Image Color Cube Dimensional Filtering and Visualization

Jose-Luis Lisani, Antoni Buades, Jean-Michel Morel

published • 2011-06-22 → BibTeX
reference • Jose-Luis Lisani, Antoni Buades, and Jean-Michel Morel, *Image Color Cube Dimensional Filtering and Visualization*, Image Processing On Line, 1 (2011). http://dx.doi.org/10.5201/ipol.2011.blm-cdf

Communicated by Julie Delon
Demo edited by Jose-Luis Lisani

- Antoni Buades toni.buades@uib.es, MAP5, Université Paris Descartes
- Jose-Luis Lisani joseluis.lisani@uib.es, TAMI, Universitat de les Illes Balears
- Jean-Michel Morel morel@cmla.ens-cachan.fr, CMLA, ENS Cachan

Content

Overview

We call *RGB cube* the three dimensional representation of the colors of a digital image in RGB space. This cube simply shows the *(R,G,B)* color points which actually appear in the digital image. This visualization has, however, two main defects: it does not show the real *RGB histogram*. The difference is that in the color cube each point is plotted with its own color, while in the RGB histogram it has a grey level representing the number of pixels in the image having this color value. It is of course impossible to display these two features simultaneously. The perspective views, which are most used, hide the real dense cloud, which is surrounded by a fume of isolated color points. These isolated color points, mainly due to the image noise and to compression artifacts, make an opaque obstacle to the perspective view of the cloud. Perspective views remain nonetheless the classic visualization tool, because the cube occupation by the color points of a natural image is far from complete, as it can be seen in the following example:

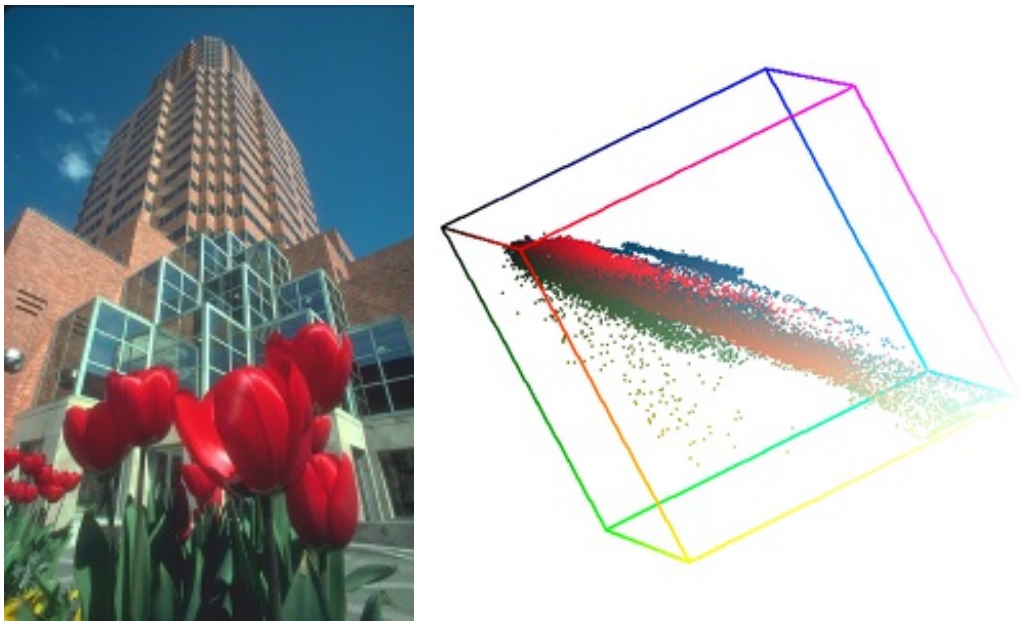


Figure 1. Original image and its color cube.

In ref1 the authors give a comprehensive understanding of all spatial structures appearing in a color

In ref1 the authors give a comprenensive understanding of all spatial structures appearing in a color histogram. They explore the building parts of the color cube by a qualitative analysis of image details. They also investigate how these elementary structures are combined, and their physical causes. Their conclusion is that 2D structures predominate in the color cube. They are qualitatively explained by two main causes: image blur and texture.

Besides the qualitative analysis of spatial structures in RGB space, the authors of ref1 make a quantitative evaluation supporting the 2D dimensional character of the RGB cube. To this aim a filtering algorithm, the *Local Linear Projection (LLP2)* is used to smooth the color distribution and to reveal the underlying 2D structures of the color clusters.

LLP2 replaces each RGB color in the cube by its projection onto a locally defined regression plane, computed by Principal Components Analysis (PCA). The process is iterated some two to four times, until the average error between corresponding colors in consecutive iterations is below a threshold of 0.5.

Thus LLP2 estimates a 2D manifold from the unstructured cloud of points allowing for a better visualization of the colors. This 2D manifold is usually very close to the initial cloud: on average the displacement of the color points is around 3 (see Table 3 below), which is quite small in relation to the cube dimensions (255,255,255). On this manifold, the color density is displayed by a brightness value proportional to its logarithm. The density has become visible because the cloud is flattened and the dense color points in direct sight, being no more hidden by low density color points (see Fig. 2).

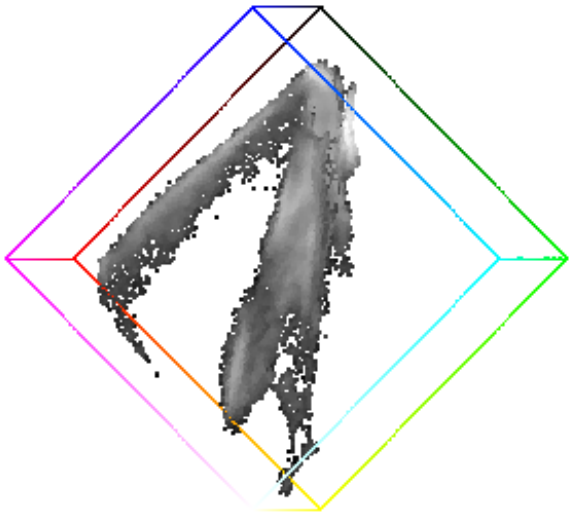


Figure 2. Perspective view of the color cube of image in Fig. 1 after filtering with LLP2. Each color is displayed with a grey level proportional to the logarithm of the density in RGB space (lighter for higher densities). The density of colors points is defined here.

The algorithm is described in the next sections, together with implementation details and documented source code. Moreover, an online demonstration is available to anybody willing to test the results of LLP2 on his (her) own images. Several examples are displayed below.

References

1. T. Buades, J.L. Lisani and J.M. Morel *The dimensionality of color space in natural images* Journal of the Optical Society of America A, 28(2), pp. 203--209, 2011. DOI: 10.1364/JOSAA.28.000203. (preprint)
2. D. Levin *The approximation power of moving least-squares* Math. Comput. 67, 224 pp.1517--1531, 1998
3. J. Digne, J.M. Morel, C. Mehdi-Souzani and C. Lartigue *Scale Space Meshing of Raw Data Point Sets* preprint CMLA 2009-30, 2009
4. X. Huo and J. Chen, J. *Local linear projection (LLP)* Proc. of First Workshop on Genomic Signal Processing and Statistics (GENSIPS}, 2002
5. D. Martin, C. Fowlkes, D. Tal and J. Malik *A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics*, Proc. 8th Int'l Conf. Computer Vision, pp.416--423, 2001 DOI: 10.1109/ICCV.2001.937655.

Online Demo

Online Demo

Try this algorithm on your own images with the online demonstration.

Algorithm

Given a *k*-dimensional manifold given by noisy samples, the problem of recovering the manifold by denoising the samples and meshing them is a classic problem whose solution can be taken advantage of for processing color clouds. The founding method seems to be the *moving least square* (MLS) method introduced in ref2. It is proved in ref3 that the MLS method of order 1, which projects the point cloud locally on its regression plane, moves the underlying surface by mean curvature motion. This means that if the surface is flat, it hardly moves by MLS1. However, if the point cloud is noisy, a smooth manifold is fast restored and can be meshed, as proved in ref3. MLS1 is a particular case of the Local Linear Projection (LLP) algorithm ref4, which does not assume a particular dimension. LLP with *k* dimensions (LLP*k*) takes for each point its closest points in Euclidean distance and computes the PCA of this neighborhood. Then, the current point is projected on the regression affine manifold generated by the *k* first PCA components. This algorithm was proposed by Huo *et al.* ref4 for the filtering of high dimensional data. Notice that MLS1 and LLP2 are the same algorithm.

The adaptation of LLP to the color point clouds is described below. Each RGB color actually may represent more than one pixel in the image. When applying this filtering strategy each color is counted as many times as it appears in the image.

LLP algorithm

For each RGB color y_i, *i=1, 2, ..., N*

1. Find the *K* neighbors whose distance to y_i is smaller than a given threshold *T*. The neighboring colors are denoted by x_1, x_2, ..., x_K;
2. Compute the principal components of the set x_1, x_2, ..., x_K;
3. Let *k* (*k=0*, *k=1* or *k=2*) be the assumed dimension of the embedded manifold, then project y_i onto the affine subspace spanned by the first *k* principal components and passing through the center of mass of the set x_1, x_2, ..., x_K.

The algorithm is iterated until the average difference between corresponding colors in two consecutive iterations is below a precision parameter ε.

In practice only LLP2 is used, since the 2D model has proven ref1 to be the best adapted to the color distributions found in natural images.

Implementation details

The complexity of the algorithm is a function of the number of colors C in the image. For each color, a neighborhood of size $(2T+1)^3$ is explored (step 1 of the algorithm) and then the color point is processed. This simple analysis leads to an upper bound of $O(CT^3)$ operations per iteration. The total number of iterations depends on the speed of convergence of the algorithm.

It is not possible to give an *a priori* estimate of the computation time in terms of the size of the image, since there is no exact correspondence between number of pixels and number of colors. Statistical analysis performed on 100 images from the Berkeley test dataset ref5 (size 481x321) gives an average ratio of 4.24 pixels/color. Moreover, this ratio decreases as the image size is reduced (see Table 1).

scale	*size (pixels)*	*colors*	*pixels/color*
1	154401	36363.13	4.24
0.75	86400	26676.66	3.24
0.5	38400	16482.54	2.33
0.25	9600	6155.44	1.56

Table 1. Number of pixels and average number of colors computed from 100 images from the Berkeley

Table 1. Number of pixels and average number of colors computed from 100 images from the Berkeley test dataset ref5 for different scale factors. The last column shows the ratio between the number of pixels with respect to the number of colors.

The following specific implementation of the LLP algorithm has been adopted in order to reduce the computation time while ensuring the quality of the results:

- in step 1, L1 distance is used and the value of threshold *T* is fixed to *10*. In the Examples section it is shown that the value of this parameter is not critical for the algorithm results, provided that the right model of color distribution (2D model) is used.
- in step 2, the minimum number *K* of neighbors required for principal components analysis is set to *10*, therefore the algorithm is not applied to isolated points in RGB space. These points keep their initial RGB value throughout the filtering process.
- an additional step has been added:
  3b. Project onto the same affine subspace described in 3 all the neighbors of y_i at distance smaller than δ. These points are no longer processed in the current iteration. δ must be smaller than *T* so the affine subspace found in 3 is a good approximation of the subspace associated to these points. The addition of step 3b implies a maximum reduction of the computation time of the order of $(2\delta+1)^3$. After experimental testing on several images (see section below) we have fixed δ=*T*/2.
- the number of iterations is controlled by ε. We fix it to *0.5*, which implies an average of 3 iterations for LLP2. The value was chosen after experimental testing on several images (see section below).

Statistical analysis of the computation time of the algorithm with parameters fixed to *T*=10, δ=5 and ε=0.5) has been performed using 100 images of the Berkeley test dataset ref5. The average computation time for these images has been computed for decreasing scale factors. The results, summarized in Table 2, show that the computation time has a non-linear dependency on the image size (at least for small scale factors).

scale	*size(pixels)*	*size/size*$_{(scale\ 1)}$	*time/time*$_{(scale\ 1)}$
1	154401	1	1
0.75	86400	0.56	0.59
0.5	38400	0.25	0.33
0.25	9600	0.0625	0.18

Table 2. 100 images from the Berkeley test dataset ref5 have been zoomed down with decreasing scale factors. The size of the images is shown in the second column. The ratio between the size of the zoomed images and their original size is displayed in the third column. The last column shows the ratio between the average computation times of LLP2 for the zoomed images with respect to the computation time for the original images.

Source Code

An ANSI C++ implementation is provided and distributed under the GPL licence: source code, documentation, online documentation

Note from the editor: the source code was updated on July 5, 2011 to fix a bug that occurs on images whose colors are sparsely distributed in RGB space.

Check for source code updates here

Basic compilation and usage instructions are included in the `README.txt` file. This code requires the `libpng` library.

- Linux. You can install `libpng` with your package manager.
- Mac OSX. You can get `libpng` from the Fink project.
- Windows. Precompiled DLLs are available online for `libpng`.

Parameter setting

Tests with T

The distance *T* is used to select the neighboring points which take part in the PCA computation. The value of this parameter is not crucial, as illustrated in the following figure and table. It must be remarked that *T* is a distance in RGB space (not a distance between image pixels) and that in all our experiments 3 bytes per pixel are used to store color information, therefore the maximum Euclidean distance between two points in the color cube is $255x3^{1/2}=441.67$.

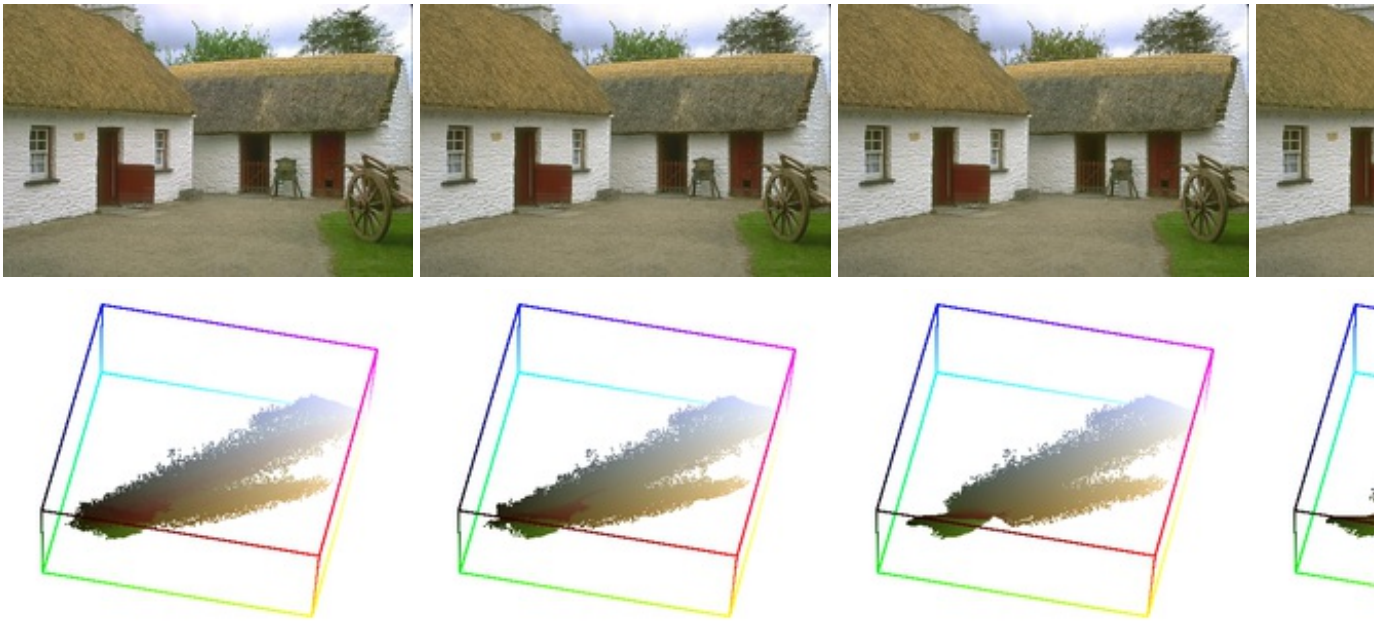


Figure 3. The original image (top-left) is filtered with LLP2, the radius *T* of the filter being respectively 10, 15, 20. The corresponding clouds are in the bottom row. No color alteration is visible, which is consistent with the small variation of the cloud itself. This visual experiment is complemented by the quantitative analysis of Table 3.

In order to numerically verify the 2D filtering stability, the root mean square errors (RMSE) between the color clouds filtered with consecutive radii are shown in Table 3. The errors displayed in this table are the average errors over 100 images from the Berkeley test database ref5. The outcome of the experiment is decisive. The average distance between the cloud and its LLP2 filtered version remains small (less that 5), regardless of the value of the filtering radius *T*. In addition, the distance between filtered points for different values of the radius are also small (of the order of 2), proving that the filtered cloud is essentially the same regardless of the filtering parameter. This table therefore demonstrates the validity of the 2D manifold assumption. In all our experiments we have fixed parameter *T*=10.

	RMSE	*dT=5*
T=3	1.51	2.82
T=5	2.43	2.54
T=10	3.32	2.17
T=15	3.80	2.03
T=20	4.18	1.94
T=25	4.48	1.76
T=30	4.72	1.64

Table 3. Statistical results applying LLP2 (ε=0.5, δ= *T*/2) on 100 images from the Berkeley test database ref5. First column: the radius *T* for LLP. Second column: average RMSE between filtered point and original after applying LLP2. Third column: average RMSE between the final position of a point filtered by LLP2 with radius *T*, and the same point filtered by LLP2 with radius *T*+*dT*, where *dT*=5.

About the computation of RMSE

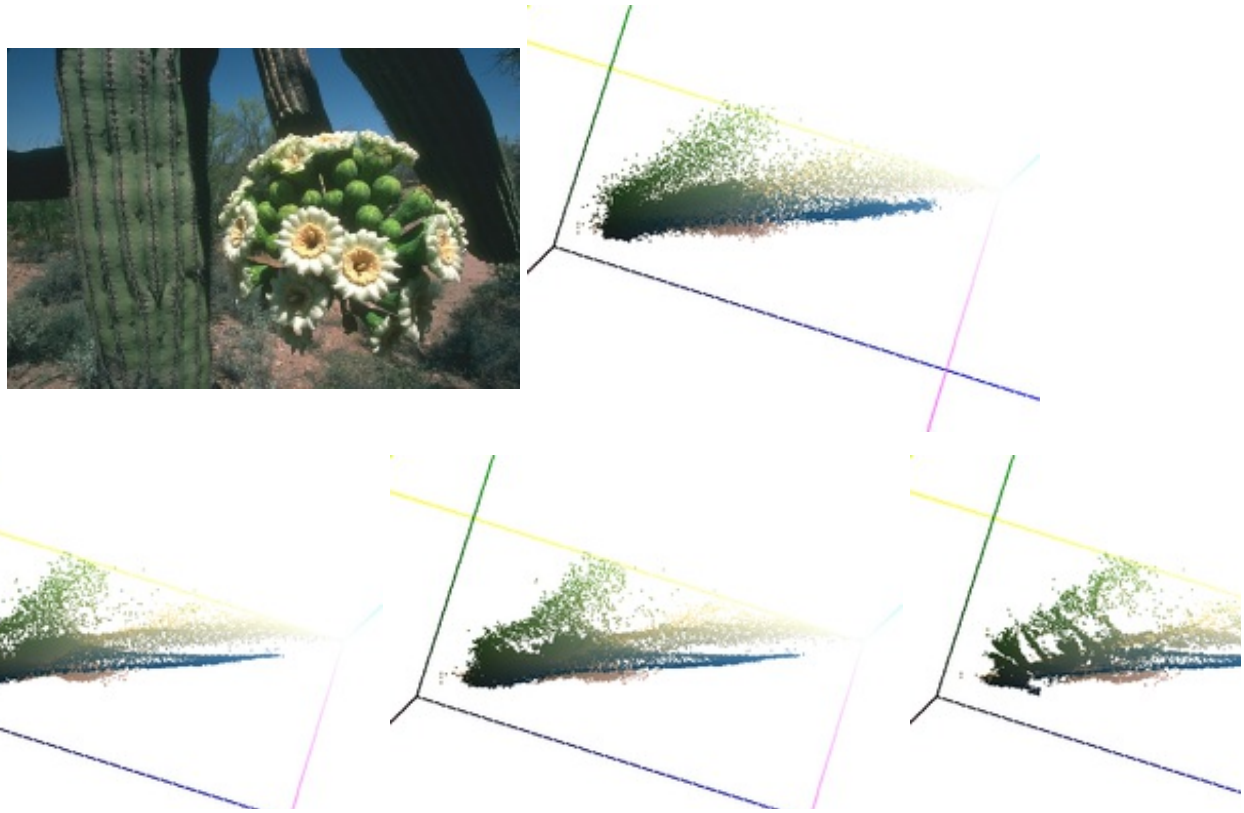
Tests with δ

The parameter δ is used in the algorithm to speed up computations. Table 4 shows the average reductions in computation time for different values of δ (with fixed radius T=10), computed from 100 images of the Berkeley database ref5. The average RMSE between original and filtered images is also shown.

The results show that the computation time is reduced as δ increases, while the average RMSE remains small even for large values of the parameter. This result is explained by the smooth variations in orientation of the local regression planes computed for each RGB point, which makes it possible to approximate the local plane at a given point by planes computed at relatively far apart points. It must be remarked however that RMSE is an average measure over the set of points in a color cloud. Close observation of the clouds shows that when large values of δ are used the filtered color clouds tend to be composed of unconnected 2D manifolds, instead of forming a smooth surface (see Fig. 4). For this reason we have chosen a compromise value δ=T/2=5 (since T=10) in all our experiments.

	$Time/Time_{(\delta=0)}$	*RMSE*
δ=0	1	3.30
δ=1	0.25	3.30
δ=2	0.11	3.30
δ=3	0.065	3.31
δ=5	0.036	3.32
δ=7	0.03	3.37
δ=10	0.028	3.56

Table 4. The first column shows the ratio between the average computation time of the LLP2 algorithm (T=10, ε=0.5) for different values of the speed-up parameter δ with respect to the computation time without acceleration (δ=0). The average RMSE with respect to the original images is also displayed. Averages were taken over 100 images in the Berkeley test dataset ref5.



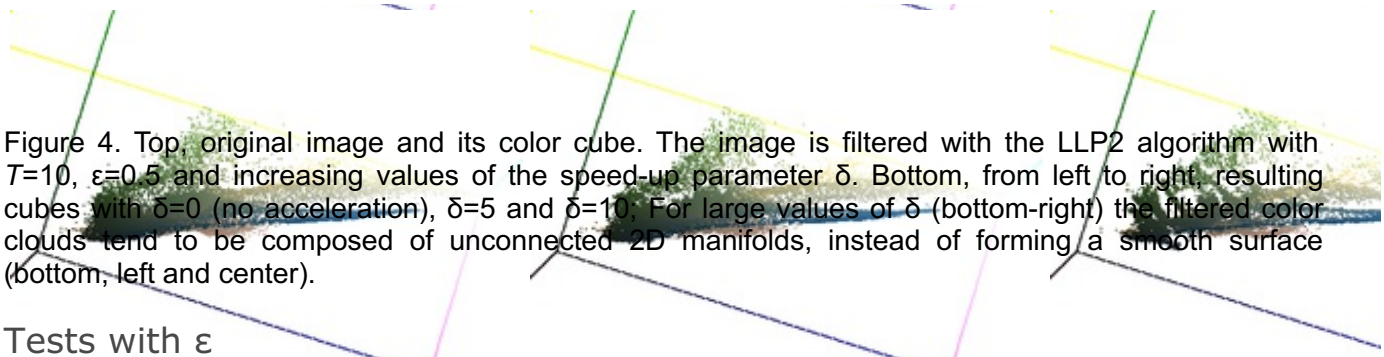


Figure 4. Top, original image and its color cube. The image is filtered with the LLP2 algorithm with *T*=10, ε=0.5 and increasing values of the speed-up parameter δ. Bottom, from left to right, resulting cubes with δ=0 (no acceleration), δ=5 and δ=10; For large values of δ (bottom-right) the filtered color clouds tend to be composed of unconnected 2D manifolds, instead of forming a smooth surface (bottom, left and center).

Tests with ε

ε is used to control the number of iterations of the algorithm. Table 5 displays the average number of iterations for 100 images in the Berkeley test database ref5 for different values of ε. The average RMSE between original and filtered images is also shown. As the number of iterations (and therefore the computation time) increases the dominant 2D structures become more visible, at the expense of losing details of the color distribution (see Fig. 5). As a consequence the difference (RMSE) with the original values increases. A compromise solution has been to set ε=0.5.

	Number of iterations	*RMSE*
ε=0.05	29.3	4.71
ε=0.1	11.92	4.09
ε=0.25	4.51	3.55
ε=0.5	2.91	3.32
ε=0.75	2.48	3.24
ε=1	2.23	3.17

Table 5. The second column shows the average number of iterations of the LLP2 algorithm (*T*=10, δ=5) computed from 100 images in the Berkeley test dataset ref5. Average RMSE with respect to the original images is also displayed. As the number of iterations increases the difference with the original values increases.

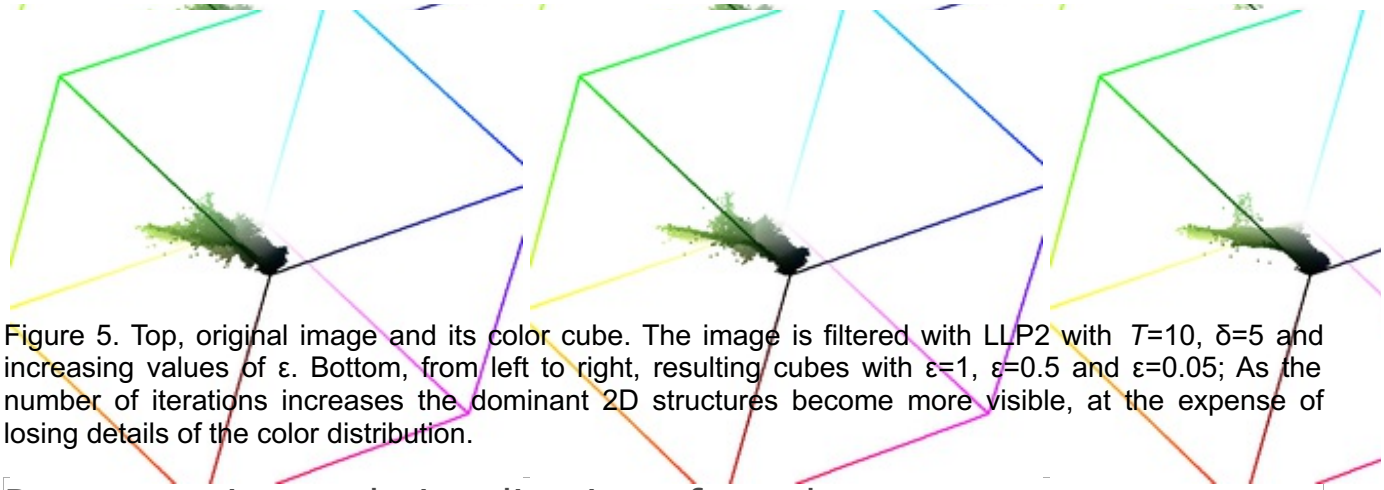


Figure 5. Top, original image and its color cube. The image is filtered with LLP2 with T=10, δ=5 and increasing values of ε. Bottom, from left to right, resulting cubes with ε=1, ε=0.5 and ε=0.05; As the number of iterations increases the dominant 2D structures become more visible, at the expense of losing details of the color distribution.

Preprocessing and visualization of results

In the Overview section it was remarked that isolated colors in the RGB cube make an opaque obstacle to the perspective view of the cloud of RGB points. LLP2 recreates a 2D manifold from the unstructured cloud of points allowing for a better visualization of the colors. However, isolated colors are not eliminated by LLP2 (their number of neighboring colors is below threshold K and therefore their original value is preserved) and they still hinder visualization of the processed cube.

In order to improve the visualization of the original and filtered RGB cubes the following procedure is applied:

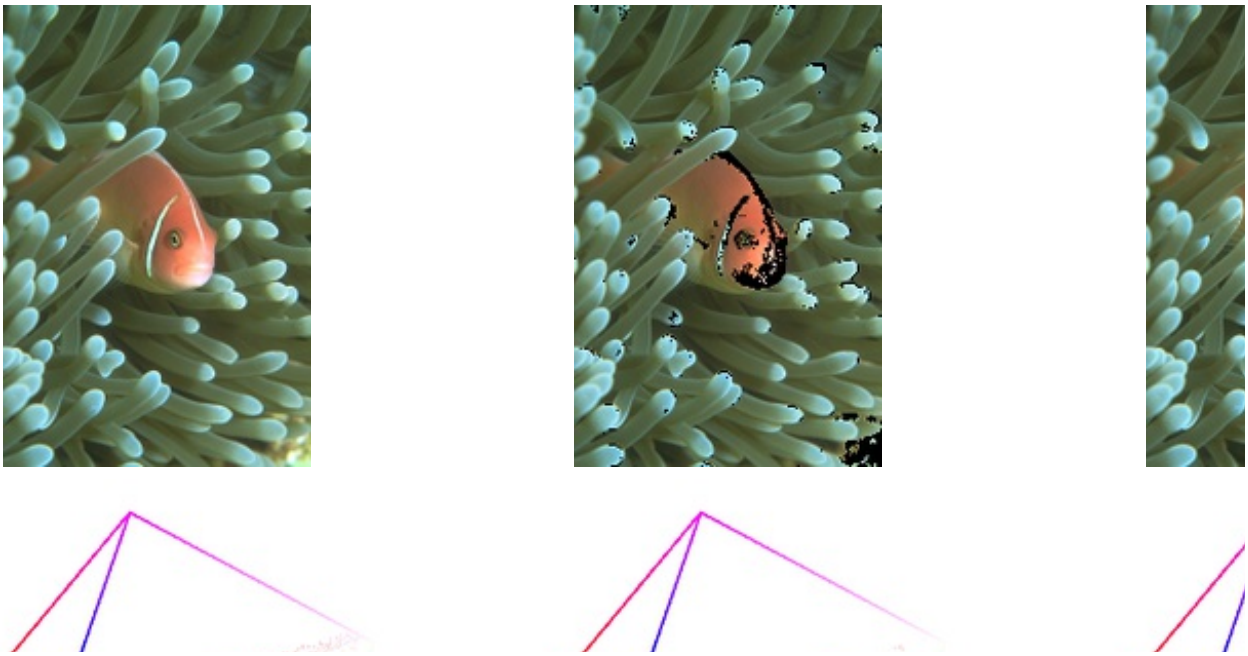
1. Eliminate "isolated" colors from the original image. One color is labeled as "isolated" if its number of neighbors (number of color points at L1-distance ≤ r) is below some threshold $nmin$. Values r=5 and $nmin$=10 are used in all our experiments.
2. Apply LLP2 on the remaining colors.

This procedure is used in the online demo and in the examples below. Results are displayed as follows:

- Original image.
- RGB cube of the original image *after* removal of isolated colors.
- RGB cube after processing with LLP2 the cube from the previous step.
- Filtered image, where pixels with "isolated" colors in the original image keep their original color.

Isolated color points are mainly located on boundaries, and they represent a small fraction of the total number of image colors. Tests with the 100 images of the Berkeley test dataset ref5 show an average value of 2% isolated colors per image. An average of 1% image pixels correspond to isolated colors.

Fig. 6 displays an original image, the resulting image after removal of isolated colors (pixels set to black), the filtered image and their corresponding color cubes.



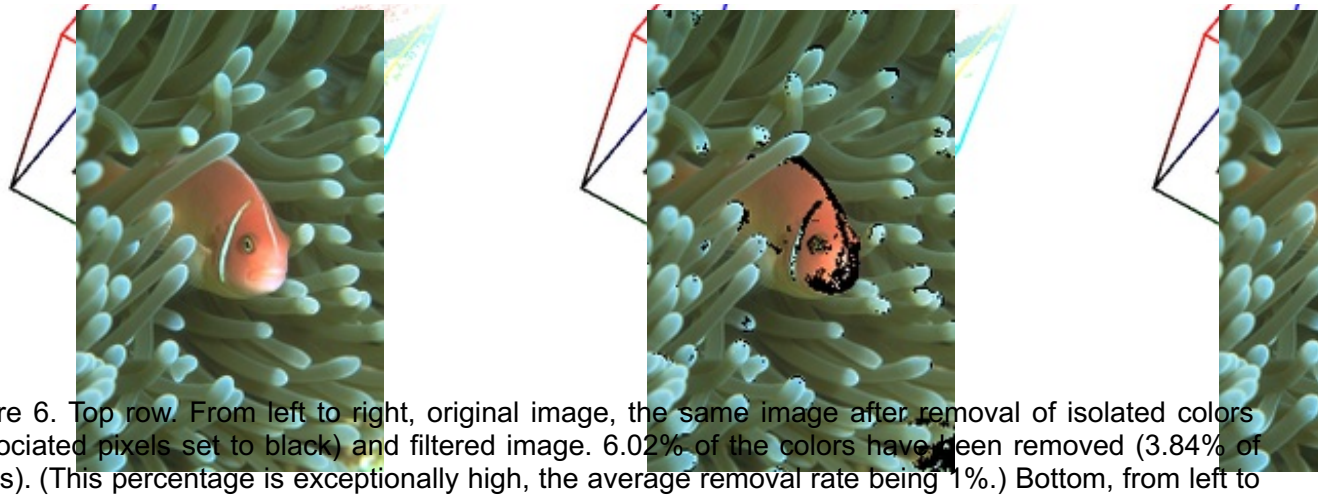


Figure 6. Top row. From left to right, original image, the same image after removal of isolated colors (associated pixels set to black) and filtered image. 6.02% of the colors have been removed (3.84% of pixels). (This percentage is exceptionally high, the average removal rate being 1%.) Bottom, from left to right, original color cube, cube after removal of isolated colors and result of LLP2 on this last cube.

More examples

This section displays some results of the application of LLP2 (including removal of isolated points) on different images from Berkeley test dataset ref5.

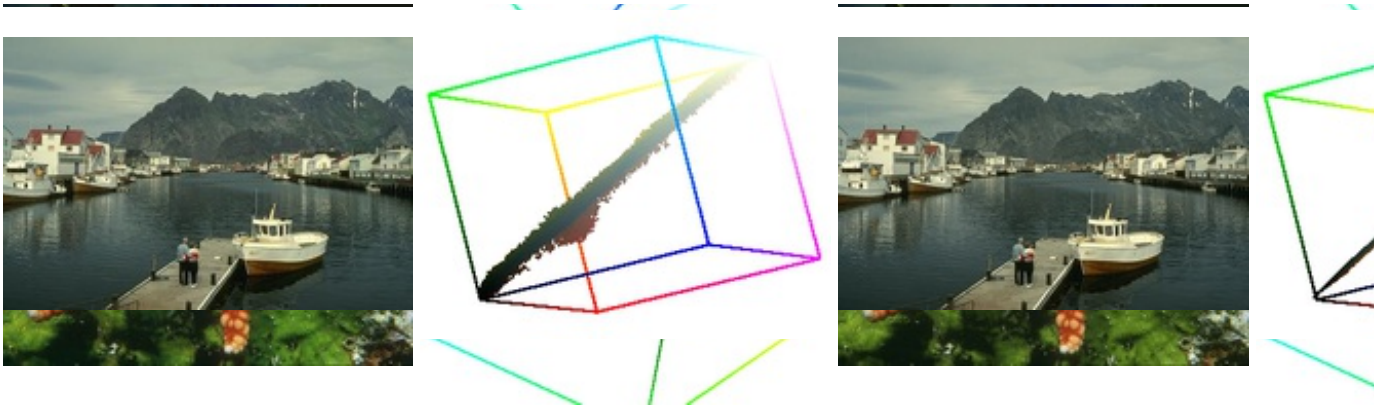


Figure 7. Five images, their original cloud, their filtered cloud by LLP2 (*T*=10, δ=5; ε=0.5) , and the image with the filtered colors. The filtered cloud is a 2D numerical manifold, being a steady state for LLP2. There is no conspicuous alteration in the color cloud when passing to 2D, and the image with filtered colors looks identical to the original.

Color density visualization

The 2D structure of color histograms permits a simple and reliable visualization tool (LLP2) recreating a 2D color manifold from the unstructured color cloud point. On this manifold the color density itself can be displayed by a brightness value proportional to the density of the color point.

Fig. 8 shows another perspective of the filtered point clouds of the images in Fig. 7, and also displays their histogram density (lighter grey levels indicate higher densities). The density is displayed in logarithmic scale, because a few color points can have a huge occupancy (the typical ``blue sky effect ´´) and therefore a linear scale would squeeze the density values. The density is only now visible *because* the filtered clouds are flat and the dense color points in direct sight.

How the color density is computed

Figure 8. Densities of the filtered RGB cubes in Fig. 7. Left: original image. Center: a view of the filtered RGB cube. Right: Same view for the density. The real density is now visible because the cloud has been flattened into a 2D manifold. Otherwise the dense color points would be hidden, being occluded by less dense ones.