ADVANCED COMPUTER VISION AND GRAPHICS COLLABORATION TECHNIQUES FOR IMAGE-BASED RENDERING

SAMUEL BOIVIN AND ANDRÉ GAGALOWICZ [*]

Abstract. The idea of using real images to generate photorealistic computer graphics (scenes) has led to the development of Image-based modeling and rendering. These techniques are very similar to those developed some years ago within the framework of analysis/synthesis collaboration. In this paper, we present a new approach to reconstruct the 3D geometry and photometry of a scene based upon two distinct processes. A vision process uses two digital images of a scene captured with a camera to reconstruct its full geometry. A computer graphics process uses a single image to recover the photometry of the the surfaces (i.e. the surface reflectances) and the radiance-to-pixel function by minimizing an error function. The generated images are then used as a feedback to modify the surface reflectances. Our aim is to find the simplest reflectance model allowing to faithfully reconstruct the original image of a scene, keeping in mind that the related photometric analysis is highly dependent on the complexity of the searched model.

Our approach generates photorealistic images using a rapid global illumination algorithm including the computation of a specular component. This algorithm is driven by the mean square error between the real image and the synthetic one, and minimizes it with respect to the parameters of the photometric model and of the radiance-to-pixel conversion functions. Several applications of this method are presented, such as augmented reality.

Key words. analysis/synthesis, image-based rendering, image-based modeling, computer graphics, augmented reality, radiosity, model-based vision, image segmentation, region matching, object reconstruction

1. Introduction. Collaboration between computer graphics and computer vision has become a very developed research domain. In fact, there are two (dual) interesting outcomes to this collaboration.

On the computer graphics side, image synthesis was limited by the fact that all generated images did not use real data or real images to be computed. Therefore, no realistic images could be synthesized. This was clearly a limitation of image synthesis methods. With the (latest) advances in image-based modeling and rendering research, it is now possible to reconstruct the full geometry and photometry of a scene, using digital images of the real world, captured with a camera. This gives the opportunity to apply computer graphics algorithms merged with vision ones to compute synthetic images of real world. A lot of applications are made possible with these techniques: some new synthetic or even unreal objects can be added to the original image, new viewpoints of the scene can be computed using classical rendering methods of computer graphics, etc.

On the vision side, this cooperation oriented image interpretation towards a model-based approach. Computer graphics techniques bring feedback and offer improved possibilities to realistic vision solutions. These solutions show (considerably) better stability when comp ared to previous open-loop solutions.

Our presentation introduces a new method which is original in two aspects: on one hand, only two images are necessary to reconstruct the full 3D geometry of the scene, and on the other hand, only a single image is necessary to reconstruct the full photometry of the same scene. This last method is based upon a new algorithm that is at the hear t of the analysis/synthesis concept: computer vision is used to reconstruct

[*]This work was done at I.N.R.I.A. Rocquencourt, MIRAGES Project, Domaine de Voluceau, 78153 Le Chesnay Cedex, {Samuel.Boivin|Andre.Gagalowicz}@inria.fr

the photometry and a computer graphics rendering algorithm produces feedback to drive this operation. Finally, a synthetic image is obtained, simulating the real world very realistically.

2. Vision. The purpose of image analysis is to produce an intrinsic interpretation and representation of a scene from images. In the general case, we have to consider image sequences. But in this paper, we restrict ourselves to the case of analysis of a stereo pair (see figure 2.1), leaving to future research the updates of the techniques which will be presented below, for the case of image sequences.

In general, an intrinsic representation of a scene consists of:

- a three dimensional geometric model of this scene which includes light sources and cameras (positions and extensions)
- a photometric description of the object surfaces and of the light sources. This description has to account for the simulation of the energetic transfer among these objects.

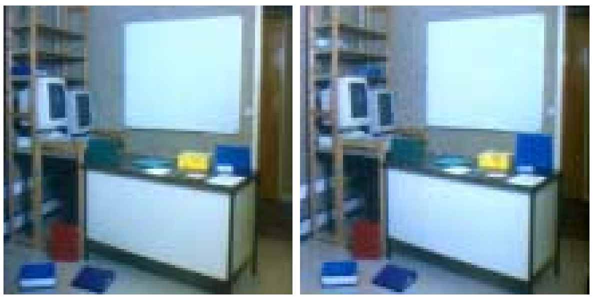

FIG. 2.1. *Original stereo pair to analyze.*

If both types of information are available, they can be used by a domestic robot for example, because it will have a perfect understanding of the scene. It can also be used in multimedia application s such as virtual reality, augmented reality, post-production, etc.

First, we detail how to obtain the geometric representation of the scene from a stereo pair. We suppose that we use calibrated cameras to do this analysis.

2.1. Segmentation. We begin with the segmentation of both images of the stereo pair. Numerous techniques, either bottom-up or top-down, were tried. We have chosen a non parametric method which is fully data driven (see [2] for more details). This method is a regularization technique which minimizes some weighting between a mean square approximation error and the total frontier length produced by the segmentation.

Let s be a pixel in the image domain I, Z_s is its grey level intensity (or the R, G, B vectors for color images) in the original image, L_s its label in the segmentation. This criterion is defined by:

$$E_Z(L) \sum_{k=1}^{N} \sum_{s \in I_k} (Z_s - \bar{Z}_k)^2 + \lambda \sum_{<s,t> \in I \times I} (\mathbf{I}_{L_s \neq L_t}) \tag{2.1}$$

where : $< s, t >$ denotes a couple of adjacent pixels

$$\mathbf{I}_X \begin{cases} 0 & \text{if } X \text{ is true} \\ 1 & \text{if } X \text{ is false} \end{cases}$$

a region I_k is a set of connected pixels having the same label L

N is the (variable) number of regions

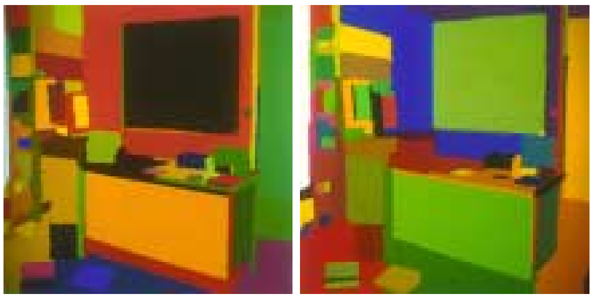

FIG. 2.2. *Segmentation results of the stereo pair (each color represents a distinct region).*

The first term of equation (2.1) is the sum of the distances between the original image and its piecewise constant approximation. The second term is the length of the boundary between regions and it penalizes excessive splitting of the image.

The only parameter defining the segmentation is λ and this technique minimizing equation 2.1 is fully automatic and does not contain any parameter itself (threshold, ...). So except for λ, it is effectively driven by the image data. If λ is small, this segmentation produces many regions and few if λ is big. In fact, λ drives the refinement of the segmentation.

The results corresponding to the stereo pair shown in figure 2.1 are presented in figure 2.2. Of course, they are not perfect as segmentation is an ill-posed problem, but a certain number of obtained regions are correct.

2.2. Region Matching. The next step consists of the region matching of the two images. We have to find out what are the region pairs (left and right) which come from the same three dimensional region that we suppose planar. Our experimental conditions are the following: the distance between the two cameras (or camera positions) is small with respect to the camera-to-object distances, and the angle between the two optical axis varies from five to ten degrees. Thus, both images are very similar (as it can be seen in figure 2.1) as we have almost two parallel projections. All real 3D facets must therefore have very close projections on both images. So, we have to find out similar regions on left and right segmentations of figure 2.2. We use global region features and the epipolar constraint to obtain the matches. The computation requires a very short time, as this combinatorial problem is considerably reduced thanks to the constraints (see [21]).

Results of such a matching are shown in figures 2.3. Matched regions get the same color. Regions which were not matched are colored in black. Fortunately, a certain number of regions are matched correctly as we allowed only very strong similarities among regions. This gives us the opportunity to start the 3D reconstruction of the identified parts of this scene.

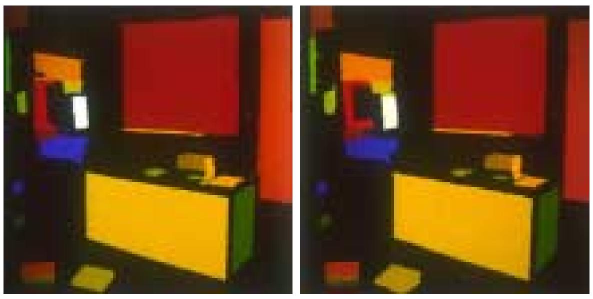

Fig. 2.3. *Matching results using the segmentation images 2.2.*

2.3. 3D reconstruction. Given previous region pairs and calibration data, it is possible to compute the 3D position of the planar facets from which each region pair is the projection[43]. The originality of our method lies in the use of the epipolar constraint and of the coherence of the region to perform the reconstruction. It is a global technique (and not a point-to-point one). Roughly speaking, the position of the 3D facet obtained is such that it mirrors the left region exactly on the right one, and conversely. More precisely, we maximize the overlap between the right region and its mirror projection coming from the left one.

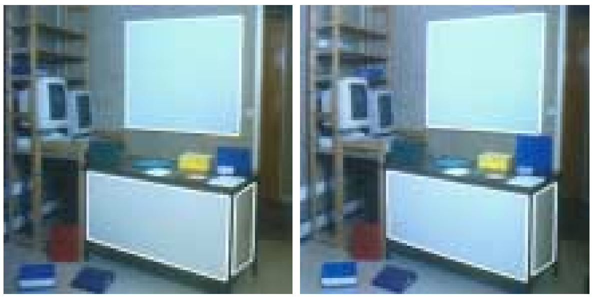

Fig. 2.4. *Reconstruction results showing the reprojections of the 3D facets found (white wireframe) on the natural images.*

Once again, to avoid errors we kept only the most important regions for the reconstruction. Results in figure 2.4 have been computed using this technique. The 3D facets were reprojected on the left and right images and visualized in a white wireframe. The reconstruction seems good, but, if we display its true three dimensional location with respect to the real scene database, it appears very bad (see figure 2.5).

The error is very important in the direction of the focal axis of the camera. When we reproject it on left and right images, this error disappears.

2.4. Recognition. The final step in geometric analysis is recognition. Our approach is a model-based one: we suppose that we dispose of a 3D model of the scene. The problem is restricted to the case where we have to find out what is the part of the database which is seen from the camera. How to get a scene database is a very important issue but it will not be discussed here (readers interested in this topic are referred to [32]). In other words, we have to find what is the position of the camera with respect to this database.

To do so, we compare the 3D facets obtained from section 2.3 with the set of all

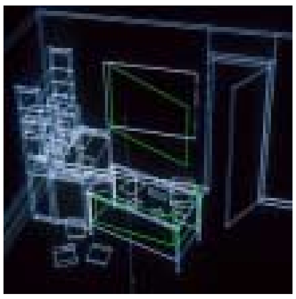

FIG. 2.5. *Result showing the 3D reconstruction of a three facets(green) in the wireframe of the real scene (white).*

facets of the database. The problem is to determine what the relationships between these two sets are. The solution we propose uses a hashing technique[27] in order to perform, for each model facet, a first pruning of the scene ones which will be candidates for future matching. The algorithm then looks for the maximum number of matches compatible with the fact that this matchings must correspond to a rigid transform between the two sets of facets. We use a displacement error threshold to evaluate this property. If several solutions exist, we take the one which minimizes the mean square error between the scene facet vertices and the rigidly displaced model ones. For this purpose, we use the classical algorithm described in [17], which gives the best displacement (R, T) fitting the database model with the observed scene. Finally the image geometry is obtained by windowing and clipping the displaced database (using (R, T)) in order to retain only what is perceived by each camera. A Z-Buffer algorithm allows to get rid of all non visible facets. The result of this operation is shown in figure 2.6.

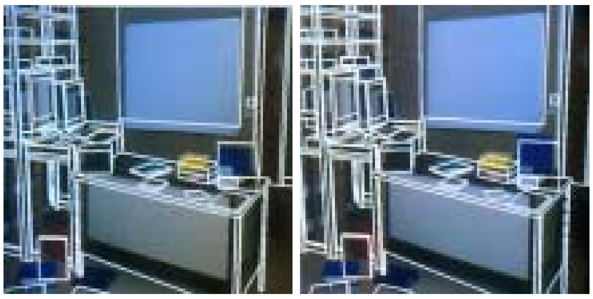

FIG. 2.6. *Recognition results using the reconstructed facets of figure 2.4. The original stereo pair is visible on the background and the white wireframe shows the position of the part of the database that the algorithm found.*

As it can be observed, the model database proposes a scene interpretation but with a very approximative position of the objects. This is due to the fact that the reconstructed facets were themselves badly positioned. Nevertheless, this recognition gives a model-based segmentation of both images which is much more interesting than in figure 2.2. How to improve the recognition ? The idea consists of using what we see, which is the difference between the segmentation of the real images(figure 2.2) and the segmentation proposed by recognition. We restrict ourselves to the use of the regions corresponding to the reconstructed facets. More precisely, we compute the error between the projected vertices of the model facets (associated with the reconstructed ones) and the corresponding vertices of the segmented regions (blackboard and desk facets of figure 2.2) as a function of (R,T). This is a non linear function and we minimize it using a simple gradient technique initialized by the (R,T) values coming from the linear step described above. A first iteration is shown in figure 2.7.

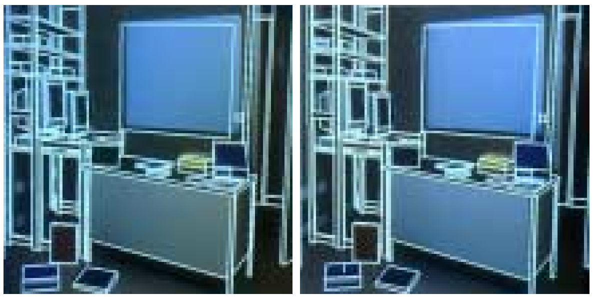

FIG. 2.7. *Recognition results: non linear improvement after one iteration.*

The procedure is further iterated; a second iteration is shown in figure 2.8. This seems to be an acceptable recognition result. Some discrepancy between the recognition result and the ideal solution still remains, which comes from various defaults. The major one is due to segmentation errors: we try to fit a best reprojection of ideal facets with segmented regions which are not exactly what they should be. Another important defect is the database itself that was constructed interactively. It could be possible to improve its position by comparing the model-based segmentation with the original data of both images themselves. But this was not performed in our case.

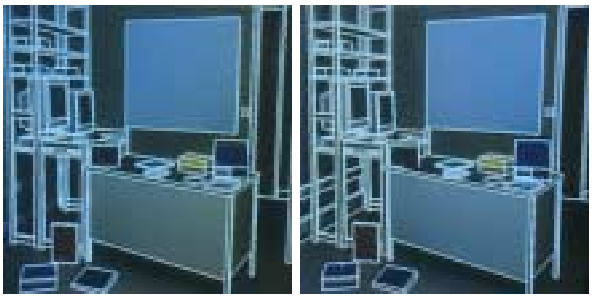

FIG. 2.8. *Final recognition results.*

In conclusion, our results prove that a model-based approach allows to solve the problem of the 3D automatic geometric reconstruction of a scene. The result presented corresponds to what we called *a global case* where we have a global model database. But in essence, this method could be investigated in a local approach where the database would consist of a set of distinct possible objects. The local method was still not solved completely and will be the subject of future research.

After having obtained a three dimensional representation of the scene shown in the images, we are interested now in its photometry analysis. This is a difficult problem that we propose to solve using an analysis/synthesis collaboration procedure. Before that, we describe some computer graphics techniques which are necessary for the task.

3. Computer Graphics Techniques.

3.1. Generating a photorealistic synthetic image. A lot of techniques have been developed to synthesize images. Here we are going to introduce the methods that are necessary to compute a photorealistic one.

The rendering process is subdivided in two important parts: a hidden-surface removal technique (which is used to determine for each pixel of the image to synthesize what is the 3D object, which geometric element of the object will be seen there), and the color computation of this pixel.

There are several hidden-surface removal techniques. The most popular are the Z-Buffer method (Catmull[7]) and the Ray Tracing one[3, 30, 47] which are used today in many applications, especially in special effects for movies. Of course, there are other famous but less used techniques like the Warnock subdivision[45], the Newell-Sancha algorithm[33], etc. We are interested in the first two ones. We also want to describe the A-Buffer[6]; it is an advanced Z-Buffer technique yielding nice anti-aliasing effects. Although it presents very interesting properties, it is not so currently used.

The techniques for computing color are numerous. This is generally called *illumination*: one can estimate the color of a surface while taking into account only its own photometric properties (*local illumination*), or, could include the relations and the influence that the other objects have on it (*global illumination*).

3.2. Famous hidden-surface removal techniques.

3.2.1. Z-Buffer. The Z-Buffer[7] is certainly the most famous hidden-surface removal technique in the world. Almost all graphics hardware now include hardware Z-Buffering. This method proceeds as follows: given a viewpoint and its viewing direction and a three dimensional scene subdivided into facets, the Z-buffer technique computes a buffer containing the information of depth for a surface of the scene. Firstly, the facet coordinates are transformed into the camera space, and projected onto the screen. Then, a filling algorithm is applied to fill each facet and compute its depth (for all interior points). This depth is compared to the one which is contained in the depth buffer (Z-buffer). If this depth is smaller than the one inside the Z-buffer, then the new depth is the newly computed one.

Z-Buffer Pseudo-Algorithm:
```
For each $x, y$ of the image buffer
    {
    $ZBuffer[x][y] = +\infty$
    }

For each facet $i$ in the scene
    {
    $i'$ = Transform $i$ in the camera space coordinates
    $i''$ = Perspective projection of $i'$ on screen
    For each pixel $x_{i''}, y_{i''}$
        {
        if $z''[x_{i''}][y_{i''}] < ZBuffer[x_{i''}][y_{i''}]$
            {
            $ZBuffer[x_{i''}][y_{i''}] = z''[x_{i''}][y_{i''}]$
            $Image[x_{i''}][y_{i''}]$ = Compute Color For Surface $i$
            }
        }
    }
```

3.2.2. A-Buffer. The A-Buffer technique [6] is close to the Z-buffer one because it uses the same principle of surface projection and filling. The biggest differences with the Z-buffer algorithm are that, in one hand, for each pixel we store the list of surfaces which are hitting it rather than just one surface, and on the other hand, we compute for each surface the part that its projection occupies in the pixel (this is called a *fragment*, see figure 3.1). These operations correspond implicitely to a pixel oversampling, which has the property to reduce aliasing effects.

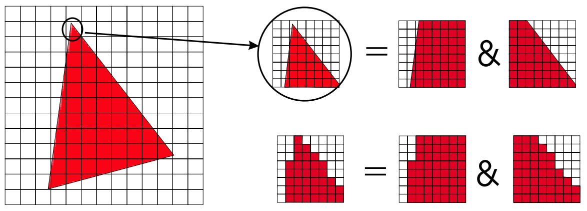

FIG. 3.1. *Fragment example. Here the fragment is the combination of two precomputed fragments, where the top part of the figure is the real representation and the bottom one, the (discrete) computer one.*

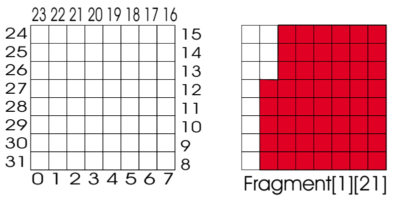

FIG. 3.2. *Fragment coding and array access to get a fragment.*

In a pixel, a fragment of a surface is approximated by a convex polygon, so that a polygon area can be considered as the logical intersection between the parts of the pixel obtained (using a counter-clockwise orientation of the polygon) by the right half-space of this pixel area determined by each line of the polygon. In order to perfom this operation efficiently, we precompute a two dimensional array containing all possible basic fragments obtained by cutting the pixel with a single line, starting from the border of the pixel and leaving this border as shown in figures 3.2 and 3.3. This operation has a very low CPU cost since it is done only once: the precomputed basic fragments are stored in a static table directly in software.

During facet filling, all fragments (except pixel entries and exits, of course, which are computed each time) are read from the previous array. For each pixel we build a dynamical list containing (in depth order) the depth and the fragments (8×8 array) of each surface having a visible contribution to the pixel. This is different from the Z-Buffer technique where only a depth value is stored. When all facets have been treated, we traverse all the lists of pixels and take in to account the occupancy percentage(figure 3.4) of the correspond ing facet in the fragment in order to compute the final color. Each time a new fragment is discovered, to check if it has a visible contribution, a XOR operation is applied to remove the parts of the surfaces that have been previously processed(figure 3.5).

3.2.3. Ray Tracing. Ray Tracing[3, 30, 47] (sometimes called Ray Casting when it is limited to hidden face removal) is a powerful technique that is used in many

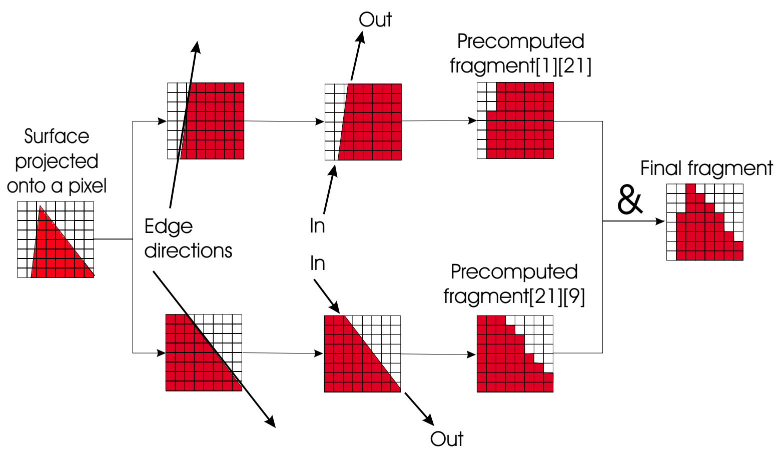
Out
Precomputed fragment[1][21]
Surface projected onto a pixel
Final fragment
&
Edge directions
In
In
Precomputed fragment[21][9]
Out

FIG. 3.3. *Fragment Computation. First we use the facet edges as line supports. Then we can compute the pixel entry and exit: this entry and exit (integers) are the direct indexes to the corresponding precomputed fragment.*

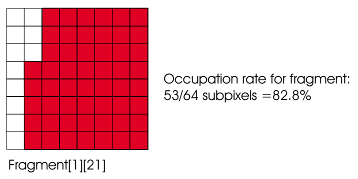

FIG. 3.4. *Fragment occupation.*

applications today. Ray Tracing starts by sending rays from the observer through a virtual screen placed in front of the scene (see figure 3.6).

All these rays are mathematically intersected with all the objects of the scene, which do not need to be subdivided into facets (as opposed to Z-Buffer). All the intersections found are depth-sorted in such a manner that the nearest one will correspond to the closest visible object. This is clearly an advantage of Ray Tracing method because objects occupy a smaller amount of memory than if they were approximated by facets. Another advantage of this technique is that we may compute the exact intersection between the ray and the object: the Z-Buffer method needs to interpolate depths from vertices where the filling algorithm is processing the interior of the facet.

The greatest advantage of Ray Tracing is the possibility to take into account specific properties of objects like transparency or specularity (mirror) for example. If a mirror object is intersected, a secondary ray is computed using Descartes'Law to find the next object that is seen from this mirror: this can be applied recursively and this recursive approach is specific to Ray Tracing.

Two big problems of Ray Tracing method are that, on one hand, it is very slow because of the intersection computations (this could be optimized using specific techniques such as Octrees[29], BSP[19], Bounding Boxes[46]), and on the other hand it generates aliasing effects due to the screen sampling and could miss some far objects (see figure 3.7). This could be resolved too, using adaptive subdisivion of pixels or stochastic supersampling (more than one single ray per pixel are sent). Ray Tracing

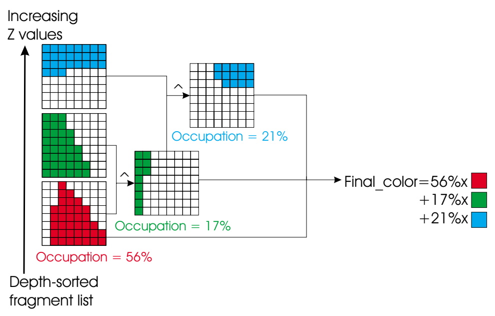

FIG. 3.5. *Fragment list processing: each fragment is XORed with the previous one to compute the final contribution.*

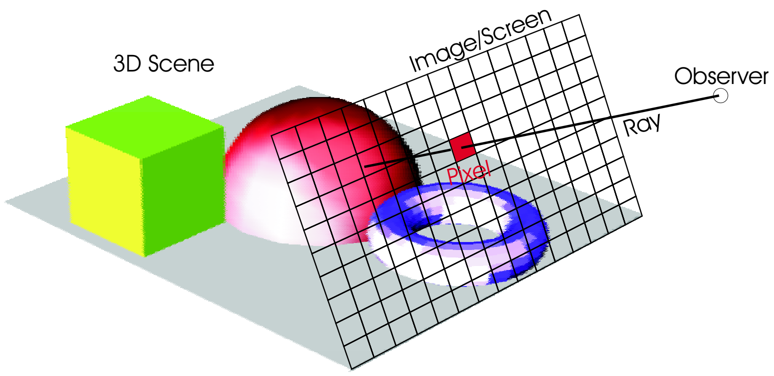

FIG. 3.6. *Ray Tracing principle: one ray is sent from the observer through each pixel of the screen and an intersection is eventually computed (here with a red sphere).*

is very simple to implement and can be very efficient using all classical optimizations. It can process very easily specular, transparent objects and more complicated ones like participating media (fire, gazeous phenomena).

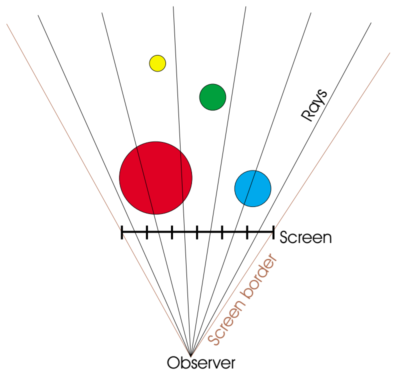

FIG. 3.7. *Ray Tracing sampling problem: rays could miss some objects. Here the yellow and the green sphere passed through the intersection tests.*

3.3. Illumination techniques.

3.3.1. Local Illumination. A local illumination model is generally used for the scene, where we do not want to take into account diffuse or specular interreflections, and we consider that object illumination mainly comes from the direct effect of light sources. Such models are very easy to compute but they suffer from a lack of realism. These techniques are neither physical enough nor sophisticated enough to render diffuse interreflexions, for example. So we will not discuss this part too much since we just want to introduce techniques that could bring photorealism.

Let us consider one of the most popular local illumination models: the Phong model[35] (often associated with Phong shading[35]).

$$I_\lambda \;=\; I_{a\lambda} k_a O_{d\lambda} + \sum_{i=1}^{m} f_{att_i} I_{p\lambda_i} [k_d O_{d\lambda} (\vec{N} \cdot \vec{L_i}) + k_s O_{s\lambda} (\vec{R_i} \cdot \vec{V})^n] \tag{3.1}$$

I_λ is the final computed color for λ wavelength;
$I_{a\lambda}$ is the ambient intensity which represents a type of constant luminosity applied to all the scene;
f_{att} is the attenuation function of the light source intensity, depending on the distance separating this light source from the surface. $f_{att}\, D\, \frac{1.0}{d^2}$ for example, with d the euclidien distance between the light and the facet;
$I_{p\lambda}$ is the light source color for each λ wavelength;
$O_{d\lambda}$ and $O_{s\lambda}$ are the diffuse and the specular color of the object, respectively;
k_a, k_d, k_s are the ambient, diffuse and specular material properties of the object;
$\vec{N}$ is the normal vector;
$\vec{R_i}$ is the reflection vector (see figure 3.8) for light source i;
$\vec{L_i}$ is the light vector for light source i defined by the intersected point and the light source i position;
$\vec{V}$ is the view vector defined by the intersected point and the observer position;
n is the rugosity coefficient of the object;
m is the number of light sources.

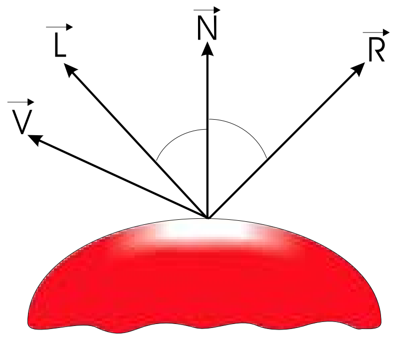

FIG. 3.8. *Representation of the vectors needed in the Phong illumination model.*

When computing the normal at each point of a surface to estimate the color of this point in this framework, one is using Phong Shading. If the normal and the color at each vertex of a facet are computed, and then these colors are interpolated, one is using Gouraud Shading[26] (this technique is often included in 3D graphic hardware and is faster to compute because the illumination model is not applied at each point).

Local illumination models do not take into account diffuse interreflections or specific photometric properties of materials such as mirrors, glasses or fires. Therefore, we need to introduce a more complicated illumination model like the Radiosity approach, which is a global illumination model.

3.3.2. Global Illumination: Radiosity and Ray Tracing. We distinguish two methods: the radiosity[25] technique and the ray tracing one[3, 47, 30]. A combination of these two techniques gives more realistic images. The radiosity technique is dedicated to diffuse interreflections while ray tracing is efficient to process specular surfaces.

The radiosity equation 3.2 is directly derived from physics[39]. It consists of a simulation of the energy transfers that occur in a scene. This simulation computes the energy balance when surfaces are shooting and receiving their energy in the 3D space. As all objects are purely lambertian, the energy sent from one facet another is simply the product of the energy density (radiosity) of this facet multiplied by the solid angle from which the other facet is seen from the first (it explains the existence of form factors F_{ij} in equation 3.2.

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j \, F_{ij} \tag{3.2}$$

where:
B_i is the radiosity of surface i
B_j is the radiosity of surface j
F_{ij} is the form factor between facets i and j with the reciprocity relation: $A_i F_{ij} = A_j F_{ji}$ (A_i and A_j are the facet i and j surface areas)
F_{ij} is equal to $\frac{1}{\pi} \times \phi_{ij}$ with ϕ_{ij}, the solid angle from which facet i sees facet j.
ρ_i is the reflectivity of surface i. It can be related to a complex reflection function like the bidirectional reflection distribution function (BRDF).

This equation tells that the radiosity B_i of facet i is equal to its internal energy density (emittance E_i which is non-zero only if it a light source) plus ρ_i times the energy coming from all facets j (where $(1 - \rho_i)$ times this coming energy is simply transformed into heat by object i). The radiosity equation is not hard to solve, but it is very CPU and memory demanding. Indeed, to solve it, it is necessary to compute the form factors or configuration factors[39]. This is discussed later in this section.

A plethora of resolution methods[10] exist to solve the radiosity equation like Southwell, Gauss-Seidel, Jacobi Iteration. One of the most common techniques is Progressive Radiosity[9], which consists in sending the energy starting from the patches (surfaces) that have the biggest radiosity. This considerably speeds up the convergence to the solution, and some other acceleration techniques can be added too (Ambient term, Positive Overshooting). Progressive Radiosity computes only one row of form factors by iteration (the form factors F_{ij} related to the facet i considered), and produces an illumination iteration where only all previously considered facets propagate their energy in the 3D scene. Each iteration can be displayed to see convergence (see pseudo-algorithm in figure 3.9). Of course, of lot of other methods exist to solve linear systems and what we cited is not an exhaustive list of all possible techniques.

```
For all patch i in the scene
    {
    $B_i = E_i$ ; // radiosity initialization to surface emittance
    $\delta B_i = E_i$ ; // initialization of the radiosity to send
    }
While (radiosity to send > threshold)
    {
    // a patch is selected to shoot its energy through the scene
    Find Patch i with largest $\delta B_i \times A_i$ ;
    // a patch is subdivided into smaller parts called elements for
    // energy gathering
    For each element $j$;
        {
        Compute one row of form factors $Fij$ ;
        Compute reciprocical $F_{ji}$ using $A_i F_{ij} = A_j F_{ji}$ relation;
        $\delta B = \delta B_i \times \rho_j F_{ji}$ ;
        $\delta B_j += \delta B$ ;
        $B_j += \delta B$ ;
        }
    $\delta B_i = 0$ ;
    For each element $j$
        {
        // (Rescaling is used for visualization purposes)
        Rescale radiosity of element $j$: $Bd_j = \frac{B_j}{max_{\forall \texttt{elements}_k}(B_k)}$ ;
        }
    For  tt each vertex $l$ in the scene
        {
        Compute radiosity $B_l$ as the average of all rescaled
        radiosities $Bd_k$ of elements $k$ which are connected to vertex $l$.
        }
    Linearly interpolate vertex radiosities using Gouraud Shading
    to display image.
    }
```

FIG. 3.9. *Progressive Radiosity pseudo-algorithm.*

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{cos\theta_i cos\theta_j}{\pi r^2} V_{ij} dA_j dA_i \tag{3.3}$$

where V_{ij} is the visibility term (0 or 1) between facet i and j.

Form factors computation using equation 3.3 (see figure 3.10) is the hardest task in the radiosity process[34]. There are a lot of techniques to evaluate the form factors: hemicube[8], proxel plane[40], Monte Carlo ray tracing[36] are the most famous. Some form factors can be analytically estimated for sufaces in specific positions, e.g. if they are perpendicular or parallel.

In all cases, one needs to take into account occlusions: that is to say, it is possible that another surface blocks the energy leaving a surface, by intercepting it. In the form factor determination, the visibility term (V_{ij} term in equation 3.3) represents the major part of the total CPU load. If a hemicube is used (a half cube is placed on the surface i to simulate the fact that it sends energy in the half-space), one can use classical hidden face removal techniques such as Z-Buffer or A-Buffer, to avoid

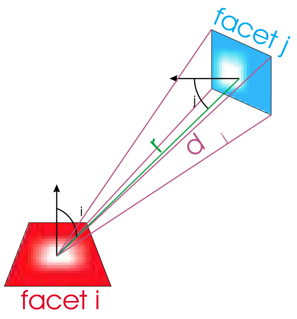

FIG. 3.10. *Form factor geometry: form factor F_{ij} is the fraction of energy leaving element i and reaching element j (see equation 3.3).*

form factor aliasing. This is the technique used by *Phoenix*, a rendering software developed at INRIA(see figure 3.11). It is interesting to note that, since graphics hardware got Z-Buffering, especially on the powerful SGI stations, this form factor computation is greatly accelerated[4]. In *Phoenix*, some form factors are stored to avoid their recomputation in the reflectance recovery process, decribed later. The combination of A-Buffer and hardware Z-Buffering gives speed and enough realism, as shown by the rerendering result images of figure 4.10.

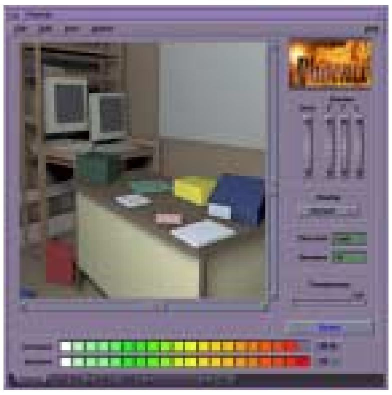

FIG. 3.11. *The Phoenix photorealistic rendering software.*

Ray Tracing could be used with well-known optimizations such as octrees[29], shaft-culling[28]. The rays are sent from a surface through a hemisphere centered on it, and its sampling can be non-uniform. All the rays reach other surfaces in the scene and tell what are the elements seen from the shooting surface.

To finish with, some recent developments led us to the concept of visibility complex[15, 16]: a structure that stores all the geometric relations between surfaces. This technique is very powerful but could require a lot of memory space.

Radiosity is very efficient for perfect diffuse interreflections simulations. To take into account specular surfaces such as mirrors for example, a combination with Ray Tracing techniques is necessary. Therefore, to compute photorealistic images it is important to develop a hybrid method that integrates both Radiosity and Ray tracing techniques. *Phoenix* integrates all these techniques using all possible optimizations to perform photorealistic rendering, as fast as it can. *Phoenix* is a powerful computer graphics software used to develop image-based rendering algorithms as described in the next section.

4. Photometry Analysis.

4.1. Introduction. We return to the problem of a 3D scene photometry determination. The approach that we propose consists of using a photorealistic technique as a feedback which estimates the quality of the photometry obtained, but also provides some useful concepts necessary for the photometry analysis. To solve this photometry determination problem, we suppose that we know completely the 3D geometry of the scene to analyze and that we have at least one image of that scene.

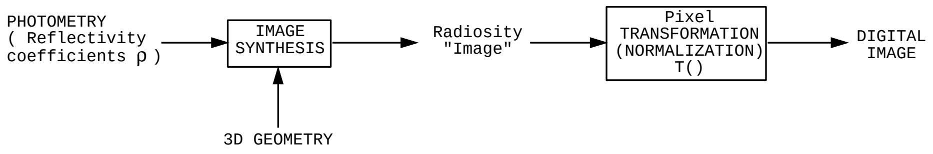

FIG. 4.1. *Graph of a computer graphics image generation.*

We now explain the details of our procedure. Realistic rendering algorithms use as inputs the 3D geometry and the photometry of the scene (see figure 4.1). For the photometry, most algorithms use simple lambertain models which have the property to approximate well real scenes in spite of their simplicity. Object surfaces are considered as non textured so that one reflectivity ρ is sufficient to completely model an object surface reflectance. Image synthesis solves the light energy balance equation 3.2 for all object facets. Then, the projection on the image plane of the light energy density obtained (that we call a radiosity "image", in $watt/sr/m^2$) is rendered. When texture is present it is usually added after synthesis as a modulation of the object radiosities. (R, G, B) values of each pixel have to be normalized (pixel transformation $T()$) so that we obtain values defined in the [0, 1] interval for visualization in a framebuffer of a computer. What happens in nature when we get an image of a scene from a camera ? The energy balance is achieved physically almost in real time when light is switched on in a scene. In fact, it is this stable state which is modeled in the rendering process. A camera simply projects the observation of the scene radiosities on its image plane and the (R, G, B) CCD matrices convert the radiosity energy in pixel values. So, to obtain a realistic image, the rendering technique must be such that the **pixel transformation $T()$ corresponds to the (R, G, B) CCD camera transfer function**. Usually this transfer function is not given by the camera manufacturers, so that it is unknown; nevertheless, we know that it is a non-linear monotonically increasing transfomation, so that it is invertible.

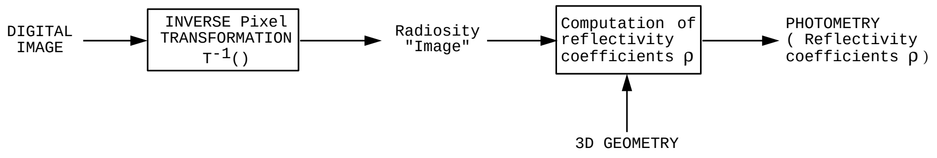

FIG. 4.2. *Graph of the analysis process.*

Consider now the photometry analysis problem. We propose to solve it using the reverse path of figure 4.1. The inputs of this problem are the digital image and the 3D geometry and we proceed as follows (see figure 4.2): we start from this original image and transform it with $T()^{-1}$ (that has to be computed from the transfomation $T()$), so that we obtain a radiosity image. With a simple backprojection of the radiosity image (using the 3D geometry), we get the true radiosity of the (scene) object surfaces. The advantage of the computation of radiosities is that equation 3.2, which relates the ρ reflectivities to the radiosities, is linear. It is then easy (we will see how below) to compute the ρ coefficients by inverting this equation. The problem is that we do not know $T()$, though we know that it is an increasing operator. The solution we propose is to use image synthesis as a feedback loop to the previous scheme of figure 4.2, which will stabilize this open-loop system (solving the very difficult photometry problem). The final solution we propose is thus summarized in figure 4.3. This scheme implies that we have to use the image synthesis path a certain number of times (which gives an iterative aspect to our procedure) and tune the photometry analysis until the error between the original image and the synthetic one becomes small.

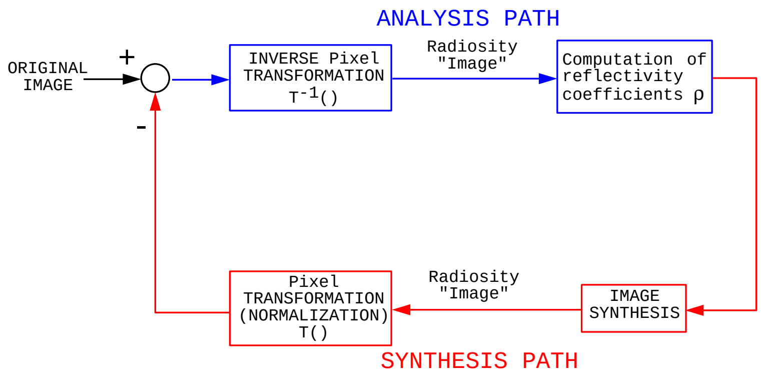
ANALYSIS PATH
ORIGINAL IMAGE
+
-
INVERSE Pixel TRANSFORMATION T-1()
Radiosity "Image"
Computation of reflectivity coefficients ρ
Pixel TRANSFORMATION (NORMALIZATION) T()
Radiosity "Image"
IMAGE SYNTHESIS
SYNTHESIS PATH

FIG. 4.3. *Graph of the full inverse rendering process.*

4.2. Previous works and overview of the problem. First work was presented by André Gagalowicz [20, 22] as a cooperation between computer vision and computer graphics techniques, known as Analysis/Synthesis. Gagalowicz's technique came from the idea that we can use a real image captured with a camera as a strong provider of information to regenerate a synthetic one. The dual idea was to use this synthetic image as a feedback to the real image, in order to test the validity of the vision tasks (geometric and photometric analysis of the scene). A first complete geometric reconstruction and photmometric analysis technique of a real scene was proposed in [23], leading to a first realistic synthesis of this scene. Here, we show an extension of this idea as an iterative (feedback) technique which uses the information of the original image and the 3D geometry of the scene to correct the generated image. This method approximates the surface reflectances, precisely enough to get a photorealistic image, using the previously described computer graphics algorithms. This new image synthesis is close in spirit to the original one, and can be exploited for other applications like augmented reality. This is discussed in the last section of this chapter.

Image-based rendering techniques have really been developed with the recent papers published by the Berkeley team [11, 12, 13, 48, 49]. These works propose important advances in image generation using real images. [48] describes a technique that can recover the bidirectional reflection distribution function (BRDF) over all surfaces inside a three dimensional scene, using a set of images and the 3D geometry of the scene. A limitation of this techique resides in the fact that they need a large set of images (40 images are selected from a set of 150 exposures) to compute these BRDFs.

In [18, 20, 23, 37], some interesting ideas are given to compute photorealistic images using one single image. These algorithms are based on the use of pixel intensities covered by the surface projection onto the image, to compute an approximation of their reflectances. Like [14, 31], our research uses this technique as an initialization process, but we use a feedback loop technique (see figure 4.3) instead of former techniques which are open-loop and are consequently less stable. We also add a new algorithm that can recover the perfect specular and the perfect diffuse surface reflectances and the radiosity-to-pixel conversion function, iterating if necessary by minimizing an error function between the original image and the synthetic one. This technique has been enhanced since [5], including the global illumination algorithm that needs to be very fast because of rerendering iterations (we compute several radiosity images to get the final result). The rendering software *Phoenix* is based on the technique described in the previous section, but also has this rerendering functionality.

The photometry problem is subdivided in two parts: the surface reflectance recovery and the estimation of the radiosity-to-pixel conversion function. The solution of the first problem will be discussed in a following section. As it is a linear problem, the reader will recognise that it should be easy to solve. Let us begin by considering the second problem. Several methods are possible supposing that this function is of a known type (γ correction function for example) or unknown. The next section presents results when we suppose that this camera transfer function is completely unknown.

It is important to note that, in the following, objects which have the same photometric properties are put together in *groups*. This gives us the opportunity to compute the radiosity for the surfaces that may not be directly seen in the real image. All objects are subdivided into *surfaces* or *facets*.

4.3. Unknown camera transfer function.

4.3.1. Linear case. We suppose that the transformation function $T()$ is linear. Practically, due to the lack of physical data (coming from the camera), radiosity values can be only evaluated modulo a scaling factor. In the linear case, pixel intensity values can be considered as radiosity values. Therefore, the photometry is directly coming from the solution of the first problem, and from the direct use of pixel intensities.

The left image of figure 4.4 shows a reference synthetic image produced by *Phoenix*. The linear solution consists of computing the reflectivity coefficients ρ, feeding them to the rendering software and simply visualizing the radiosity image. This result is presented on the middle image of figure 4.4. We can see immediately that the rerendered image is very different from the original one, which proves that $T()$ cannot be approximated in a linear way.

To investigate the linear case further, we also decided to compute the minimum and maximum pixel intensity values for each group in the original image (in the (R,G,B) channels) and do the same with the radiosity image. Finally, we apply a linear transformation for each group which rescales the $[min, max]$ interval of the synthetic image groups to their equivalents in the real image. The result is presented on the right image of figure 4.4. The result is more similar to the reference image but still not satisfactory, even though the reference (synthetic) image is very simple.

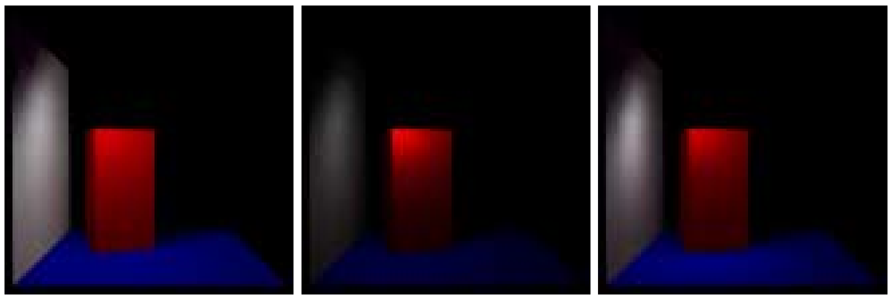

FIG. 4.4. *Linear solution for camera transfer function.*

4.3.2. Non-linear case: Function approximation using histogram modification. Considering that the radiosity image gives the stucture of the image, and the real one is simply due to the camera transfer function, it is natural to propose for $T()$ the transformation which maps the histogram of the radiosity image to the histogram of the reference one. If these two images have the same histogram and the same structure, they should be similar.

If E_1 and E_2 are the cumulative distribution function(CDF)[24] of the radiosity and the real images respectively, we know that the researched $T()$ transform is equal to $E_1 \circ E_2^{-1}$.

A pixel intensity is read on the $\vec{x}$ axis from E_1 and its corresponding value on the $\vec{y}$ axis is picked. This last value is positioned on the $\vec{y}$ axis of E_2; we then read its corresponding value on the $\vec{x}$ axis of E_2 to be used as the new pixel intensity for the rerendered image.

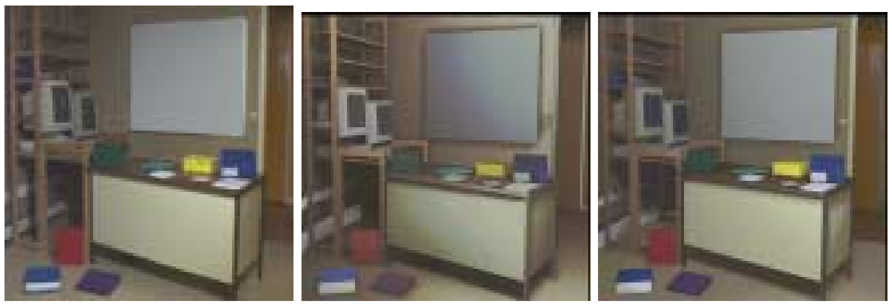

FIG. 4.5. *Radiosity-to-pixel conversion function estimation by histogram modification on all image (one single function is computed). The left image is the original. The middle image is the regenerated one, after ρ convergence and after a CDF transformation has been applied. The right image corresponds to the case where the histogram modification was perfomed separately for each group.*

The results using this technique are shown in figure 4.5. The left image is a reference (natural) image. The middle image is the rerendering result using histogram modification for $T()$. Notice that the corrected image by histogram modification is far from the original. This simply means that the conversion function cannot be approximated by histogram modification. That was a very surprising result to us as the histogram modification technique that we implemented, guaranteed that the both (synthetic and real) histograms are almost equal. We also know that the $T()$ transform used is the only one which can perform this histogram modification.

As in the linear case, we implemented the histogram modification at the group level too. The result can be seen on the right image of figure 4.5. The rerendered image is much more appealing when compared to the reference one. Some problems appear in the right panel of the desk and the upper-right corner of the white blackboard. Even though it is an interesting result, the group transformation cannot be interpreted as a real camera transfer function. We have to look for a better solution.

In the next section, we show that the best solution to approximate the camera transfer function is to use a γ correction function.

4.4. Camera transfer function modeled by γ correction function (non-linear). A simple but efficient idea is to approximate the radiosity-to-pixel conversion function by γ correction: $T(B) = B^{\frac{1}{\gamma}}$. It has been inspired by Tumblin and Rushmeier[42] who present some results estimating a conversion function for a better display of the final image computed by the radiosity technique. They propose to use 2.2 as γ value. Nevertheless, we have not limited our algorithm to the research of γ inside a restricted interval: our technique finds the best γ value by minimizing the error between the original image and the rerendered one as a function of γ only (all reflectivity coefficients are fixed). The best γ value for our images we obtained is 2.0.

First of all, γ is initialized to 2.2 (see [42]). The algorithms begins to refine the best possible ρ that create an image synthesis with the smallest possible difference from the original image (see next section to understand how the reflectances are recovered). When all ρ have been recovered, the γ correction function is used again to transform the radiosities into pixel intensities. The total difference between the original image and the synthetic one (see equation 4.1) is minimized with respect to γ and gives the best possible rerendered image. A classical gradient method is used to minimize this error. As the gradient of the mean square error between the pixel values and $T(B_\lambda)$ with respect to γ can be computed analytically (equation 4.2), we use a simple gradient descent procedure for optimization. A conjugate gradient technique[38] is also very effective as the error function is convex with respect to γ. Let us denote the error Err_λ for all (i,j) pixel locations and for each λ channel by:

$$Err_\lambda(\gamma) = \sum_{i,j} (I_\lambda(i,j) - (B_\lambda(i,j))^\gamma)^2 \qquad (4.1)$$

The derivative of $Err_\lambda(\gamma)$ with respect to γ is equal to:

$$\frac{dErr_\lambda(\gamma)}{d\gamma} = -2 \times \sum_{i,j} (I_\lambda(i,j) - (B_\lambda(i,j))^\gamma) \times (B_\lambda(i,j)^\gamma \times Log(B_\lambda(i,j))) \qquad (4.2)$$

A result image using this method is shown in figure 4.6, with the reference image on the left and rendered image on the right, corresponding to the best (found) approximation.

4.5. Reflectance estimation and iterative correction. We first investigate the lambertian case.

4.5.1. Case of perfect lambertian surfaces. As explained in section 4.4, the radiosity-to-pixel conversion function used for our technique is a γ correction function with $\gamma = 2.2$. When the reflectances are recovered, we minimize the error between the two images to find an optimum γ as discussed above.

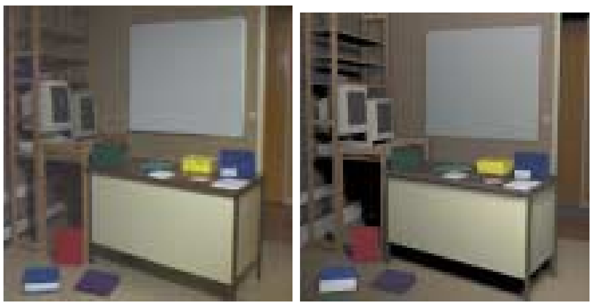

FIG. 4.6. *Natural image(left) and rerendered image(right) with an optimum γ determination.*

The problem to solve is to invert the radiosity equation 3.2 in section 3.3, where we know the radiosities of all facets and where we have to find the reflectivity coefficients of these facets and the emittance value of the light sources. When there is more than one light source, the problem is theoretically underdetermined, as there are more unknowns than equations. Practically, we restrict this presentation to the case of one light source. In this case, each equation of type 3.2 is such that ρ and E always appears together via the product $\rho \times E$. The reflectances can only be evaluated modulo a scaling factor, so that we can always suppose that $E = 1.0$ for the light source.

Nevertheless, we have shown in [23] that it is possible to solve the problem of having many light sources in the scene, and that the problem is strongly overdetermined in this case. It comes from the fact that there are many equations of a type $\rho_i = \rho_j$ for facets i and j made of the same material.

Solving equation 3.2 is much more complicated than it may look, because we need the radiosities of all facets of the scene and all form factors F_{ij} which is both tedious and in practice quite impossible to obtain. Therefore we had to implement a non natural but efficient method.

Principle. The principle of our rendering technique is to refine the computation of the surface reflectances ρ from a real scene, given its 3D model, the light source position, the camera(without the transfer function) and an image of this scene.
The technique consists in reading the image and projecting all 3D surfaces on the real image to compute a first approximation of ρ values. All facets are then rendered with these photometric parameters and an approximation of the transfer function (i.e. γ correction function which is supposed to be fixed) using the fast rendering software *Phoenix* and described above. We can now compute the error between the original image and this synthetic one, group by group. This error is minimized to obtain the best possible ρ with the initial constraints.

ρ extraction from real image. In the process of recovering the ρ values, one has to extract pixel intensities from the original image to compute ρ for a group.

We propose, like Drettakis[14], to compute offscreen index buffers using hardware Z-Buffering, except that we apply some filters to the index buffer to avoid error computations. Since we have the 3D geometry of the scene and the camera properties, we can compute an index buffer which contains the number of the visible group in the real image. A hidden-surface buffer (left image of figure 4.7) is computed too, and is used to remove edge matching problems which may occur when the 3D model of the scene is not perfectly positioned on the real image: a classical edge detector tool in combination with the two previous buffers is employed to suppress object boundaries and to obtain the final buffer index image (right image of figure 4.7).

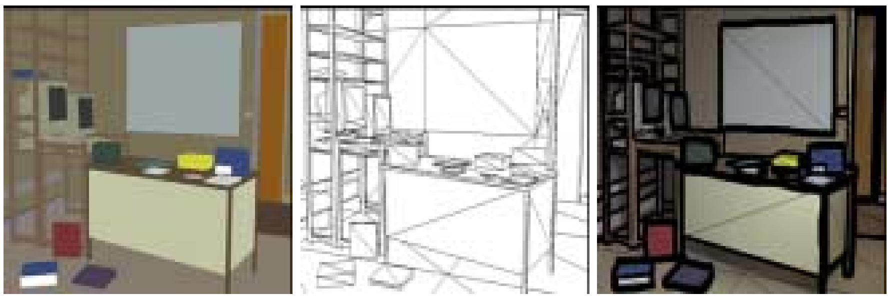

FIG. 4.7. *Left image: index buffers with a specific color by group. Middle image: hidden-surface image. Right image: combination of the the left and middle image, adding an edge detector to remove positioning problems of the 3D model.*

Furthermore, a planar approximation is performed on each facet to avoid local illumination problems or noise produced by the camera. We only keep half all the pixels covered by a surface to get its ρ. Image 4.8 gives the pixels necessary to compute the final averaging.

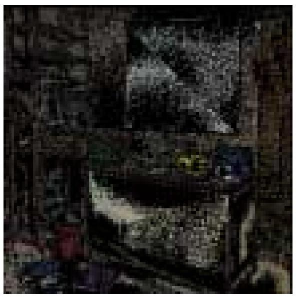

FIG. 4.8. *Filtered pixels in original image using a planar approximation. Only non-black ones are kept for ρ computation.*

Finally, the combination of the filtered image (figure 4.8) and the right image of figure 4.7(right) gives the pixels used for the computation of the group reflectances. This buffer is computed once and used at every iteration of the ρ computation.

ρ Reflectances initialization. As we know the transformation $T()$, the radiosity associated to each pixel is determined immediately. All the ρ_i reflectances of a surface i are initialized by the pixel average of the image area obtained by the projection, on the original image, of all object surfaces belonging to the group i (see equation 4.3). This means that the ρ values are computed for an object group and then, propagated to the object surfaces.

$$\rho_i = \frac{1}{n}\sum_{j=1}^{n}(P_{i,j})^{\gamma} \tag{4.3}$$

FIG. 4.9. *Average ρ estimation for a group.*

Error computation and ρ correction. We can obtain a new image using the previously described technique (initialization phase for ρ). It is compared to the original one by computing an object error estimate ε:

$$\varepsilon_j = \frac{\widehat{P_{org_j}}}{\widehat{P_{new_j}}}) \tag{4.4}$$

where $\widehat{P_{org_j}}$ is the mean of the filtered pixels covered by the projection of object j in the real image and $\widehat{P_{new_j}}$ is the mean of the filtered pixels covered by the projection of the same object in the synthetic image.

As ρ_j and $\hat{B}_j$ the average radiosity of object j are proportional (see equ. 3.2), the readjustment of ρ_j is such that $\frac{\rho_{j_{k+1}}}{\rho_{j_k}} = \frac{\widehat{B_{org_j}}}{\widehat{B_{j_k}}}$, where ρ_{j_k} is the reflectivity of object j at iteration k and $\widehat{B_{j_k}}$ is the average radiosity of object j at iteration k.

In order to suppress biases created by small objects having important errors, it is possible to correct the ρ_i value for the group i in the next iteration by:

$$\rho_{di_{k+1}} = \rho_{di_k} \times \varepsilon_i = \rho_{di_k} \times \frac{\sum_{j=1}^{n_i} f(\varepsilon_j)(\varepsilon_j \times m_j)}{\underbrace{\sum_{j=1}^{n_i} f(\varepsilon_j) m_j}_{\neq 0}} \tag{4.5}$$

$$\text{with } f(\varepsilon_j) = \begin{cases} 0 \text{ if } \varepsilon_j \geq 2\min_{n_i}(\varepsilon_j) \cdot \left[1 + \underbrace{\frac{max_{n_i}(\varepsilon_j) - min_{n_i}(\varepsilon_j)}{max_{n_i}(\varepsilon_j) + min_{n_i}(\varepsilon_j)}}_{\text{spread criteria}} \right]; \\ 1 \text{ otherwise.} \end{cases} \tag{4.6}$$

k is the iteration number;
$\widehat{K_i}$ the total error between the synthetic image and the original one, for group i;
$\widehat{K_j}$ the total error between the synthetic image and the original one, for object j;
n_i the number of objects for group i;
K_j, the error for object j;
$\max_{n_i}(K_j)$ the maximum error for all n objects;
$\min_{n_i}(K_j)$ the minimum error for all n objects;
m_j the number of pixels in the projection of object j.

This correction gives us new ρ values allowing to compute a new image synthesis. The new obtained image is compared, by the same process to the original one. The algorithm iterates the ρ correction until a user-defined threshold has been reached. The right image of figure 4.10 has been computed using this technique.

Case of perfect specular surfaces. We now present an extension of this method to the scenes which contain some perfect specular surfaces (perfect mirrors). These perfect specular surfaces have a specular reflectance ρ_s initialized to 1.0 and no diffuse reflectance($\rho_d = 0$). During the pure rendering process, if a specular surface is

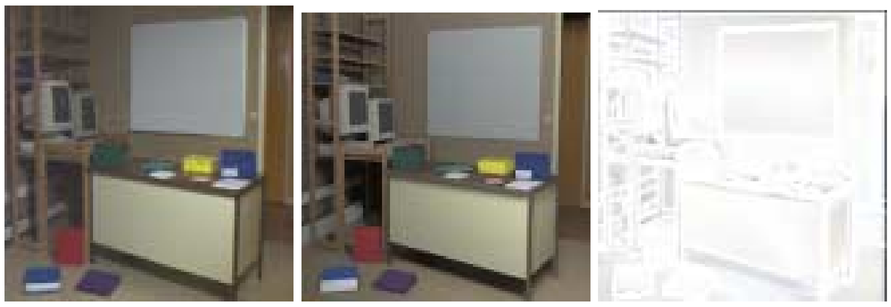

Fig. 4.10. *Left image is the original one. Middle image is the final result of the ρ convergence after only three rerendering iterations. Right image is the difference between the real and the synthetic image, where the darkest areas correspond to the largest error areas.*

reached, then the received energy is redistributed following the same solid angle as the incident one, using Descartes'law. This behaviour is different from perfect lambertian surfaces which send a constant energy in all directions of the space.

If we apply the former lambertian analysis to such a scene, perfect diffuse surfaces will be rather well reconstructed but it is not the case of the specular ones, which produce important errors all over the surface. This situation is detected automatically comparing once again the original image with its lambertian synthesis. Then, the algorithm treats these mirrors as perfect specular reflectors (the ρ_s of these surfaces is then equal to 1.0 and $\rho_d = 0$).

At this stage of the method, it is important to note that the specularity property of the surface gives us an important feature: considering the virtual viewpoint which is the symmetrical of the real viewpoint with respect to the mirror plane, the mirror facet becomes a virtual screen (not a rectangle in general) where the 3D scene is projected. Here we suppose that the part of the 3D scene seen from this virtual viewpoint (and restricted to the specularity volume) is lambertian (if it is not the case, our procedure may produce errors). From here, we have a screen image, a 3D model of a purely lambertian sub-scene; so we can apply the former lambertian procedure described above. In fact, we still use the index buffers to recover the index values, but we chose to use ray-tracing to compute them. This gives us the possibility to analyze the reflectance of the diffuse objects that are projected onto the mirror. Following the properties of the surface reached by the reflected ray, we compute its diffuse properties by averaging the pixels it covers on the mirror (or we could send a ray again if it is a specular surface too). This technique has been used to generate the middle image of figure 4.11.

Case of colored specular surfaces. The case of colored specular surfaces is very close to that of perfect ones, except that we allow for ρ_s to be different from 1.0 for all wavelengths. The idea is to compute the error in a manner similar to the perfect diffuse case. This error is then used to correct the ρ_s of the specular surface, instead of the ρ_d for the diffuse ones.

A new image synthesis is then computed using the new ρ_s and the global error on the specular objects is reestimated again. If this error is still large, we consider that this surface presents more complex photometric properties (partially diffuse and specular, including complex BRDF like isotropic and anisotropic ones); we call these surfaces glossy.

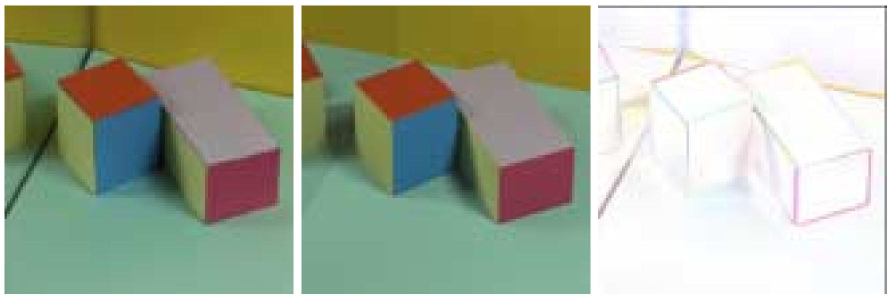

FIG. 4.11. *The original image is on the left. The middle one is the result of the ρ convergence after only 3 rerendering iterations with specular processing. The right image is the difference between the real and the synthetic image, where the darkest areas correspond to the biggest errors.*

Case of glossy surfaces. This is clearly the most complicated case. In the glossy case, we need to use a more complicated technique to recover photometric properties. To simulate these particular reflections, we use a *Ward* reflection model [44], because it incorporates anisotropic reflection properties, which are not simulated by the Phong model. The procedure also consists in directly recovering the anisotropic direction from real images. We then minimize the error between the original surface pixels and those produced by the *Ward* model as a function of α_x and α_y (rugosity coefficients in *Ward* model). This cases will be described in more details in future publications.

Use of rerendered image in applications. Our image-based rendering method has found a lot of possible applications, especially in augmented reality (see images 4.12), post-production software and web merchandising, because of its simple technique and its small number of required data. Indeed, it gives the opportunity to only use a 3D geometric model and one single image on-line, no BRDFs or texture maps are needed. *Phoenix* does the rest, recovering reflectance parameters and regenerating a new synthesis image very close to the original one, and with all its advantages (viewpoint changes, adding objects, etc.).

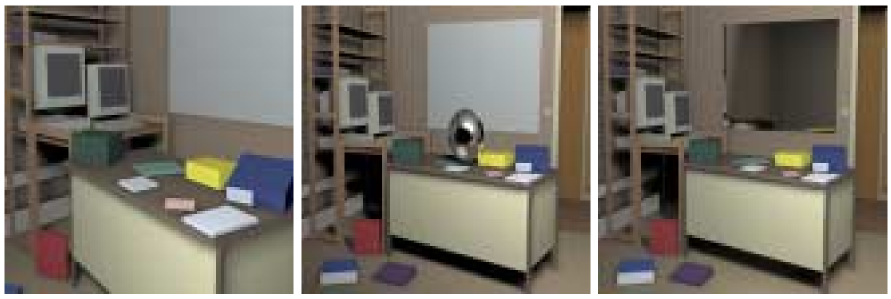

FIG. 4.12. *Some augmented reality scenes using Phoenix with viewpoint change (left), adding ellipsoidal specular object (middle) and with new specular properties for the white blackboard (right).*

5. Conclusion. We presented an original image-based rendering technique which is very fast and which generates photorealistic images, using a restricted set of data. It does not yet take into account glossy surfaces, which are still under study. It would be interesting to extend future research to the case of textured surfaces or of participating media.

REFERENCES

[1] A. Ackah-Miezan and A. Gagalowicz, *Energy Minimizing Segmentation of an image*, in proceedings of International Symposium on Computer and Information Science, ISCIS VII, Antalya, Turkey, 1992, pages 631-634.

[2] A. Ackah-Miezan and A. Gagalowicz, *Discreet Models for Energy Minimizing Segmentation*, in proceedings of the 3rd International Conference on Computer Vision, Berlin, BRD, May 1993, pages 200-207.

[3] A. Appel, *Some Techniques for Shading Machine Renderings of Solids*, in proceedings of the Spring Joint Computer Conference, 1968, pages 37-45.

[4] D.R. Baum and J.M. Winget, *Real time radiosity through parallel processing and hardware acceleration*, in Computer Graphics (1990 Symposium on Interactive 3D Graphics) 24(2), March 1990, pages 67-75.

[5] S. Boivin and L. Doghman, *A rendering method for the realistic simulation of natural scenes*, in proceedings of IMAGE'COM 96, Arcachon, France, 1996, pages 302-307.

[6] L. Carpenter, *The A-Buffer, an antialiased hidden surface method*, in proceedings of ACM SIGGRAPH 84, 18(3), July 1984, pages 103-108.

[7] E. Catmull, *A Subdivision Algorithm for Computer Display of Curved Surfaces*, Ph.D. Thesis, Report UTEC-CSc-74-133, Computer Science Department, University of Utah, Salt Lake City, UT, December 1974.

[8] M. Cohen and D.P. Greenberg, *The hemi-cube: A radiosity solution for complex environmemts*, in proceedings of ACM SIGGRAPH 95, 19(3), August 1985, pages 31-40.

[9] M. Cohen, S.E. Chen, J.R. Wallace and D.P. Greenberg, *A progressive refinement approach for fast radiosity image generation*, in proceedings of ACM SIGGRAPH 88, 1988, pages 75-84.

[10] M. Cohen, J.T Wallace, P. Hanrahan and D.P. Greenberg, *Radiosity and Realistic Image Synthesis*, Book published by Academic Press, 1993, pages 109-130.

[11] P.E. Debevec, C.J. Taylor and J. Malik, *Modeling and Rendering architecture from photographs: a hybrid geometry and image-based approach*, in proceedings of ACM SIGGRAPH 96, 1996, pages 11-20.

[12] P.E. Debevec and J. Malik, Recovering High Dynamic Range Radiance Maps from Photographs, in proceedings of ACM SIGGRAPH 97, 1997, pages 369-378.

[13] P. Debevec, *Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography*, in proceedings of ACM SIGGRAPH 98, 1998, 199-206.

[14] G. Drettakis, L. Robert and S. Bougnoux, *Interactive Common Illumination for Computer Augmented Reality*, Rendering Techniques '97, J. Dorsey, P.Slusallek (eds), Springer-Verlag, Wien (Proc. 8th Eurographics workshop on Rendering, Saint-Etienne, France), 1997, pages 45-56.

[15] F. Durand, *Visibilité tridimensionnelle: étude analytique et applications*, Thèse de l'université Joseph Fourier, Grenoble, Juillet 1999.

[16] F. Durand, G.Drettakis and C. Puech, *The visibility skeleton: a powerful and efficient multi-purpose global visibil= ity tool*, in proceedings of ACM SIGGRAPH 97, 31(3A), August 1997, pages = 89-100.

[17] O. Faugeras and M. Hebert, *The representation recognition and locating of 3D objects*, in International Journal of Robotics Researc, 5(3), 1986.

[18] A. Fournier, A.S. Gunawan and Chris Romanzin, *Common illumination between real and computer generated scenes*, in proceedings Graphics Interface'93, Morgan Kaufmann publishers, 1993, pages 254-263.

[19] H. Fuchs, Z.M. Kedem and B.F. Naylor, *On Visible Surface Generation by a Priori Tree Structures*, in proceedings of ACM SIGGRAPH 80, 1980, pages 124-133.

[20] A. Gagalowicz, *Cooperative Computer Vision and Computer Graphics*, Invited Paper, in proceedings of the AFCET conference, Paris, 1989, pages 1727-1758.

[21] A. Gagalowicz, L. Vinet, *Region matching for stereo pairs*, in proceedings of the sixth Scandinavian Conference on Image Analysis, Oslo, 1989.

[22] A. Gagalowicz, *Collaboration between Computer Vision and Computer Graphics*, in proceedings of the ICCV'90 Conference, Osaka, Japan, 1990.

[23] A. Gagalowicz, *Modeling Complex indoor scenes using an analysis/synthesis framework*, Chapter of the book Data Visualisation, Rosenblum editor, published by Academic Press, 1994.

[24] R.C. Gonzales and R.E. Woods, *Digital Image Processing*, Book published by Addison Wesley, 1992.

[25] C.M. Goral, K.E. Torrance, D.P. Greenberg amd B. Bataille, *Modeling the Interaction of light between diffuse surfaces*, in proceedings of ACM SIGGRAPH 84, 18(3), July 1984, pages 212-222.

[26] H. Gouraud, *Continuous Shading of Curved Surfaces*, IEEE Transactions on Computers, C-20(6), June 1971, pages 623-629.

[27] E. Grimson, *Object Recognition by Computer*, Book published by MIT Press, 1990.

[28] E. Haines and J. Wallace, *Shaft culling for efficient ray-traced radiosity*, in proceedings of the Second Eurographics Workshop on Rendering, Barcelona, Spain, 1991.

[29] G.M. Hunter, *Efficient Computation and Datda Structures for Graphics*, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Princeton University, Princeton, NJ, 1978.

[30] D.C. Kay, *Transparency for Computer Synthesized Images*, M.S. Thesis, Progran of Computer Graphics, Cornell University, Ithaca, NY, January 1979.

[31] C. Loscos, M. Frasson, G. Drettakis, B. Walter, X. Granier and P. Poulin, *Interactive Virtual Relighting and Remodeling of Real Scenes*, Rendering Techniques '99, G. Larson, D. Lischinski (eds), Springer-Verlag, Wien (Proc. 10th Eurographics workshop on Rendering, Granada, Spain), 1999.

[32] V. Mehas-Yedid, J.P. Tarel and A. Gagalowicz, *Calibration métrique faible et construction interactive de modèles 3D de scènes*, in proceedings of RFIA, Paris, France, January 1994, pages 121-133.

[33] M.E. Newell R.G. Newell and T.L. Sancha, *A Solution to the Hidden Surface Problem*, in proceedings of the ACM National Conference, 1972, pages 443-470.

[34] A. Ng, *Techniques for Rapid Computation of Form Factors in Radiosity*, Technical Report No 680, Department of Computer Science, Queen Mary And Westfield College, 1994, University of London.

[35] B.T. Phong, *Illumination for computer generated pictures*, Communications of the ACM, 18(6), 1975, pages 311-317.

[36] M. Sbert, *An integral geometry based method for form-factor computation*, Computer Graphics Forum, 12(3), 1993, pages 409-420.

[37] V. Serfaty, A. Ackah-Miezan, E. Lutton and A. Gagalowicz, *Photometric Analysis as an aid to 3D Reconstruction of indoor scenes*, in proceedings of the IS and T/SPIE Symposium on Electronic Imaging: Science and Technology, 1993.

[38] J. Shewchuk, *An introduction to the conjugate gradient method without the agonizing pain*, Technical Report CMU-CS-TR-94-125, Carnegie Melon University, 1994.

[39] R. Siegal and H.R. Howell, *Thermal Radiation Heat Transfer*, Book published by Taylor and Francis, 1993.

[40] F. Sillion and C. Puech, *A general two-pass method integrating specular and diffuse reflection*, in proceedings of ACM SIGGRAPH 89, 23(3), July 1989, pages 335-344.

[41] J.P. Tarel and A. Gagalowicz, *Calibration de caméra à base d'ellipses*, in Traitement du Signal, 12(2), 1995, pages 177-187.

[42] J. Tumblin, and H. Rushmeier, *Tone Reproduction for Realistic Images*, in IEEE Computer Graphics and Applications, 13(6), November 1993, pages 42-48.

[43] J.M. Vezien and A. Gagalowicz, *Reconstruction 3D basée sur une analyse en régions d'une paire stéréoscopique*, in proceedings of the eighth RFIA Conference, Lyon, France, November 1991, pages 649-660.

[44] Gregory J. Ward, *Measuring and Modeling Anisotropic Reflection*, in proceedings of ACM SIGGRAPH 92, 1992, pages 265-272.

[45] J. Warnock, *A Hidden-Surface Algorithm for Computer Generated Half-Tone Pictures*, Technical Report TR 4-15, NTIS AD-753 671, Computer Science Department, University of Utah, Salt Lake City, UT, June 1969.

[46] H. Weghorst, G. Hooper and D.P. Greenberg, *Improved Computational Methods for Ray Tracing*, in ACM Transactions on Graphics, 3(1), January 1984, pages 52-69.

[47] T. Whitted, *An Improved Illumination Model for Shaded Display*, Communications of the ACM, 23(6), June 1980, pages 343-349.

[48] Y. Yu, P. Debevec, J. Malik and T. Hawkins, *Inverse Global Illumination: Recovering Reflectance Models of Real Scenes from Photographs*, in proceedings of ACM SIGGRAPH 99, 1999, pages 215-224.

[49] Y. Yu and J. Malik, *Recovering Photometric Properties of Architectural Scenes from Photographs*, in proceedings of ACM SIGGRAPH 98, 1998, pages 207-218.