Palette Based Recolouring

Devika Bej
Centre for Visual Information Technology
IIIT Hyderabad

Harshvardhan Pandey
Signal Processing and Communication Research Center
IIIT Hyderabad

***Abstract*—The study presented aims to recreate and analyze the work of Wang et al on palette extraction from images using a convex colour combination model. The pipeline is extended with intrinsic decomposition as a pre-processing step and the overall results are analyzed.**

***Index Terms*—palette-based editing, linear programming, non-linear programming, mean value coordinates**

I. Introduction

Palette-based editing methods are significantly motivated by the classical painting process and its emulation in digital art. The problem has been studied extensively with different techniques like formulating the problem as an optimization problem, clustering methods etc. This study aims to analyze the problem from a geometric point of view.

We start with a convex model of colour combinations from the palette colours.

$$\sum_{\mathbf{v}\in\mathbf{V}} w_i^v \mathbf{v} = \mathbf{v} \in \mathbf{V}$$

Using this model, a natural place to start for palette extraction is the vertices of the convex hull of the colours in RGB space. However, this exact convex hull is a tight wrapping of the image colours and typically has too many vertices. (As these vertices are the colour palette, having too many vertices produces an unmanageable number of layers for the user.) Therefore, we simplify the convex hull to a user-specified palette size (vertex count) that still encloses the image colours.

However, having a "less tight" enclosing polygon reduces the "representativeness" of the resulting palette, as shown in Figure 1.

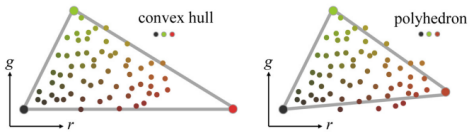

Fig. 1. Illustration of representativeness of Polygon Colour Palette

Our base reference paper "An Improved Geometric Approach for Palette-based Image Decomposition and Recoloring" focuses on coming up with a convex polygon that doesn't necessarily enclose all the points, but whose vertices are more representative of the colours in the image.

A natural extension worth studying is considering the intrinsic decomposition of the image. The Intrinsic decomposition problem is to decompose an image I into its colour component (albedo) A and shading component S, such that $I = AS$. The palette information should depend only on the albedo. However, intrinsic decomposition in of itself is an underconstrained problem and notoriously difficult to solve. We use the pre-trained model of Careaga and Aksoy to compute intrinsic decomposition.

II. Convex Hull Simplification

The first step of the process is to find a convex polygon which encloses the colours while having a small number of vertices. The progressive hull is a mesh simplification technique introduced by Sander et al. [2000]. The approach is based on a sequence of edge contractions, in which an edge is contracted to a vertex. The vertex is placed according to a constrained optimization problem. The constraints are that the vertex must be placed outside (the plane of) every face incident to the edge. Equivalently, every tetrahedron formed by connecting the new vertex to a face incident to the edge must add volume to the shape. The progressive hull objective function minimizes the total added volume. The next edge to contract is the one whose contraction adds the least volume. (Edges with no solution satisfying the constraints are skipped.) This approximation strictly adds volume, so all colours are guaranteed to be contained in every step of the progressive hull. Moreover, the constraints and objective function are linear, since the (oriented) volume of a tetrahedron with three fixed vertices is linear:

$$\frac{A}{3}\mathbf{n}\cdot(\mathbf{v}-\mathbf{v}_0)$$

where $\mathbf{v}$ is the free vertex, $\mathbf{v}_0$ is any one of the fixed vertices, and $\mathbf{n}$ and A are the outward unit normal and area of the triangle formed by the three fixed vertices. As a result, the constrained optimization for each edge contraction is a linear programming problem, which can be solved efficiently

III. Palette Refinement

Wang et al. [2019] shows that the gap in the earlier use of convex hull for palette extraction is the lack of representativeness. This refinement seeks to reach a trade-off between the error of reconstruction and this representativeness. This section can be divided into two parts - the mathematical formulae and the algorithm implemented

A. Mathematical Formulation

The convex hull or the color palette is denoted as polyhedron $\mathbf{P}(\mathbf{V}, \mathbf{F})$ where $\mathbf{V}$ is the set of vertices and $\mathbf{F}$ is the set of faces. The image is denoted as $\mathbf{I}$.

The function to be minimised is as follows

$$\arg\min_{\mathbf{P}} F(\mathbf{P}) = \lambda \cdot R(\mathbf{P}, \mathbf{I}) + S(\mathbf{V}, \mathbf{I})$$

F here is the total loss, a weighted (λ) sum of reconstruction loss (R) and representative loss (S)

For each point of image $\mathbf{i}$, consider $\mathbf{i}_P$ which is its projection on the polyhedron $\mathbf{P}$. For points inside the hull, the $\mathbf{i}_P$ remains the same. For the points outside the hull, the it is the point on the polyhedron closest to $\mathbf{i}$. This means that the loss is simply the summation of the distances of these outside points from their projections. This is because during reconstruction, the projection of the point will be used instead of the original point. Hence the overall error during construction will be equal to this.

$$R(\mathbf{P}, \mathbf{I}) = \frac{1}{|\mathbf{I}|} \sum_{\mathbf{i} \in \mathbf{I}} ||\mathbf{i} - \mathbf{i}_P||$$

For each vertex of the hull $\mathbf{v}$, consider the $\mathbf{v}_c$ which is the center of its M nearest neighbours. This M can be adjusted as needed. This $\mathbf{v}_c$ will be the average of the neighbours, and representative of this cluster. Thus the distance of $\mathbf{v}$ from $\mathbf{v}_c$ is a measure of how much $\mathbf{v}$ can represent its neighbours. This is the goal of the paper, to minimise this while not increasing other losses.

$$S(\mathbf{V}, \mathbf{I}) = \frac{i}{|\mathbf{V}|} \sum_{\mathbf{v} \in \mathbf{V}} ||\mathbf{v} - \mathbf{v}_c||$$

$$\mathbf{v}_c = \frac{1}{M} \sum_{\mathbf{i} \in N(\mathbf{v})} \mathbf{i}$$

B. Algorithm

Since considering each point in the image would be computationally very expensive, with higher resolutions, we do not directly use the image. $\mathbf{I}_{sample}$ is used for the algorithm, where K points are sampled from the image, uniformly from throughout the image.

The initial hull is obtained from the Convex Hull Simplification module. This hull is refined vertex by vertex, and one iteration consists of one round of adjusting all the vertices of the hull. Therefore, the following process is run for each vertex $\mathbf{v}$, $n_{iteration}$ times.

1) $\mathbf{v}_0$ is the initial position of $\mathbf{v}$. Compute the $\mathbf{v}_c$ for it, denoted by $\mathbf{v}_{c,0}$
2) We assume the new $\mathbf{v}$ to be on the line between $\mathbf{v}_0$ and $\mathbf{v}_{c,0}$. Here we optimize on a scalar $k \in [-0.5, 1]$ s.t.
$$\mathbf{v} = (1 - k)\mathbf{v}_0 + k\mathbf{v}_{c,0}$$
3) The initial guess of k is from $[0, 1]$ with step of 0.1, s.t. the $\mathbf{v}$ generated has minimum cost. This initial guess is then given to sklearn.optimize.minimize to find the optimal k.
4) The vertex is then updated according to the k obtained.

The $n_{iteration}$ is usually around 2 or 3. In our implementation this value is set at 2.

IV. Mean Value Coordinates

The weights given to each colour in the palette are a function of the colour at the given pixel. Now, we know the value of this function at the vertices of the convex polygon, thus we can use some interpolation to compute the function at the remaining points. The paper suggests to use the "Mean Value Coordinate Interpolation". However, there is a caveat. Some points lie outside the hull. The paper suggests to project the points on the hull, but doesn't suggest a method. We formulated the projection as the following quadratic programming problem:

$$\text{minimize } (\mathbf{x} - \mathbf{x}_0)^T (\mathbf{x} - \mathbf{x}_0)$$

$$\text{subject to } A\mathbf{x} \preceq b$$

Where $\mathbf{x}_0$ is the point we are trying to project on the hull and $A\mathbf{x} \preceq b$ defines the polyhedron.

After the projection, we utilize the pseudo-code provided by Ju et al to compute the coordinates.

V. Results

A. Qualitative Analysis

We pick two images from the paper labelled "Apple" and "Turquoise", and we pick two images ourselves labelled "Ice Cream" and "Sunset Palette". For each of these, we see the reconstruction after the simple pipeline, albedo, albedo after the pipeline, and the albedo multiplied by the shading. The parameters picked for this are $\lambda = 20$ and $E_{vertices} = 6$.

B. Quantitative Analysis

1) Parameter $E_{vertices}$: The parameter $E_{vertices}$ is used in the Convex Hull Simplification module. It is the number of vertices that the algorithm tries to minimize convex the hull size to. The values used for testing are 4, 6, 8. Table shows that the error in reconstructed image reduces as the hull size increases. However, from a practical perspective, a colour palette is best with lesser number of colours. Hence, the ideal value of $E_{vertices}$ is 6.

2) Parameter λ: The parameter λ is used in the Palette Refinement module. It is the relative weightage given to reconstruction loss as compared to representative loss when calculating the total loss, i.e. the cost function. This factor determines what the refined palette prioritises while converging (during the optimization).

The reconstruction loss is the sum of distances of the points outside the hull, from the hull. It takes into account how many pixels will be modified slightly during reconstruction. The representative loss is the sum of distances of each hull vertex from the center of its nearest neighbours.

$$L_{total} = \lambda * L_{reconstruct} + L_{represent}$$

According to the paper implemented, the ideal value of λ is 20.

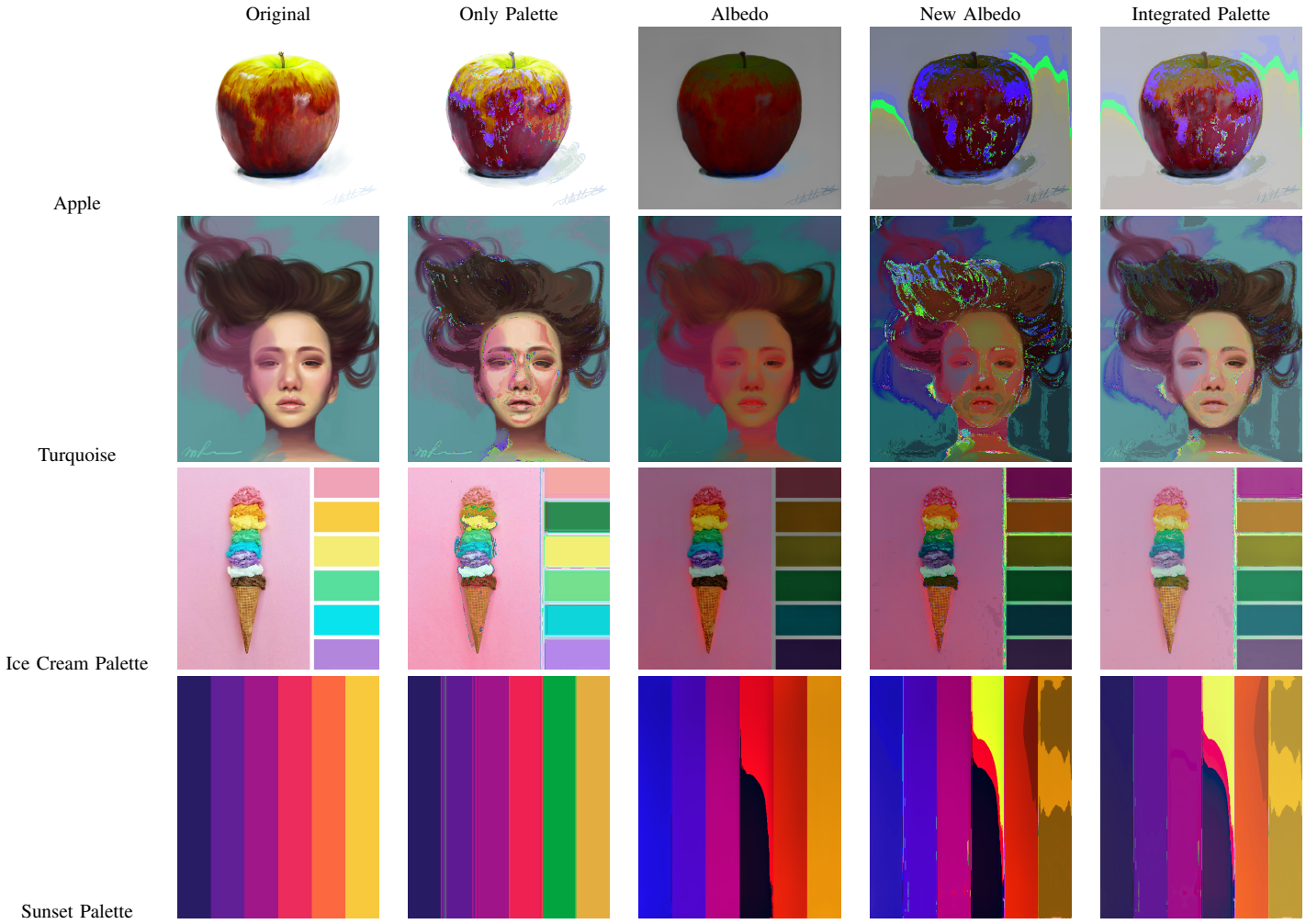

TABLE I
QUALITATIVE COMPARISON

	Apple	Turquoise	Ice Cream Palette	Sunset Palette
4	0.2389	0.0094	0.1435	0.1285
6	0.0172	0.0049	0.0214	0.0577
8	0.0146	0.0031	0.0171	0.0022

TABLE II
VARIATION OF LOSS WRT $E_{vertices}$ WITHOUT INTRINSIC DECOMPOSITION

	Apple	Turquoise	Ice Cream Palette	Sunset Palette
4	0.0691	0.0053	0.0294	0.0404
6	0.0717	0.0073	0.0417	0.0337
8	0.0730	0.0053	0.0397	0.0157

TABLE III
VARIATION OF LOSS WRT $E_{vertices}$ WITH INTRINSIC DECOMPOSITION

	Apple	Turquoise	Ice Cream Palette	Sunset Palette
20	0.0172	0.0049	0.0214	0.0577
80	0.0264	0.0017	0.0447	0.0604
200	0.0326	0.0021	0.0809	0.0613

TABLE IV
VARIATION OF LOSS WRT λ WITHOUT INTRINSIC DECOMPOSITION

	Apple	Turquoise	Ice Cream Palette	Sunset Palette
20	0.0717	0.0073	0.0417	0.0337
80	0.0672	0.0086	0.0470	0.0202
200	0.0535	0.0103	0.0567	0.0287

TABLE V
VARIATION OF LOSS WRT λ WITH INTRINSIC DECOMPOSITION

VI. CONCLUSION

From the above experiment of integrating Intrinsic Decomposition and Convex Hull based Palette Extraction, the following two conclusions can be made.

- Use of the decomposition worsens the reconstruction performance. This implies that light information is necessary

to the overall colour information of the image.

- Use of decomposition helps speed up the process. This is because the albedo has fewer variety in colours of pixels

REFERENCES

[1] Y. Wang, Y. Liu, and K. Xu, "An Improved Geometric Approach for Palette-based Image Decomposition and Recoloring An Improved Geometric Approach for Palette-based Image Decomposition and Recoloring," vol. 38, no. 7, 2019.

[2] J. Tan, J.-M. Lien, and Y. Gingold, "Decomposing Images into Layers via RGB-Space Geometry," ACM Transactions on Graphics, vol. 36, no. 1, pp. 1–14, Feb. 2017, doi: https://doi.org/10.1145/2988229.

[3] C. Careaga and Y. Aksoy, "Colorful Diffuse Intrinsic Image Decomposition in the Wild," ACM Transactions on Graphics, vol. 43, no. 6, pp. 1–12, Nov. 2024, doi: https://doi.org/10.1145/3687984.

[4] C. Careaga and Y. Aksoy, "Intrinsic Image Decomposition via Ordinal Shading," ACM transactions on graphics, vol. 43, no. 1, pp. 1–24, Nov. 2023, doi: https://doi.org/10.1145/3630750.

[5] T. Ju, S. Schaefer, and J. Warren, "Mean value coordinates for closed triangular meshes," ACM Transactions on Graphics, vol. 24, no. 3, pp. 561–566, Jul. 2005, doi: https://doi.org/10.1145/1073204.1073229.

[6] P. Sander, J. Snyder, S. Gortler, and H. Hoppe, "Texture Mapping Progressive Meshes."