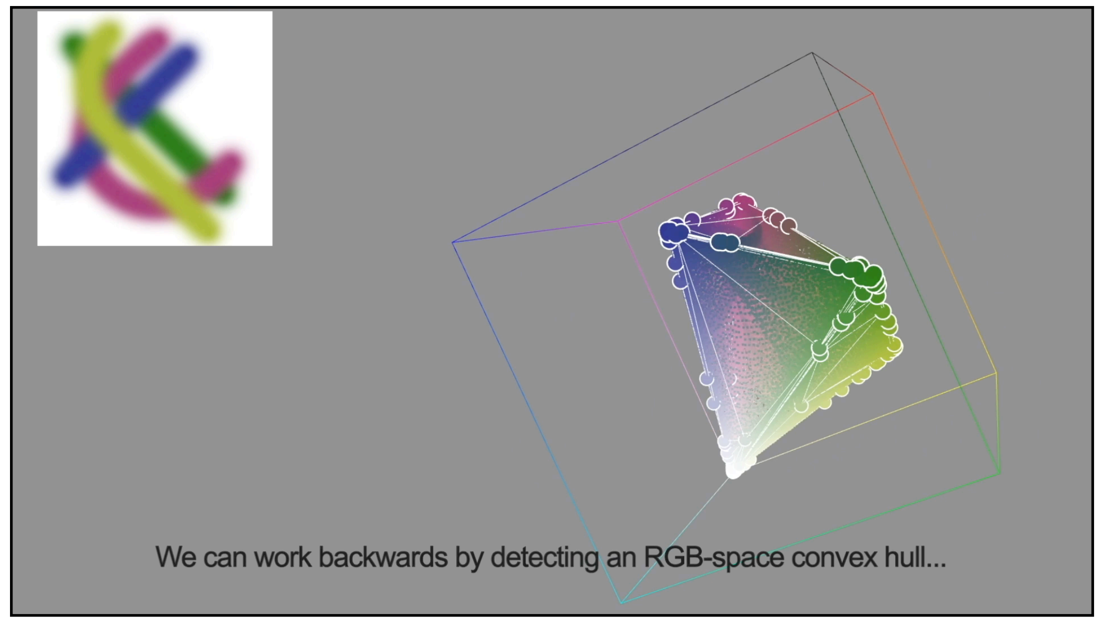


Layer opacity

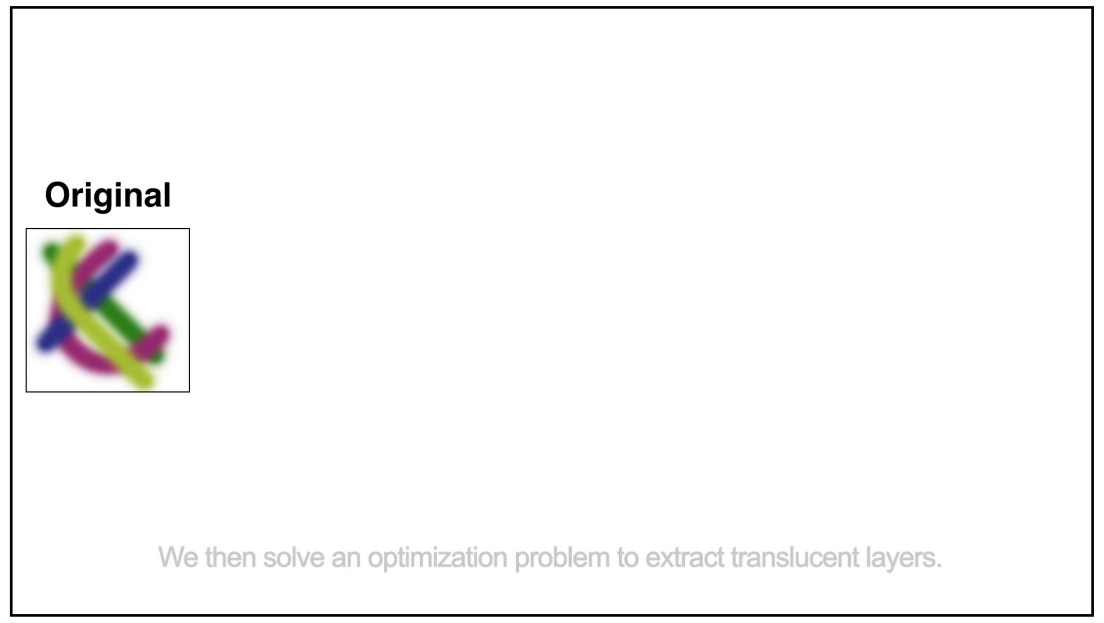


Edit

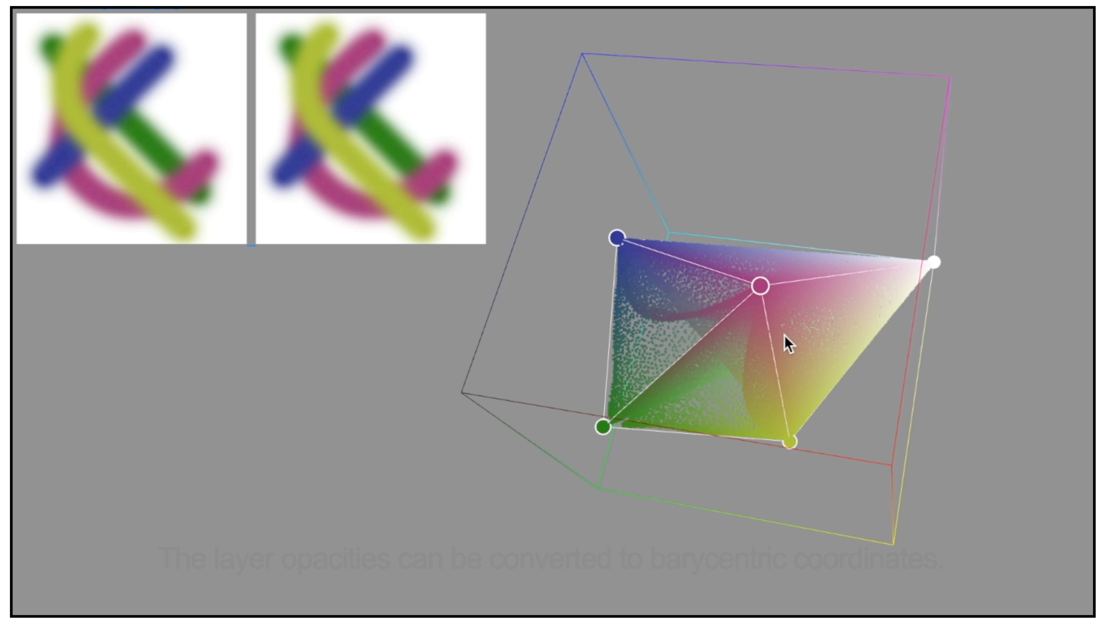
Input



Palette selection

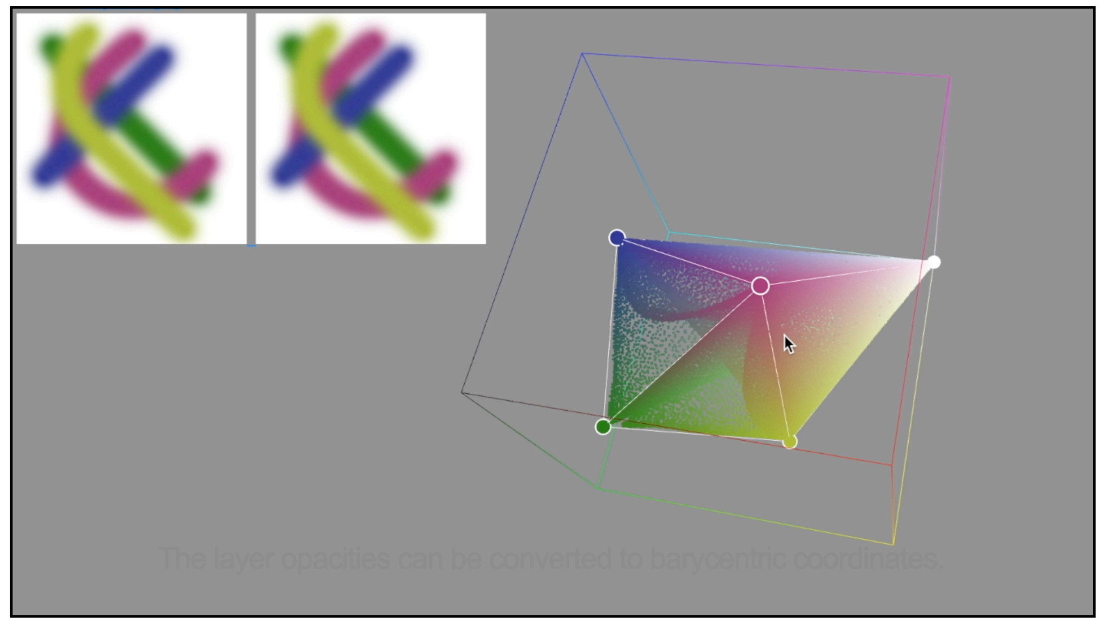


Layer opacity

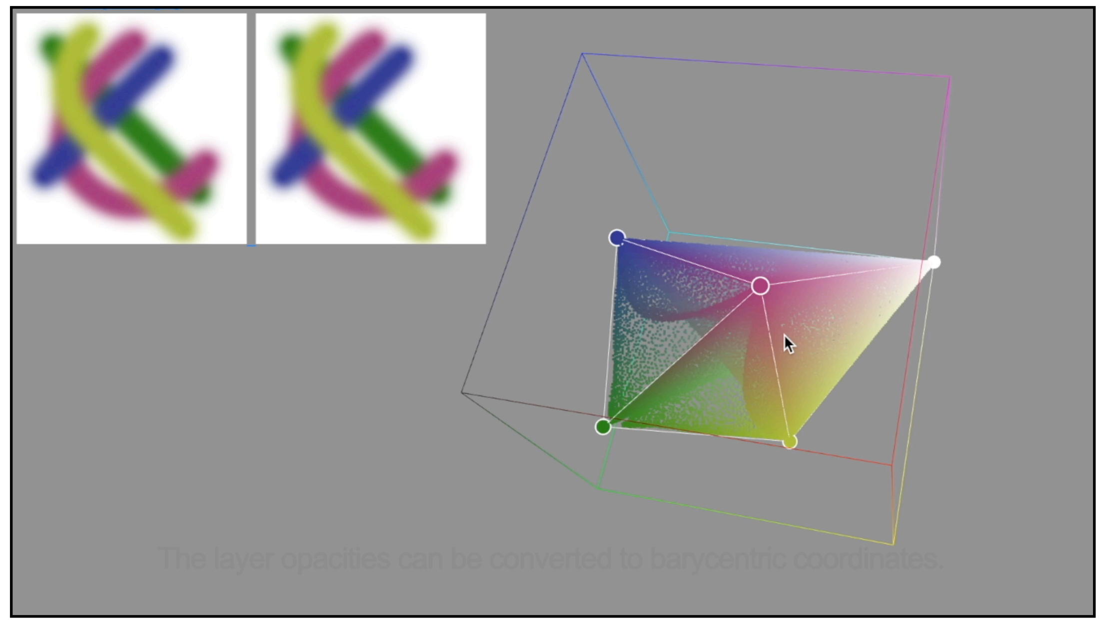


Edit

Input

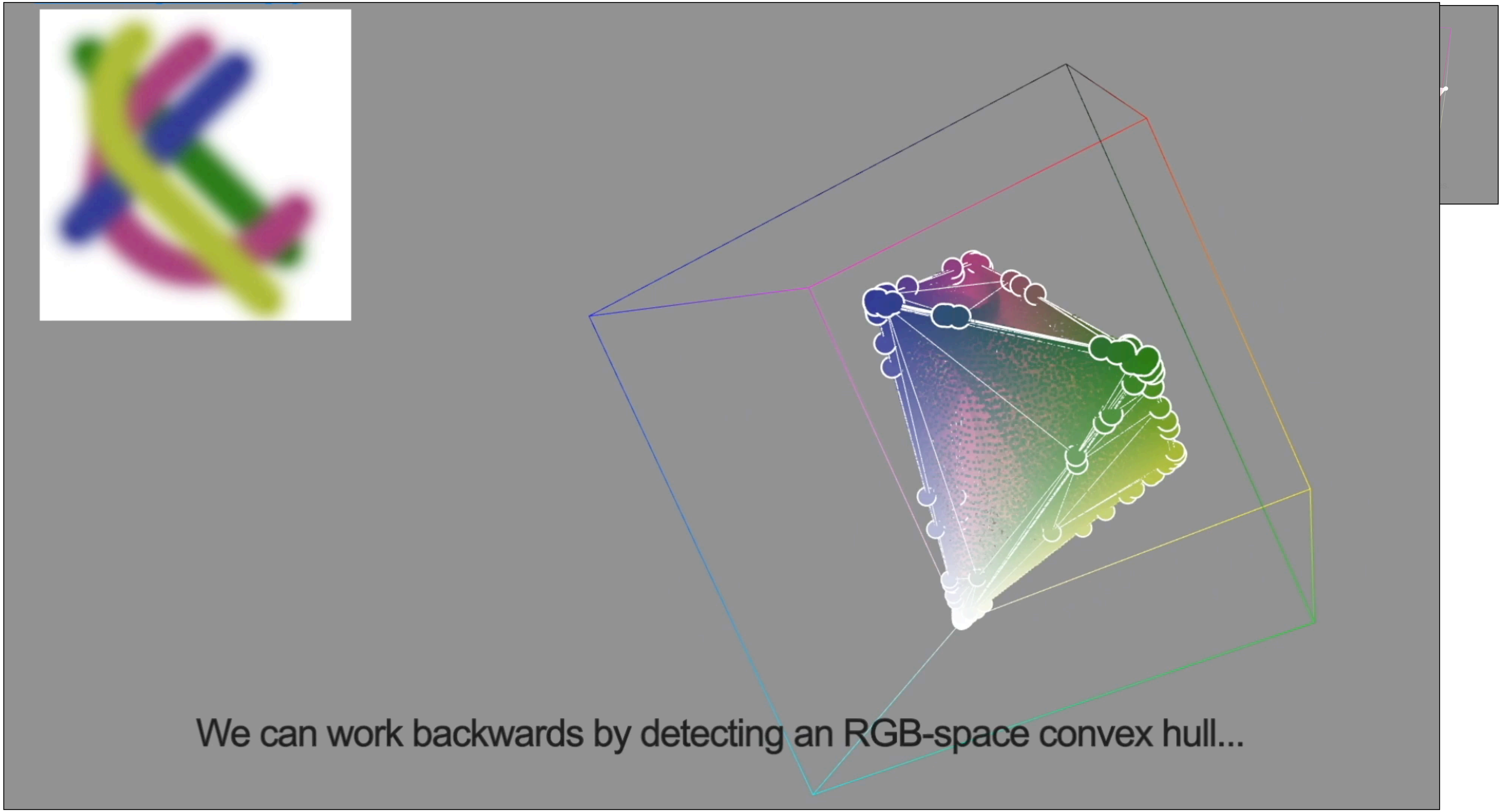


Edit

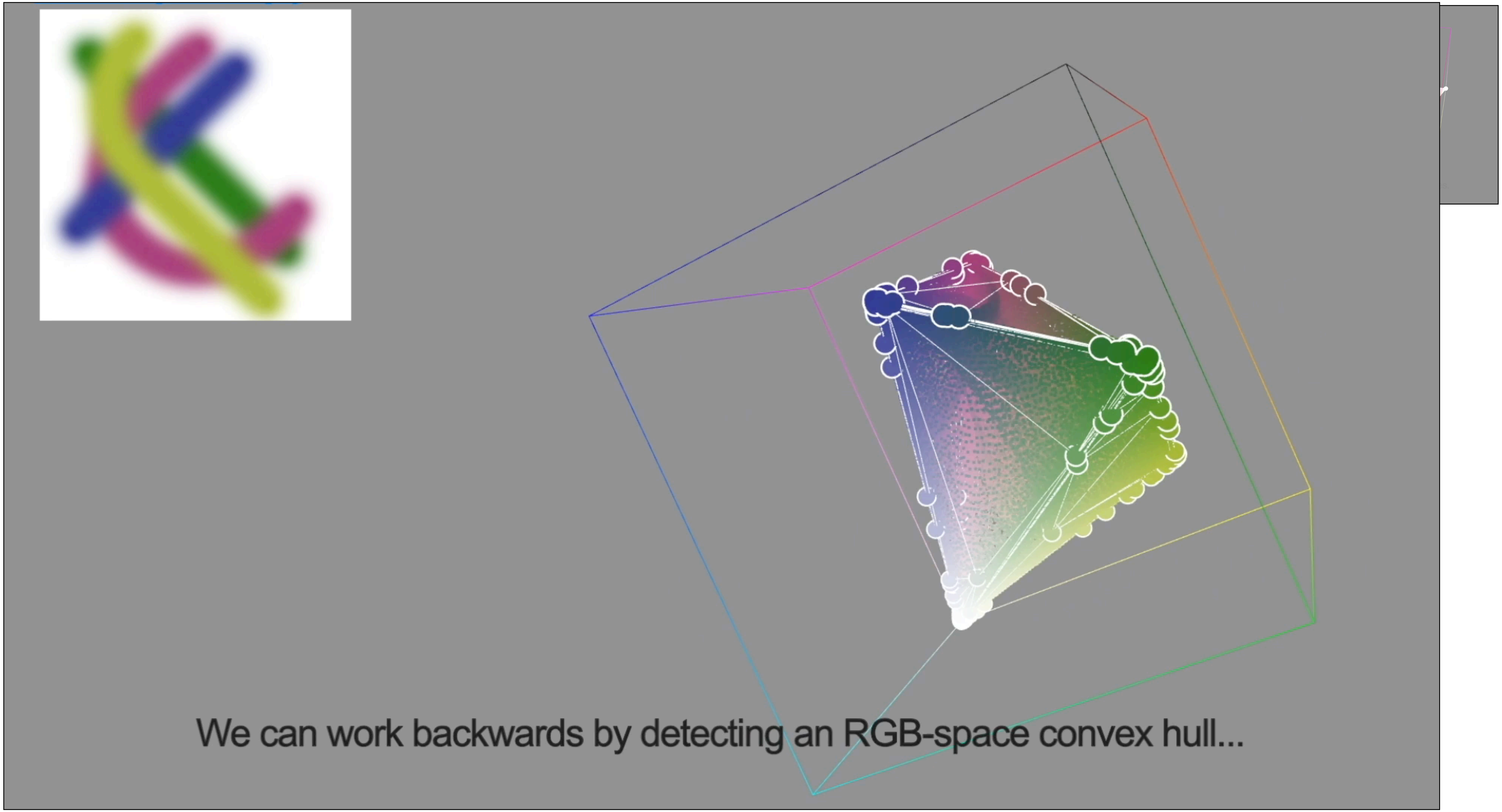


Edit

Input

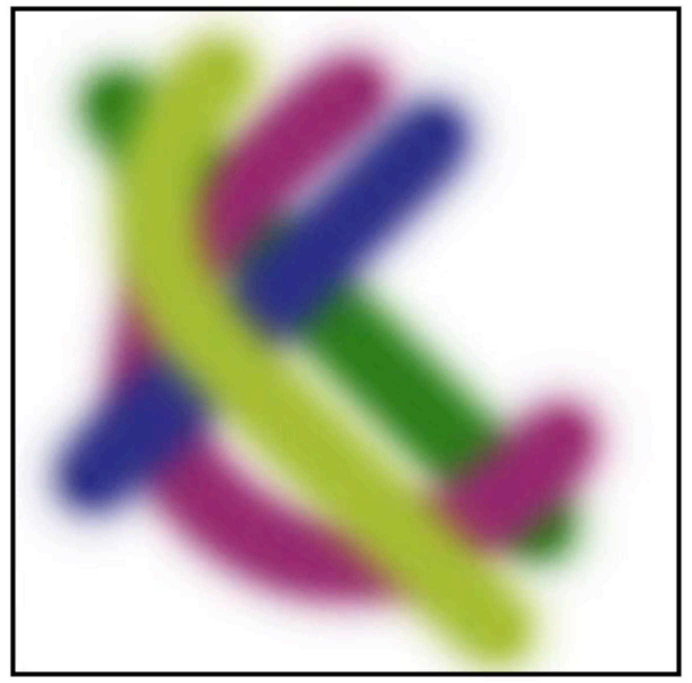


Palette selection



Palette selection

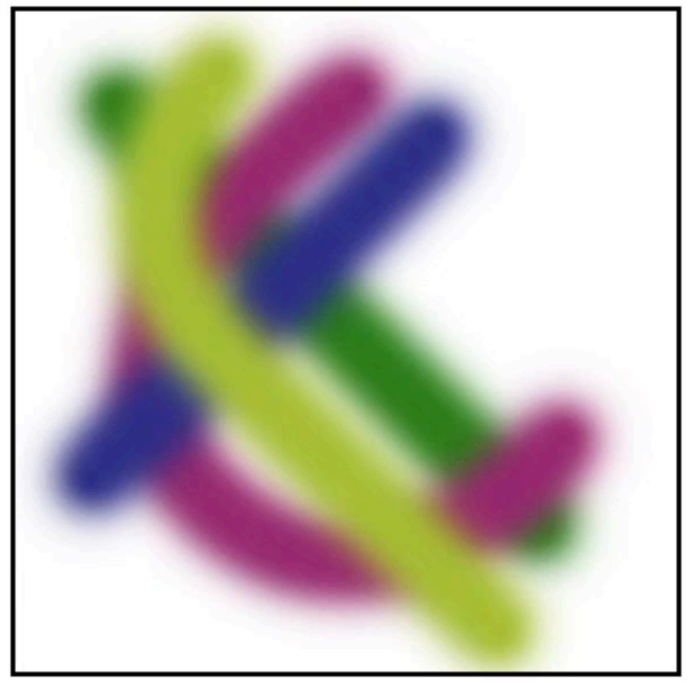
Original



We then solve an optimization problem to extract translucent layers.

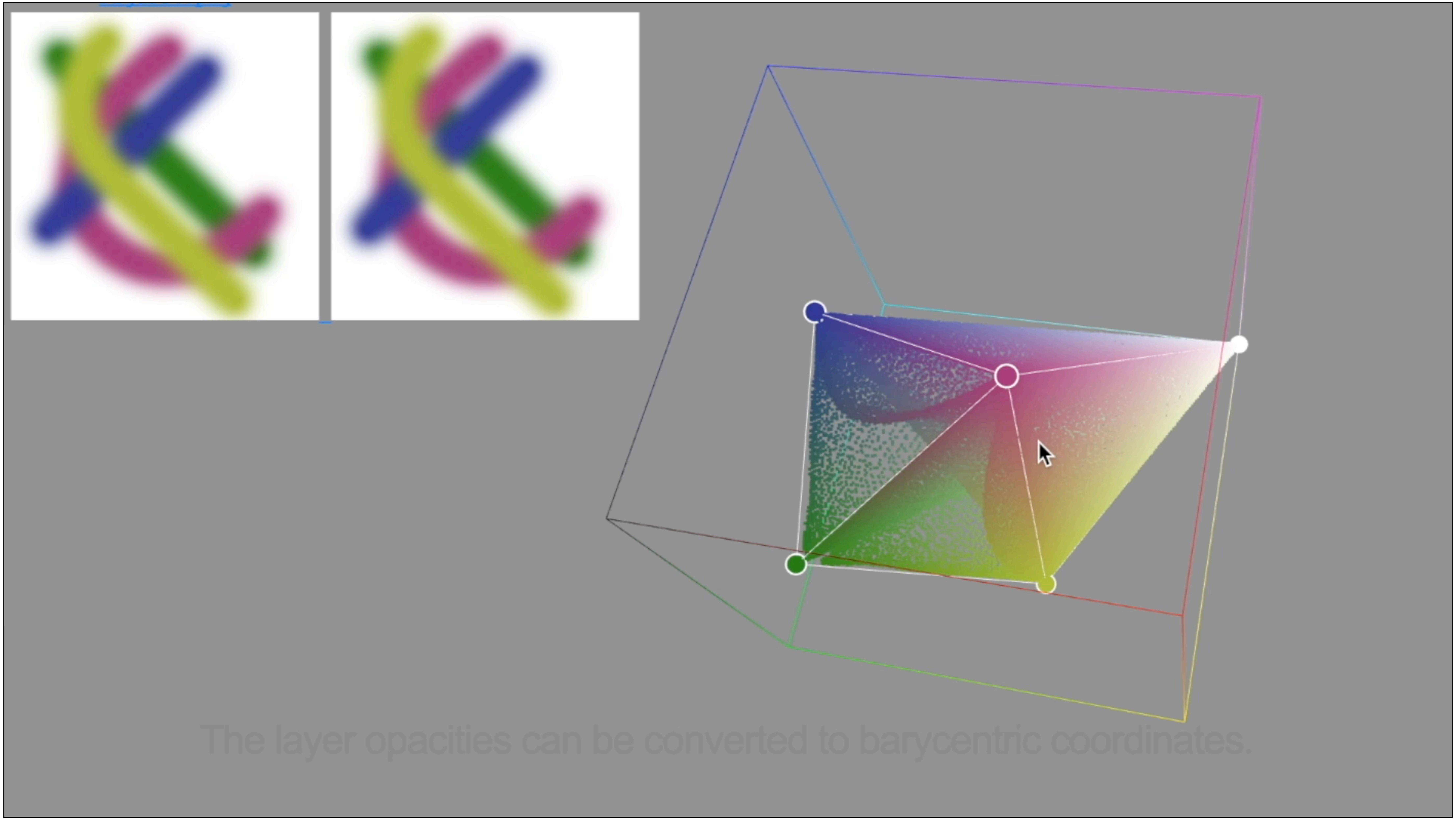
Layer opacity

Original

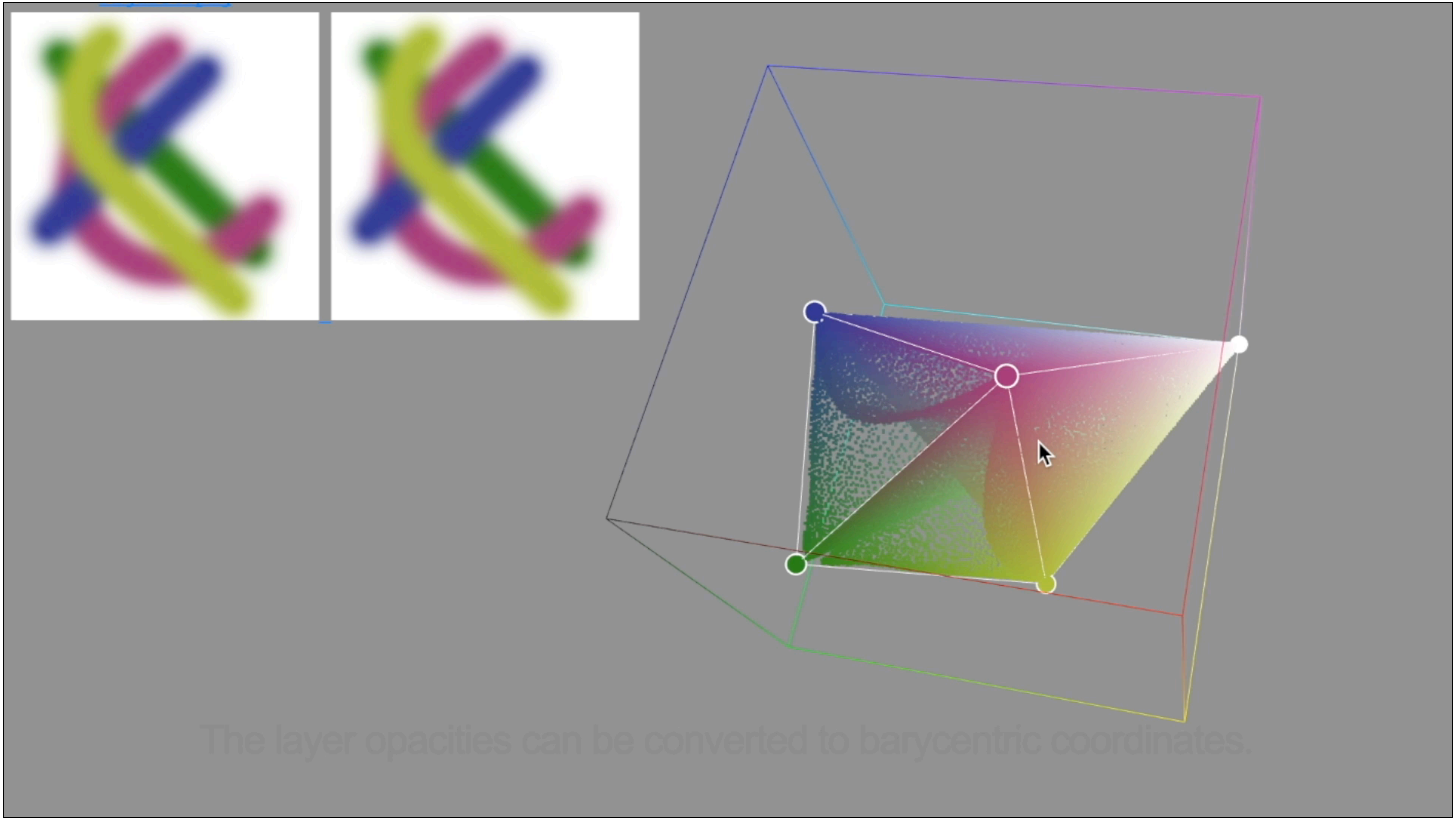


We then solve an optimization problem to extract translucent layers.

Layer opacity

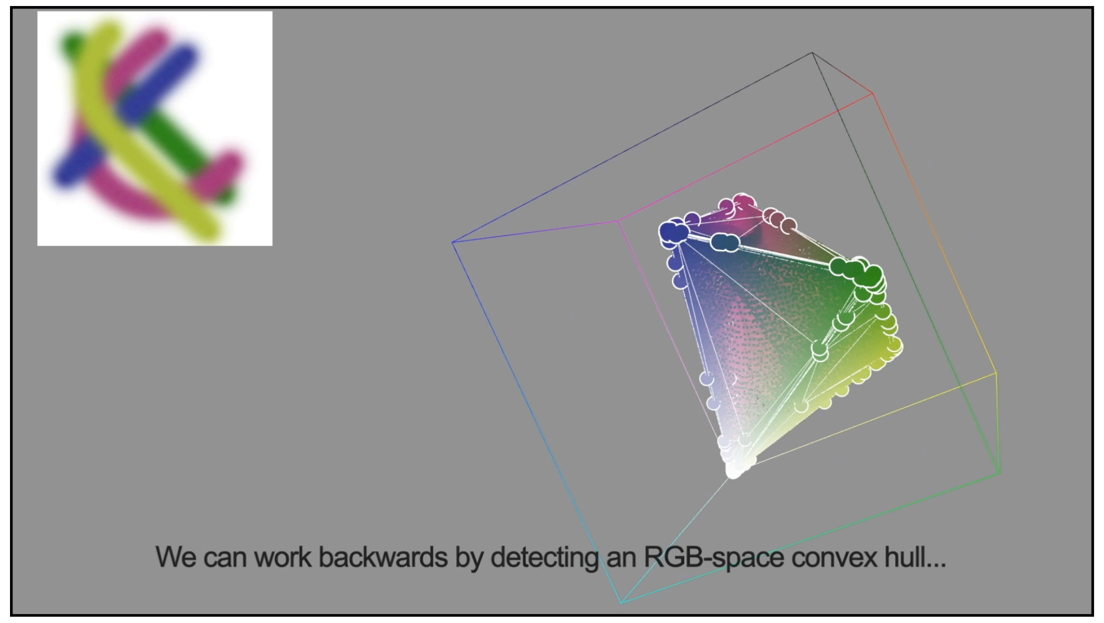

The layer opacities can be converted to barycentric coordinates.

Edit

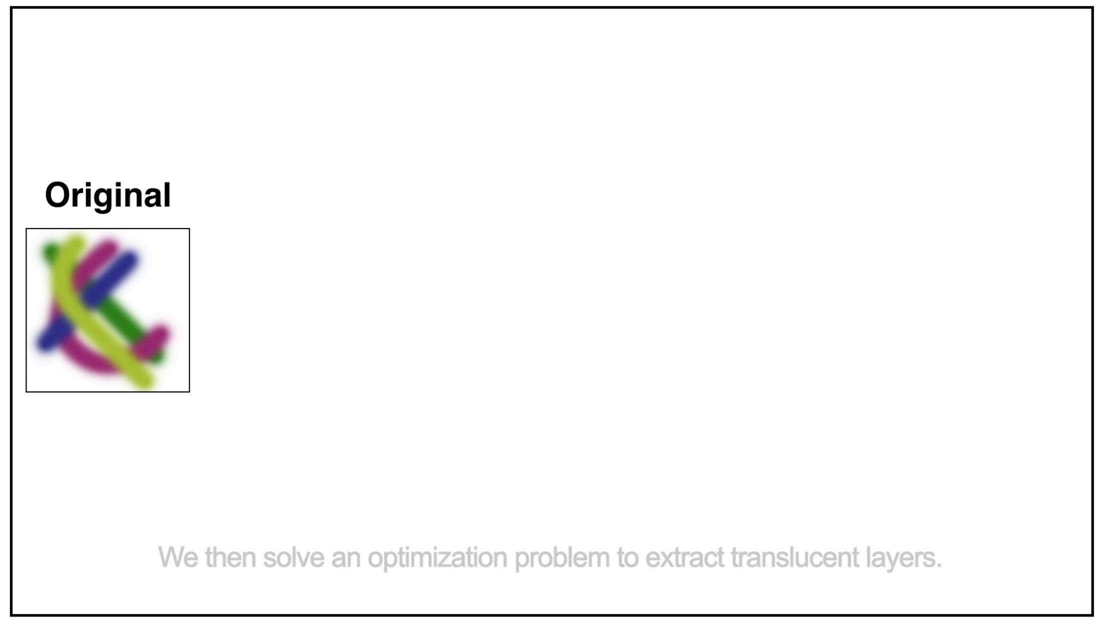

The layer opacities can be converted to barycentric coordinates.

Edit

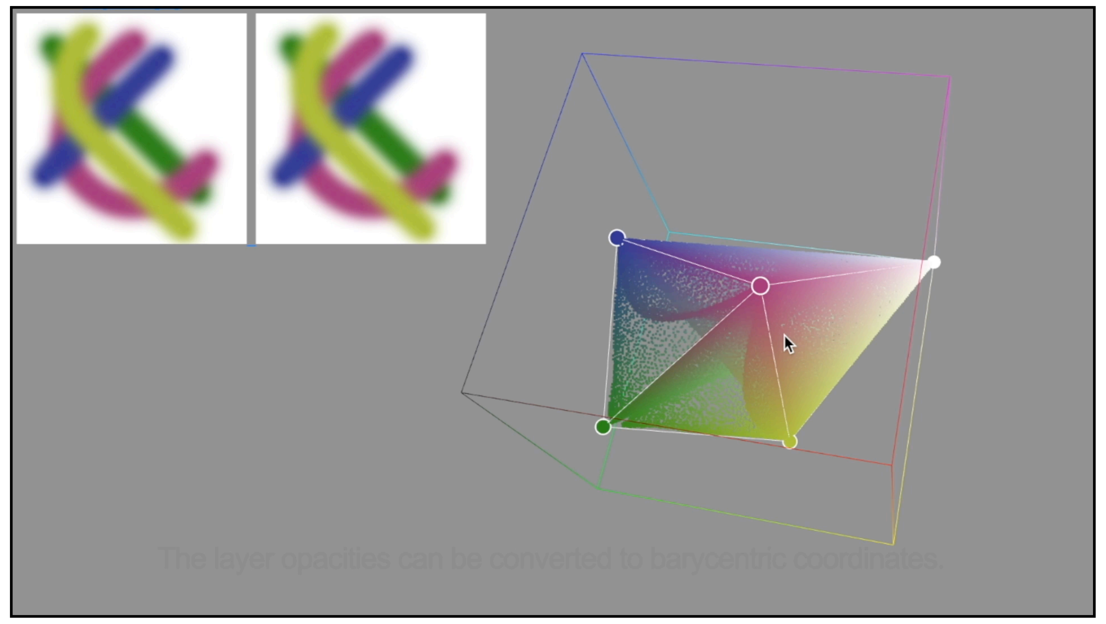
Input



Palette selection

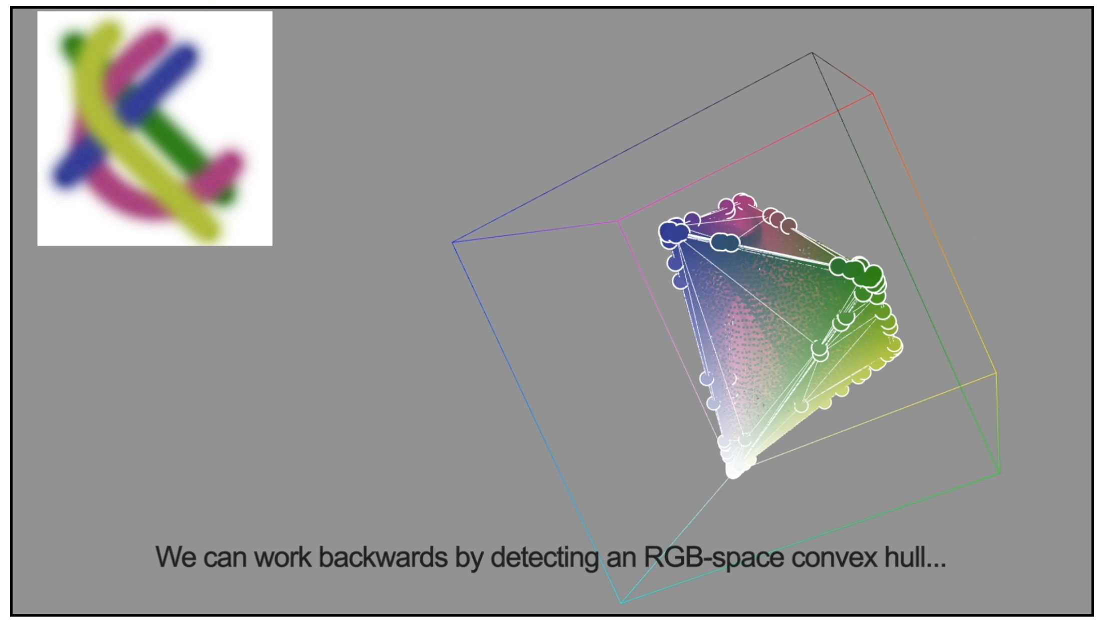


Layer opacity

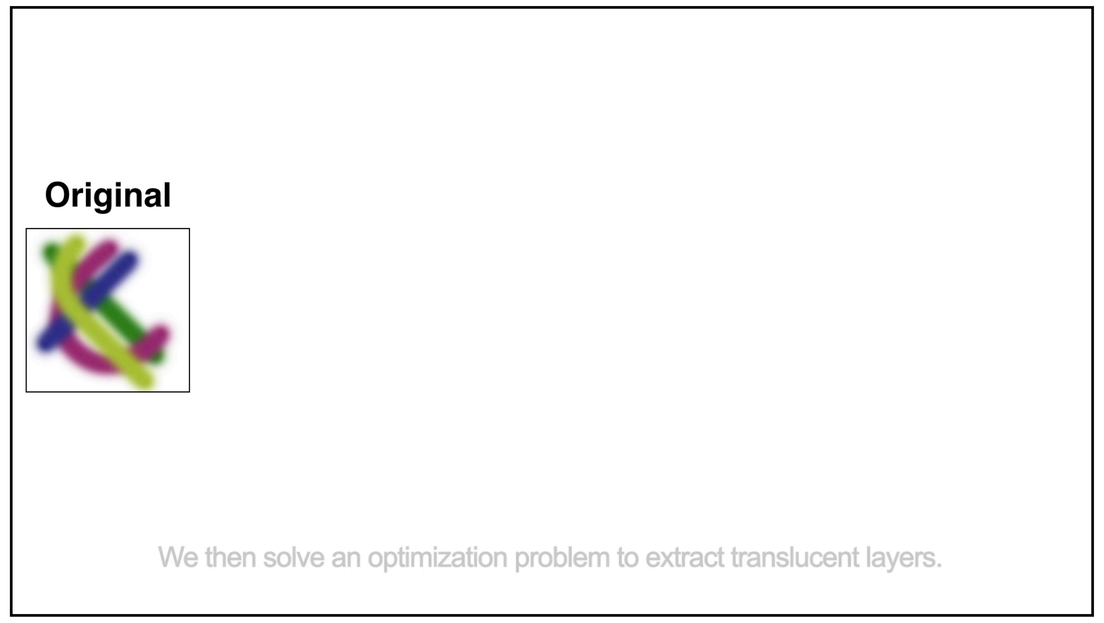


Edit

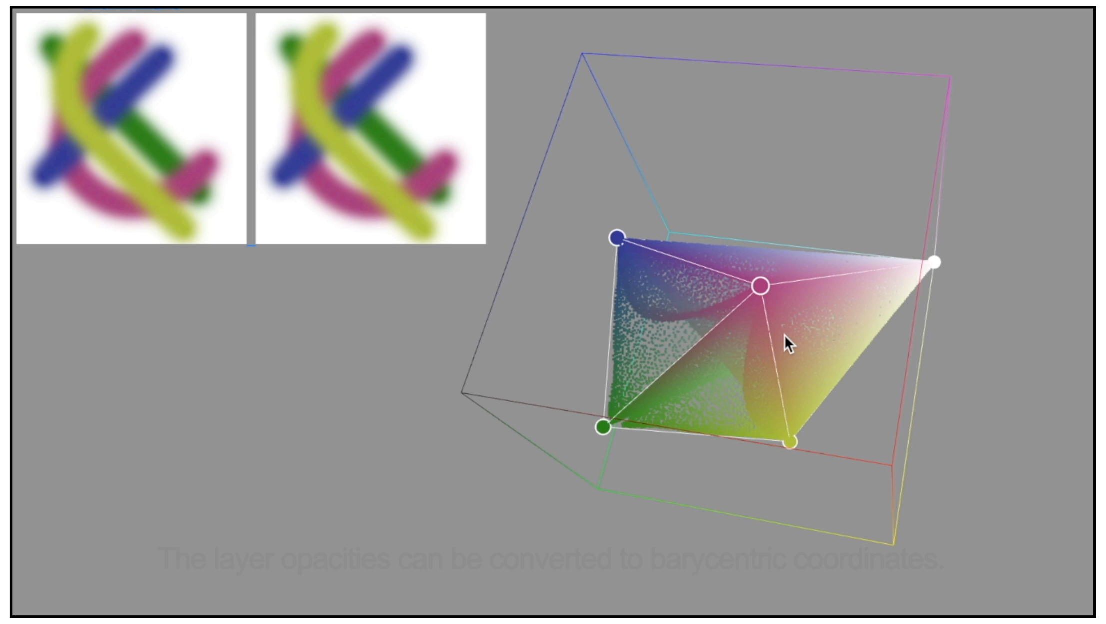
Input



Palette selection



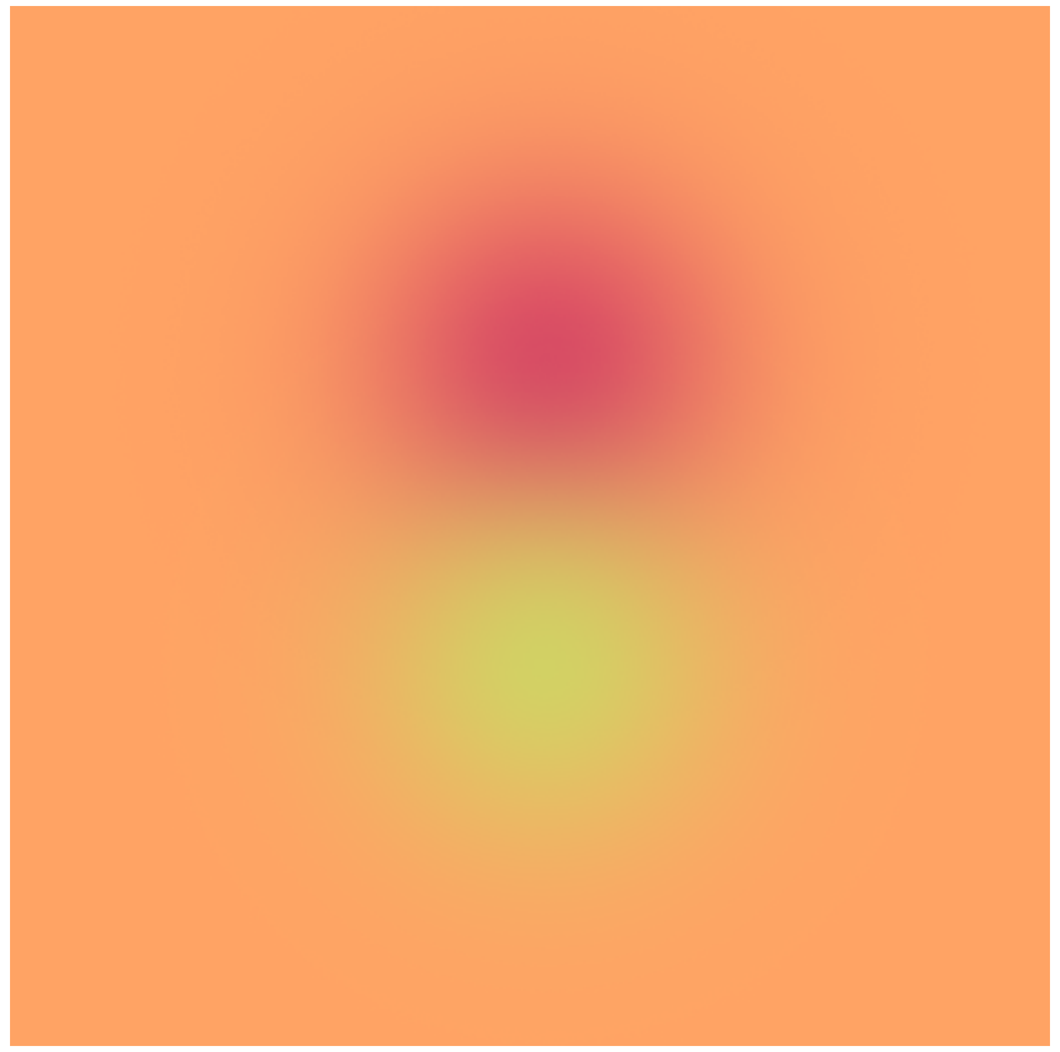
Layer opacity



Edit

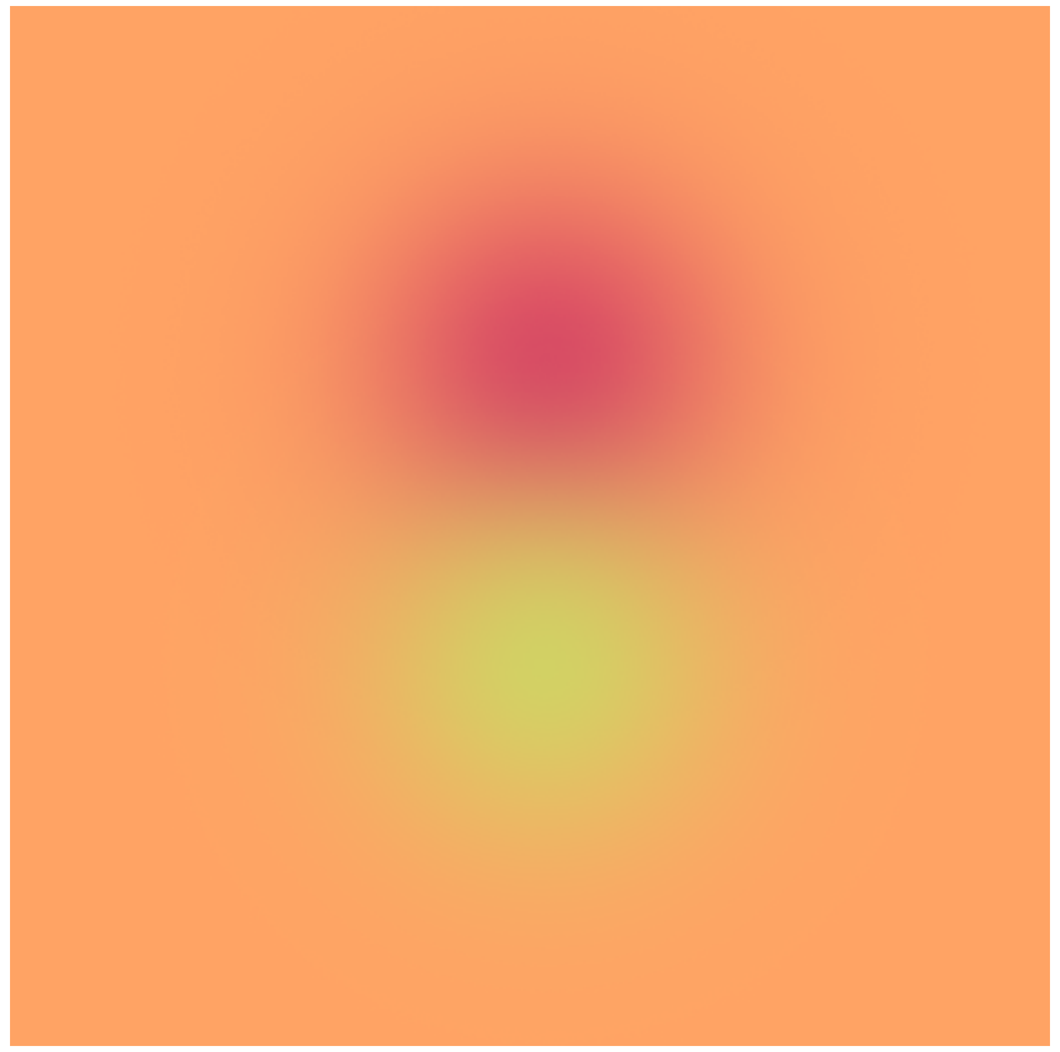
Palette Selection

Here is an image

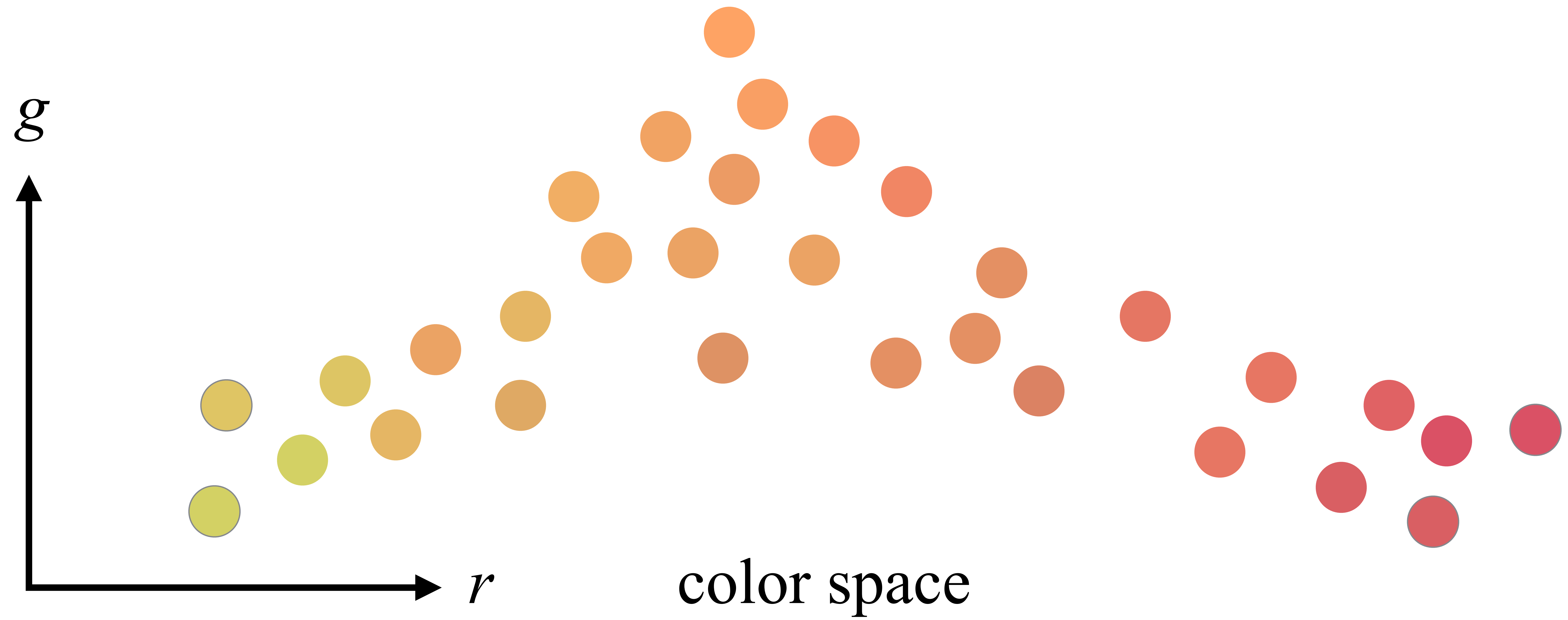


image

and its pixels in Red-Green space

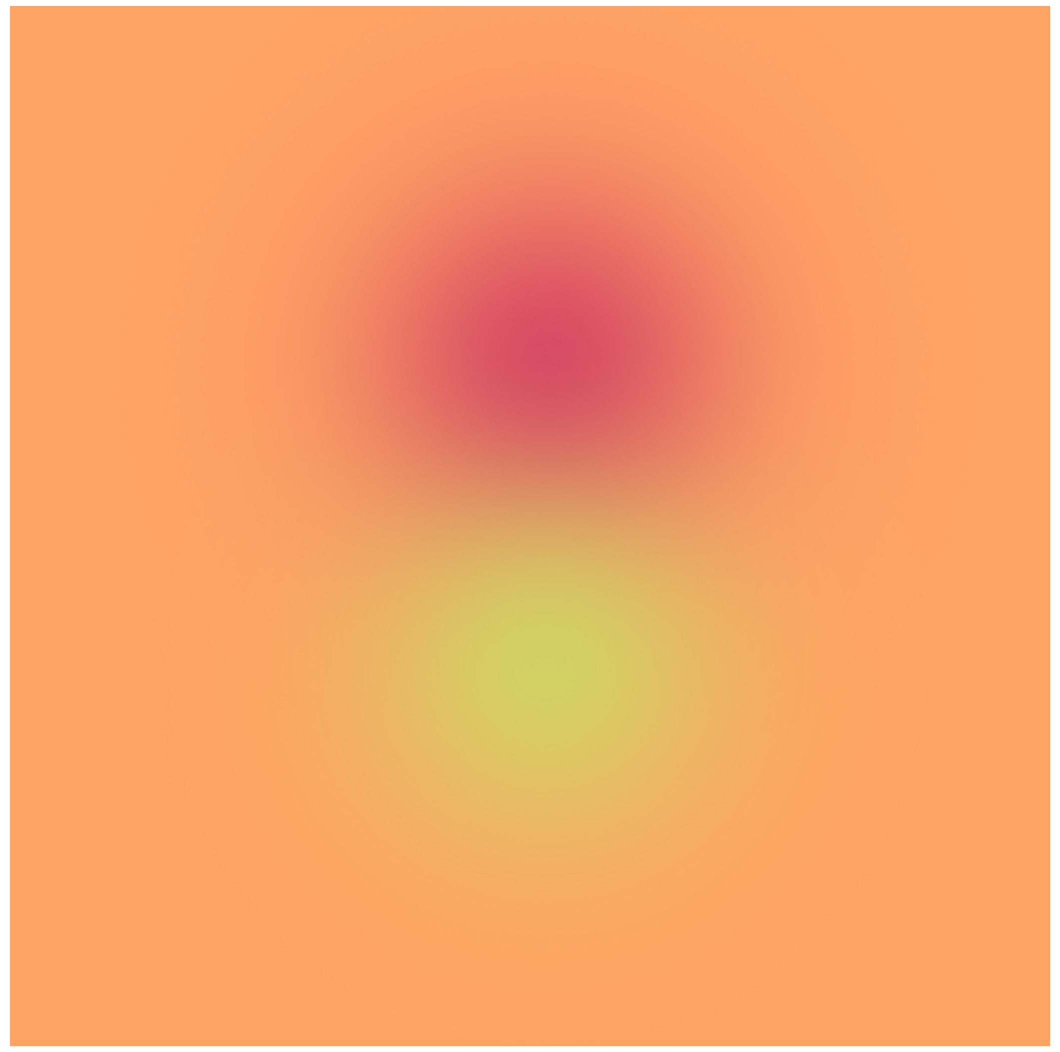


image

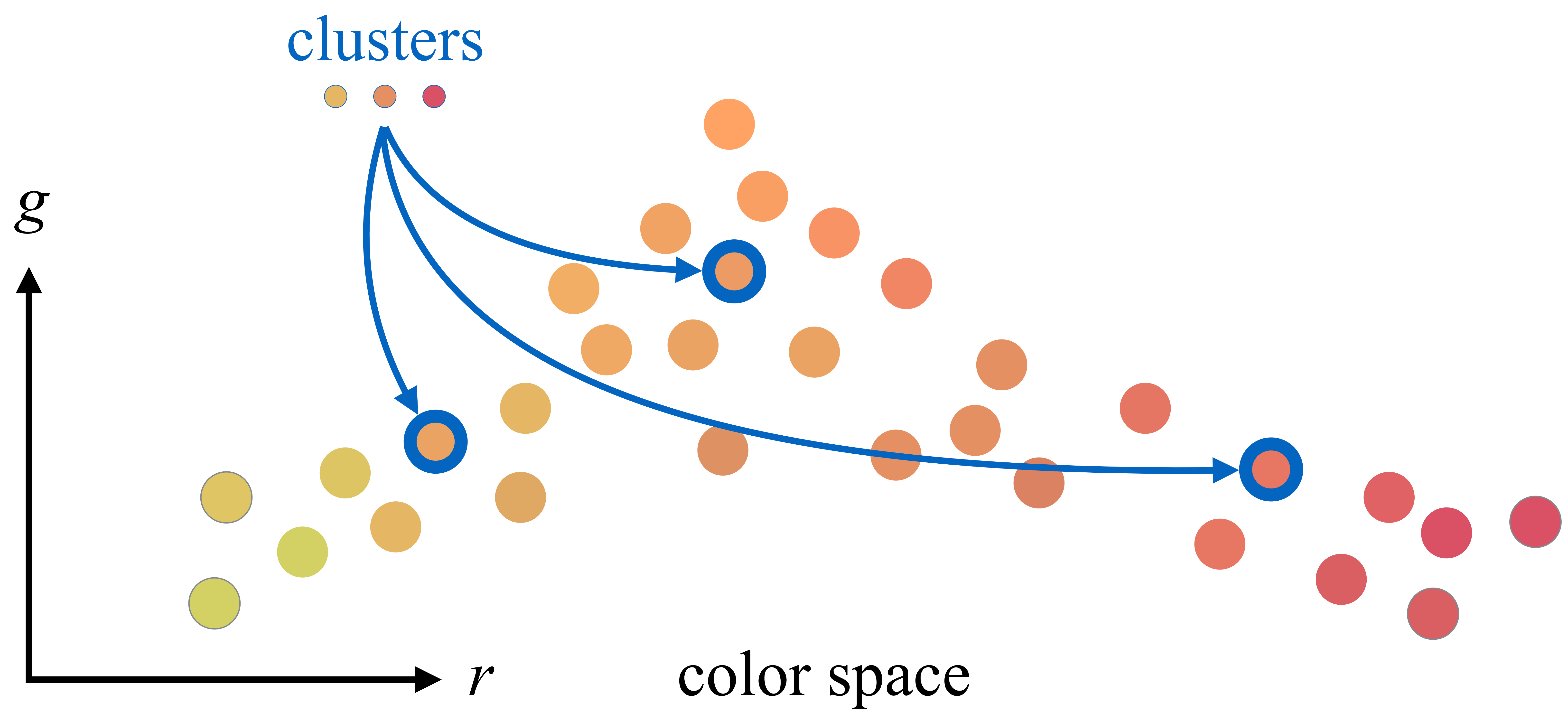


color space

Clustering finds these interior colors

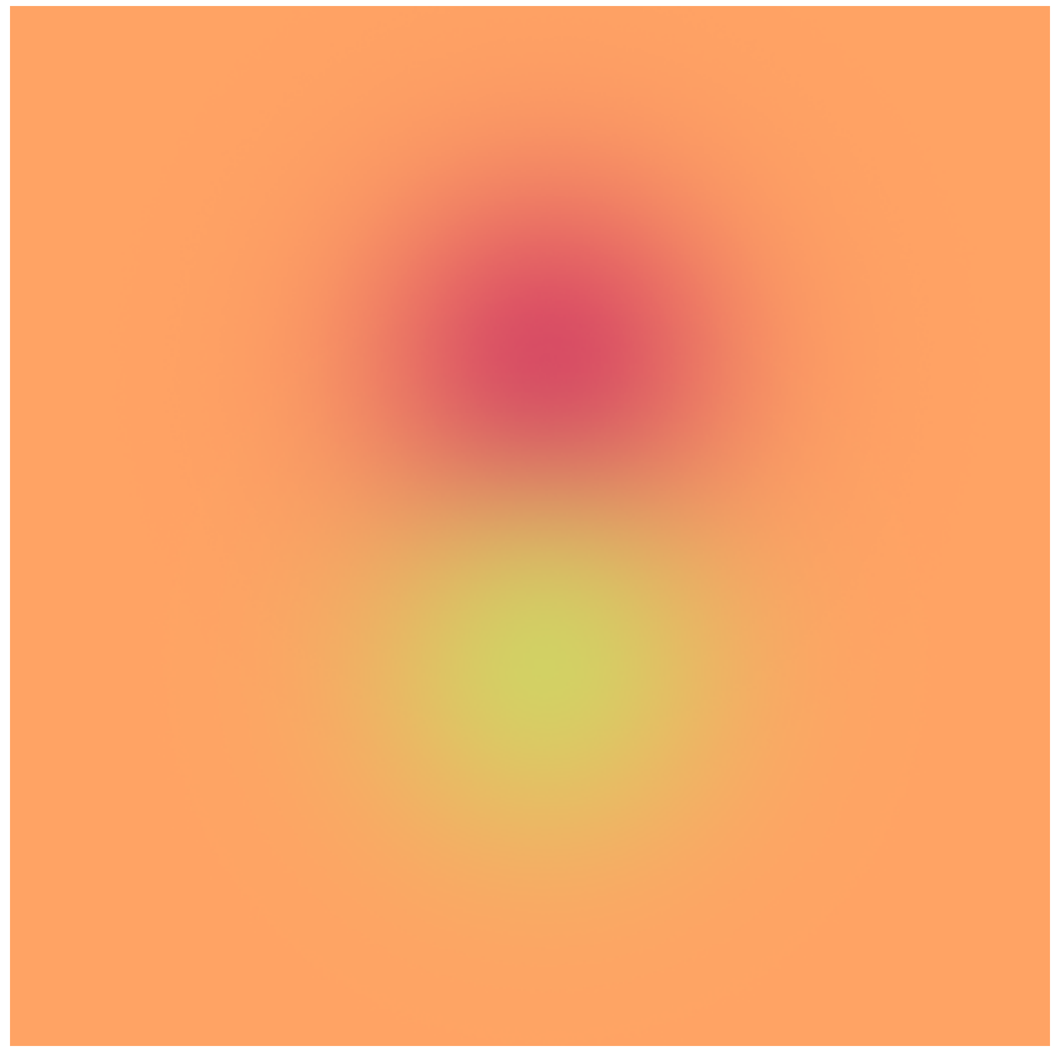


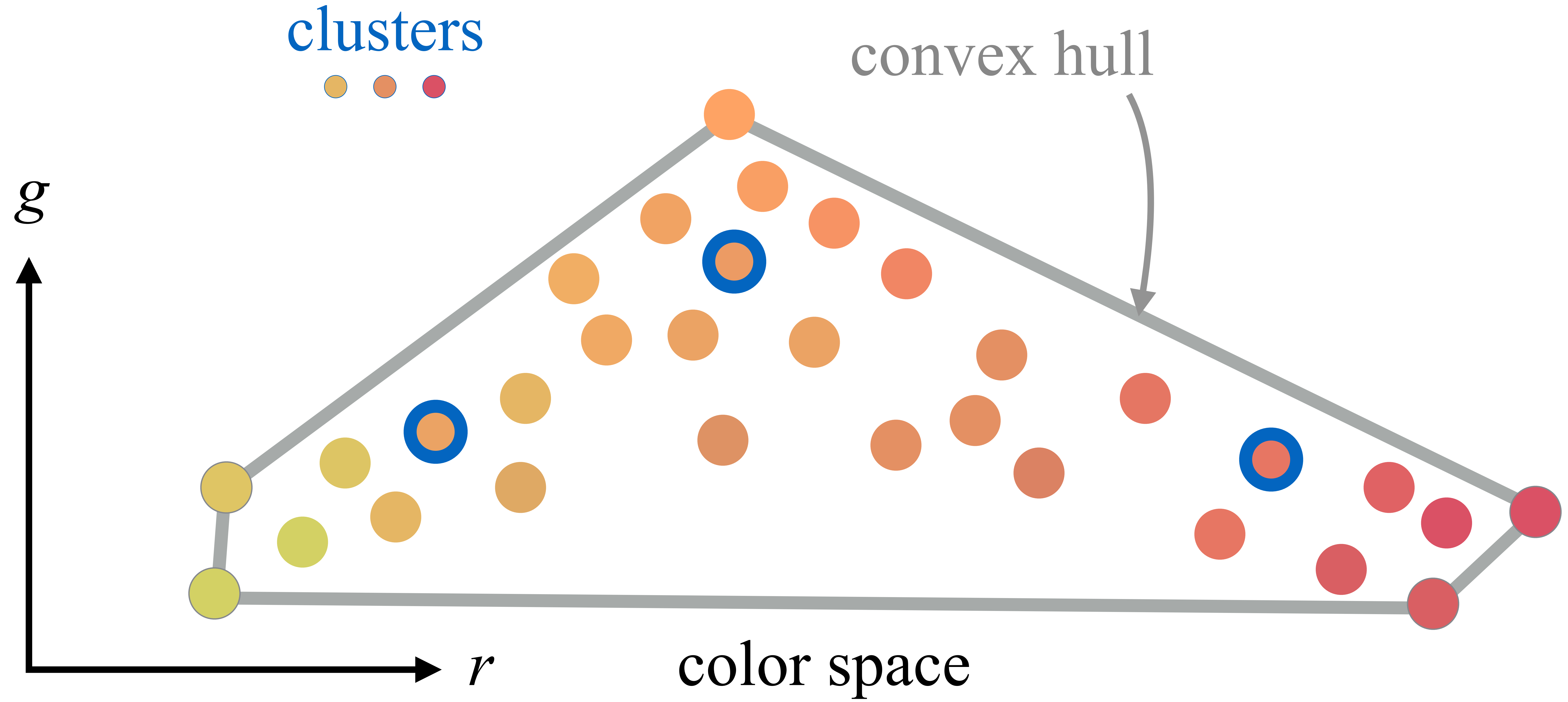
image



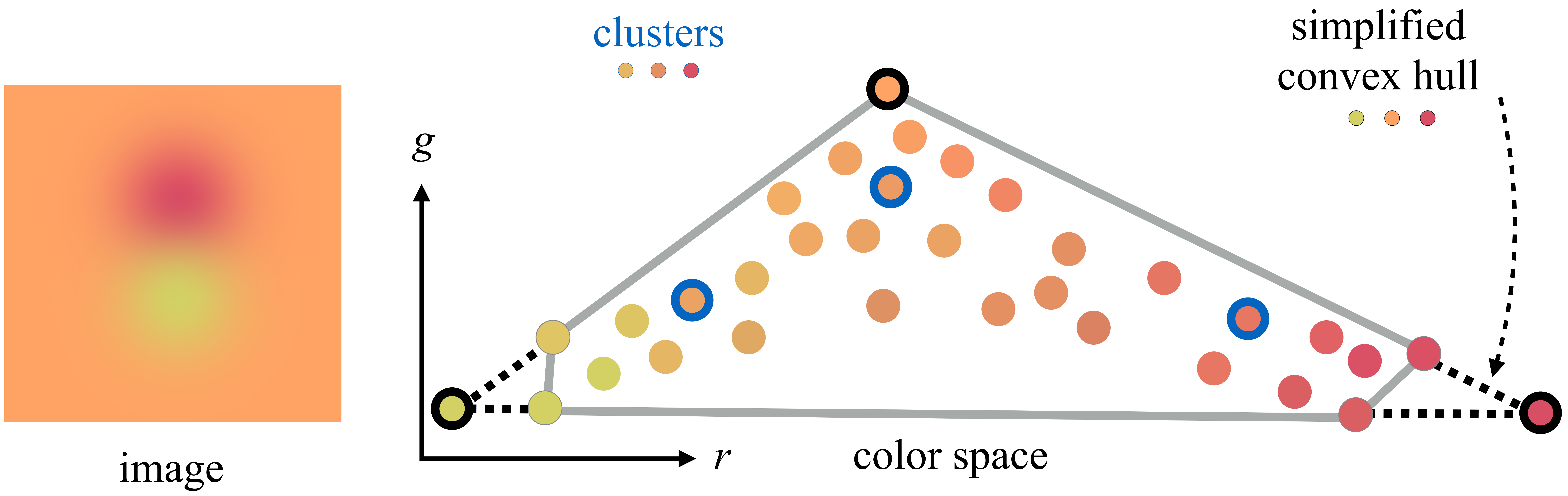
color space

The convex hull is complex

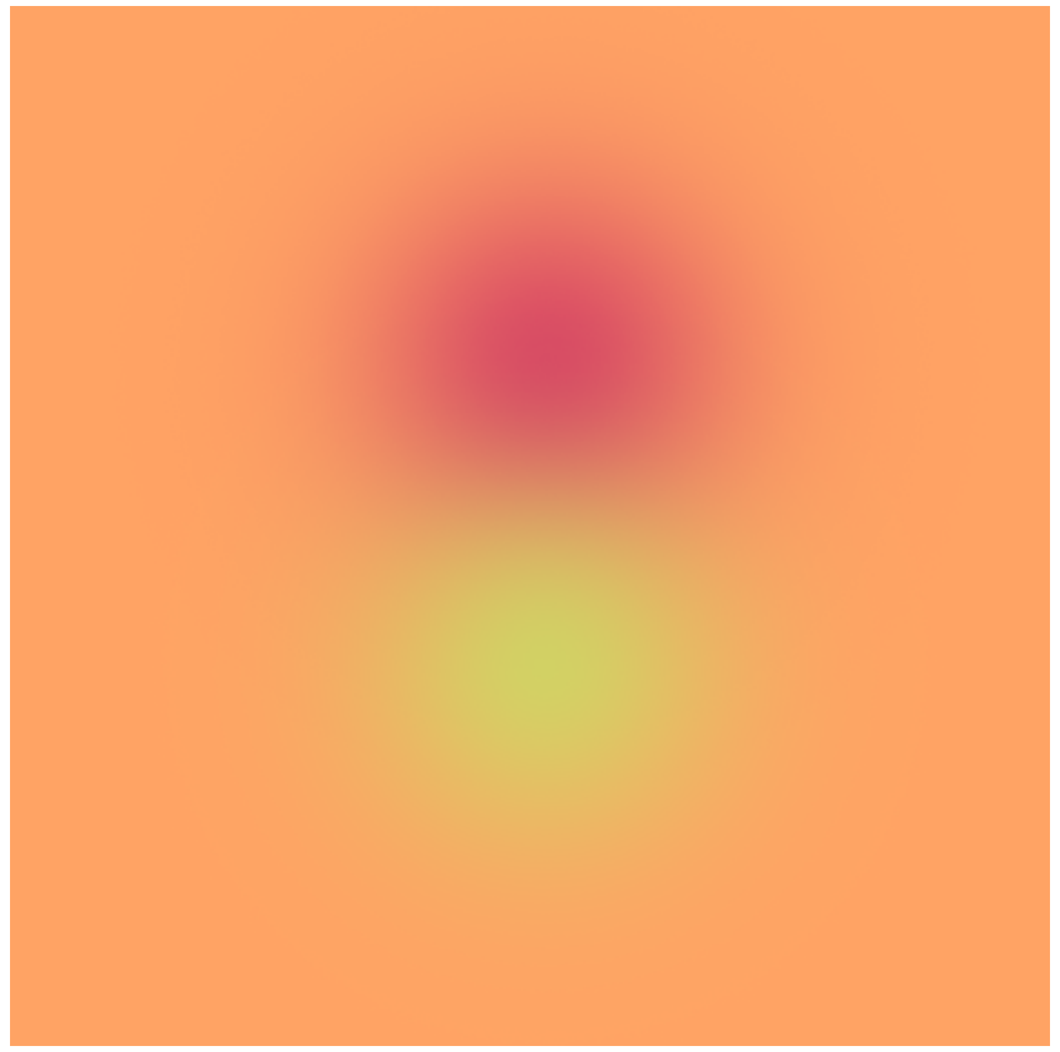




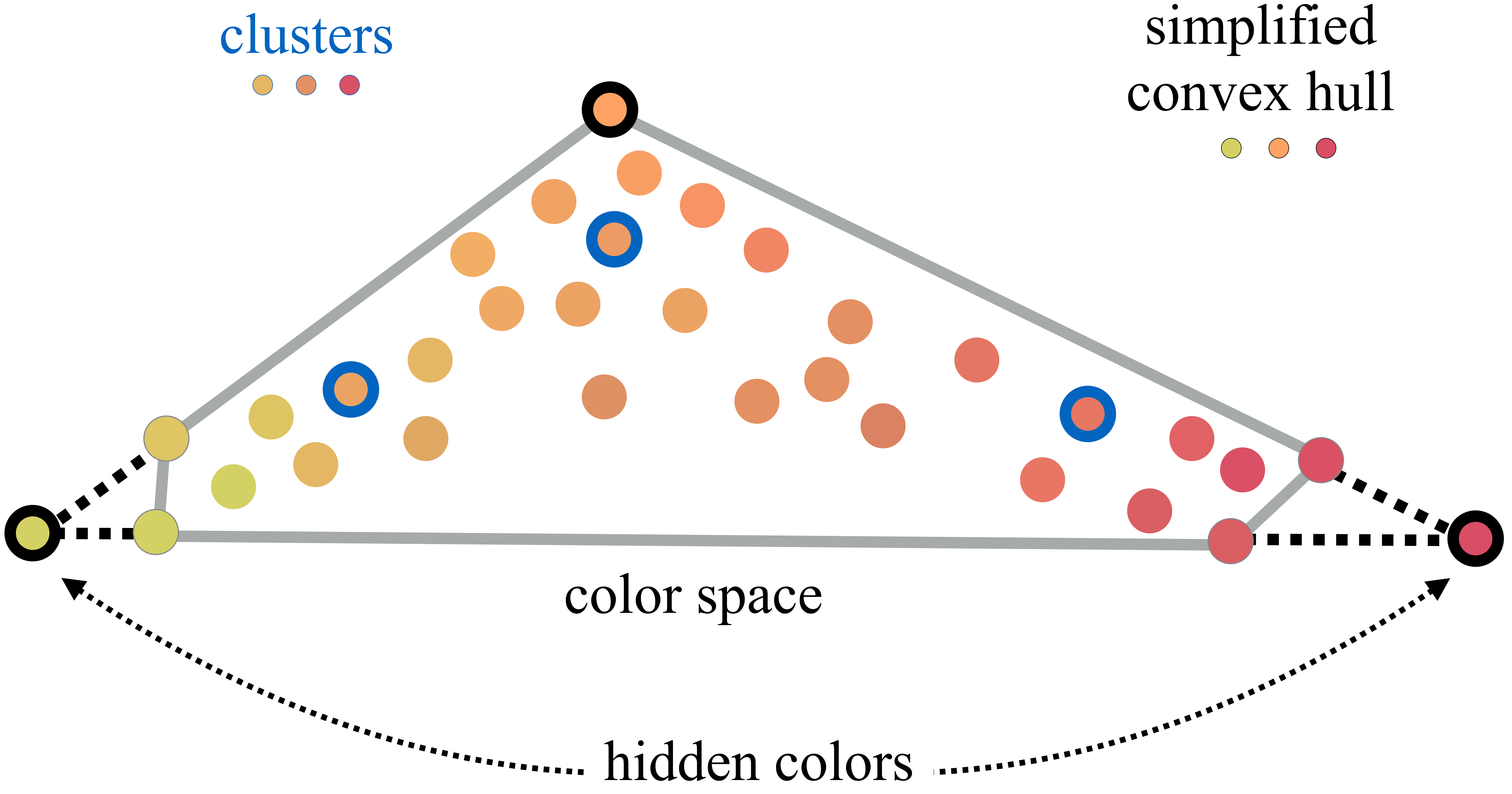
If we could simplify it…



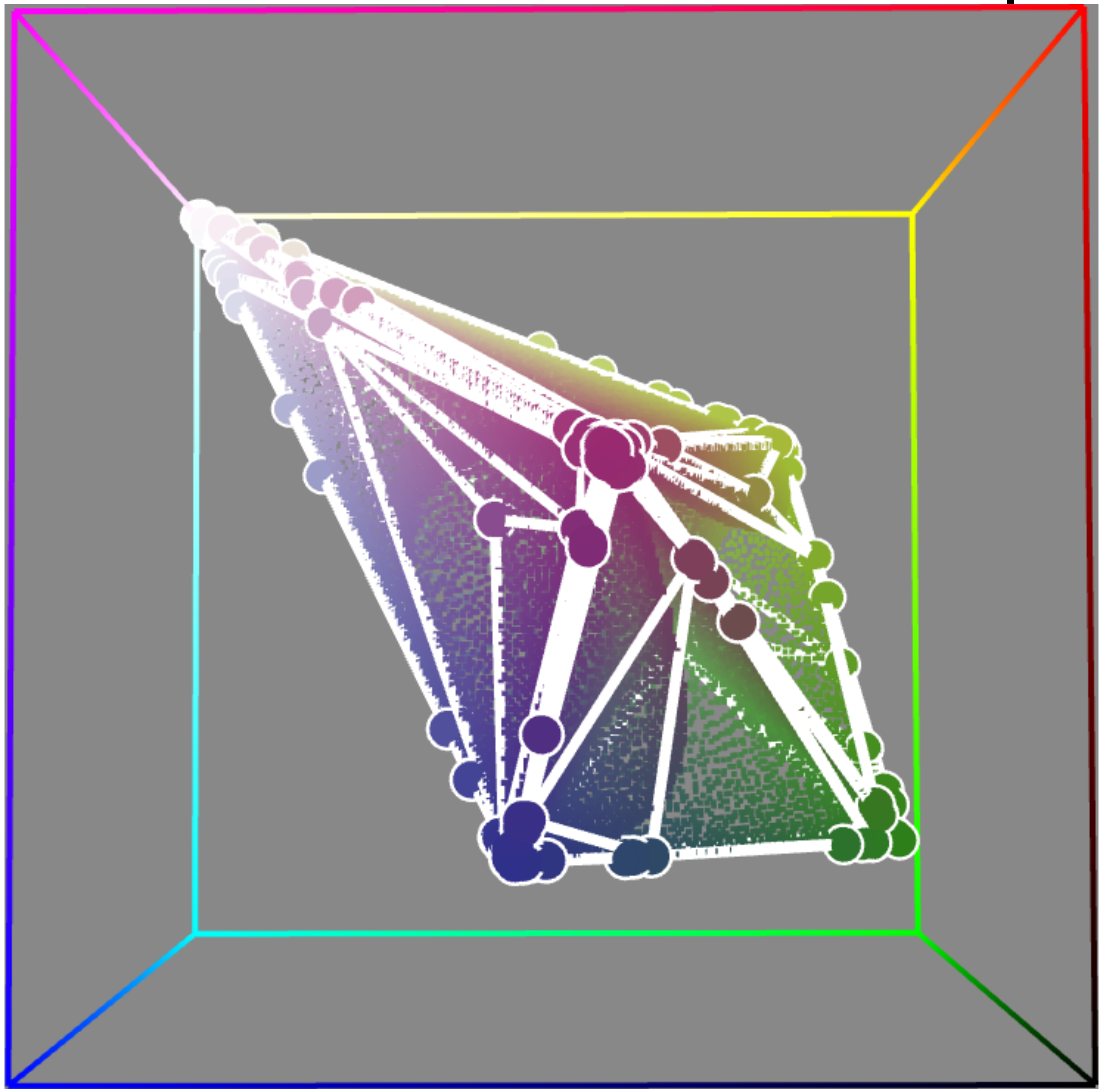
We would find the original, hidden palette



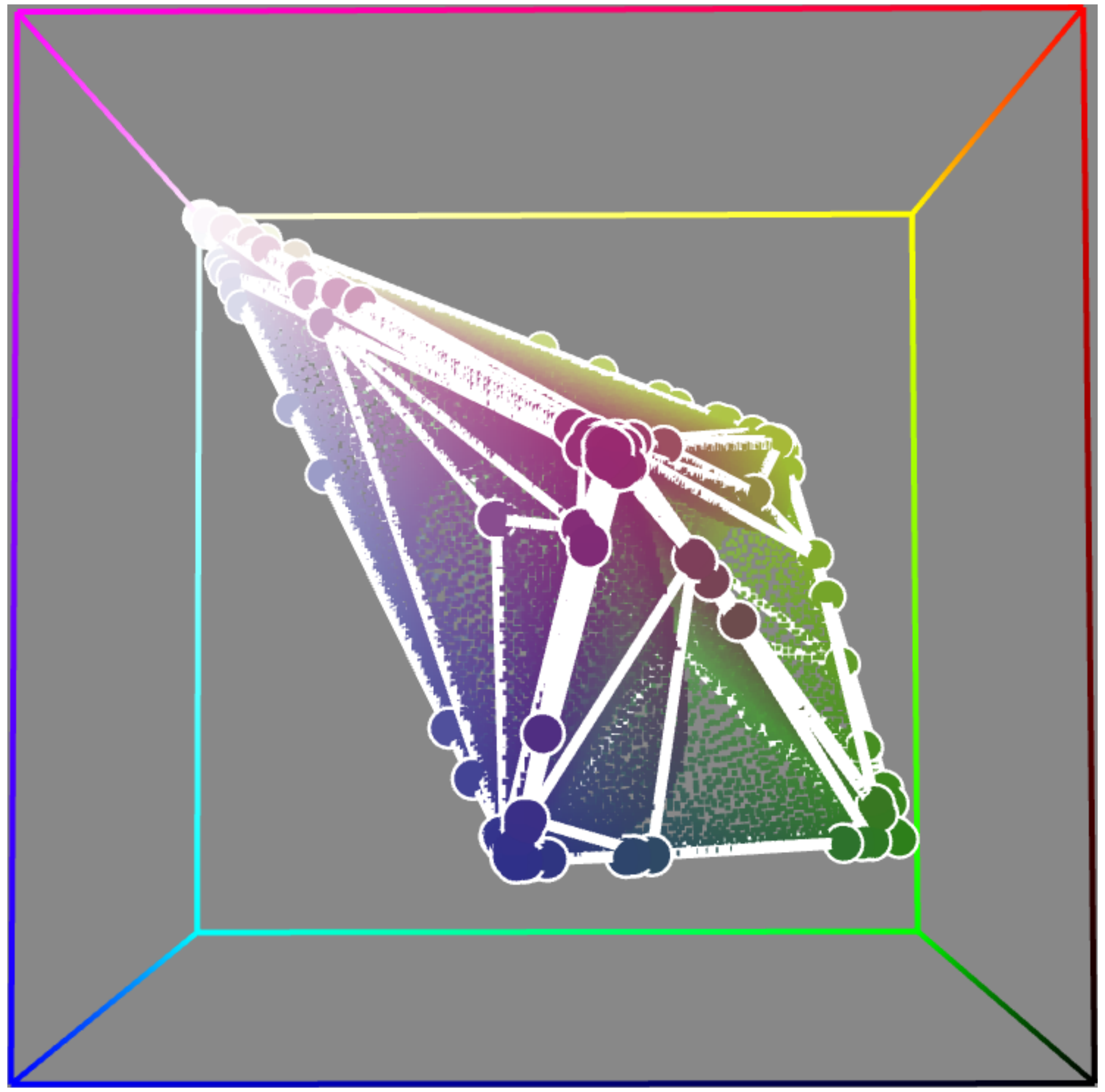
image



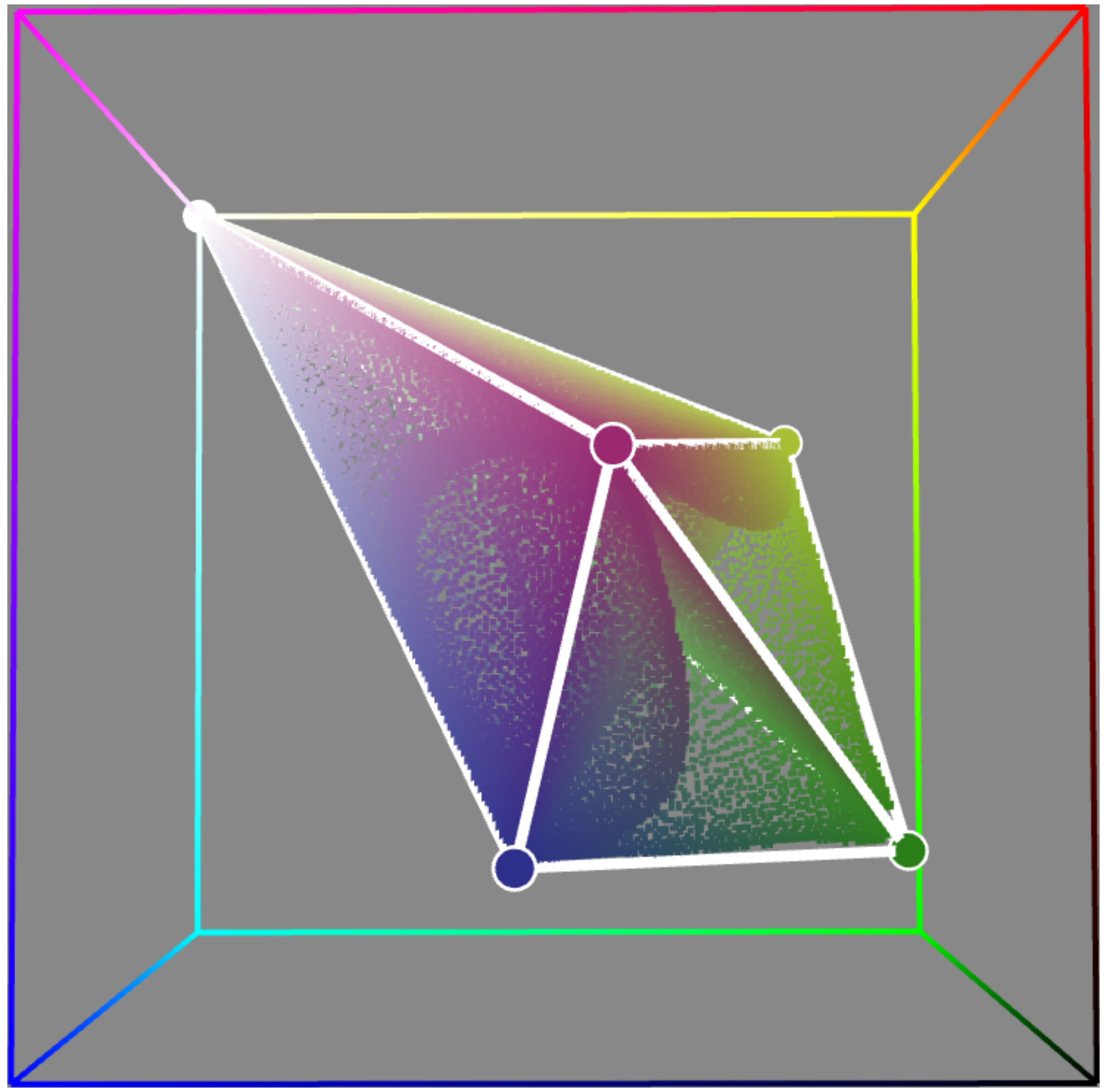
Convex Hull in RGB-space



Convex Hull simplification

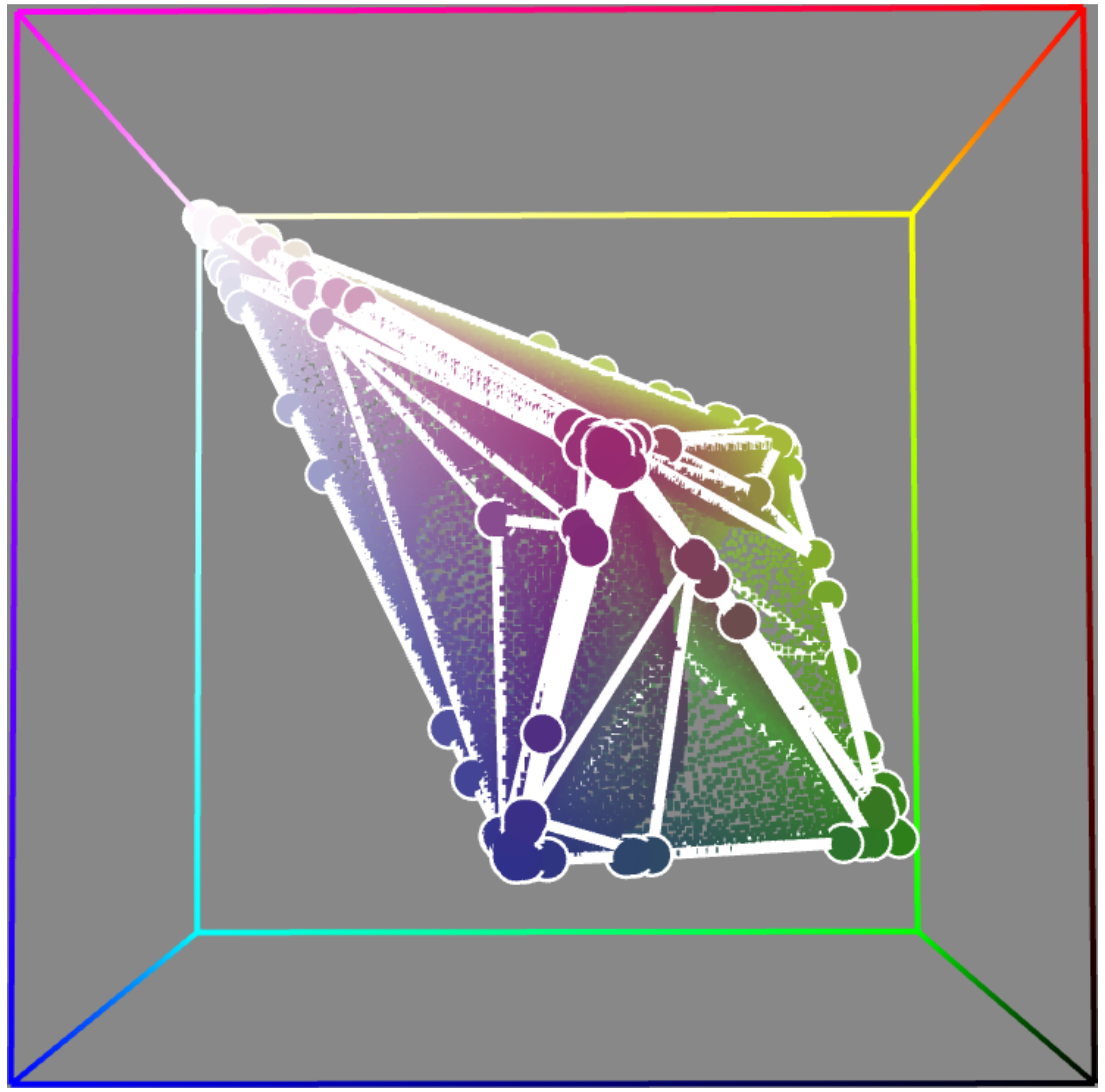


convex hull



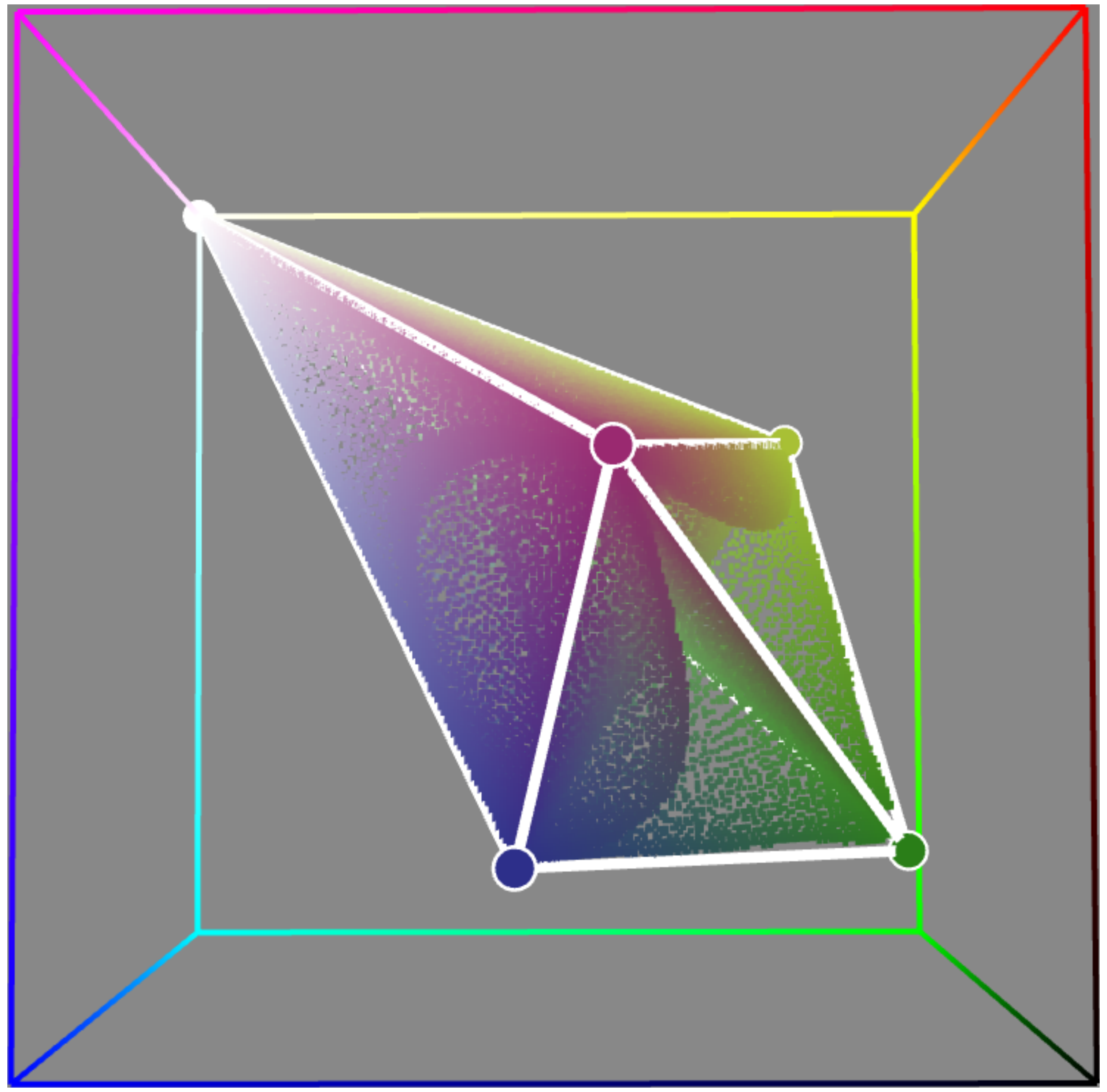
simplified convex hull

Convex Hull simplification



convex hull

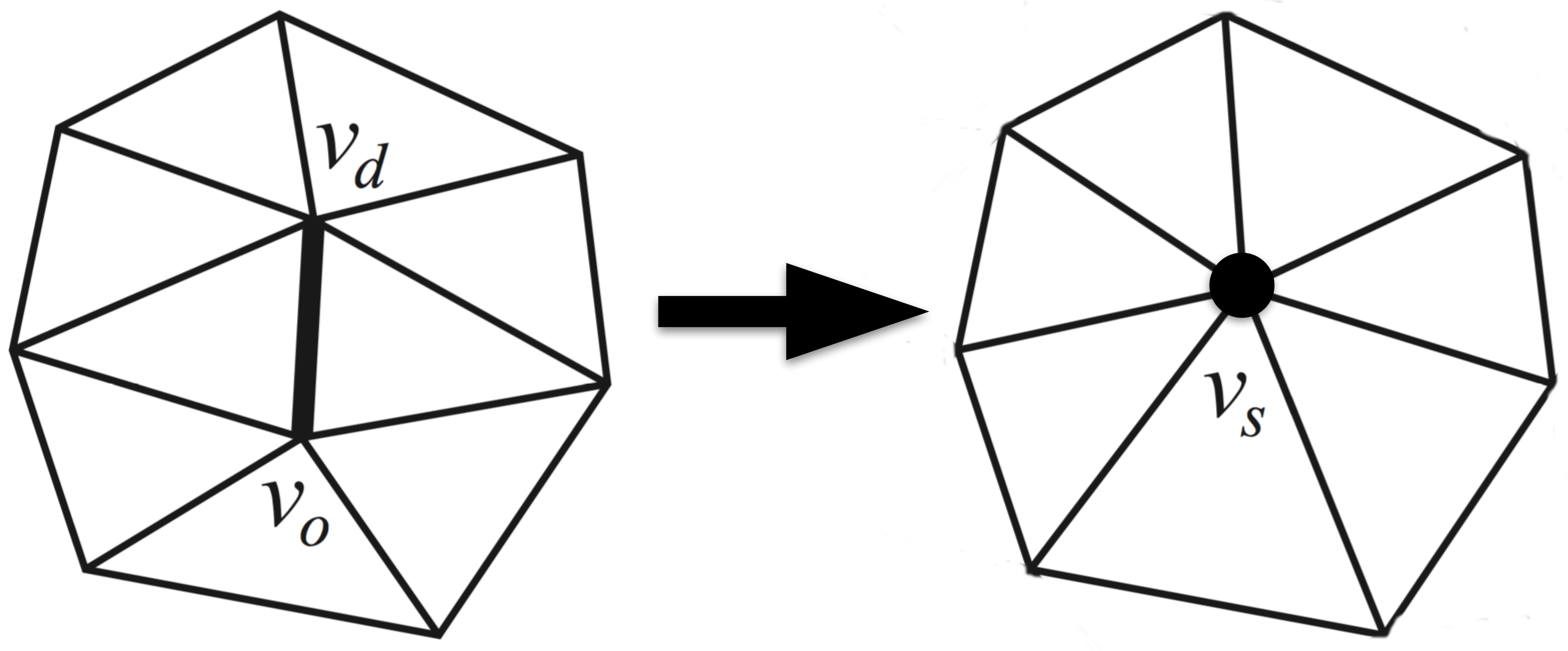
Iteratively collapse edges with a modified Progressive Hull method



simplified convex hull

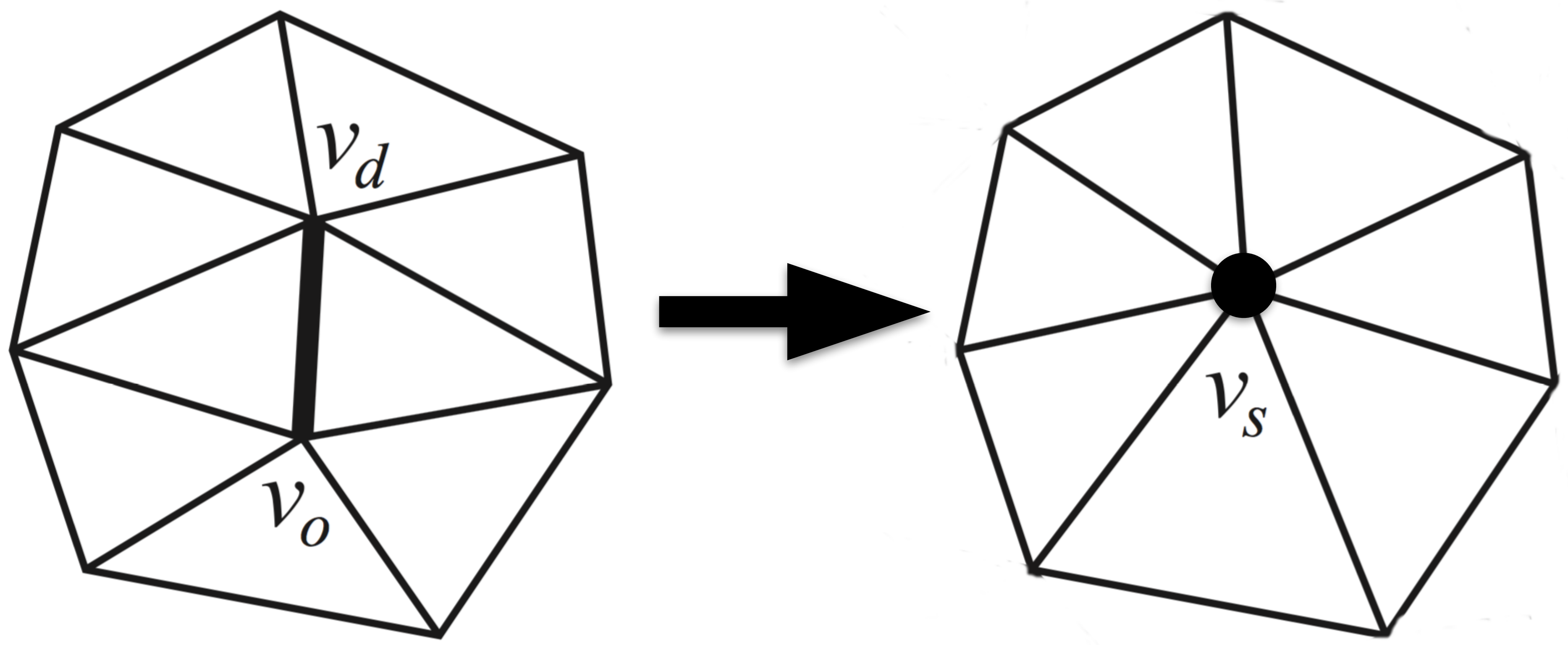
Progressive Hull [Sander et al. 2000]

- Greedily collapse edges whose new vertex position adds the smallest additional volume.

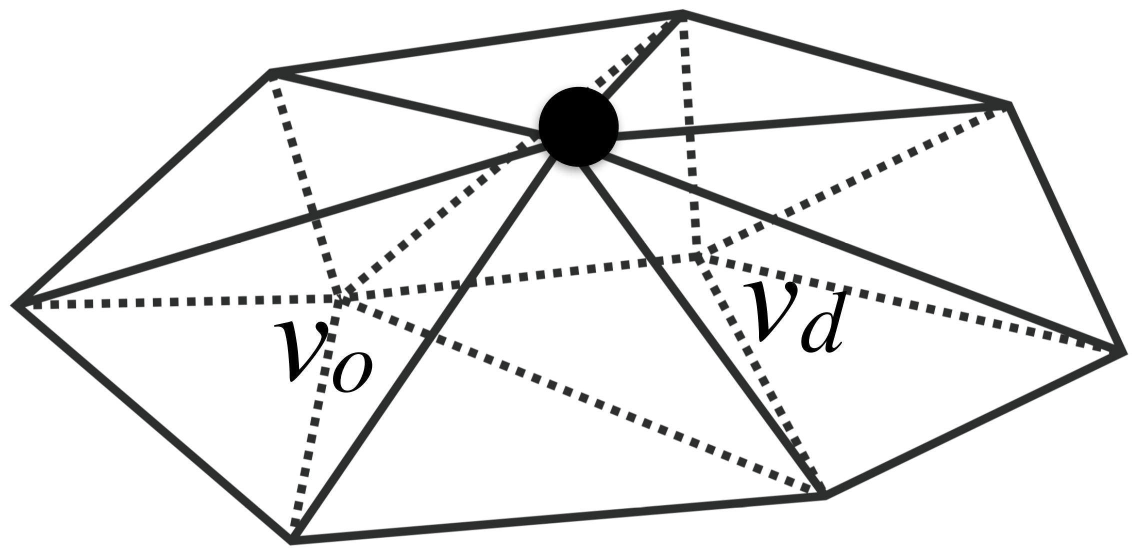


Progressive Hull [Sander et al. 2000]

- Greedily collapse edges whose new vertex position adds the smallest additional volume.

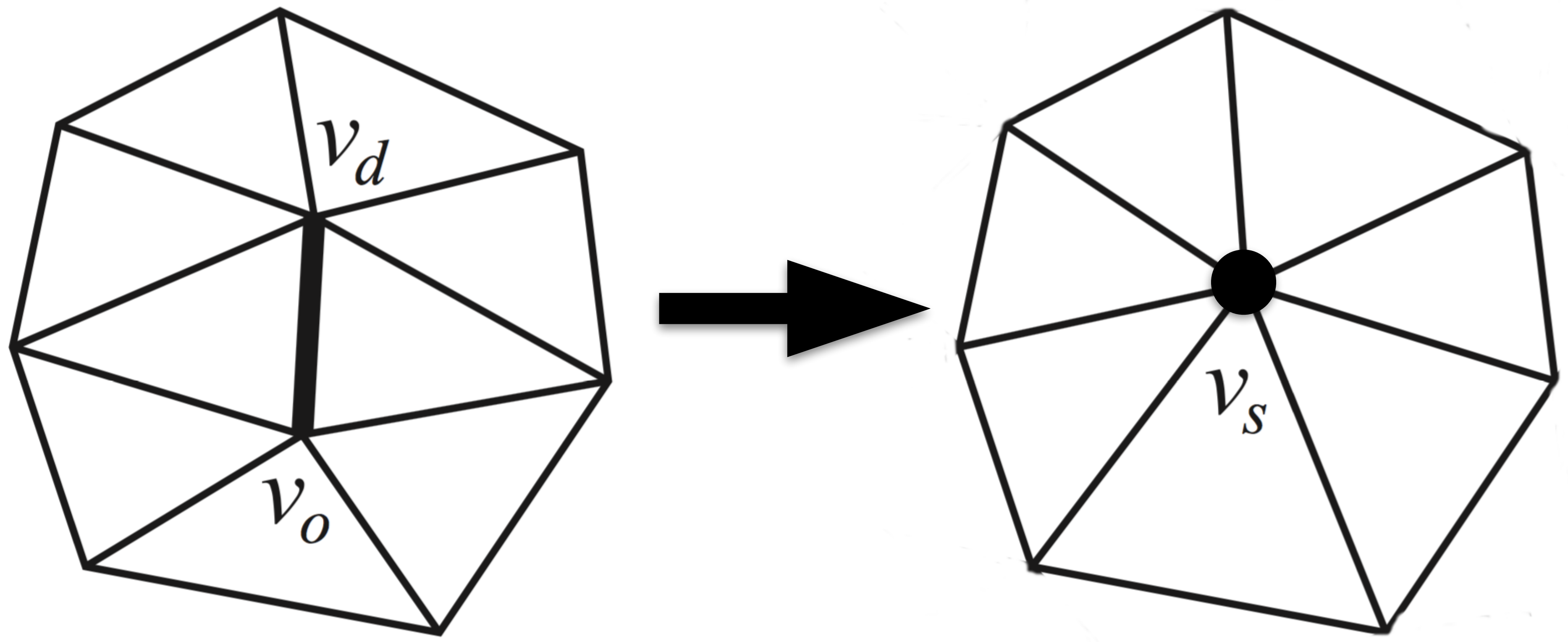


- New vertex position guarantees that volume expands (linear constraint)

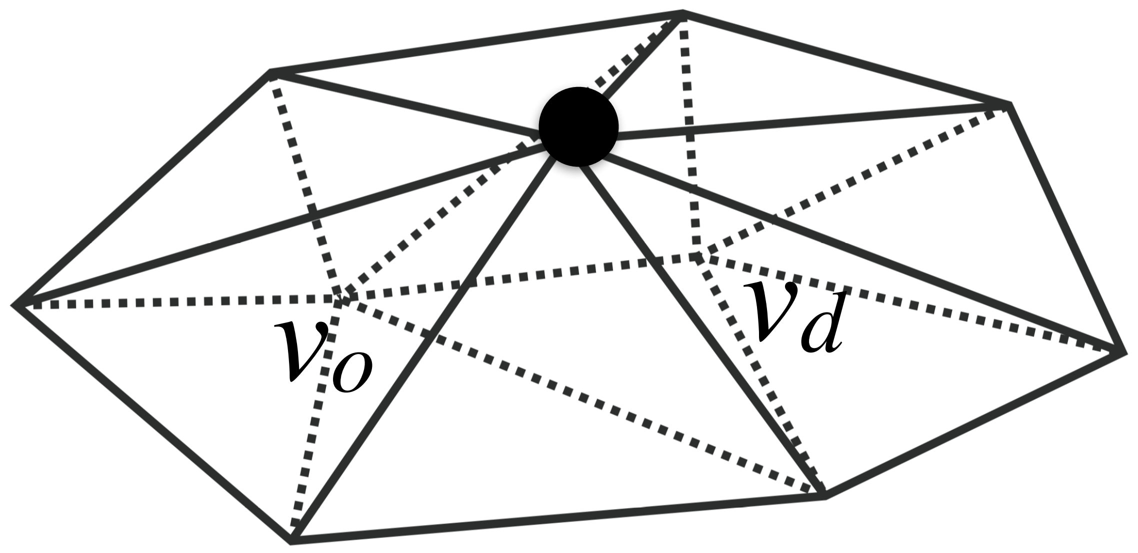


Progressive Hull [Sander et al. 2000]

- Greedily collapse edges whose new vertex position adds the smallest additional volume.



- New vertex position guarantees that volume expands (linear constraint)

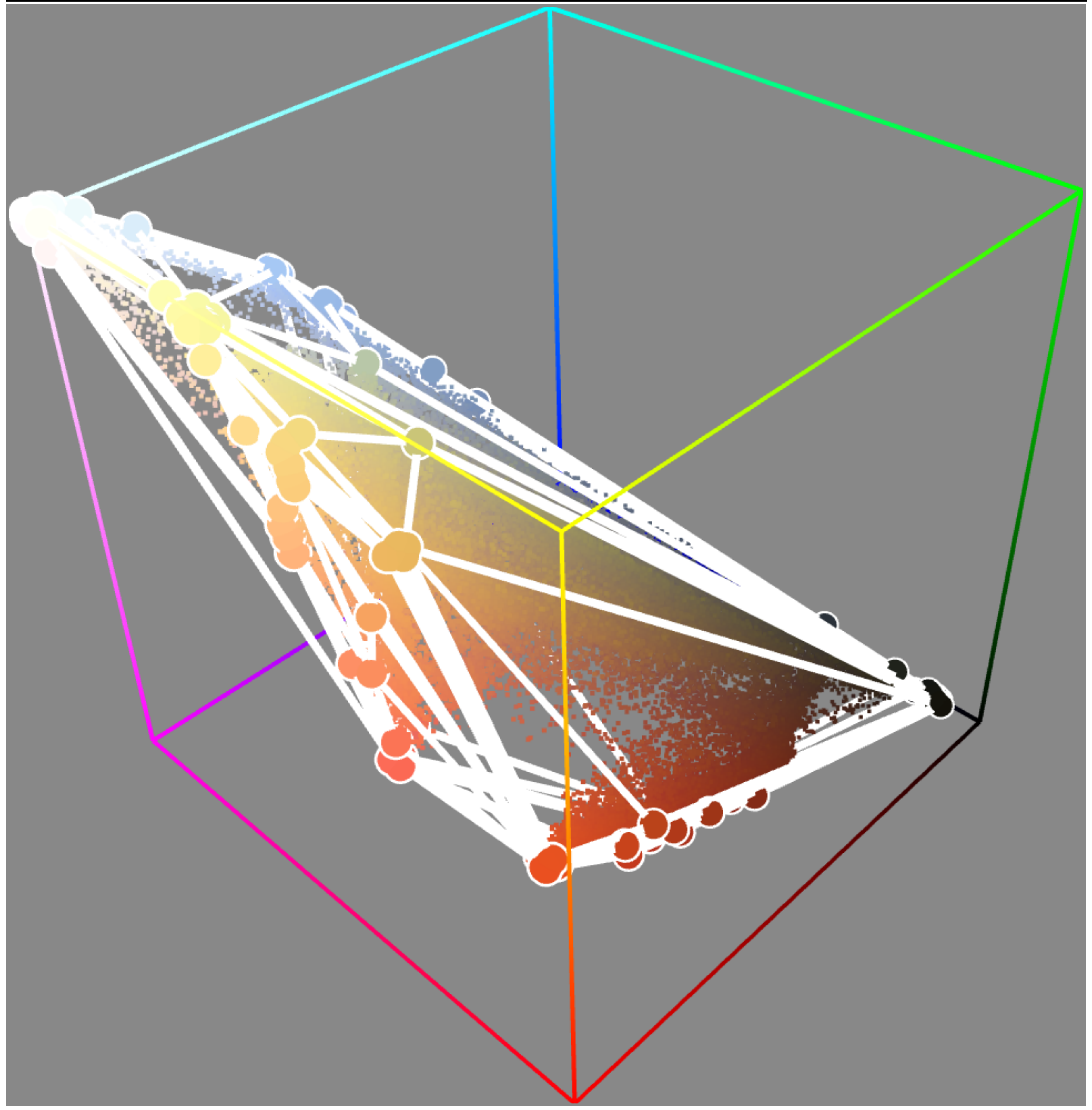


- We modify the algorithm: choose the new vertex that minimizes the distance to incident faces of the collapsing edge.

Palette Size

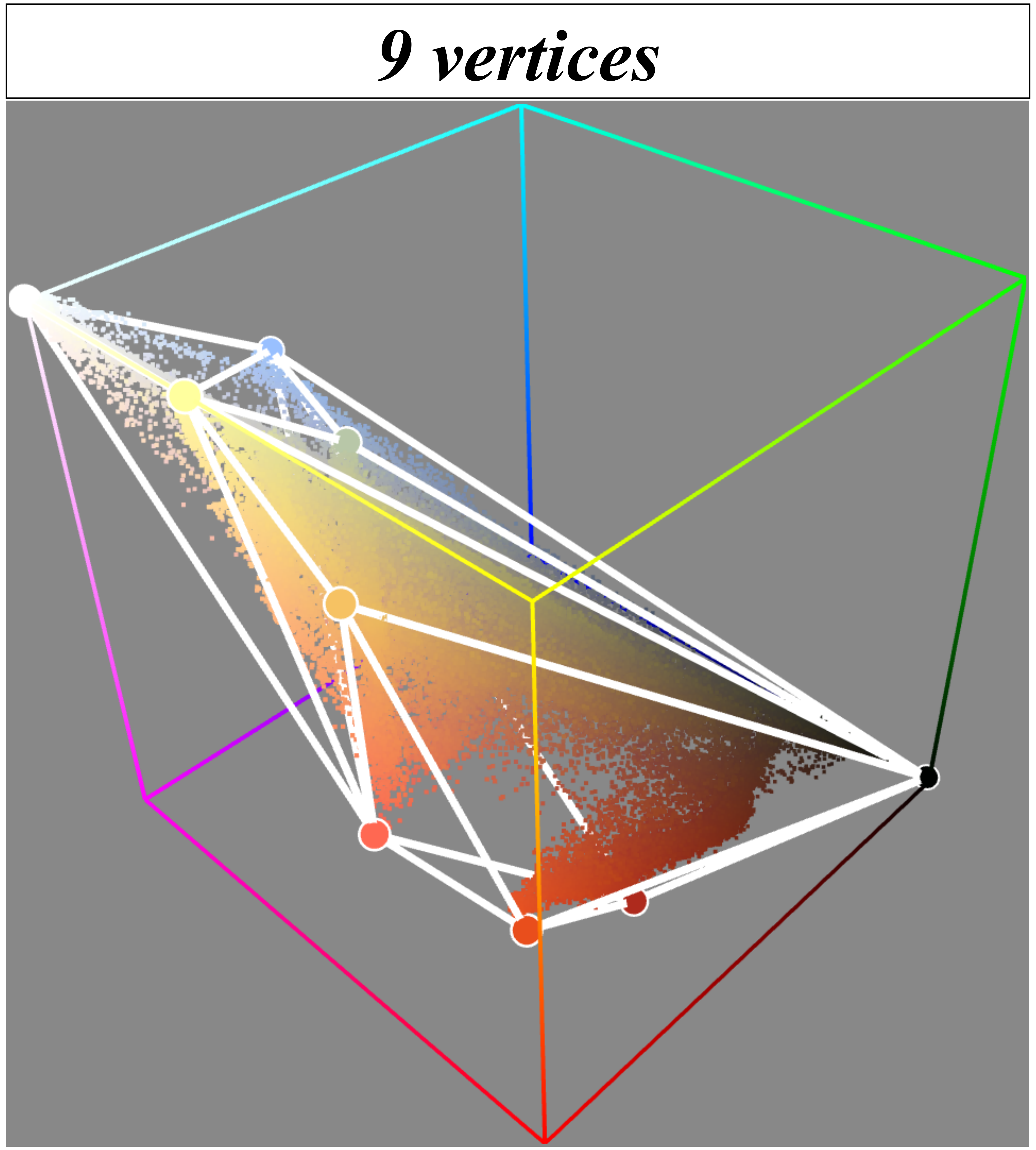
- The convex hull can be simplified to any complexity level.

RGB-space convex hull



Palette Size

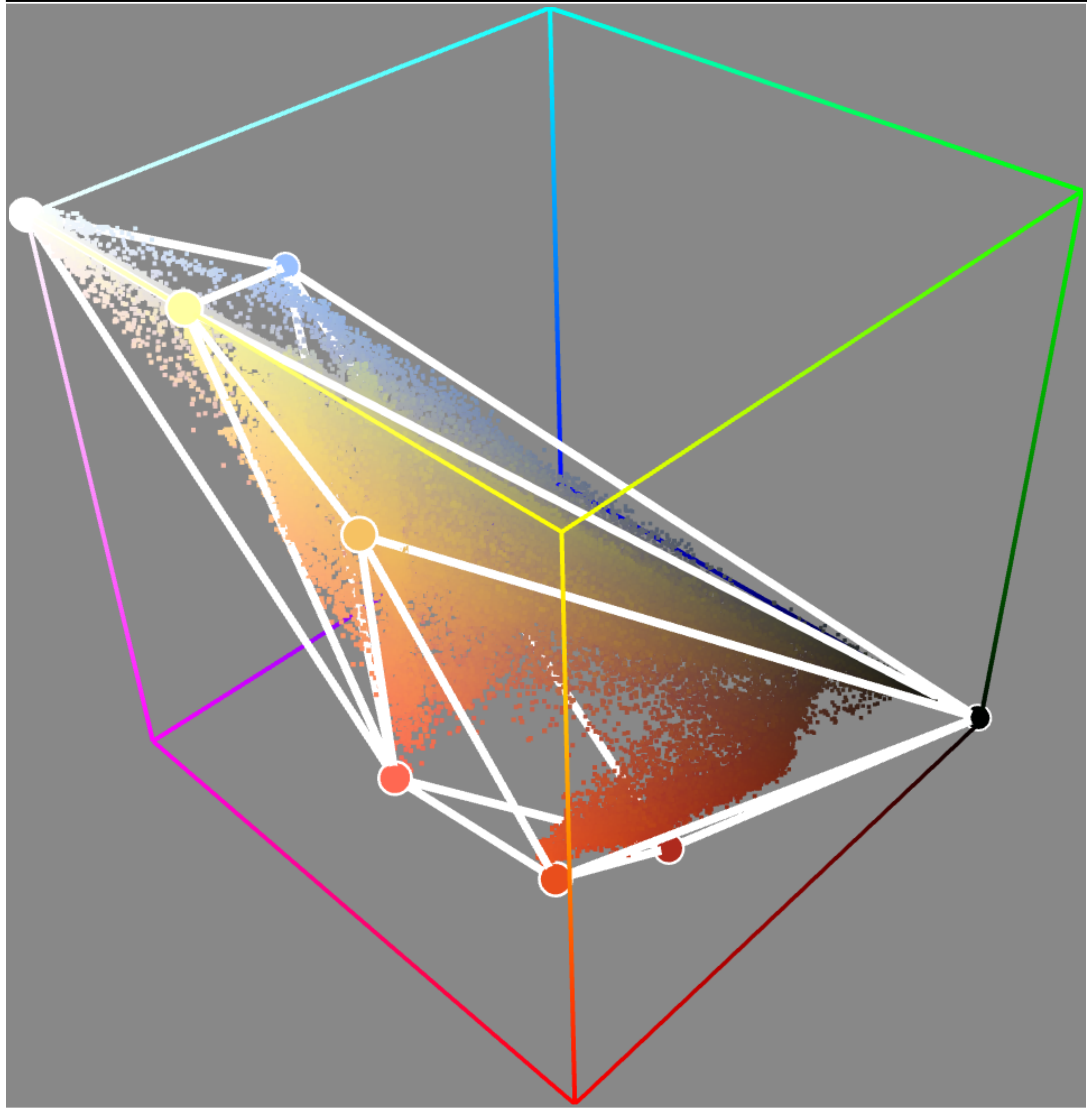
- The convex hull can be simplified to any complexity level.



Palette Size

- The convex hull can be simplified to any complexity level.

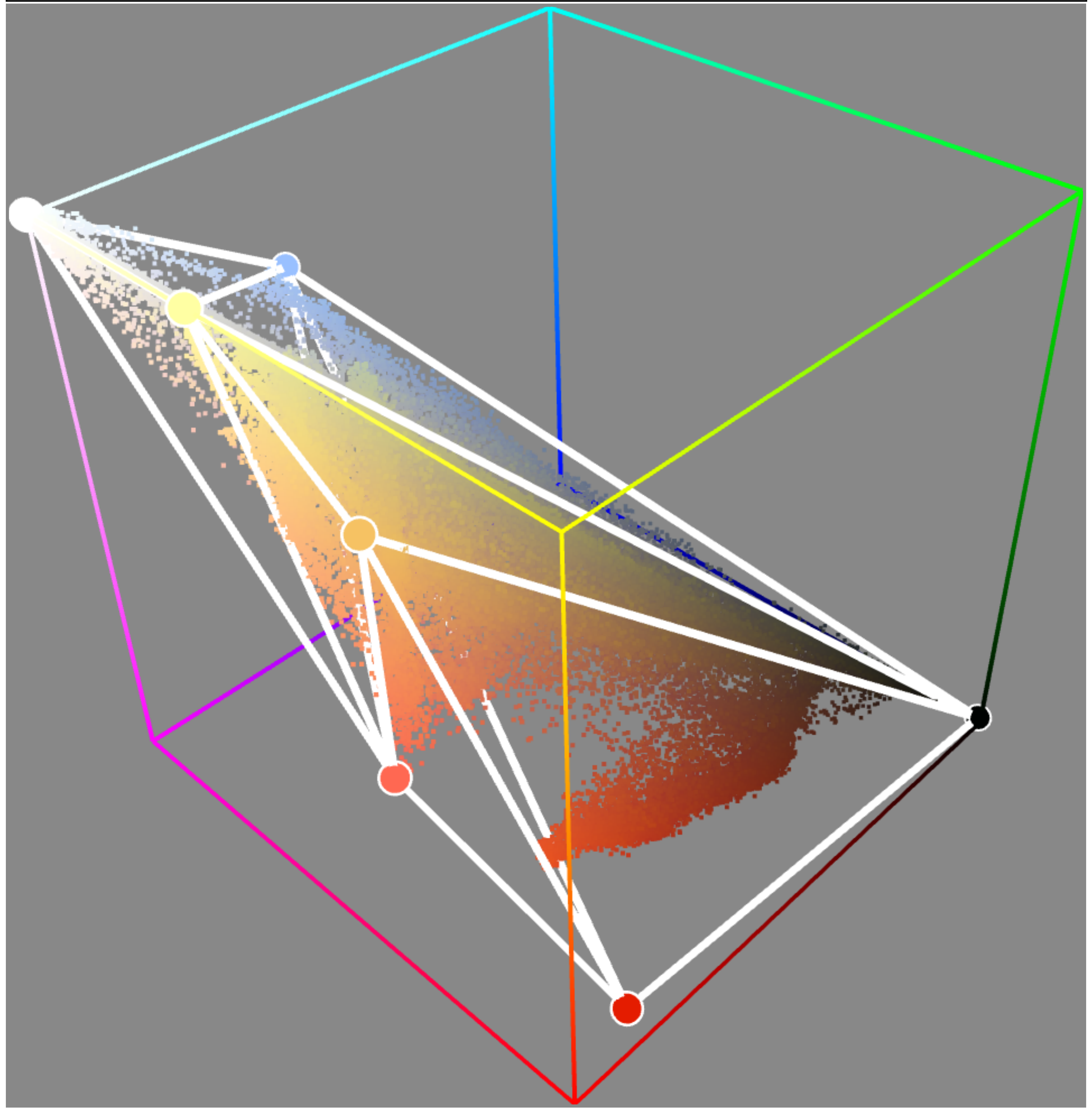
8 vertices



Palette Size

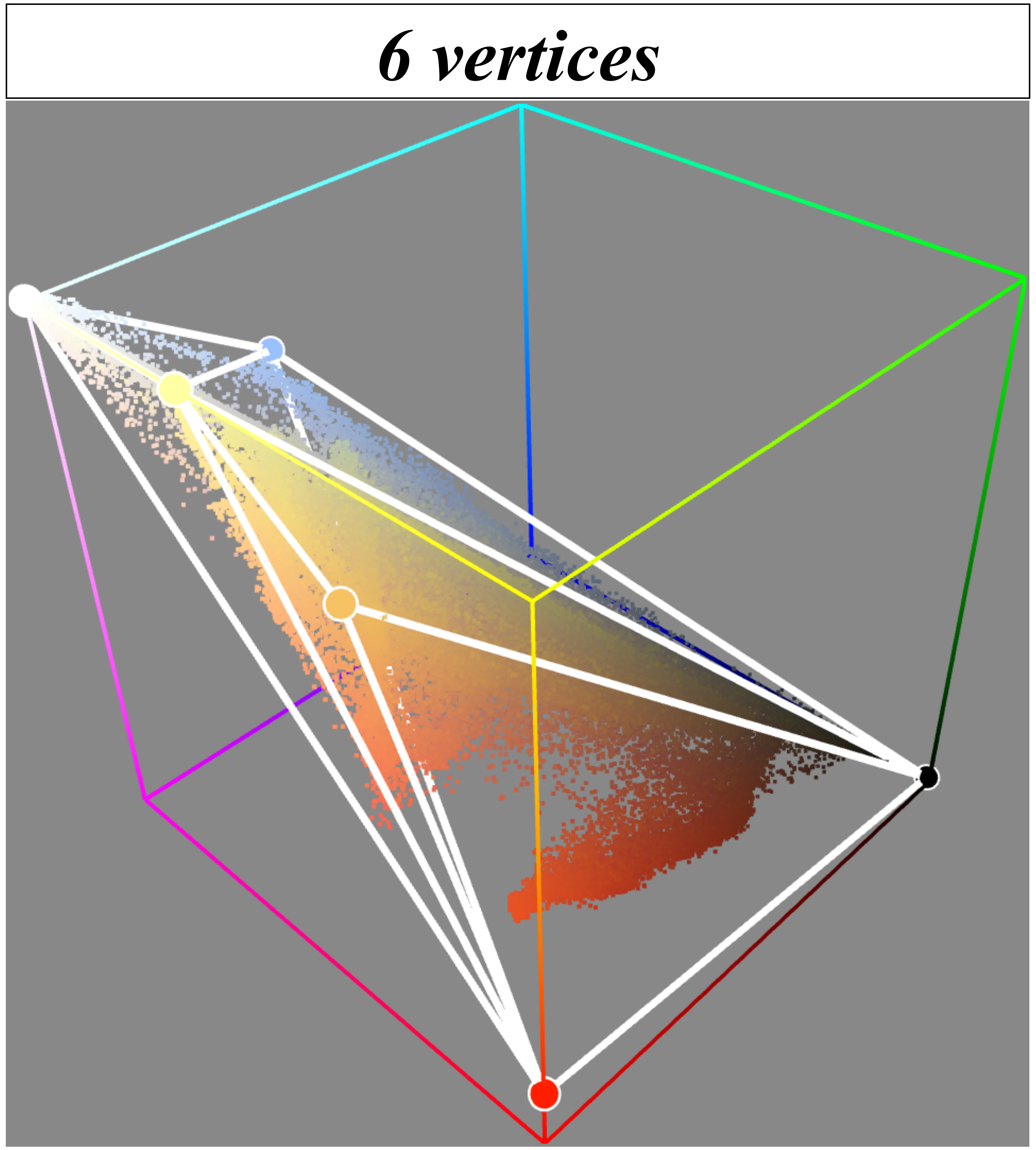
- The convex hull can be simplified to any complexity level.

7 vertices



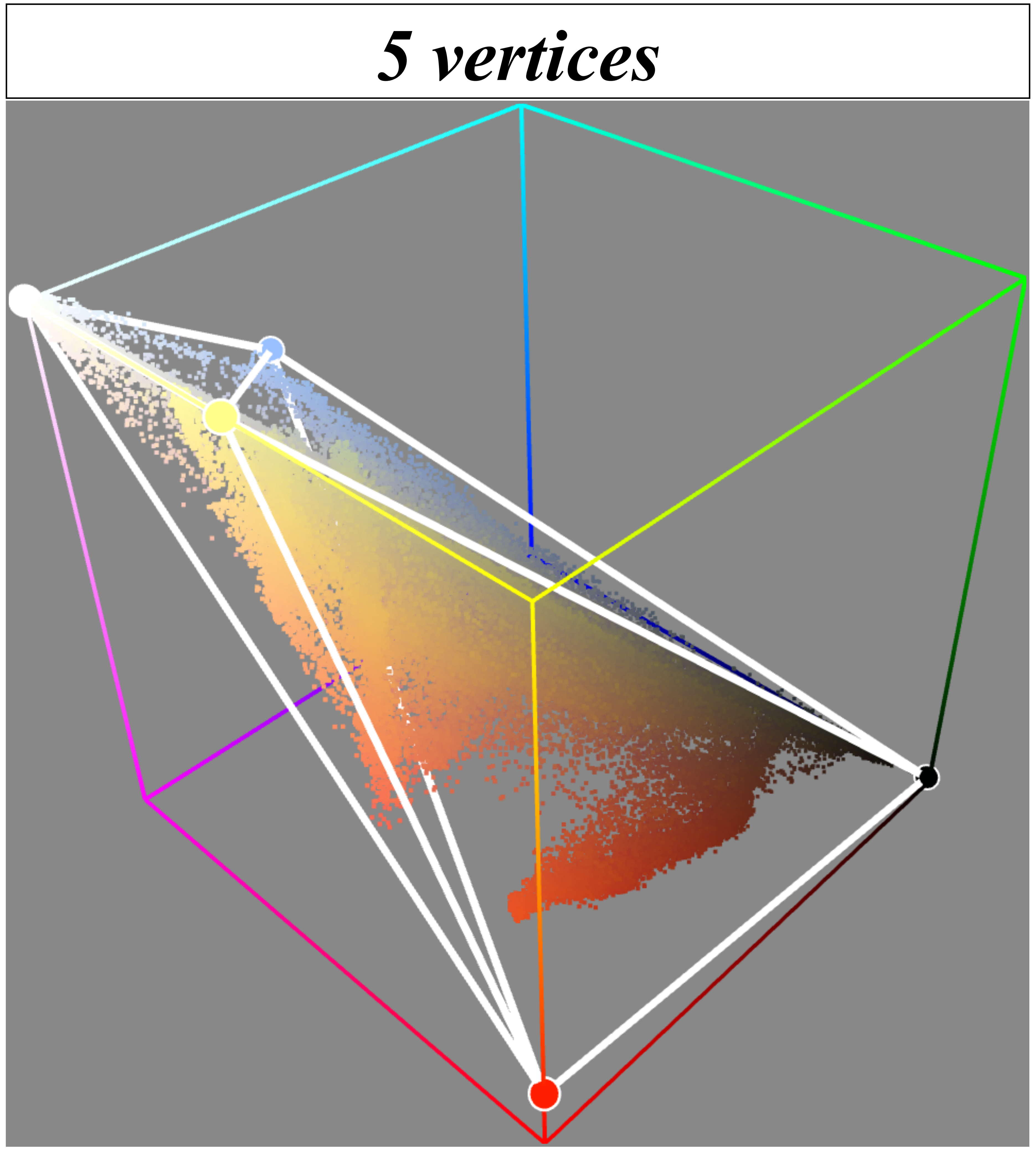
Palette Size

- The convex hull can be simplified to any complexity level.



Palette Size

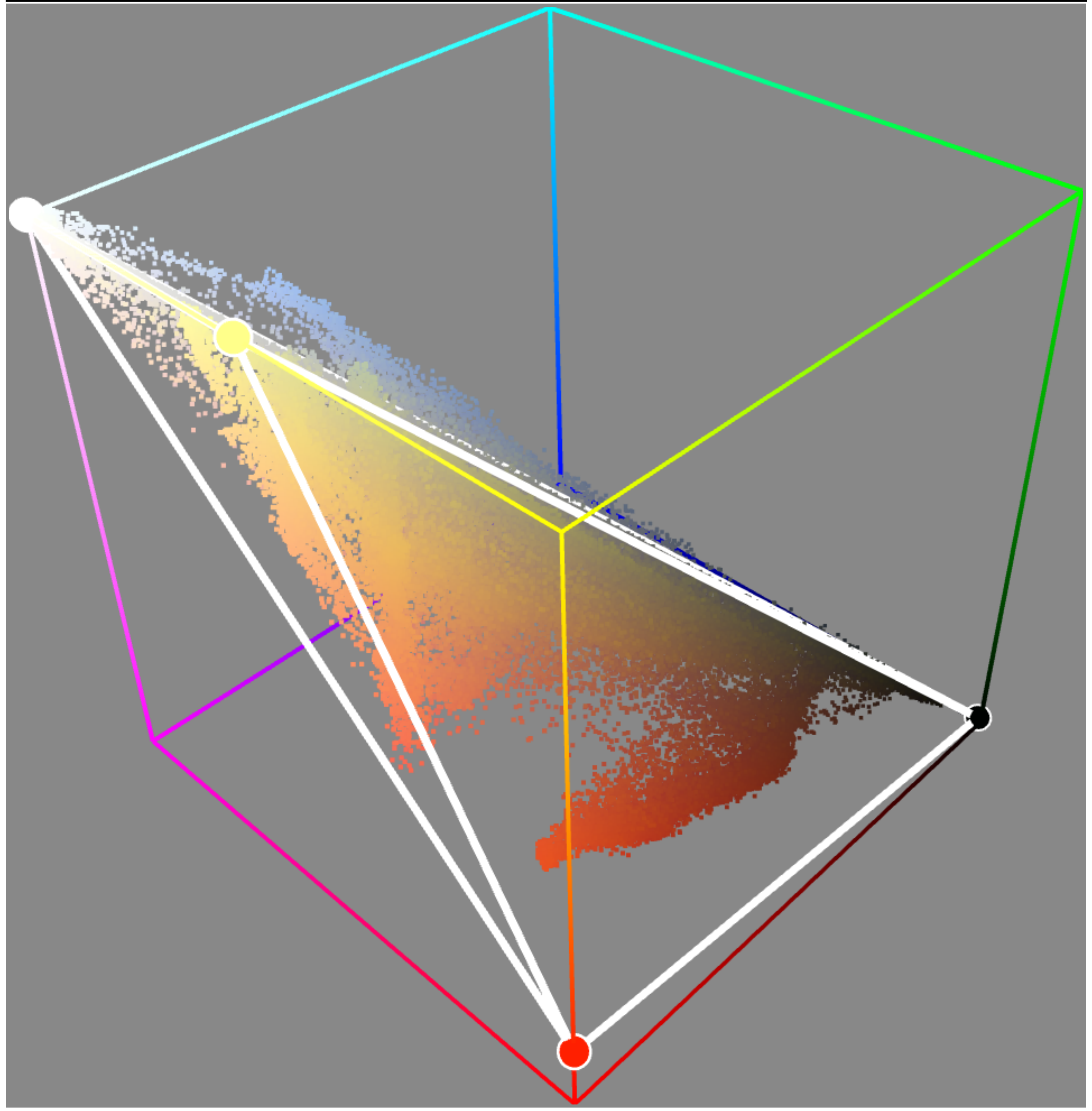
- The convex hull can be simplified to any complexity level.



Palette Size

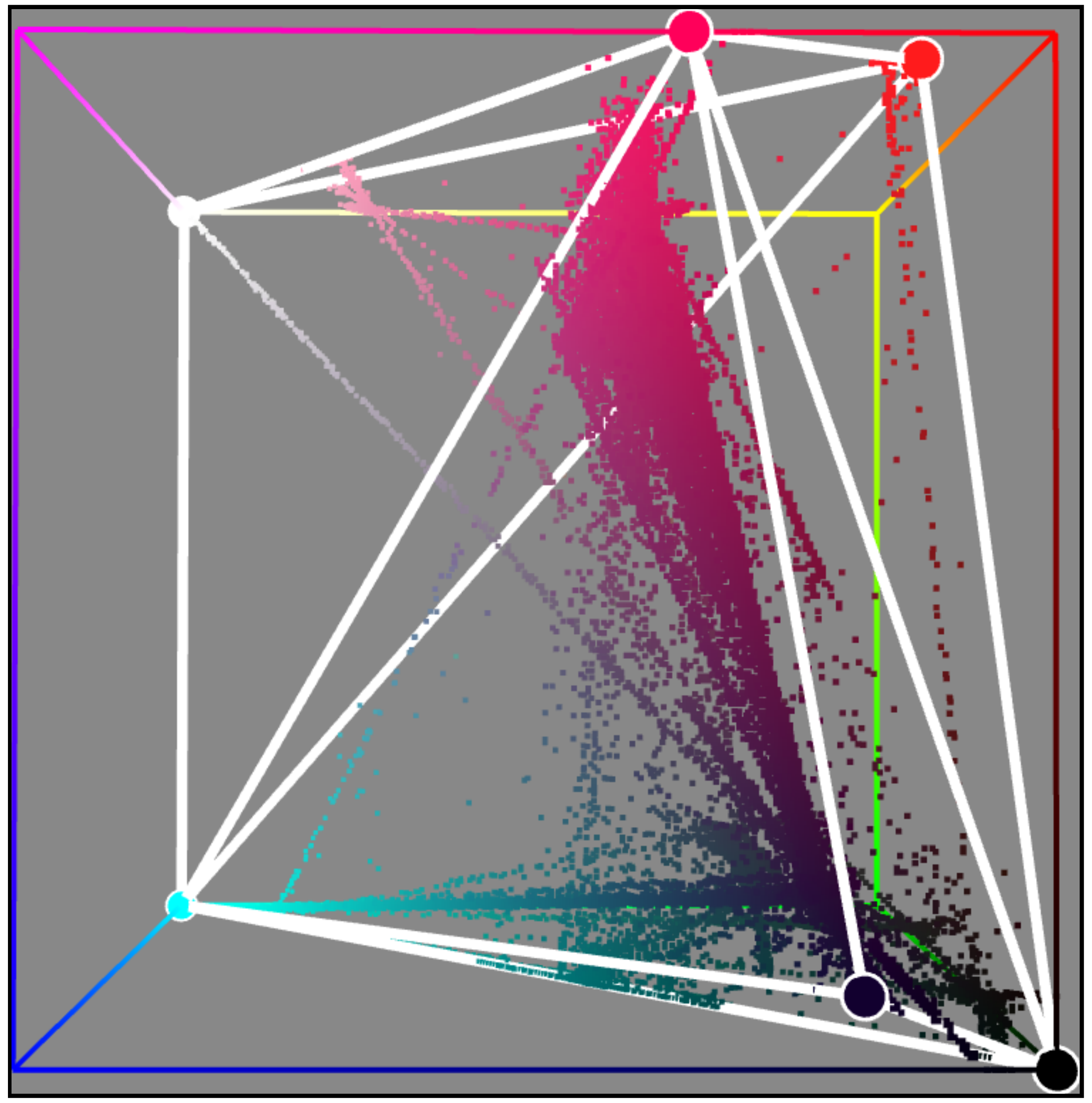
- The convex hull can be simplified to any complexity level.

4 vertices

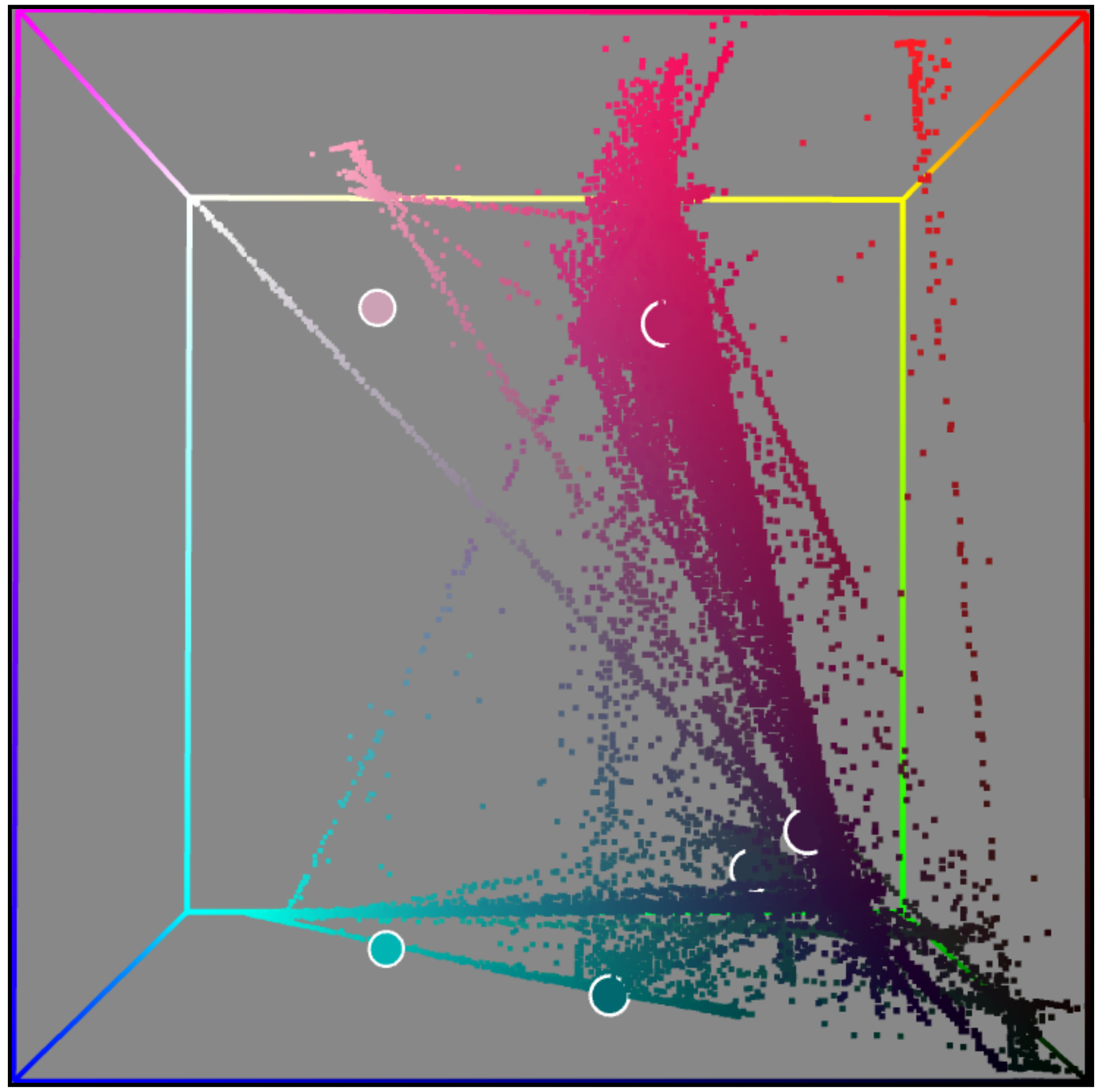


Compared to Clustering

Input



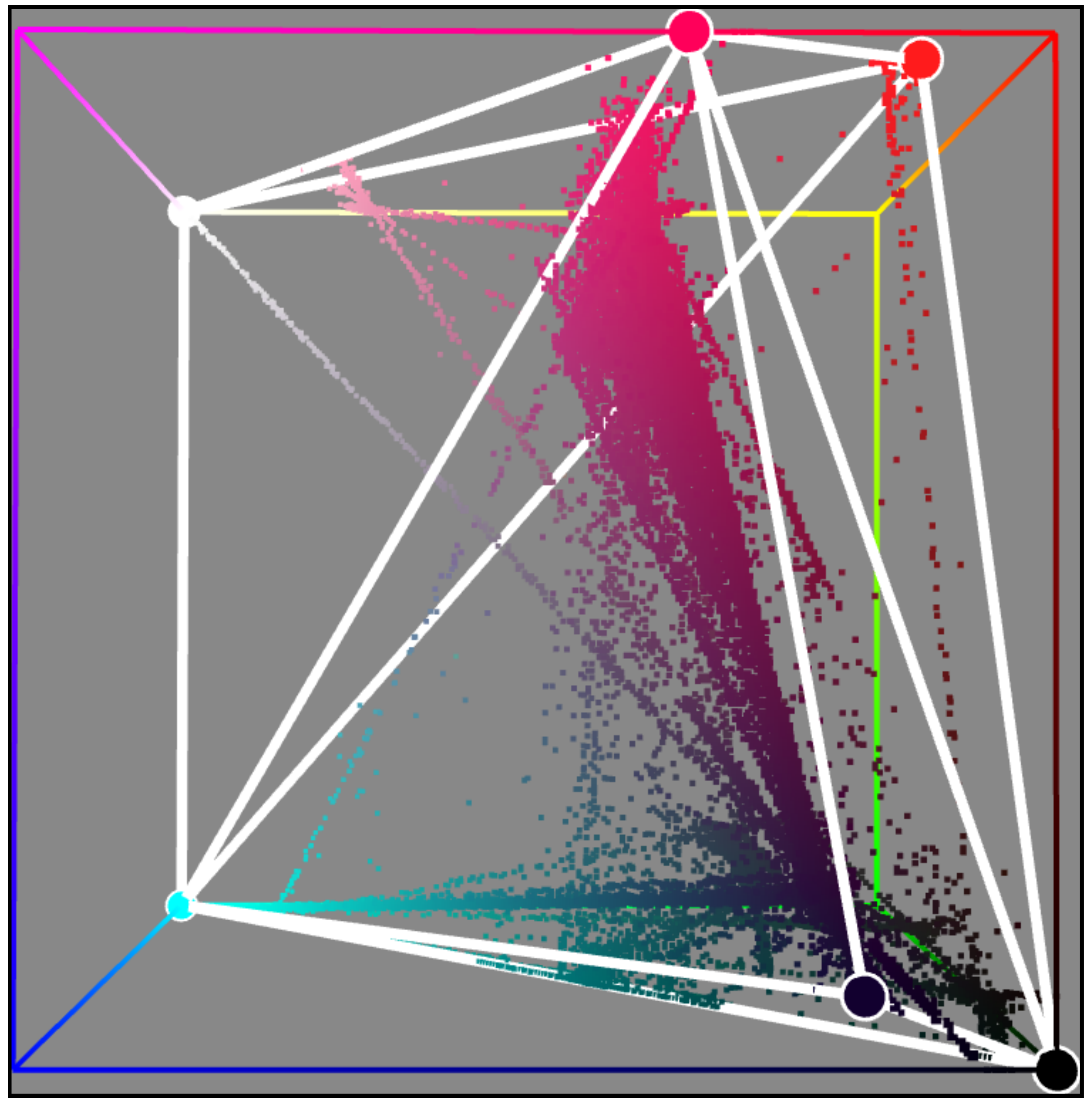
Ours



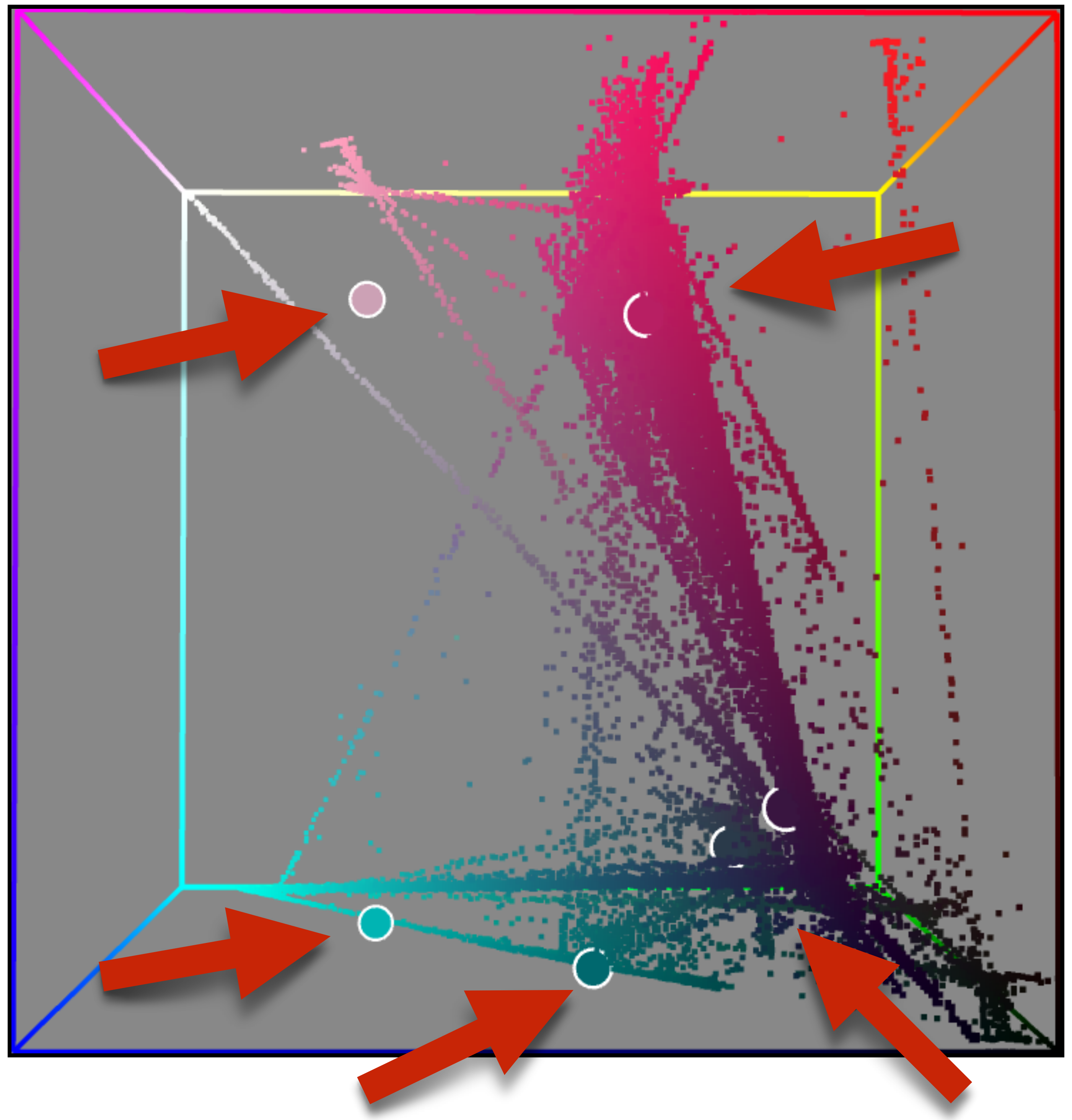
Chang et al. 2015

Compared to Clustering

Input



Ours



Chang et al. 2015

Layer Opacity

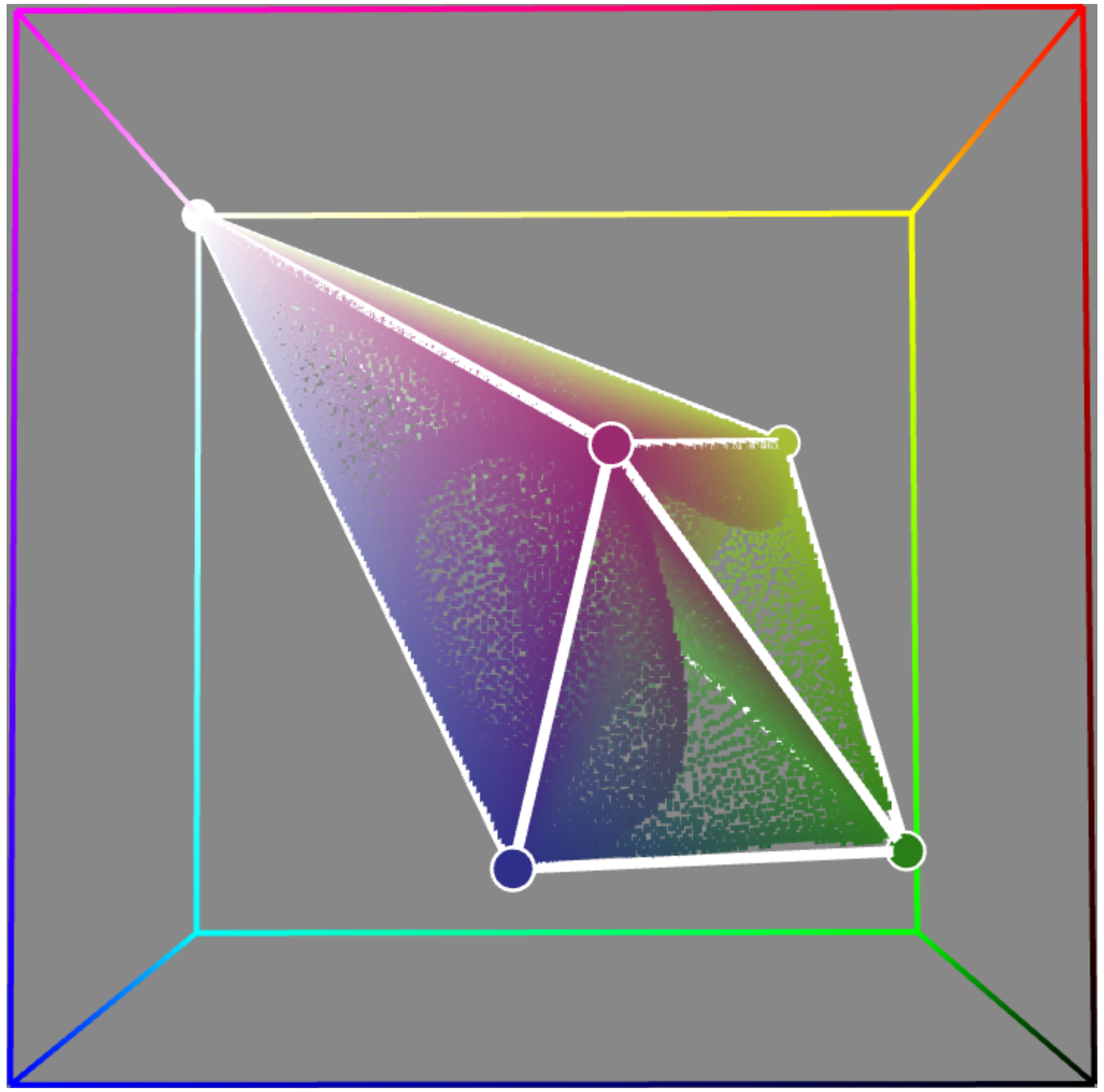
Layer Order

input

Layer Order

input

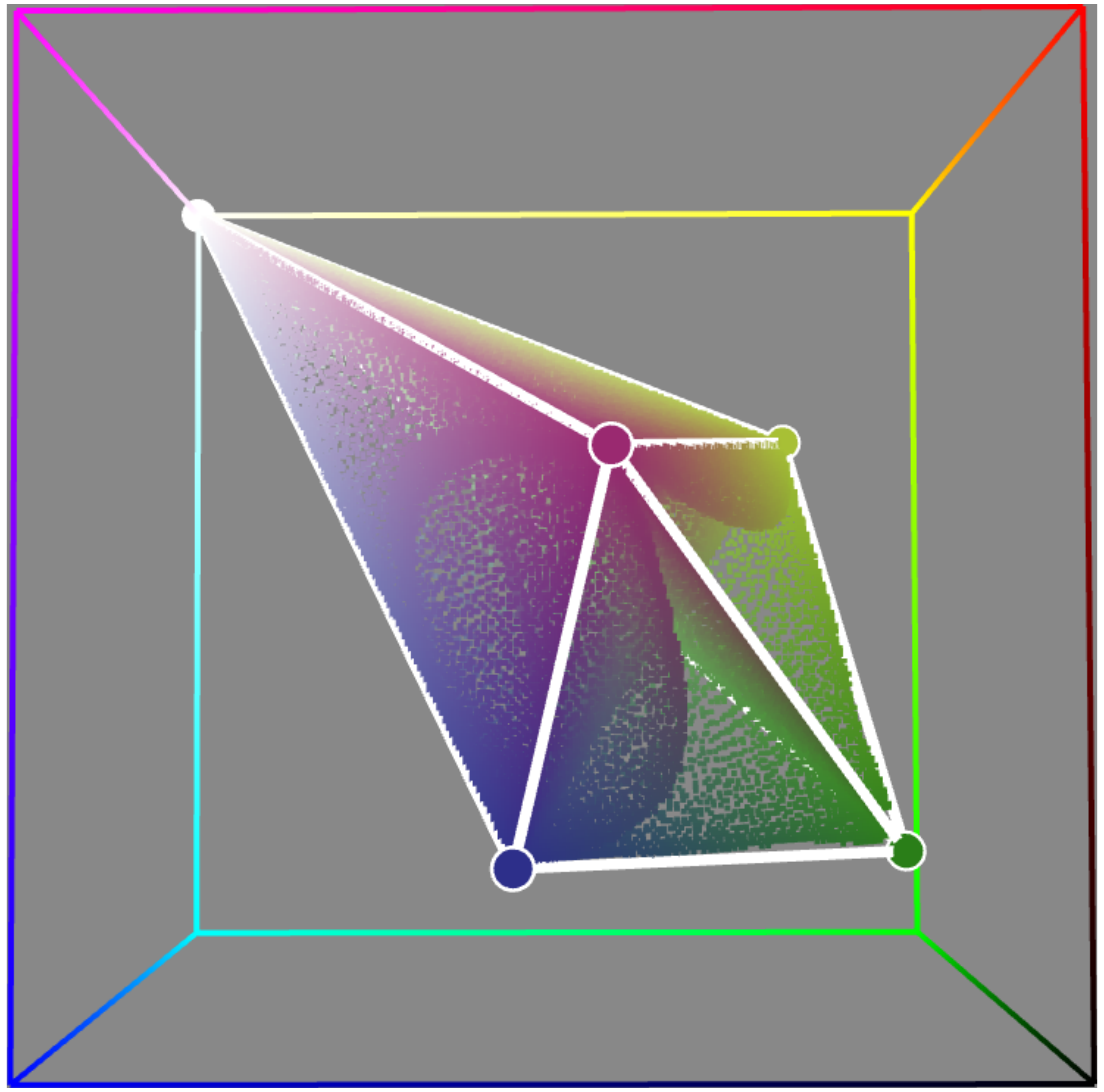
palette selection



Layer Order

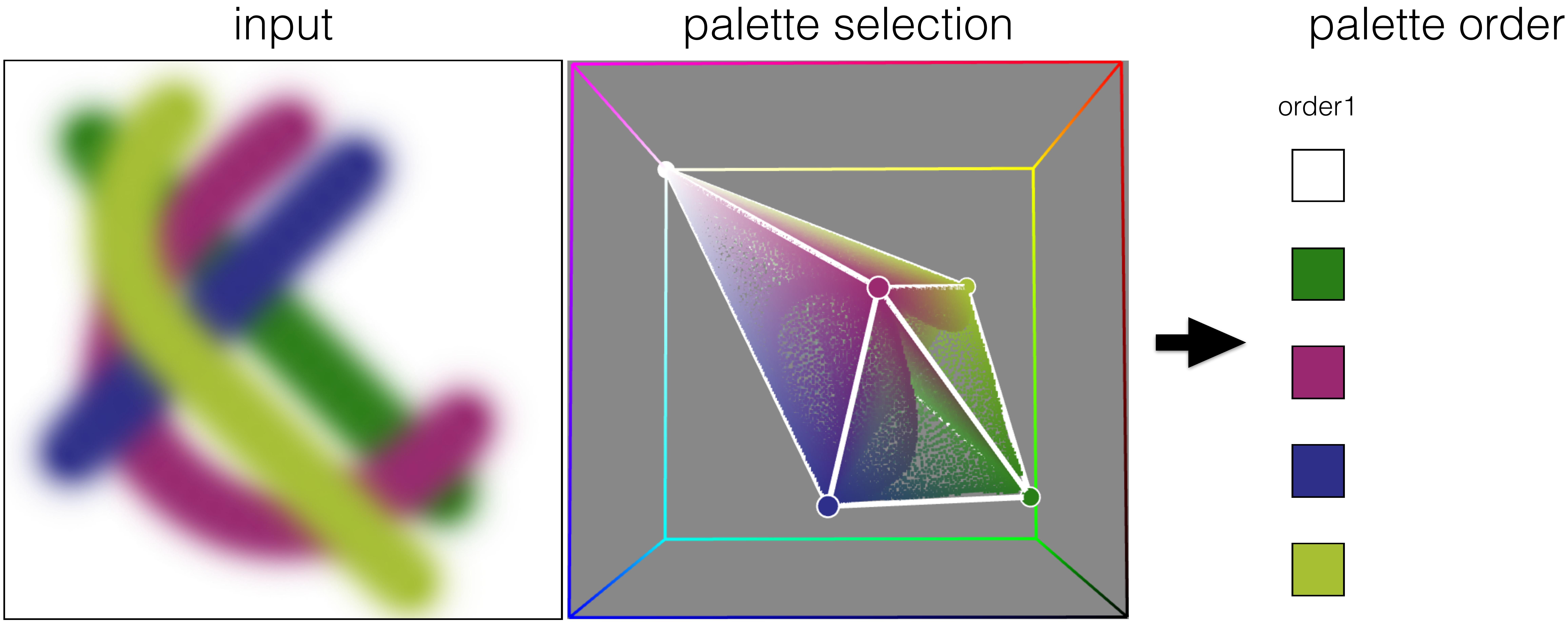
input

palette selection



palette order

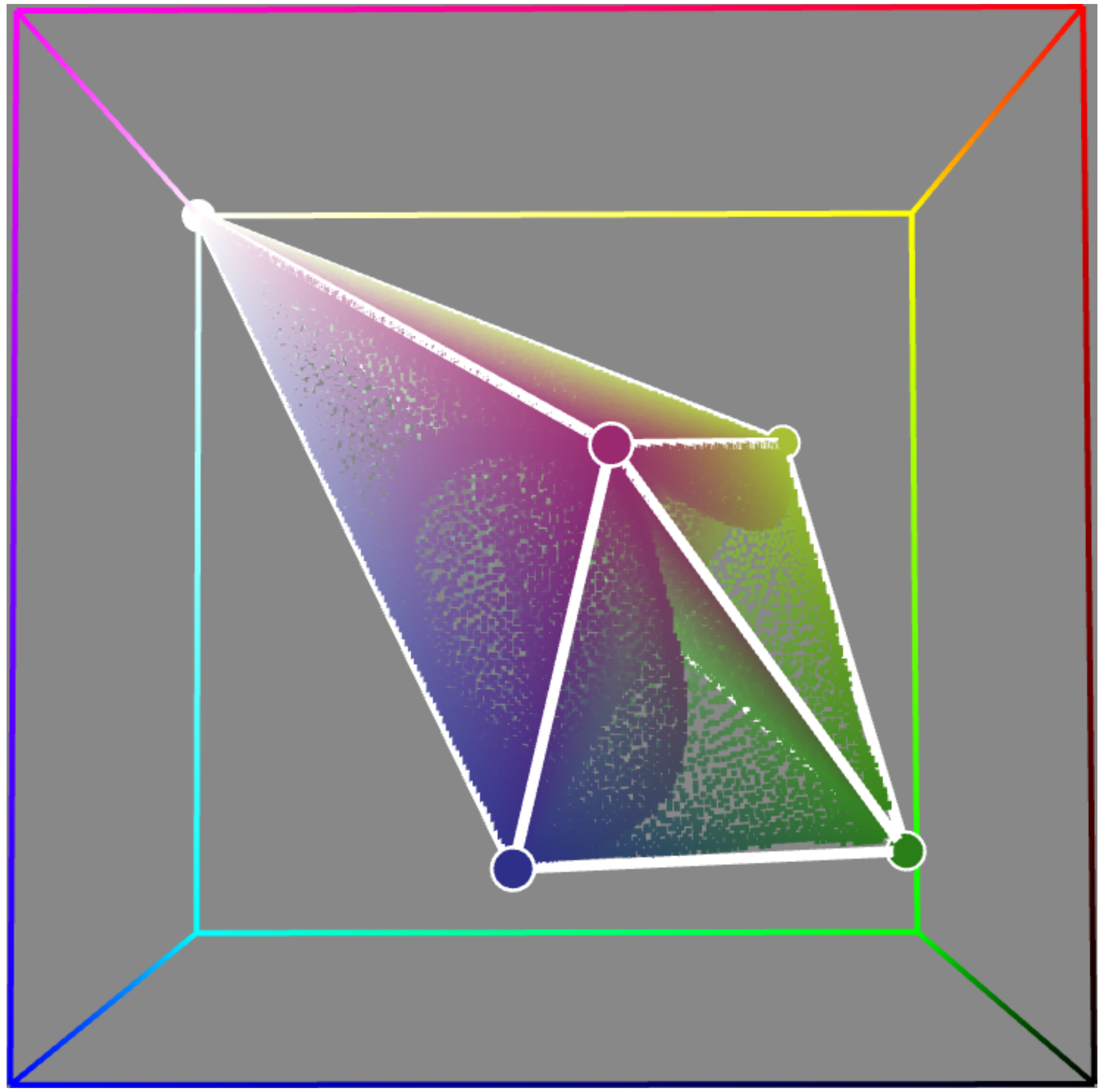
Layer Order



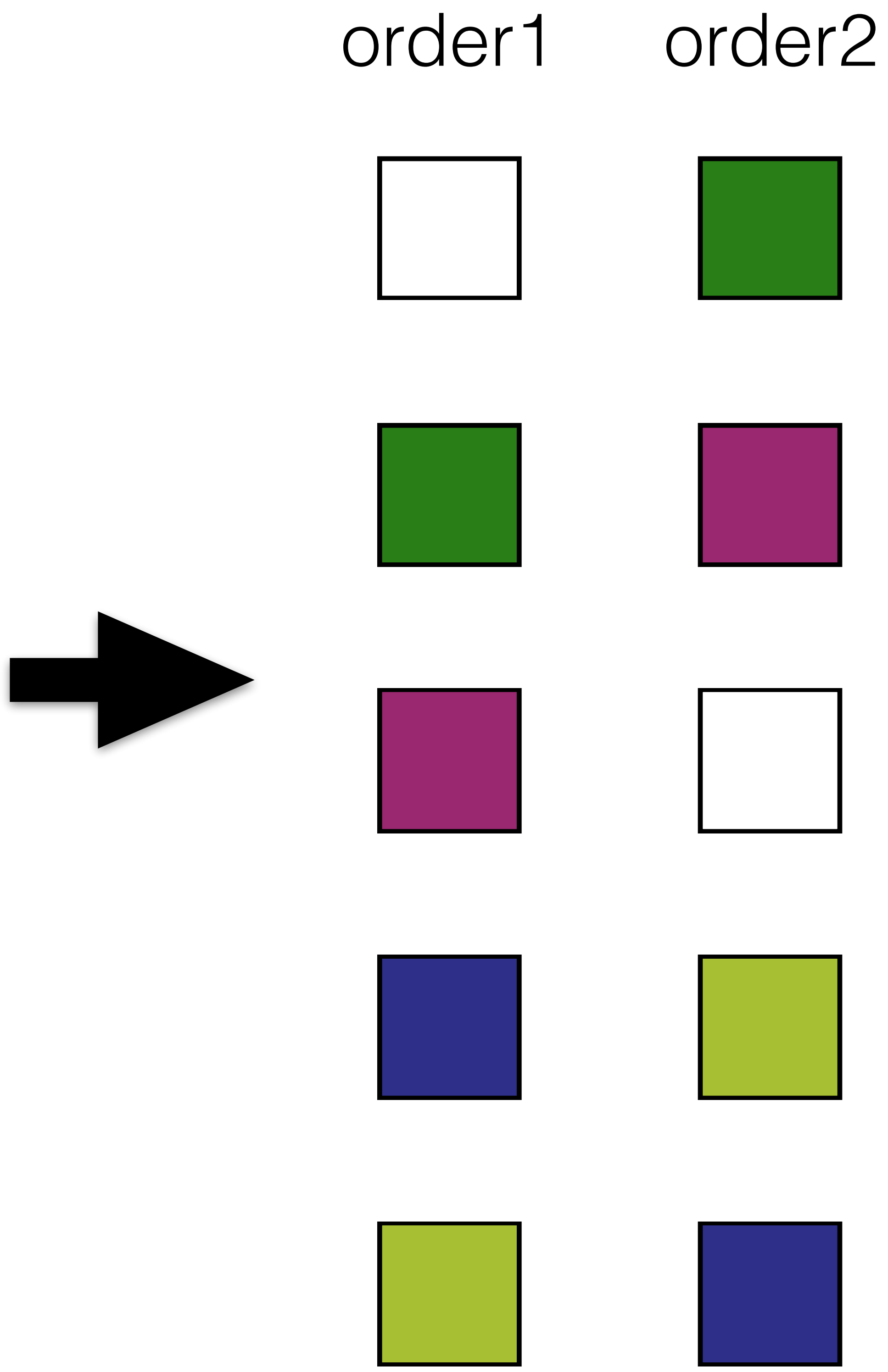
Layer Order

input

palette selection



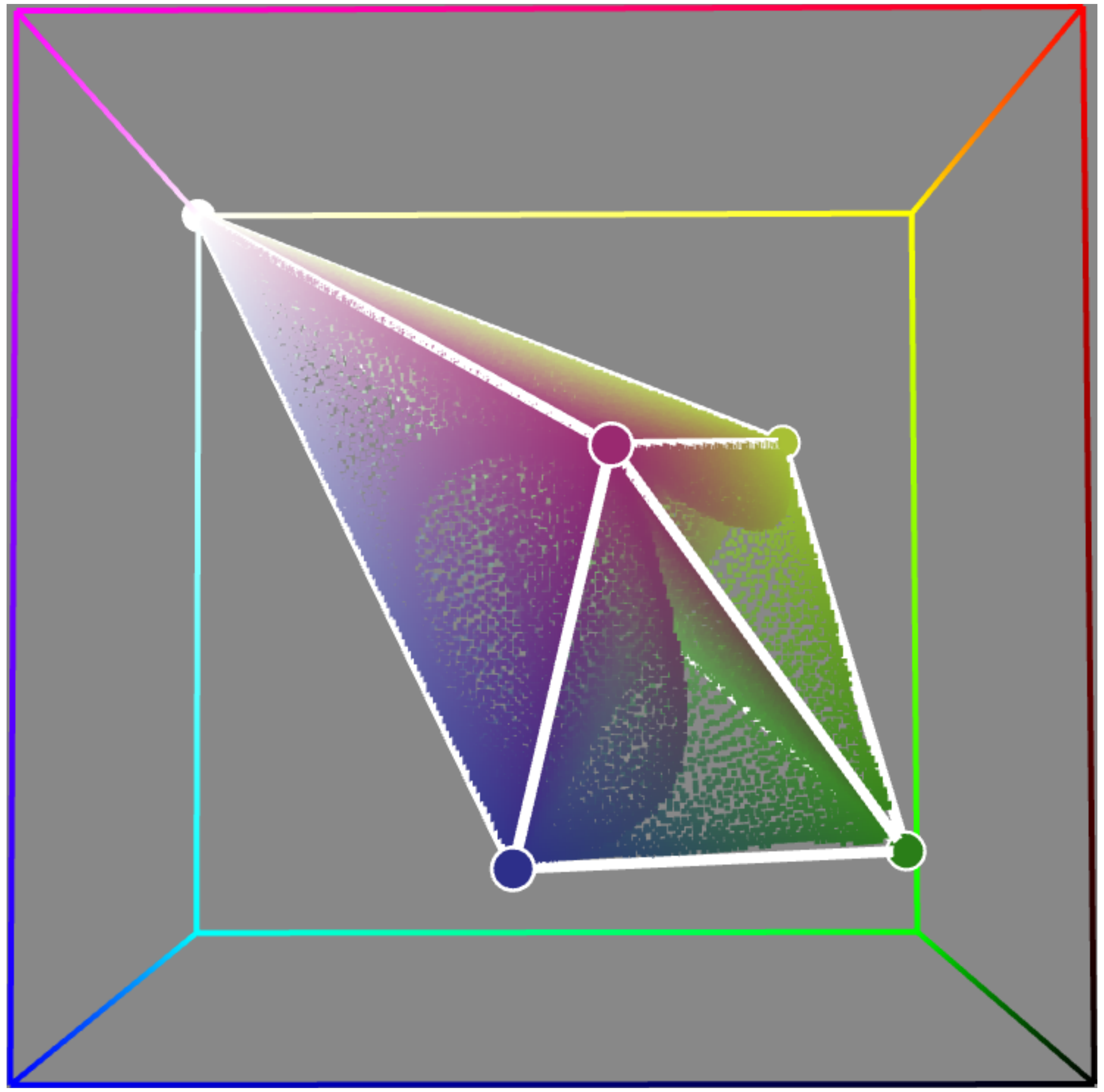
palette order



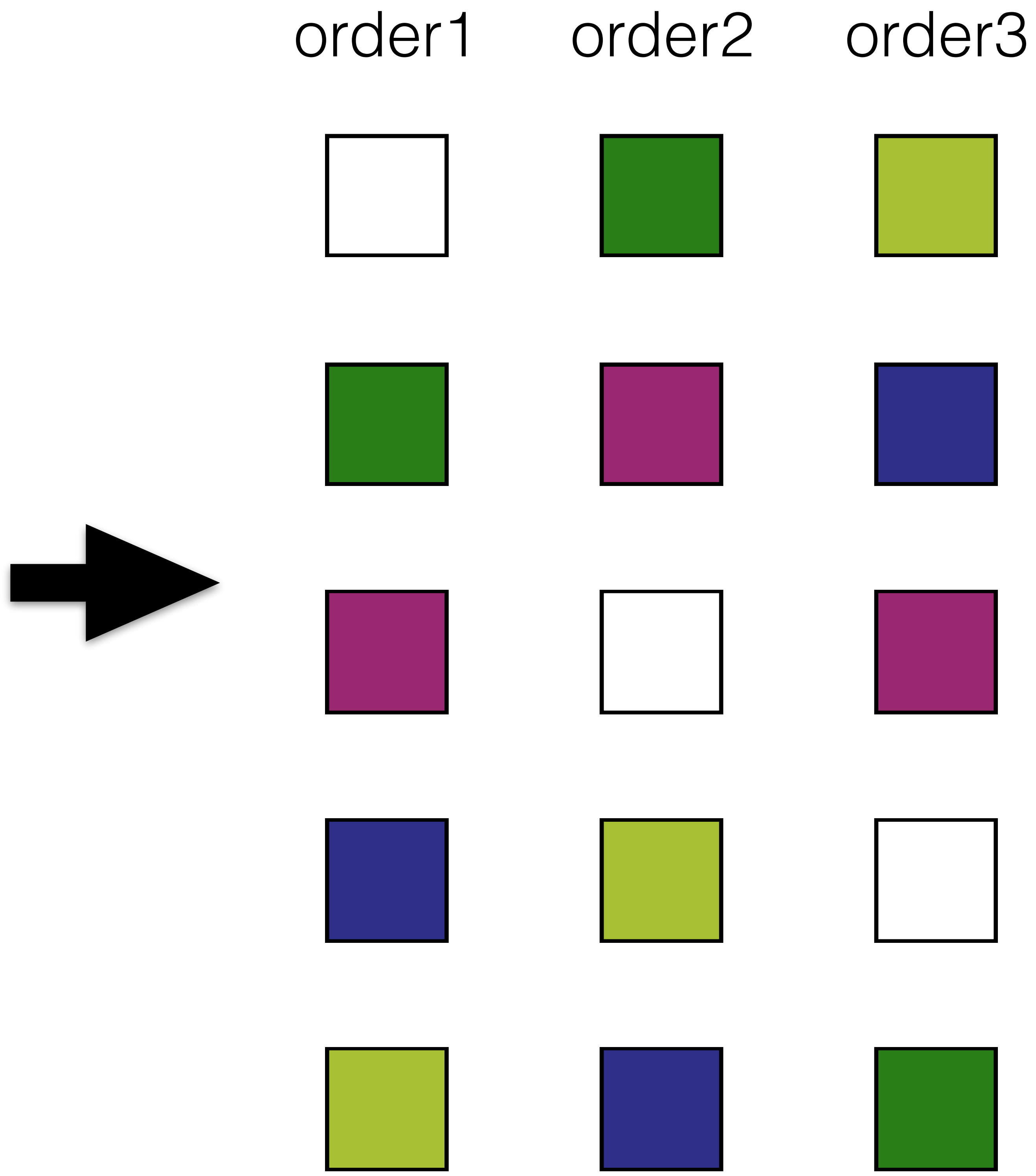
Layer Order

input

palette selection



palette order

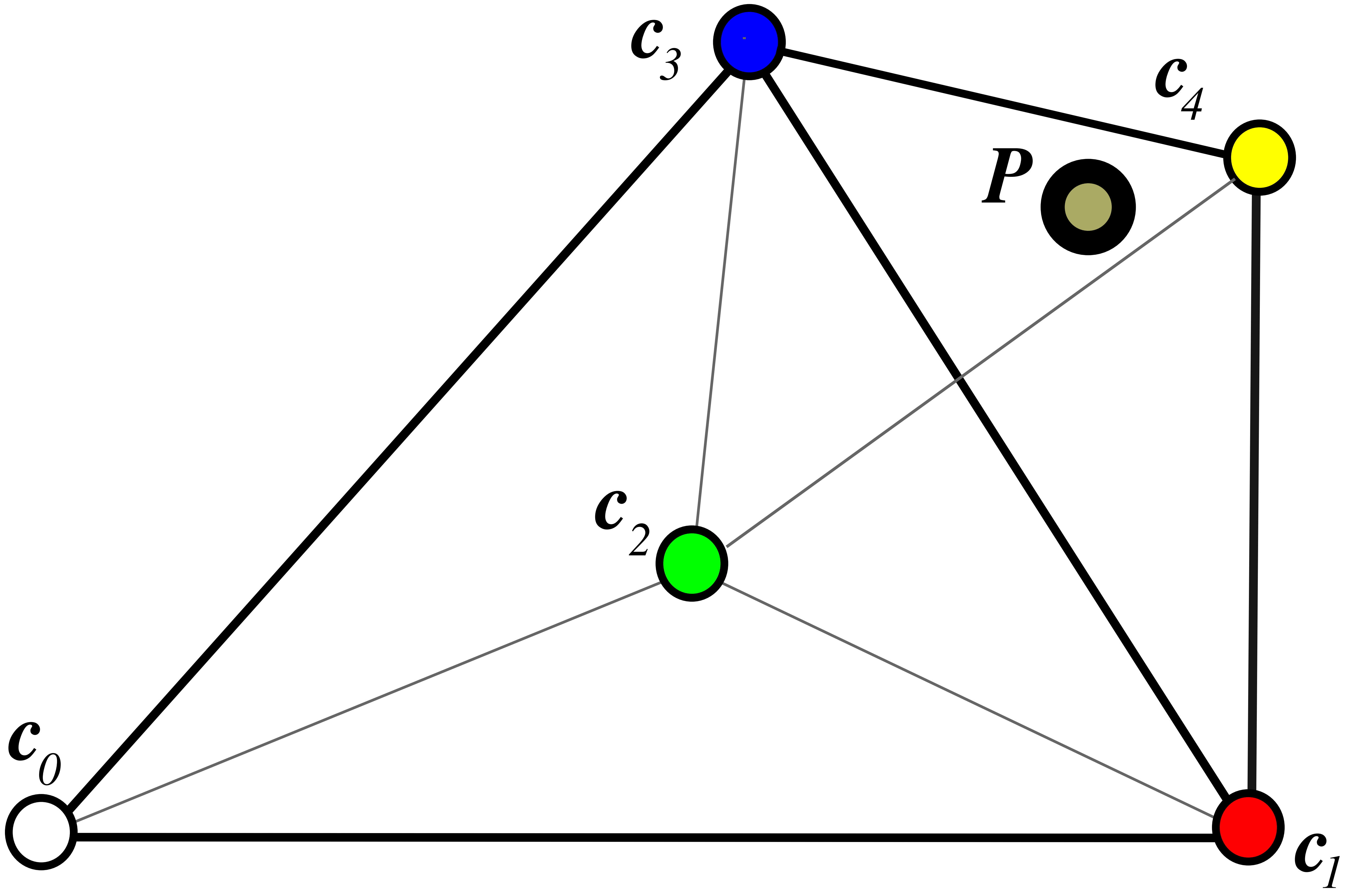


Layer Order


input
palette selection
palette order
order1
order2
order3
...

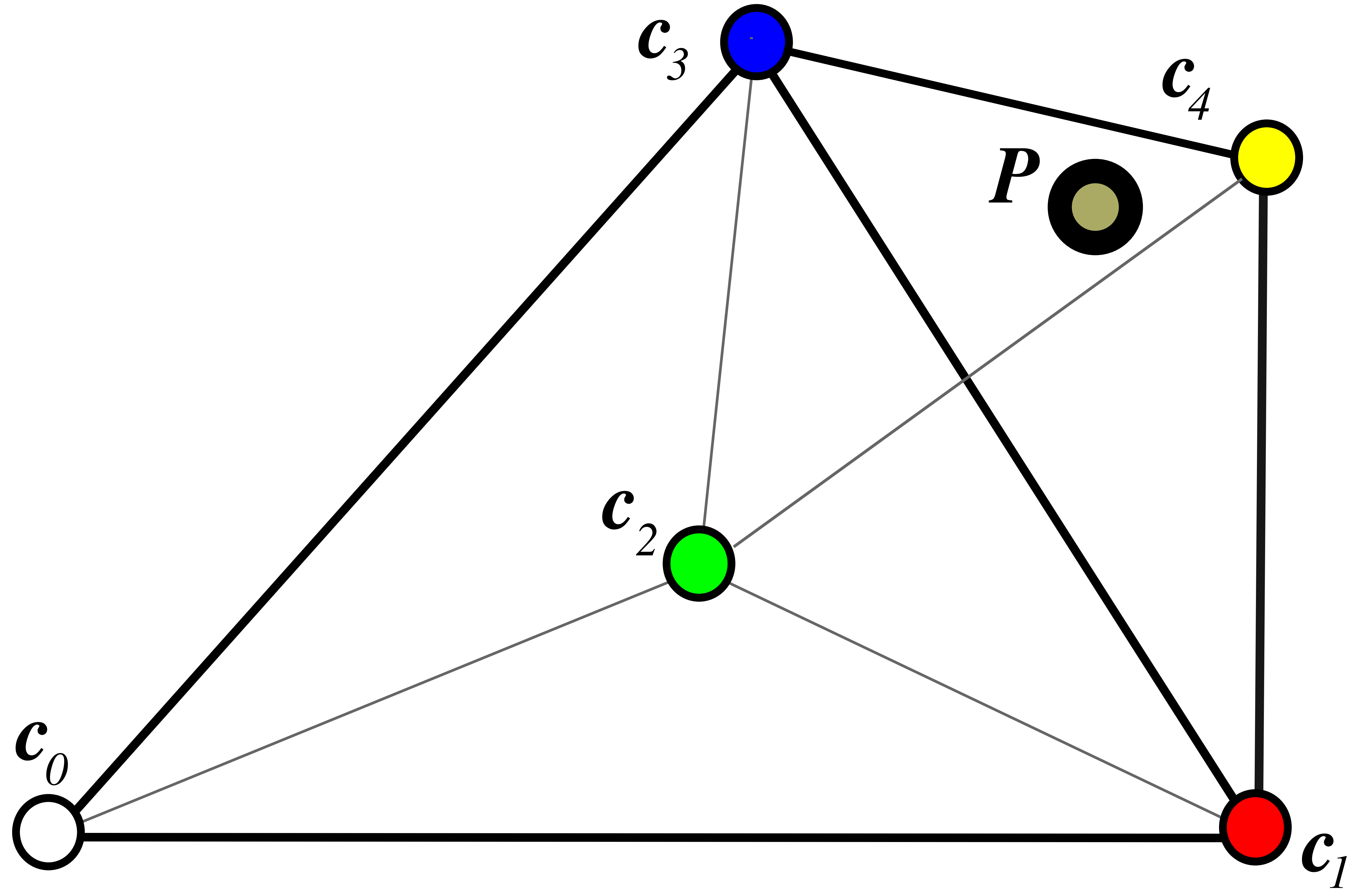
Layer Order

- Alpha compositing is not commutative



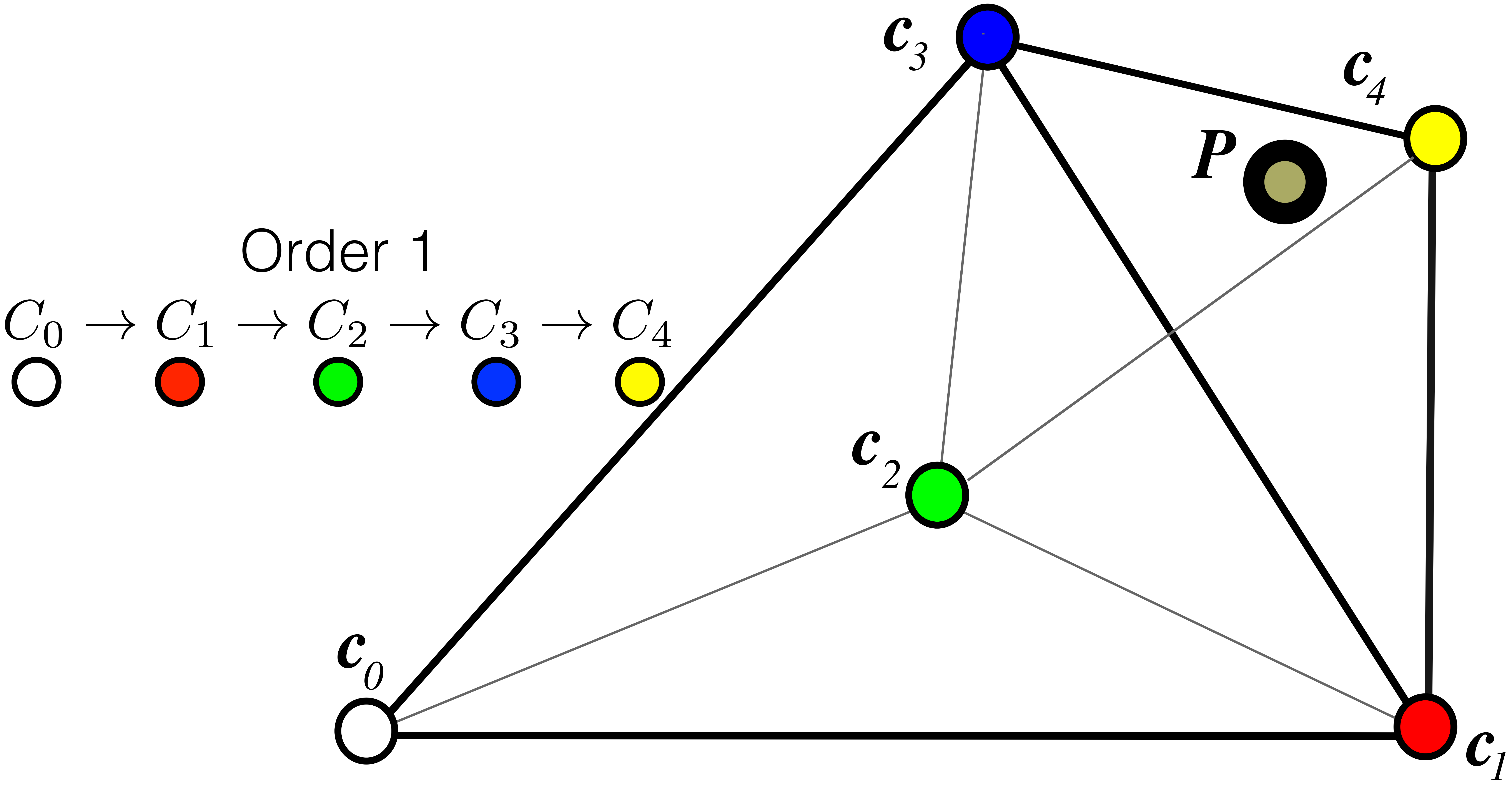
Layer Order

- Alpha compositing is not commutative
- For **n** layers, there are **n!** orderings



Layer Order

- Alpha compositing is not commutative
- For **n** layers, there are **n!** orderings

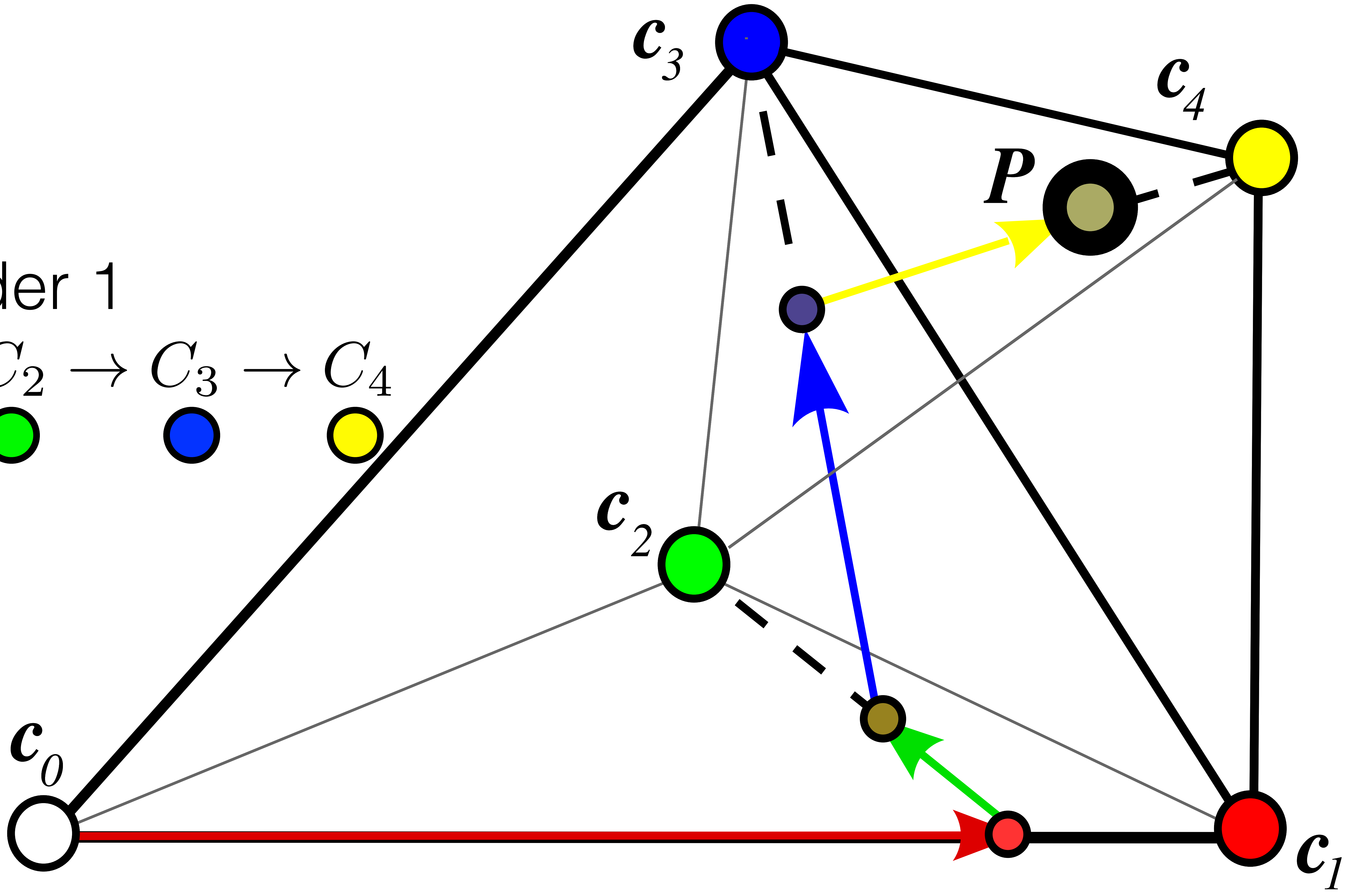


Layer Order

- Alpha compositing is not commutative
- For **n** layers, there are **n!** orderings

Order 1

$$C_0 \to C_1 \to C_2 \to C_3 \to C_4$$

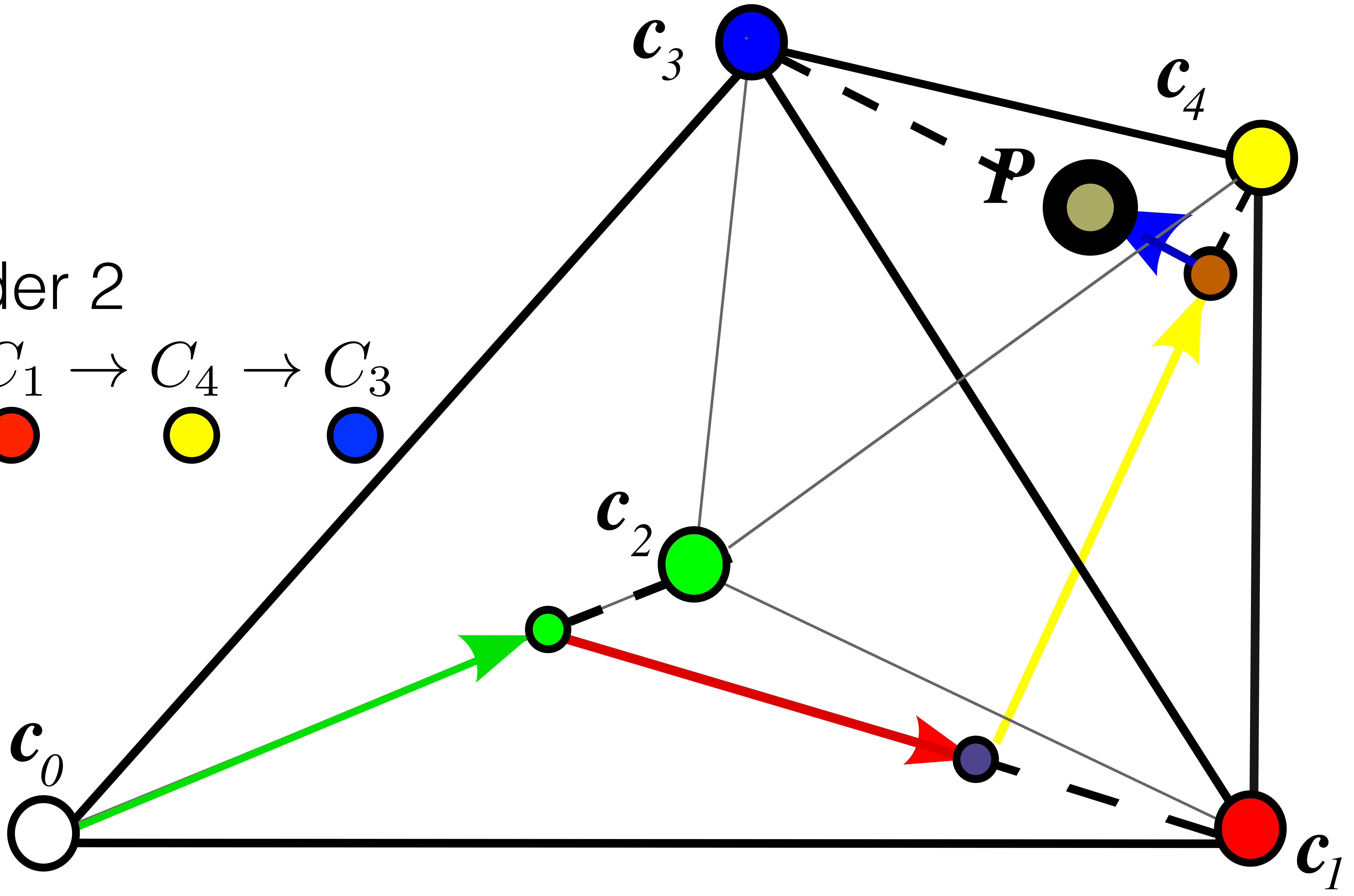


Layer Order

- Alpha compositing is not commutative
- For **n** layers, there are **n!** orderings

Order 2

$C_0 \to C_2 \to C_1 \to C_4 \to C_3$



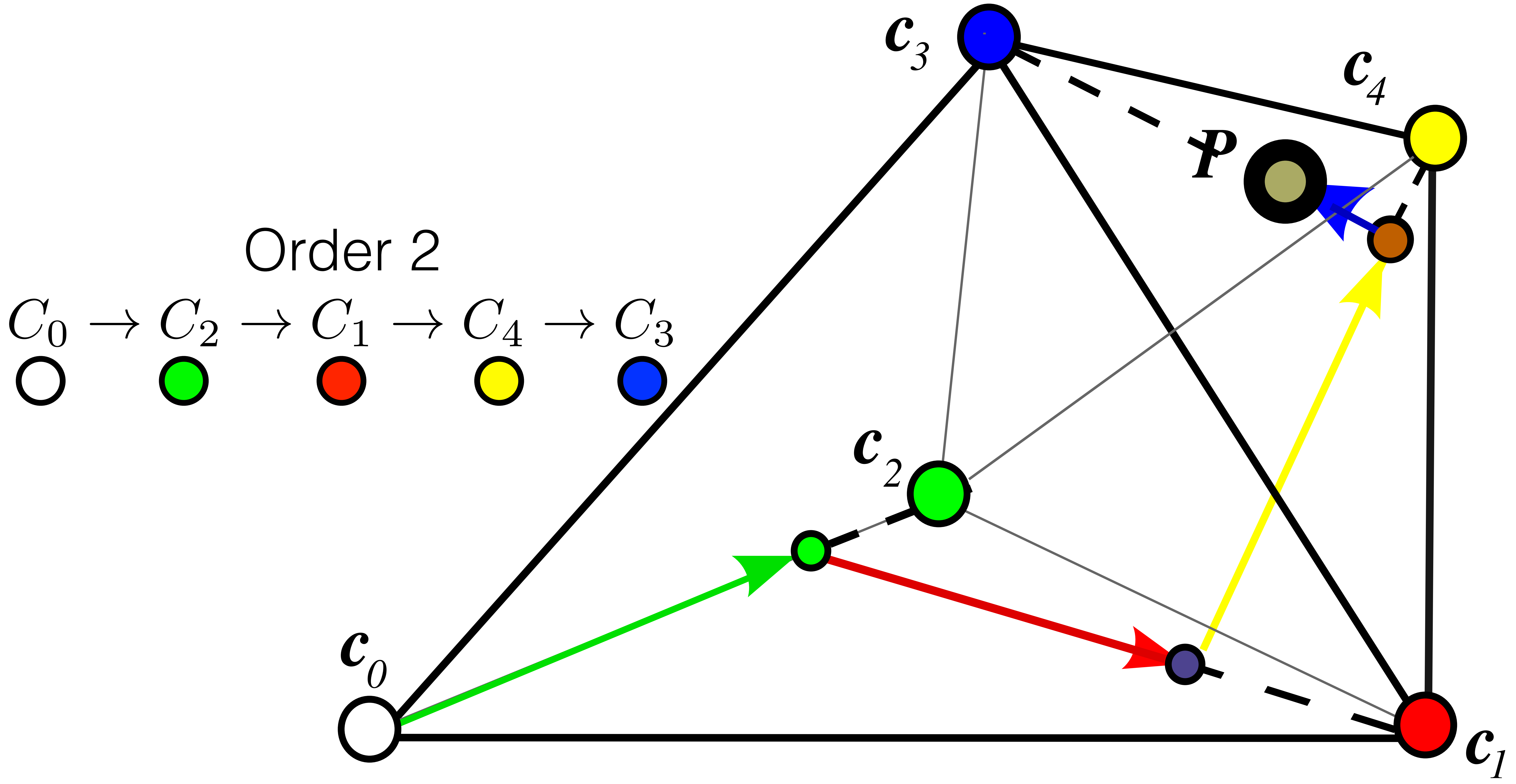
Layer Order

- Alpha compositing is not commutative
- For **n** layers, there are **n!** orderings

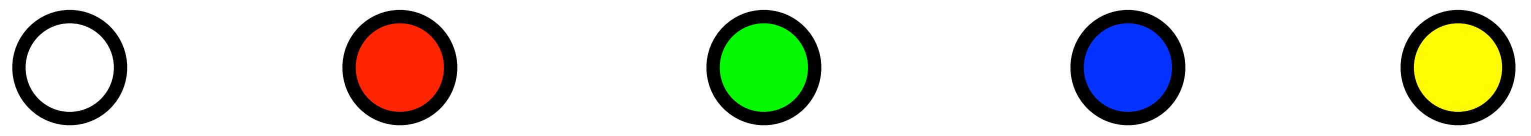
- **Hard to find good metric**.
- **User manually chooses.**

Order 2

$C_0 \to C_2 \to C_1 \to C_4 \to C_3$



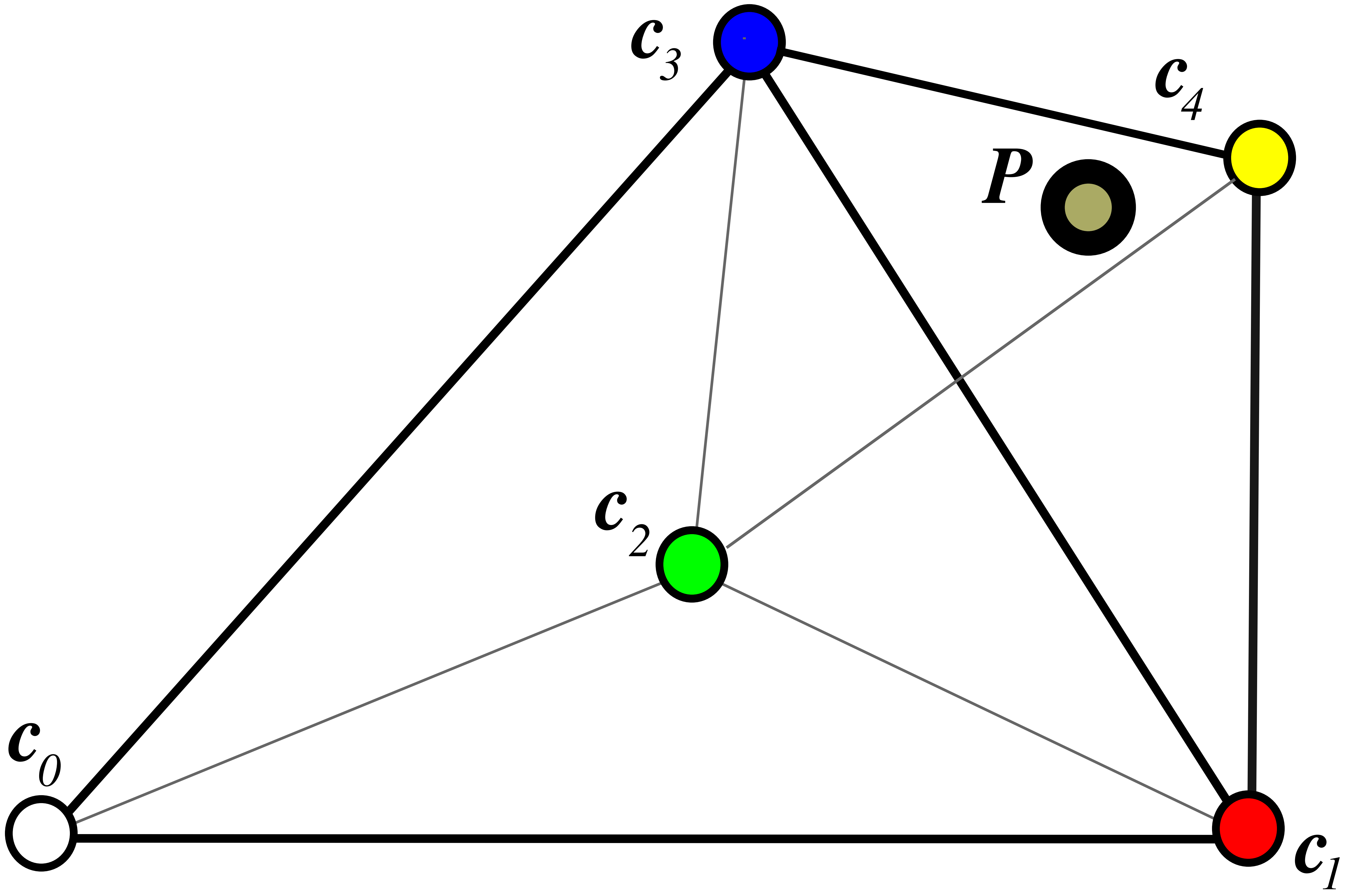
Color Compositing Path



- After user chooses a layer order: $C_0 \to C_1 \to C_2 \to C_3 \to C_4$
- Still have **infinite** paths from C_0 to P

Color Compositing Path

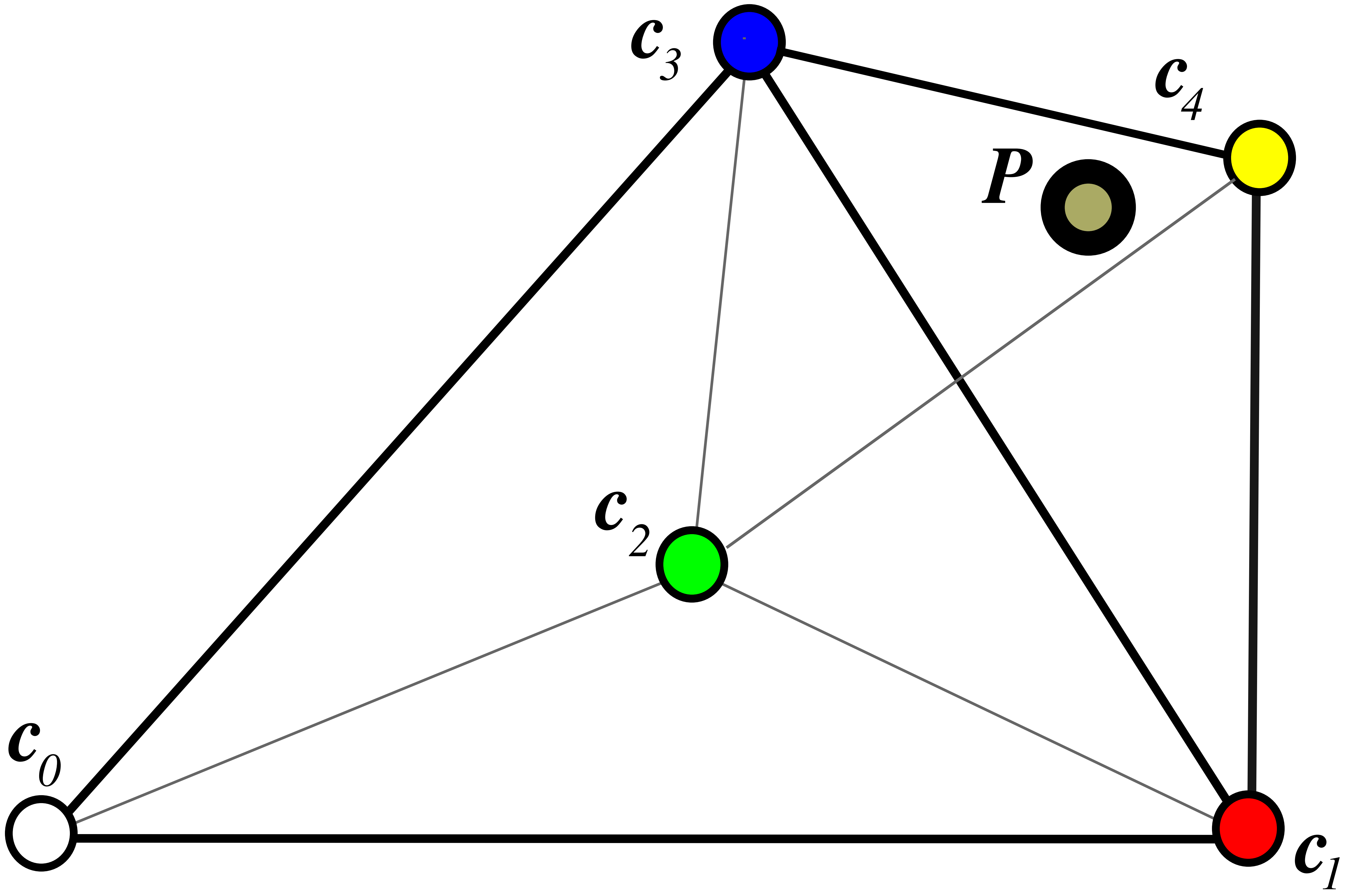
- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P



Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P

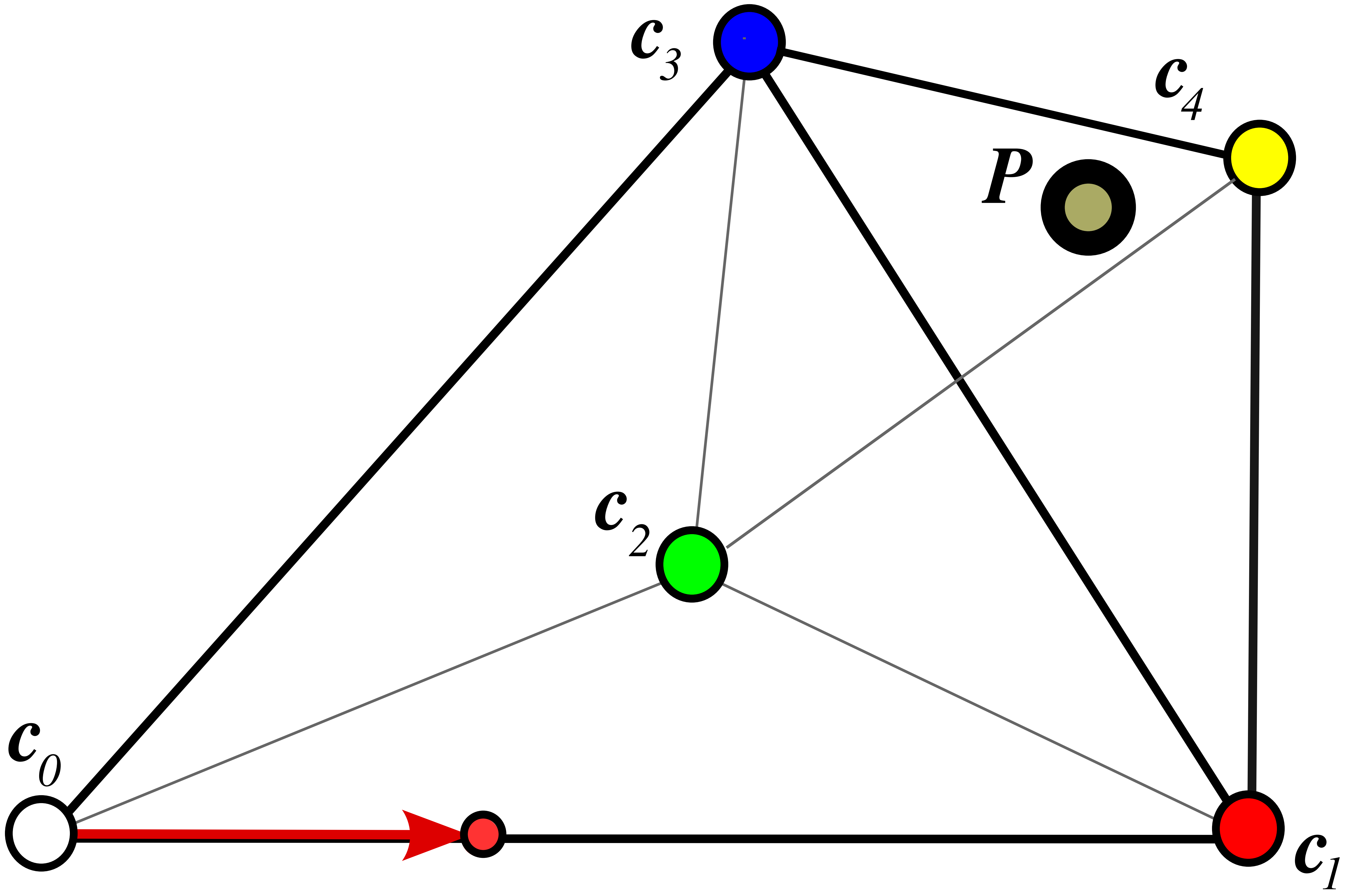
Path 1



Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P

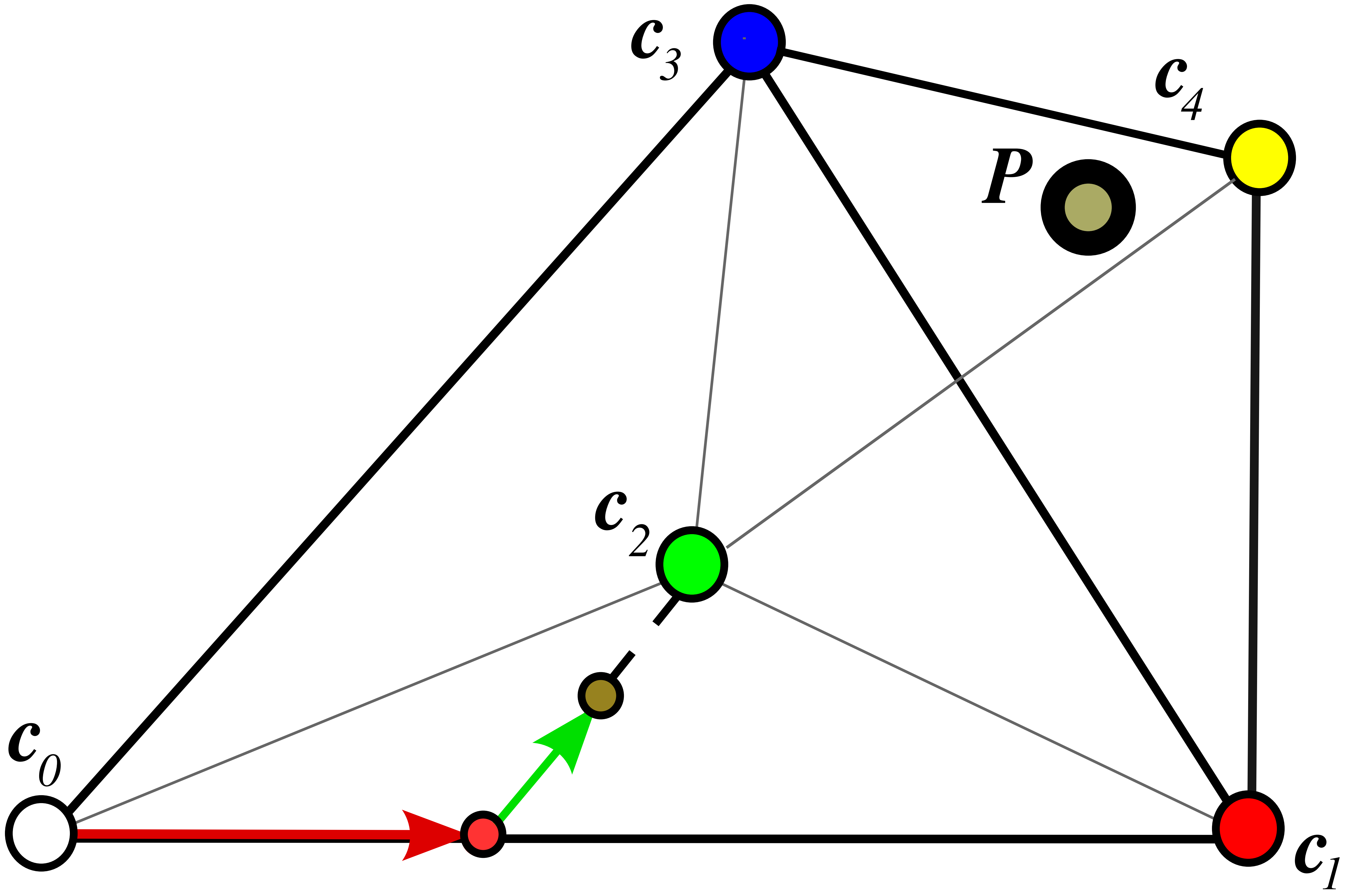
Path 1



Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P

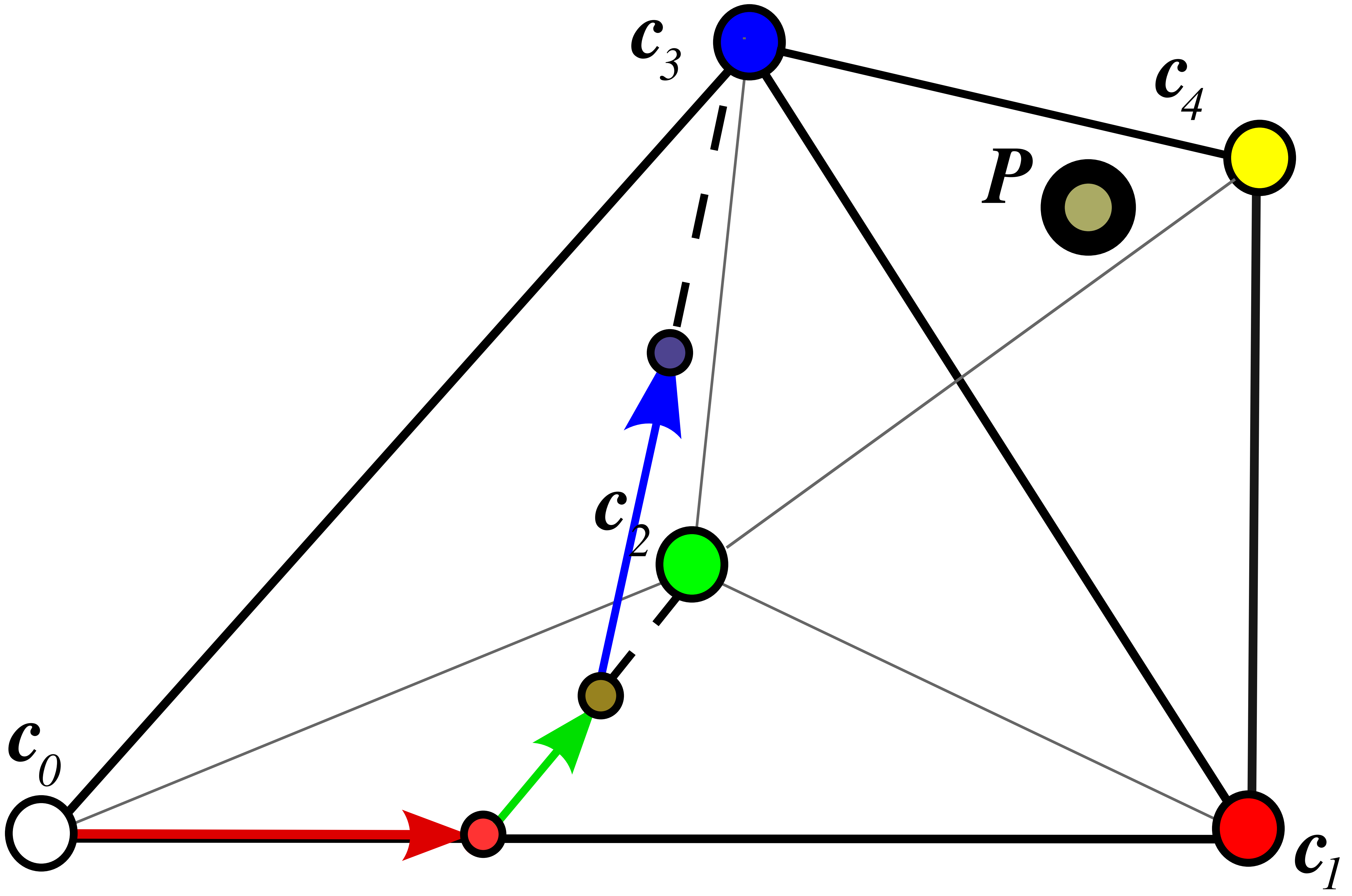
Path 1



Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P

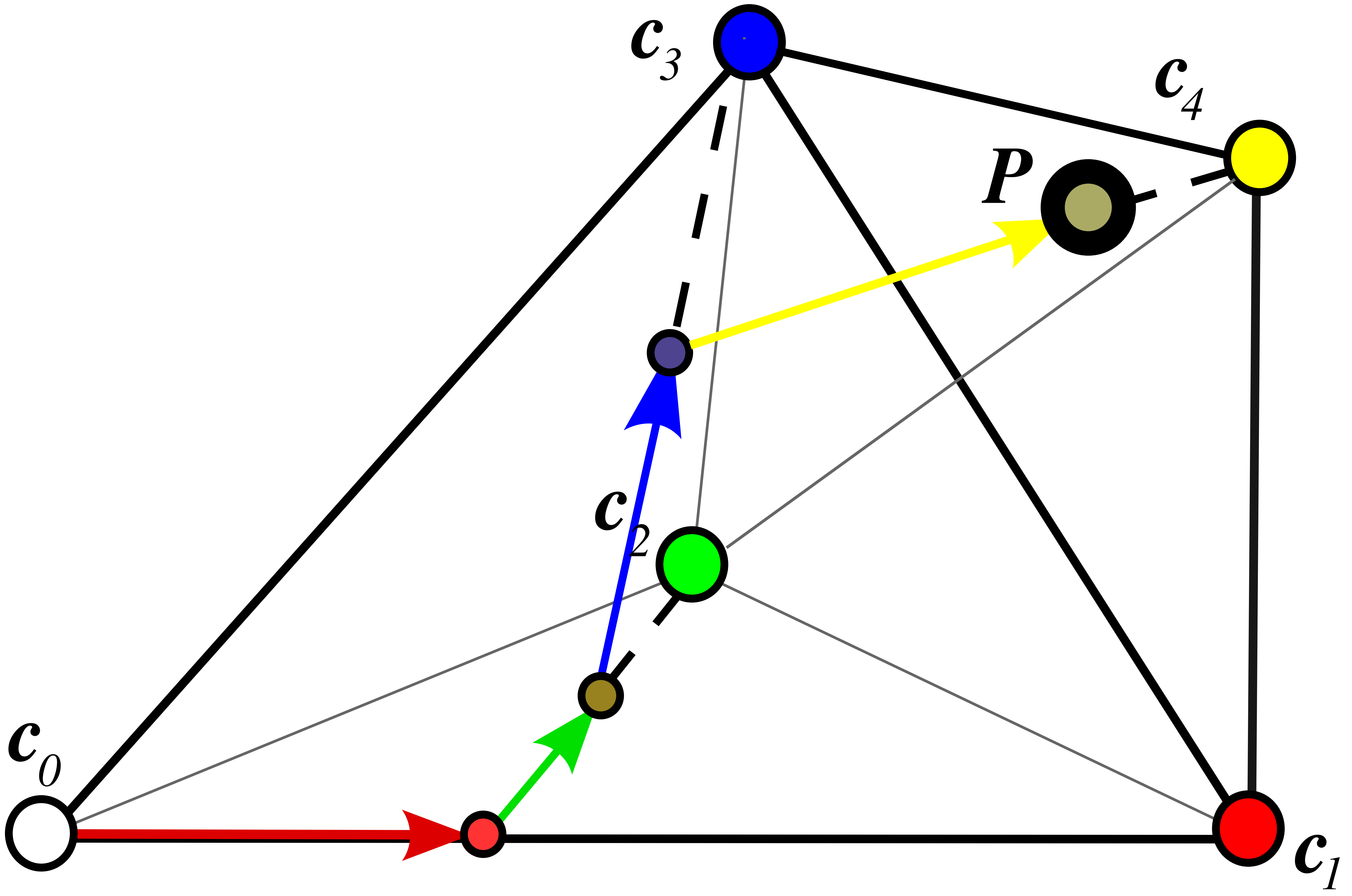
Path 1



Color Compositing Path

- After user chooses a layer order: $C_0 \to C_1 \to C_2 \to C_3 \to C_4$
- Still have **infinite** paths from C_0 to P

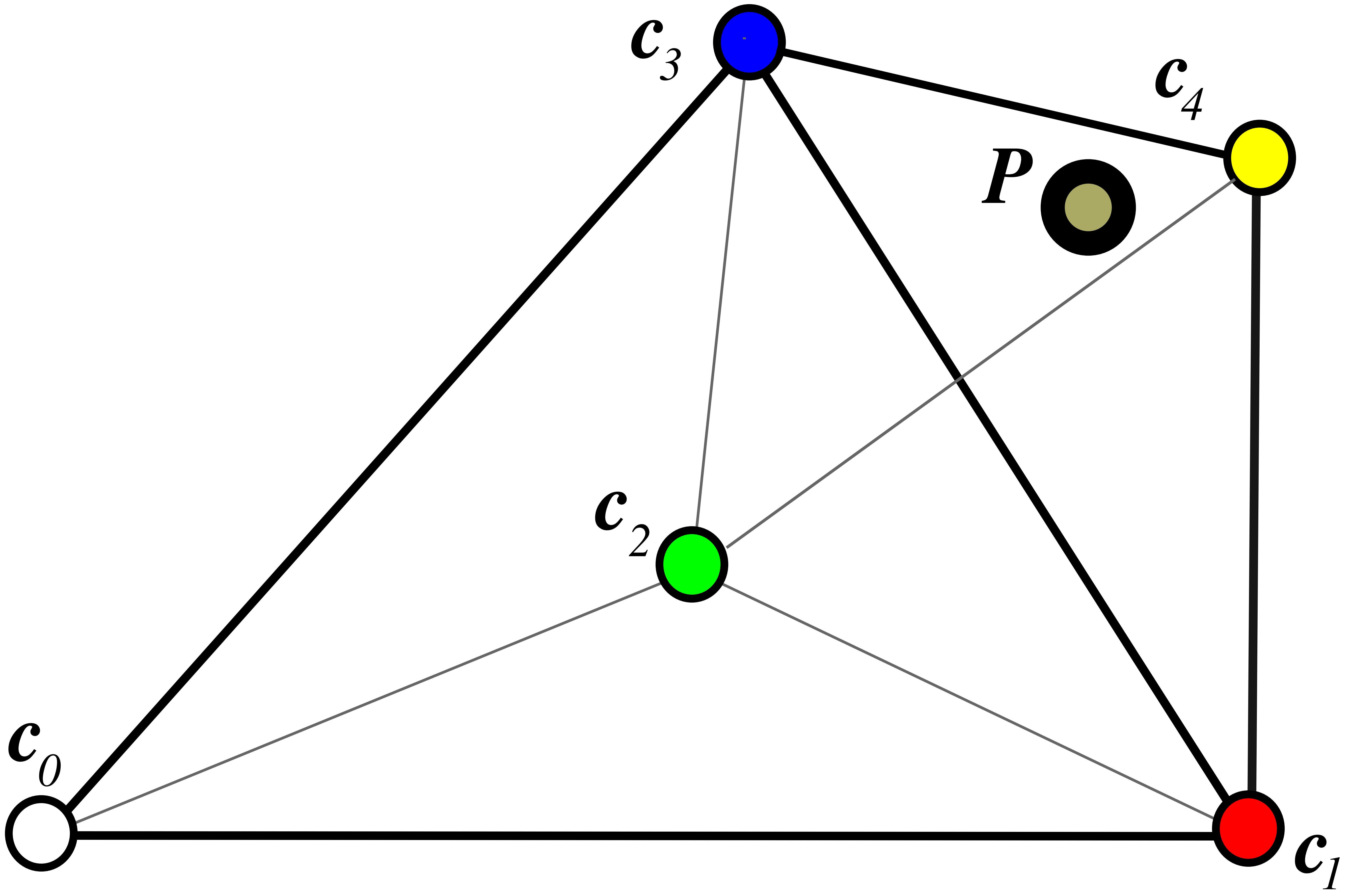
Path 1



Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P

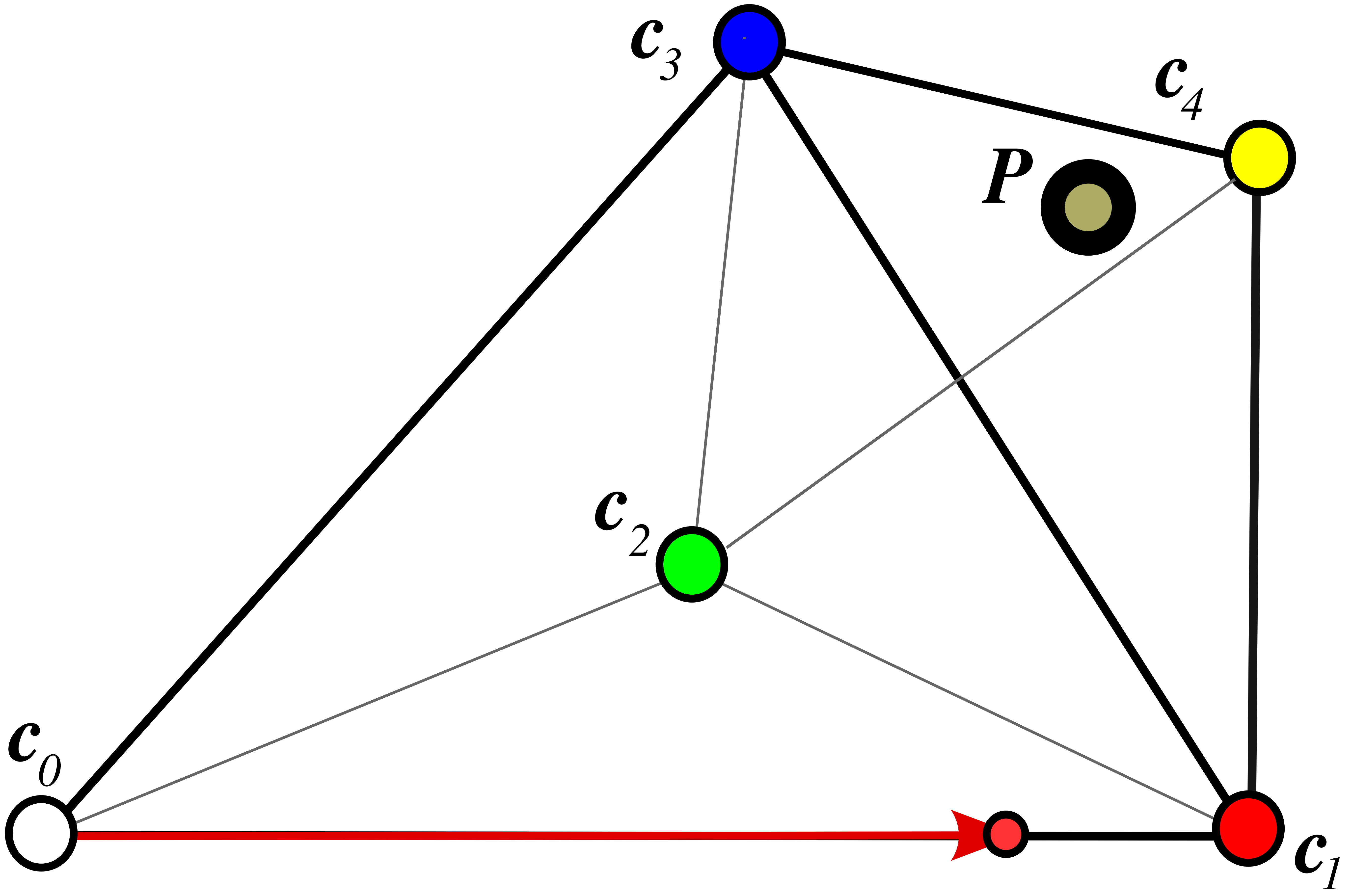
Path 2



Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P

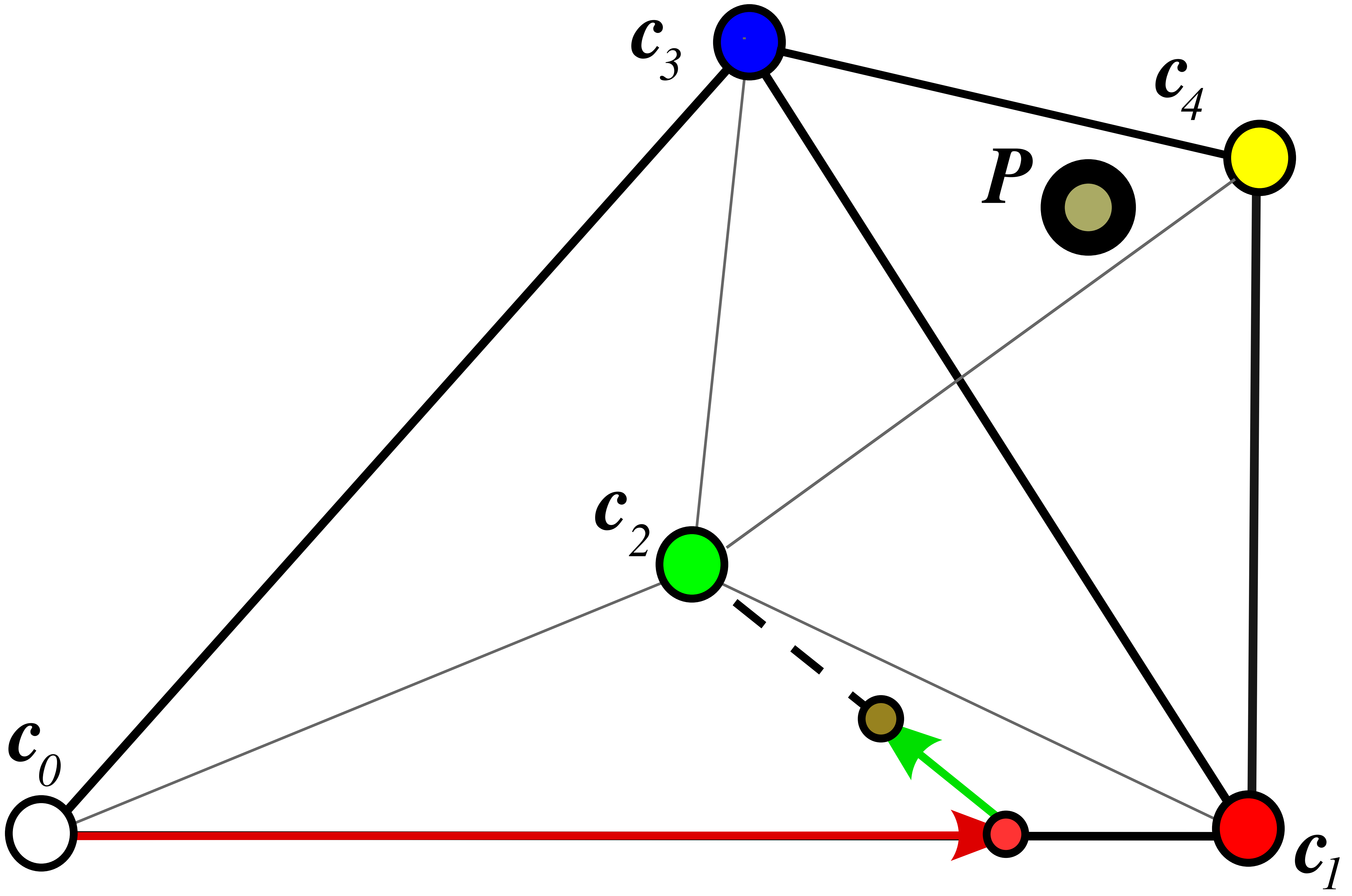
Path 2



Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P

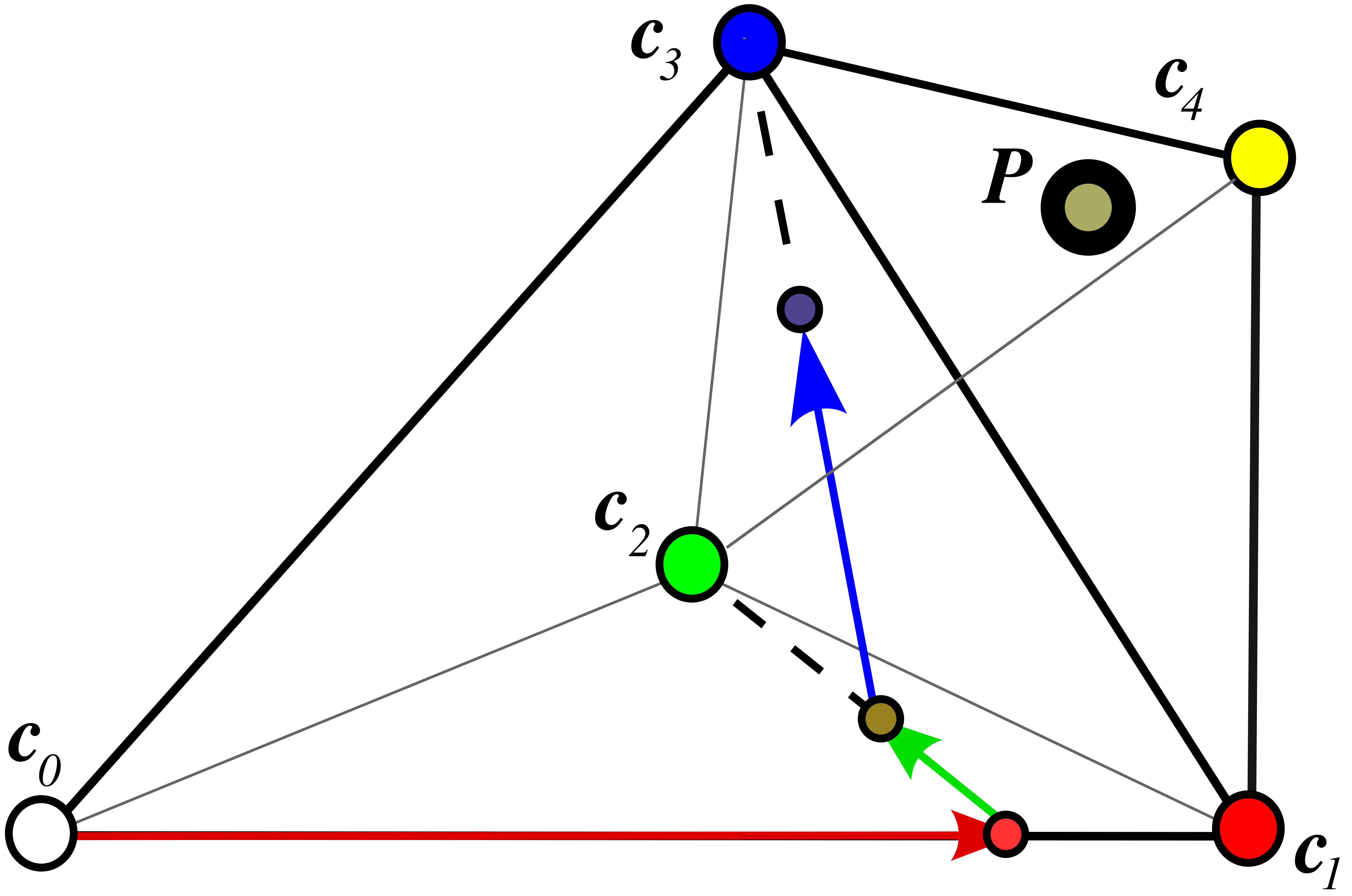
Path 2



Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P

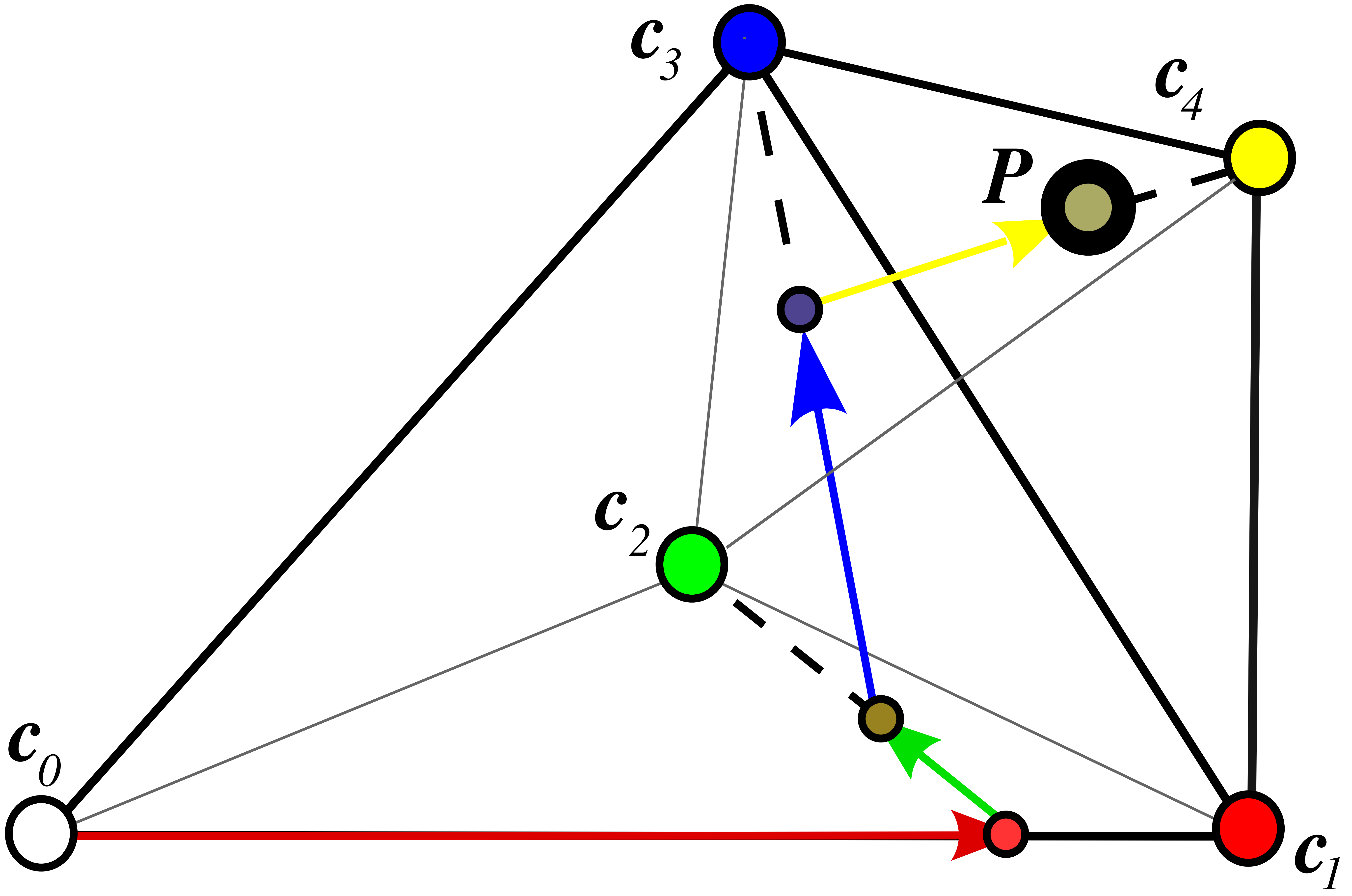
Path 2



Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P

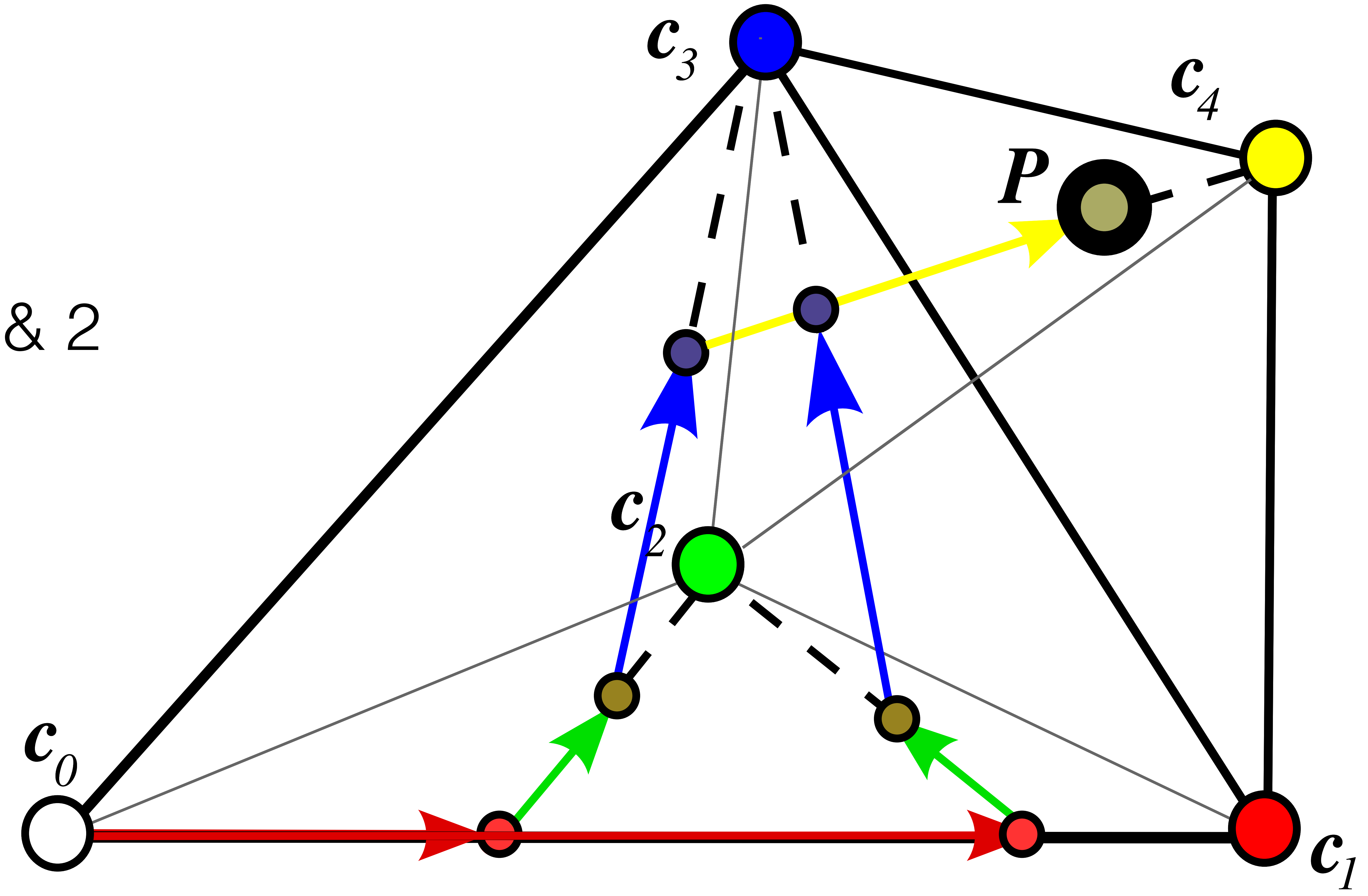
Path 2



Color Compositing Path

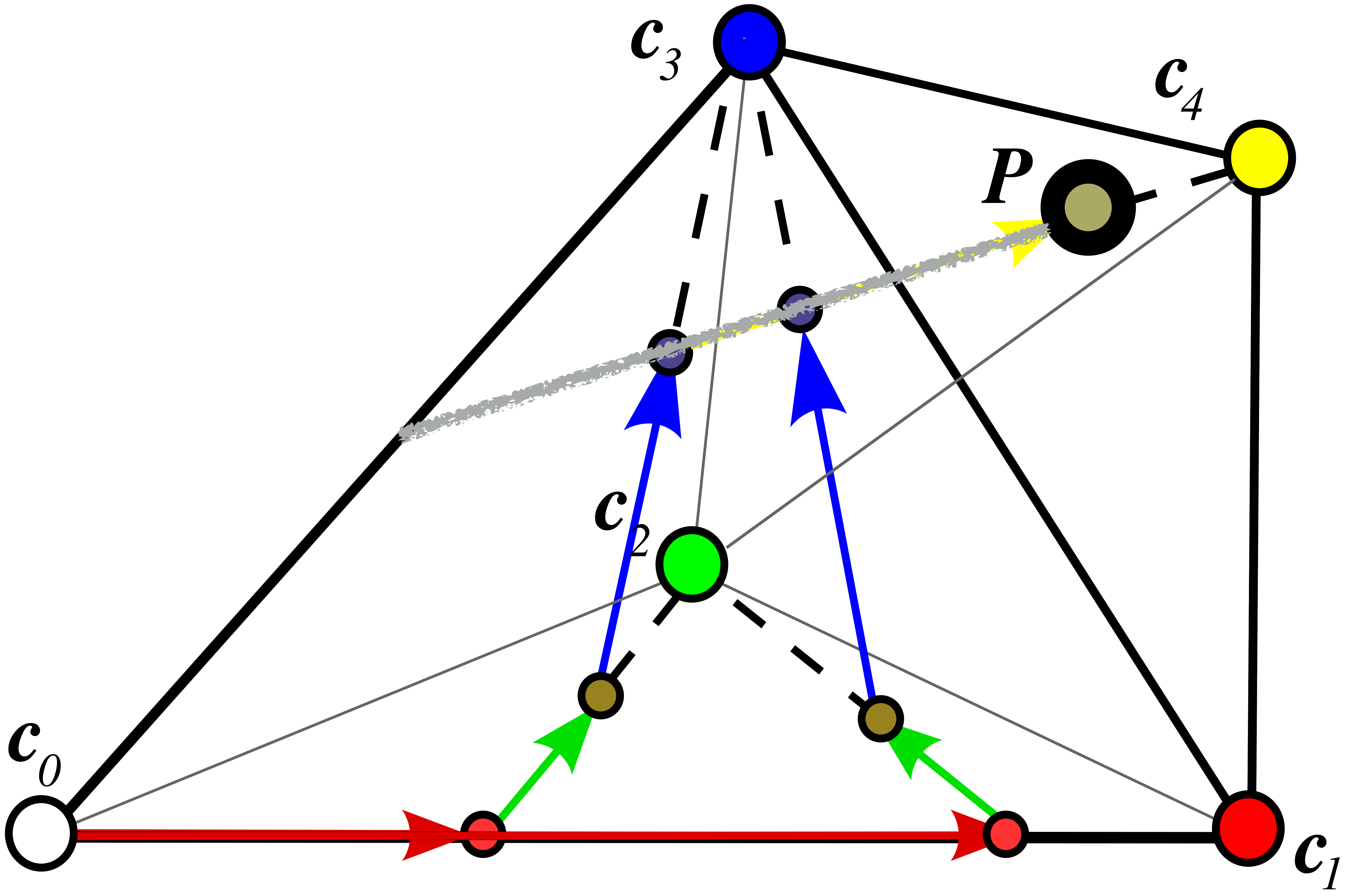
- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P

Paths 1 & 2



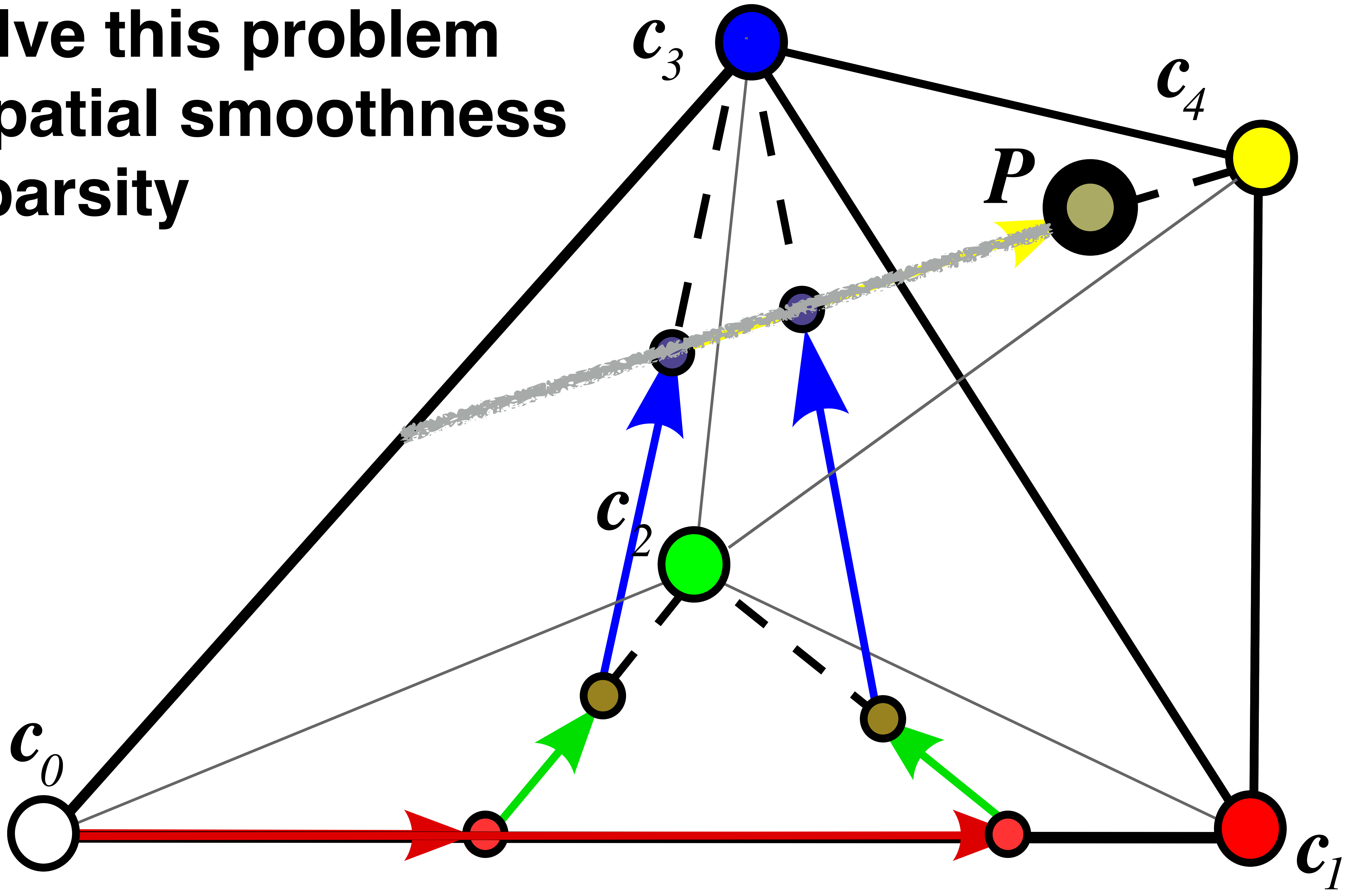
Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P



Color Compositing Path

- After user chooses a layer order: $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_4$
- Still have **infinite** paths from C_0 to P
- **We solve this problem with spatial smoothness and sparsity**



Layer Opacity Optimization

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

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** =

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

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** =

$\|$original − reconstructed image$\|^2$ (**polynomial**)

+

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

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** =

||original − 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

$\|$original − 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

||original – reconstructed image||^2 (**polynomial**)

+

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

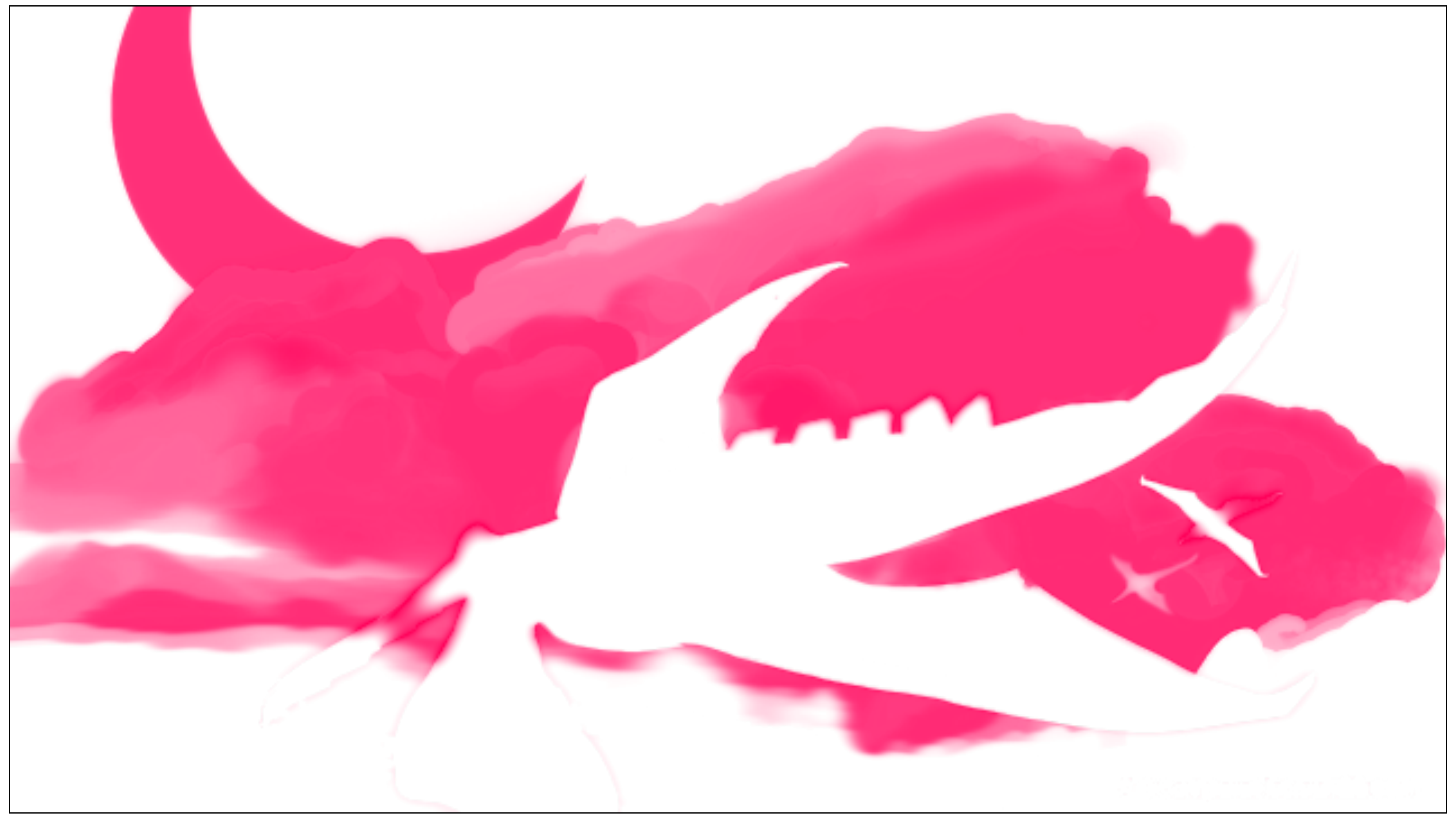
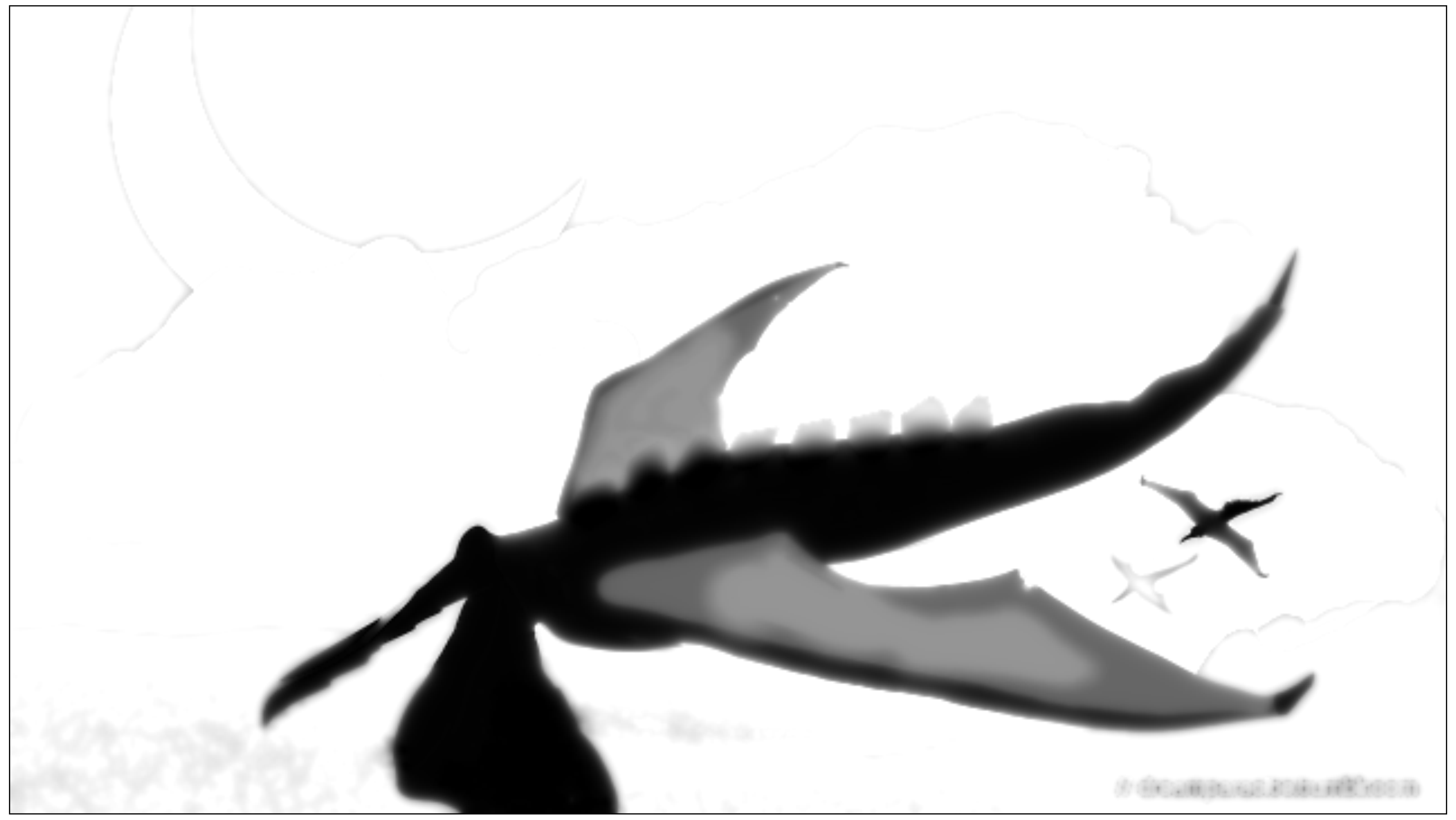
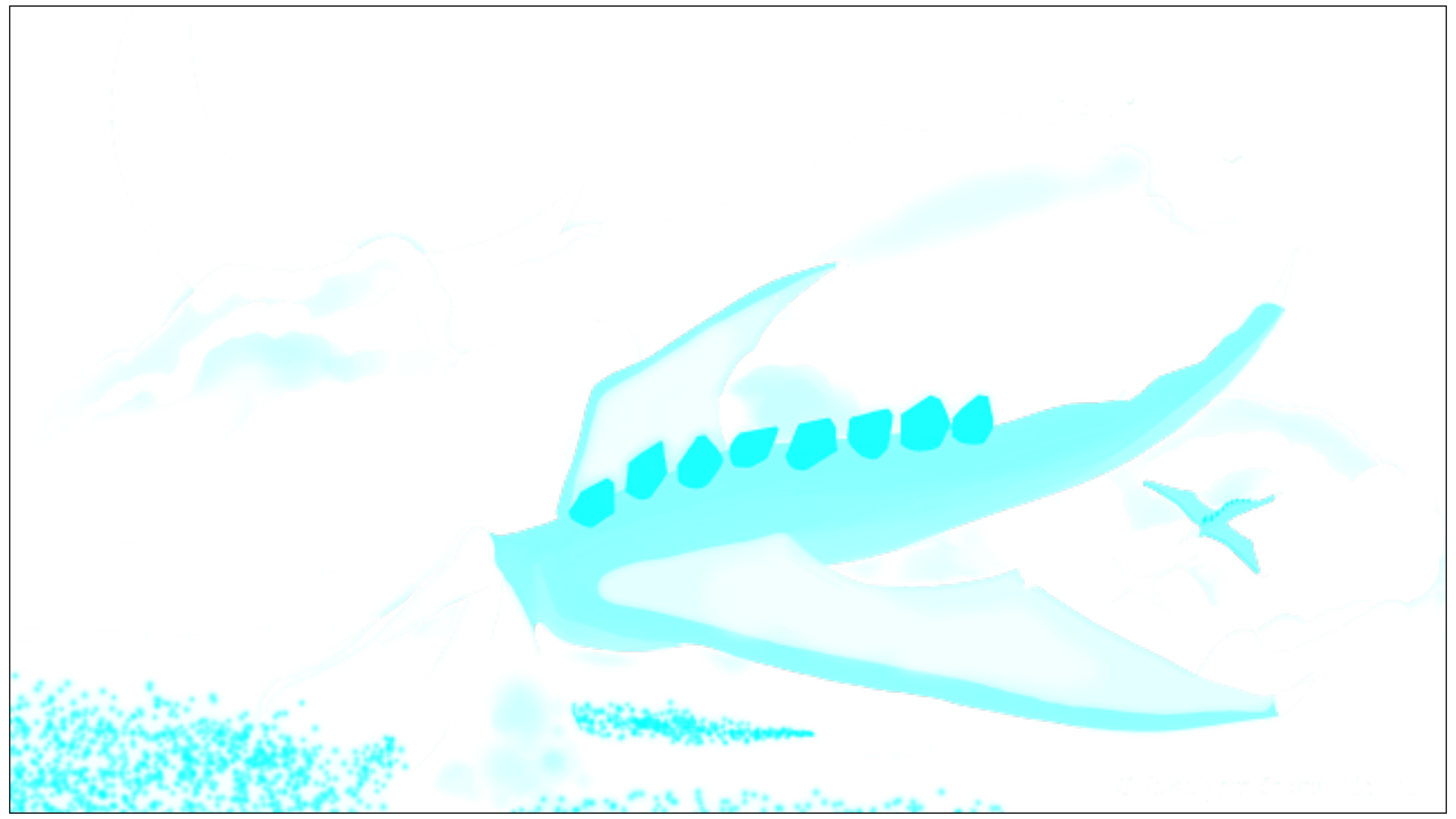
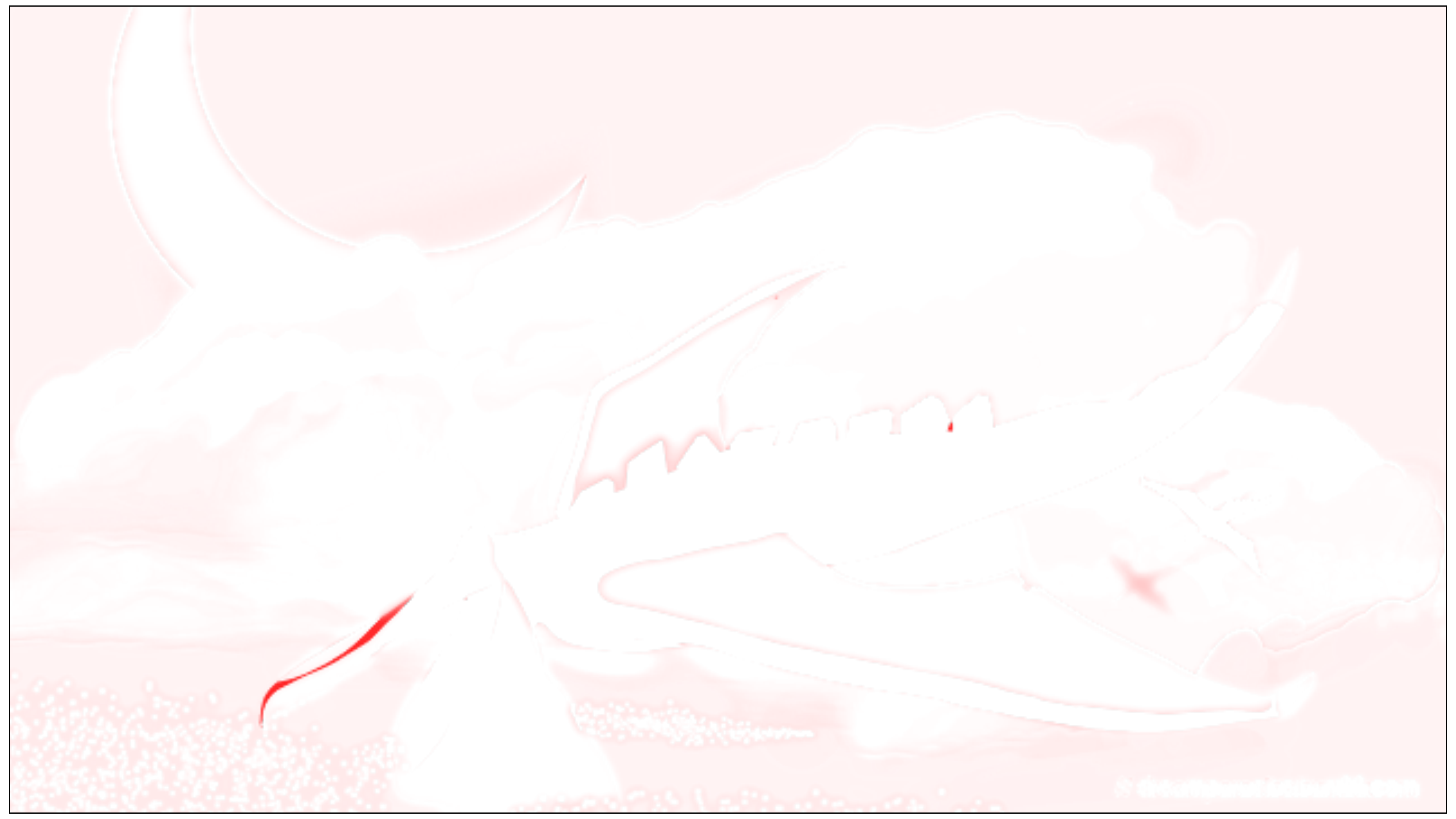
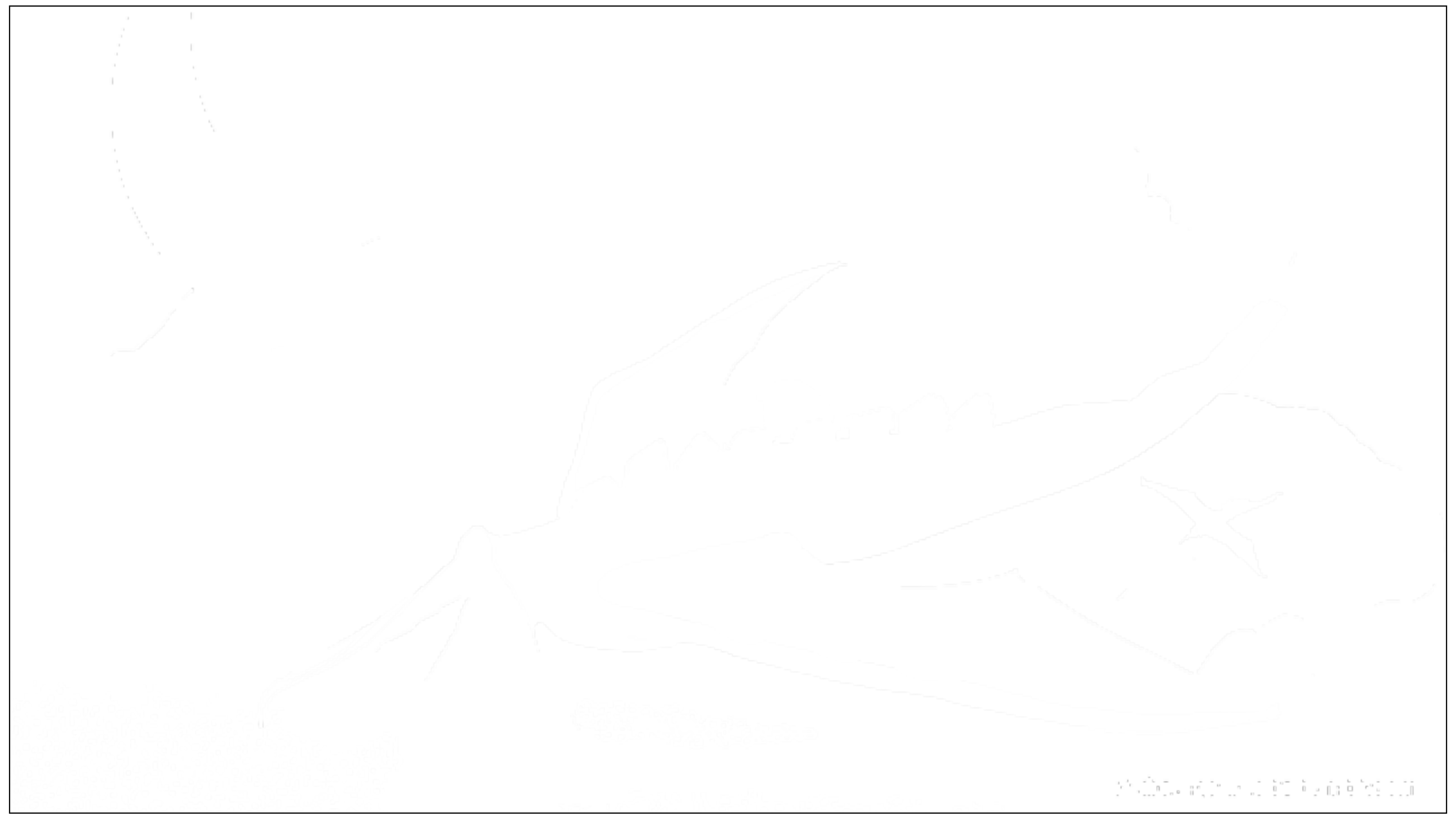
+

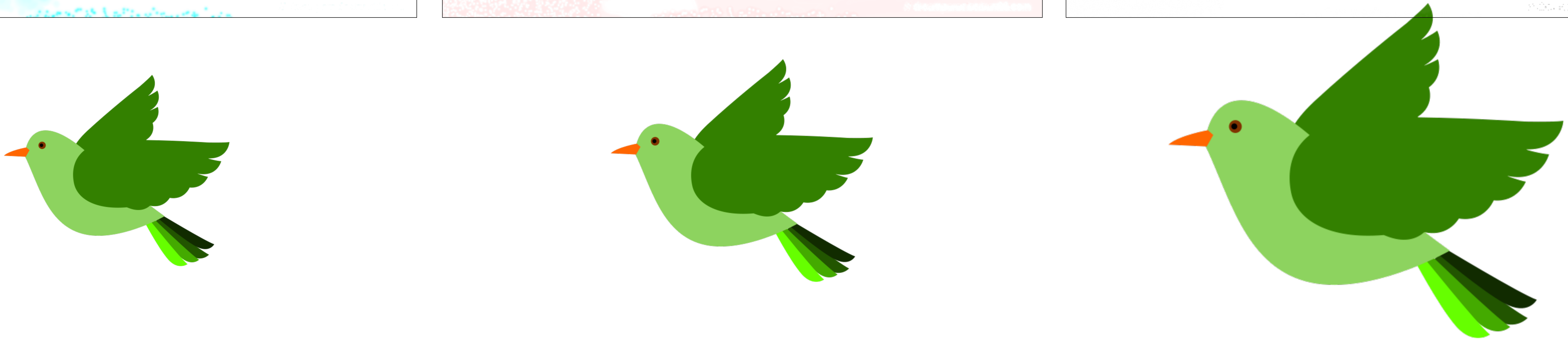
Opacity spatial smoothness (**Laplacian**)

We also have a closed form expression for an "**As-Sparse-As-Possible**" solution if you don't care about spatial smoothness. See our paper for details.

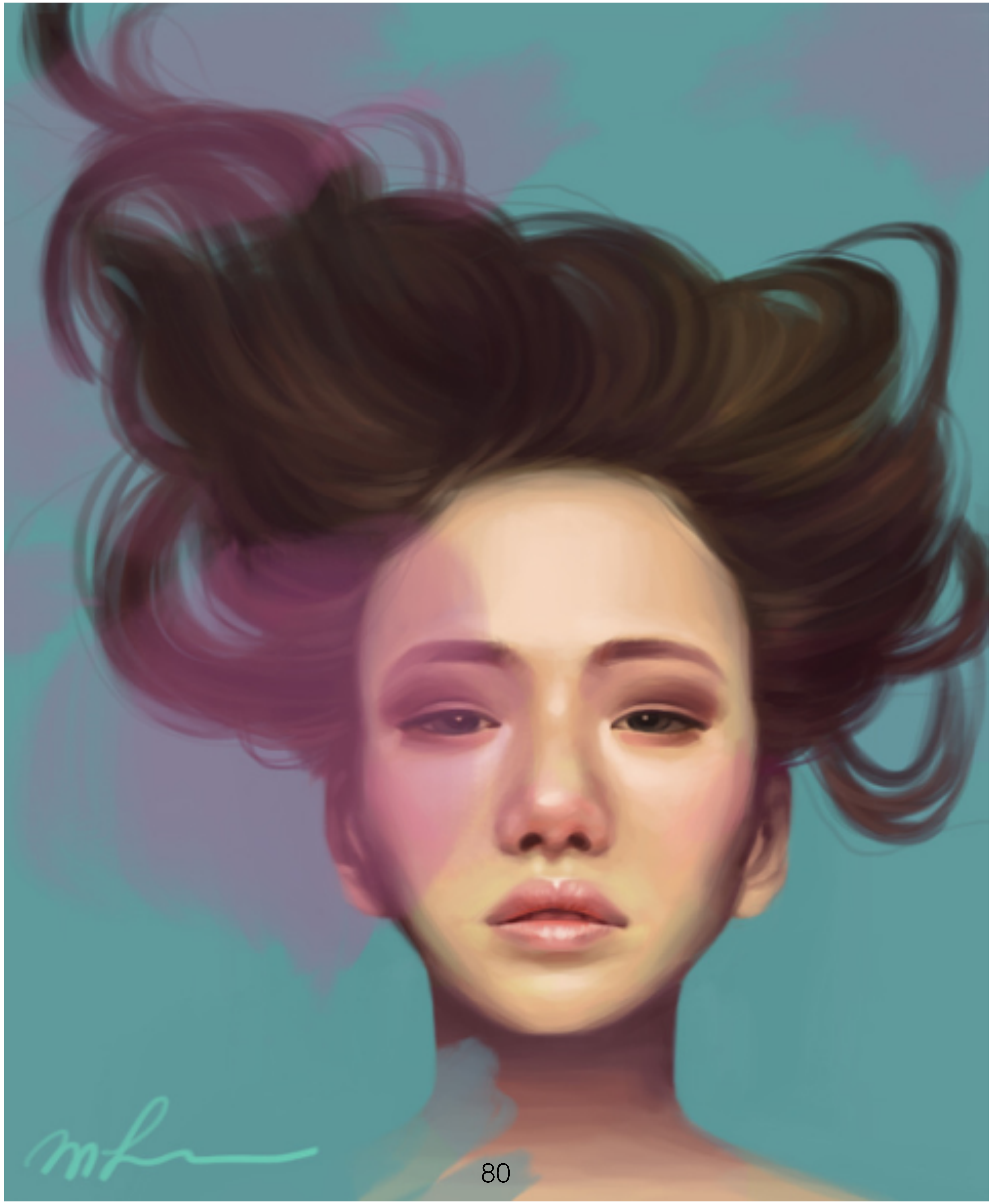
Results


// dreamparacite.tumblr.com




// dreamparacite.tumblr.com

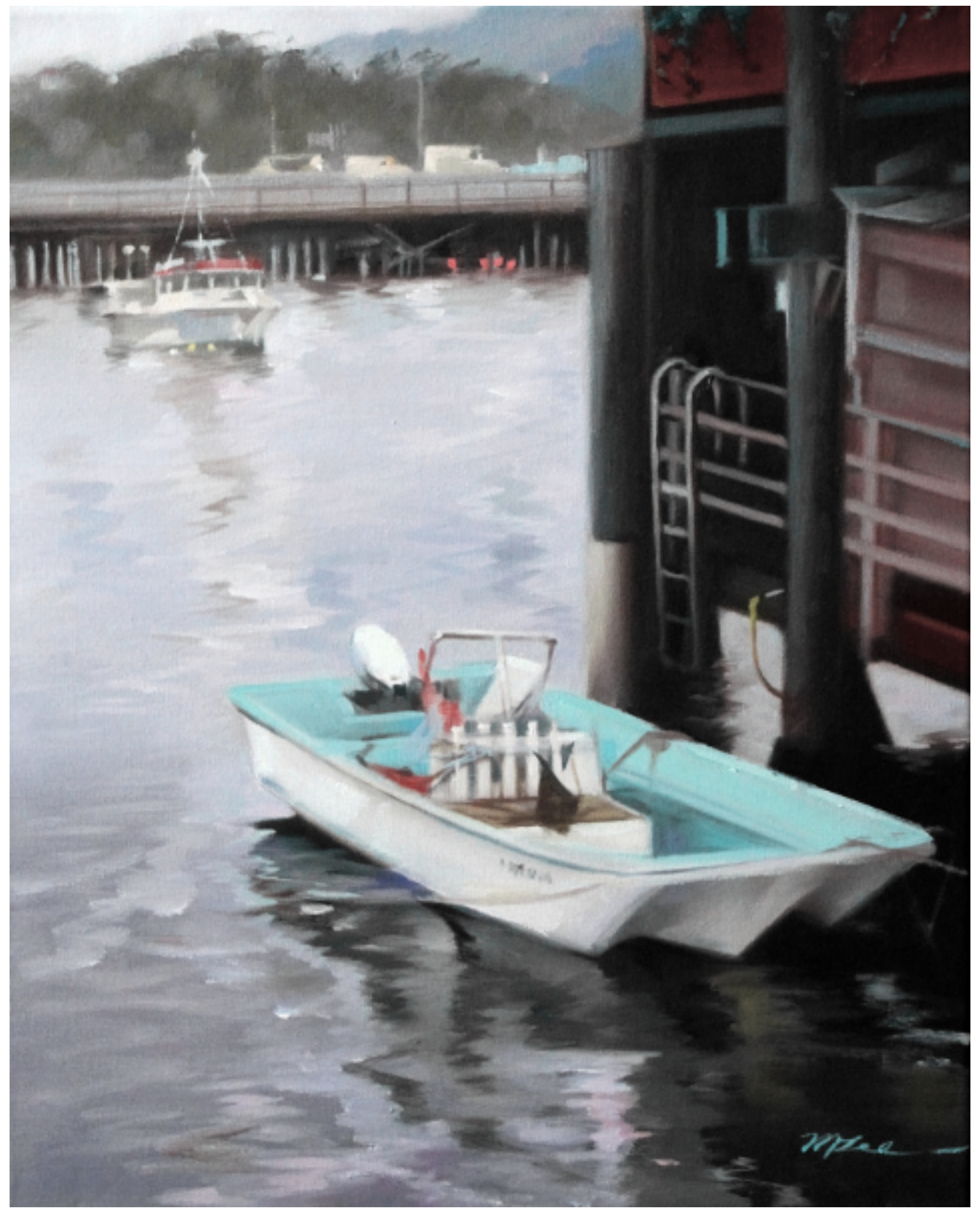


http://cargocollective.com/michellelee/

http://cargocollective.com/michellelee/

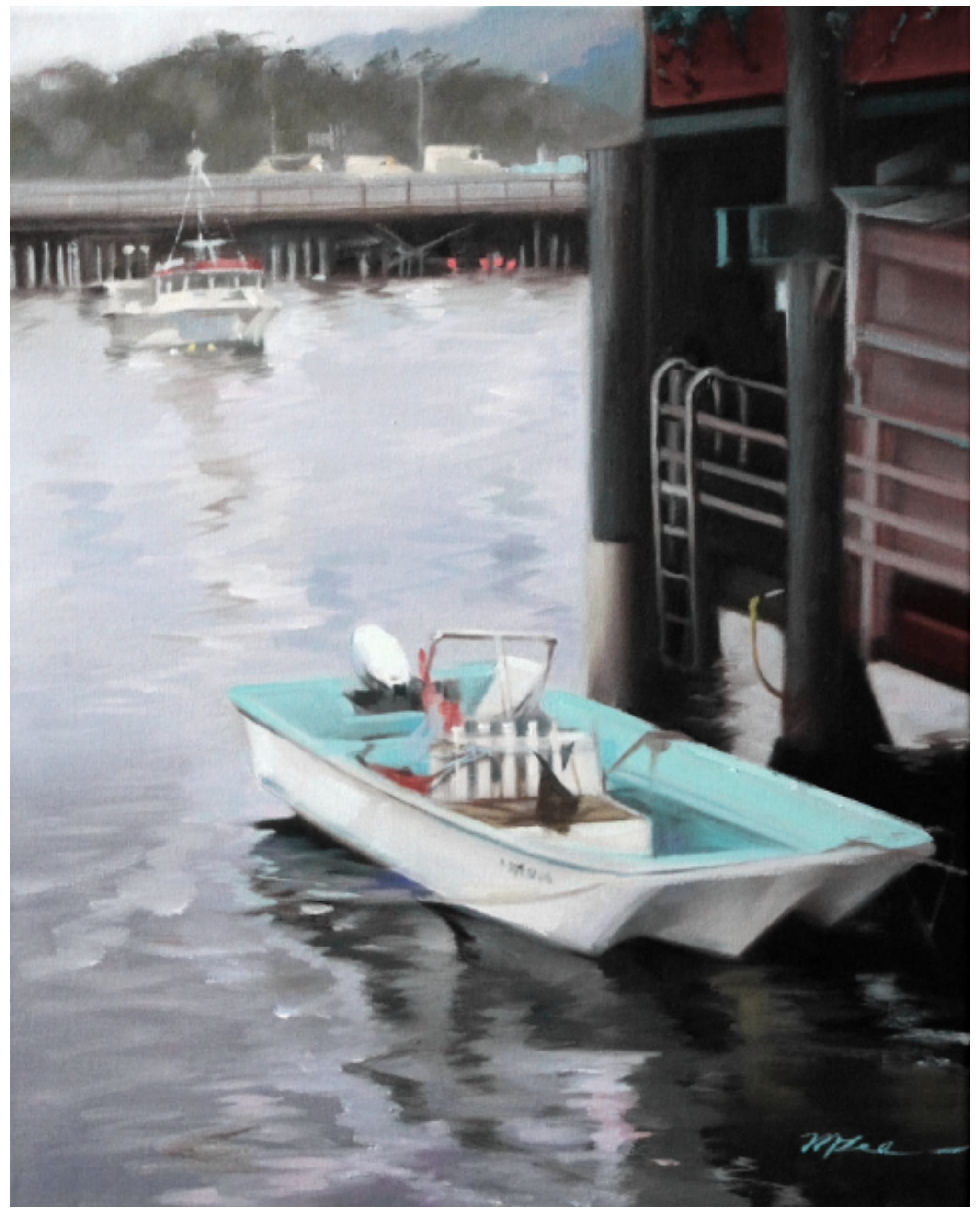
Local Recoloring

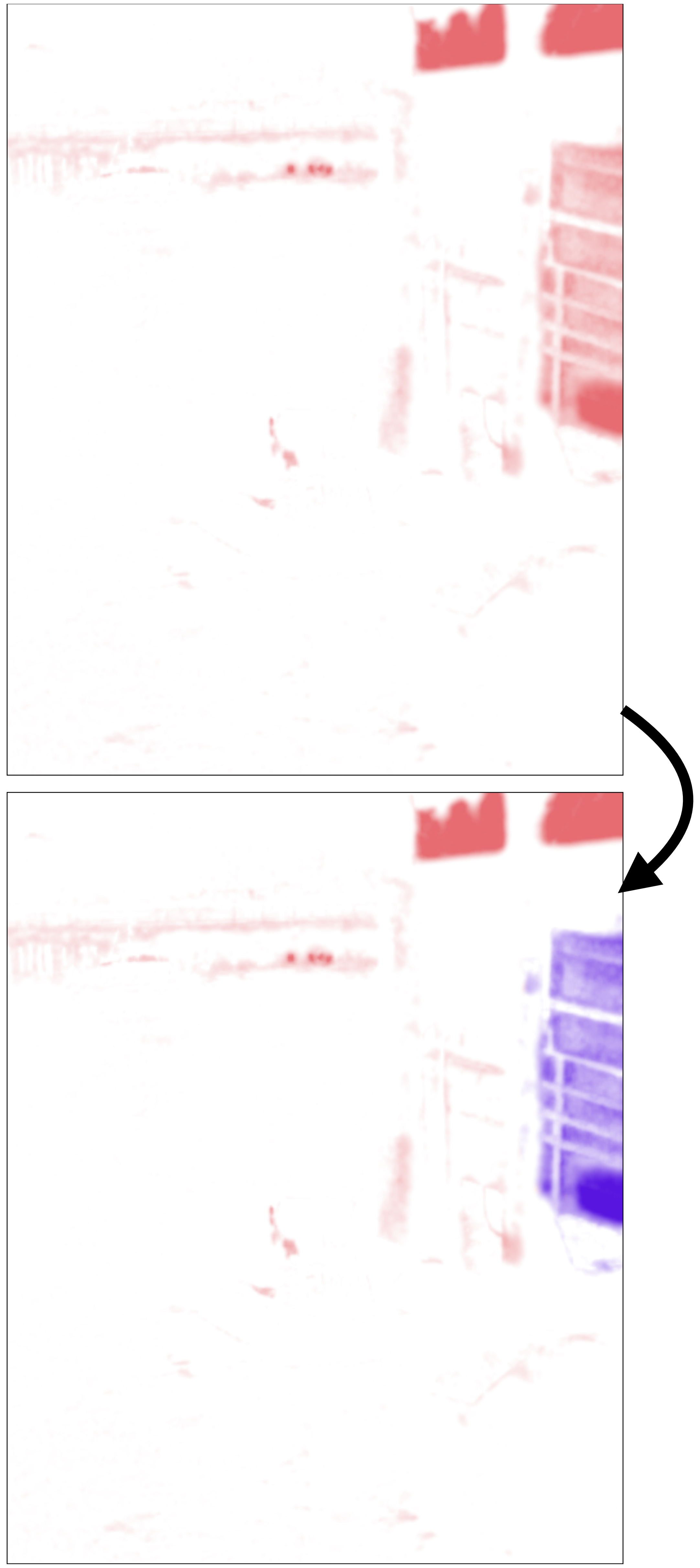
Original



Local Recoloring

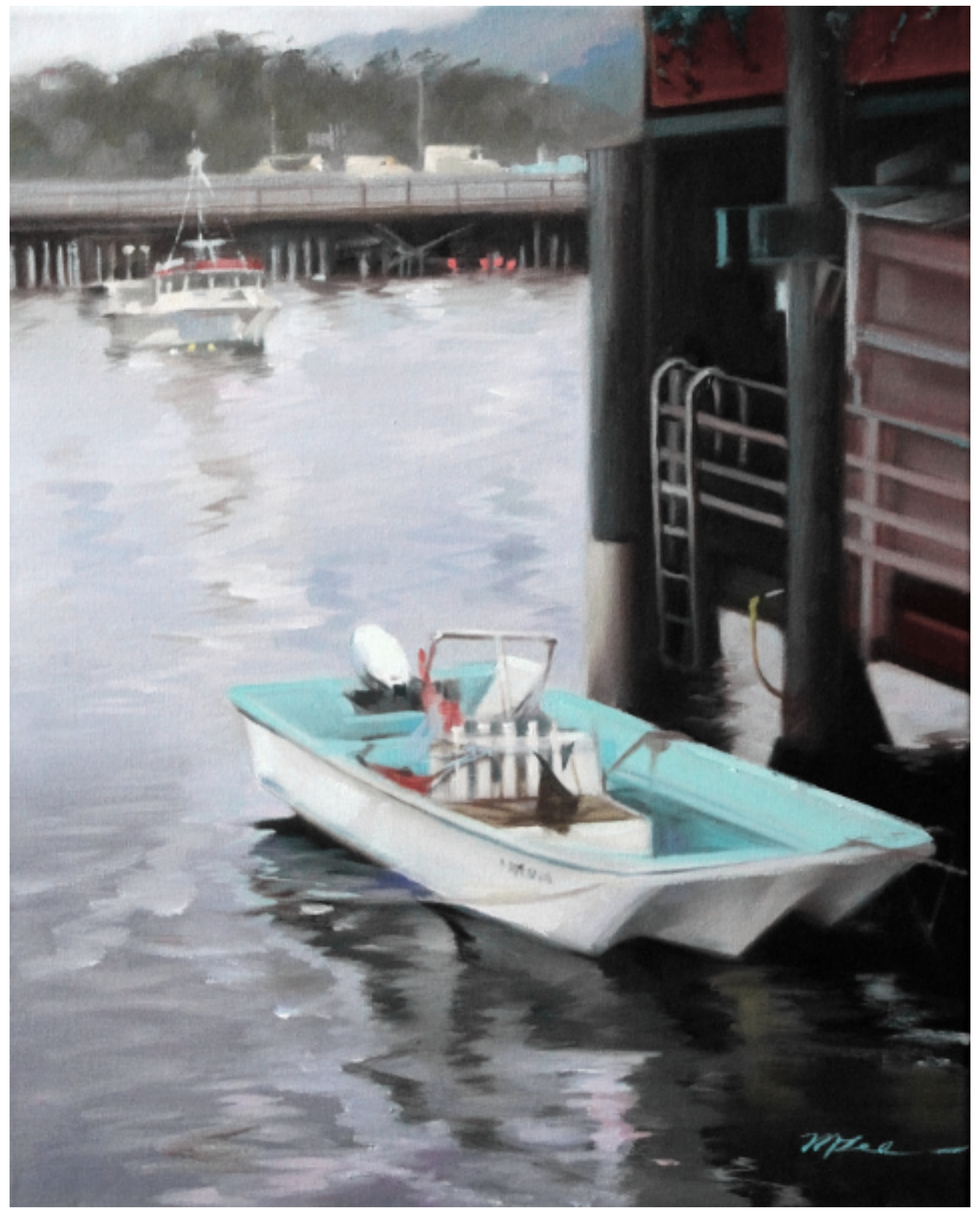
Original

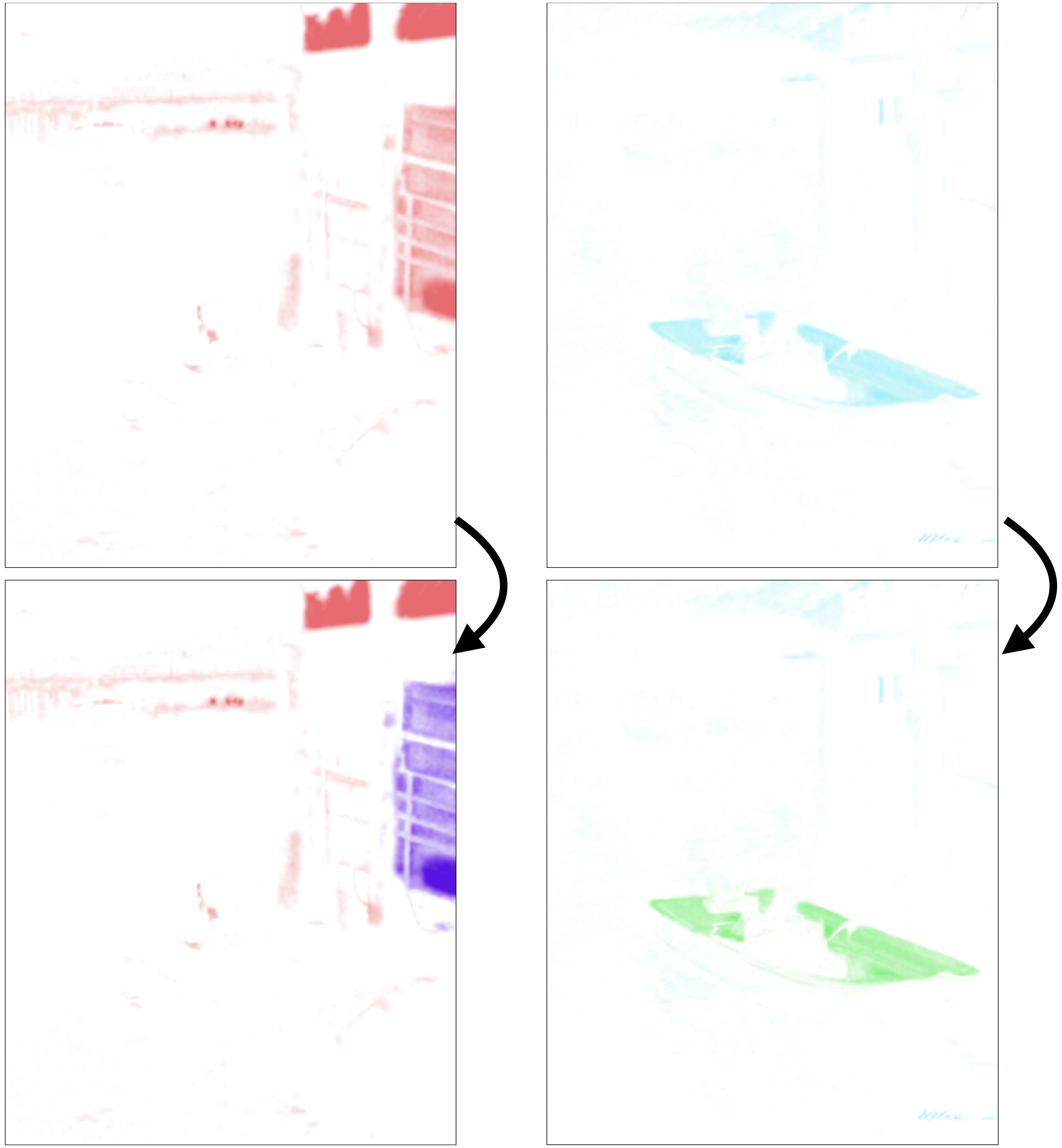




Local Recoloring

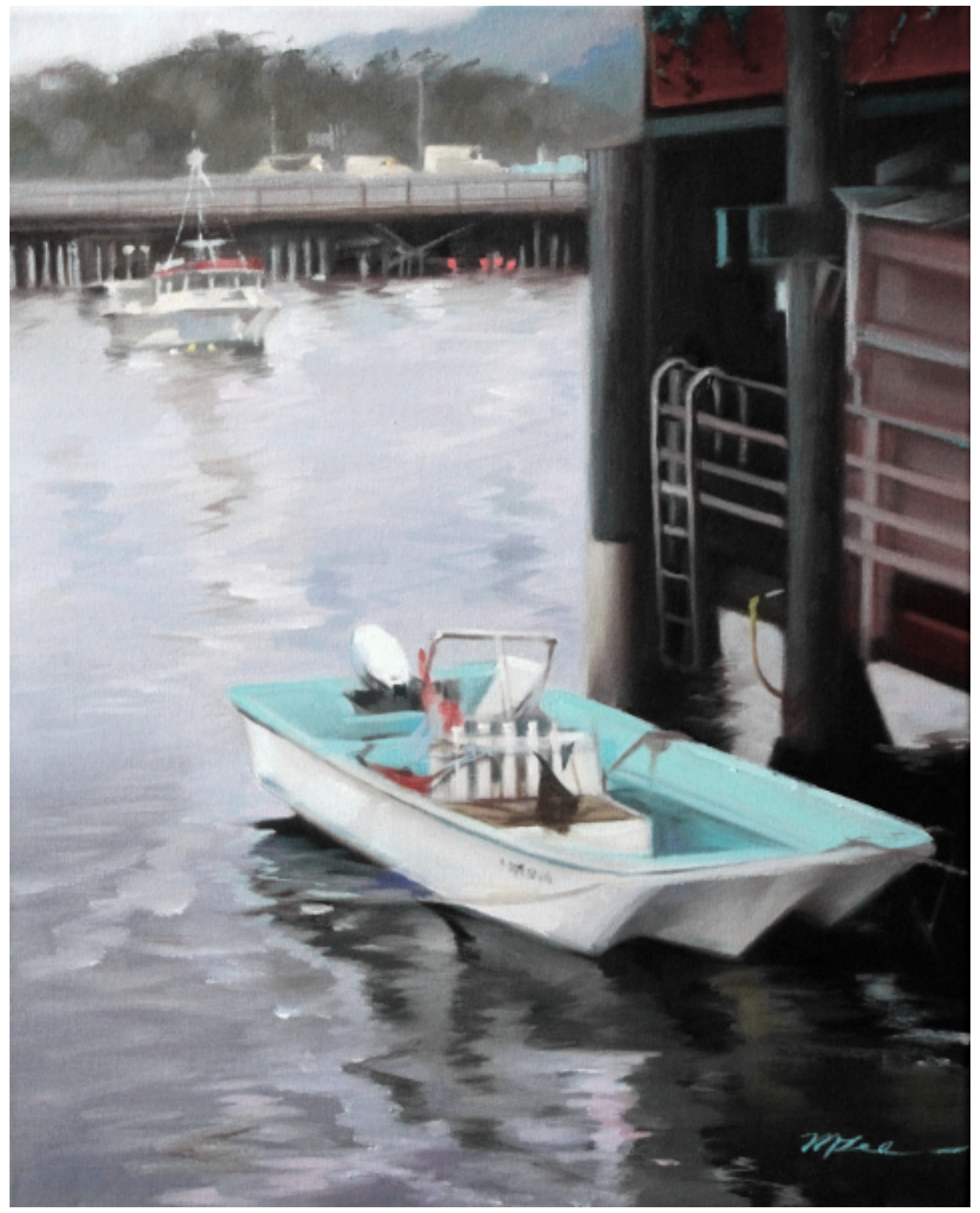
Original

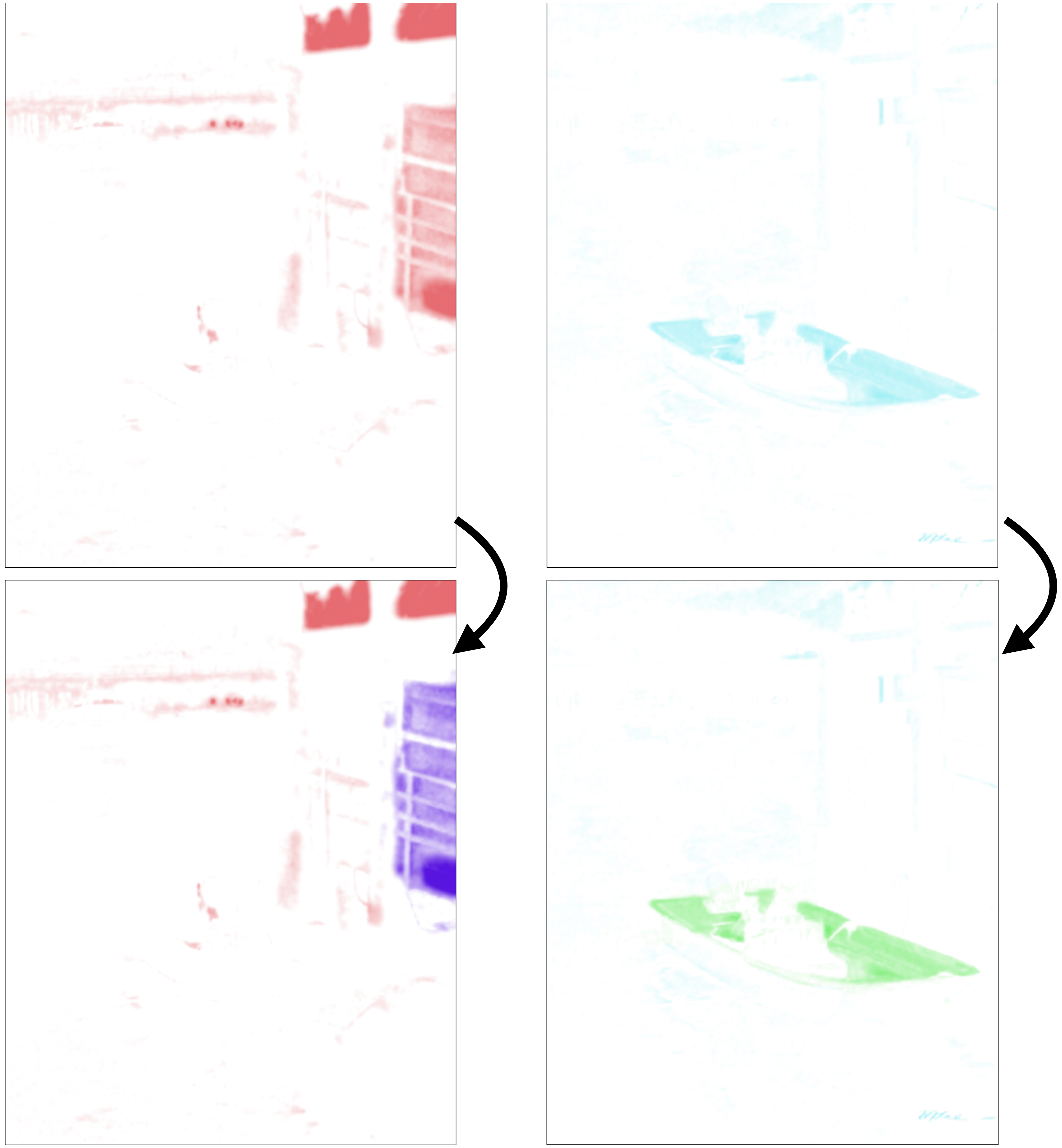




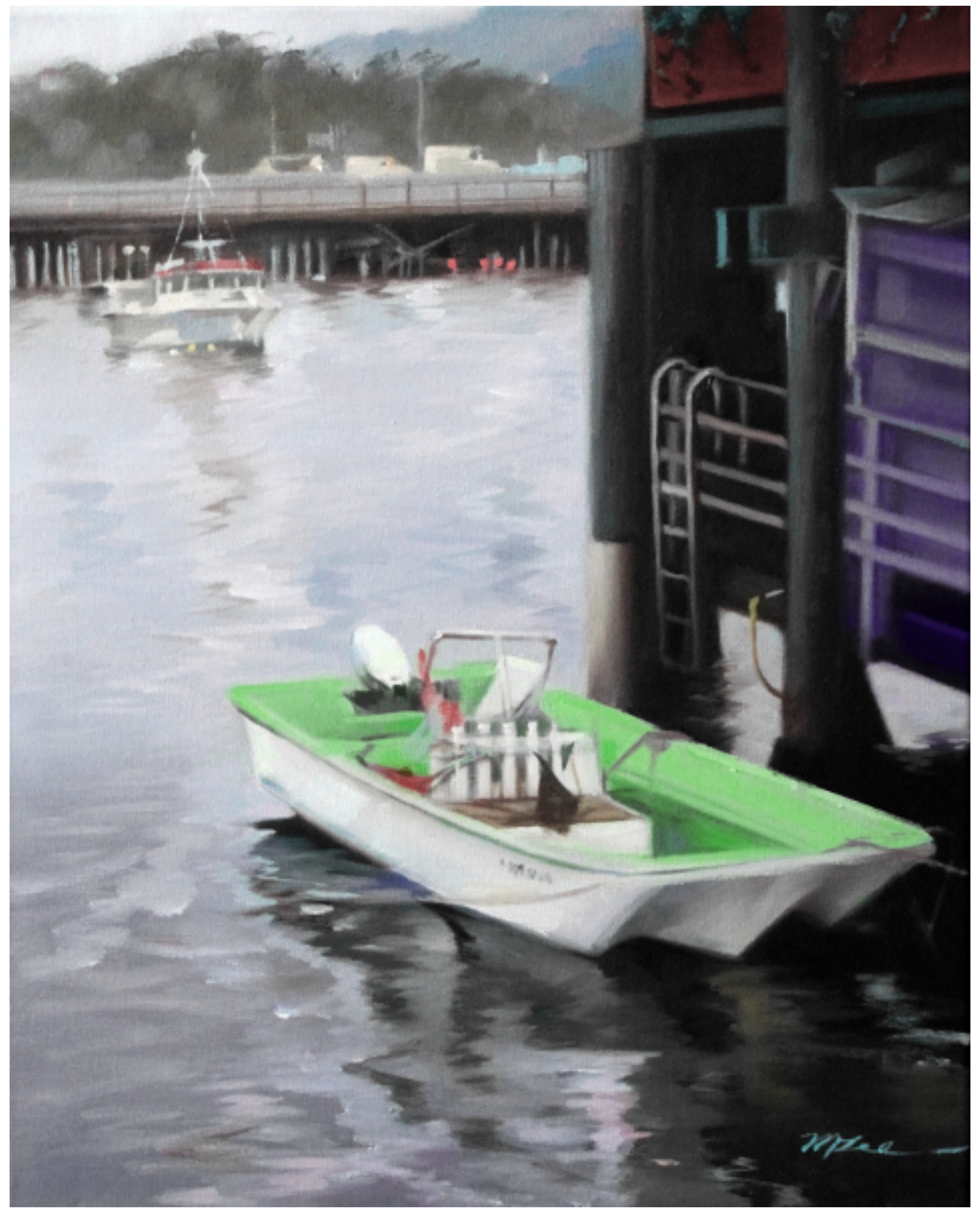
Local Recoloring

Original





Modified



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

$$p = \sum \boxed{w_i} c_i$$

linear mixing weights

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

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

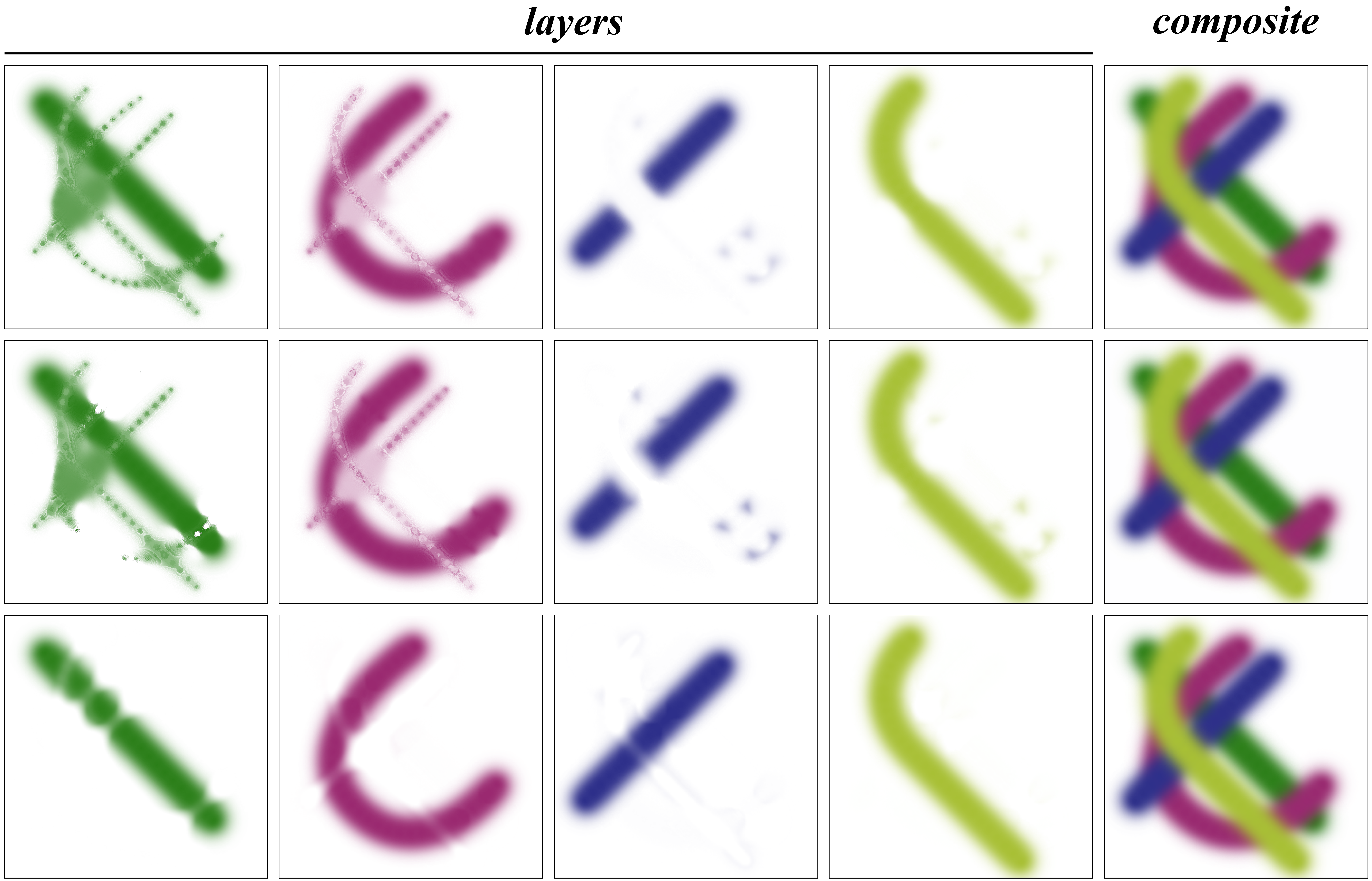
Ambiguous

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

layer opacities

Alpha Compositing

Generalized Barycentric Coordinates



MVC
Mean Value Coordinates
[Floater et al. 2005]
[Ju et al. 2005]

LBC
Local Bary. Coordinates
[Zhang et al. 2014]

Ours

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

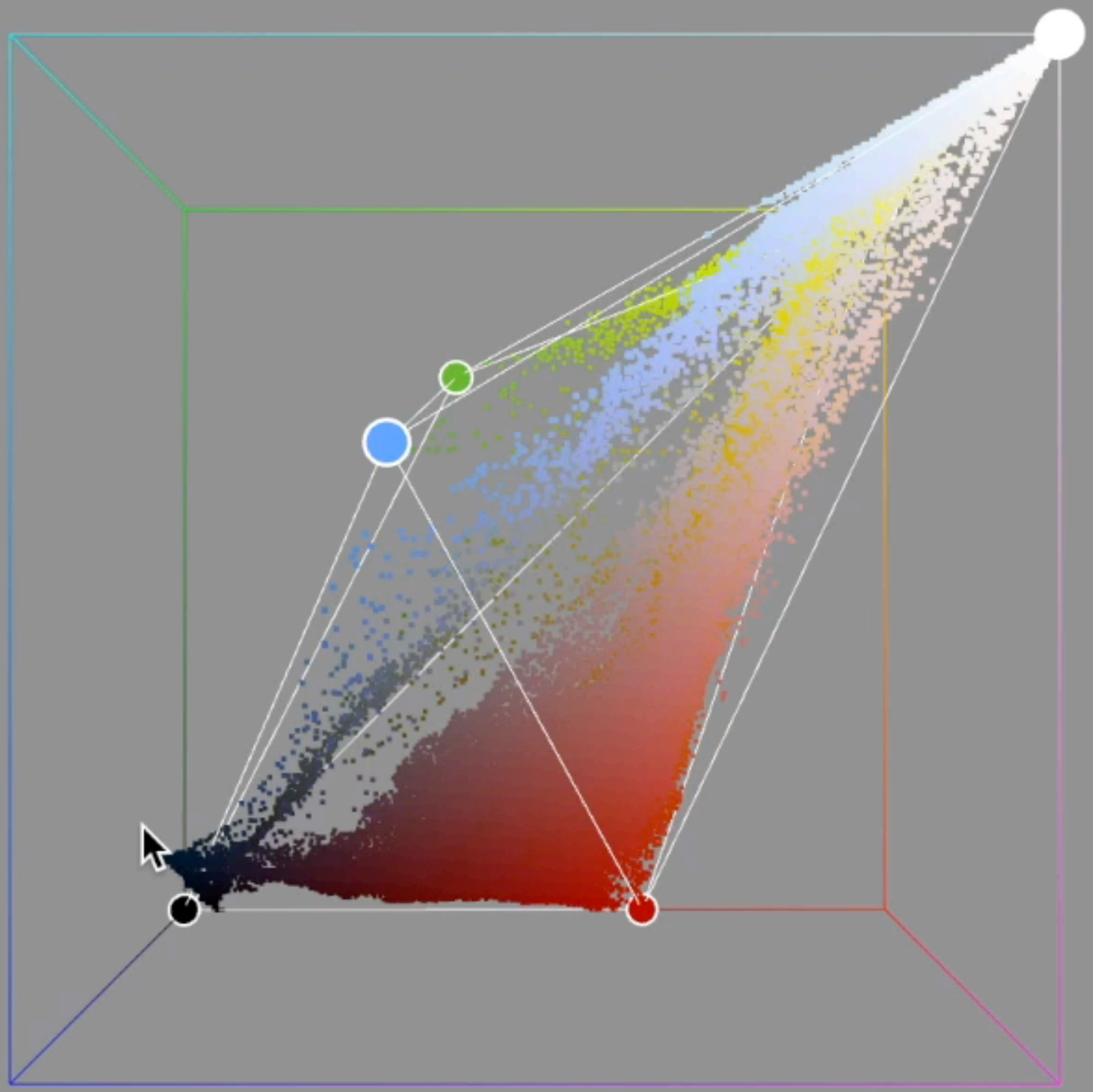
Choose File No file chosen

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

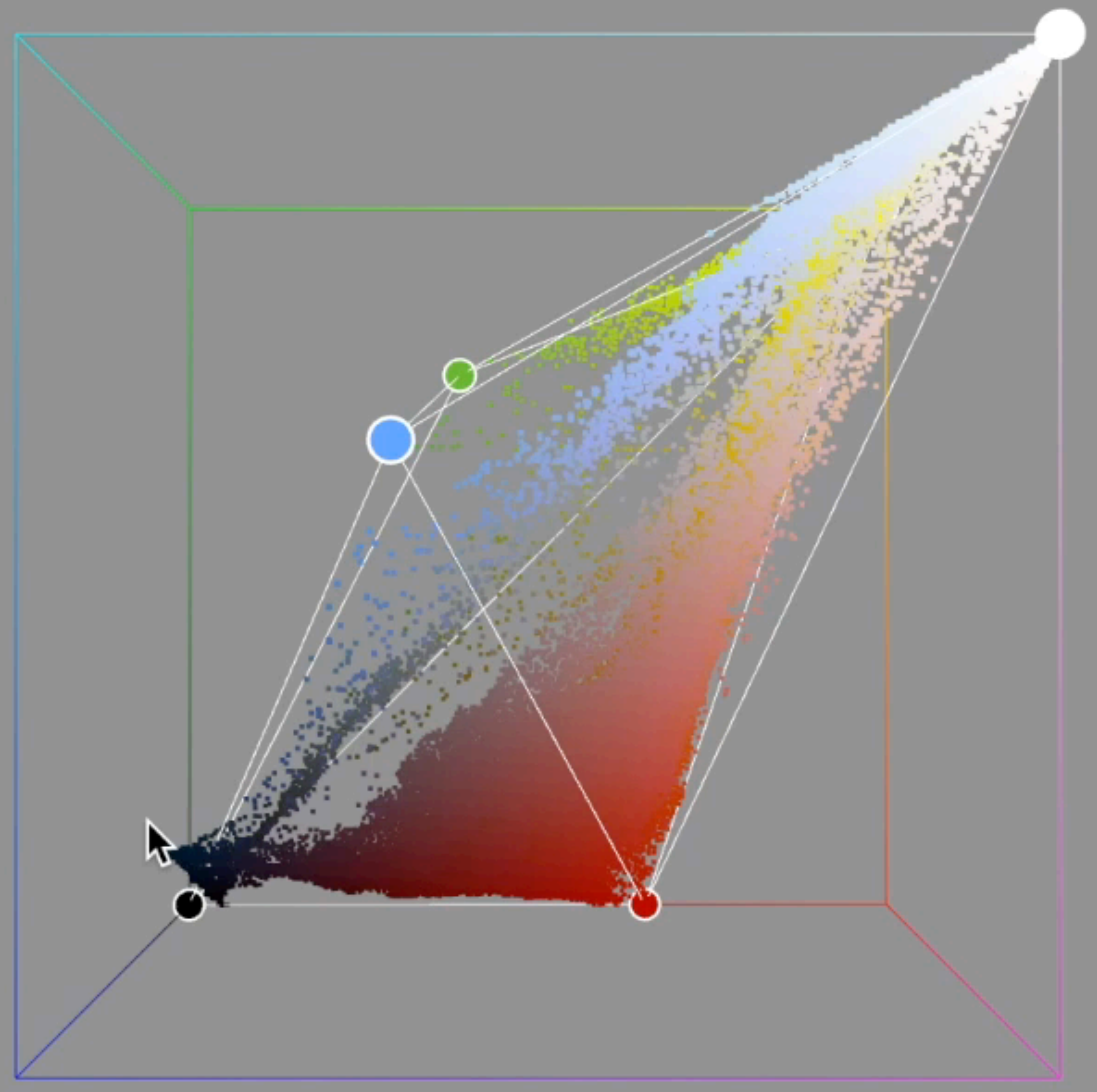
Choose File No file chosen

Rotation has intertia: ☐

Look from white

Save Everything

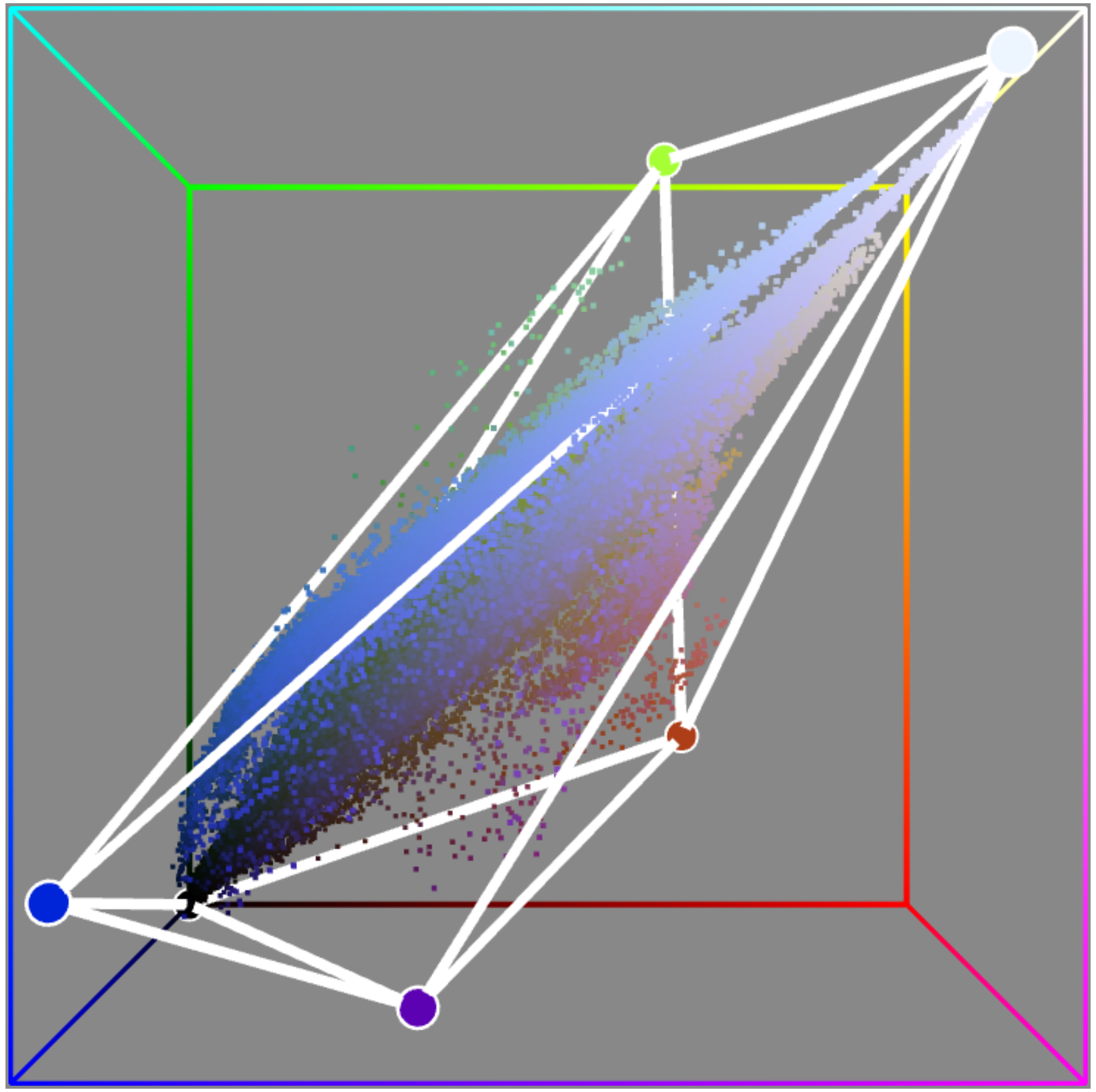
Save Camera Only



Natural Images

Original

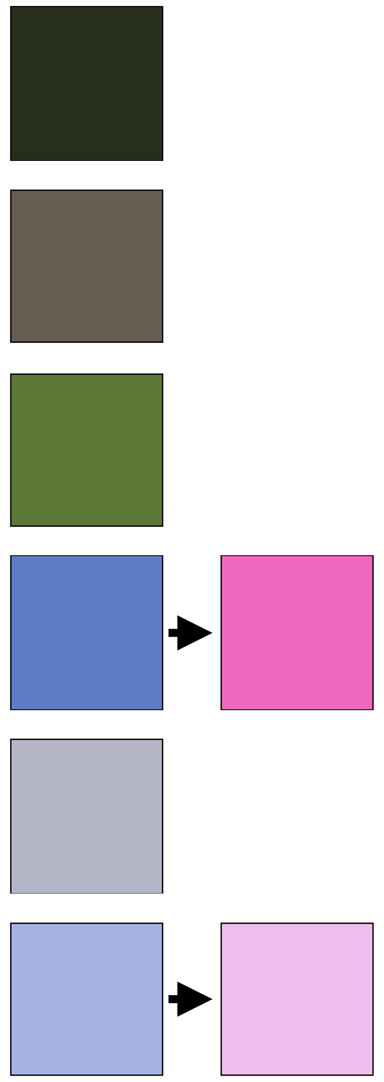
Simplified hull



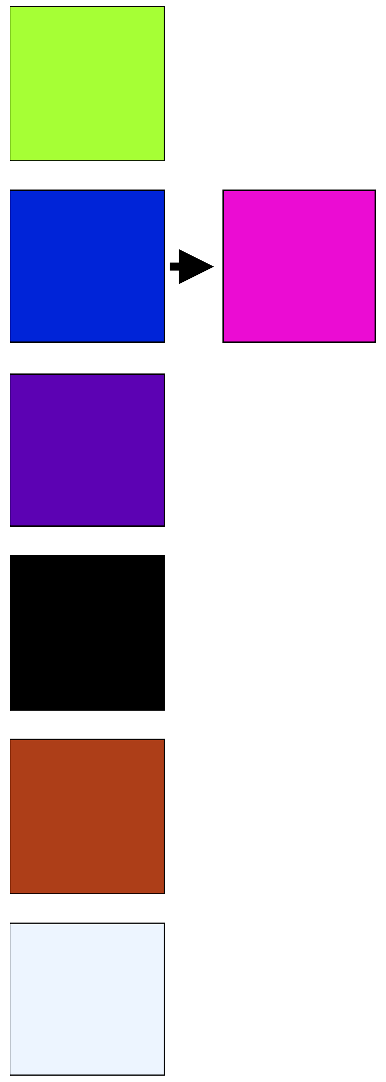
layers

Global Recoloring Comparison

Original



Chang et al. 2015



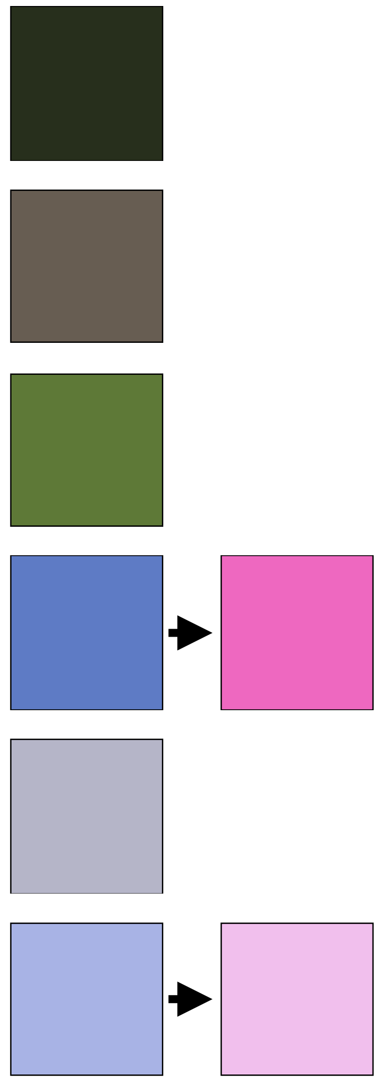
Ours

MVC

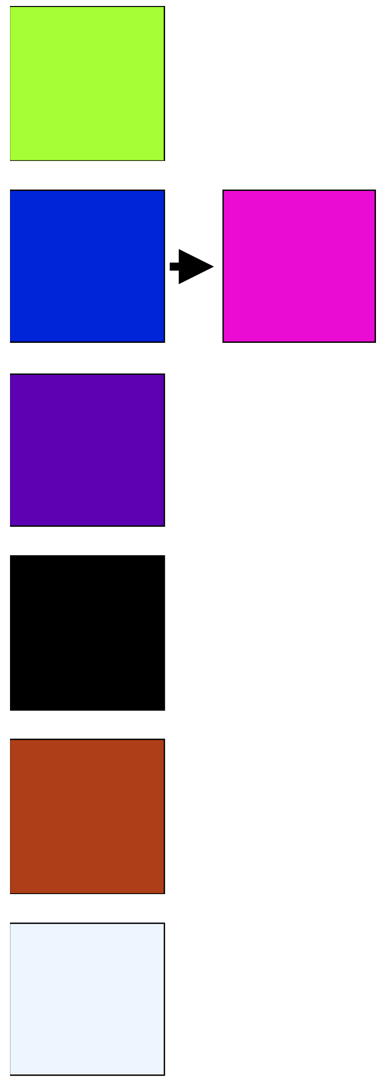
LBC

Global Recoloring Comparison

Original



Chang et al. 2015



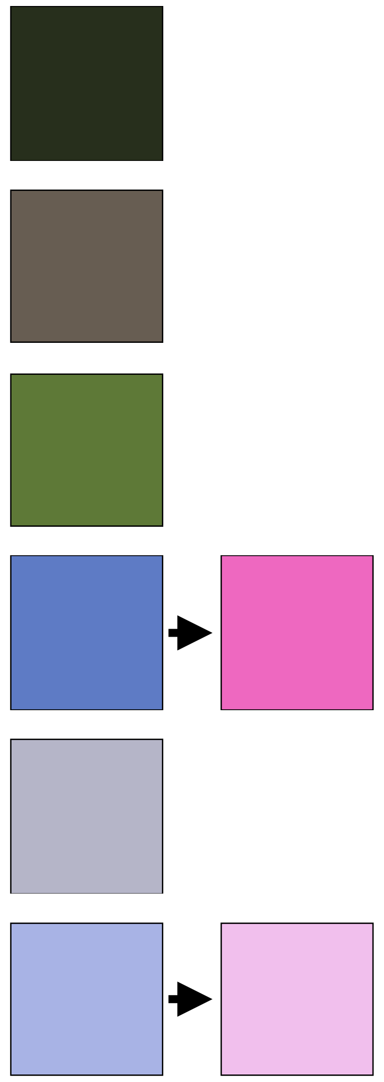
Ours

MVC

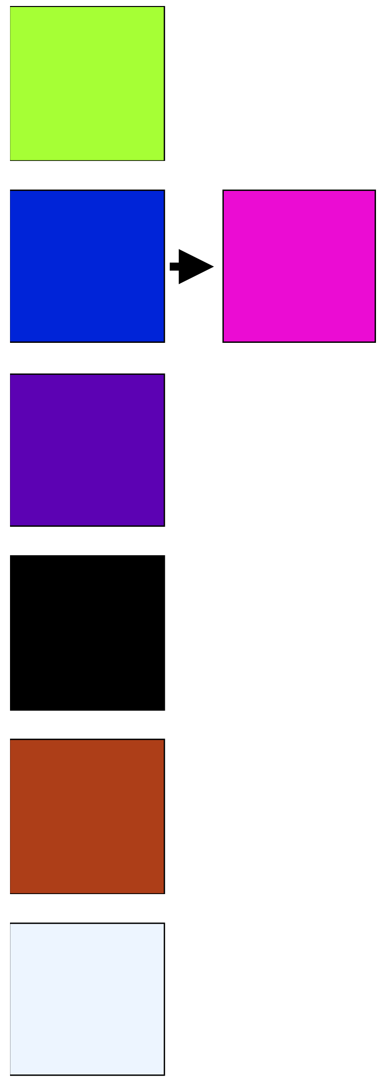
LBC

Global Recoloring Comparison

Original

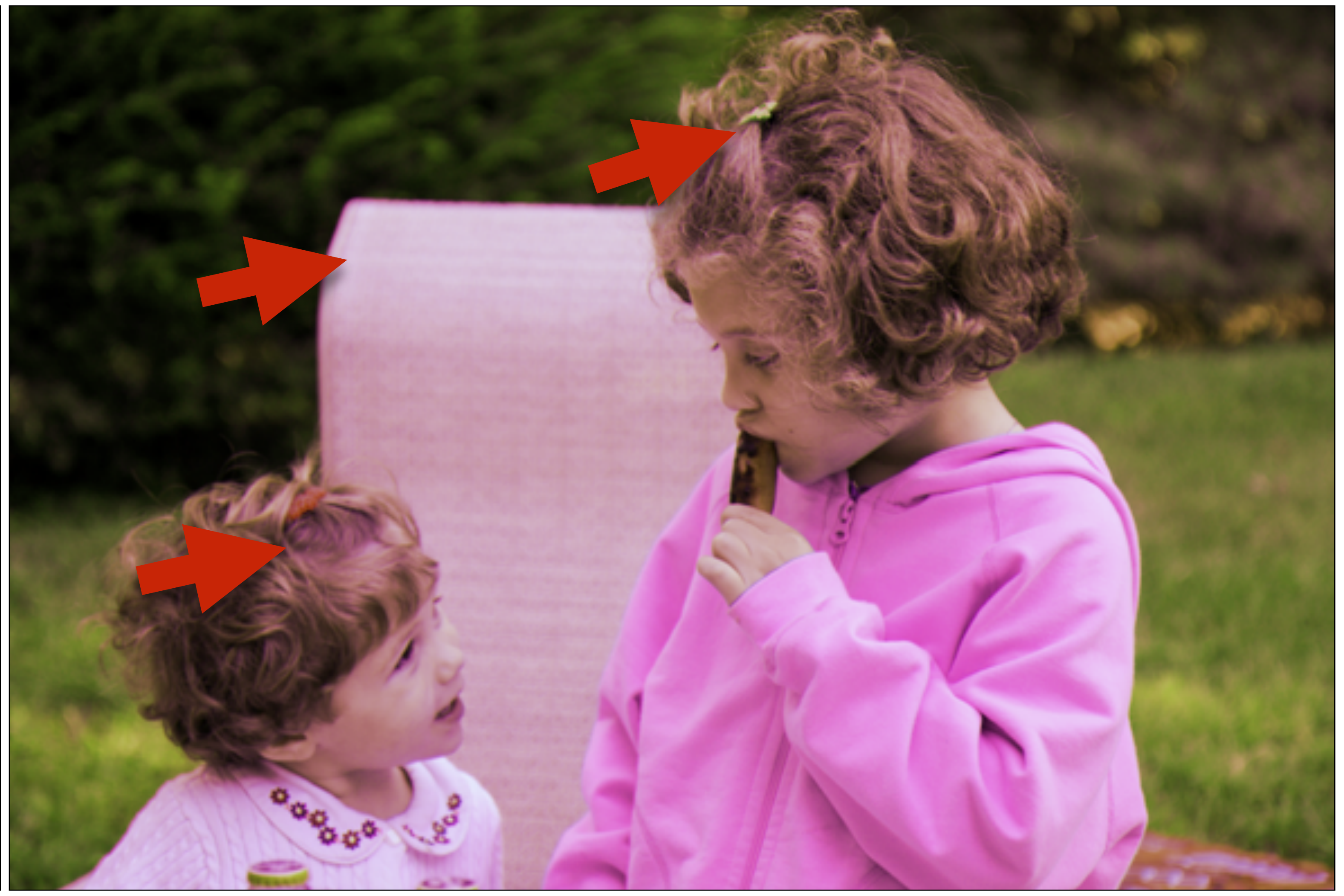


Chang et al. 2015



Ours

MVC



LBC

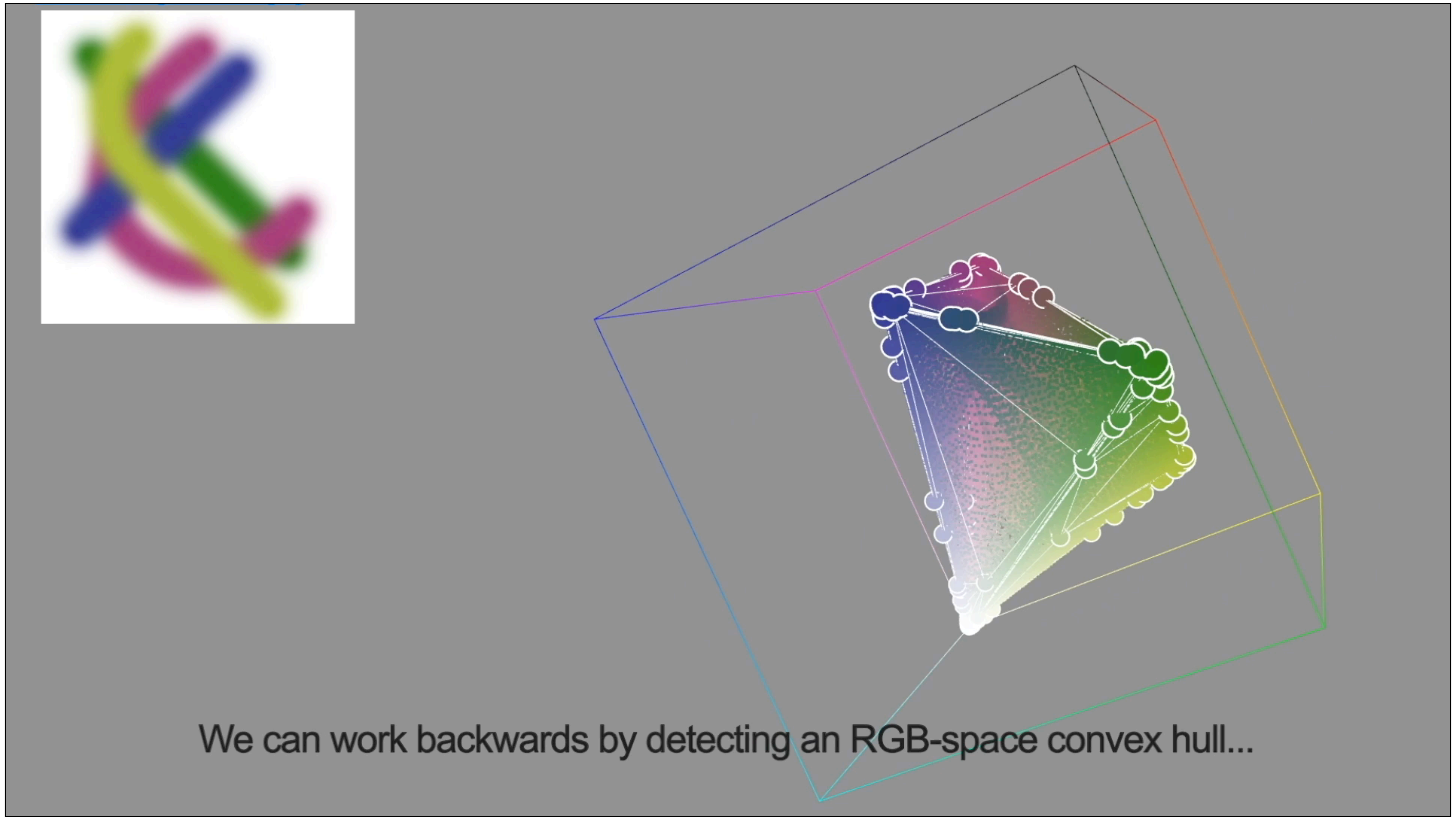
Global Recoloring Comparison

Global Recoloring Comparison

Global Recoloring Comparison

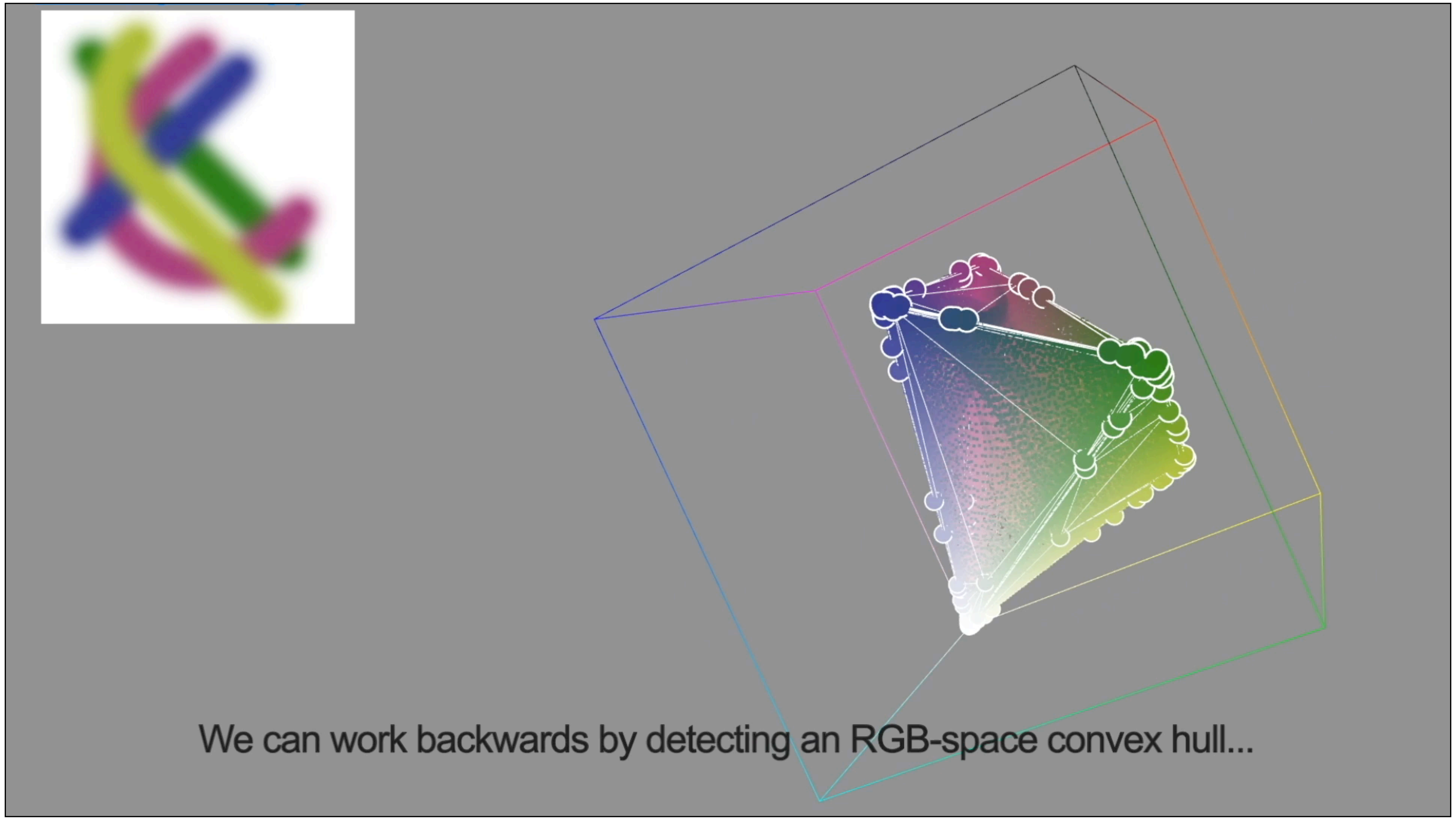
Summary

- 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.



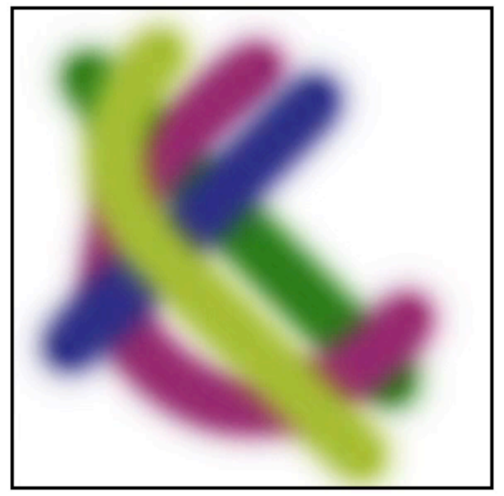
Summary

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

Summary

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

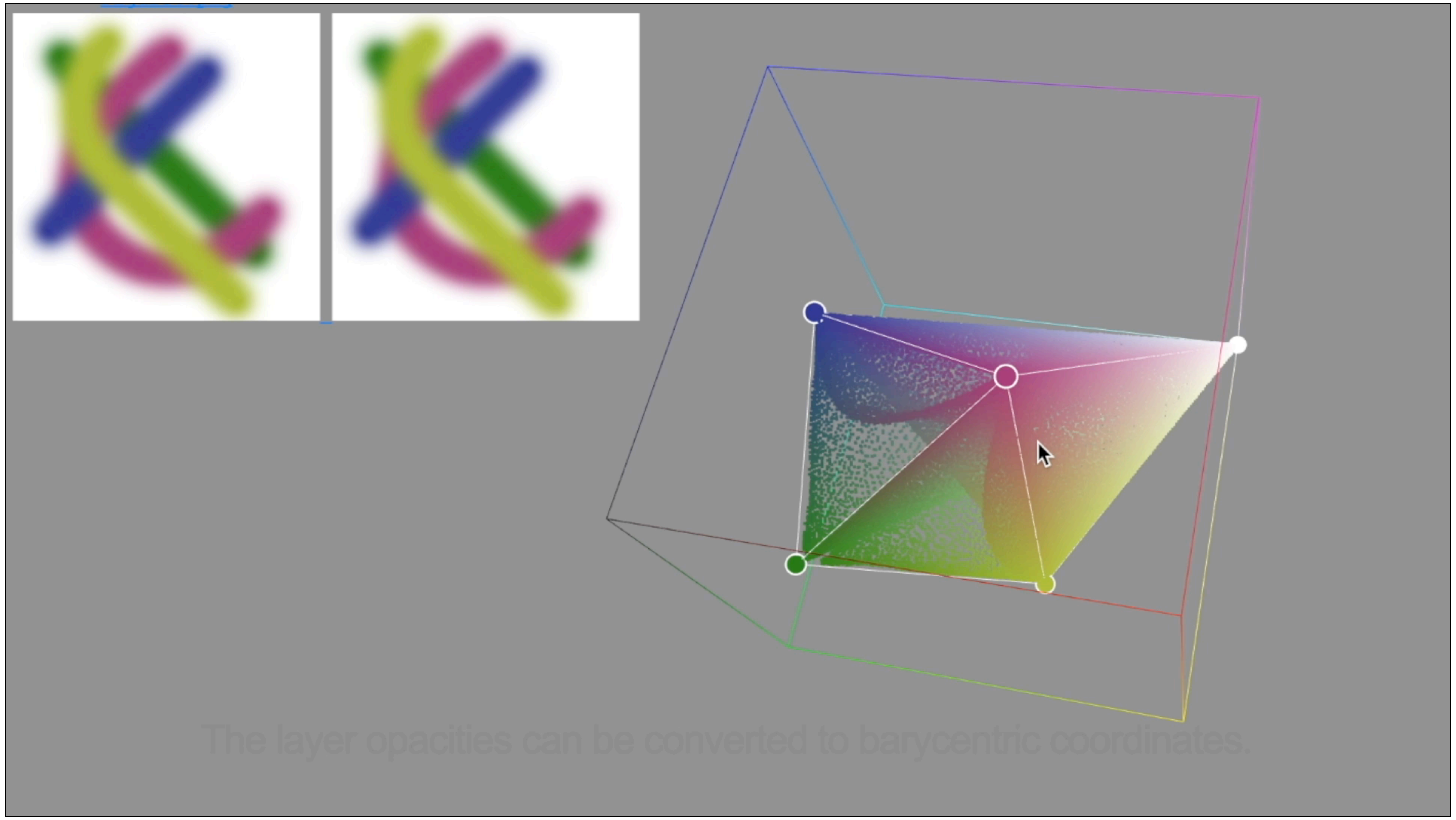
Original



We then solve an optimization problem to extract translucent layers.

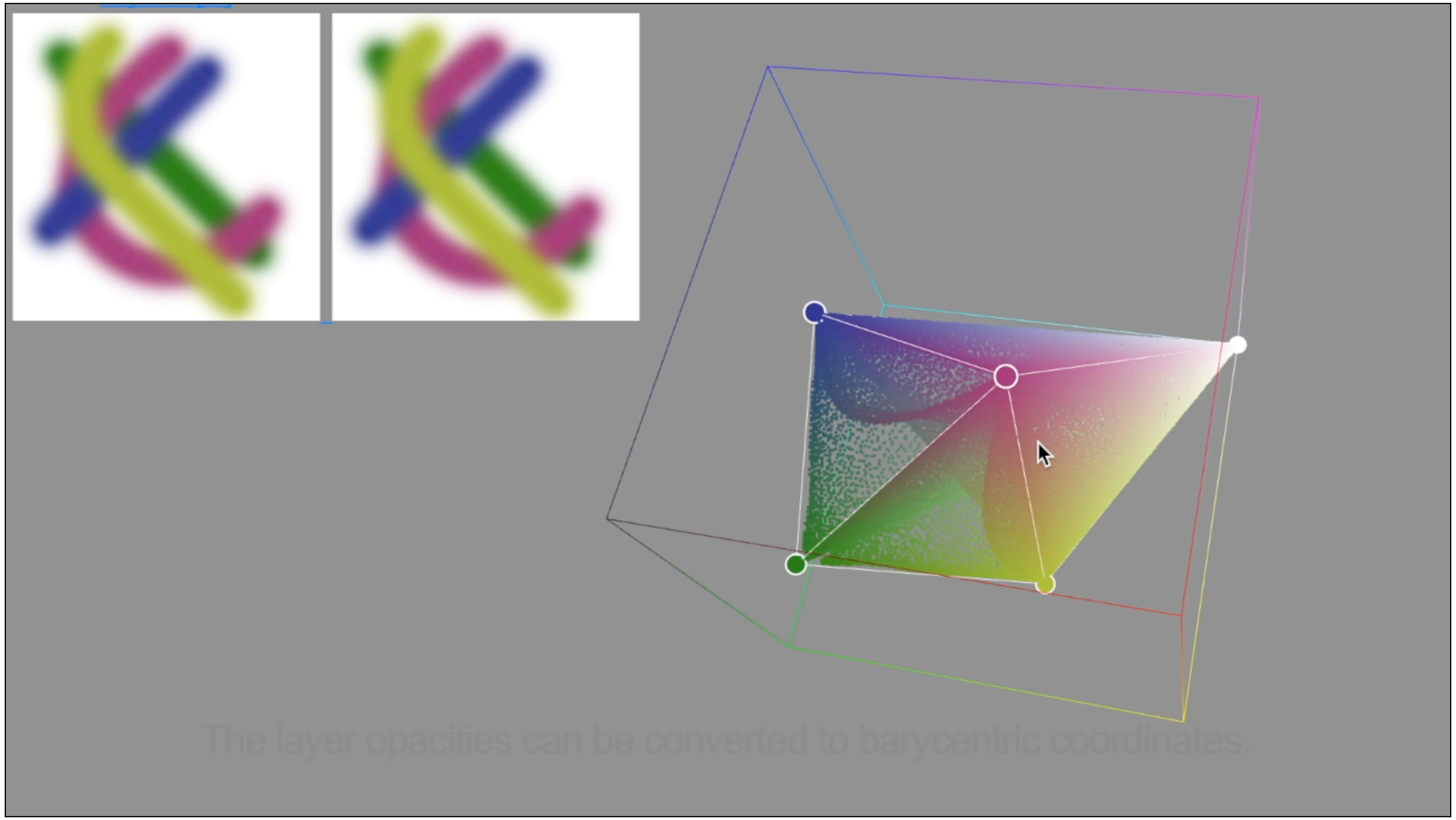
Summary

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



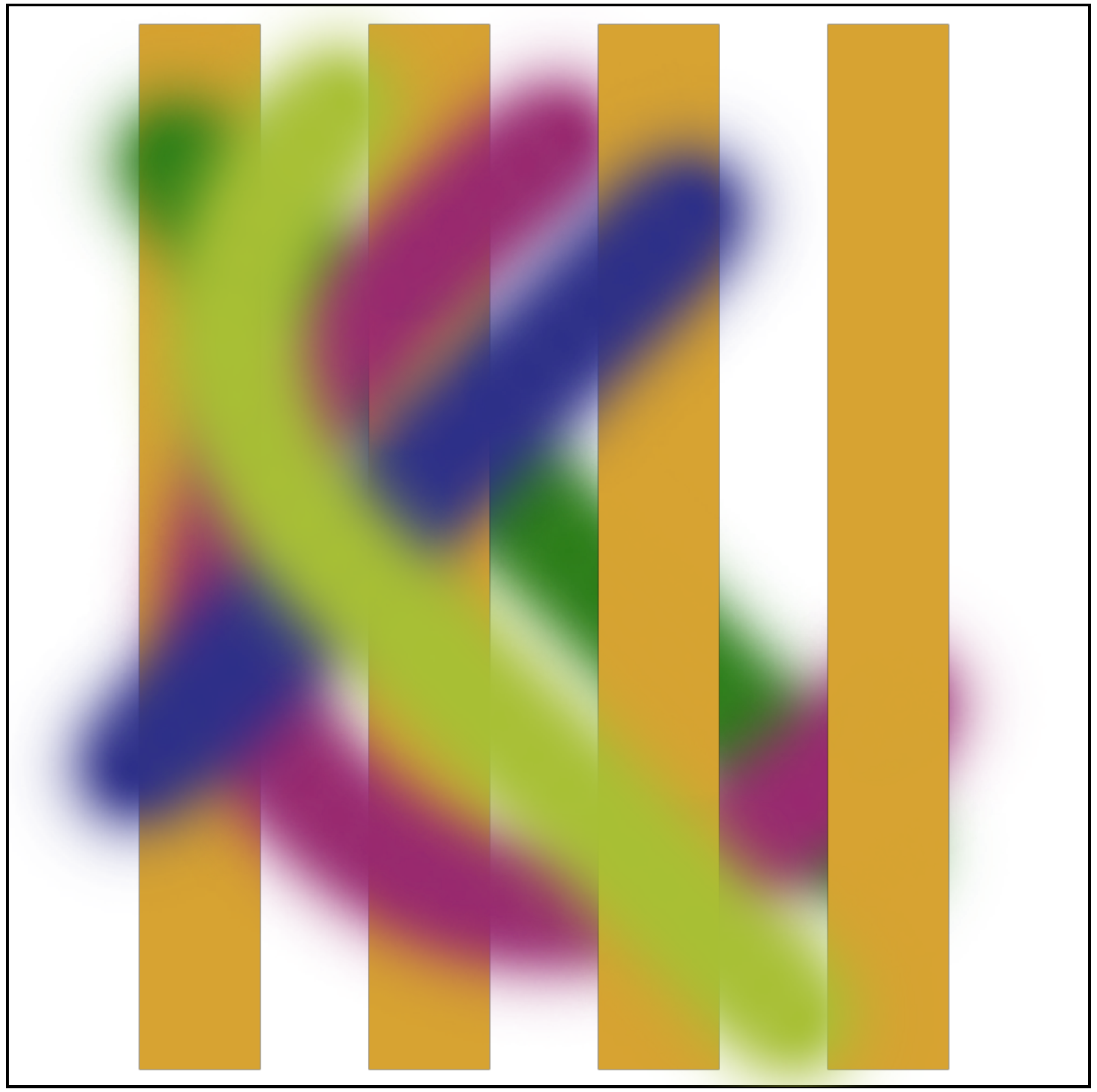
Summary

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



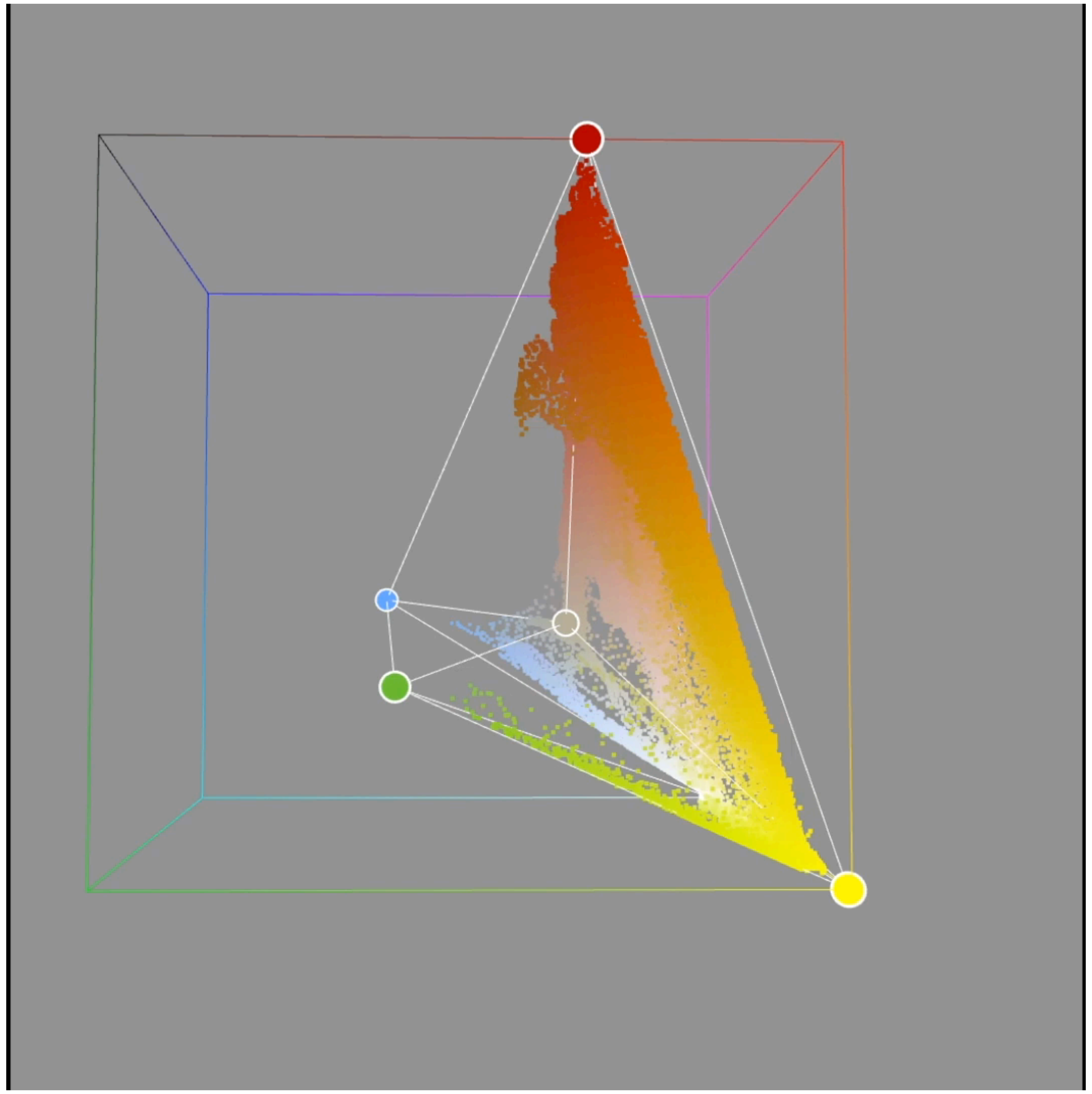
Limitation

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



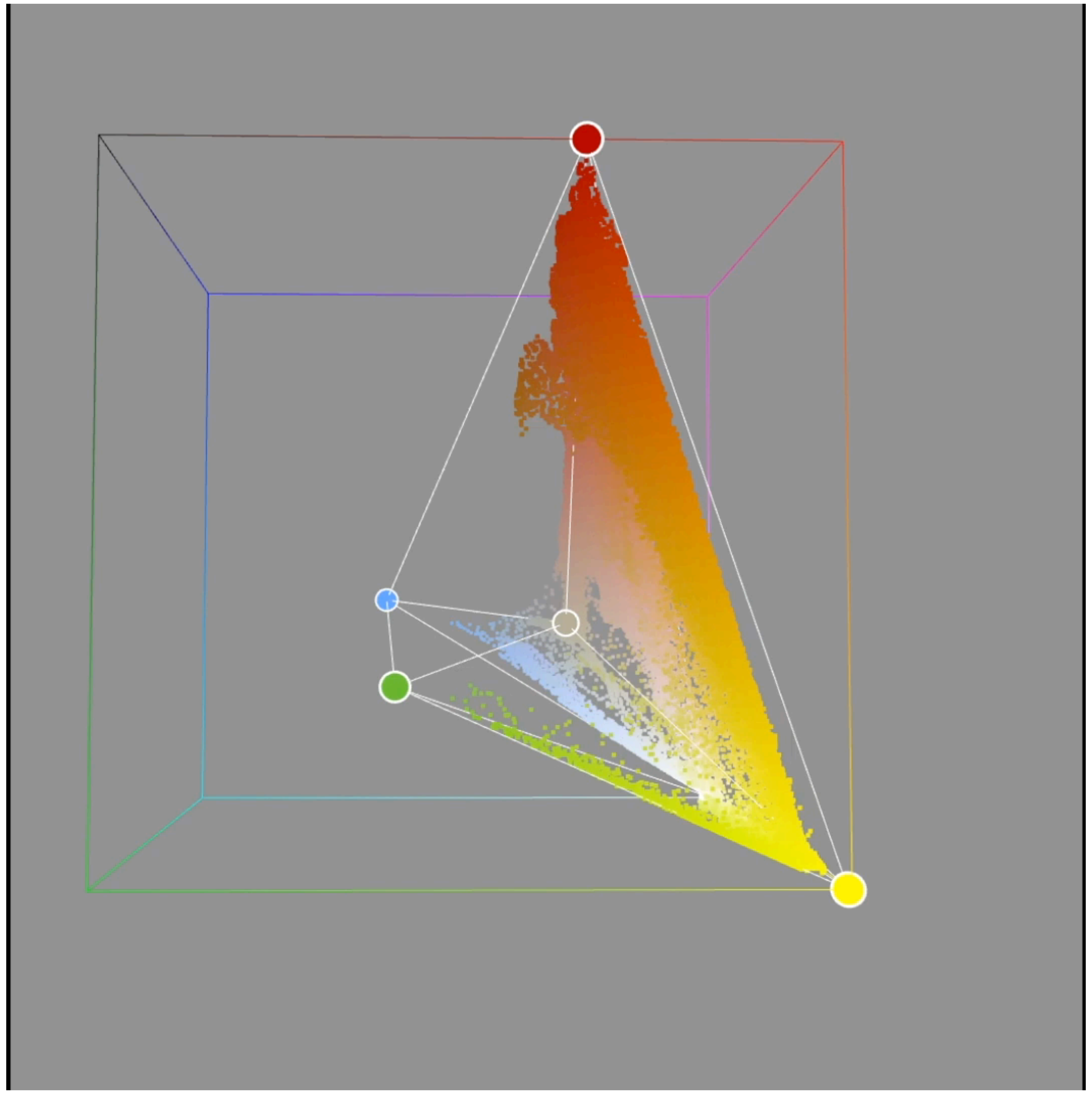
Limitation

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



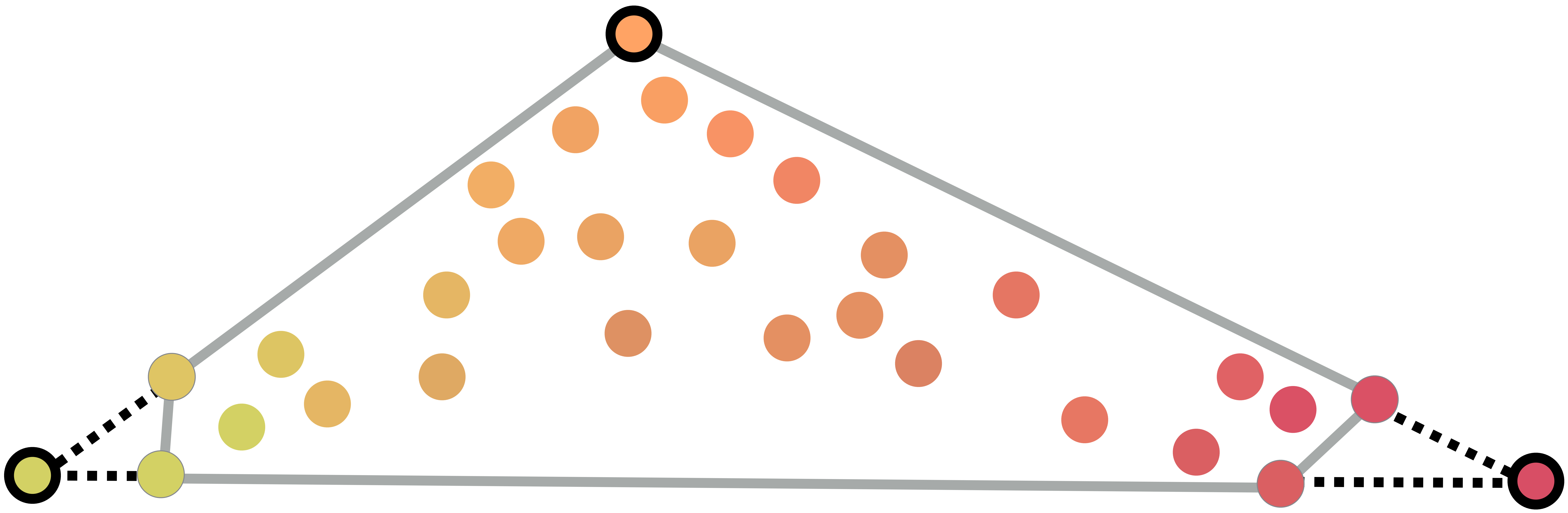
Limitation

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



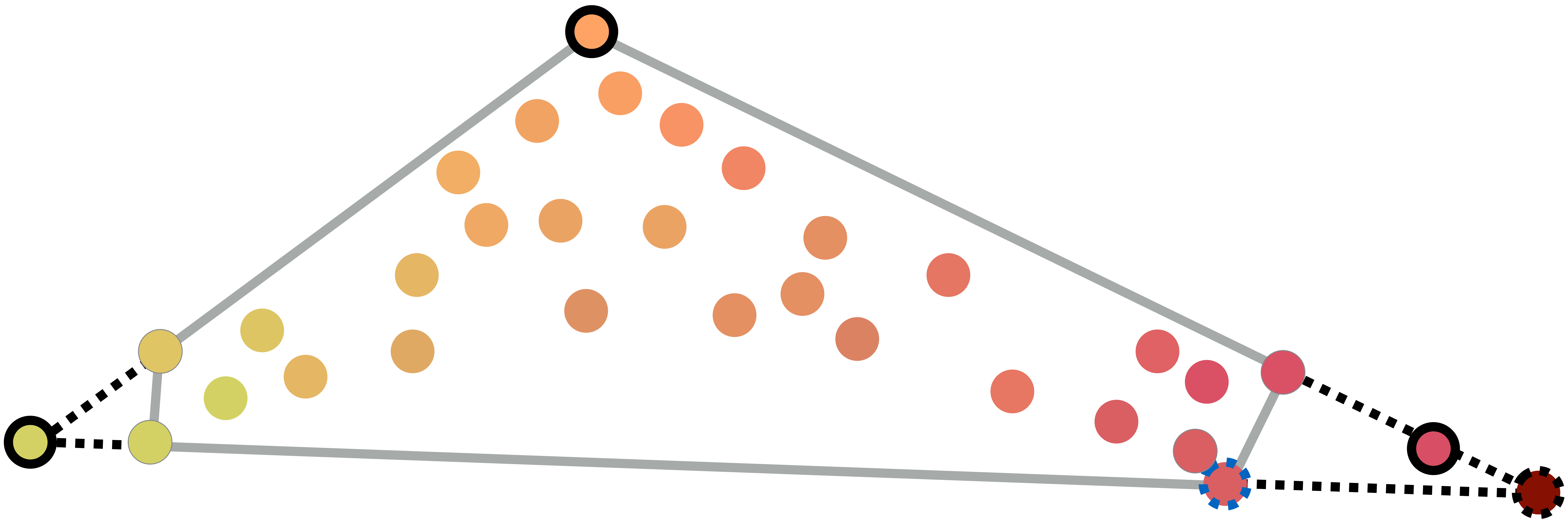
Limitation

- 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).

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

image	width × height	opacity optimization			
		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

image	width × height	opacity optimization 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

image	width × height	opacity optimization: runtime (seconds)	opacity optimization: RMSE	opacity optimization: median error	opacity optimization: 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!

Decompose Image into layers

Solve an optimization problem, with some regularization terms

Decompose Image into layers

Solve an optimization problem, with some regularization terms

$$E = w_{polynomial}E_{polynomial} + w_{opaque}E_{opaque} + w_{spatial}E_{spatial}$$

Decompose Image into layers

Solve an optimization problem, with some regularization terms

$$E = w_{polynomial} E_{polynomial} + w_{opaque} E_{opaque} + w_{spatial} E_{spatial}$$

$$E_{polynomial} = \left\| \mathbf{c}_n - \mathbf{p} + \sum_{i=1}^{n} \left[(\mathbf{c}_{i-1} - \mathbf{c}_i) \prod_{j=i}^{n} (1 - \alpha_j) \right] \right\|^2$$

Decompose Image into layers

Solve an optimization problem, with some regularization terms

$$E = w_{polynomial} E_{polynomial} + w_{opaque} E_{opaque} + w_{spatial} E_{spatial}$$

$$E_{polynomial} = \left\| \mathbf{c}_n - \mathbf{p} + \sum_{i=1}^{n} \left[(\mathbf{c}_{i-1} - \mathbf{c}_i) \prod_{j=i}^{n} (1 - \alpha_j) \right] \right\|^2$$

$$E_{opaque} = \frac{1}{n} \sum_{i=1}^{n} -(1 - \alpha_i)^2$$

Decompose Image into layers

Solve an optimization problem, with some regularization terms

$$E = w_{polynomial} E_{polynomial} + w_{opaque} E_{opaque} + w_{spatial} E_{spatial}$$

$$E_{polynomial} = \left\| \mathbf{c}_n - \mathbf{p} + \sum_{i=1}^{n} \left[(\mathbf{c}_{i-1} - \mathbf{c}_i) \prod_{j=i}^{n} (1 - \alpha_j) \right] \right\|^2$$

$$E_{opaque} = \frac{1}{n} \sum_{i=1}^{n} -(1 - \alpha_i)^2$$

$$E_{spatial} = \frac{1}{n} \sum_{i=1}^{n} (\nabla \alpha_i)^2$$

Decompose Image into layers

Solve an optimization problem, with some regularization terms

$$E = w_{polynomial} E_{polynomial} + w_{opaque} E_{opaque} + w_{spatial} E_{spatial}$$

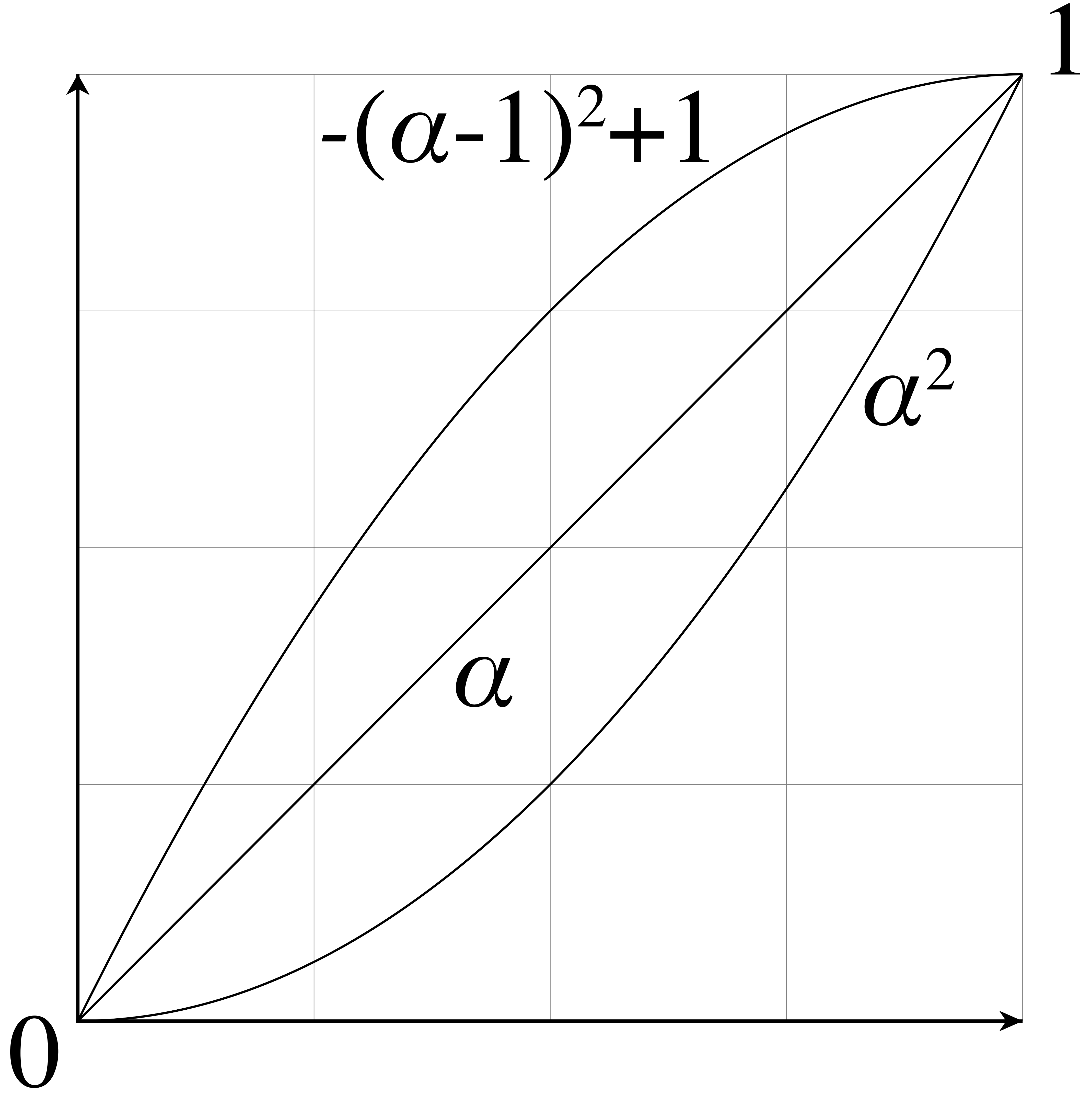
$$E_{polynomial} = \left\| \mathbf{c}_n - \mathbf{p} + \sum_{i=1}^{n} \left[(\mathbf{c}_{i-1} - \mathbf{c}_i) \prod_{j=i}^{n} (1 - \alpha_j) \right] \right\|^2$$

$$E_{opaque} = \frac{1}{n} \sum_{i=1}^{n} -(1 - \alpha_i)^2$$

$$E_{spatial} = \frac{1}{n} \sum_{i=1}^{n} (\nabla \alpha_i)^2$$

$$w_{polynomial} = 375, w_{opaque} = 1, w_{spatial} = 100.$$

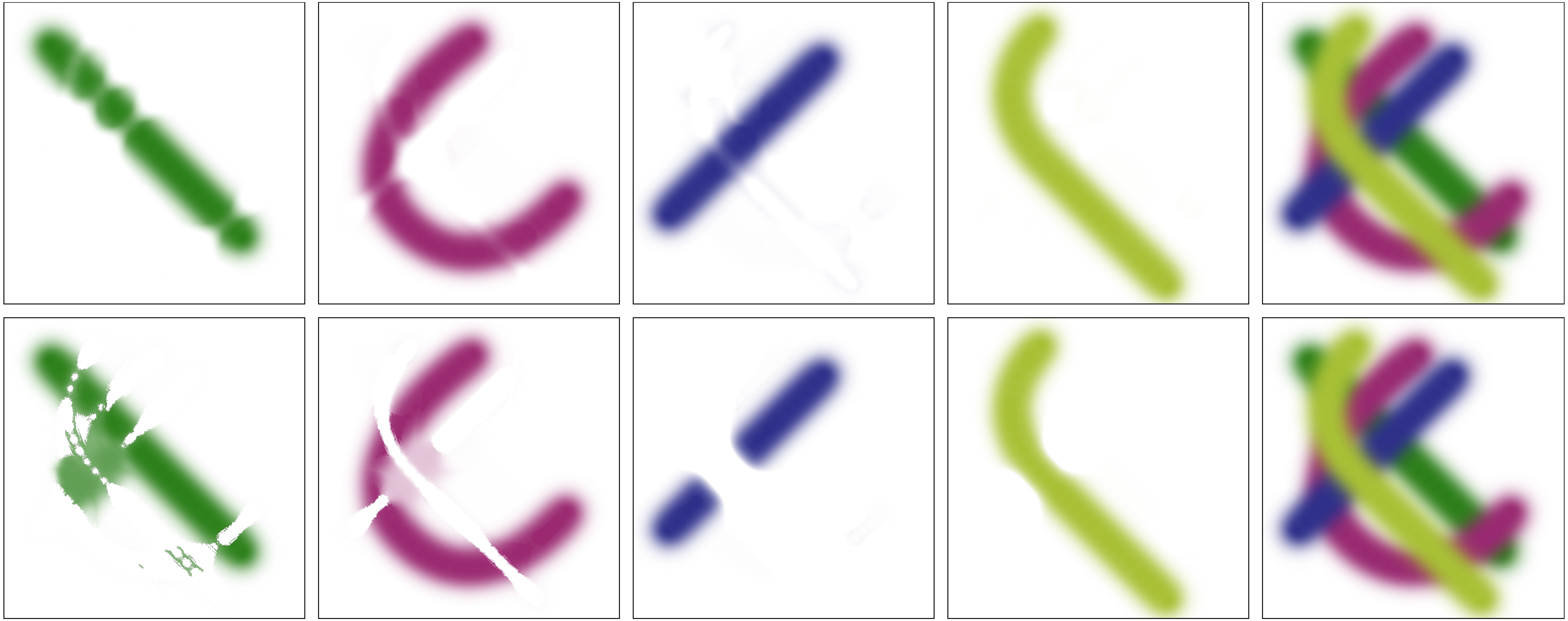
Our sparse regularization term



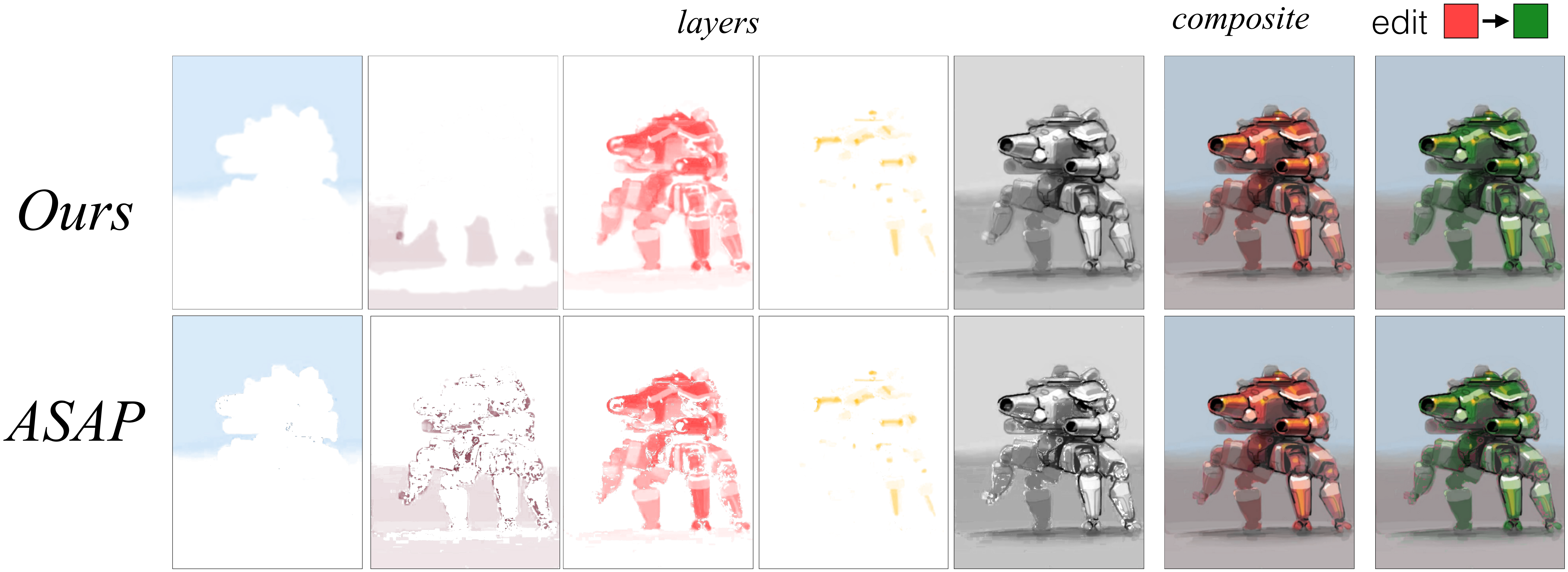
As-sparse-as-possible(ASAP)

Ours

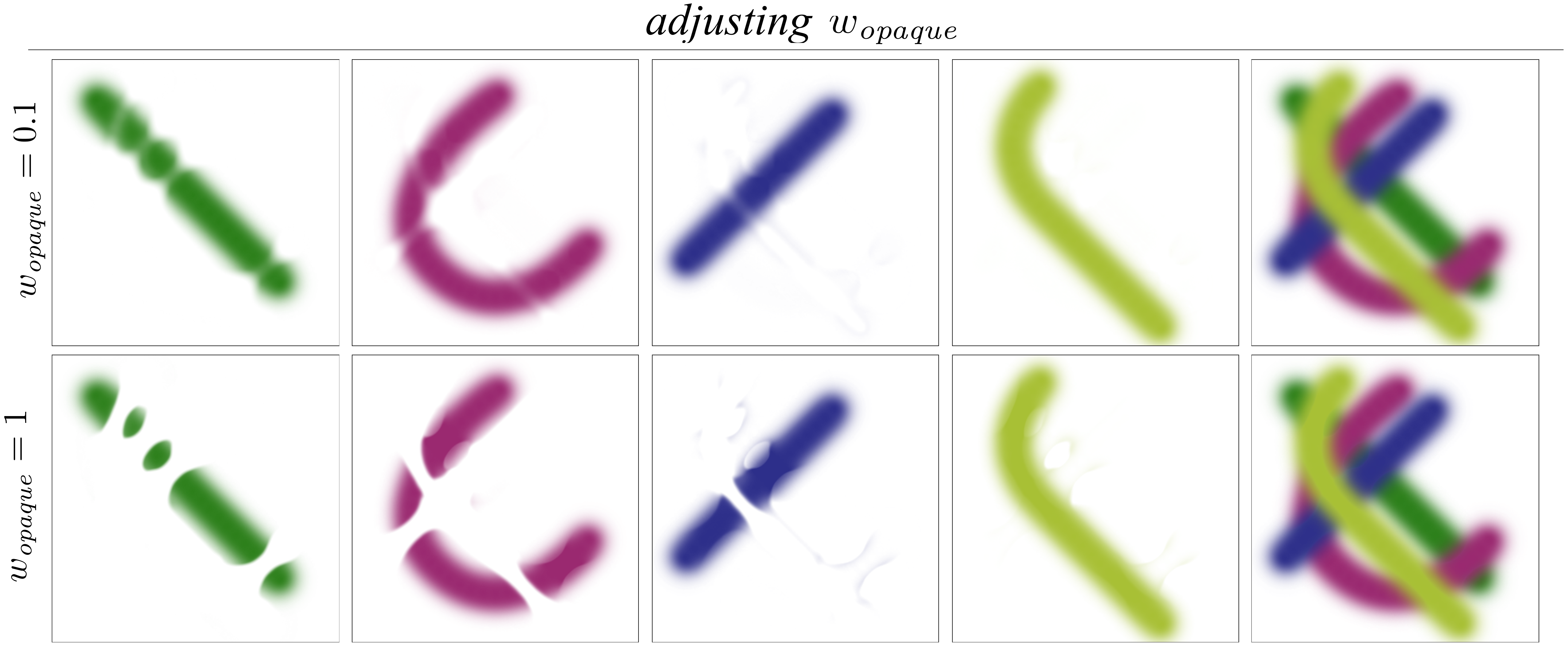
ASAP



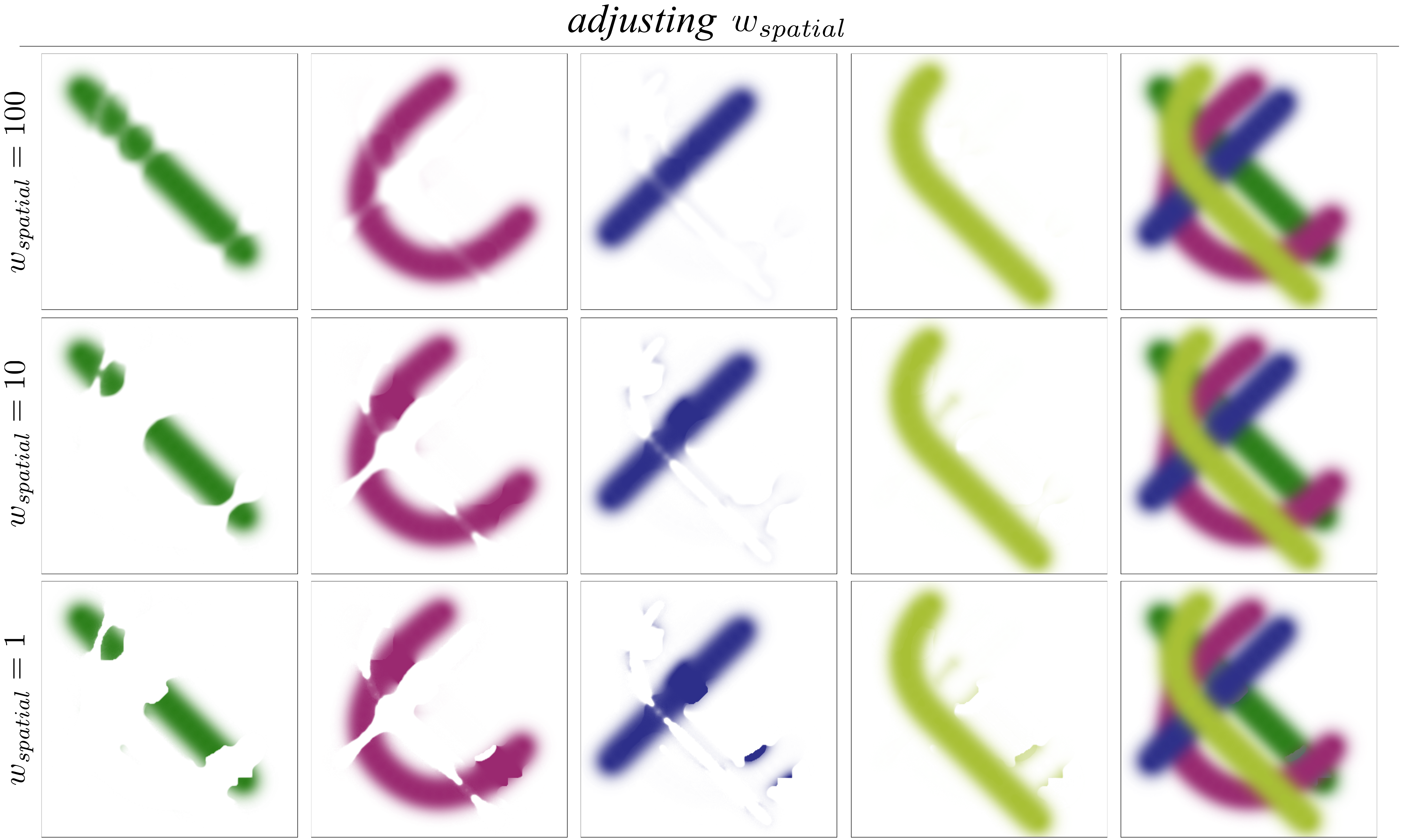
As-sparse-as-possible(ASAP)



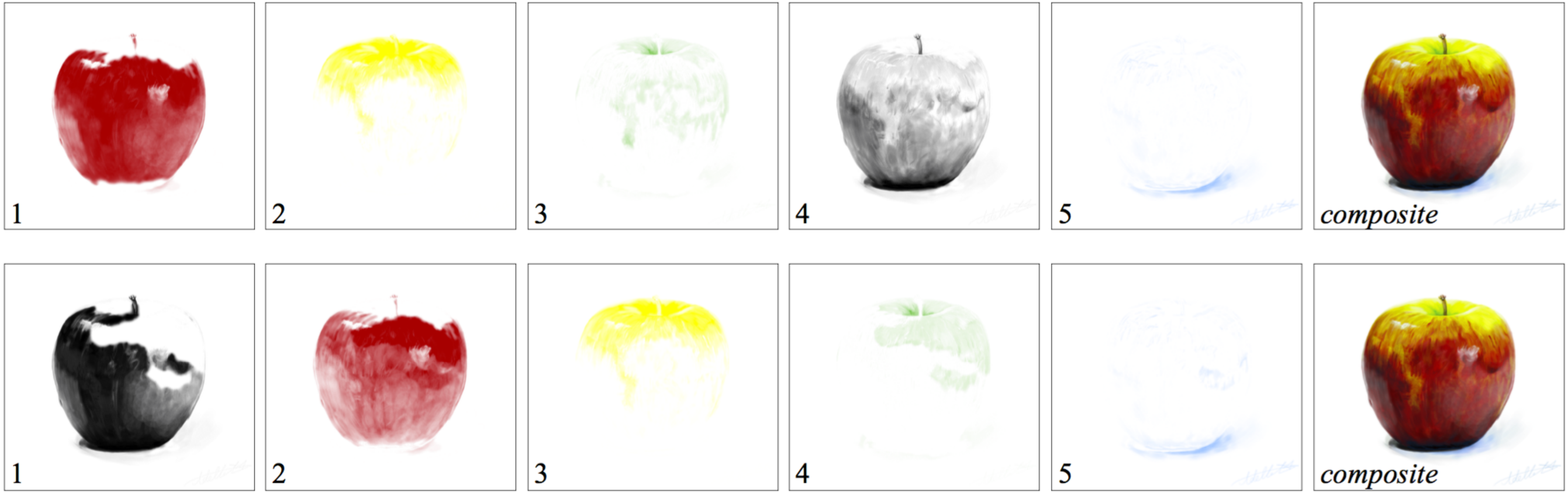
optimization parameters influence



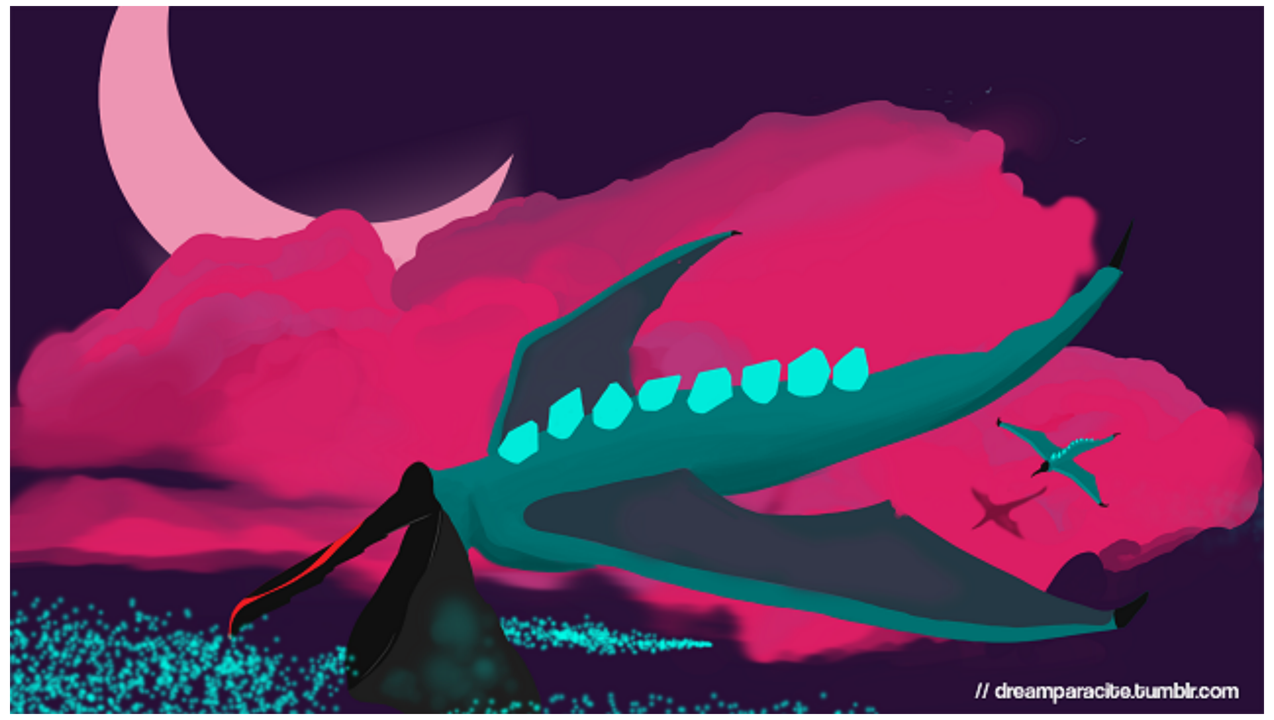
optimization parameters influence



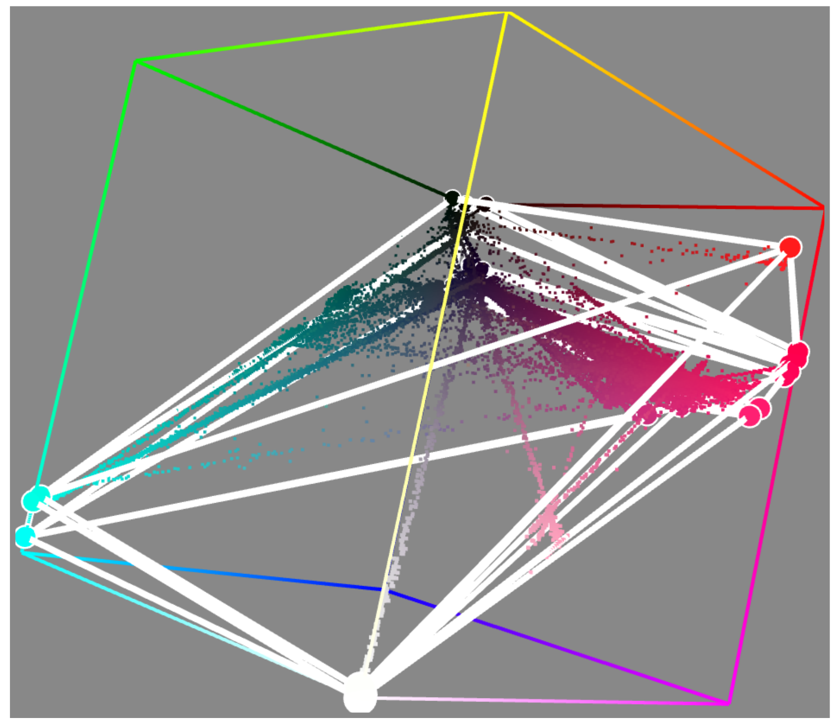
Layer order influence



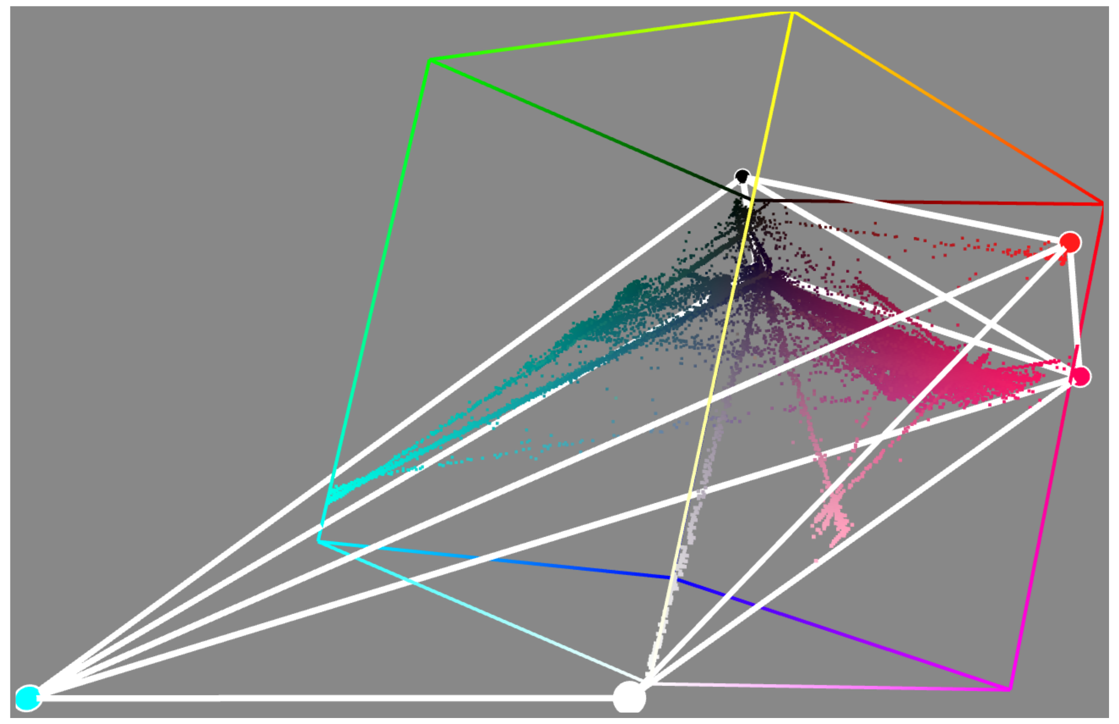
Post vertex brute force optimization led to an improvement in vertex positions.



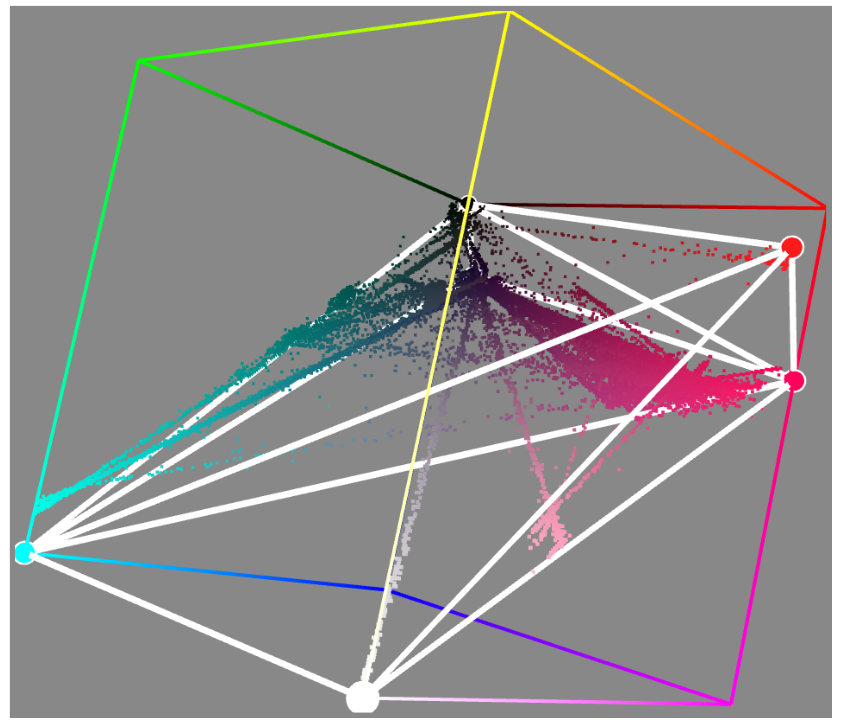
input image



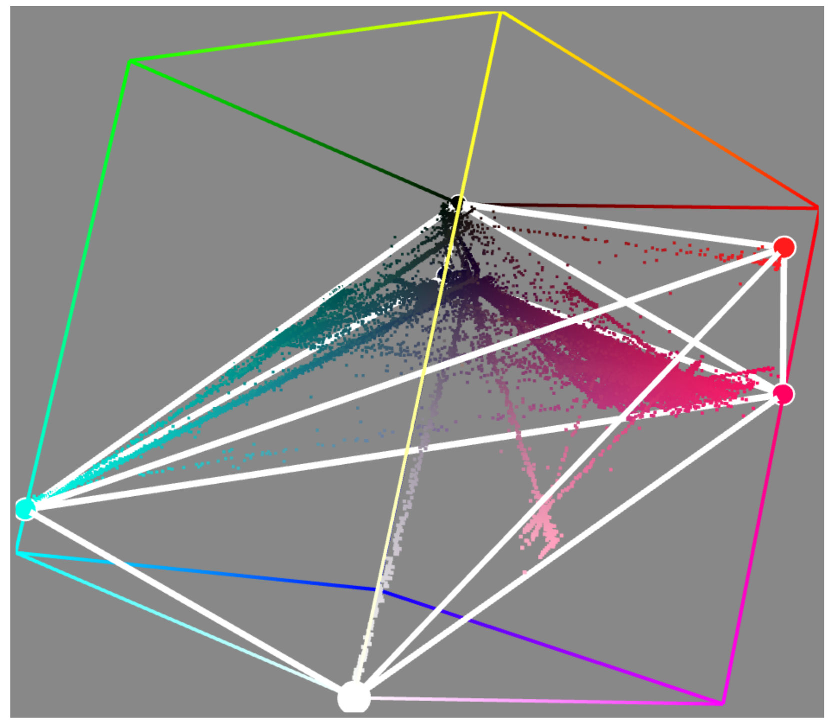
convex hull



simplified hull with invalid colors



projected hull



optimized hull

Extract additive mixing layers using our optimization

Extract additive mixing layers using our optimization

E =

$\|$original − reconstructed image$\|^2$ $\sum ||P_i - w_{ij}C_j||^2$

+

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

+

Mixing weights spatial smoothness (**Laplacian**)

Spectral Matting [Levin et al. 2008]