Fast and Robust High Dynamic Range Image Generation with Camera and Object Movement

Thorsten Grosch

Institute for Computational Visualistics
University of Koblenz-Landau
Universitätsstr. 1, 56070 Koblenz, Germany
Email: grosch@uni-koblenz.de

Abstract

High dynamic range (HDR) images play an important role in todays computer graphics: Typical applications are the improved display of a photograph with a tone mapper and the relighting of a virtual object in a natural environment using an HDR environment map. Generating an HDR image usually requires a sequence of photographs with different exposure times which are combined to a single image. In practice two problems occur when taking an image sequence: The camera is moving or objects are moving during the sequence. This results in a blurry HDR image due to the camera movement and moving objects appear multiple times as ghost images in the combined image. In this paper, we present a simple and fast algorithm for a robust HDR generation that removes these artifacts without prior knowledge of the camera response function.

1 Introduction and Previous Work

Nowadays, high dynamic range images are an essential part in the field of computer graphics. Especially for natural illumination, images are required that contain a radiance value at each pixel. A photograph taken with a usual camera can not capture the whole dynamic range of the visible radiances. There are always saturated regions, underexposed regions, or both. Moreover, each digital camera uses an unknown non-linear scaling function for converting the incoming sensor exposure to a pixel color. Determining this camera response function was first mentioned by Mann et al. [4]. Here, a simple gamma function is used to describe the radiometric response function of a camera. A more general calculation was presented by Debevec and Malik [1]. They use a series of images with different exposure times to combine the images to a single HDR image and to recover the camera response function by solving a linear system of equations. The camera curve is described as a table of 256 exposure values for each possible grey value. An alternative algorithm was presented by Robertson et al. [9] based on a probabilistic approach. Mitsunaga and Nayar [5] calculate the camera curve as a polynomial function with fixed degree. Their method is able to determine the response function with only rough estimates of the exposure times. A robust method for determining the camera curve was also presented by Grossberg and Nayar [2]. Instead of comparing pixel values, they compare the brightness histograms of the images to determine the camera response function. By contrast, only a few hardware solutions for capturing HDR images exist, like the Spheron camera, HDRC (IMS Chips) or the digital micromirror array introduced by Nayar et al. [6]. At present, the conventional way to create an HDR image is to take a sequence of images with different exposure times and to combine them to a single HDR image with one of the methods described in [1][2][4][5][9].

Two problems occur when combining these images:

1. If the images are taken with a hand-held camera, the combined HDR image will look blurry because of small camera movements. This can mainly be avoided by placing the camera on a tripod, but even pressing the capture button or changing the exposure time can result in an offset of a few pixels.

2. Ghost images appear in the combined image because of moving objects while capturing the image series. This is not a problem in indoor environments, but often unavoidable in outdoor environ-

ments. Typical examples are moving people, clouds or trees waving in the wind.

One example with both camera and object motion is shown in Figure 1, the combined image without alignment is shown in Figure 2.

Ward [10] presented a method to solve the first problem by assuming a pure translational camera movement. Each image of the sequence is converted to a so-called median threshold bitmap (MTB). These binary images are almost identical for all exposure times and can be used for aligning the images (Figure 3). The algorithm starts with a low resolution image to get an initial estimate for the translation and refines the estimate for the high resolution images. The method is very fast because of many hardware optimizations.

The second problem is mentioned by Kang et al. [3] for creating an HDR video. Here, a video sequence is recorded with two alternating exposure times. Neighbouring images in a video sequence with known exposure time are first globally registered and then locally registered. First, a global registration computes an image warp which aligns the two images optimally. Then, the local registration is computed using a hierarchical homography to determine the optical flow of small pixel movements. The authors show that their method is also applicable to still images taken with a digital camera with convincing results. However, their algorithm requires a precomputed camera response function and it is quite time consuming. They report that generating an HDR image from a sequence of five images takes about 30 seconds. Since we do not expect that the response function is known for all possible camera adjustments, we decided to develop an algorithm for robust and fast HDR generation without prior knowledge of the response function. Our algorithm can be summarized as follows:

1. Calculate Median Threshold Bitmaps
2. Translational Alignment
3. Translational and Rotational Alignment
4. Calculate Camera Response Function
5. Calculate Error Map
6. Combine HDR Image
7. Manual Corrections

Each step will be described in the following sections: Section 2 explains the alignment of the images, in section 3 we describe our method for removal of ghost images. The results are presented in section 4. An intuitive tool for manual correction of the remaining artifacts is presented in section 5 before we conclude in section 6.

Figure 1: Image sequence taken with a hand-held camera

Figure 2: Combined HDR image without alignment

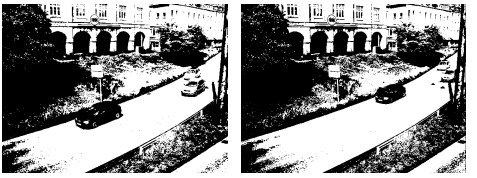

Figure 3: Median threshold bitmaps for two images of the car sequence. The bitmaps are almost identical except of the moving objects.

2 Image Registration

Since we have to solve two problems, we assume that we have a static scene with a few moving objects. We solve the correct alignment first and remove the ghost images in the second step.

2.1 Basic Ideas

Our alignment method is based on the algorithm presented by Ward [10], but we include an image rotation. During our test series we found that a pure translation is often not sufficient. Several image sequences contain a small rotation, especially when the camera is held vertically. We use an optimization algorithm to calculate the best rotation angle for aligning two binary threshold images. The starting value for the algorithm is the translation determined with the method described in [10] and a rotation angle set to zero.

When XOR-combining both bitmaps, we count the number of remaining white pixels which is used as a function value for the optimization algorithm. The Downhill Simplex method [7] is used to find the best transformation since it is known as a robust optimization algorithm.

2.2 Graphics Hardware

We use the graphics hardware for a fast calculation of the XOR combination of two binary threshold bitmaps. The basic idea is to draw both images as rectangular, screen-filling polygons with the median threshold image as a texture. The first rectangle is kept fixed, while the second rectangle is translated and rotated around the view axis, as shown in Figure 4. After a logical XOR combination, the resulting bitmap can be read from framebuffer to main memory for counting the white pixels. Because reading pixels back to main memory is quite slow, we developed an optimized version, which is described in the next section.

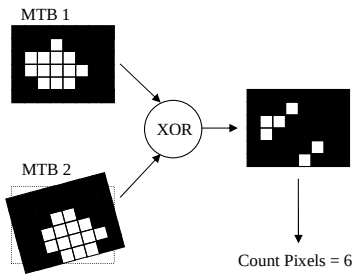

Figure 4: Image alignment: The first image is kept fixed while the second image is translated and rotated around the view axis.

2.3 Fast Summation of White Pixels

To avoid reading the pixels from framebuffer back to main memory, we do not use the logical operations included in OpenGL. Instead, we use a special fragment program for the XOR combination of both bitmaps. Both median threshold bitmaps are represented as textures, but we draw *only the transformed bitmap* and calculate the superposed texture pixels in the fragment program. As can be seen in Figure 5, the texture pixel of the transformed bitmap is accessed by the texture coordinates (s,t). The corresponding texture coordinates for the first bitmap are simply the window coordinates (x,y) of the current fragment.

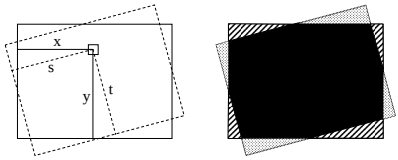

Figure 5: Texture coordinates (left) and intersection region (right) of both median threshold bitmaps.

Because current fragment programs do not support logical operations, we discard the current fragment in case of *identical* texture pixels to simulate the XOR combination. The exclusion bitmaps are used as textures, too. If at least one of them is zero, the fragment is discarded. The fragment program is shown in Figure 6.

To count the white pixels, we use an *occlusion query* when drawing the transformed rectangle. Because we discard the black pixels in our fragment program, the occlusion query returns the number of white pixels.

2.4 Intersection Region

When combining both images, care must be taken to count only pixels in the *intersection* of both rectangles, as can be seen in Figure 5. Because we draw only the transformed rectangle, we restrict to the intersection region automatically. Regions outside the viewport are removed by clipping (dotted regions) and fragments outside the transformed rectangle (striped regions) are never accessed.

```
void combineFP(out float4 color : COLOR,
                   float4 tc    : TEXCOORD0,
                   float4 wpos  : WPOS,
      uniform samplerRECT mtbSampler1,
      uniform samplerRECT mtbSampler2,
      uniform samplerRECT ebSampler1,
      uniform samplerRECT ebSampler2)

{
  // discard identical pixels
  // to simulate XOR operation

  if (texRECT(mtbSampler1, wpos.xy).r ==
      texRECT(mtbSampler2, tc.xy).r)
      discard;

  // discard fragment if at least
  // one exclusion bitmap is zero

  if (texRECT(ebSampler1, wpos.xy).r == 0)
      discard;

  if (texRECT(ebSampler2, tc.xy).r == 0)
      discard;

  // return white color

  color = float4(1.0, 1.0, 1.0, 1.0);
}
```

Figure 6: Fragment program: Both median threshold bitmaps and exclusion bitmaps are used as textures. When drawing the transformed rectangle, the bitmaps are combined and the fragment is discarded in case of a black pixel.

2.5 Comparison

The correct alignment could be found in most of our testcases even in case of object motion during the series. Figure 7 shows the XOR combination of two median threshold bitmaps before and after the optimization, the aligned HDR image is shown in Figure 8.

Without a rotational alignment, several artifacts appear in the HDR image, as can be seen in Figure 9. The translation and rotation values for this image are listed in table 1.

3 Removing Ghost Images

After aligning the images, we are able to remove the ghost images. To safely detect the affected pixels, we calculate the camera response function first.

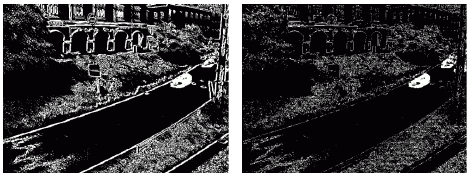

Figure 7: XOR combination of the median threshold bitmaps. Left: Original images. Right: Aligned images after 41 iterations of the Downhill Simplex algorithm.

Figure 8: Aligned HDR image: The alignment works despite the moving objects which appear as ghost images.

3.1 Camera Response Function

Most algorithms for calculation of the camera response function require static scenes and a direct correspondence between the pixels. Because we have a non-aligned sequence with object motion, we use the algorithm presented by Grossberg and Nayar [2]. This algorithm calculates the response function based on cumulative histograms and is mostly unaffected by camera or object motion. In the following, we denote the camera function f as a conversion of sensor exposure X to pixel color z: $f(X) = z$.

3.2 Predicted pixel colors

With a known camera response function f, we can predict the pixel color from one image to another. Suppose we have a pixel with color $z_{i,j}$ at position i, taken with exposure time Δt_j. In a second image, taken with exposure time Δt_k, the corresponding

Figure 9: Difference between pure translational alignment (left) and alignment including a rotation (right). Without a small image rotation, the combined image contains a slight blur. Moreover, several parts in the image appear twice like the top of the towers shown in the insets.

pixel color is $z_{i,k}$. The resulting exposure values at sensor i are:

$$f^{-1}(z_{i,j}) = X_{i,j} = E_i \cdot \Delta t_j \quad (1)$$

$$f^{-1}(z_{i,k}) = X_{i,k} = E_i \cdot \Delta t_k \quad (2)$$

where E_i is the sensor irradiance.

Rearranging these equations leads to a *predicted* pixel color $\tilde{z}_{i,k}$ for the second image:

$$\tilde{z}_{i,k} = f(\frac{\Delta t_k}{\Delta t_j} f^{-1}(z_{i,j})) \quad (3)$$

For each pair of consecutive images, we test if the real color $z_{i,k}$ in the second image is well approximated with the predicted color $\tilde{z}_{i,k}$ from the first image:

$$|\tilde{z}_{i,k} - z_{i,k}| < \epsilon \quad (4)$$

The user parameter ϵ depends on the amount of noise in the camera images. A significant difference between the two colors indicates object motion. In this case, we mark the pixel as invalid in a special *error map*. Pixels set to invalid are not used when the HDR image is combined from the image sequence. Here, we use the idea introduced in [3] and fill the invalid pixels with a reference image. The difference is, that our reference image is selected by counting the number of underexposed and saturated pixels *only in invalid regions* marked in the error map. The image with the lowest number is selected as the reference image. In this way we minimize the loss of dynamic range. The resulting error map for the car sequence is shown in Figure 10.

Because a direct insertion from one image increases the amount of noise, we test if some of the other images can be used as well. Therefore, we calculate the predicted color from the reference image to all the other images (Equation 3) and calculate the average of all pixels which fulfil Equation 4.

The color prediction only works if we are not in a saturated or underexposed region. In these regions, the pixel color is always black or white. We therefore use two threshold values to exclude the black and white pixels. Typically, we only check predicted colors if at least one of the two pixel colors is in the range [20..240].

Table 1: Alignment values for the image shown in Figure 9. The top rows shows the translational alignment. The bottom row shows the alignment with a rotation. Note that the best translation values for alignment are floating point numbers.

Alignment	1 - 2	2 - 3	3 - 4	4 -5
t_x	-3	-7	-5	7
t_y	5	3	-4	7
t_x	-3.0	-6.83	-5.06	7.09
t_y	5.0	3.35	-3.71	6.9
α	0.0	-0.31	-0.17	0.09

4 Results

We tested our algorithm with 180 hand-held image sequences. In 79 cases, the visual image quality could be improved with an additional image rotation. In the remaining series, the rotation angle was too small for a visible difference. Because we use the translational alignment as an initial guess for our algorithm, we never decrease the quality of the alignment. In a few cases, loss of visual image quality could be observed in certain regions when using a rotation. Nevertheless, the overall appearance of

Figure 10: Error map for the car sequence. All pixels that violate the predicted color condition are marked in black. Note that all moving cars have been detected. Some pixels inside the cars are not marked due to a similar background color.

Figure 11: Aligned HDR image without ghost images. Note that the cars are not completely removed because we replace defective regions from a *single* image.

the image could be improved. One such example is shown in the middle row of Figure 13 in the color image section.

We observed alignment problems with 13 series. These problem cases contain large object movements, smoke or fire with changing illumination during the series and images with noisy bitmaps. After manually adjusting the threshold from 50% to a different value, seven of these series could be aligned. Despite these problems, the alignment based on median threshold bitmaps proved to be very robust. Several problematic sequences containing water (rivers, fountains), reflective or refractive objects (light probes, glass sculptures) or wind (waving trees and flags) were aligned properly.

In our test series, the average translation between two hand-held photographs is 8.19 pixels (for a resolution of 1024 x 768 pixels), the average rotation angle is 0.16 degrees.

We compare our alignment with the alignment used in Photoshop CS2. In the majority of cases, we observed a better image quality with our alignment, as can be seen in the color images. On the other hand, Photoshop CS2 achieves better results with the few problematic image series.

The computation time for creating an HDR image from a sequence of five (1024 x 768) images together with the camera response function is about four seconds on a dual Athlon PC, running at 2.21 GHz, equipped with a GeForce 7800 GTX graphics card. For comparison, the software implementation of the optimization algorithm takes about 20 seconds for the same task.

5 Manual Adjustments

Most problems in the combined HDR image can be solved with the algorithm described in the preceeding sections. However, some problems still remain:

- Pale ghost images because of similar colors of moving objects and background
- Clipped colors because of over - or underexposed regions in the reference image
- Small image alignment errors due to highly varying distances of the visible objects

Recently, Ward [8] proposed heuristics for ghost removal using all images, but this algorithm cannot guarantee the complete elimination of all ghost images. Since we believe that there is no way to ensure a consistent HDR image automatically, we decided to create a tool for interactive correction of the combined HDR image. Nowadays, some image manipulation programs support HDR images, like Photoshop CS2 or IDRUNA Photogenics. However, we developed our own tool because we can use the following information collected during HDR combination to aid the user with the correction:

- The aligned images
- The error map
- The camera response function

The main idea is to select a different reference image for a problem region and copy the information manually. Therefore, saturated or possibly damaged

regions can be visualized by showing the error image as red pixels. The main user interaction is the selection of one image from the sequence which is suitable for a defective region. The region is corrected with a special *clone brush* which copies the information from the selected image to the HDR image. The camera response function is used for conversion from the LDR image to the HDR image. Figure 12 shows a typical example: The left image is the combined HDR image with a few ghost images. In the center image, possible problem regions are shown as red pixels. One of the original images is selected and the problem region is filled with the information from the selected image with a few mouse strokes. If none of the images fits for a replacement, we use a standard clone brush to copy information inside the HDR image or simple painting tools.

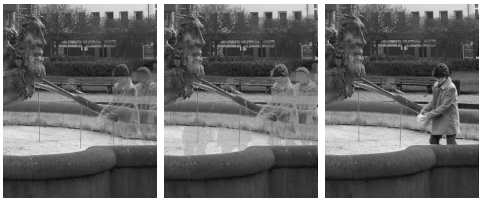

Figure 12: Left: Image region with a few ghost images. In this case, the reference image was too dark for a replacement. Center: Visualization of the error map. Right: Image correction by painting information from a different reference image with a clone brush.

6 Conclusion and Future Work

We presented an algorithm for robust high dynamic range image generation that can handle both camera and object motion without a precomputed camera response function. We extended the translation alignment based on median threshold bitmaps with an additional rotation. Using an additional rotation improved the image quality significantly in more than 40% of our test series. By exploiting the fast graphics hardware, we are able to align a sequence of five images in about four seconds. Regions containing ghost images can be detected by testing a predicted color for each pixel. These regions are filled with a reference image that minimizes the loss of dynamic range. For the remaining problem regions, we developed an intuitive tool for manual corrections.

As future work, we are searching for methods to further improve the HDR image quality. In comparison with the original LDR image, the HDR image ofter contains a slight blur. Moreover, images with a very short or long exposure time are often misaligned. In these cases, a different threshold value has to be used for the bitmaps instead of the median. Currently, we select this value manually, an automatic detection is still an open problem.

7 Acknowledgements

We would like to thank Markus Geimer, Jens Krueger, Matthias Biedermann, Oliver Abert, Tobias Ritschel, Rodja Trappe and Angelika Braeunig de Colan for helpful suggestions and proofreading the paper.

References

[1] P. Debevec and J. Malik. Recovering High Dynamic Range Radiance Maps from photographs. In *Proceedings SIGGRAPH 1997*, pp. 369-378

[2] M. Grossberg and S. Nayar. What can be known about the radiometric response from images? In *Proc. of European Conference on Computer Vision (ECCV) 2002*, pp. 189-205, 2002.

[3] S. Kang, M. Uyttendaele, S. Winder and R. Szeliski. High dynamic range video. In *ACM Trans. on Graphics 2003*, pp. 319-325, 2003.

[4] S. Mann and R. Picard. Being 'undigital' with digital cameras: Extending dynamic range by combining differently exposed pictures. In *Proc. of IS&T's 48th annual conference 1995*, pp. 422-428, 1995.

[5] T. Mitsunaga and S. Nayar. Radiometric self calibration. In *Proc. of IEEE Conference on CVPR 1999*, pp. 374-380, 1999.

[6] S.K. Nayar, V. Brazoi and T.E. Boult. Programmable Imaging Using a Digital Micromirror Array. In *Proc. of IEEE Conference on CVPR 2004*, pp. 436-443, 2004.

[7] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery. *Numerical Recipes in C++*. Cambridge University Press, 2002.

[8] E. Reinhard, G. Ward, S. Pattanaik and P. Debevec. *High Dynamic Range Imaging.* Morgan Kaufmann, 2005.

[9] M.A. Robertson, S. Borman and R.L. Stevenson. Dynamic range improvement through multiple exposures. In *Proc. of International Conference on Image Processing 1999*, pp. 159-163, 1999

[10] G. Ward. Fast, robust image registration for compositing high dynamic range photographs from hand-held exposures. *Journal of Graphics Tools, Volume 8/2003*, pp. 17-30.

Figure 13: Comparison between the different alignment techniques: Pure translational alignment (left), our method with translation and rotation (center) and Photoshop CS2 alignment (right).

Figure 14: No alignment (left), alignment (center) and ghost removal (right).