Hello everyone, I am Zhengjun Du from Tsinghua University. In this talk, I will introduce a spatial-temporal geometry-based approach to video recoloring.

Motivation

- Palette-based image recoloring is intuitive and simple

- Can we extend it to recolor a video?

Challenge: The palette can change over time!

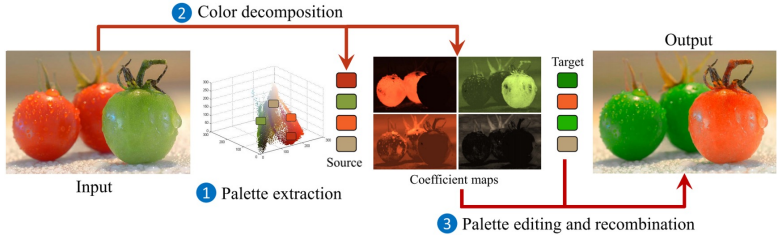
Palette-based image editing is a recent paradigm for color adjustment. These approaches extract a palette with several salient colors from the image to represent the color distribution. The palette provides users with an intuitive set of handles to edit the image. And users can easily recolor the image by modifying its palette colors.

The natural idea is that can we extend the palette-based image recoloring to video editing?

The key challenge is that the palette should capture color changes in the video, so the user can edit them. The palette should also allow the user to introduce color changes over time.

Related work

- Clustering-based methods
  - [Chang et al. 2015; Nguyen et al. 2017; Zheng et al. 2017]



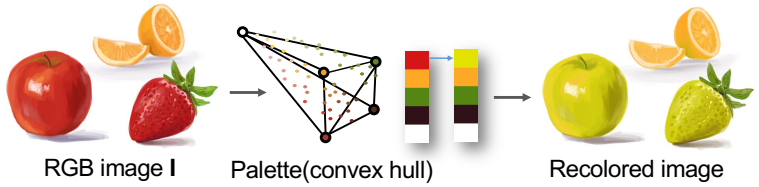
Palette-based image recoloring using color decomposition optimization [Zheng et al. 2017]

As for palette-based image recoloring, there are two main kinds of approaches.

The first is based on clustering, these methods employ k-means clustering of pixels to extract palette colors, these palettes capture the dominant colors in an image. And then decompose pixel into linear combination with the palette, which enables efficient image recoloring.

Related work

- Convex hull-based methods
  - [Tan et al. 2017; Tan et al. 2018; Wang et al. 2019]



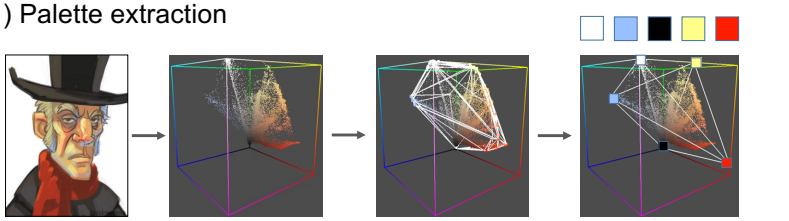
An improved geometric approach for palette-based image decomposition and recoloring [Wang et al. 2019]

The second is based on convex hull. These methods extract palettes from images based on simplified RGB convex hulls. The simplified convex hull has a simple geometric shape, and its vertices intuitively represent palette colors. Pixel colors can be naturally represented as linear combinations of palette colors. Its linear nature makes the recoloring process intuitive, efficient and results in better recoloring quality.

Background

- Our work is based on RGB convex hull-based palette recoloring

1) Palette extraction



Input image → RGB point cloud → RGB convex hull → Simplified RGB convex hull (Palette)

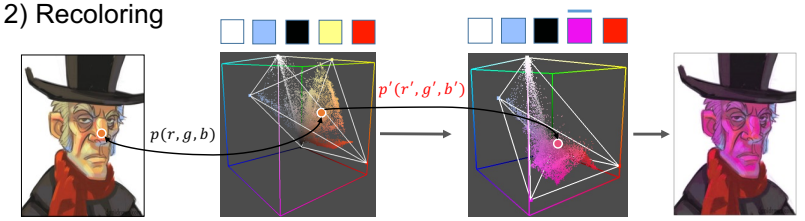
Our work is mainly based on geometry-based image recoloring. This method contains two stages: palette extraction and palette-based recoloring.

Geometry-based palette extraction takes an image as input, and projects all image pixels into 3D RGB space, computes its convex hull, and followed by simplification and optimization. The refined convex hull is used as the palette, and its vertices intuitively act as the palette colors.

Background

- Our work is based on RGB convex hull-based palette recoloring

2) Recoloring



Input image | Palette V | Palette V' | Recolored image

- Mixing weights: $p = \sum_{i=1}^{|V|} w_i V_i \quad 1 = \sum_{i=1}^{|V|} w_i$ Recoloring: $p' = \sum_{i=1}^{|V|} w_i V_i'$

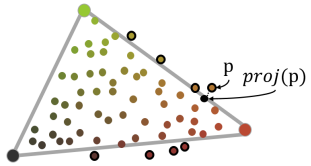
Once we obtain the convex hull, each pixel in the image can be expressed as a weighted sum of its palette colors. Keep the mixing weights fixed; when we modify the palette colors, the pixel is then updated accordingly. When we recompute all pixels to the modified palette, it produces a recolored image.

Background

• Measurement of the convex hull-based palette V

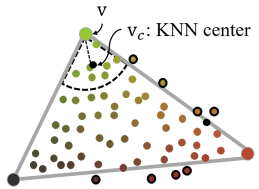
➢ Reconstruction accuracy

$$R(\mathrm{V},\mathrm{I}) = \frac{1}{|\mathrm{I}|}\sum_{\mathrm{p}\in\mathrm{I}} ||\mathrm{p} - proj(\mathrm{p})||$$



➢ Compactness

$$C(\mathrm{V},\mathrm{I}) = \frac{1}{|\mathrm{V}|}\sum_{\mathrm{v}\in\mathrm{V}} ||\mathrm{v} - \mathrm{v}_c||$$



The key of geometry-based image recoloring is to build a simplified convex hull. To get such a convex hull, we take both the reconstruction accuracy and compactness into account.

We measure the reconstruction loss as the average distance of all exterior [ɪkˈstɪəriə(r] pixels to their projection on the convex hull.

and we measure the compactness loss as the average distance of all convex hull vertices to their neighbors' centers.

In general, reconstruction loss enforces the convex hull to enclose as many pixels as possible, and compactness loss enforces the convex hull to wrap the pixels as tight as possible, so that the palette colors are more representative. [ˌreprɪˈzentətɪv]

Background

- Measurement of the convex hull-based palette V

➢ Reconstruction accuracy

$$R(\mathrm{V},\mathrm{I}) = \frac{1}{|\mathrm{I}|}\sum_{\mathrm{p}\in\mathrm{I}} ||\mathrm{p} - proj(\mathrm{p})||$$

➢ Compactness

$$C(\mathrm{V},\mathrm{I}) = \frac{1}{|\mathrm{V}|}\sum_{\mathrm{v}\in\mathrm{V}} ||\mathrm{v} - \mathrm{v}_c||$$

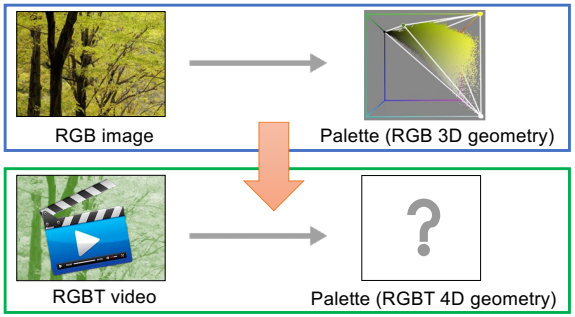
$$L(\mathrm{V},\mathrm{I}) = \lambda R(\mathrm{V},\mathrm{I}) + C(\mathrm{V},\mathrm{I})$$

Image palette overall loss

Reconstruction error and compactness are in tension with each other. The overall loss of the palette is defined as their weighted sum, and λ controls the relative contribution of the two terms.
As a result, the extracted convex hull is with lower reconstruction loss and higher compactness.

Our main idea

- Extend the convex hull-based image recoloring to video scenarios



Geometry-based image recoloring employ a 3D RGB convex hull to extract the palette.
Similarly, our main idea is to extend this method to video scenario, and employ a 4D RGBT geometry to extract the video palette.

Choices of RGBT 4D geometry

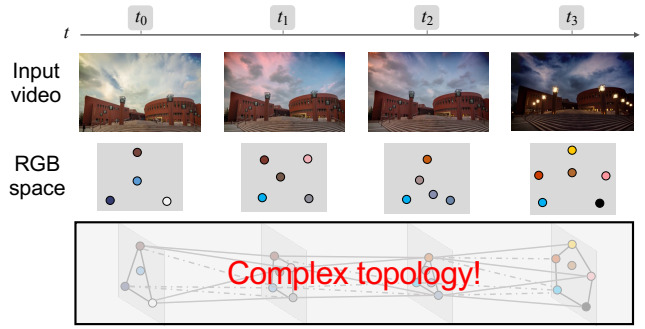
We have four choices to build such a 4D geometry palette in RGBT space.
The 1st approach is to pull a 3D convex hull along temporal dimensional to form a 4D geometry.
But it suffers from poor compactness

Choices of RGBT 4D geometry

RGBT convex hull

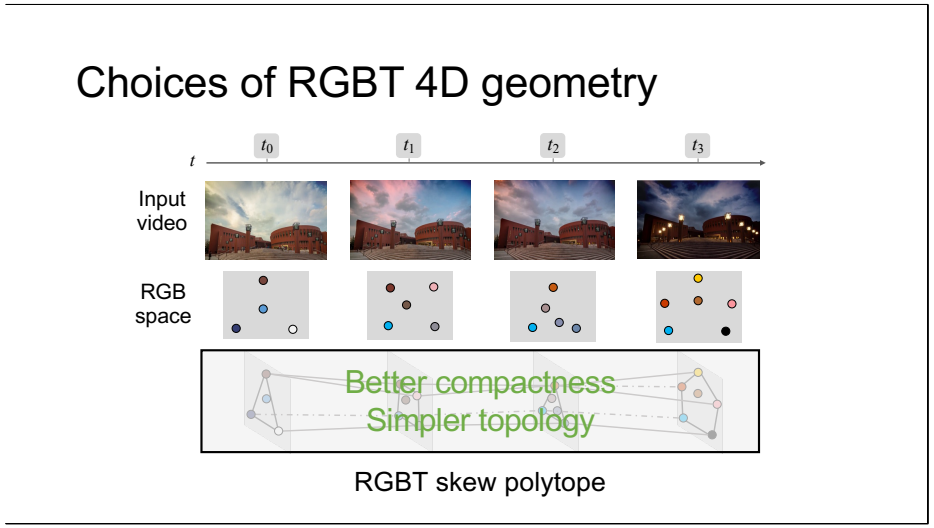
The 2nd approach is to build a 4D RGBT convex hull that encloses all video pixels.
It also suffers from poor compactness

Choices of RGBT 4D geometry



RGBT regular polytope

The 3rd approach is to build a regular 4D RGBT polytope.
Although the RGBT regular polytope looks more compact, its topology is relatively complex.



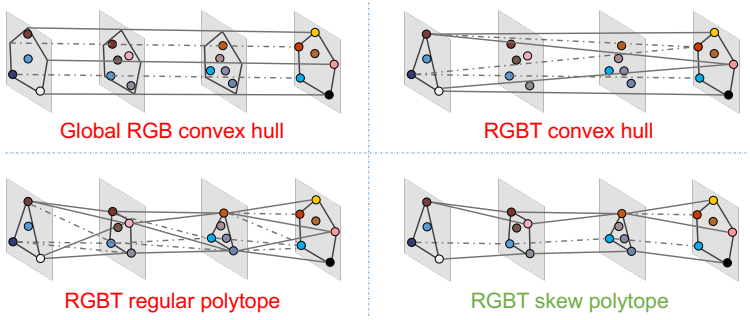
The last approach is to build an RGBT skew polytope.

It should be noted that, unlike with regular polytope, skew polytope allows non-planar [ˈpleɪnə(r)] faces.

Compared with the first three shapes, the skew polytope not only has better compactness, but also has simple topology[tə'pɒlədʒɪ].
Complex topology means many colors split and merge over time, while simpler topology means colors change stably over time.

Therefore, we employ a 4D RGBT skew polytope to extract the video palette.

Choices of RGBT 4D geometry



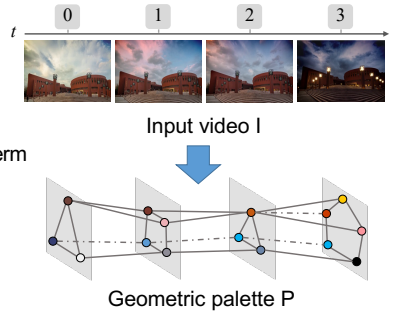
A comparison of these 4d geometry shapes is shown on this page.

Geometric palette measurement

- Our loss function

$$L_{\mathrm{v}}(\mathrm{P},\mathrm{I}) = L_{frame}(\mathrm{P},\mathrm{I}) + L_{smooth}(\mathrm{P})$$

Average polyhedral palette loss Smooth term



Geometric palette extraction takes a video as input and outputs its geometric palette. Our goal is to
extract a geometric palette with fewer representative colors that accurately reconstruct the video. Meanwhile we expect the polyhedral palettes changes smoothly between adjacent frames.

To measure the quality of the geometric palette, we define the loss function like this. It contains two terms, the first term is the average polyhedral palette loss, and the second is a smooth term.

And a satisfactory video palette can be achieved by minimize the loss function.

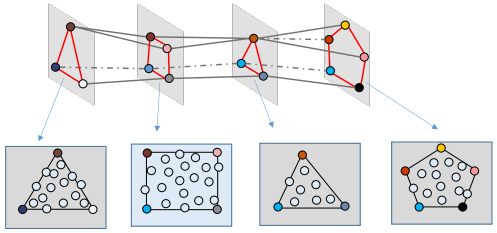
Geometric palette measurement

- Lower reconstruction loss

$$L_{\mathrm{v}}(\mathrm{P},\mathrm{I}) = L_{frame}(\mathrm{P},\mathrm{I}) + L_{smooth}(\mathrm{P})$$

Polyhedral palettes

Polyhedral palettes and video pixels



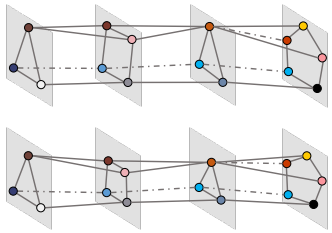
Our loss function encourage video palette with lower average polyhedral palette loss

Firstly, for each single frame, we expect the frame pixels can be accurately reconstructed by its corresponding polyhedral palette.
This ensures that there is no artifact in subsequent video recoloring.

Geometric palette measurement

- Rewards compactness

$$L_{\mathrm{v}}(\mathrm{P},\mathrm{I}) = L_{frame}(\mathrm{P},\mathrm{I}) + L_{smooth}(\mathrm{P})$$



Polyhedral palettes with larger volumes

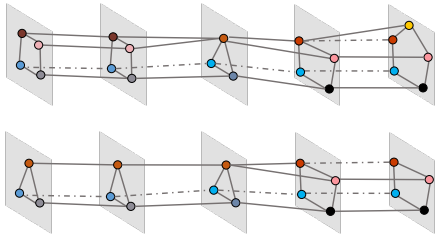
Polyhedral palettes with smaller volumes

Secondly, we also take the compactness loss into account, Under the compactness constrain[kənˈstreɪn] , our loss function rewards a palette with compact topology [təˈpɒlədʒɪ]. It leads to a video palette with representative colors.
Here, we provide two palettes, our algorithm tends to generate the bottom one with more compact shape that wrap the video pixels tightly.

Geometric palette measurement

- Prefers palettes with simpler topologies

$$L_{\mathrm{v}}(\mathrm{P},\mathrm{I}) = L_{frame}(\mathrm{P},\mathrm{I}) + L_{smooth}(\mathrm{P})$$



More vertices and edges

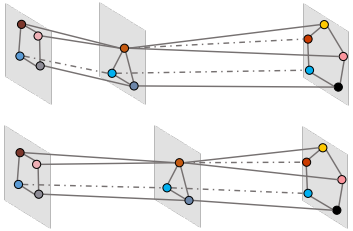
Fewer vertices and edges

The palette should remain easy to use, which means it should have a small number of representative colors.
Our algorithm prefer to produce a palette with fewer vertices and simpler topology. It also ensures palette colors changes stably.
Here, we provide two palettes on the right, our algorithm tends to generate the bottom one with fewer vertices and edges.

Geometric palette measurement

- Smooth change over time

$$L_{\mathrm{v}}(\mathrm{P},\mathrm{I}) = L_{frame}(\mathrm{P},\mathrm{I}) + L_{smooth}(\mathrm{P})$$



Color changes sharply

Color changes smoothly

We also expect the 4D skew polytope to be temporally coherent so that polyhedral palettes change smoothly along the timeline.
So we add a smooth term into the loss function. It sums up the color variation with time on all edges. It helps to generate a video palette with smooth edges.

Here we just briefly introduce the design intent of the loss function. More details please refer to our paper.

Geometric palette extraction

Step1 Initialization

Step2 Block Merging

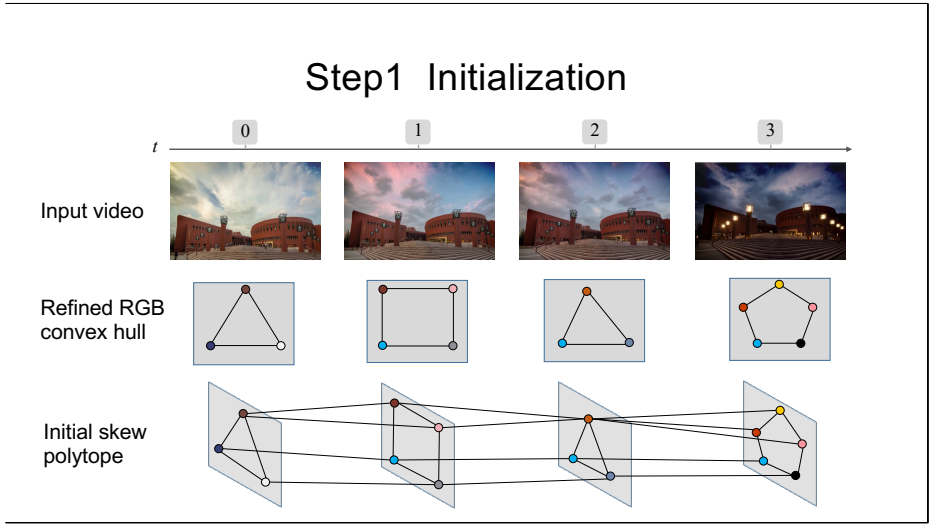
Step3 Vertex Removal

Step4 Vertex Refinement

As mentioned above, the loss function of the geometric palette is highly non-linear, cannot be easily solved end-to-end, so we propose a progressive approach to extract the video palette.

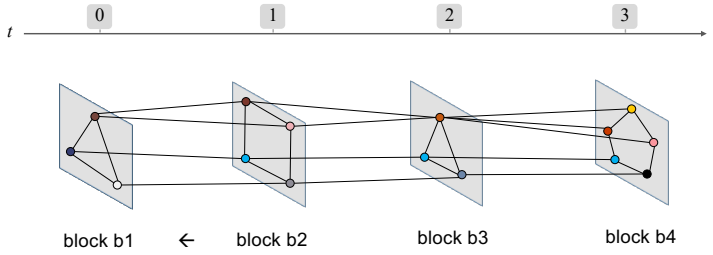
Generally, the pipeline of the geometric palette extraction contains four stages: initialization, block merging, vertex removal and vertex refinement.

In the following, we give a toy example to show the palette extraction process.



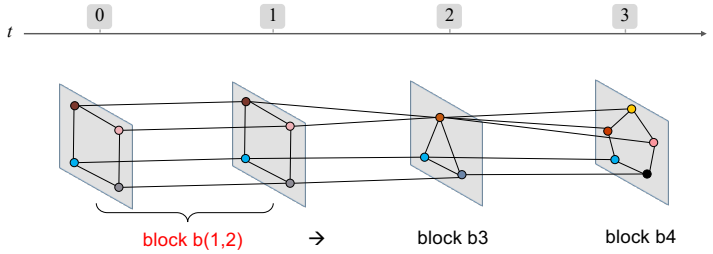
Given an input video, we first extract its simplified convex hull in RGB space and refine it. Next, we glue all these convex hulls together to form an initial skew polytope in RGBT space.

Step2 Block Merging



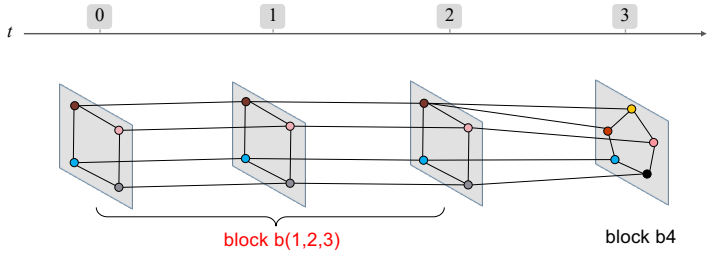
In the initialization step, each video frame generated its convex hull independently. As a result, it is likely that the polyhedral palettes of adjacent frames have different topologies[təˈpɒlədʒiz] . We introduce block merging to reduce the unnecessary topological changes between frames. here, a block is defined as a sequence of contiguous frames with consistent topology[təˈpɒlədʒi] . in this demo, there are 4 blocks in the initial skew polytope. Here block b2 first merges block b1

Step2 Block Merging



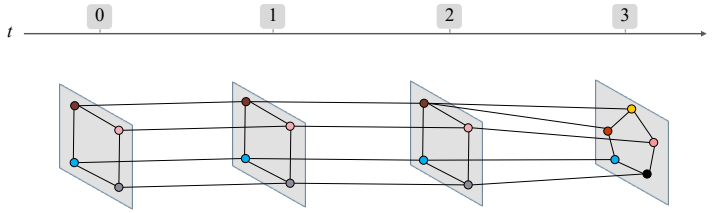
So that frame 1 and frame 2 form a new block b12
And then, block 12 merges block b3

Step2 Block Merging



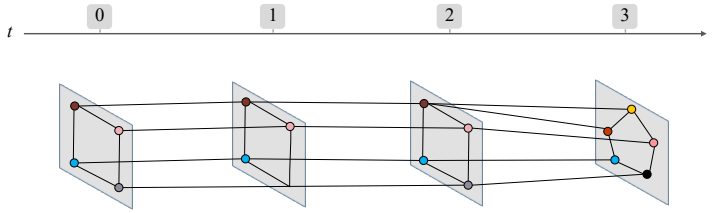
So that frame 12 and frame 3 form a new block b123

Step2 Block Merging

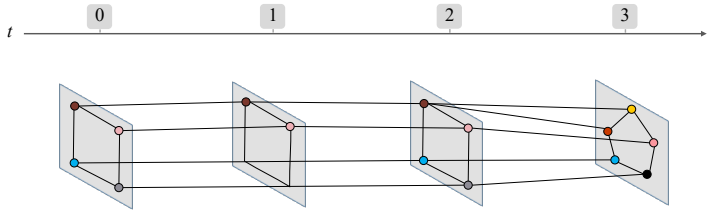


After block merging, the 4D skew polytope has simpler topology but still contains a large number of vertices. To further reduce these redundant vertices, we progressively simplify the 4D skew polytope through iterative vertex removal. In the following, we will show this process step by step.

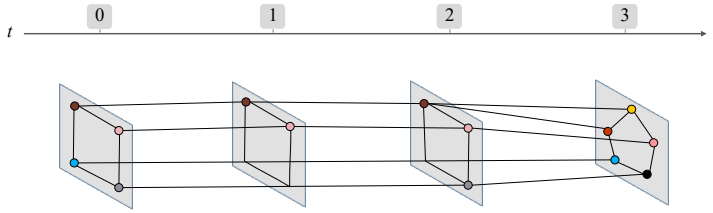
Step3 Vertex removal



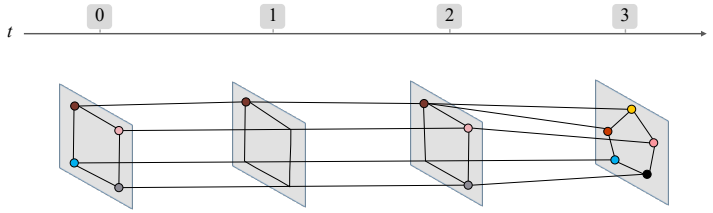
Step3 Vertex removal



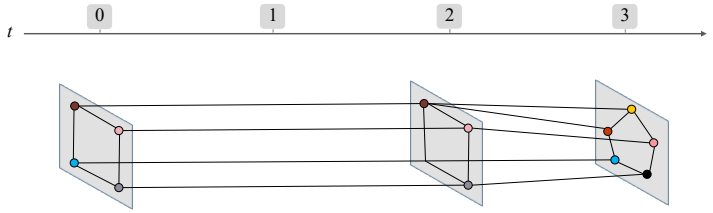
Step3 Vertex removal



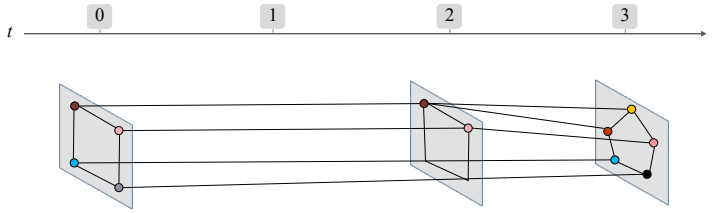
Step3 Vertex removal



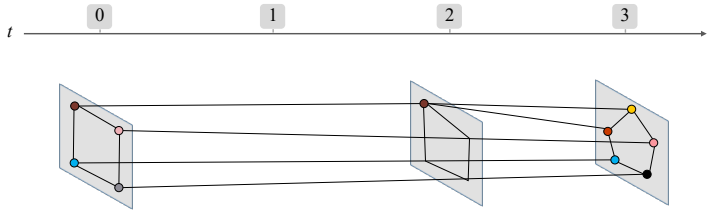
Step3 Vertex removal



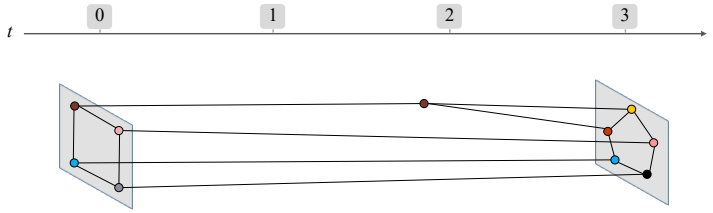
Step3 Vertex removal



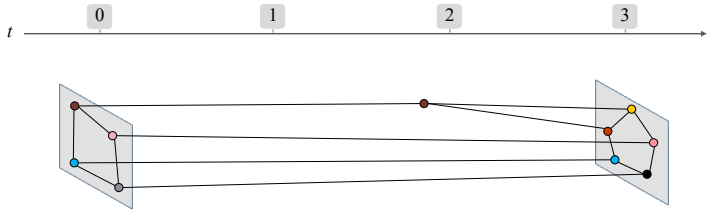
Step3 Vertex removal



Step3 Vertex removal

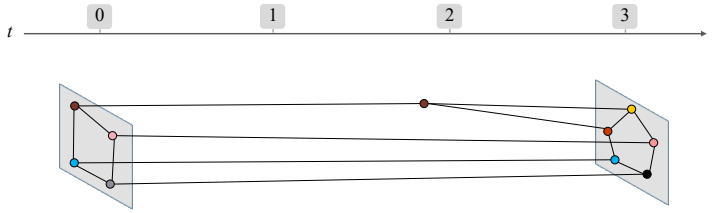


Step4 Vertex refinement



After simplifying the 4D skew polytope, we perform additional vertex refinement to further reduce the overall video loss and increase the smoothness of edges in the skew polytope.

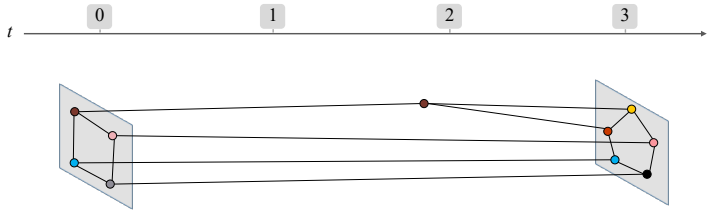
Step4 Vertex refinement



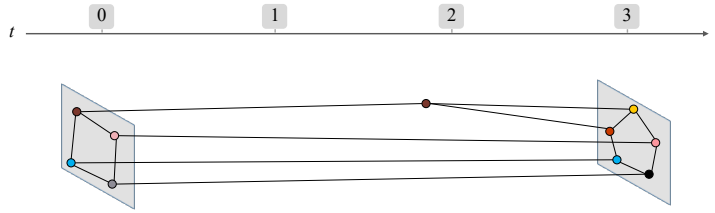
We employ an iterative [ˈɪtəˌreɪtɪv] scheme to refine the simplified skew polytope. Specifically [spəˈsɪfɪkli] , at each iteration, we select one vertex for local adjustment and keep all other vertices fixed.
We cycle through all vertices several times for the above iterative refinement process until convergence.

In the following, we will this process step by step.

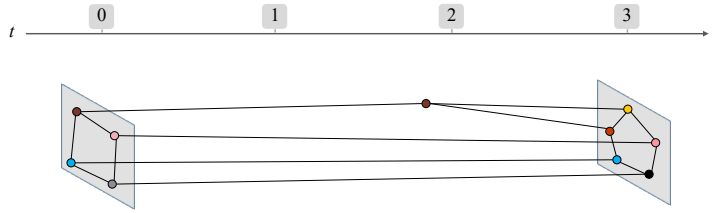
Step4 Vertex refinement



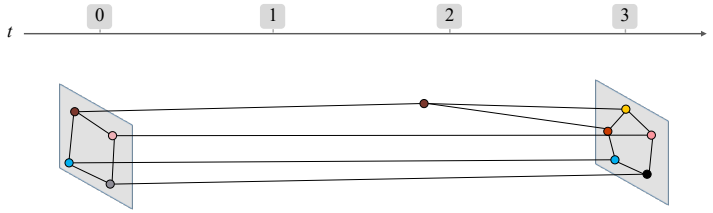
Step4 Vertex refinement



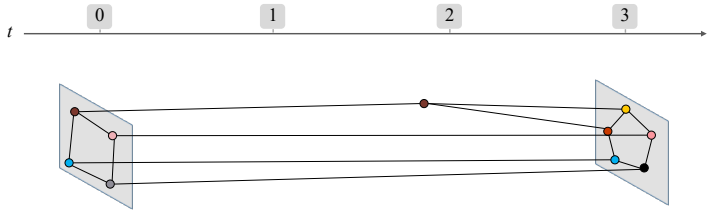
Step4 Vertex refinement



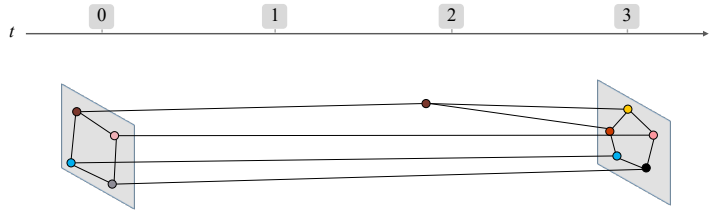
Step4 Vertex refinement



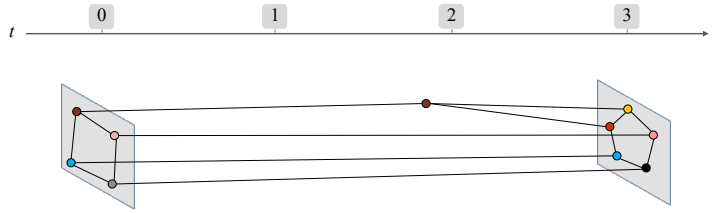
Step4 Vertex refinement



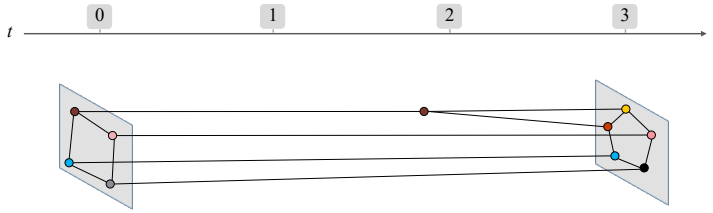
Step4 Vertex refinement



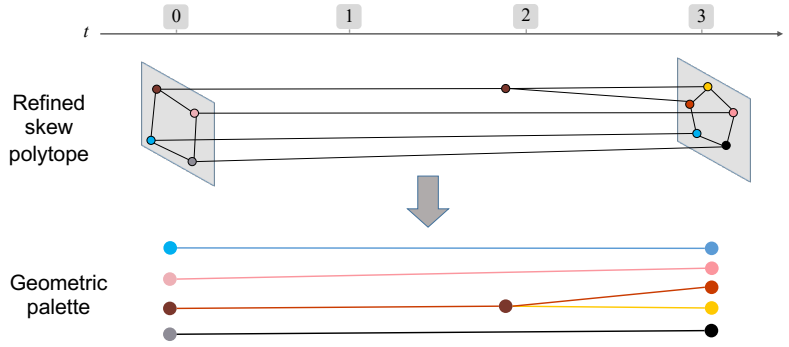
Step4 Vertex refinement



Step4 Vertex refinement

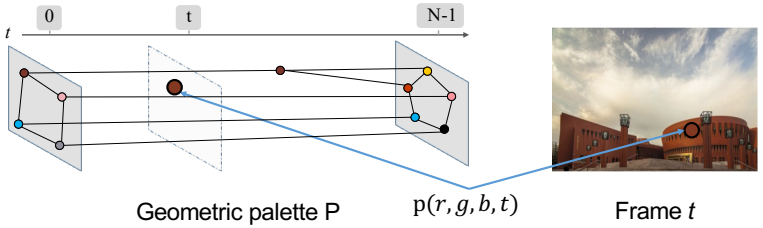


Output the geometric palette



The resulted 4D skew polytope corresponds to the generated geometric palette. So far, we have already extracted the geometry palette from a given video, next we show how to use it to recolor the video.

Video recoloring



Geometric palette P $\quad \mathrm{p}(r,g,b,t)$ $\quad$ Frame t

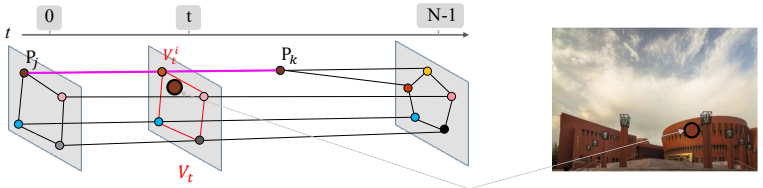
$$\mathrm{p} = \sum_{i=1}^{|V|} w_i \mathrm{P}_i = W_{RGBT}\mathrm{P} = W_{RGBT} \left[\quad \right]^{\mathrm{T}}$$

Geometric palette P vertices (colors)

In a video, any pixel p with color(r,g,b) in frame t can be regarded as a point p(r,g,b,t) in RGBT space.
Similar to image recoloring, Our goal is to express p as a linear combination of the geometric palette vertices.

But how to compute the mixing weights matrix?

Video recoloring



Geometric palette P $\quad$ $\mathrm{p}(r,g,b,t)$ $\quad$ Frame t

1) Get the slice (polyhedral palette) at frame t : $V_t = S(\mathrm{P}, t)$

Each vertex of V_t lies in an edge of P, it can be linearly interpolated as:

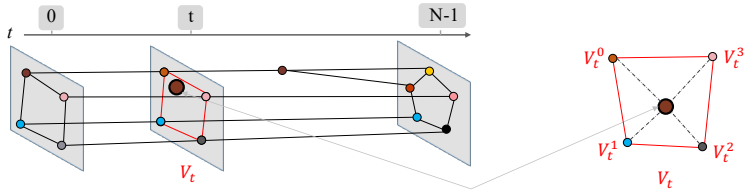
$$V_t^i = \alpha P_j + (1-\alpha) P_k \longrightarrow V_t = W_t \mathrm{P}$$

Firstly, we slice the skew polytope at time t to get the polyhedral palette v_t that encloses p.

since each vertex of the slice V_t lies in an edge of the skew polytope of P, so the vertices of the polyhedral palette can be calculated by linear interpolation.

so the vertices of the polyhedral palette can be further expressed as matrix form:
V_t = W_tP

Video recoloring



Geometric palette P $\quad \mathrm{p}(r,g,b,t)$

2) Inside the slice (closed polyhedron V_t), the point p can be expressed as:

$$\mathrm{p} = \sum_i w_{\mathrm{p}}^i V_t^i \longrightarrow \mathrm{p} = W_{\mathrm{p}} V_t$$

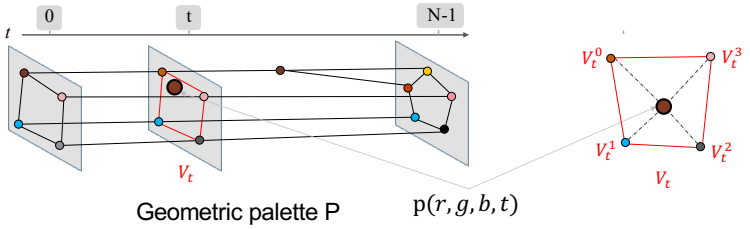
Mean value coordinates for closed triangular meshes [Ju et al. 2005]

Secondly, any vertex inside V_t can be expressed as a linear combination of V_t's vertices.

In this paper, we use Mean value coordinates to compute the mixing weights.

And other generalized barycenteric coordinates are also available for this task.

Video recoloring



Geometric palette P $\quad$ $\mathrm{p}(r,g,b,t)$

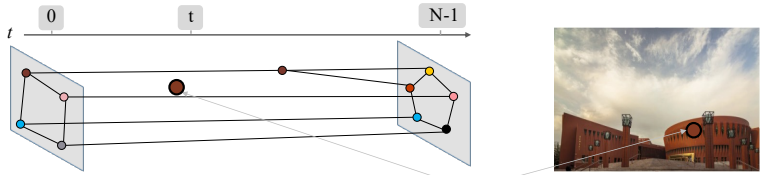
1) Get the slice of $\mathrm{p}(r,g,b,t)$ in P : $V_t = W_t \mathrm{P}$

2) $\mathrm{p}(r,g,b,t)$ is inside the slice V_t: $\mathrm{p} = W_\mathrm{p} V_t$

$$\mathrm{p} = \underbrace{W_\mathrm{p} W_t}_{W_{RGBT}} \mathrm{P}$$

Since both steps are linear, they can be combined through matrix multiplication. In this way, any point p(r,g,b,t) can be directly expressed as linear combination of the 4D geometric palette.

Video recoloring

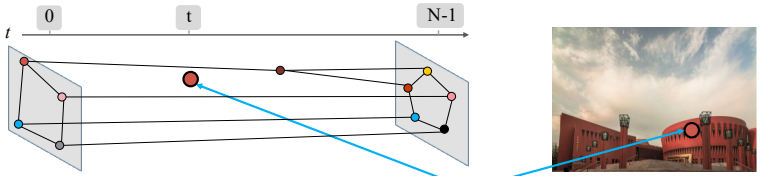


Geometric palette P $\quad \mathrm{p}(r,g,b,t) \quad$ Frame t

$$\mathrm{p} = W_{RGBT}\mathrm{P}$$

Keep the weight matrix fixed, when we modify the geometric palette, all video pixels are updated accordingly, and resulting a recolored video.

Video recoloring

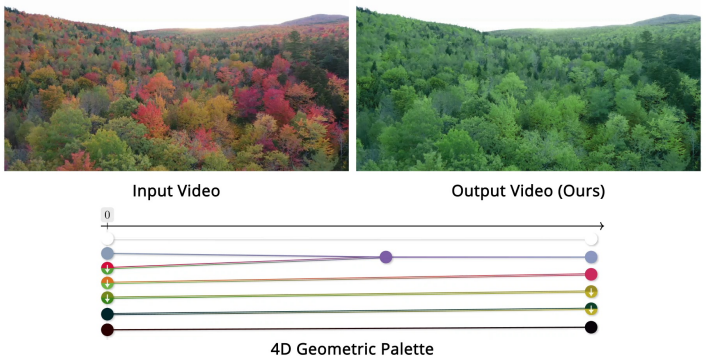


Geometric palette P' $\quad \mathrm{p}'(r', g', b', t)$ Recolored frame t

$$\mathrm{p} = W_{RGBT}\mathrm{P} \longrightarrow \mathrm{p}' = W_{RGBT}\mathrm{P}'$$

Keep the weight matrix fixed, when we modify the geometric palette, all video pixels are updated accordingly, and resulting a recolored video.

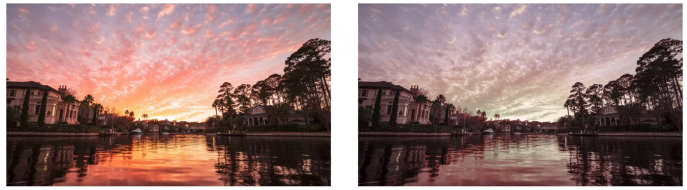
Results



Here we show some video recoloring results.

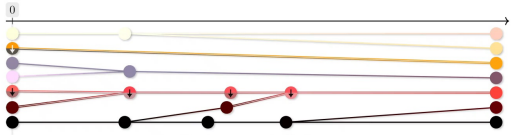
In this video, we make the mountain gradually change from spring to autumn.

Results



Input Video

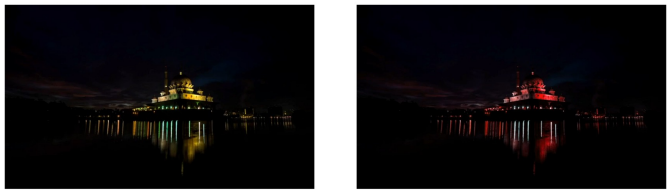
Output Video (Ours)



4D Geometric Palette

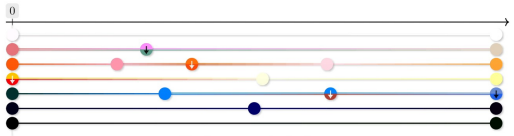
In this video, we make the cloud color changes from gray to red.

Results



Input Video

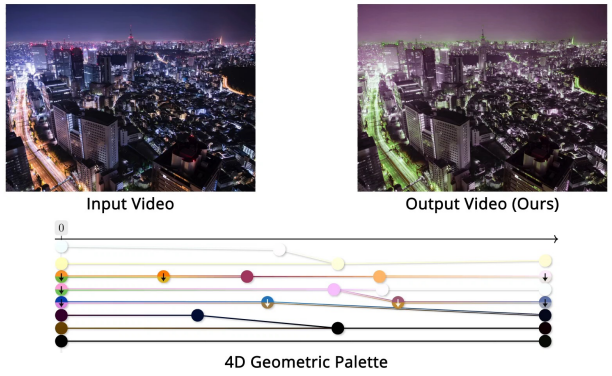
Output Video (Ours)



4D Geometric Palette

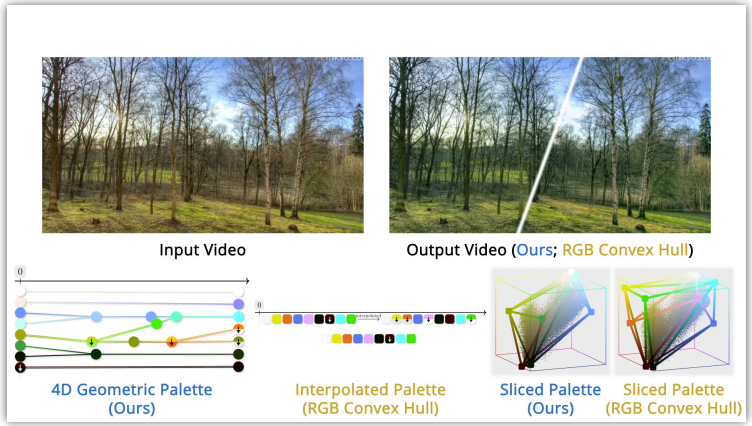
In this video, we perform more color variations to the cloud and the mosque to make them more colorful.

Results



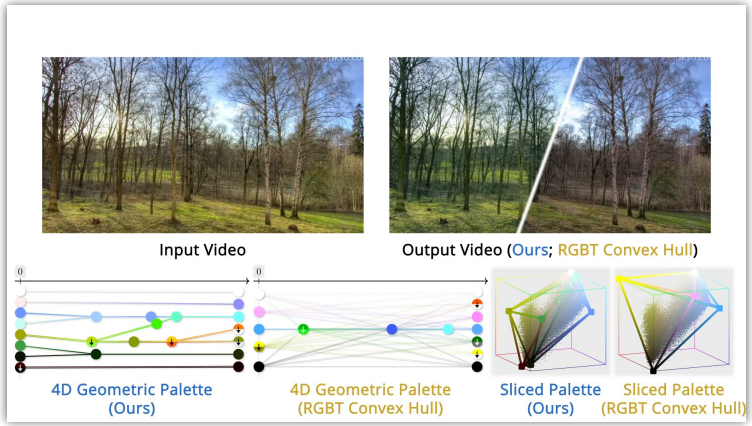
In this video, we make the light color gradually changes from green to yellow, and the morning looks warmer.

Compare to RGB convex hull



Here, we compared our method with global RGB convex hull-based method. in this video, The recoloring intent is to emphasize the change in seasons by making the earth darker in spring, the leaves greener in summer and redder in autumn, and the fallen leaves whiter in winter. Our palette captures object color changes in the video, and achieve natural time-varying effects. While RGB convex hull-based method fails to capture the colors change.

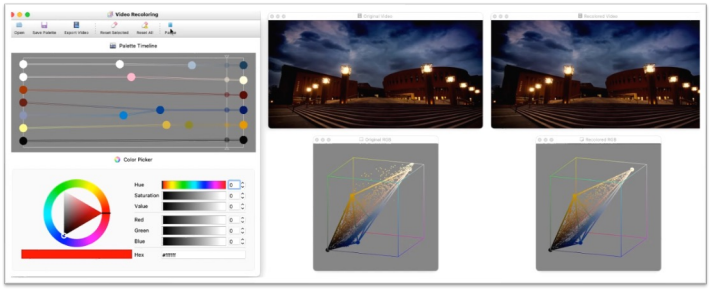
Compare to RGBT convex hull



Here, we compared our method with global RGBT convex hull-based method. It is clear that the RGBT convex hull-based palette with complexity topology, suffer from poor compactness and fails to recolor the video as expected.

Conclusion

- We proposed the first palette-based video recoloring
- Our method produces natural, artifact-free recoloring



In conclusion, we propose the first palette-based video recoloring, and our method produces natural, artifact-free recoloring.

Limitation

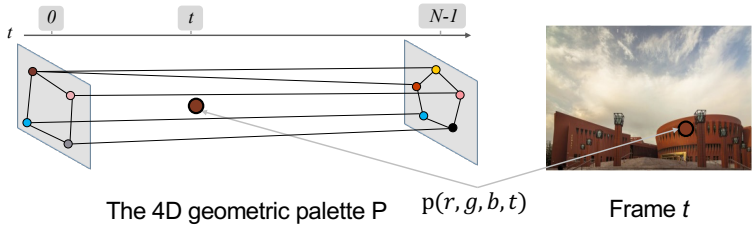
- Can not realize object aware local recoloring

Different objects with similar colors will be recolored simultaneously

The first limitation of our method is that palette-based editing methods perform global recoloring, so some spatially-varying or per-object changes cannot be captured without additional semantic information.

Limitation

- Non-linear color editing such as tone mapping is not supported



The 4D geometric palette P $\mathrm{p}(r, g, b, t)$ Frame t

$$\mathrm{p} = W_{RGBT}\mathrm{P}$$

Secondly, our method is able to support a range of useful color manipulations, but linear-mixing-weight-based approaches cannot handle non-linear color manipulations such as tone mapping.

Thank You!

- **Contact Information:**
  - Zheng-Jun Du: duzj19@mails.tsinghua.edu.cn
  - Kai-Xiang Lei: leikx18@mails.tsinghua.edu.cn
  - Kun Xu: xukun@tsinghua.edu.cn
  - Jianchao Tan: tanjianchaoustc@gmail.com
  - Yotam Gingold: ygingold@gmu.edu

Project page

That's all my presentation.
Thanks for your attention!
If you have any questions, please feel free to contact us.