Automatic Relighting of Overlapping Textures of a 3D Model

Etienne Beauchesne Sébastien Roy
Département d'informatique et recherche opérationnelle (Université de Montréal)
{beauchee, roys}@iro.umontreal.ca

Abstract

This paper presents a new method to correct overlapping textures of a single 3D model where each texture was obtained under possibly different lighting conditions and color response of the camera. This situation arises frequently when a single object is digitized using multiple 3D scanners. Our goal is to remove any visible seam and improve color consistency between the textures in order to merge them afterward. To achieve this, we propose an ef£cient algorithm to compute the "ratio lighting" of two textures, derive a common lighting from it, and use it to "relight" each texture. We illustrate our method by correcting textures of human faces acquired with several structured-light 3D scanners. Experimental results are realistic and demonstrate how this method can reduce the need to calibrate colors or explicitly solve for the illumination.

1 Introduction

Let us suppose that the object of interest is £xed and images of it are acquired by one or more cameras. We are given a geometric model of the object, which may come from merging of several 3D models. For example, in the case of faces, we merge three 3D geometries. The viewpoints and the illumination may be different for each acquisition. Each image is used to create a texture and we receive the correspondence between the texture and the 3D model from a previous step.

The problem we address here is how to correct these textures so that they can be merged. The elementary operation in our method is to take two textures and correct them relatively to each other. We consider our technique as a preprocessing step for *texture merging* (for example Debevec [6]). Our main concern is that the result should look as realistic as possible. See Figure 1. We will talk later about the constraints on textures.

A common approach to the problem of texture merging is to simply remove the seams near texture boundaries, the constraint being that the textures be modi£ed as little as possible. Unfortunately, it lacks realism, because the merged texture may have non-coherent colors, for example if one

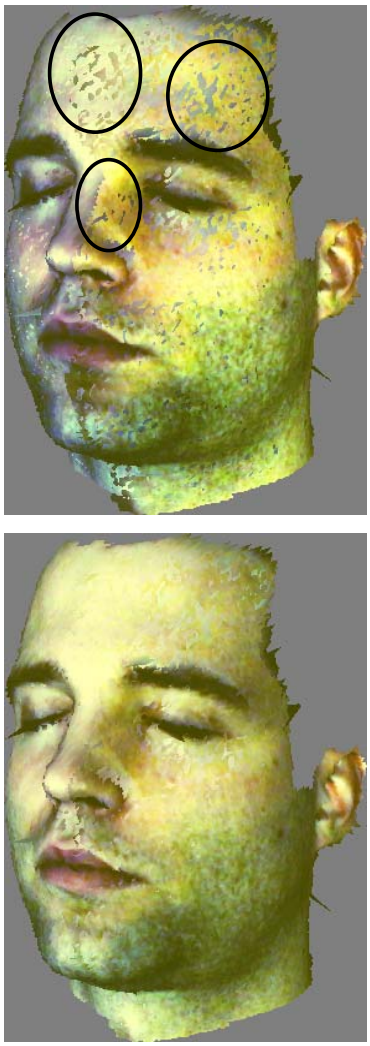

Figure 1: A texture correction operation. The histograms were equalized to increase contrast. *Left*. Before: Two juxtaposed models with their textures shown at the same time, the speckles in the ellipses are due to interlacing of the meshes. We can see that the speckles are not of the same color as the surrounding region. *Right*. After: The same two juxtaposed models with their textures corrected. The speckles are much less apparent.

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

original texture has a green color bias relatively to the other one. We will now see several approaches to texture correction in order to understand their limitations, and the usefulness of our method.

In one of the earliest work on the subject, Milgram [13] has shown how the difference between two textures can be distributed in the neighborhood of the seam. The input is two rectangle images that overlap. Let us see how it works. First, a "zeroth-order" histogram adjustment is done: the colors of the images are modi£ed by adding a constant value to each pixel so that overlapping regions have the same intensity averages (R, G and B processed separately). In the rest of the algorithm, each row will be processed separately, and only a neighborhood of size $2N$ of the seam will be modi£ed. Second, an algorithm determines on each row where the seam will be. The seam is a point between two pixels in the overlapping region that speci£es which image should be used: on the left of it we use the left image and on the right of it, we use the right image. On each side of the seam, a linear ramp of length N pixels (starting at 0 and increasing in absolute value toward the seam) is added to the intensity values so they meet at the average of the two pixels nearest to the seam. There are some problems with this approach. The fact that it depends on the difference of two pixels makes it vulnerable to the noise they contain, and it creates horizontal lines of slightly different intensity which are readily visible [4]. Also it works only for rectangular images of ¤at objects where the overlapping region is rectangular. In Milgram [14], some improvements are described related to the selection of the seam point and histogram equalization.

Rocchini et al. [20] developed an algorithm for partitioning the vertices of a triangle mesh depending on which texture each vertex gets its color from. That partitioning creates different regions and frontiers between them. The £rst step is to do an initial partitioning according to a simple criteria. The partitioning is re£ned with a greedy algorithm to reduce the number of frontier triangles, which are triangles that have vertices assigned to more than one texture. In other words, they do all reassignments of a node to a texture that, if considered alone, would reduce the number of frontier triangles in the current assignment. The new assignment becomes the current one and it continues like this until no further reduction is possible. To calculate the merged texture, interpolation is done inside all frontier triangles using areal coordinates, which are barycentric coordinates normalized so that their sum is 1. Barycentric coordinates of a point inside a triangle are the weights one must put on each of the three vertices so that their center of gravity (the barycenter) is on that point. Consequently, barycentric coordinates are unique up to scale factor.

In contrast, Pulli et al. [17] do not assign a single texture to each vertex. Instead, they do a weighted averaging over all the overlapping regions. The weights are calculated by multiplying several factors: proximity to texture border, cosine of the angle of the view ray with the normal, three weights for three views calculated with the direction of the virtual view. The calculation of these coordinates implies several steps. First, a Delaunay triangulation on the sphere of the real view directions is calculated. Then, the spherical triangle in which the virtual view direction lies is determined, and this gives three points, the endpoints of the directions forming this triangle. These points form a plane triangle. The weights are the areal coordinates in this triangle of the intersection of the virtual view direction with this triangle.

Levoy et al. [10] also use a weighted average where the weights are the con£dence in each pixel value. That con£dence takes into account several factors: obliquity of the surface with respect to the light, projected area of the surface with respect to the camera, proximity to the mirror direction (to reduce highlights), proximity to a silhouette edge with respect to the camera, proximity to a silhouette edge with respect to the light, and £nally proximity to the edge of the color image. Also to prevent rapid changes in con£dence from triggering sudden switches from one color image to another, the con£dences are smoothed among neighbors on the mesh. To remain conservative the con£dence is never increased, only decreased.

For weighted averaging, there are some recurring problems. If the transition width (of the blending function) is too large, a slight misalignment may produce a *double exposure effect*, where some features appear twice, whereas if it is too small the transition will be made in a few pixels thereby often creating a seam.

Burt and Adelson [3] successfully address these limitations with *multiresolution image splining*. They consider the case where the blending function $H(x)$ has the shape of a sigmoid, the £rst image has a weight of $1 - H(x)$ and the second $H(x)$. The transition width is a £xed parameter of that function specifying how long (in pixels) it takes to go from 0 to 1. They argue that the ideal transition width will vary for each spatial frequency in the images. So they use a pyramid of band-passed images that are blend one level at a time and then recombined.

But the main problem, from the viewpoint of realism, with the blending approaches without global adjustment, is that the blending only affects the overlapping region (however, in the case of multiresolution image splining [3], this might not be as severe). The seam is actually made invisible but the regions without overlap may have inconsistent colors, for instance one texture may be more red or green than the other. This is because we allow different cameras and different illuminations. Even for the approaches using global adjustment, the correction will be the same over all the texture, although each part of the object (because of the

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

varying normal) is not affected by the same illumination.

The naive way to achieve global adjustment would be to "recover the *real color* of the object", that is the re¤ectance properties. In practice, this is hard to realize. Sato et al. [21] have developed an approach where they compute re¤ectance using color and range images of an object. One major disadvantage of that technique is that it needs a lot of images and range images. In the example given in their paper, they use 120 color images from different angles and 12 range images. It is therefore dif£cult to use in certain cases, especially when scanning humans or animals.

Another way to correct textures is to recover the illumination and then relight the textures. However this problem is ill-posed so that more constraints are needed. For example, Marshner [12] calculates *inverse lighting* (a particular version of *inverse rendering*), which could also be called "solving for the illumination". The inputs are the scene description (3D model and re¤ectance) and a query image. First they divide the in£nitely far sphere of light into several pieces called *basis lights*. Then they render the scene under each basis light. The results are the *basis images*, which they combine linearly to get the query image. The coef£cients of the linear combination (which are also the intensity of the basis lights) are found using the Generalized Singular Value Decomposition and some plausibility constraints. A modi£ed version of that algorithm could be used to compute the *relative illumination*, which we will talk about later.

Ramamoorthi and Hanrahan [19, 18] developped a more general technique of inverse rendering. They developped independently and used the same basic ideas as Basri and Jacobs [1] (subspace of dimension nine in spherical harmonics space). Their technique allows to recover the illumination and the BRDF for an object, given that the object has the same isotropic BRDF over all its surface.

The following paragraph is taken from Gumustekin [8] and explains some approaches to image compositing, which is related to texture correction and is a step in image mosaicking (the steps in order are geometric correction, image registration, and image compositing):

> Finding the best separation border between overlapping images [14] has the potential to eliminate remaining geometric distortions. Such a border is likely to traverse around moving objects avoiding double exposure [5, 7]. The uneven exposure problem can be solved by histogram equalization [7, 11], by iteratively distributing the edge effect on the border to a large area [16], or by a smooth blending function [3].

Our approach modi£es not only the overlapping region but also the rest of the textures in order to get a realistic texture, that is, as it would appear under a new illumination. That means consistent colors across textures, so that, for example, the color of the face seems the same all over. It is sometimes surprising how different the colors look when taken with different cameras and illuminations. Yet, in each image seen separately, the colors seem completely natural. The kernel of the method consists in the calculation of the "relative illumination", which is related to the *lightsphere* concept of Blicher and Roy [2].

The major advantages of our method are that it makes few hypotheses, gives realistic results, and does not involve any calibration or solving for the illumination. It is much more realistic than weighted averaging, which is the most commonly used technique. It is much easier to use than the re¤ectance estimation techniques. Furthermore, the textures may have been taken under different illuminations and with different cameras. The main constraint is that our method requires similar lighting conditions for points with similar normals. This implies objects that are convex or close to convex, such as faces.

2 Hypotheses

Here are some hypotheses we did in order to simplify our model.

Parallel rays We suppose that all the light sources are punctual and in£nitely far away, thereby making all the rays coming from a source parallel near the object of interest. When taken as a whole, these light sources are what we call the *illumination sphere*.

Smoothness In this article, we consider only objects with a relatively smooth surface, or in other words, objects having no normal discontinuity (except at the edge of the mesh). Indeed, because we use triangle meshes, the normals do not vary in a continuous way. Rather, this hypothesis disallows rapid variations in the normal. Also, it means that most of the time the normals vary slowly. This implies that, for each point, one can £nd an in£nitesimal surface patch around it and a normal at this point, and that the patch and the normal are perpendicular.

View factor For each point on the surface of the object, we de£ne the *view factor* as the fraction of the illumination sphere that is visible from the in£nitesimal surface patch around this point. Assuming that a surface patch can only receive light from the hemisphere centered on its normal, the view factor should be at most $\frac{1}{2}$.

Convexity Also, we only consider objects which are convex or almost convex (such as faces). In other words, only objects where most points have a view factor near $\frac{1}{2}$. That is not true in general for objects, since for some points in concavities, it is signi£cantly lower. Combined with the previous hypotheses, it implies that all the points with similar normals have similar illumination.

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

BRDF There are two hypotheses concerning the BRDF: it is isotropic and similar for the points having similar normals. The isotropy of a BRDF means that if we turn the object around the normal at a point on the surface, the intensity of that point will not change (if the point is hit by the same rays of light). We also assume there is no subsurface scattering (Jensen [9]), this implies that illumination at a point does not in¤uence its neighbors. An important point is that the BRDF doesn't need to be lambertian. A BRDF is lambertian when the intensity of a point does not change with the position of the observer.

Orthographic projection We suppose that each image is taken from far enough to approximate all the rays as parallel. Therefore, the BRDF for a point depends only on the surface normal at that point, the incoming light direction, and linearly on some albedo that varies from point to point on the surface.

Reliable normals For the tens of faces we scanned, the geometry seemed very realistic and accurate. So we assume that we can rely on the geometries to deduce the normals.

Texture overlap We assume that, in each pair of textures we want to adjust, there is an overlapping region between them. This region is needed to provide estimates of the "relative illumination", which will be explained in the next section.

3 The method

We present a method to adjust the textures two-by-two, however the method could be modi£ed to adjust n textures all at the same time, thereby making a global adjustment. In order to adjust n textures using our method, simply take two overlapping textures, adjust them, merge them into one, and continue to do that with other textures until there is only one left. We do not specify the algorithm used to merge two textures into one, because any algorithm should do the job, the textures being almost equal after the correction.

Conceptually, the elementary procedure in our method re-renders both textures, texture-1 and texture-2, as they would appear if they had been taken under another illumination.

3.1 Notation

Some simpli£cations have been done for clarity. Instead of showing the equations for each channel (R, G and B), we show them for only one channel, but they are the same for each.

Also we adjust textures, but we do not mention indices of pixels in these textures. The reason is that we can associate each point of the surface to its corresponding pixel in the texture and vice versa. This implies that each pixel used in the texture corresponds to a region on the surface of the object.

Let:

- G be the Gaussian sphere (the set of unit vectors which we use to represent directions).
- S be the surface of the object.
- $A(p) : S \rightarrow \mathbb{R}^+$ be the albedo (intrinsic re¤ectance) function.
- $N(p) : S \rightarrow G$ be the Gauss map of the surface, i.e., the function that maps each point p to its normal.
- $L_i(g) : G \rightarrow \mathbb{R}^+$ be an illumination sphere (it can represent any number of point light sources). $L_1(g)$ and $L_2(g)$ are those present when the corresponding image was acquired (this is the illumination map of Miller and Hoffman [15]).
- $I_i(p) : S \rightarrow \mathbb{R}^+$ be the observed image intensity for texture-i at a given point p.
- $d_i \in G$ be the orientation of the orthographic projection of image-i.
- $BRDF(v_1, v_2, n) : G^3 \rightarrow \mathbb{R}^+$ be the BRDF function for a certain normal n, where v_1 is the viewpoint direction, v_2 is the incident illumination direction.
- $V(p, g) : S \times G \rightarrow \mathbb{R}^+$ be the visibility function of the illumination sphere from a given point p in a given direction g (0 = not visible, 1 = visible).

3.2 The model

In this notation, and considering an in£nitesimal solid angle $d\mu$ of G with position g, the general model of surface re¤ectance with attached shadows can be written as (for $i = 1, 2$):

$$I_i(p, d_i) = A(p) \int\int_G V(p, g) L_i(g) BRDF(d_i, g, N(p)) d\mu$$

By applying the hypothesis that the view factor is near $\frac{1}{2}$ everywhere, we can transform it into:

$$I_i(p, d_i) = A(p) \int\int_{G_N(N(p))} L_i(g) BRDF(d_i, g, N(p)) d\mu$$

where $G_N(n) = \{g \in G | g \cdot n \geq 0\}$. Notice that this equation is of the form:

$$I_i(p, d_i) = A(p) B_i(N(p), d_i)$$

with $B_i : G \rightarrow \mathbb{R}^+$. B_i can be thought of as a brightness function for camera-i that depends on L_i and $N(p)$, which captures the interaction of the lighting distribution with the normal.

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

3.3 Choice of a new lighting

What we want is to re-render both textures as they would look under a different illumination sphere L', that gives us:

$$\begin{aligned} I'_i(p,d_i) &= A(p)B'(N(p),d_i) \\ &= \frac{B'(N(p),d_i)}{B_i(N(p),d_i)} I_i(p,d_i) \\ &= C_i(N(p),d_i) I_i(p,d_i) \end{aligned}$$

where I'_i are the re-rendered texture, C_i is the lightsphere (relative illumination), and B' is the brightness function corresponding to L'. B' is not an arbitrary function, it must be the result of a double integral as mentioned in the previous subsection. Using B_1 and B_2, a valid value would be an interpolated value or, more generally, a linear combination with non-negative coeff£cients. In this case, we conclude:

$$\begin{aligned} C_i(N(p),d_i) &= \frac{B'(N(p),d_i)}{B_i(N(p),d_i)} \\ &= \frac{k_1 B_i(N(p),d_i) + k_2 B_{3-i}(N(p),d_{3-i})}{B_i(N(p),d_i)} \\ &= \frac{k_1 I_i(p,d_i) + k_2 I_{3-i}(p,d_{3-i})}{I_i(p,d_i)} \end{aligned}$$

We can generate several interesting cases for B' by varying k_1 and k_2 : $B' = B_i$, $B' = (B_i + B_{3-i})/2$, and $B' = B_{3-i}$. In the £rst and the last cases, one texture is modi£ed to use the illumination of the other texture, which is not modi£ed.

We can verify that B' is indeed a valid brightness function:

$$\begin{aligned} &B'(N(p),d_i) \\ &= k_1 \int\int_{G_N(N(p))} L_i(g) BRDF(d_i,g,N(p)) d\mu + \\ &\quad k_2 \int\int_{G_N(N(p))} L_{3-i}(g) BRDF(d_{3-i},g,N(p)) d\mu \\ &= \int\int_{G_N(N(p))} [k_1 L_i(g) + k_2 L_{3-i}(g) \tfrac{BRDF(d_{3-i},g,N(p))}{BRDF(d_i,g,N(p))}] \times \\ &\quad BRDF(d_i,g,N(p)) d\mu \\ &= \int\int_{G_N(N(p))} L'(g) BRDF(d_i,g,N(p)) d\mu \end{aligned}$$

3.4 Relative illumination estimation and extrapolation

We can get one estimate of $C_i(N(p),d_i)$ for each p in the overlapping region. We could therefore collect all the available estimates, however a more robust way is to discretize the domain of C_i, which is G, and estimate it by voting in the bins. The result of the vote for a bin is the average of the votes that fell into it. To have more accurate results, the points for which either texture had a value too low are excluded (we chose empirically 5 as a threshold, and the intensity goes from 0 to 255). Also we did not take into account bins that have too few votes (in our case: fewer than about 0.1% of the votes).

In the general case, the discretization for voting could be done by partitioning G in regions of similar area and shape. However, for faces, the discretization we used is simpler: it consists in dividing evenly along the x and y components of the normal, each in the range $[-1,1]$ to make a M by M array of bins. In our case, $M = 20$ was enough, see later for a discussion on this parameter. This works for the faces we scanned, because the face generally do not have normals with a negative z, and when they do, these normals are ignored. However, the discretization for more complicated situations would have to be more elaborate.

At this point, the method has some problems associated with the discretization, the voting, and the extension of the vote, which will be described later. The processing that is described below applies to the array representing $C_i(n)$.

First, there is noise in the pixels of the acquired images and therefore, there will be noise in the estimate of $C_i(n)$. This should be partially compensated by the averaging of the votes in each bin.

Second, in order to correct the textures, we need to have $C_i(n)$ de£ned over the range of the normals of the points of that texture. However, in general, it will not be the case, unless the texture to correct is entirely covered by the other, which would mean there is redundancy in the views taken.

A solution to both problems is to £lter the obtained $C_i(n)$ by a weighted average of the de£ned values in the neighborhood. The result of this operation is to smooth the de£ned values and to extrapolate the unde£ned ones. The weights are given by a gaussian kernel $(2M-1)$ by $(2M-1)$ bins (with a variance of $\frac{M}{40}$ bin widths) multiplied by the number of votes for a certain bin. For a bin more than $\frac{M}{10}$ bins away from any determined one (including itself), the effect is comparable to a nearest-neighbor extrapolation, because of the radial symmetry of the gaussian and because it drops fast.

Third, what sometimes happens is that, because of the pseudo-nearest neighbor extrapolation, there are rapid ¤uctuations of the values, see for example Figure 2. To counter that, the values which were initially unde£ned are replaced by the result smoothed by weighted averaging (with a gaussian kernel $(\frac{M}{2}+1)$ by $(\frac{M}{2}+1)$ bins with a standard deviation of $\frac{M}{10}$ bin widths).

Fourth, because of the discretization, we may see discontinuities in the result when the normal changes from one bin to another. To reduce that, once we have processed $C_i(n)$ as mentioned, we can calculate $I'_i(p)$ using bilinear interpolation into $C_i(N(p),d_i)$.

As an example, suppose we have a model of a face from which three images have been acquired from $-45°$ (left), $0°$ (front), and $45°$ (right). We slightly modi£ed the procedure for n textures here in order to have symmetry between left and right. Let us assume there is no overlap between the left and the right textures. We receive the merged geometries

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

Figure 2: Intermediate steps in the processing of a lightsphere. Black represents unde£ned values and other intensities represent ratios ($intensity = 100 * ratio$). The red circle contains the valid values of the lightsphere. *Left*, the result of the vote. *Middle*, the lightsphere after weighted averaging. We notice the discontinuity caused by pseudo nearest neighbor extrapolation (near the lower right corner). *Right*, the values initially unde£ned (the black pixels in the left image) are smoothed.

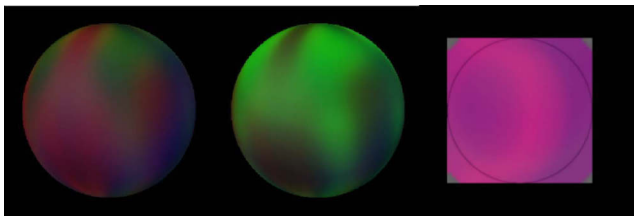

Figure 3: A synthetic test of our algorithm (using three colored lights). The object is a sphere. Image-1 is on the left. Image-2 in the middle. They overlap completely. Image-2 is re-rendered as if it was under the same illumination as when image-1 was taken ($k_1 = 0, k_2 = 1$). The result is not shown because it is equal ± 1 intensity level to image-1. On the right, we see the lightsphere (inside the circle) where each pixel is the ratio of the corresponding bin multiplied by 100.

from a previous step. Second, we do some texture corrections. We correct the left and the front textures together, to get a new texture called LF. Then the right and front textures, to get FR. Finally, the textures LF and FR to get the £nal result LFR. The coef£cients k_i to calculate LF and FR are $\frac{1}{3}$ for the front texture and $\frac{2}{3}$ for the other textures. The coef£cients k_i for creating LFR are both $\frac{1}{2}$. Because of the interpolation, it results in an illumination which is more uniform. Because of the coef£cients, the contributions of each texture in the £nal result is the same.

4 Results and discussion

First, we tried our algorithm on synthetic data, a sphere pictured twice from the same viewpoint, but under two different illuminations. The results, which were perfect except for the rounding errors, can be seen in Figure 3.

We also tried with real data. We used 22 faces to test our algorithm. They were acquired with three structured-light 3D scanners from InSpeck Inc. To do a multi-scanner acquisition, we £rst do spatial calibration using a dodecahedron. This allows to £nd the position of each scanner relative to the others. Second, the scanners acquire data one after the other. Each acquisition produces one 3D model containing more than ten thousands points, each precise to 0.5mm and a 24-bit RGB color texture mapped on the model. Because of the spatial calibration, the models can be easily aligned together and each texture can be stitched onto their mesh.

Some results are shown in Figure 1. In Figure 4, we can see the errors on the corrected texture. Although there are errors (caused by violations of the hypotheses), the results still seem very natural. In Figure 5, we can see histograms of the differences of the pixels in the overlapping region. In Figure 2, we see the result of the processing of the lightsphere (weighted averaging plus smoothing of the extrapolated unde£ned values). We can see the effect of interpolating between B_1 and B_2 in Figure 6.

We used a value of $M = 20$ for the discretization width in bins and noticed that it gives good results for the faces. However, our technique could also be used for surfaces with different BRDF (for example a shiny BRDF). We would then need to raise M in order to get a good approximation for C_i.

4.1 Strengths and weeknesses

Let us now examine the strengths and the weaknesses of our approach. Before deciding to use a correction factor dependent on the normal, we tried several other models. We tried the additive model ($I'_2(p) = I_2(p) + k$), the multiplicative model ($I'_2(p) = kI_2(p)$), the exponential model ($I'_2(p) = I_2(p)^k$). All of them gave bad results, where we could instantly see the seam. Blending the textures would not solve our problem even if it could give acceptable results in the overlapping region: the parts with no overlap would have different colors. So clearly, this is a strength of our approach, the seam is almost invisible and the color is consistent.

We could overcome the convexity limitation by allowing only certain points to vote: only the points within a certain range of view factor near $\frac{1}{2}$. We would also have to devise a way to calculate a modi£ed correction factor for the point where the view factor is signi£cantly less than $\frac{1}{2}$.

The constraint requiring a common BRDF for all points with the same normal was a good approximation for all the faces, and allows good recovery of specular surfaces (assuming the BRDF is equal at points with the same normal).

Even if the different images were taken with different cameras, which each have their own bias, it does not matter

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

Figure 4: The absolute difference between the corrected textures for the red channel (it is similar for the two others channels). The completely white regions on the left and on the right are regions where there is only one texture. Black represents 0 and white, 10 or more.

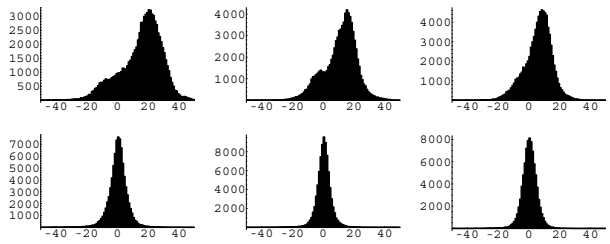

Figure 5: Histograms of the differences between pixels in the overlapping region for a face. In the top row before the correction (red, green, and blue), and in the bottom row after the correction (red, green, and blue). We can see that the correction centers the error (it is nearer to 0) and reduces it signi£cantly.

Figure 6: Interpolating the illumination. The textures shown are texture-1 on the left, and texture-2 on the right. In the usual order: the original image, the results for $B' = B_2$, $B' = (B_1 + B_2)/2$, and $B' = B_1$. The dark spots on the right of each face are not errors, simply the third texture (not modi£ed here).

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE

because that is taken into account in the "relative illumination". If everything seen by one camera is a factor off what the other one saw, it is already included into $C_i(n)$. This can remove the effect of Automatic Gain Control.

4.2 Conclusion

There are some questions that we have not considered. We have not investigated the case where there are one or more cycles in the graph de£ned by the relation "texture a and texture b overlap". For example, it would be the case when one has acquired images all around an object. A simpler case: what if the intersection of three textures were not empty? How would we treat these cases, could we get more information?

Also we have not taken into account the fact that some points have a view factor much less than $\frac{1}{2}$. Some possibilities are to reduce their in¤uence during the vote, exclude them from voting, or compute their correction factor differently.

References

[1] R. Basri and D. Jacobs. Lambertian re¤ectance and linear subspaces. Technical report, NEC Research Institute, 2000.

[2] Albert Peter Blicher and Sébastien Roy. Fast lighting/rendering solution for matching a 2d image to a database of 3d models: Lightsphere. *IEICE Transactions on Information and Systems*, E84-D(12):1722–1727, December 2001.

[3] Peter J. Burt and Edward H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, November 1983.

[4] Chia-Yen Chen. Image stitching - comparisons and new techniques. Technical Report CITR-TR-30, Tamaki Campus, University of Auckland, October 1998.

[5] J. Davis. Mosaics of scenes with moving objects. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pages 354–360, 1998.

[6] Paul E. Debevec, George Borshukov, and Yizhou Yu. Ef£cient view-dependent image-based rendering with projective texture-mapping. In *Automated Texture Registration and Stitching for Real World Models*, pages 105–116, 1998.

[7] S. Gumustekin and R.W. Hall. Mosaic image generation on a ¤attened gaussian sphere. In *Proc. of IEEE Workshop on Applications of Computer Vision*, pages 50–55, 1996.

[8] Sevket Gumustekin. An introduction to image mosaicing. http://likya.iyte.edu.tr/eee/sevgum/research/ mosaicing99/, July 1999.

[9] Henrik Wann Jensen, Steve Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH'2001*, pages 511–518, 2001.

[10] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. In *Proc. of SIGGRAPH 2000*, July 2000.

[11] J. Lim. *Two Dimensional Signal and Image Processing*. Prentice Hall, 1990.

[12] S.R. Marshner. *Inverse rendering for computer graphics*. PhD thesis, Cornell Univ., 1998.

[13] D. L. Milgram. Computer methods for creating photomosaics. *IEEE Trans. on Computers*, C-24:1113–1119, November 1975.

[14] D. L. Milgram. Adaptive techniques for photomosaicking. *IEEE Trans. on Computers*, C-26:1175–1180, November 1977.

[15] Gene S. Miller and C. Robert Hoffman. Illumination and re¤ection maps: Simulated objects in simulated and real environments. Course Notes for Advanced Computer Graphics Animation, SIGGRAPH 84.

[16] S. Peleg. Elimination of seams from photomosaics. *Computer Graphics and Image Processing*, 16:90–94, May 1981.

[17] K. Pulli, H. Abi-Rached, T. Duchamp, L. Shapiro, and W. Stuetzle. Acquisition and visualization of colored 3d objects. In *Proc. of Int. Conf. on Pattern Recognition*, pages 11–15, 1998.

[18] Ravi Ramamoorthi and Pat Hanrahan. An ef£cient representation for irradiance environment maps. In *SIGGRAPH 2001*, 2001.

[19] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *SIGGRAPH 2001*, 2001.

[20] Claudio Rocchini, Paolo Cignomi, Claudio Montani, and Roberto Scopigno. Multiple textures stitching and blending on 3d objects. In *Proc. of Eurographics Rendering Workshop 1999*, pages 173–180, June 1999.

[21] Yoichi Sato, M. Wheeler, and Katsushi Ikeuchi. Object shape and re¤ectance modeling from observation. In *Proc. of ACM SIGGRAPH 97, in Computer Graphics Proceedings, Annual Conference Series 1997, ACM SIGGRAPH*, pages 379–387, August 1997.

Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)
1063-6919/03 $17.00 © 2003 IEEE