Convex Programming Color Constancy

G.D. Finlayson and R.Xu
School of Computing Sciences
University of East Anglia
Norwich NR4 7TJ United Kingdom
Email: graham@sys.uea.ac.uk

Abstract

Gamut mapping color constancy algorithms (originally introduced by Forsyth[For90]) attempt to map RGBs of surfaces viewed under an unknown light to corresponding RGBs under a known reference illuminant. With respect to a reference light source the set of all observable RGBs occupies a 3-dimensional convex region, or gamut, of RGB space. If a triple of 3 simple scalar factors defines the map from image colors to reference conditions then it has been shown that the set of all maps taking RGBs into the reference gamut is also a convex set. Gamut mapping algorithms work in 2 stages. First, the set of feasible maps is computed then in a second stage an optimal member of this map set is chosen.

Here we show that feasible map set does not need to be computed. Rather we combine the final map selection stage with the enforcement of the constraint that image colors are mapped inside the reference gamut. The main contribution of this paper is to show that this combined computation can be set up as a simple convex programming problem. Two main advantages result from this reformulation of gamut mapping. First, several reasonable designations of 'optimal' map can be formulated and tested within the convex programming framework. If we maximize the sum of the triplet of map parameters then color constancy is shown to be a linear programming problem. Maximizing the Euclidean magnitude of the mapping triplet gives a quadratic programming formulation and maximizing the volume of the mapped image gamut (Forsyth's original gamut mapping algorithm) is also possible The second advantage is that convex programming provides a fast solution to the color constancy problem. Indeed, we show that linear programming color constancy is a strictly more efficient implementation of gamut mapping compared to previous methods.

Experiments indicate that, for the Simon Fraser Data set (synthetic and real images), linear programming color constancy provides the best performance over all the convex programming methods tested.

1. Introduction

The RGBs recorded by a color camera depend on the color of the light and the color of the surfaces in the scene. Unless otherwise corrected the same scene viewed under progressively yellower light will result in RGBs which are more and more yellow. Identifying and removing color casts due to the prevailing illumination conditions is called color constancy.

There is a long history of research in color constancy in computer vision. The simplest approaches attempt to identify the illuminant color using a simple statistical estimator. For example, if the average surface color in a scene is grey then the average RGB in a image will have the same color as the scene illumination[Buc80, GJT88]. Of course the average scene color is rarely grey and so this 'grey-world' approach does not work well in practice. Alternatively, the maximum R, G and B can be used as an illumination estimate[Fin97, Lan77]. This estimator is justified in a number of scenarios. If for example, there is a white reflectance in a scene then the maximum RGB present will correspond to the RGB of white and this will give a correct estimate of the illuminant color. Moreover, even if white is not present but say there is a bright blue and a bright yellow then the maximum RGB still gives a correct estimate (since Blue is indistinguishable from white for the blue channel and yellow is indistinguishable from white for the Red and Green channels). The max RGB assumption turns out to deliver much better illuminant estimation than the grey world algorithm: though, it is still wrong much of the time. There are also many other statistical methods used[Sap98, DI94, Yui87, MW86]. While the most recent approaches[FHH01, RHT01, BF97] do deliver improved constancy, they still fail some of the time.

Indeed, such failures must be expected: color constancy is an inherently ill posed task. Consider a yellow image recorded for a surface of unknown color viewed under an unknown illuminant. It is possible that the illuminant is yellow and the surface is white or vice versa. That is, based on the available evidence it is not possible to solve the color constancy problem. Forsyth[For90] placed this observation

at the heart of his gamut mapping algorithm for color constancy. In gamut mapping color constancy one attempts to solve for all feasible answers to color constancy. It is only at a second stage where a single 'optimal' answer is chosen. In contradistinction to other color constancy algorithms, when a choice is made there is enough information to give a measure of certainty with the estimate[FH99].

So how does gamut mapping work? The key insight is that the range, or gamut, of colors should depend on illumination. Intuitively, we do not expect to observe the bluest RGB under a reddish light. To understand better how RGBs depend on illumination let us measure RGBs for as many surfaces as we can find with respect to a particular reference light. Of course it possible that our set of surfaces is not complete and so we allow convex combinations of the surfaces and this results in convex combinations of the corresponding RGBs. Note this operation is perfectly legal: we are merely simulating what happens when two or more surfaces (present in some proportion in some area of a scene) are mapped (averaged) to the same pixel. Thus, the gamut of reference colors, henceforth denoted Γ, is a closed convex region of RGB space.

Now let us suppose we record an image under a second unknown illuminant. How can the color bias due to the illuminant be removed? In gamut mapping we succeed in solving the color constancy problem if we can map the image gamut (the convex closure of the image RGBs) so that is inside the reference gamut. But, what do we mean when we say 'map image colors'? It is common (and almost always justified[FDF94] to use a simple set of three $[\alpha\ \beta\ \gamma]$ scaling factors as the map that models illumination change. Under a change in illumination $[R\ G\ B]$ is mapped according to

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} \Rightarrow \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

That is, $[\alpha\ \beta\ \gamma]$ parameterize a diagonal matrix. Using the subscript $_i$ to denote the ith image RGB then $[\alpha\ \beta\ \gamma]$ is a solution to gamut mapping color constancy if and only if

$$\forall(i) \quad \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix} \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix} \in \Gamma \quad (2)$$

Or, equivalently a diagonal matrix $\mathcal{D}$ is a solution for color constancy if and only if it maps all image RGBs into Γ.

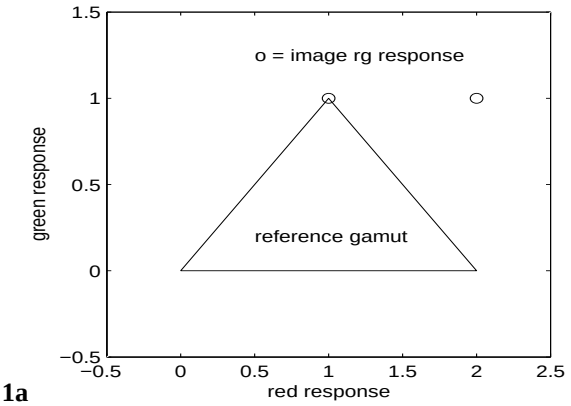
Forsyth showed (see also section 2 below) that for each $[R_i\ G_i\ B_i]$ the set of $[\alpha\ \beta\ \gamma]$ scaling parameters is a 3 dimensional convex set. It follows then that to determine the set of all maps taking all image colors inside the reference gamut involves intersecting the mapping sets computed for each image color. In fact the task is not quite as onerous as it sounds since it can be shown that we only need to examine the mapping sets corresponding to points on the convex hull of the image RGBs (though this can still be large). The cost of intersecting two 3-dimensional convex polyhedra is $O(n \log n)$ (assuming both polygon have order n points). Given reference and image gamut with $O(n)$ points the upper bound complexity for computing the set of all plausible maps is $O(n^2 \log n)$.

We sketch 2-dimensional gamut mapping in Figure 1 (i.e. the gamut mapping problem for a 2 sensor camera). In Figure 1a we show a reference gamut (the closed triangle) and an image containing 2 pixels (the open circles at (1,1) and (2,1)). In gamut mapping we wish to find the set of **feasible maps** that take the two image colors within the reference gamut. To do this we calculate the maps taking (1,1) inside the reference gamut and the maps for (2,1). We then intersect the two mapping sets. This process is illustrated in Figure 1b.

In this simple example the overall feasible map set contains many solutions. Yet in color constancy we seek a single answer. Thus, at a second stage an optimality criterion is applied in order to select a single answer. Because the gamut of the image is, in (2), mapped by a diagonal matrix, the mapped image gamut volume must be proportional to the determinant of the map. The maximum determinant map is found at $(2/3, 2/3)$. So, $(2/3, 2/3)$ is the gamut mapping solution to color constancy.

The main contribution of this paper is to show that equation (2) coupled with the specification of optimal estimator can be cast as a convex program. Specifically we show that (2) implies a set of linear inequalities (defining a convex region in which the optimum must lie) with respect to which we can maximize an objective function. We believe there are two main advantages of convex programming color constancy. First, in convex programming we might explore many reasonable definitions of optimal feasible map. In this paper we consider the maximum determinant map proposed by Forsyth, the map which maximizes the L_1 norm (sum of the diagonal matrix terms) and the L_2 norm (Euclidean length of the map). The second advantage is that convex programming is generally a rapid computational procedure Indeed, linear programming color constancy (a particular instantiation of convex programming) is, in a computational complexity sense, strictly faster than conventional gamut mapping. Finally we remark that that convex programming methods are often simple to implement (e.g. the simplex method) and their operating characteristics are well understood.

Of course we might wonder whether the choice of optimality constraints makes much difference. In each case L_1, L_2 and max determinant norms are all measures of the magnitude of the map (note the determinant is the cube of the geometric mean). How much might the solution depend on the definition of magnitude used? We explore this question in Figure 2 where we show that it is possible that the



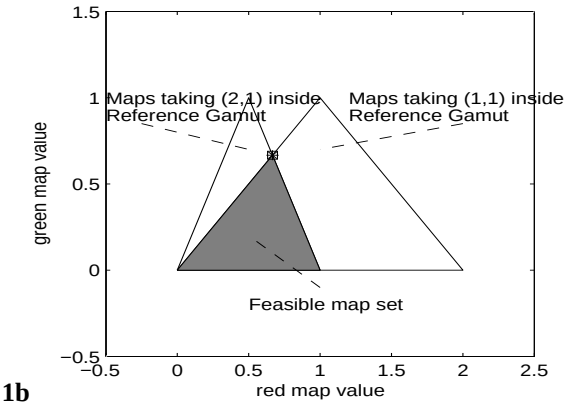


Figure 1: In Fig 1a the reference gamut and 2 image points (1,1) and (2,1) are shown. In 1b the map sets taking (1,1) and (2,1) inside the reference gamut. The intersection of maps sets (grey region) and optimal map (dark point) are also indicated.

optimal map can vary significantly depending on the norm used.

At the top of Figure 2 a hypothetical feasible set of solutions to color constancy is shown (where as in Figure 1 we, for ease of illustration, consider gamut mapping in 2 dimensions). The top panel shows the set of feasible maps. Any diagonal matrix parameterized by the (x, y) coordinate within the region is bounded by the x and y axes and the upper curve is a solution to color constancy. In the 2nd panel of Figure 2 we move along the x axis and for each x we find the y in the gamut that maximizes the L_1 norm $x + y$ (this is always a point on the curve). Thus, as a function of x (the 1st component of the diagonal transform) we can plot the maximum L_1 norm. According to this metric it is clear the best map overall x to choose is in a region around $x = 0.7$ (but points close to either side of 0.7 are equally good). The third panel shows a similar plot for the L_2 norm. Here the best map is at the ends of the interval (a wide range of points close to 0 or 1). The last panel shows the maximum determinant xy as a function of x. Clearly the maximum determinant overall is again near 0.7. Only the maximum determinant criterion gives a unique maximum.

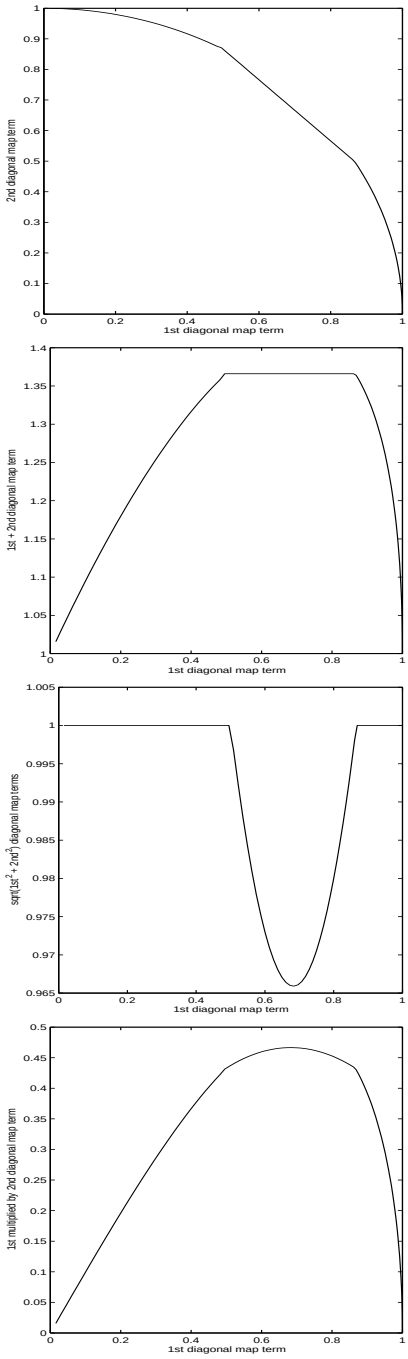


Figure 2: Top: 2-dimensional feasible map set. Second: the maximum L_1 plotted as a function of the first diagonal component. Third left: maximum L_2 norm as function of first diagonal component. Bottom maximum determinant as a function of the first diagonal component.

The reader should be aware that we chose the feasible set shown at the top of Figure 2 in order to arrive at different outcomes given our three optimality criterion. However, experiments demonstrate that real constancy results do differ dependent on which criterion is used: we found that optimizing the L_1 norm leads to the best constancy performance. This, from our point of view, is serendipitous: the fastest convex programming solution to color constancy also delivers the best performance.

In casting gamut mapping as convex programming we note that existing gamut mapping algorithms probably use convex programming to find the optimal map from the computed map set (we say probably because the implementation of optimal map selection is never discussed in the literature). For example maximizing $\alpha+\beta+\gamma$ subject to $[\alpha\ \beta\ \gamma]$ belonging to the set of feasible maps is naturally solved as a linear program.

2. Gamut Mapping

Formally, the reference gamut Γ is defined as:

$$\Gamma = C(\mathcal{P}) \tag{3}$$

where $\mathcal{P}$ are RGBs recorded for surfaces viewed under the reference light.

$$\mathcal{P} = \{\underline{p}_1, \underline{p}_2, \cdots, \underline{p}_m\} \tag{4}$$

The function $C()$ returns the set of all convex combinations of $\mathcal{P}$. An RGB $\underline{x}$ is in Γ if it is a convex combination of the points in $\mathcal{P}$:

$$\underline{x} = \sum_{i=1}^{m} w_i \underline{p}_i \ ,\ \sum_{i=1}^{m} w_i = 1 \ ,\ w_i \geq 0 \tag{5}$$

Suppose that if $\mathcal{P} - \underline{p}_i$ denotes the set $\mathcal{P}$ with the ith RGB removed then, without loss of generality, we can choose $\mathcal{P}$ such that

$$\underline{p}_i \neq C(\mathcal{P} - \underline{p}_i) \tag{6}$$

In this case $\mathcal{P}$ is the set of points on the convex hull of the reference gamut. We make this choice as the complexity of gamut mapping depends on the size of $\mathcal{P}$ (and $\mathcal{Q}$ below).

Let $\mathcal{Q}$ denote the n points on the convex hull of the set of image RGBs (it is these RGBs we wish to map inside the reference gamut):

$$\mathcal{Q} = \{\underline{q}_1, \underline{q}_2, \cdots, \underline{q}_m\} \tag{7}$$

The image gamut I is defined as:

$$I = C(\mathcal{Q}) \tag{8}$$

A triple of scalars $\underline{d}$ taking the jth point of the convex hull of the image gamut inside Γ satisfies:

$$diag(\underline{d})\underline{q}_j \in \Gamma \tag{9}$$

where the function *diag* places the 3 components of $\underline{d}$ along the diagonal of a diagonal matrix. Equivalently we can interchange the positions of $\underline{d}$ and $\underline{q}_j$ and write:

$$diag(\underline{q}_j)\underline{d} \in \Gamma \tag{10}$$

Let $\mathcal{D}_j = [diag(\underline{q}_j)]^{-1}$. Then

$$\underline{d} \in \mathcal{D}_j\Gamma \tag{11}$$

The meaning of (11) is that the set of maps taking the jth image gamut point into the reference gamut is a diagonal matrix transform from the reference gamut Γ: the mapping set is itself a bounded 3-dimensional convex region. Formally the set of maps $\mathcal{M}_j$ taking the jth point on the convex hull of the image gamut is defined as:

$$\mathcal{M}_j = C(\{\mathcal{D}_j\underline{p}_1, \mathcal{D}_j\underline{p}_n, \cdots, \mathcal{D}_j\underline{p}_n\}) \tag{12}$$

By intersecting the n individual mapping sets we arrive at the feasible set of maps i.e. any map in the feasible set maps all $\underline{q}_i$ inside the reference gamut:

$$\mathcal{M} = \bigcap_{i=1}^{n} \mathcal{M}_j \tag{13}$$

Of course if a diagonal transform in $\mathcal{M}$ takes the points $\mathcal{Q}$ within the reference gamut then it follows that convex combinations of $\mathcal{Q}$ are also mapped inside Γ. $\mathcal{M}$ is exactly the set of maps that takes I inside Γ.

In order to find a single member of $\mathcal{M}$ as an illuminant estimate we must introduce an optimality criterion. For example we might find the $\underline{d} \in \mathcal{M}$ such that $|\underline{d}|_1$ is maximized (where $|.|_1$ denotes L1 norm). Equation (14) is a formal statement of the gamut mapping problem.

$$\max_{\underline{d} \in \mathcal{M}} |\underline{d}|_1 \tag{14}$$

3. Gamut mapping and convex programming

Both the reference and image gamuts and the mapping sets are all 3-d convex polyhedra. So, one of the central questions that must be addressed in gamut mapping is the representation of of these convex sets. It is normal practice to represent convex sets as convex combinations of the vertices of the convex hull (as we reviewed in the last section). Alternately, we can represent the interior of the convex hull as the intersection of a set of half spaces. As we shall see this second representation leads to a natural convex programming formulation of (14).

To make this second representation clear consider the 2-d triangle is shown in Figure 1a. The interior of the triangle is defined by convex combinations of the vertices (0,0), (2,0) and (1,1). Let us now consider the line joining (2,0) and (1,1). This line can be written as $y = -x + 2$ or $-x - y = -2$. It is straightforward to show that points that fall below this line (e.g. (0,0)) that $-x - y > -2$ and points above the line $-x - y < -2$. As such we say that $-x - y > -2$ defines the half space (it splits the plane into 2) containing all points below the line. Clearly then we can define the interior of the triangle by the intersection of 3 half spaces:

$$\begin{array}{c} -x - y \geq -2 \\ x - y \geq 0 \\ y \geq 0 \end{array} \tag{15}$$

In 3-dimensions, half-spaces are defined by plane equations of the form

$$ax + by + cz \geq e \tag{16}$$

A 3-dimensional convex hull that has n vertices has N faces where N is $O(n)$ (proportional to the number of vertices). Thus, we can write the reference gamut Γ as:

$$\Gamma = C(\mathcal{P}) = \begin{array}{c} a_1R + b_1G + c_1B \geq e_1 \\ a_2R + b_2G + c_2B \geq e_2 \\ \vdots \\ a_NR + b_NG + c_NB \geq e_N \end{array} \tag{17}$$

where $[R\ G\ B]$ is a point in the reference gamut. Let us define an $N \times 3$ matrix A and place in the ith row: $[a_i\ b_i\ c_i]$. Further let $\underline{e}$ be an $N \times 1$ vector whose ith component is equal to e_i. If $\underline{p}$ is an RGB within the reference gamut then the following must be true:

$$A\underline{p} \geq \underline{e} \tag{18}$$

Let $diag(\underline{d})\underline{q}$ denote an image rgb transformed by a diagonal matrix which is within the reference gamut. Then by substitution:

$$Adiag(\underline{d})\underline{q} \geq \underline{e} \tag{19}$$

or equivalently interchanging $\underline{d}$ and $\underline{q}$:

$$Adiag(\underline{q})\underline{d} \geq \underline{e} \tag{20}$$

It follows that the gamut of maps for the jth point on the convex hull of the image gamut I can be written as (Equation 12 can be rewritten as):

$$\mathcal{M}_j = Adiag(\underline{q}_j)\underline{d} \geq \underline{e} \tag{21}$$

Assuming there are $O(N)$ points on the image gamut then there $O(N)$ sets of inequalities of the form (20) and so $O(N^2)$ inequalities in total. Taken together these inequalities demarcate the set of maps which take all the image colors inside the reference gamut. We call this set, denoted $\mathcal{M}$, the **feasible map set**:

$$\mathcal{M} = \mathcal{A}\underline{d} \geq \underline{\mathcal{E}} \tag{22}$$

where $\mathcal{A}$ has the matrices $Adiag(\underline{q}_j)$ stacked one on top of the other and $\underline{\mathcal{E}}$ is a vector with multiple copies of the vector $\underline{e}$.

Of course we must find a single diagonal map $\underline{d}$ in $\mathcal{M}$. One way we can do this is to optimize some property e.g. find the $\underline{d}$ in $\mathcal{M}$ with maximum L_1 norm. Maximizing an objective function (such as L_1 norm) subject to linear inequalities is called convex programming. By changing the objective function we can explore different variants of convex programming color constancy.

1. **Maximum L_1 norm**

   Maximize $d_1 + d_2 + d_3$ subject to the constraint that

   $$\mathcal{A}\underline{d} \geq \underline{\mathcal{E}}$$

   This is a linear program whose solution can be found in $O(N^2)$ (where there are $O(N^2)$ inequality constraints). In fact there are several linear time algorithms, see for example[Cla56], but even the simple Simplex search algorithm has expected linear complexity in the number of constraints[Chv83].

2. **Maximum L_2 norm**

   Maximize $d_1^2 + d_2^2 + d_3^2$ subject to the constraint that

   $$\mathcal{A}\underline{d} \geq \underline{\mathcal{E}}$$

   This is a quadratic programming problem. For the given objective function the optimal solution can be found in $O(N^2 \log N)$ (since the max L_2 solution is always a vertex and vertices can be solved for in this time bound). In experiments we found a search based Quadratic Programming method to be as fast as linear programming.

3. **Maximum determinant (max volume)**

   Maximize $d_1 d_2 d_3$ subject to the constraint that

   $$\mathcal{A}\underline{d} \geq \underline{\mathcal{E}}$$

   Here we maximize the determinant of the diagonal matrix taking the image gamut within the reference gamut. Unfortunately, determinant maximization is a more difficult problem and while the maximum can be found, it is found at significant computational cost[VBW98] (it is significantly slower than linear or quadratic programming).

The complexity of the linear programming formulation is particularly favourable. The time to find the optimal solution is of the same order as the time it takes to enumerate the plane constraints. However, if we were to calculate the feasible set $\mathcal{M}$ prior to maximizing the L_1 normal (the normal procedure in gamut mapping) we would need to intersect all $O(N^2)$ planes. It is known that the optimal algorithm for computing the intersection of n planes in 3-space has complexity $O(n \log n)$[PM79]. it follows then that to compute the vertices in $\mathcal{M}$ costs $O(N^2 \log N^2) = O(N^2 \log N)$. Thus, carrying out an explicit computation of the feasible region is strictly more expensive $O(N^2 \log N)$ than the linear programming formulation proposed above ($O(N^2)$).

A couple of further comments on complexity are useful here. First, the complexity results quoted for plane intersection are for the general case. In gamut mapping the planes generated for different image RGBs are related to one another (by a diagonal matrix). Perhaps, plane intersection for gamut mapping has lower complexity? Also, it has been shown (e.g. see [FHH01]) that a discrete approximation to gamut mapping has a very fast implementation. The complexity result above however remains important as it says something fundamental about the hardness of gamut mapping per se independent of implementation.

4. Experiments

A priori we know that the convex programming color constancy method should work. After all, it can be viewed as an alternate, albeit flexible, implementation of Forsyth's gamut mapping algorithm. However, experiments will cast light on whether a max L_1 norm, max L_2 norm or max determinant criterion affords the best constancy performance (or whether indeed the criterion makes much difference).

We begin by running a test on synthetic images using the dataset proposed by Barnard[BMFC02]. This dataset consists of 1995 reflectances, 81 illuminants and the spectral sensitivity curves of a SONY DXC-930 digital camera. Using numerical integration we can create a set of RGBs for a range of surfaces viewed under one of the illuminants.

To form the reference gamut we calculated the convex hull of 1995 RGBs for all the surfaces viewed under a cool white fluorescent light (Sylvania 50MR16Q). The convex hull is represented in the plane format set forth in (18). We now randomly selected k surfaces (where $k = 4, 8, 16, 32, 64$) and 1 of the 81 illuminants. Then k RGBs are generated using numerical integration. The vertices of the image gamut is calculated and these are used to set up the convex programs. We then solve for the optimal diagonal map taking the image gamut into the reference gamut. For each value of k we repeat this experiment 1000 times.

In order to evaluate color constancy performance we must measure in some way our estimate of the illuminant. Note, that the diagonal matrix mapping image RGBs into the reference gamut is not a direct measure of illumination but it is indirectly related to it. In particular if $\underline{p}_w$ and $\underline{q}_w$ are respectively the RGB of a white surface under reference and image illuminants and the map to the reference light is denoted $\mathcal{D}$ then it should follow that $\underline{p}_w \approx \mathcal{D}\underline{q}_w$ (assuming the algorithm is working well). Equally $\mathcal{D}^{-1}\underline{p}_w \approx \underline{q}_w$. We define our estimate of the white point of the image illuminant to be $\hat{\underline{q}}_w = \mathcal{D}^{-1}\underline{p}_w$. The error in in the estimate is calculated:

$$\text{recovery error} = angle(\hat{\underline{q}}_w, \underline{q}_w) \tag{23}$$

In Figure 3, we plot recovery error (averaged over 1000 cases) for 4, 8 ,16, 32 and 64 surfaces for convex programming color constancy where we maximize the L_1 norm, the L_2 norm and the determinant. Clearly, all methods return comparable constancy performance though the L_1 norm is perhaps a little better overall and the L_2 norm a little worse. That L_1 is at least as good as the other optimality criteria is encouraging since L_1 optimization is computationally easier than L_2 or determinant maximization.

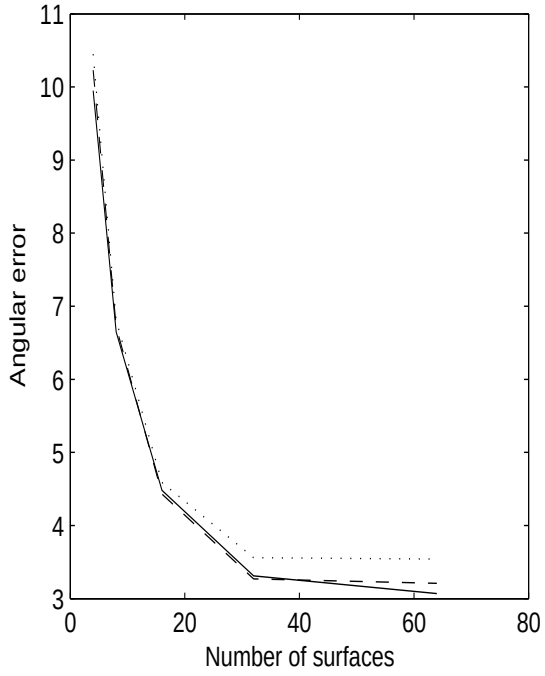


Figure 3: Recovery error as a function of the number of surfaces in a scene. Solid line for max L_1 norm, dashed line for max L_2 norm and dotted line for max determinant.

In Figure 4, we plot the recovery error for the max RGB and grey-world algorithms (see introduction for a description) and show for comparison the performance of the L_1

convex programming method. It is clear that the grey world method delivers poor constancy and that as there are more surfaces in a scene the performance of the max rgb algorithm and convex programming converge; though, convex programming is always provides significantly better estimates. This is as we might expect as eventually, if we choose enough surfaces it is likely the conditions for max rgb to produce good constancy (see discussion in the introduction).

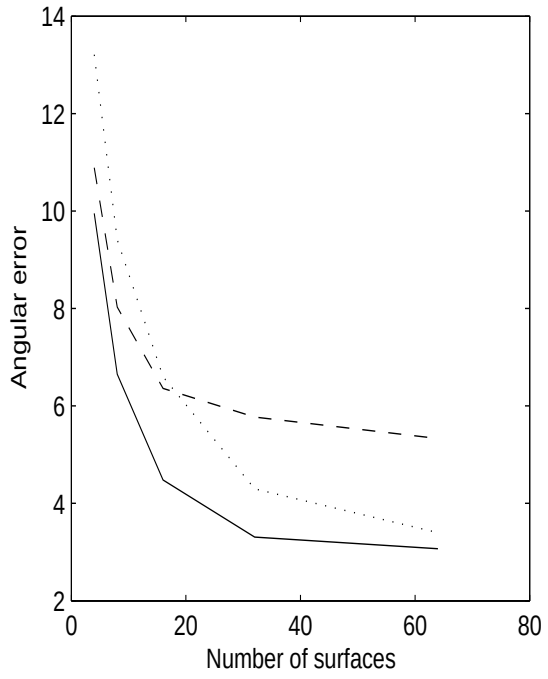


Figure 4: Recovery error as a function of the number of surfaces in a scene. Solid line for max L_1 norm, dashed line for grey world and dotted line for max rgb.

In a second experiment we wished to evaluate convex programming color constancy on real images. To this end, we used the Simon Fraser Calibrated test set comprising 321 images[BMFC02]. These images are of 30 objects viewed under up to 11 colored lights (almost all objects are imaged under all 11 lights). We preprocess images to remove very dark and very bright pixels. Specifically all images are normalized by applying a single scalar such that the brightest image value (in either the red, green or blue channel) is 255. We then reject any pixels where R, G or B is less than 5 or larger than 245. The lower hard threshold helps attenuate the effect of image noise and the large threshold helps to deal with images containing specular highlights or where there is clipping.

Again cool white fluorescent (Sylvania 50MR16Q) is used as our reference illuminant. We take all 11 images and apply the hard thresholding described above and then build the canonical gamut. For each of the remaining 291 images we compute convex programming color constancy and also the max RGB and grey world estimates of the illuminant color. Along with each image in the Simon Fraser Data set there is a measurement of the RGB of the white patch (for the 11 viewing illuminants). Thus, we can calculate the recovery error as before. Results are summarized in Table 1.

Algorithm	Recovery Error
Convex programming (L_1)	5.54
Convex programming (L_2)	5.53
Convex programming (max det)	5.54
Grey world	13.57
Max RGB	12.27

Clearly convex programming affords significantly better constancy performance than the simple grey world or max rgb algorithms. The figure of 5.5 is a little lower than the 5.8 degree angular error quoted for the max volume color constancy that Barnard arrives at in[Bar99] but a little higher than the 4.7 he reports in[BMCF02, BCF02]. This variation in performance is due to two factors: the definition of the reference gamut and the nature of the preprocessing. Convex programming color constancy is formally equivalent to conventional gamut mapping and so given the same inputs (the same reference gamut and the same preprocessed images), we must arrive at the same solution.

5. Conclusion

Gamut mapping algorithms are among the most promising solutions to color constancy. However, they are quite complex: they have high computational complexity and, not insignificantly, they are difficult to implement. In this paper we have shown that one variant of gamut mapping can be implemented as a linear program. Not only is linear programming less complex both in terms of complexity and the difficulty of implementation but it also helps us better understand gamut mapping per se. Indeed, we show that gamut mapping is intrinsically a constraint satisfaction problem where we attempt to find an optimal illuminant estimate subject to linear inequalities which are intrinsic to the problem. The linear programming solution is a special case where we use a linear objective function. However, we also show it is possible to optimize an L_2 norm and a volume-based norm.

We calculate color constancy performance for the Simon Fraser Data set and show that convex programming color constancy provides results consistent with previous implementations of gamut mapping. Moreover, the performance

of the L_1 norm (or linear programming solution to color constancy) is better (for the synthetic test data and no worse for real images) than the other objective functions.

Acknowledgments

The authors gratefully acknowledge the support of the Hewlett Packard Corporation. Ruixia Xu is grateful for the financial support of the University of East Anglia.

References

[Bar99] K. Barnard. Practical olour constancy, 1999. PhD thesis, Simon Fraser University, School of Computing Science.

[BCF02] K. Barnard, V. Cardei, and B.V. Funt. A comparison of computational color constancy algorithms-part i: Methodology and experiments with synthesized data. *IEEE Transactions on Image Processing*, 11(9):972–983, September 2002.

[BF97] David H. Brainard and William T. Freeman. Bayesian color constancy. *Journal of the Optical Society of America, A*, 14(7):1393–1411, 1997.

[BMCF02] K. Barnard, L. Martin, A. Coath, and B.V. Funt. A comparison of computational color constancy algorithms-part ii: Experiments with image data. *IEEE Transactions on Image Processing*, 11(9):985–996, September 2002.

[BMFC02] K. Barnard, L. Martin, B. Funt, and A. Coath. A data set for colour research. *COLOR research and applications*, 27(3):147–151, 2002.

[Buc80] G. Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin Institute*, 310:1–26, 1980.

[Chv83] V. Chvatal. *Linear Programming*. Freeman, 1983.

[Cla56] K.L. Clarkson. A las vegas algorithm for linear programming when the dimension is small. In *Foundations of Computer Science*, 452-456.

[DI94] M. D'Zmura and G. Iverson. Probabalistic color constancy. In R.D. Luce, M. M. D'Zmura, D. Hoffman, G. Iverson, and K. Romney, editors, *Geometric Representations of Perceptual Phenomena: Papers in Honor of Tarow Indow's 70th Birthday*. Laurence Erlbaum Associates, 1994.

[FDF94] G.D. Finlayson, M.S. Drew, and B.V. Funt. Spectral sharpening: sensor transformations for improved color constancy. *J. Opt. Soc. Am. A*, 11(5):1553–1563, May 1994.

[FH99] G.D. Finlayson and S.D. Hordley. Colour constancy with error bars. In *IEE conference on Image Processing and Applications*, 1999.

[FHH01] G.D. Finlayson, S.D. Hordley, and P.M. Hubel. Color by correlation: A simple, unifying framework for color constancy. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1209–1221, November 2001.

[Fin97] G.D. Finlayson. Retinex viewed as a gamut mapping theory of colour constancy. In *The 8th Congress of the International Colour Association*, 1997. to appear.

[For90] D. Forsyth. A novel algorithm for color constancy. *Int. J. Comput. Vision*, 5:5–36, 1990.

[GJT88] R. Gershon, A.D. Jepson, and J.K. Tsotsos. From $[r, g, b]$ to surface reflectance: Computing color constant descriptors in images. *Perception*, pages 755–758, 1988.

[Lan77] E.H. Land. The retinex theory of color vision. *Scientific American*, pages 108–129, 1977.

[MW86] L.T. Maloney and B.A. Wandell. Color constancy: a method for recovering surface spectral reflectance. *J. Opt. Soc. Am. A*, 3:29–33, 1986.

[PM79] F. Preparata and D. Muller. Finding the intersection of n halfspaces in time o(nlogn). *Theoretical Computer Science*, 8:45–55, 1979.

[RHT01] C. Rosenberg, M. Hebert, and S. Thrun. Color constancy using kl-divergence. In *ICCV01*, pages I: 239–246, 2001.

[Sap98] G. Sapiro. Bilinear voting. In *IEEE International Conference on Computer Vision*, pages 178–183, 1998.

[VBW98] L. Vandenberghe, S. Boyd, and S.P. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19(2):499–533, 1998.

[Yui87] A. Yuille. A method for computing spectral reflectance. *Biological Cybernetics*, 56:195–201, 1987.