Interactive Virtual Relighting and Remodeling of Real Scenes

Céline Loscos, Marie-Claude Frasson, George Drettakis, Bruce Walter, Xavier Granier, Pierre Poulin

Interactive Virtual Relighting and Remodeling of Real Scenes

Céline Loscos†, Marie-Claude Frasson†‡, George Drettakis†,
Bruce Walter†, Xavier Granier†, Pierre Poulin‡
†*i*MAGIS[1]-GRAVIR/IMAG-INRIA
B.P. 53, F-38041 Grenoble, Cedex 9, France
‡Département d'informatique et de recherche opérationnelle, Université de Montréal

Abstract. Lighting design is often tedious due to the required physical manipulation of real light sources and objects. As an alternative, we present an interactive system to *virtually* modify the lighting and geometry of scenes with both real and synthetic objects, including mixed real/virtual lighting and shadows.
In our method, real scene geometry is first approximately reconstructed from photographs. Additional images are taken from a single viewpoint with a real light in different positions to estimate reflectance. A filtering process is used to compensate for inaccuracies, and per image reflectances are averaged to generate an approximate reflectance image for the given viewpoint, removing shadows in the process. This estimate is used to initialise a global illumination hierarchical radiosity system, representing real-world secondary illumination; the system is optimized for interactive updates. Direct illumination from lights is calculated separately using ray-casting and a table for efficient reuse of data where appropriate.
Our system allows interactive modification of light emission and object positions, all with mixed real/virtual illumination effects. Real objects can also be virtually removed using texture-filling algorithms for reflectance estimation.

1 Introduction

Designing the illumination of real environments has always been a difficult task. Lighting design for home interiors for example, is a complex undertaking, requiring much time and effort with the manipulation of physical light sources, shades, reflectors, etc. to create the right ambiance. In addition other physical objects may need to be moved or otherwise changed. The problem is even more complex on movie sets or exterior lighting design. The fundamental trial-and-error nature of the relighting process makes it painful and often frustrating; more importantly, the requirements of constructing and moving real objects and light sources make testing many different potential designs often impossible.

Ideally, we would like to perform such processes entirely synthetically. The lighting designer would simply photograph the environment to be relit and/or remodeled, and then create the different conditions by computer simulation so that they can be evaluated appropriately.

Evidently, such a goal is very hard to accomplish. In this paper we provide first solutions to a subset of this goal, inspired by techniques developed for computer augmented reality, and common illumination between the real and the synthetic scenes [2, 10].

Our method starts with a preprocess, in which real geometry is reconstructed from a series of photos [20], taken from several different viewpoints. A second set of im-

[1] *i*MAGIS is joint project of CNRS, INPG, INRIA and Université Joseph Fourier.

ages (which we call *radiance images*) are taken from a *fixed* viewpoint with a real light source in different positions. The geometry and radiance images are used to extract an approximate reflectance at each pixel for the given point of view. Because reflectance is harder to estimate in shadowed regions, we try to have each visible surface point unshadowed in at least one image. We compensate for geometric and photometric imprecision by filtering and combining results from the individual radiance images. The result of this new approach is an acceptable estimate of reflectance, called a *reflectance image*; in the process, shadows are removed in a satisfactory manner.

Our main goal is to provide *interactive* manipulation of mixed real and virtual environments with common illumination. To achieve this we have separated the calculation of direct and indirect illumination. The reflectance image is used to initialise a hierarchical radiosity system with clustering [23], optimized for dynamic updates [6]. This structure is used for rapid updates of *indirect* light, while *direct* light is computed on a pixel-by-pixel basis. For direct light many components can be pre-computed and cached in a table for rapid use, and in other cases the changes are limited to small regions of screen space, permitting interactive updates. Working on a pixel-by-pixel basis results in high quality direct shadows and also facilitates the removal of real objects, since we can simply manipulate the reflectance image using texture generation methods.

It is important to note outright that we do not attempt to extract *accurate* reflectance values. The goal is to achieve *convincing* relighting at interactive rates. To this end we can ignore inaccuracies and small artifacts, if the overall effect is believable.

2 Previous work

A large body of literature exists in computer vision on reconstructing 3D scenes from photos [8]. However the quality of the extracted 3D models has only recently become satisfactory for computer graphics applications with the presentation of interactive systems such as *Photomodeler* [19], *REALISE* [9, 15], *Façade* [4], and others [20]. While they all include some form of texture extraction and mapping, none treat the extraction of surface properties and re-illumination. Sato *et al.*[21] present a system to extract 3D geometry, texture, and surface reflectance, but it is limited to controlled environments.

With the development of an ever increasing number of computer augmented reality applications, it becomes important to handle the common illumination between real and synthetic scenes. While some previous papers [10, 5] present preliminary solutions, they all require significant user intervention and are limited in different ways in the lighting or geometric conditions they can treat. Recent developments to *Façade* [2] include surfaces property extraction, but rendering times of the *Radiance* [24] system used for image generation are far from interactive.

Nakamae *et al.*[18] developed a solution for merging virtual objects into background photographs, and estimated the sun location to simulate common illumination effects in outdoor environments. More recently Yu and Malik [27] proposed a solution to virtually modify the illumination with different virtual positions of the sun in outdoor scenes.

Loscos and Drettakis [16, 17] have developed an approach to remove shadows, thus enabling synthetic relighting. This technique attempts to remove shadows by computing the best possible approximation using a single image. Despite successful results for certain cases, certain visual artifacts remain in the shadow regions.

In our method, as mentioned in the introduction, we separate direct lighting, which can be easily computed for each pixel, from indirect, or global lighting. Since we will be interactively modifying the scene, we need to be able to update the global illumina-

Fig. 1. The 7 radiance images used for the example presented in this paper.

tion rapidly. To do this, we have used some of the ideas developed by Shaw [22] and Drettakis and Sillion [6].

Removal of real objects from a reconstructed scene requires some form of hole-filling in the real images/textures containing the real objects being removed. Heeger and Bergen [13] have developed a method to synthesize texture images given a texture sample. They use a series of linear filters to analyse the sample and create a texture that matches the sample appearance. Their method is successful on "stochastic" textures (e.g., stucco) but fails on "deterministic" textures (e.g., bricks). El-Maraghi [7] has provided a public domain implementation of their algorithm.

Igehy and Pereira [14] integrate a composition step into the Heeger and Bergen algorithm in order to "erase" flaws (e.g., stains or undesired features) from images. They manually create a mask which indicates which part of the image is to be covered by the synthesized texture and which part keeps its original texture.

3 Overview of the Method

Our goal is to allow interactive synthetic relighting and remodeling of real environments including both removing real lights or objects, and adding virtual ones. To accomplish this, we need to build approximate geometric and reflectance models of the environment and quickly estimate the illumination in modified configurations. We also want our method to be tolerant of measurement and modeling errors in order to work on a broad class of environments. Our process consists of several preprocessing steps followed by an interactive relighting session.

We begin by taking two sets of photographs of the target environment. The first is taken from multiple viewpoints under normal lighting conditions and is used to build an approximate geometric model provided by our photomodeling system [20]. The second set is taken from the fixed viewpoint that will be used during the interactive editing session. These photos use controlled lighting that consists of a single known light source that is moved between photos. We typically use between 5 and 7 such photos (e.g., Fig. 1). This second set, which we will refer to as the *radiance images*, is used to estimate the reflectance on all the visible surfaces.

To recreate sharp shadows, the direct lighting is estimated on a per pixel basis from the fixed viewpoint using ray casting. For each pixel we store its corresponding 3D point and surface, its estimated local reflectance, and its visibility and form-factors to each light. This structure is illustrated in Fig. 2.

Each radiance image is used to estimate the reflectances at each pixel, but may be unreliable in some regions such as shadows. We generate a more robust reflectance

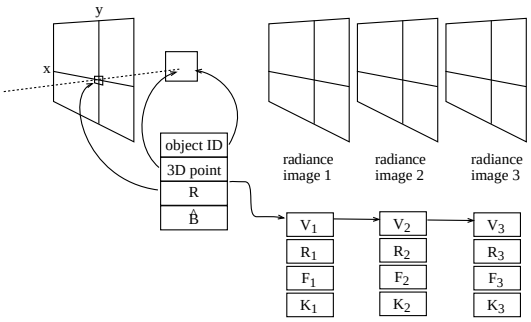

Fig. 2. A per pixel data structure is stored for the interactive view as well as for each radiance image. The visibility to each light V_i, the form-factor to each light F_i, the estimated reflectance at this pixel R_i, and the confidence level K_i of the pixel are stored for each radiance image i. The interactive view stores the merged reflectance R, the ambient term $\hat{B}$, the object's surface ID and the 3D point corresponding to each pixel.

estimator by assigning a confidence for each estimate and combining them from the multiple images accordingly. If we remove real objects, we also estimate the reflectance in regions of the image that become visible. This is accomplished by adapting a texture-filling algorithm.

Once the geometric and reflectance models are extracted, they are used to initialise an hierarchical radiosity system that enables dynamic simulation of the indirect lighting in the environment.

After completing these preprocessing steps, we are ready to interactively model and relight our scene. When we modify the lighting or the geometry of the scene (either real, virtual or both), we efficiently update direct and indirect light. The regions of the image for which direct illumination must be recomputed are efficiently identified in screen space using polygon ID maps and the shaft data structures used for dynamic global illumination. These same structures also allow efficient recomputation of indirect light.

4 Preprocessing

The main goal of the preprocessing steps is to initialise the data structures that will be used during the interactive session. First surface reflectance at each pixel is estimated, and a pixel-based data structure for precomputed direct lighting quantities is initialised. Finally the hierarchical radiosity system is set up for rapid indirect lighting updates.

The process begins by building a geometric model of the environment using our photomodeling system [20]. The user specifies a set of corresponding points in the set of photographs taken from multiple viewpoints. The system uses these to solve for the camera parameters and 3D positions of the points. The user connects these points together into polygons to form a geometric model of the scene and can specify additional constraints to improve the model. All further processing uses the radiance images, with the light source positions measured by the user.

4.1 Pixel Data Structure

The radiance images are all taken from the fixed viewpoint that we will use in our interactive remodeling session. The physical light source we used is a simple garden

light covered by white semi-transparent paper to achieve a more diffuse effect. Using a fixed viewpoint simplifies the capture of the real scene (since we need a small number of images); in addition working in image space allows more efficient data structures to be used for display, and generally simplifies the algorithms developed.

Much of the computation is done on a per pixel basis for this viewpoint using an augmented pixel data structure. At each pixel we store (see Fig. 2):

- The 3D point P which projects to the center of this pixel
- The polygon ID of the visible surface containing this point
- The form-factor F_i to each light source from this point
- The visibility V_i to each light source from this point
- The estimated surface reflectance R at this point

We create one such data structure for each radiance image plus an additional one for interactive use which also stores the indirect radiance $\hat{B}$ estimated by the radiosity system for this point. The radiance images additionally store a confidence K_i (≤ 1) at each pixel which indicates how reliable we think its reflectance estimate is.

The polygon ID and 3D point P are obtained by using an item buffer [25] and z-buffer depth values. The form-factor F_i is computed using a standard point-to-polygon technique [1]. The visibility V_i is the fraction of the light source which is visible from point P and is estimated by ray casting from the point to the light source. The number of rays is varied adaptively from 4 to 64, with the higher number being used in regions of penumbra. Initially, confidence K_i is set equal to V_i, since we have less confidence in regions in shadow.

4.2 Reflectance Recovery Algorithm

If we assume that our surfaces are diffuse then there is a simple relation between the radiance L seen by a pixel in the camera, the reflectance R at point P, and the incident light on point P given by:

$$L = R\left(\sum_i F_i V_i E_i + \hat{B}\right) \tag{1}$$

where E_i is the emittance of light i, $F_i V_i E_i$ is the direct illumination due to light i and $\hat{B}$ accounts for all indirect light. The emittance value is currently set arbitrarily, and an appropriate scaling factor applied to compensate during display.

If all the quantities in question were available and exact, we could solve exactly for the reflectance at each pixel using a radiance image i with its single light source via:

$$R_i = \frac{T^{-1}(C_i)}{F_i V_i E_i + \hat{B}} \tag{2}$$

where C_i is the pixel color recorded by the camera and $T()$ is the response function of the camera. This function was unavailable for our camera[2] so we have used a simple scaling factor, though it could be accurately estimated using the method of Debevec and Malik [3].

As a first approximation to indirect lighting $\hat{B}$, we have used an ambient term equal to the average image color times a user specified average reflectance [10]. The resulting reflectance gives satisfactory results for our test cases, although more involved indirect

[2] A Kodak DC260 digital camera.

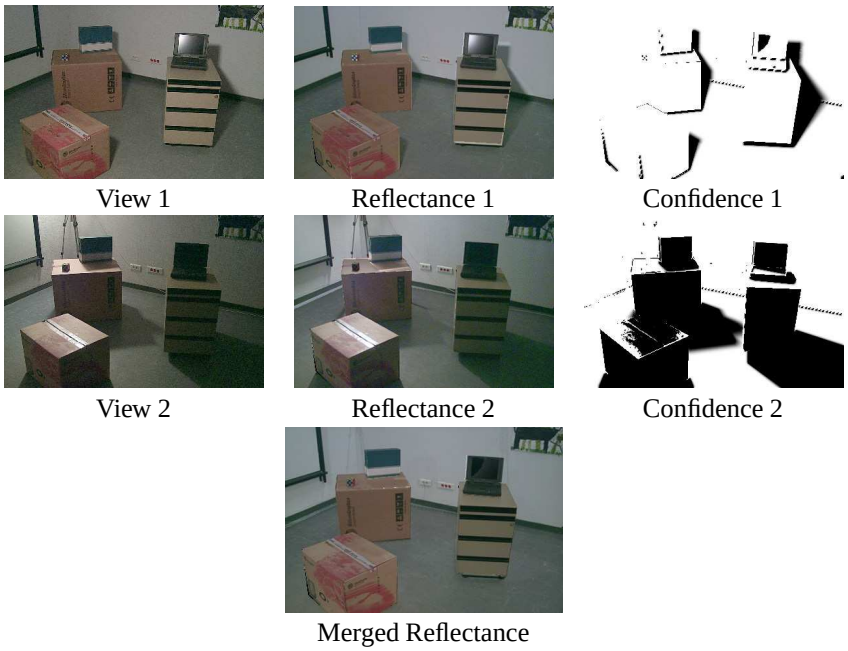

Fig. 3. Two of the seven radiance image views (left), the confidence images (right), and the resulting reflectance (center), extracted using Eq.(2). Dark values are for lower confidences. The merged reflectance is shown at the bottom.

lighting calculations may be necessary in other contexts when more accurate reflectance is needed. Some experiments were performed with an iterative approach to reflectance estimation using our radiosity solution, without much improvement in the reflectance estimate. Nonetheless, this is clearly a topic of future work.

Because of the many approximations in our system including the geometry, indirect light, and diffuse assumption, we know that our reflectance estimates will sometimes be quite inaccurate (e.g., in shadow regions where the indirect term dominates). We compensate for this by combining the reflectance estimates from multiple radiance images to form a much more robust reflectance estimator.

For each radiance image i, we also estimate our confidence K_i for each pixel reflectance estimate. The computation of K_i values is explained in next section. The merged pixel reflectance is formed by a weighted average of individual reflectance estimates:

$$R = \frac{\sum_{i=0}^{n} K_i \times R_i}{\sum K_i} \tag{3}$$

4.3 Filtering Confidence Values

As mentioned above, we initially set the confidence equal to the visibility V with respect to the light source, to reflect the fact that our reflectances are often inaccurate in shadow regions where indirect light dominates. However there are also other conditions that can cause inaccurate reflectance estimates including geometric error, specular highlights, saturation in the camera, and even the movable light source being visible in some images. We use a series of filters to try to identify and reduce the confidence in such problem regions.

Near shadow boundaries visibility V depends heavily on the exact geometry configuration and thus may be unreliable due to inaccuracies in our reconstructed model. To reflect this, we first expand low confidence regions using a 5×5 minimum filter where the pixels confidence is replaced by the minimum confidence in its neighborhood. Abrupt changes in the confidence can also cause objectionable artifacts in the combined results, therefore we next apply a 5×5 smoothing filter.

Lastly, to detect other problem regions, we apply an outlier filter. For each pixel, we compute the median of its high confidence reflectance estimates (e.g., those with $K_i > 0.75$) from the individual radiance images. Estimates which differ by more than a user supplied threshold from this median are assumed to be outliers and have their confidence set to zero. This allows to automatically detect and discount problem regions such as specular highlights and the light source tripod which is visible in some radiance images. Afterwards another smoothing filter (3×3) is applied. Examples of resulting confidence images are shown in Fig. 3 for two views.

Once the confidences have been computed, we combine the reflectance estimates using Eq. (3). The result is more robust and contains fewer artifacts than any of the individual reflectance estimates from the radiance images as shown in Fig. 3.

4.4 Texture Filling for Real Object Removal

Removing a real object from the scene leaves a gap, or previously invisible region, for which we need reflectance estimates. We fill in this missing information using texture synthesis in a technique similar to Igehy and Pereira [14]. We use El-Maraghi's [7] implementation of Heeger and Bergen's [13] texture synthesis in our system.

To synthesize the textures needed, we extract a texture sample from the *reflectance image* from every polygon that now covers the region to fill. The extraction of the sample is currently done manually, but we are experimenting with automatic extraction procedures. This sample is fed to the synthesis algorithm which generates a texture of the same size as the region to fill. The generated texture is applied to the reflectance using a masking process, described in Section 5.3. The generated textures are stored for objects marked as "removable" accelerating the interactive remodeling operations.

It should be noted that texture generation is performed on the reflectance image and is thus not hindered by shadows or lighting variations during the object removal. The reprojection of the shadows with the new scene will generate a correct image of the scene without the real object.

4.5 Initialising the Hierarchical Radiosity System

To bootstrap the hierarchical radiosity system, the reflectance values recovered by Eq. (3) are reprojected onto the corresponding polygons, initialising the reflectance values. For the polygons invisible in the image used for the interactive session, we take a sample of the texture during the photomodeling session and get an average value using Eq. (2). For parts of polygons invisible from the fixed viewpoint, we use an average reflectance value computed from the visible parts.

With this approximation, a first radiosity solution is computed by our system, using an implementation of hierarchical radiosity with clustering [23]. The subdivision is set to a relatively coarse level since such a level is sufficient for computing indirect light, which varies slowly. An example mesh is shown in Fig. 4(b).

Recall that direct effects, including direct shadows, are treated separately for display. Direct light is however computed by the radiosity system, but simply ignored for display. The subdivision is fixed at the beginning of the process to a minimum area

(a) (b)

Fig. 4. (a) The original view of the scene and (b) the corresponding radiosity mesh used to simulate indirect light and dynamic updates; note the coarse subdivision.

threshold. Nonetheless, we maintain the hierarchical nature of the radiosity algorithm, since links are established at different levels of the hierarchy, using a "standard" BF refiner [12]. Thus we will only need to update links and the radiosity values when performing interactive modifications.

5 Interactive Modification of Scene Properties

Once the reflectance has been computed for each pixel and the radiosity system set up, we can perform interactive modification of scene properties. The modifications that our system permits are related to lighting and geometry. The former includes changing a real light or adding virtual lights; the latter includes adding and moving virtual objects and removing real objects.

The web page *http://www-imagis.imag.fr/Membres/Celine.Loscos/relight.html*, contains high-resolution images and online movie sequences of interactive sessions. All timing results reported below have been taken on a SGI R10000 195Mhz processor.

5.1 Modifying Illumination

When we modify a light source emittance, two operations need to be performed:

- For indirect illumination, we need to compute a new radiosity solution. Given that the subdivision and the link structure are fixed after the initial solution, updating indirect illumination simply requires a few successive sweeps of the hierarchy to "gather" and "push-pull" [12] radiosity and is very fast (less than .05 seconds in our test scenes, since their polygon count is low).
- For display, the direct lighting component is recomputed at each pixel. Indirect illumination is displayed using hardware smooth-shading of the elements of the hierarchical radiosity subdivision, which are then blended into the final image. This results in the addition of the indirect irradiance $\hat{B}$ at each pixel.

In the pixel structure, we have stored the visibility and form-factor with respect to each light source. Thus the computation of the direct component is very rapid.

When displaying an image, we compute the following color at each pixel:

$$C = R\left(\sum_{s=0..n_s} F_s V_s E_s + \hat{B}\right) \tag{4}$$

for the n_s (real or virtual) light sources in the scene. Before inserting any virtual light source, the scene is lit only with its original light ($n_s = 0$). Shadows are *reprojected* due

to the visibility term V_s, since they have been removed from the reflectance.

An example is shown in Fig. 5. The original photo is shown in (a), reprojected initial lighting conditions in (b), and we show the addition of a virtual light source in (c). The entire update for adding the virtual light takes 3.1 seconds broken down as follows: visibility 2.5 sec., shaft/radiosity operations 0.4 sec., indirect light blending and other 0.2 sec. Recall that in the case of the light source insertion, we are required to update *all* the pixels of the image. During dynamic updates, we cast a small number of rays to the light sources, resulting in aliased shadows. An additional "shadow clean-up" could be performed when the user stops modifying the scene, with a higher shadow sampling rate.

5.2 Modifying Scene Geometry

To allow interactivity when adding, removing or moving objects and lights, we maintain a shaft data structure [11], inspired from the work of Drettakis and Sillion [6]. Updating the entire table requires in the order of a few minutes for visibility values, especially when using many rays per light source; using the method described below reduces this time to fractions of a second.

A hierarchical shaft [11] data structure is constructed from the first radiosity solution, and corresponds to each light transfer link. When we add an object it is first attached to the root cluster of the scene; links are established to the light sources as appropriate, based on the refinement criterion, and visibility information is computed.

The hierarchy of shafts is used for two purposes: (a) to identify the pixels for which *direct* illumination has changed (i.e., the shadow regions of the new object); and (b) to identify the links for which visibility needs to be updated (i.e., all links whose shaft is cut by the new object), for both direct and indirect light transfers.

To achieve the above, we descend the hierarchy of shafts, finding those intersected by the new object. The hierarchical elements attached to the end of a shaft originating at a light source are marked as "changed". While descending, the visibility of the modified links is updated. With all necessary links updated, we recompute a radiosity solution with only gather and push-pull steps.

The pixel data structure is then updated and displayed. The bounding box of the initial and final position of the moving object are first projected onto the image-plane, limiting the region of the screen directly affected by the motion. For this region a new item buffer is performed, and the pixels under the previous object position are found as well as those under the new position, since the polygon IDs will have changed. For these pixels, reflectances are kept to the original values for the "uncovered" pixels and updated to that of the virtual object for the newly covered pixels. New form-factors and visibility values are then computed for all the pixels changed in the modified region.

For the pixels associated with patches tagged as "changed", visibility with respect to the sources is recomputed. These are not as localized as the directly affected pixels, but their number is often small.

The entire pixel table is then traversed to update the indirect illumination value at each pixel, based on the new global illumination calculation; again, this is performed with hardware rendering of the hierarchical radiosity patches.

When inserting a new light source, the form-factor and visibility with respect to the source need to be computed for every pixel.

When removing an object, we perform a similar process. We delete every link and all corresponding shaft structures of the removed object.

When moving an object, the process is equivalent, but we do not have to delete the

links. We just have to update the information (form-factors and visibilities). Shafts due to the moving object are deleted and reconstructed with its new position.

In Fig. 5(d) we show the insertion of a virtual object in a scene lit with the original light and an added virtual light source. The insertion requires 1 sec., of which visibility accounts for .5 sec., shafts .1 sec. and the rest .4 sec. When moving the virtual object, we achieve update rates of about 1 sec. per frame, with a similar breakdown to that of the object insertion (Fig. 5(e)).

5.3 Removing Real Objects

When the user chooses to remove an object, she indicates the object to the system. Similarly to virtual objects, we know *exactly* which region of the screen will have to be filled, since the correspondences between polygons and pixels are known through the polygon IDs stored in the pixel data structures. We automatically create two masks corresponding to this region: a weight mask and a texture mask [14]. At first, each contains "1" over the region to fill and "0" elsewhere. We extend the weight mask a few pixels to compensate for inaccuracies in the removed object geometry (to avoid leaving any color from the removed object in the image).

The object is then removed from the scene and a new item buffer is performed to update the polygon IDs. The polygon IDs present in the region to be filled indicate from which polygons we have to extract textures. The texture mask is filled with these new IDs and the weight mask is blurred around its "0/1" borders. This allows the composition of the synthesized texture with the texture from the image: when the mask is 0, the color of the pixel will be the color in the reflectance image, when the mask is 1 the color will be taken from the synthesized texture and a fractional weight will allow a smooth transition from the synthesized texture to the original image (e.g., the original colors present in the image).

The reflectance is then updated for the pixels affected, as well as the visibility and form-factors, as in the case of virtual object motion/removal. Results of object removal are shown in Fig. 6.

A second example of real object removal is shown in Fig. 7. In the context of an interior redesign, we may want to remove doors for example, which is hard to do in the real world. This is shown Fig. 7(b). Note that due to approximate reflectance estimation, the texture generation results in slightly visible discontinuities. A virtual object has been added in (c) and a different lighting configuration created in (d).

6 Conclusion

We have presented a new approach to synthetic relighting and remodeling of real environments. Our approach is based on a preprocessing step to recover approximate reflectance properties from a sequence of radiance images. Radiance images are taken from a fixed viewpoint with varying illumination (i.e., different positions of the same light source), using a simplified reconstructed model of the scene. Using the information in the images and the 3D reconstructed model, we create reflectance images for each light position by estimating direct illumination and light source visibility as well as indirect light. The reflectance images are merged by a weighted average based on the confidence level we have in the reflectance at each pixel in each radiance image. In our case, this is based on visibility (points in shadow have low confidence); a filtering step is applied to compensate for errors in geometric reconstruction and illumination computation.

After the reconstruction has been performed we can interactively modify scene properties. This is achieved by efficiently identifying regions of the screen which need updating, and performing a pixel-by-pixel update for direct light. Indirect lighting is treated separately with an efficient hierarchical radiosity structure, optimized for dynamic updates.

In our implementation we can virtually modify real light intensity, insert and move virtual objects, and even remove real objects *interactively*. Despite inevitable artifacts, the quality of the images is sufficient for the purposes of interactive lighting design and limited remodeling.

Independently to our work, Yu *et al.*[26] have recently developed more robust techniques for reflectance estimation, including specular effects in particular. These are based on capturing images of the entire scene, and computing radiosity to estimate the reflectance using clever iterative methods and high-dynamic range images. We believe that our approach can benefit from such improved reflectance estimation (for example to remove the artifacts in texture generation in Fig. 7) as well as for the reflectance of objects which are not visible in the radiance image. On the other hand, we believe that both our interactive approach, especially for global illumination, as well as our confidence maps could be useful for such approaches.

In future work, using the high dynamic range radiance images of Debevec and Malik [3] will allow us to achieve more accurate reflectance extraction. Once we have more confidence in the original radiance most of the errors in the reflectance estimation will be due to indirect light. The hierarchical radiosity framework has the added advantage that it can be used to bound indirect illumination errors and thus should allow us to achieve better results.

We also need to investigate ways to allow motion of the viewpoint, which is currently an important limitation of our approach. Also, the texture generation approaches we have used are limited to stochastic textures. With some user intervention, it may be possible to achieve satisfactory results with deterministic textures also.

From a more practical point of view, we can add the synthetic motion of real objects simply into our system. A view-independent texture of the real object is required, which can be provided by our photomodeling system, as well as a modified rendering routine. As was discussed in the results, the largest expense in the updates is the calculation of visibility for direct lighting. These calculations can be easily parallelized, and we hope to achieve good speedups in a parallel version, enhancing interactivity.

References

1. D. R. Baum, H. E. Rushmeier, and J. M. Winget. Improving radiosity solutions through the use of analytically determined form-factors. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 325–334, July 1989.
2. P.E. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98 Conference Proceedings*, Annual Conference Series, pages 189–198, July 1998.
3. P.E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH '97 Conference Proceedings*, Annual Conference Series, pages 369–378, August 1997.
4. P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH '96 Conference Proceedings*, Annual Conference Series, pages 11–20, july 1996.
5. G. Drettakis, L. Robert, and S. Bougnoux. Interactive common illumination for computer augmented reality. In *Rendering Techniques '97 (8th Eurographics Workshop on Rendering)*, pages 45–56. Springer-Verlag, June 1997.

6. G. Drettakis and F. Sillion. Interactive update of global illumination using a line-space hierarchy. In *SIGGRAPH '97 Conference Proceedings*, Annual Conference Series, pages 57–64, August 1997.
7. T. El-Maraghi. An implementation of Heeger and Bergen's texture analysis/synthesis algorithm with source code. http://www.cs.toronto.edu/~tem/2522/texture.html.
8. O. Faugeras. *Three-Dimensional Computer Vision — A Geometric Viewpoint.* MIT Press, 1993.
9. O. Faugeras, S. Laveau, L. Robert, G. Csurka, and C. Zeller. 3D reconstruction of urban scenes from sequences of images. Tech. report 2572, INRIA Sophia-Antipolis, May 1995.
10. A. Fournier, A.S. Gunawan, and C. Romanzin. Common illumination between real and computer generated scenes. In *Proc. of Graphics Interface '93*, pages 254–262, May 1993.
11. E. A. Haines and J. R. Wallace. Shaft Culling for Efficient Ray-Traced Radiosity. In *Photorealistic Rendering in Computer Graphics (Proceedings of the 2nd Eurographics Workshop on Rendering)*, New York, NY, 1994. Springer-Verlag.
12. P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.
13. D.J. Heeger and J.R. Bergen. Pyramid-Based texture analysis/synthesis. In *SIGGRAPH '95 Conference Proceedings*, Annual Conference Series, pages 229–238, August 1995.
14. H. Igehy and L. Pereira. Image replacement through texture synthesis. In *Proceedings of the 1997 IEEE International Conference on Image Processing*, 1997.
15. F. Leymarie, A. de la Fortelle, J. Koenderink, A. Kappers, M. Stavridi, B. van Ginneken, S. Muller, S. Krake, O. Faugeras, L. Robert, C. Gauclin, S. Laveau, and C. Zeller. Realise: Reconstruction of reality from image sequences. In *International Conference on Image Processing*, volume 3, pages 651–654, Lausanne (Switzerland), 1996. IEEE Signal Proc. Soc.
16. C. Loscos and G. Drettakis. Interactive relighting of real scenes. Tech. report 0225, INRIA Rhône-Alpes, November 1998.
17. C. Loscos, G. Drettakis, and L. Robert. Interactive modification of real and virtual lights for augmented reality. In *SIGGRAPH '98 Technical Sketch (Visual Proceedings)*, July 1998.
18. E. Nakamae, K. Harada, T. Ishizaki, and T. Nishita. A montage method: The overlaying of the computer generated images onto a background photograph. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 207–214, August 1986.
19. Photomodeler. http://www.photomodeler.com.
20. P. Poulin, M. Ouimet, and M.-C. Frasson. Interactively modeling with photogrammetry. In *Rendering Techniques '98 (9th Eurographics Workshop on Rendering)*, pages 93–104. Springer-Verlag, June 1998.
21. Y. Sato, M.D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. In *SIGGRAPH '97 Conference Proceedings*, Annual Conference Series, pages 379–387, August 1997.
22. E. Shaw. Hierarchical radiosity for dynamic environments. *Computer Graphics Forum*, 16(2):107–118, 1997.
23. F. X. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):240–254, September 1995.
24. G.J. Ward. The RADIANCE lighting simulation and rendering system. In *Proceedings of SIGGRAPH '94*, Annual Conference Series, pages 459–472, July 1994.
25. H. Weghorst, G. Hooper, and D. P. Greenberg. Improved computational methods for ray tracing. *ACM Transactions on Graphics*, 3(1):52–69, January 1984.
26. Y. Yu, P.E. Debevec, J. Malik, and T. Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *SIGGRAPH '99 (to appear)*, 1999.
27. Y. Yu and J. Malik. Recovering photometric properties of architectural scenes from photographs. In *Computer Graphics (ACM SIGGRAPH '98 Proceedings)*, July 1998.

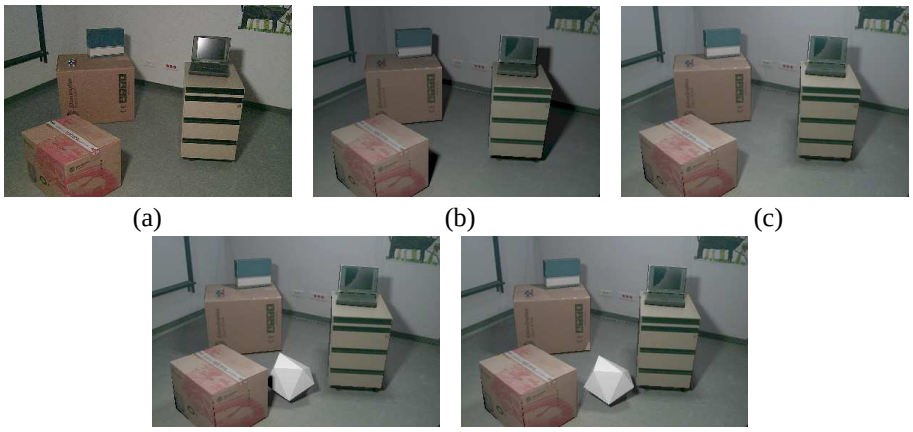

Fig. 5. (a) The original radiance image (photo). (b) Original reprojected lighting conditions, displayed using the recomputed direct and indirect components, (c) a virtual light has been inserted into the scene adding the light took 3.1 seconds (for 400x300 resolution). (d) A virtual object has been inserted into the scene with both lights on; adding the object required 1 sec. (e) Moving the virtual object requires 1.1 sec.

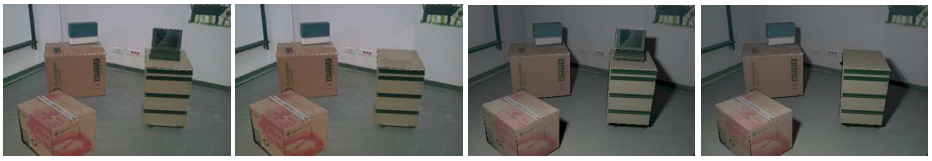

Fig. 6. Texture filling examples for real object removal. (a) Initial reflectance image (b) The laptop is removed. The laptop was removed entirely *synthetically* since no additional image was captured. (c) The original relit image. (d) The relit image after removal. Removal of the laptop took 0.7 sec., since generated textures are pre-computed for "removable" objects.

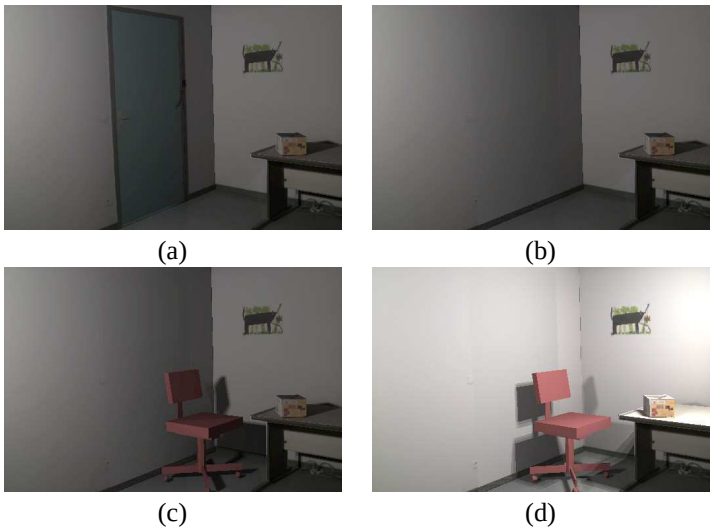

Fig. 7. A second real object removal example. (a) The original relit image, (b) the relit image after removal of the door, which took 2.9 sec., for a resolution of 512x341. (c) A virtual chair has been added to the scene, requiring 3.4 sec., and (d) a virtual light added (needing 6.6 sec.).