Illumination Problems in Computer Augmented Reality

Alain FOURNIER

University of British Columbia
Department of Computer Science, UBC, Vancouver, V6T 1Z4
fournier@cs.ubc.ca

Résumé: The ability to merge a real video image (RVI) with a computer-generated image (CGI) enhances the usefulness of both. To go beyond "cut and paste" and chroma-keying, and merge the two images successfully, one must solve the problems of common viewing parameters, common visibility and common illumination. The result can be dubbed *Computer Augmented Reality* (CAR). The solution needs contributions from both computer graphics and computer vision. The problems of common illumination are especially challenging, because they test our understanding and practice of shadow and global illumination computation. In this paper we will describe and illustrate work in our laboratory where the emphasis is on extracting illumination information from real images and computing the common illumination between the real and the computer generated scene.

1. Introduction

Advances in computer hardware and results in computer graphics made it possible to build graphics workstations which can produce real-time images of good resolution (about 1Kx1K or greater) and moderate complexity (several thousands polygons). In addition, specialized hardware allows users to have real-time video windows on the same workstations. While computer graphics has made great strides towards increased realism in modelling shape and light effects, neither the models nor the hardware is close to the point where it can give real-time realistic (that is good enough to "fool" us) images of our usual environment. It is also clear that in many, if not most, applications there is a real, existing environment within which the modelled objects should eventually be inserted.

The usefulness of the two sources of information, *real video images* (RVI) and *computer generated images* (CGI) can only be enhanced by the ability to merge them freely in real time on the workstation screen. By merging we mean ideally in a way that appears "seamless", where one cannot distinguish between the "real" part and the "synthetized" part. We call the ensemble of techniques to reach that goal *Computer Augmented Reality.* It is different from so-called virtual reality in that computer generated objects and effects are only part of the viewed scene. Of course the result can be viewed in a typical virtual reality display system, but it is not the only intended environment. We stress also that real-time is only a goal, and that for now we are quite content to investigate techniques that work, regardless of their current speed.

The main issues can be divided into *geometric* issues and *illumination* issues. The geometric issues in turn divide into *viewing parameters* and *visibility* problems. The viewing problem is to establish common viewing parameters between the RVI and the CGI. The visibility problem consists in resolving mutual priority while compositing the two images.

The illumination problem is to compute illumination (both *local* and *global*) of RVI from computer generated light sources, and of CGI from real light sources. The local illumination problem assume the local illumination conditions are known, and consists in computing the light reflected in the direction of the eye; the global illumination problem is to compute the local conditions for every visible point, given the scene description. Secondary illumination problems, such as shadows, reflections and transparencies, fall somewhere between these two categories. In local illumination, many models have been developed for computer graphics, giving reasonably realistic images. It is only recently that global illumination problems have been seriously investigated, and *radiosity*-based methods are the most popular ways used to solve these. The main new problems here are to identify the positions and characteristics of the lights in the RVI to illuminate the CGI, and to acquire enough knowledge about the geometry of the picture (*eg* getting the "shape from shading") to apply an illumination model and shade it according to the CGI lights. While the local illumination of CGI is rather straightforward after this, the global illumination of CGI is still rather costly. Computing global illumination on the RVI from computer generated lights is in a sense impossible, since it can be affected by surfaces not seen in the real images. One of the challenges is to develop heuristics to "infer" the hidden reflectors from the observed illumination of the real images.

Shadows present a particularly interesting challenge. We can note that in our context, the shadow problem, involving lights, objects and potential shadowing objects from two different origins, has eight "flavours", one of them "free" (when all are real), and some others well controlled (*eg* when all are computer generated). There is also the interesting problem of "reconstructing" the colour of a real surface when a computer generated light forces us to remove a real shadow.

2. Conditions for a Complete Solution

It is pretty obvious that for a complete and accurate solution (at least accurate within common standards in computer graphics) one should be able to extract from the RVI a rather complete model of the objects and light sources, including the illumination models, and then render all of them as CGI. Since this involves solving a reconstructionist version of computer vision and then some, we will not wait for such a solution. The rest of this paper will describe several research efforts aimed at various elements of the solution. Sometimes the route to it will seem somewhat circuitous, but we are also trying to start from relatively solid positions. The topics addressed are compositing with shadows, estimating illuminant characteristics from images and highlight detection, light direction from shading and reshading, common global illumination and surface charateristics from surface colour.

3. Shadows in Composition of Depth Images

In our context shadows can usefully be divided into *local shadows* and *global shadows*. Global shadows are those where the objects casting the shadow is rather large and at a distance from the object on which the shadow is cast of the order of its size or more. Local shadows are the rest, that is where the objects and distances involved are small. At the limit, local shadows involves self-shadowing and gets included in a local illumination model. Global shadows are dealt with by the global illumination computations, as described in Section 6. Our work on local shadowing has been limited so far to a particular aspect: given two rendered images, each with a depth map, how can they be compositied together such that their mutual shadowing is taken into account. We assume that the two scenes have a common known illumination. This is easy to achieve if the two images are computer generated, and the image files contains a depth map, in addition to the placement and characteristics of the lights. We do not claim that we can extract accurately the same information from RVI, but the work described in the other sections, on light direction from shading and on highlight detection are steps in that direction.

The merging of two CGIs while resolving common visibility is usually done with a variant of classic compositing techniques [port84] using depth buffers, best explained by Duff [duff85]. To address the visibility problems when rendering of 3D scenes are to be correctly composited, Duff used the Z-buffer for visibility determination Duff's method requires that for each pixel in a rendered image, a corresponding depth value is stored. This depth value represents (in some form) the distance the pixel is from the eye plane. When compositing images, the depth information of corresponding pixels in each image is compared, and the pixel with the smallest depth is displayed in the resultant image. This allows parts of animations to be composited easily, with some increase in storage. While this method provides an easy way to composite images with common visibility, this does not address common illumination, in particular the problem of common shadows. We will here summarize our work on the *reshadowing* of images when compositing.

To obtain a manageable problem, we will place restrictions on the situations we will consider. The fact that the knowledge of objects in the scene is limited to what is actually seen gives rise to the first restriction. If two images are to be reshadowed such that shadows from one scene are to fall on the other, but *not* vice versa, then the scene that is causing the shadows must be made up only of non-occluding objects, but the other may be of arbitrary complexity. If the reshadowing is to be mutual, then both images must contain only non-occluding objects [1] The second restriction on images is that all must have the

[1] Consider the case where a ball moves above a plane such that the ball never occludes the plane, but casts a shadow. If the ball and plane were two separate images that were to be

same viewing parameters (such as image dimensions, eye position, eye roll angle, and such). The last restriction requires that all lighting parameters be the same (that is, number of lights, position, colouration etc.).

Since the initial restriction placed on the images has two cases, the outline of the solution is slightly different. In the first case, where only one image is used to reshadow the other, it is first necessary to get some notion of the original three-dimensional shape of the objects in the restricted (non-occluding) scene. To do this, world coordinates for each visible pixel of every object are retrieved using the stored depth data and existing ray-tracing techniques. These world coordinates are used to create a simple triangulated description of the visible surface of each object. Once this surface description is created it is passed to a modified ray-tracing renderer. If there are no problems with the reconstruction, a shadow map is created by the ray-tracing software. This shadow map is an image with the same dimensions as those being reshadowed but with all pixels having a full white value. The world coordinates of each visible pixel in the other (arbitrarily complex) scene are retrieved by the same method as used in the non-occluding scene, but without reconstruction. From each of these coordinates, the ray-tracer sends a ray to each light in the scene (since the assumption is that light positions are known). If any ray intersects a triangle in the reconstruction of the non-occluding image, a black pixel is placed in the shadow map at the same location as the pixel (from the arbitrarily complex image) whose world coordinate was the origin of the ray. The shadow map is filtered to smooth sharp intensity transitions, and *overlaid* on the arbitrarily complex image. This overlaying causes pixels in the image to have their intensity reduced in proportion to the amount of black present in the corresponding pixel of the shadow map. This is the actual *reshadowing* step, and once performed, the two images are composited using the modified depth composition method to give the final result.

The second case is solved in a similar manner, but the process is repeated for both images. Thus, the first image is initially treated as the complex image (even though it is not), and reshadowed. The images then reverse roles, the second image is reshadowed, and the images are composited.

Both methods of solution are discussed in detail in Russ Krywolt's thesis [kryw93], and the results are presented. It should be noted that for a satisfactory result in general, real shadows should be detected to avoid darkening areas which are already blocked from the light by objects in their own image. This problem has not been yet directly addressed in our work, but it is a real and difficult one.

The applications of these methods to the context of CAR given a satisfactory depth map is straightforward, with the same possibilities and drawbacks. If only reshadowing is desired the viewing parameters need not be retrieved, but the depths of the computer generated image and real image must be matched in some fashion so as to prevent errors of scale when compositing. This might be achieved by using an interactive technique where the user dynamically scales the depths of one image against the other so as to get a satisfactory correlation

reshadowed, only the image of the plane need be reshadowed.

between them. Problems with perspective and aspect ratio could occur as well, in which case the viewing parameters of the real image would have to be retrieved. In this case, and if the real image was to be used in other CAR related operations, some way of finding the shape of an object, and not just its visible surface, should be found. This would facilitate the recovery of the viewing parameters, as well as be of assistance when performing global illumination tasks. Once the two images have been altered enough so that reshadowing can proceed, the procedures used for computer generated images can be employed to reshadow the real image. If the computer generated image is to be reshadowed as well, some way of finding the light position in the real image must be found.

There is still much interesting work in that area. One important question (actually relevant to most of the topics here) is finding a good image segmentation method, so that arbitrarily complex depth images can be composited and reshadowed. Edge detection techniques may prove useful in doing this, as might modified surface fitting and discontinuity finding algorithms, such as [terz83], [marr84], [gamb87], and [terz88], which have been successfully applied to restricted domain problems. After this is done, a better reconstruction method should be found. Initially a better visible surface finding algorithm should be created, where a smoother surface is constructed from the points that will be at most as expensive to intersect in the shadow map calculations as the current triangular mesh. Some heuristic method should also be used to guess at the best way to reconstruct the complete object, using human input to verify the guesses, unless it becomes possible to easily retrieve the entire object description. Current shape from shading methods, as well as reconstruction from image sequences, coupled with the depth and colour information, will hopefully provide a basis for doing this.

Once it is possible to reshadow arbitrarily complex images, a method of dealing correctly with multiple, extended, and area light sources should be found. This entails not only the creation of shadows cast from such light sources, but also the removal of lessening of those shadows that would be illuminated by lights if the scenes were composited. This leads into the more general reshading problem of CAR, where methods from there could be borrowed to create new highlights on objects illuminated by a light from a composited image, as well as removing highlights from reshadowed areas of images. At the same time, ways should be found of overcoming problems encountered when compositing images with separate viewing parameters.

4. Estimating Illuminant Characteristics and Highlight Detection

Techniques available for estimating the illuminant colour of an RVI can be divided into two categories: those that assume the surfaces are perfectly diffuse (Lambertian), and those that assume the surfaces are made of dielectric materials, or the familiar computer graphics model surfaces which reflect as a mixture of diffuse and specular modes. The standard expression for each colour channel ($c \in \{r, g, b\}$) can be written as:

$$L_{pixel}(c) = k_a(c)L_{ia}(c)\pi + k_d(c)\int_\omega L(c)(\vec{N}\cdot\vec{L})\,d\omega + k_s(c)\int_\omega L(c)(\vec{N}\cdot\vec{H})^n\,d\omega. \quad (1)$$

where
k_a is the proportion of ambient reflection,
k_d is the proportion of diffuse reflection,
k_s is the proportion of specular reflection,
$\vec{N}$ is the surface normal,
$\vec{L}$ is the light direction,
$\vec{H}$ is the bisector direction between $\vec{N}$ and $\vec{L}$,
n is the roughness coefficient,
L is the source radiance,
L_{ia} is the scene ambient radiance,
ω is the solid angle formed by the light source.

Thus, techniques to estimate illuminant colour that exploit the characteristics of such materials are more useful. Furthermore such an estimate would allow us to classify pixels as to the degree of specular reflection they contain, and thus help us detect and remove highlights.

One such technique is the chromaticity convergence method of Lee [lee86], [lee88]. However, this technique has its shortcomings. For example, it fails when all the points in the CIE coordinates are collinear. It also fails when the input image is filtered or when surfaces in the input image are reflective or textured since the loci of the points in the CIE diagram are scattered such that it is very difficult to determine the straight lines formed between the illuminant chromaticity coordinate and each surface chromaticity coordinate.

It is easy to observe that in the CIE diagram, the pixel intensity of the points along the straight line between the surface chromaticity coordinate (x_s, y_s) and the illuminant chromaticity coordinate (x_i, y_i) increases in the direction of (x_i, y_i). This is due to the fact that at a given point on a dielectric opaque surface the specular intensity component is linearly added to the diffuse intensity component, and in general, the specular reflection exponent is relatively large (around 20 to 40) such that, spatially, the rate of change of the specular intensity component is faster than the diffuse one. Therefore, at and around the highlight region, the intensity of the pixels increases in the direction of the center of the highlight. Thus, the occurrence of intensity change together with the chromaticity shift among the pixels at certain image region can be use as a cue to the existence of a highlight region. We extended the chromaticity convergence method [guna91] by using the pixel intensity as the third dimension in the CIE chromaticity diagram. In this three dimensional CIE diagram, the loci of all the points converge towards the illuminant chromaticity coordinate. This chromaticity convergence method to be less sensitive to changes in surface reflectivity or surface texture. It is also be able to handle the cases where all the points are collinear in the chromaticity (x, y) plane.

This is turn leads to a simple method to remove locally the highlights, since by fitting a function to the boundary curve of the enveloped of the 3D cluster of points we can move any colour point of the highlight towards the colour it should have for a diffuse surface. This assume, unfortunately that some segmentation has already been done, and this is still the main step to apply the technique successfully in a totally automatic fashion.

5. Light Direction from Shading

Researchers in computer vision have worked extensively on problems of determining 3D shape from stereopsis, motion, shadows and texture gradients. There are also many reported works on determining the geometry of the light sources from similar information. Some methods for shape determination assume that the geometry of the light sources is known, while some methods for light sources assume that the local surface charateristics are known (see also Section 7 below).

Two principal techniques for deriving shape from shading are reported in the literature. One known as the *reflectance map* technique, assumes that the imaging geometry, i.e., illumination direction, surface reflectivity, and illuminant strength, is known *a priori*. This technique then views *shape from shading* as the solution of a problem of partial differential equations. In the other technique, mainly explored by Pentland, the imaging geometry is not needed; instead the surface points are considered to be *umbilical* points (points on a sphere, in practice). This is inspired by the way humans are able to determine shape from shading even in cases where the illumination direction, illuminant strength, and surface reflectivity are not known.

The technique has many drawbacks in practice, but we wanted to explore its possibilities in the context of CAR for two main reasons: first in many cases we cannot assume known the charateristics of the real lights, but we need them to shade the computer generated objects; second in our application the ultimate goal is to obtain a reshading that is believable and consistent, and making local decisions with little assumed known seems to be a good general strategy.

According to Pentland [pent82b], [pent82a], the direction of illumination is required to be known in order to obtain accurate three-dimensional surface shape from two-dimensional shading because changes in image intensity are primarily a function of changes in surface orientation *relative* to the illuminant. For example, small changes in surface orientation parallel to the illuminant direction can cause large changes in image intensity, whereas large changes in surface orientation which occur in a direction perpendicular to the direction of illuminant will not change the image intensity at all. Thus the illuminant direction must be known before one can determine what a particular change in image intensity implies about the changes in surface orientation.

The assumptions to be made for an effective algorithm are as follows. The first is that changes in surface curvature are isotropically distributed. This condition is true of images of convex objects bounded by a smooth occluding contour and is true on average over all scenes. Pentland deduced the light direction and surface orientation by using least square methods together with other qualitative analysis to solve the partial differential equations approximately. For shape from shading, Pentland used the assumption that surface points are umbilical points, which happens to be a strong enough assumption to yield a unique interpretation for the surface shape by usi ng local shading analysis. Pentland presented two local analysis methods: point-by-point and regional constraint. A follow-up paper by Lee and Rosenfeld [lee85] essentially used the same assumptions as Pentland, but use statistical methods to determine the light direction and then estimate shape orientation in the light source coordinate system. We implemented these two methods for our case, studied their properties and developed a slightly modified third way.

5.1. Pentland's Method

This assumes a simple model of image formation: there is a infinitely distant point source illuminant in direction L, a patch of surface (assumed to be Lamb ertian surface) with surface normal N, and a viewer in direction V (figure 1).

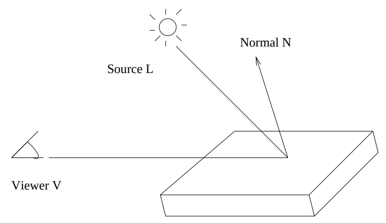

Figure 1: A simple model of image formation

The surface normal N, the viewer's direction V, and the illuminant direction L are all unit vectors in Cartesian three-space, and only two parameters are necessary to define of them. The parameters chosen here are the slant σ and the tilt τ. The direction of tilt of a vector is the direction of steepest descent, that is, the direction of image-plane component of the local surface normal. The tilt itself is the arctangent of the ratio of the x and y components of the unit vector. The slant of a vector is the slope in the direction of steepest descent. The slant itself is the arccosin e of the z component of a unit vector.

The basic model is summed up in the formula:

$$I = \rho f(N \cdot L)$$

where ρ is the fraction of incident light that is reflected and f is the amount of light(flux) per unit area arriving the surface.

If we stay within a relatively small, homogeneous area of the image, it is reasonable to assume that the albedo and illumination are constant:

$$dI = \rho f(dN \cdot L) \tag{2}$$

Thus the variation in image intensity dI depends on the variation in the surface normal dN relative to the illuminant.

Table 1: Estimated illumination direction (tilt τ/slantσ) for Pentland

$\tau \backslash \sigma$	0	20	40	60	80	100	120	140	160	180
0	-/28									
10	10/29	25/29	46/28	66/27	78/28	94/27	112/26	134/26	156/27	170/27
20	25/27	55/28	66/28	78/27	81/27	96/27	107/25	125/27	154/28	172/26
30	7/30	32/30	54/30	67/28	81/26	97/25	115/25	133/27	151/25	174/23
40	4/26	28/30	48/30	60/29	78/30	98/27	117/28	135/29	154/24	176/24
50	3/26	26/30	47/30	63/29	77/30	99/27	114/28	130/26	155/27	173/29
60	2/26	22/30	45/30	59/29	78/30	98/27	111/26	132/29	156/27	174/27
80	1.3/26	22/30	45/30	63/24	78/27	100/25	111/24	130/25	151/27	174/23
90	1.6/26	22/30	45/30	63/24	76/27	101/25	117/24	133/27	151/27	174/23

In the previous formula we assume unknown both N and L. Illumination L can be estimated from the image by making assumptions about the variations of surface normals in the image. One sufficient assumption that changes in surface orientation are isotropically distributed. This is true of images of convex objects bounded by smooth occluding contour and is true on average over all scenes. In this case the expected value of the sum of dN over all image points is zero:

$$E(\sum_{x,y,\theta} dN) = 0$$

Under this assumption along any one image direction (dx, dy) $d\bar{x}_N$ is proportional to $cos\theta$, and $d\bar{y}_N$ is proportional to $sin\theta$, where $\theta = tan^{-1}(\frac{dy}{dx})$ and $d\bar{z}_N$ is zero. This allows setting up a system of equations which given the mean intensity gradients along several directions on the image give a least square estimate of the tilt and slant of the light source. It is worth noting that from this least-square regression the confidence intervals for interval for the tilt and slant can be estimated. With a single light source the tilt of the mean light direction can be derived from the weighted sum of tilts for all regions, with the weights inversely proportional to the confidence interval.

We tested this algorithm with computer generated spheres, and table 1 shows typical results. It seems from our experiments that Pentland's method is a very good estimator of tilt, but not not so good with slant, which is what Lee and Rosenfeld [lee85] observed.

5.2. Lee and Rosenfeld's Method

Lee and Rosenfeld modified Pentland's method in the following way. First they compute the illumination direction L, then compute the surface normal N in light source coordinate system and finally transform the surface normal into the viewer's coordinate system. The illumination direction at a point is derived from the distribution of intensity within a region. They also assume an isotropic distribution of surface orientations. We tested Lee and Rosenfeld's

Table 2: Estimated illumination direction (tilt/slant) for Lee and Rosenfeld

$\tau \backslash \sigma$	0	20	40	60	80	100	120	140	160	180
0	-/26									
10	17/26	29/28	42/29	56/30	67/28	79/27	91/28	105/26	120/25	139/26
20	28/39	50/44	60/44	69/44	73/41	86/38	101/40	114/30	140/30	163/24
30	8/47	25/49	49/52	60/50	75/49	89/48	106/45	126/40	147/35	170/33
40	6/58	41/58	41/57	59/58	76/60	90/59	108/55	128/49	150/46	171/44
50	6/64	35/68	47/64	61/64	76/65	90/64	107/61	118/63	150/58	151/59
60	6/71	21/70	41/67	61/67	78/71	89/78	106/66	130/64	154/66	172/69
70	6/78	19/76	42/75	62/74	82/79	85/80	105/72	129/72	158/74	173/78
80	4/84	21/81	40/81	61/80	84/82	85/83	108/79	128/80	152/80	174/84
90	4/88	20/86	41/87	61/87	78/86	92/86	109/86	129/86	145/86	176/88

algorithm with the same images of computer generated sphere, and got the results given in table 2. It seems that this algorithm gives a good estimate of the tilt and the slant. The larger the slant is the better it seems. A problem, however, is that when we evaluate the slant within small regions, the estimated slant varies considerably from region to region, though the mean value is good. This is because this algorithm is strongly dependent on the assumption of local sphericity.

5.3. An Improved Algorithm

We designed [liu94] a slightly different algorithm, also based on the assumption that surface orientations are isotropically distributed. From Pentland's analysis, we know that the derivative of image intensity along an image direction (dx, dy) satisfies equation 2. Since dI is measured as a difference of intensity between two pixels along a particular direction (dx, dy), dN for both points has the same tilt with respect to the direction (dx, dy). This leads to a zero expected value for $cos(\sigma_{dN})$ and estimates for $tan(\tau_L)$ and $tan^2(\sigma_L)$ from the least square regression similar to the one used by Pentland.

Using again the synthetic spheres, we obtained the results shown in table 3. The new algorithm does indeed better on slant for computer generated spheres than Pentland's original, but not as well as Lee and Rosenfeld's, which is not surprizing given that they assume they deal with spheres. It does better, however on real objects quite far from the assumption of local sphericity. As an example we applied it to the image of a "marshmallow" man, illuminated with from three different directions (see figure 2), and obtained the results given in table 4. We are currently pursuing our experiments with more real scenes, especially with material on objects or bodies, and the preliminary results are encouraging.

Table 3: Estimated illumination direction (tilt/slant), New Algorithm

$\tau \backslash \sigma$	0	20	40	60	80	100	120	140	160	180
0	-/52									
10	-1/51	18/51	38/50	60/51	78/51	98/51	118/52	140/52	161/52	181/52
20	21/45	49/47	62/46	73/47	80/51	98/51	113/45	129/52	160/52	181/50
30	0/50	20/51	49/48	63/49	81/49	95/49	117/50	140/49	160/51	181/49
40	0/47	40/45	40/45	61/45	82/43	97/45	118/47	139/47	161/52	181/48
50	0/49	30/37	45/40	63/44	81/45	98/45	116/45	128/43	160/47	166/44
60	0/47	17/43	40/43	63/44	82/47	95/42	116/48	139/47	162/47	181/49
80	0/58	20/43	40/43	61/43	83/59	97/56	119/53	139/52	160/50	180/55
90	0/73	19/47	42/36	60/45	81/57	99/60	120/62	139/58	157/50	179/64

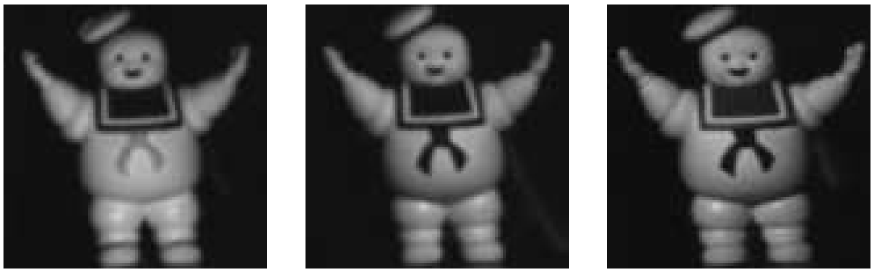

Figure 2: Image a. Image b. Image c.

Table 4: Estimated illumination direction(tilt/slant)

	$\tau \backslash \sigma$	Pentland	Lee	New
Image a.	-90/3	145/52	-60/89	-18/66
Image b.	17/65	31/48	13/88	39/66
Image c.	160/65	71/40	71/89	96/63

6. Common Global Illumination

The general strategy for common illumination is to generate a very simplified model of the real scene with few geometric primitives, and use this model mostly to compute approximate common global illumination, and additionally to retrieve viewing parameters and to determine visibility while re-rendering. Our restricted domain of application so far has been indoor scenes [four93].

6.1. Global Scene Modelling

The scene from the RVI is modelled with few (10 to 100) *boxes* (parallelepipeds in our case, not necessarily aligned with the axes). These boxes are chosen with several purposes in mind. First they will be substitutes for the objects for the global illumination computation. Therefore they should be few for fast computation, but at the same time their sides should be relatively coincident with the big flat areas in the pictures. For indoor scenes, which is our example application, one box will always be chosen so that its sides are the floor, ceiling and main walls. In addition each large object that moves with respect to the room should have its box. Since the box will also be used in the global illumination for shadow computation, and will be used in re-rendering for visibility with respect to the computer generated objects, a way to enhance its usefulness without too much additional modelling effort is to use transparent texture mapping on the sides of the boxes. Orthographic views of the object(s) enclosed in the box along the six (or less if not all needed) sides are digitized, the outline of the objects in each view is extracted (so far manually) and the texture is made transparent for the part which is not covered by the object. It is not of course equivalent to an accurate model of the object(s) in the box, but will produce more realistic form factors, shadows and visibility determination.

For each box a number of fiduciary points are chosen and measured with respect to the box frame of reference. If the box does not move, these points can be used to help retrieve the viewing parameters. If the box moves these points should be numerous enough to position the box accurately within the room frame of reference if their screen position within a frame is known. Four points at least are necessary, but more are used to compensate for hidden ones and for redundancy.

6.2. Estimation of Surface Reflectance

The relationship between the actual radiance (power/area*solid angle) or radiosity (power/area) and the pixel values are not known, since they depend in complex ways on the characteristics of the imaging and digitizing system. We are, however, only interested in generating CGI that are "matched" to the digitized versions of RVI, and therefore if we assume that the pixel values are proportional to the radiance (a big assumption for real imaging systems, but one that can be corrected for in precisely calibrated systems), then we only have to respect the same proportionality. At this stage we will treat the surfaces seen in the RVI as diffuse reflectors, and therefore for these the radiosity and the radiance is proportional as well. For a given pixel of value p_{xy}, given that the surface element S_i is seen at that pixel, we have:

$$B_i = K\ p_{xy} = E_i + \rho_i \sum_{\forall j} B_j F_{ij} \tag{3}$$

where B_i is the radiosity of element i, K is a constant of proportionality common to the whole image, E_i is the emission of element i (0 is only a reflecting surface), ρ_i the reflectivity of element i, and F_{ij} the form factor between j and i (fraction of energy leaving element j which is received by element i). Since we do not need K, we will just assume that in all the subsequent formulas the radiosities and the emissions have been divided by K, and therefore K is no longer appearing explicitly.

Even if we know that $E_i = 0$, we cannot on a single pixel separate the reflectivity (which we need for any global illumination computation) from the contribution of the other elements to the illumination of element i. We can, however, use a few heuristics to help us.

Following Cohen *et al* [cohe88] we can estimate the average form factor:

$$F_{*j} = \frac{A_j}{\sum_{\forall j} A_j}$$

where A_j is the area of element j. We can also estimate an overall inter-reflectivity factor as $R = \frac{1}{1-\hat{\rho}}$ where $\hat{\rho}$ is the average reflectivity (average weighted by area). The latter is easily estimated for a given environment. Given this the *ambient* radiosity can be computed as $B_A = \frac{R \sum_{\forall\ i} E_i A_i}{\sum_{\forall\ i} A_i}$ On the other hand, the ambient radiosity can be estimated by the average radiosity of a pixel divided by the average reflectivity: $B_A = \frac{\sum_{\forall\ xy} p_{xy}}{N\ \hat{\rho}}$ where N is the total number of pixels. This therefore gives us an estimate of the total power of the light sources present in the scene.

Surface elements are created from the sides of the boxes. The appropriate level of subdivision and how to compute it is currently an active subject of research. We arbitrarily chose a low level of subdivision since we are mainly dealing with correction to the illumination. To determine the radiosity from the real scene, ray-casting is used to match pixels and surface elements. For each surface element which has visible pixels associated with it, its radiosity is assigned the average of all the pixels it contains. The reflectivity initially assigned to the element is the the average reflectivity, multiplied by the ratio between its radiosity and the the average radiosity of the neighbouring pixels (we take a neighbourhood that contains four times as many pixels as the element). The rational for this heuristic is that if the neighbourhood radiosity is the same, there is no reason to assume anything about the reflectivity of the element. If the neighbourhood is darker, that indicates (but does not prove, of course) that the reflectivity is likely to be higher than average, and similarly if it is brighter. Reflectivity is clamped at 1. The surface elements with no visible pixels are assigned the ambient radiosity and the average reflectivity.

6.3. Modelling of Lights

Often in the case of indoor scenes, the position of the lights, if not their intensity, is known. In this case they are modelled (usually as a collection of polygonal emitters). We can then compute a global radiosity computation with each of the light sources separately assigned a default emission ($wlg\ E_0 = 1$) and the model boxes with their assigned reflectivity. The solution assigns each element a radiosity used as a relative base value. At the end of these computations, for each element i we have: $B_i = \sum_{\forall\ lights\ k} \frac{E_k}{E_0} B_{ik}$ where E_k is the emission of light k (unknown), B_i is the radiosity assigned to the element from the image, and B_{ik} is the radiosity computed for element i with light k at emission 1. Picking $m = k$ elements, we can compute the E_k which best fit our original estimates, and use these in the rest of the computation. Picking all of them for this computation provides a best fit for the distribution of power from the known light sources. Notice that we can constrain the sum to fit the estimate of total light intensity given by the ambient radiosity.

If nothing is known about the lights, then each element in the real scene is considered to be a light source with emission equal to its radiosity and its reflectivity is estimated as before.

The computer generated light sources are modelled as the other computer generated objects, and their emission is chosen depending on the application (but can be compared to the total emission of the real light estimated as given above).

6.4. Correction for Shadowing and Interreflections from CG Objects

The general attitude is to use what is already "computed" in the real scene, and compute only corrections for it. There are essentially two kinds of corrections: modifications due to the computer generated objects which block from the real lights or add inter-reflection from the real lights, and additional light from the computer generated light source(s). For the former, we deal with them differently depending on whether we have modelled the real light sources or not. If the real light sources are known, we perform a global illumination computation with the models of the light sources (at the emission estimated for them according to the above section) and the models of the CG objects. The result gives us for each element a B_{i*}, the new radiosity with all the real lights and the CG objets. The ratio B_{i*}/B_i tells us how to modify the radiosity of each pixel which belongs to element i. Notice that a decrease means that CG objects are casting a shadow on the real object, an increase that they are adding inter-reflection. If the real light sources are not known, we consider every element to be an emitter. To take shadowing into account, from each element i we shoot negative radiosity towards each other element j equal to the radiosity B_j of the target element. If the negative radiosity is not blocked, nothing happens, but if it is blocked this is subtracted from element i, as it means that the radiosity from j cannot reach i and i should be darker.

6.5. Global Illumination Computation

To compute the global illumination with all the elements now in place, we use *progressive radiosity*, as described in Cohen *et al* [cohe88]. The form factors are currently computed as the analytical version of form factors of discs standing in for the surface elements (which can be parallelograms, or any n-sided regular polygon) as discussed in Wallace *et al* [wall89]. and visibility among elements is determined by ray-casting.

The difference with normal CG radiosity computations is that for computer generated objects the whole radiosity is accumulated (which then include light reflected from all sources and inter-reflected from real and computer generated objects), but for the models of real objects only the additional radiosity ΔB_i (from the computer generated light sources, directly or indirectly) is stored separately.

6.6. Re-rendering

To re-render the scene we use ray-casting. For each ray R_{xy} at pixel xy which hits element i, if i belongs to a computer generated object then $p_{xy} = k \times B_i \times C_i \times T_i + \textit{specular component}$, else $p_{xy} = \text{old } p_{xy} + k \times \Delta B_i \times$ old p_{xy}, where C_i is the colour of the element, and T_i an optional texture value. In effect for RVI elements the old pixel value plays the role of the texture. Note that we compute the results in three separate colour channels (RGB), not because it is right, but because a spectral computation would not affect the steps of our computations.

6.7. An Example

To illustrate the steps and the results, we have produced an animated sequence whose characteristics are briefly described in this section. In the RVI the scene consists of the corner of a room in which a desk supports a workstation monitor, keyboard, mouse and soccer ball. Under the desk is the CPU of the workstation. In the middle of the room is a small square table with a book on it. In the corner of the room is a file cabinet, and on the left a small white box on the floor. The main lights illuminating the scene (out of view for all the frames) are a fluorescent light panel on the ceiling near the far corner, and an incandescent "luxo" style lamp pointing at the ceiling, roughly above the camera.

The whole video sequence is about 700 frames of video. During about the first 200 frames the camera zooms out, and for the rest the camera slowly rotates from right to left (from the desk to the corner of the room). About 300 frames into the sequence the soccer ball starts rolling, falls off the desk, rolls on the floor, bumps into the white box on the floor and comes to rest by a leg of the little table. We therefore have both a moving camera and a moving object within the scene. Figure 3 shows a frame near the end of the original RVI.

The average reflectivity $\hat{\rho}$ was set at 0.7. The position of the fluorescent light overhead was determined, and it was modelled by a rectangle. The incandescent lamp is modelled by two rectangles oriented appropriately. To

Figure 3: Frame from original video

Figure 4: Final Merged Image

Figure 5: Original frame and overlaid wire model

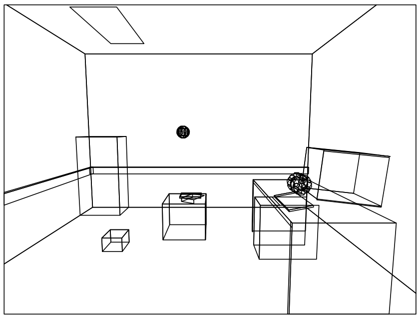

Figure 6: Wire model for the whole scene

determine the camera positions an interactive program was used which displays the wireframe of the model of the real objects, superposed to the RVI. A viewer then manipulates the viewing parameters until a satisfactory match is obtained. In practice we used only about 25 frames of the RVI to be matched, and derived the other viewing parameters by interpolation. Figure 5 shows an example of an RVI together with the matched wireframe image.

The computer generated objects consist of a book added on top of the real book on the small table and a spherical light hanging from the ceiling almost directly above the small table. At the end of the sequence a box comes out of the small box on the floor and changes shape and colour. The light source turns on near the beginning of the sequence and starts swinging when the soccer ball starts rolling. The intensity of the additional light source has been chosen to be similar to the intensity of the lights present in the RVI. Figure 6 shows a wire frame of the scene including the computer generated elements. Figure 4 shows the resulting images for the frame shown above.

6.7.1. Discussion

The result is fairly satisfactory. Of course only when viewing the video can one be persuaded of its effectiveness. One can see from the animation (we did not try to hide the artefacts) that the main problems are the usual quantization problems from global illumination computations and the mis-registration of the viewing parameters between the CGI and the RVI.

Real-time merging is still far away. Computation of the merged images took about 10mn per frame on a 30 MIPS machine (the frame resolution is 646x485). This time is reasonable enough, however, so that one could save rendering time if the alternative is to model the RVI in all its details (that is if our models are good enough).

7. Surface Characteristics from Surface Colour

To get a certain shading effect on a surface, a user must determine the surface characteristics that will produce this shading according to the strict rules of the reflection model, the light definitions and the geometry of the scene. This process is called *inverse shading*. Inverse shading has been investigated mainly for to provide a useful alternative to the model/render cycle commonly used, but it is obviously also important to be allow the retrieval of surface charateristics from a real scene for the purpose of reshading. [poul92], [poul93].

The approach is to use a *painting* paradigm. The user controls, almost in real time, the values of the surface parameters by simply painting colour points on a surface. The modeler will attempt to determine values for the surface characteristics for which the colour points will retain their colour (within a certain threshold) when the surface is finally shaded. Each colour point introduces a constraint in a system of equations. The under-constrained systems are solved via non-linear constrained optimization while the over-constrained systems are solved via a weighted least-squares approximation with penalty functions to constrain the values of some surface parameters. Surface attributes such as surface colour, quantity of ambient, diffuse, specular reflections and the ratio of dielectric and conductor properties are therefore automatically determined as the user adds colour points and interactively moves them on the surface.

This approach reduces the number of inverse shading steps that a user must perform. Inverse shading problems have been investigated in computer vision. Shape from shading, direction of illuminants and identifying some surface characteristics from a single or a series of images form some applications of this concept. Some of these problems are difficult to solve properly because of the lack of information (constraints) and the uncertainty associated with each constraint. In a computer graphics modeler, the viewing parameters and the exact scene geometry are known and therefore, some uncertainties are removed and many problems become easier to solve.

7.1. Painting Scenario

The interface of our painting system is relatively simple. We assume the geometry of the 3D scene and its lighting already designed or otherwise known. The user selects colours and applies them to any visible point in the 3D scene. A colour point is represented by a small disk aligned on the surface along the normal at this point. The center of the disk indicates the 3D location of the colour point. The disk is painted with the selected colour but is not shaded. These colour points can be moved on the surface, deleted, their colour can be modified and the size of each disk can be scaled up or down. A larger disk is convenient to properly see the colour of a point and to manipulate it. Since colour is context sensitive, it is important to be able to increase the disk in order to better perceive its colour. A smaller disk occludes less of the underneath surface, allowing to better see the shading gradient around the colour point. This can be important when designing small highlights.

7.2. Solutions

When a surface element is shaded, the radiance directly reflected to the pixel is function of the illuminants, the scene geometry and the surface parameters. For one directional light, equation 1 in Section 4 is the expression used. If the value for each variable is known, the reflected radiance is easily computed. In our painting system, we attempt to solve the inverse problem. Therefore our system must find values for the surface characteristics that would satisfy the colour points if the full shading were performed. In the above reflection model, for a given point, all the surface attributes are independent in each channel. Therefore without lost of generality, we can consider solving the problem in one channel. The approach is identical for the others.

For a given colour point, the known values in the diffuse reflection can be summed for all m lights as:

$$L_d(c) = \sum_{i=1}^{m} \int_{\omega_i} L_i(c)(\vec{N} \cdot \vec{L_i})\, d\omega_i \qquad \text{for } \vec{N} \cdot \vec{L_i} > 0.$$

And similarly for the specular reflection:

$$L_s(c) = \sum_{i=1}^{m} \int_{\omega_i} L_i(c)(\vec{N} \cdot \vec{H_i})^n\, d\omega_i \quad \text{for} \begin{cases} \vec{N} \cdot \vec{L_i} > 0 \text{ and} \\ \vec{N} \cdot \vec{H_i} > 0. \end{cases}$$

Generally speaking, $L_d(c)$ and $L_s(c)$ can be computed for any type of light source, whether it is a point light, a linear light or an area light.

Each colour point contributes to a new equation in each of the three channels. If there are as many independent equations as variables, the system of equations can be solved and a unique value identified for each surface attribute.

With the shading equation given, and if we know three colour "points", then a unique value for k_a, k_d and k_s can be computed. Unfortunately, it is unlikely a user will be able to provide the exact colours that would lead to a solution. We need to transform the problem such that it would lead to a solution.

7.2.1. Constrained Optimization

When less points than variables are placed on a surface, each colour point introduces two constraints in a system of equations. The colours are then interpreted as a range of acceptable colours for the point. We use a non-linear constrained optimization algorithm to find a unique solution for all constraints. The objective function defines the behaviour of the system. We studied various objective functions in this paper, with one of the simplest and most effective being to maximize the diffuse term. It is also possible to let the user choose her objective functions within a library of objective functions in order to control and personalize the behavior of the system.

In order for the optimization algorithm to find a solution, a feasible initial guess must be provided. One can investigate the domain of each variable to find such an initial guess. Another solution consists in enlarging the domain of some variables to find an intermediate solution. The intermediate solutions are then used as initial guesses in narrower domains until the domain is the same as originally. We can also use the search provided by the optimization algorithm. Still, all these approaches can fail. If this happens, it is possible that there is no possible solution to the system. The violated constraints are shown to the user to determine what can be done to relax the constraints. If a solution is found, the final colours are guaranteed to be within their constraints.

7.2.2. Least Square Solution

When more points than variables are placed on a surface, each colour point is considered a sample with two constraints. We use non-linear least-squares fitting to find a solution. The constraints are enforced by penalty functions and weights control the behavior of the system, depending on the location of the colour points. The results are not guaranteed to be within the constraints but are usually very close. A good choice of objective functions and weights and penalty functions provides a smooth transition between the optimization algorithm and the least-squares fitting.

7.2.3. Discussion

This system deals only with direct shading. By doing so, it provides high control of the full shading while keeping the number of constraints and variables relatively low. This allows the system to return solutions almost in real time, which used in conjunction with hardware real time rendering, provides a direct feedback to the user as she adds and moves colour points on the surface. This solution can also be tailored to the variations in different shading models.

To apply this approach to real images, one needs information about the lights and information about the local geometry of the surface. Techniques described in the previous sections can give us some elements of the answer. It should also be noted that some parameters of the surface geometry can also be included as part of the unknowns the system has to solve for. We plan to experiment with this approach in the near future.

8. Further Work and Conclusion

We still are quite far from *automatic*, *seamless* merging. It would be extremely useful to have relatively dense depth maps of the real scene. That would help both illumination computations and visibility calculations. Again passive methods (optical flow, stereo depth) or active methods (laser range finder) can be brought to bear. Better tracking of viewing parameters is also essential. Model based tracking, as described by Lowe [lowe87] can help in providing an automatic solution, with or without fiduciary marks. It should be stressed again that in some applications the precision required is quite high. We are currently investigating a combination of geometric and image registration methods to adress this particular problem.

9. Acknowledgements

The work on compositing with shadow is with Russ Krywolt. The work on estimating the illuminant characteristics is with Atjeng Gunawan. The work on light direction from shading is with Lili Liu. The work on common illumination is with Atjeng Gunawan and Chris Romanzin. The work on surface characteristics from surface colour is by Pierre Poulin. We gratefully acknowledge the support of NSERC through operating grants, an equipment grant and a Strategic grant which considerably facilitated the research described here. The support of the University of British Columbia in establishing a computer graphics laboratory in our department is greatly appreciated, as is the support of IBM Canada, which established GraFiC, an invaluable facility.

References

[cohe88] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. “A Progressive Refinement Approach to Fast Radiosity Image Generation”. *Computer Graphics (SIGGRAPH ’88 Proceedings)*, Vol. 22, No. 4, pp. 75–84, August 1988.

[duff85] T. Duff. “Compositing 3-D Rendered Images”. *Computer Graphics (SIGGRAPH ’85 Proceedings)*, Vol. 19, No. 3, pp. 41–44, July 1985.

[four93] Alain Fournier, Atjeng S. Gunawan, and Chris Romanzin. “Common illumination between real and computer generated scenes”. *Proceedings of Graphics Interface ’93*, pp. 254–262, May 1993.

[gamb87] C. Gamble and T. Poggio. “Visual integration and detection of discontinuities: the key role of intensity edges”. AI-Memo-970, Cambridge, MA, 1987.

[guna91] Atjeng S. Gunawan. “Estimating the illuminant color of a scene from its image shading”. *Proceedings of the 1991 Western Computer Graphics Symposium*, pp. 29–30, April 1991.

[kryw93] Russell Krywolt. “Post-Processed Shadow Determination for Composition of Depth Images”. M.Sc. Thesis, Department of Computer Science, University of British Columbia, October 1993.

[lee85] Chia-Hoang Lee and Azriel Rosenfeld. “Improved methods of estimating shape from shading using the light source coordinate system”. *Artificial Intelligence*, Vol. 26, pp. 125–143, 1985.

[lee86] Hsien-Che Lee. “Method for computing the scene-illuminant chromaticity from specular highlights”. *Journal of Optical Society of America A*, Vol. 3, No. 10, pp. 1694–1699, October 1986.

[lee88] Hsien-Che Lee. “Estimating the Illuminant Color from the Shading of a Smooth Surface”. Technical Report A.I.Memo No. 1068, MIT, 1988.

[liu94] Lili Liu. “Shading and Re-shading from Real Images”. M.Sc. Thesis, Department of Computer Science, University of British Columbia, 1994.

[lowe87] D. G. Lowe. “Three-dimensional object recognition from single two-dimensional images”. *Artificial Intelligence*, Vol. 31, No. 3, pp. 355–395, March 1987.

[marr84] J. L. Marroquin. “Surface reconstruction preserving discontinuities”. AI-Memo-792, Cambridge, MA, 1984.

[pent82a] Alex P. Pentland. “Finding the illuminant direction”. *Journal of Optical Society of America*, Vol. 72, No. 4, pp. 448–455, April 1982.

[pent82b] Alex P. Pentland. *The Visual Inference of Shape: Computation from Local Features.* Ph.D. Thesis, Department of Psychology, MIT, 1982.

[port84] T. Porter and T. Duff. “Compositing digital images”. *Computer Graphics (SIGGRAPH '84 Proceedings)*, Vol. 18, No. 3, pp. 253–259, July 1984.

[poul92] Pierre Poulin and Alain Fournier. “Lights from highlights and shadows”. *Computer Graphics Special Issue (1992 Symposium on Interactive 3D Graphics)*, Vol. 26, pp. 31–38, March 1992.

[poul93] Pierre Poulin. *Shading and Inverse Shading from Direct Illumination.* Ph.D. Thesis, Department of Computer Science, University of British Columbia, December 1993.

[terz83] D. Terzopoulos. “The role of constraints and discontinuities in visible-surface reconstruction”. pp. 1073–1077, 1983.

[terz88] D. Terzopoulos. “The computation of visible surface representations”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 4, pp. 417–438, July 1988.

[wall89] John R. Wallace, Kells A. Elmquist, and Eric A. Haines. "A Ray Tracing Algorithm for Progressive Radiosity". *Computer Graphics (SIGGRAPH '89 Proceedings)*, Vol. 23, No. 3, pp. 315–324, July 1989.