Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry

Jianchao Tan  George Mason University
Jose Echevarria  Adobe Research
Yotam Gingold  George Mason University
Motivation: Layers Organize Images
Motivation: Layers Organize Images
Motivation: Layers Organize Images
Input

Goal
Subproblems

1. Palette extraction

2. Palette-based layer decomposition
Related Work

- Palette extraction for image editing
  - Shapira et al. [2009]
  - O'Donovan et al. [2011]
  - Lin et al. [2013]
  - Gerstner et al. [2013]
  - Chang et al. [2015]
  - Tan et al. [2016]
Related Work

• Order-dependent translucent layers
  • Richardt et al. [2014]
  • Tan et al. [2015]
  • Tan et al. [2016]
Related Work

- Order-independent additive-mixing layers

- Lin et al. [2017]; Zhang et al. [2017], Aksoy et al. [2017].

Unmixing-Based Soft Color Segmentation for Image Manipulation [Aksoy et al. 2017]
Related Work

- Physically-based layers
  - Abed et al. [2014]; Tan et al. [2015]; Aharoni-Mack et al. [2017]; Tan et al. [2018].
Our approach
Our approach

• Geometry-based convex palettes
Our approach

• Geometry-based convex palettes
  • Simpler
Our approach

• Geometry-based convex palettes
  • Simpler
  • More general
Our approach

• Geometry-based convex palettes
  • Simpler
  • More general
Our approach

- Geometry-based convex palettes
  - Simpler
  - More general

- Additive-mixing layers
Our approach

• Geometry-based convex palettes
  • Simpler
  • More general

• Additive-mixing layers
  • Single colors
Our approach

• Geometry-based convex palettes
  • Simpler
  • More general

• Additive-mixing layers
  • Single colors
  • More general
Palette extraction
Convex hulls in RGB

- Image colors show a convex structure in RGB [Tan et al. 2016]
Palette Size

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

9 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.
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.
Palette Size

- Our automatic error-bound simplification

5 vertices
Image Decomposition
Extracting mixing weights

image

RGB-space

Optimization

palette
Extracting mixing weights

- Slow for high resolutions
Extracting mixing weights

- Slow for high resolutions
- Many parameters to tune
Extracting mixing weights

- Slow for high resolutions
- Many parameters to tune
- Per-image parameters
Extracting mixing weights

- Slow for high resolutions
- Many parameters to tune
- Per-image parameters
Extracting mixing weights

- Slow for high resolutions
- Many parameters to tune
- Per-image parameters
Extracting mixing weights

image

RGB-space

Optimization

palette
Extracting mixing weights

image

RGB-space

Optimization

Generalized Barycentric Coordinates

palette
Extracting mixing weights

- Fast

RGB-space:

Optimization

Palette

Generalized Barycentric Coordinates
Extracting mixing weights

- Fast
- No parameters to tune
Extracting mixing weights

- Fast
- No parameters to tune
- Does not guarantee spatial smoothness
Extracting mixing weights

- Fast
- No parameters to tune
- Does not guarantee spatial smoothness
Extracting mixing weights

- Fast
- No parameters to tune
- Does not guarantee spatial smoothness

RGB-space

Optimization

Generalized Barycentric Coordinates
Extracting mixing weights

image

palette

RGB-space

Generalized Barycentric Coordinates

Optimization
Extracting mixing weights

image

RGB-space

Optimization

Generalized Barycentric Coordinates

palette
(R, G, B, x, y)
RGBXY palette
(projected to RGB)
Two-level decomposition

RGB palette → mixing weights $W$ → image
Two-level decomposition
Two-level decomposition

RGB palette

RGBXY vertices (projected to RGB)

image
Two-level decomposition

RGB palette

RGBXY vertices (projected to RGB)

$W_{RGBXY}$

image
Delaunay Tessellation in RGBXY space

Extract barycentric mixing weights $W_{RGBXY}$
Delaunay Tessellation in RGBXY space

Extract barycentric mixing weights $W_{RGBXY}$
Delaunay Tessellation in RGBXY space

Extract barycentric mixing weights $W_{RGBXY}$
Delaunay Tessellation in RGBXY space

Extract barycentric mixing weights $W_{RGBXY}$
Delaunay Tessellation in RGBXY space

Extract barycentric mixing weights $W_{RGBXY}$
Delaunay Tessellation in RGBXY space

Extract barycentric mixing weights $W_{\text{RGBXY}}$
Two-level decomposition

RGB palette

RGBXY vertices (projected to RGB)

image
Two-level decomposition

RGB palette

$W_{RGB}$

RGBXY vertices (projected to RGB)

image
Extract barycentric mixing weights $W_{RGB}$

Tessellation in RGB space

RGB simplex
Tessellation in RGB space

Extract barycentric mixing weights $W_{RGB}$
Tessellation in RGB space

Extract barycentric mixing weights $W_{RGB}$

RGB simplex
Tessellation in RGB space

Extract barycentric mixing weights $W_{\text{RGB}}$
Tessellation in RGB space

Extract barycentric mixing weights $W_{RGB}$
Two-level decomposition

RGB palette $W_{RGB}$ RGBXY vertices (projected to RGB) $W_{RGBXY}$ image
Two-level decomposition

\[ W = W_{RGB} \times W_{RGBXY} \]
Tessellation in RGB space
Tessellation in RGB space

Delaunay tessellation
Tessellation in RGB space

Star tessellation
Two-level decomposition

\[ W = W_{\text{RGB}} \times W_{\text{RGBXY}} \]
Two-level decomposition

\[ W = W_{\text{RGB}} \ast W_{\text{RGBXY}} \]

Palette updates
Fixed

RGB palette

image
Two-level decomposition

\[ W = W_{RGB} \times W_{RGBXY} \]
Two-level decomposition

\[ W = W_{\text{RGB}} \times W_{\text{RGBXY}} \]

Palette updates

Fixed

RGB palette
Two-level decomposition

\[ W = W_{RGB} \times W_{RGBXY} \]

Updating \( W_{RGB} \) is independent of image size.
Two-level decomposition

\[ W = W_{RGB} * W_{RGBXY} \]

Palette updates

Fixed

Updating \( W_{RGB} \) is independent of image size.

Other methods need to re-compute everything from scratch.
our approach,

-updating palette

Performance

Quadratic [Tan et al. 2016]

[Aksoy et al. 2017]

[Zhang et al. 2017]

our approach

our approach, updating palette
Performance

Quadratic [Tan et al. 2016]

Zhang et al. 2017

Aksoy et al. 2017

our approach

our approach, updating palette

100 MP!
our approach,
updating palette

Performance

Quadratic [Tan et al. 2016]

100 MP!
Real-Time Editing!

[Zheng et al. 2017]

[Aksoy et al. 2017]
Python Implementation

```python
from numpy import *
from scipy.spatial import ConvexHull, Delaunay
from scipy.sparse import coo_matrix

def RGBXY_weights( RGB_palette, RGBXY_data):
    RGBXY_hull_vertices = RGBXY_data[ ConvexHull( RGBXY_data ).vertices ]
    RGBXY = Delaunay_coordinates( RGBXY_hull_vertices, RGBXY_data )
    # Optional: Project outside RGBXY_hull_vertices[:,:3] onto RGB_palette convex hull.
    RGB = Star_coordinates( RGB_palette, RGBXY_hull_vertices[:,:3] )
    return RGBXY.dot( RGB )

def Star_coordinates( vertices, data):
    star = argmin( linalg.norm( vertices, axis=1 ) )
    hull = ConvexHull( vertices )
    simplices = [ [star] + list(face) for face in hull.simplices if star not in face ]
    barycoords = -1*ones( ( data.shape[0], len(vertices) ) )
    for s in simplices:
        s0 = vertices[s[:1]]
        b = linalg.solve( (vertices[s[1:]]-s0).T, (data-s0).T ).T
        b = append( 1-b.sum(axis=1)[:,None], b, axis=1 )
        mask = (b>=0).all(axis=1)
        barycoords[mask] = 0.
        barycoords[ix_(mask,s)] = b[mask]
    return barycoords

def Delaunay_coordinates( vertices, data):
    # Adapted from Gareth Rees
    tri = Delaunay( vertices )
    simplices = tri.find_simplex(data, tol=1e-6)
    assert (simplices != -1).all() # data contains outside vertices.
    X = tri.transform[simplices, :data.shape[1]]
    Y = data - tri.transform[simplices, data.shape[1]]
    b = einsum( '...jk,...k->...j', X, Y )
    rows = repeat(arange(len(data)).reshape((-1,1)), len(tri.simplices[0])).ravel()
    cols = tri.simplices[0].ravel()
    vals = barycoords.ravel()
    return coo_matrix( (vals,(rows,cols)), shape=(len(data),len(vertices) ) ).tocsr()
```

48 lines of code
Comparisons
Layer quality comparison with [Aksoy et al. 2017]
Layer quality comparison with [Aksoy et al. 2017]
Recoloring comparison with three previous methods

Original

Ours
Recoloring comparison with three previous methods

Original  Aksoy et al. 2017  Ours
Recoloring comparison with three previous methods

Original

Aksoy et al. 2017

Ours
Recoloring comparison with three previous methods

Original  Aksoy et al. 2017  Tan et al. 2016  Ours
Recoloring comparison with three previous methods

Original  Aksoy et al. 2017  Tan et al. 2016  Ours
Recoloring comparison with three previous methods

Original  Aksoy et al. 2017  Tan et al. 2016  Chang et al. 2015  Ours
Recoloring comparison with three previous methods
Demo

Javascript + Python with PyOpenCL
Layer creation from scratch
Layer creation from an automatic palette
Interactive decomposition gives more control to the users.
Interactive decomposition gives more control to the users

original

recoloring with interactively edited palette

recoloring with automatic palette
Interactive decomposition gives more control to the users.
Conclusion

• An extremely efficient approach to layer decomposition via RGBXY geometry
Conclusion

• Our two-level decomposition supports real-time decomposition when palette editing.

\[ W = W_{RGB} \times W_{RGBXY} \]
Conclusion

• It’s important to capture the “line of greys”.

Star tessellation
Limitations

• In isolated cases, the 5D convex hull takes somewhat longer than usual to compute.
Limitations

• Our star tessellation assumes that palette colors are vertices of a convex polyhedron.
  • For palette colors in the interior, must use inferior Delaunay tessellation.
Future Work

• More speed via super-pixels or parallel convex hull algorithms.
Future Work

• Robustness via approximate convex hull algorithms.
Future Work

• Robustness via approximate convex hull algorithms.
Future Work

• Robustness via approximate convex hull algorithms.
Thank You!

• Contact Information:
  • Jianchao Tan: jtan8@gmu.edu
  • Jose Echevarria: echevarr@adobe.com
  • Yotam Gingold: ygingold@gmu.edu

• Project Website (GUI, code, data): https://cragl.cs.gmu.edu/fastlayers/

• Artists: Adelle Chudleigh; Dani Jones; Karl Northfell; Michelle Lee; Adam Saltsman; Yotam Gingold;
  DeviantArt user Sylar113; Fabio Bozzone; Piper Thibodeau; Spencer Nugent; George Dolgikh; DeviantArt
  user Ranivius.

• Sponsors:
  • NSF, Adobe, Google.
Extra slides
Possible questions

• Star triangulation starting from black palette color, what if no black color in extracted palette?

• Does your method require palette to cover all pixel colors when editing palette in GUI? What if I want some palette colors that is inside color point cloud?

• In your performance figure, there are one or two cases that are slower than many others. Can you describe the worst case performance of your method?

• Do you have failure case?

• How do you measure the quality of your layer results and your interactive editing GUI?