Fast Algorithm for Completion of Digital Photographs with Natural Scenes

K.K. Biswas, Siddharth Borikar and Sumanta Pattanaik, School of Computer Science, Univ. of Central Florida, Orlando, Florida, USA

Abstract

We present a fast method for completion of digital photographs of natural scenery, where the removal of an unwanted object creates a hole in the image. Because of the strong horizontal orientation in natural scenes, it is intuitive to synthesize the missing part by image fragments drawn from horizontally located areas, without the need to search whole of the image area to find the appropriate texture. We are thus able to select the texture that naturally blends into the rest of the scene horizontally. We also propose a modification to deal with sloping horizons, such as mountain areas, and show how the texture filling areas can be suitably chosen. We demonstrate the method by completion of a variety of scenes.

Introduction

The filling of missing information is a very important topic in image processing, with applications including image coding and wireless image transmission(e.g. recovering lost blocks), special effects (e.g. removal of objects), and image restoration (e.g. scratch removal).[3] The removal of portions of an image is an important tool in photo-editing and film post-production.

Image completion is an area related to texture synthesis. Inpainting techniques were naturally extended from paintings to images and films.[2] Image inpainting and image completion are not the same; they differ in the area that is to be filled or completed. In image completion regions are large and consist of textures, large scale structures and smooth areas. In image completion the region that is to be removed is specified by the user. It is generally some foreground element that needs to be taken off the scene. After removing the foreground element, the area is to be filled so that the image looks naturally complete.[5]

Various methods like clone brush strokes and compositing processes are used to complete the image, user skill is required in such methods. Texture synthesis can also be used to complete regions where the texture is stationary or structured. Reconstructing methods can be used to fill in large-scale missing regions by interpolation. Inpainting is suitable for relatively small, smooth and non-textured regions. In Computer Vision number of approaches related to image completion are available, dealing with edge and contour completion.

Our approach focuses on image based completion, with no knowledge of the underlying scene. In images with natural scenery, there is a strong horizontal orientation of texture/color distribution. We have tried to exploit this fact in our proposed algorithm to fill in missing regions. We follow the principle of figural familiarity and use the image as our training set and complete the image based on examples from horizontal neighborhood.

The assumption of horizontal orientation in natural images can be verified by taking the Fourier transform of such images. Fourier transforms of such images exhibit a distinct vertical line at the center. This indicates that the color/texture in the image is horizontally oriented.

The organization of the paper is as follows. In section 2, we briefly discuss some of the related work. In section 3 we present our algorithm on image completion. In section 4, we make a comparison of computational requirements of our algorithm with that of Ref. [5].

Related Work

Image inpainting is a method for repairing damaged pictures or removing unnecessary elements from pictures. Inpainting has its applications in restoration of damaged paintings and photographs and also could be applied in image filling problems. The objective of inpainting is to reconstitute the missing or damaged portions of the work, in order to make it more legible and to restore its unity. An algorithm related to the basic techniques used by professional restorators in image restoration is used in Ref. [2]. The basic idea is to smoothly propagate information from the surrounding areas in the isophotes (lines of equal gray values) direction. The user has to provide the region to be inpainted.

Brooks et al.[4] have presented a mechanism to edit textures. It allows the user to perform replicated texture editing operations. A single editing operation at a given location would cause global changes, affecting all similar areas of the texture. The algorithm changes (edits or warps) the texture as defined by the user. The pixel values are changed as per the user input and the result is an automatically generated texture with limited user input.

Freeman et al.[8] have discussed a method to increase the resolution of images. Constructing polygon models for complex, real-world objects is difficult. Image based rendering (IBR), a complementary approach for representing and rendering objects, uses cameras to obtain rich models directly from real-world data. When the image is enlarged, since the original information captured is less, one gets a blurry result.

Heeger et al.[9] have described a method for synthesizing images that match the texture appearance of a given digitized sample. The focus of their paper is on the synthesis of stochastic textures. Some theories of texture discrimination are based on the fact that two textures are often difficult to discriminate when they produce a similar distribution of responses for set of (orientation and spatial-frequency selective) linear filters.

The work done by Igehy et al.[10] is another approach for image filling which uses the texture synthesis algorithm described in the paper by Heeger et al.[9] In the algorithm the noise is converted to an image using the original image and a synthetic texture which is derived from the target. The algorithm is completely based on Ref. [9], except that with every iteration of the algorithm, the original is composited back into the noise image according to the mask using a multi-resolution compositing technique that avoids blurring and aliasing.

Efros et al.[6] have modeled the texture as a Markov Random Field (MRF). The probability distribution of brightness values for a pixel given the brightness values of its spatial neighborhood is independent of the rest of the image. Based on this principle the neighborhood of the pixel is considered.

The major challenges in texture synthesis as posed in the paper by Wei et al.[12] are-1) modeling- how to estimate the stochastic process from a given finite texture sample and 2) sampling- how to develop an efficient sampling procedure to produce new textures from a given model. They claim that both the modeling and sampling parts are essential for good texture synthesis because the plausibility of generated textures will depend primarily on the accuracy of the modeling, while the efficiency of the sampling procedure will directly determine the computational cost of texture generation.

The algorithm described in Ref. [1] is similar to the one by Wei et al. in which the searching operation is used. The algorithm by Wei et al. uses the L2 norm to calculate the distances between two neighborhoods. Since the L2 norm is averaging of neighborhood pixels the edges, corners and other high level features in the image are not captured. This could lead to smoothed out edges and the human visual system being very sensitive to such details, these anomalies are easily recognized.

Bertalmio et al.[3] have tackled the problem of image filling. Since most image areas are not pure texture or pure structure, this approach provides just a first attempt in the direction of simultaneous texture and structure filling-in. The basic idea of the algorithm in Ref. [3] is that the original image is first decomposed into two images, one capturing the basic image structure and the other capturing the texture (and random noise). The first image is inpainted following Ref. [2], while the second one is filled-in with a texture synthesis algorithm.[6] The two reconstructed images are then added back together to obtain the reconstruction of the original data.

The work done by Drori et al.[5] comes closest to the work reported in this paper. In that paper[5] the missing regions are iteratively filled using the known image as the training set. The goal is to complete the unknown regions in an image based on the known regions, given the inverse matte and the source image. Self-similarity in the input image is required for this algorithm to give correct results. The algorithm requires extensive computation for every fragment at every scale. It takes a lot of time to complete a small portion of the image. In the next section, we present our approach to this problem which attempts to solve it in much less time.

The Proposed Image Completion Method

Our image completion follows the principle of figural familiarity. The missing areas are filled with familiar details taken by example from rest of the image as in Ref. [5]. However, as observed in most images of natural scenes there is a strong horizontal orientation of texture/color distribution. This prompts us to limit the search only along horizontal direction and thereby reduce the search complexity extensively.

To capture properly the intended texture area, we propose to complete the unknown part with blocks of pixels instead of individual pixels. This will help to capture the appropriate texture content. We place a grid over the complete image with each cell of size m x m. The search is now made to find the appropriate block from the source image to fill up the m x m block of the matte. The algorithm is detailed below.

Grid Algorithm

For the given matte (the hole created by removing the foreground object), Drori et al. conduct the search for the appropriate pixel by looking for a match between the fragment surrounding the current pixel in the matte and fragments in rest of the remaining source image. However, to have a seamless joint across the border of the matte, it would be more appropriate to locate the pixel pa most similar to the pixel lying on the left boundary of the matte, and replace the pixel on its right by the pixel on the right of p_a. In our method, we consider m x m sized fragments (block). We propose to consider m x m blocks for searching and replacement. For this purpose we put a grid on the whole image with grid size of m x m (Figure 1). To maintain figural similarity, we start filling up the left portion of the matte with image fragments from the left side of the matte, and the right portion of the matte with fragments from portion of the image lying to right of the matte.

Completion from Left Side:

On each horizontal row we pick up 'B_b', the m x m block lying closest to the left of the matte boundary. We now search the source image for 'B_s', a block whose characteristics match closely with that of 'B_b', the measure of closeness being the L2 norm. Then we simply replace 'B_{br}', the block on the immediate right of 'B_b', by B_{sr}, the block to the immediate right of 'B_s'. The underlying idea is that it will capture the figural familiarity between 'B_s' and 'B_{sr}' and ensure that it is reflected across the blocks 'B_b' and 'B_{br}' as well. When the next horizontal row of blocks is considered, we again find the block nearest to the boundary and find the closest match as before. The replacement block now could be the one right of this block or the block located immediately below 'B_{sr}'. It is observed that in many cases they turn out to be the same blocks. If it is not so we can make a selection based on how closely it fits in with other replaced blocks in the matte and the blocks near the boundary of the matte and replace it with the block that matches the best between the two blocks.

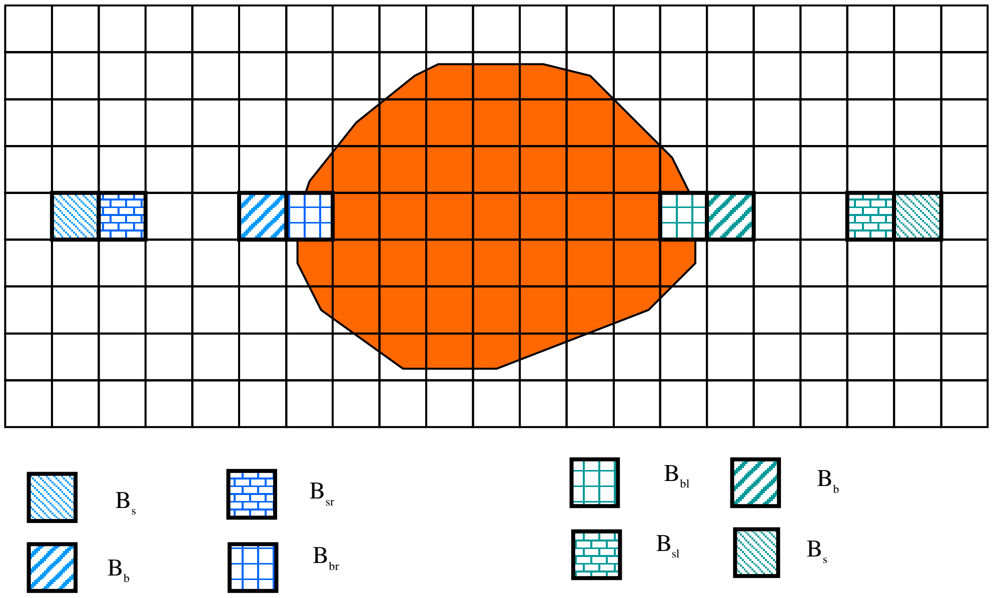

Figure 1. The m x m sized grid and the Block replacement policy.

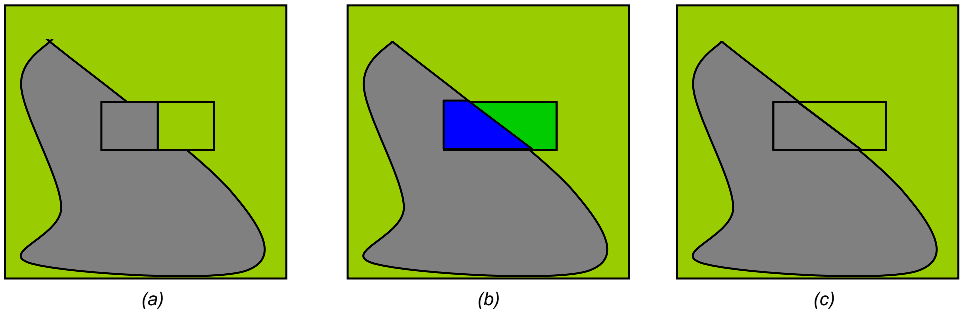

Figure 2. Two Colored Matte. (a) Original fill algorithm (b) Colored Mattes (c) Final result by using Color Mattes

Completion from Right Side:

A similar procedure is used to start filling up the blocks along the right boundary of the matte. For the block 'B_b', located next to the right boundary of the matte, we locate 'B_s', the most similar block by searching along the horizontal row in the right part of the source image. Now we replace 'B_{bl}', the immediately left block of 'B_b', with 'B_{sl}', which is located immediate left of 'B_s'. Again at each stage of filling the other blocks inside the matte, care is taken to ensure that replaced blocks match sidewise as well as with the blocks on the top. This helps in maintaining the figural continuity across the matte area. The method will result in uniformly filling up the matte row under consideration from the left and the right sides. However, it is quite likely that while blocks from left and right side of the matte are filled in properly, there may be a mismatch where the left and the right blocks meet at the centre of the matte. It is because the source image on the left side of the matte may not agree totally with the right side of the matte.

One possible solution to this problem would be a texture interpolation for middle three or five blocks, or a random displacement of the blocks by two or three pixels at the centre of the matte.

Correction for Self-Replacement

We fill-up the unknown region of the matte with grid blocks from the source image. In most of the images the removed foreground portion is one single area with a convex shape. Considering a grid row, the unknown region is continuous in such shapes. In some images, the shape of the missing region could be concave. This may cause a small part of the source image to be available between two limbs of the concave region. Due to lack of sufficient number of blocks along the horizontal row, the algorithm would attempt to select some blocks from the matte area itself (self-replacement). This may cause figural discontinuities in the missing area along one or more rows.

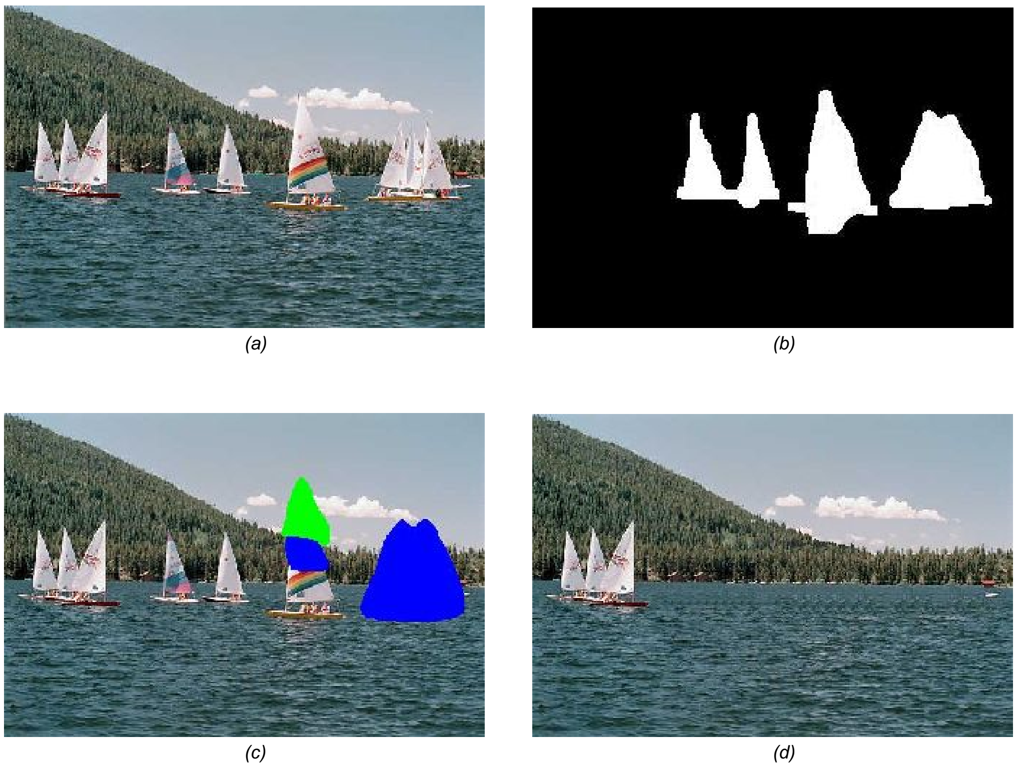
(a) (b) (c) (d)

Figure 3. Colored Matte

We apply the search procedure on the image we move along the grid row, filling in the blocks in the matte as we proceed. When we come across a row where there are such discontinuities, the result of search for source blocks might recover some blocks that are themselves unknown (low confidence). This means that the block in the matte is being filled by another block in the matte itself. As we proceed further, the error gets propagated along the row. We use the following method to handle such images. During the first pass, we keep track of the matte blocks that are incorrectly filled by blocks with matte blocks. To keep track of such blocks, we assign an error flag to the blocks that are incorrectly filled. The error flagged blocks are assigned a new matte color (say bright red color) as shown in Figure 1.

In the next pass, we create another matte with the flagged blocks as the missing regions. Now, the input image to the second pass is the output image of the first pass, containing the error blocks. Using this matte and the first pass filled image, as input to the algorithm we run the second pass of the algorithm. Since majority of the original missing blocks are filled in the first pass, there are not many missing regions in the second pass. This reduces the chances of the pixels being filled by pixels with unknown values again.

Tilted Orientations

Images of natural scenery containing mountains and similar slopes do not have perfect horizontal orientation. The slope could be quite appreciable. If the matte or a portion of the matte encompasses part of such terrain, the above algorithm would produce incorrect results, as it will try to fill in blocks from adjacent horizontal rows. The scene would not have the naturally pleasing look. In the middle of the missing region a line appears separating two parts at different heights. For example, in case of a mountain slope, blocks from the mountain area would fill one half of the matte, while the other half would be filled by blocks from the sky area. The slanting part of the mountain is lost in the replaced part and an odd shaped region shows up.

To take care of this problem we propose a two colored matte solution. The normal matte filled up from left and right side will exhibit the problem shown in Figure 2(a). The matte is divided in two parts along the slope in such a way that the two halves fall in differently textured region (Figure 2 b). This could be done either by the user, or by carrying out an examination of the blocks around the central line and keep on rotating the line till the matte is neatly divided in appropriate regions. Once this is done, it becomes very easy to fill up the two matte areas. Each color specifies the area from where it is to be filled, say, green colored matte should be filled from blocks only from the right side, and blue colored matte should be filled with blocks from left only (Figure 2c). A suitable matte for Figure 3(a) is shown in Figure 3(c) while the final results are shown in Figure 3(d).

In some cases, the matte happens to very close to the edge of the image. Since there is not enough information on both the sides of the matte, filling from only one side is preferred. So to specify the side from which the matte should be filled, colored matte can be used.

Computational Requirements

In this work, we have presented a technique for image filling. We have shown that even by using a restricted area of the source image, the matte area can be filled with background texture. We have made a comparison of computational requirement vis-à-vis another approach[5] used for image filling which is closest to our approach. The authors in Ref. [5] fill the image matte by applying extensive search over all positions and eight orientations in the image. Also, it considers the texture frequency of the image to decide the size of the fragment. Our algorithm is orders of magnitude faster than Ref. [5]. Our search is limited to a very small and restricted area in the image. The algorithm takes advantage of the fact that all the natural images have horizontal distribution of texture and color. Therefore the search is made over a small region i.e. a horizontal strip next to the missing region.

Timing Estimates

Our method does not apply the computations over many levels; also the search operation is only over a small region in the image and not over different levels and orientations of the image as in Ref. [5]. This considerably reduces the computation time of our algorithm. The authors in their paper[5] indicate that the computation times range between 120 and 419 seconds for 192 by 128 images and between 83 and 158 minutes for 384 by 256 images on a 2.4 GHz PC processor.[5] In our case, the computation time for images of 332 by 223 ranges from 10 seconds to 25 seconds depending on the matte area, while there is little degradation in the quality of the filled image as compared to Ref. [5].

Conclusions

A fast method has been proposed to complete images of natural scenery from which a foreground object has been removed and a matte created. The image is completed using appropriate regions from the rest of the image. We have shown that the size of the search areas of the known regions can be drastically reduced by making use of strong horizontal orientation of the image. Figure 4 presents the application of this idea on number of images.

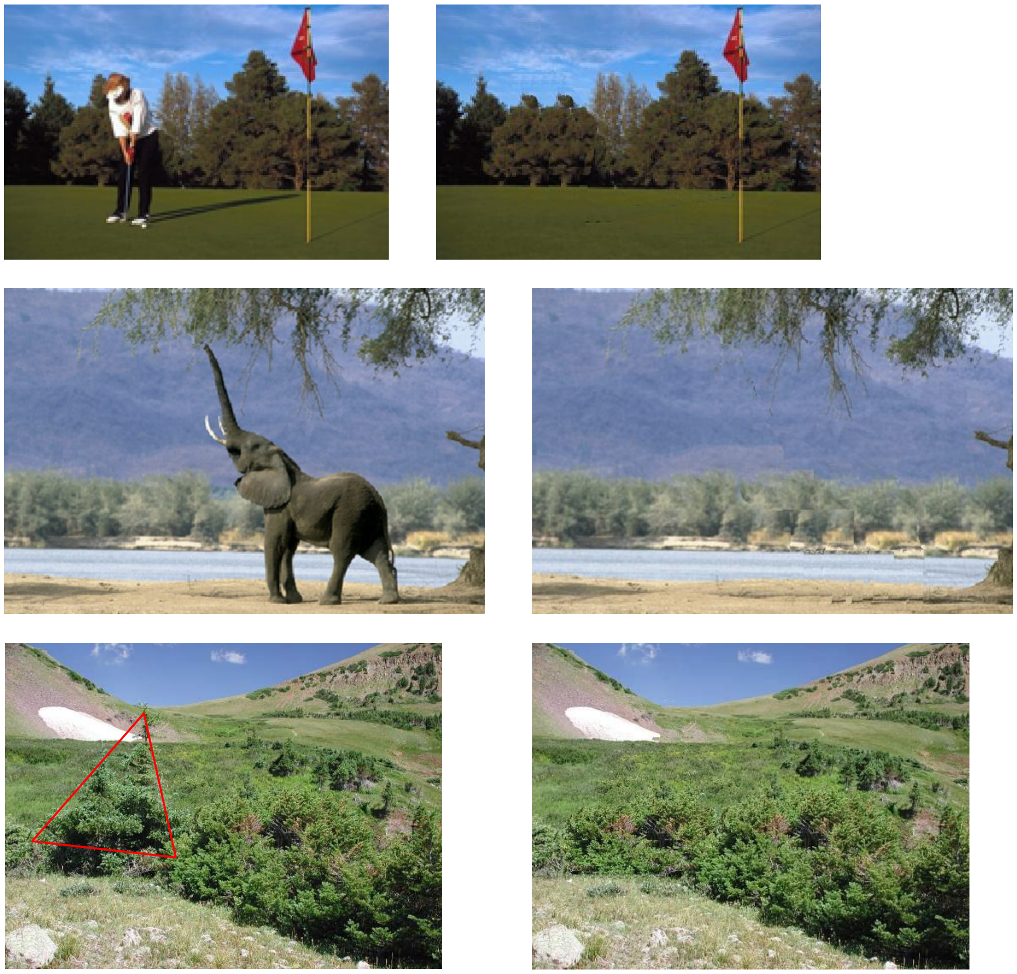

Figure 4. Results of Image completion

Computational comparison with another existing algorithm has been presented to show that image completion can be carried out quite fast.

References

1. Ashikhmin, M. Synthesizing natural textures. In ACM Symposium on Interactive 3D Graphics, 217–226. (2001)
2. Bertalmio, M., Sapiro, G., Caselles, V., and Ballester, C. Image inpainting. In Proceedings of ACM SIGGRAPH 2000, ACM Press, 417–424. (2000)
3. Bertalmio, M., Vese, L., Sapiro, G., and Osher, S. Simultaneous structure and texture image inpainting. In IEEE Conference on Computer Vision and Pattern Recognition, to appear. (2003)
4. Brooks, S., and Dodgson, N. Self-similarity based texture editing. ACM Transactions on Graphics, 21, 3, 653–656. (2002)
5. Drori, I., Cohen-Or, D., Yeshurun, H. Fragment Based Image Completion. In Proceedings of ACM SIGGRAPH 2003, ACM Press. (2003)
6. Efros, A., and Leung, T. Texture synthesis by non-parametric sampling. In IEEE International Conference on Computer Vision, 1033–1038. (1999)
7. www.freefoto.com – FreeFoto.com is a collection of free photographs for private non-commercial use on the Internet.
8. Freeman, W. T., Jones, T. R., and Pasztor, E. Example-based super-resolution. IEEE Computer Graphics and Applications, 56–65. (2002)
9. Heeger, D. J., and Bergen, J. R. Pyramid-based texture analysis/synthesis. In Proceedings of ACM SIGGRAPH 95, ACM Press, 229–238. (1995)
10. Igehy, H., and Pereira, L. Image replacement through texture synthesis. In IEEE International conference on Image Processing, vol. 3, 186–189. (1997)
11. www.mountainlake.com
12. Wei, L. Y., and Levoy, M. Fast texture synthesis using tree structured vector quantization. In Proceedings of ACM SIGGRAPH 2000, ACM Press, 479–488. (2000)
13. Welsh, T., Ashikhmin, M., and Mueller, K. Transferring color to greyscale images. ACM Transactions on Graphics, 21, 3, 277–280. (2002)

Author Biographies

K.K. Biswas is a Professor of Computer Science & Engg. at Indian Institute of Technology, New Delhi, India. His interests are in the area of high dynamic range image rendering and video indexing. He is currently a visiting faculty in the School of Computer Science of University of Central Florida .

Siddharth Borikar completed his MS in Computer Science from the University of Central Florida (2004) and is now working as a Software Consultant.

Sumanta Pattanaik is an Associate Professor of Computer Science in the University of Central Florida. His main area of research is realistic rendering. He is the Computer Graphics Category Editor of ACM Computing Reviews.