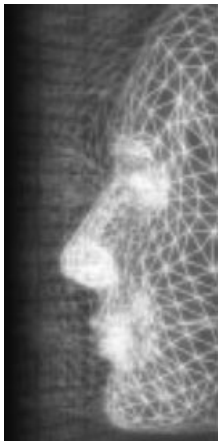
COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2006; **17**: 315–324
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cav.135

Tour into the picture with water surface reflection and object movements



*By Jinho Park, Nambin Heo, Sunghee Choi and Sung Yong Shin**

Given a still picture, tour into the picture (TIP) generates a walk-through animation of a 3D scene constructed from the picture. In this paper, we generalize TIP to deal with water surface reflection while allowing foreground objects to move. We formulate a non-linear optimization problem to find the 3D scene parameters with respect to the camera position, to automatically construct a reasonable 3D scene model, provided with a set of points and their corresponding points on the water surface. To synthesize a stream of reflected images on the water surface in accordance with the camera movement, we propose a novel image-based approach, which makes the best of the limited information available in the input picture. Furthermore, we incorporate water surface movement data acquired from water simulation on top of stochastic motion textures for objects such as trees and plants, to create a dynamic scene. Copyright © 2006 John Wiley & Sons, Ltd.

Received: 10 April 2006; Revised: 2 May 2006; Accepted: 10 May 2006

KEY WORDS: image-based rendering; tour into the picture; reflection map

Motivation

A picture, whether it is a painting or a photograph, is used as a medium to record an instance of an imaginary or real experience. This instance reminds us of lovely old memories or leads us to a world of imagination. Tour into the picture (TIP) by Horry *et al.*[1] facilitates a visual realization of such memories and imagination. The basic idea of TIP is to generate a walk-through(or fly-through) animation of a 3D scene constructed interactively from the input picture.

TIP has dealt with mainly pictures consisting of mountains and hills, water with boats, skies with clouds, trees, and buildings and streets, which together exhibit beautiful scenes to remember. In the original TIP, all objects in the picture remain still unlike in the real world, in which objects respond to natural forces in an oscillatory fashion. Moreover, each object is modeled as a billboard with a portion of the input picture attached. However, a natural water surface shows a reflected image of the environment, which varies depending on the camera position.

Recently, Chuang *et al.*[2] proposed a semi-automatic method to incorporate stochastic motion textures into a still picture, while freezing the camera parameters. This method is difficult to adapt to TIP without properly addressing the water surface reflection issue, since a reflected image on the water surface changes not only by time-varying motion textures but also by the camera position.

In this paper, we deal with two issues: how to generalize the 3D scene model of TIP[3] and how to incorporate water surface reflection for TIP. To synthesize reflected images of a scene on the water surface in accordance with the camera movement, we propose a novel image-based approach to 3D scene modeling and rendering, which makes the best of the limited information available in the input picture. We also employ water surface simulation data on top of stochastic motion textures caused by wind[2], to make objects in the input picture move in a visually convincing way. By adding object movements with the support of water surface reflection, our approach can be viewed as a generalization of TIP. By changing the viewpoint with the same support, our approach can also be viewed as a generalization of Chuang *et al.*'s work.[2]

Related Work

Image-based rendering enables real-time synthesis of a photorealistic image viewed from an arbitrary

*Correspondence to: S. Y. Shin, Division of Computer Science, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea. E-mail: syshin@jupiter.kaist.ac.kr

Contract/grant sponsors: Korea Science and Engineering Foundation; Brain Korea 21 Program of Ministry of Education and Human Resource Development.

viewpoint, exploiting scene information extracted from a set of reference images.[1,3–16] TIP is a special type of image-based rendering, where a single image is given as the input. In principle, we cannot reconstruct a 3D scene automatically from a single image. Thus, the user enjoys the freedom of modeling a 3D scene according to his/her interpretation. Horry *et al*.[1] proposed a simple scheme, called a spidery mesh to construct a 3D scene from a 2D image centered around a user-provided vanishing point. Based on projective geometry, Liebowitz[8] presented techniques for reconstructing 3D architectural models from single images, exploiting geometric constraints such as parallelism and orthogonality.

To lift the limitations imposed by a single vanishing point, Kang *et al*.[3] proposed a new modeling scheme for TIP based on a user-specified vanishing line. This scheme is simpler than that of Horry *et al*.[1] and yet more general to handle a broader class of images. By introducing the notion of a vanishing circle, the scheme can naturally be extended to 3D navigation in a panoramic image. We also adopt the latter scheme for 3D scene modeling.

Recently, Kang and Shin[14] further generalized the scheme of Kang *et al*.[3] for a single video stream and proposed an interesting scheme, the so-called TIV(tour in the video), which deals with moving foreground objects. Chuang *et al*.[2] animated a single picture by making its constituents such as trees, water, and clouds move in response to natural forces such as wind, with the viewpoint fixed. We incorporate this idea into TIP while still allowing the viewpoint to move.

Overview and Contributions

Our system consists of three steps: 3D scene modeling, motion generation, and rendering. For 3D scene modeling, we generalize the 3D scene construction scheme based on a vanishing line.[3] We first show how to estimate the distance between the image plane and the camera, the height of the camera above the water surface, and the distance between the background plane and the camera, to build the framework of a 3D scene model. We then describe how to place the foreground objects on the ground and the water surface. We finally explain how to extract three types of maps, that is, the texture map, the reflection map, and the mask map, from the input picture for each of scene components: the foreground objects, the ground, the water surface, and the background. The extracted maps are used later for rendering. In particular, the reflection maps facilitate synthesis of the reflected images of the 3D scene on the water surface while varying the camera position. For motion generation, we use stochastic motion textures in Reference [2] as well as water simulation data generated with a commercial tool.[17] For rendering, we describe how to produce an image from the 3D scene by exploiting the three maps extracted from the input picture.

From the modeling point of view, our contribution is a non-trivial generalization of the TIP scene model[3] to support the movements of the water surface and the foreground objects. From the technical point of view, our contribution lies in an image-based scheme for generating a reflected image of the 3D scene from the water surface. Our contribution also lies in a novel semi-automatic scheme to estimate the 3D scene parameters with respect to the camera position: the distance of the image plane from the camera (the camera focal length), the distance of the background plane from the camera, and the height of camera above the water surface plane. Together, the two schemes can handle moving objects in the environment, including the water surface itself, in a visually-convincing way.

The remainder of this paper is organized as follows: We describe how to model a 3D scene from an input picture, followed by motion generation and rendering. After showing results, we discuss limitations and weaknesses of our approach. Finally, we conclude this paper.

3D Scene Construction

3D Scene Framework

In this section , we describe how to construct the framework of a 3D scene model, that is, the global structure of the scene model from a picture. We generalize the 3D scene model of Kang *et al*.[3] to handle water scenes including seas, rivers, lakes, and so on. The vanishing line and the height of the camera above the ground plane are given by the user. On top of the two major components of the 3D scene framework, the background plane and the ground plane, we introduce the water surface plane as a new component. The foreground objects are placed on the ground plane or the water surface plane. We assume that the water surface plane is parallel to the ground plane and perpendicular to the background plane.

Under the assumption that the normal vector of the ground plane points in the global-up direction, this plane can be placed in the 3D scene as illustrated in Figure 1(a), since the vanishing line S^{h} of the image and the camera height h^{c} from the ground plane have been given. To also place the image plane, the background plane, and the

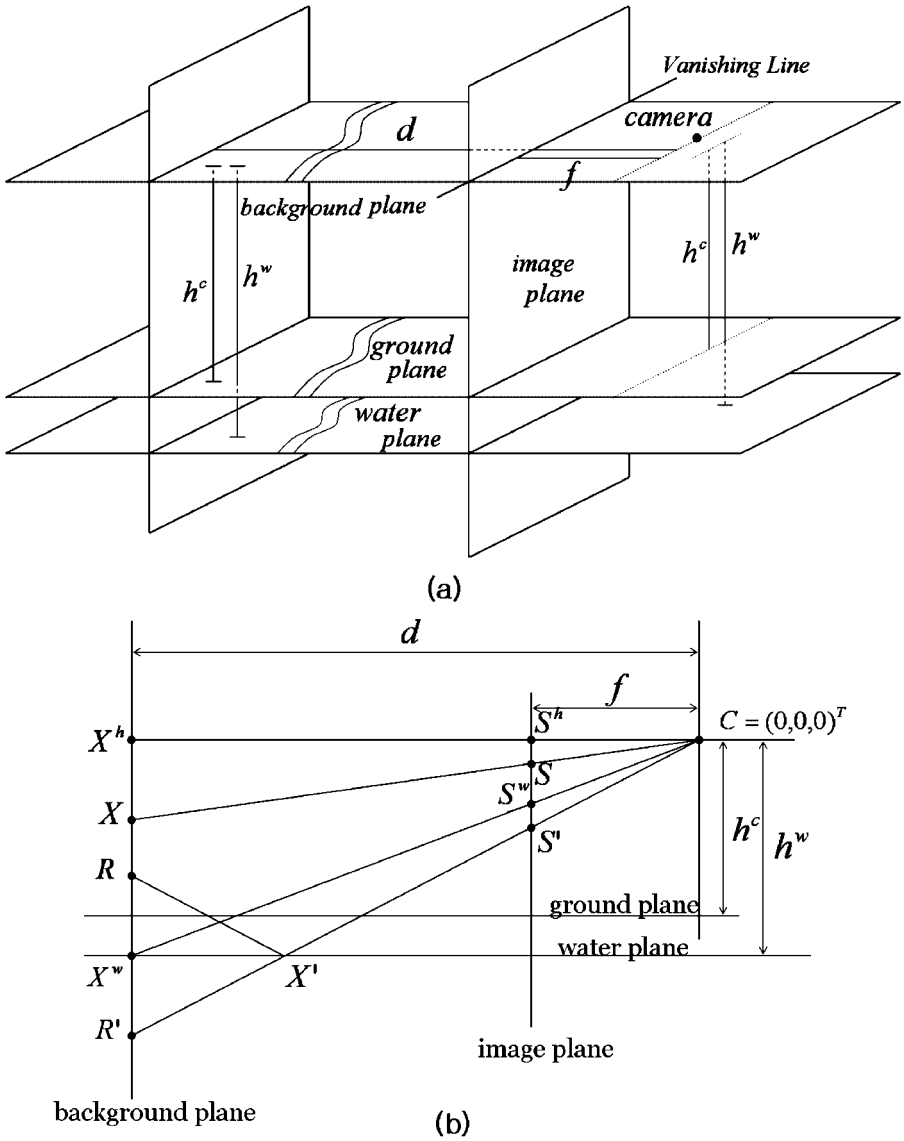


Figure 1. (a) 3D scene model and (b) 2D schematic diagram.

water surface plane, we will automatically extract three 3D scene parameters with respect to the camera position, the focal length f of the camera, the height h^{w} of the camera above the water surface plane, and the Euclidean distance d between the camera and the background plane, provided with a set of ordered pairs of points on the input picture, where each ordered pair consists of a point on the background plane and the corresponding point (reflected point) on the water surface plane.

Let S and S' be a pair of points on the image plane corresponding to the point X on the background plane and its reflected point X' on the water surface plane toward the camera, respectively. Figure 1(b) shows a 2D schematic diagram of the 3D scene model. We define X^{h}, R, X^{w}, and R' as follows:

X^{h} = the projection of the vanishing line from C onto the background plane,

R = the intersection point of the background plane and the ray $\overrightarrow{CX'}$ mirror-reflected at X',

X^{w} = the intersection line between the background plane and the water surface plane, and

R' = the reflection point of R about the water surface plane.

We also denote by S^{h} and S^{w} the projected lines of X^{h} and X^{w} from C onto the image plane, respectively.

Without loss of generality, suppose that the user provides m pairs of points, (S_i, S_i'), $1 \le i \le m$, where $S_i = (x_s^i, y_s^i, f)^{\mathrm{T}}$ and $S_i' = (x_{s'}^i, y_{s'}^i, f)^{\mathrm{T}}$ are the points on the image plane corresponding to a point X_i on the background plane and its reflected point X_i' on the water surface plane, respectively. Then, we can formulate a constrained optimization problem as follows:

$$\underset{d,f,h^{\mathrm{w}}}{\text{minimize}} \sum_i \left\| \left(\frac{x_s^i d}{f} - \frac{x_{s'}^i d}{f}, \frac{y_s^i d}{f} + \frac{y_{s'}^i d}{f} + 2h^{\mathrm{w}}, 0 \right) \right\|^2 \quad \text{subject to } f < d \quad (1)$$

$$y_{s'} d < -f h^{\mathrm{w}} \quad \text{for all } i, \quad \text{and}$$

$$f_1 < f < f_2$$

where f_1 and f_2 are user-specified values that constrain the acceptable range of the focal length f. For details in formulation, please refer to Reference [18]. We solve this problem for f, d, and h^{w} to locate the image plane, the water surface plane, and the background plane in addition to the ground plane specified by the user.

3D Scene Decoration

By placing the three planes as well as the ground plane in the 3D space in accordance with the scene parameters, we have constructed the framework of the 3D scene model, or the global structure of the model. We add local details to the model by locating the foreground objects on the ground plane and the water surface plane, and attaching the three maps to each of scene components. We now describe the map construction procedure for our generalized 3D scene model, while minimally duplicating the results in References [1,3].

Given an input picture, we first interactively extract the texture for every foreground object from the picture. As illustrated in Figure 2, the rectangular sub-picture that is bounded by the red box (specified by the user) is the texture for the object. The boundary of the object is traced manually using an interactive tool.[3] The pixels inside the object are marked with the mask map by turning the corresponding mask bit on. The pixels outside the object are regarded as transparent. After extracting the texture for a foreground object we fill the hole, the portion of the picture hidden by the texture, by using an interactive tool.[19,20] We repeat this process until all foreground objects have been extracted. Instead of image masks, one can adopt Bayesian matting to achieve more precise composition of scene components.[2]

We next extract the texture for each of the three planes. As illustrated in Figure 3(a), let X^{w} and X^{g} be the intersection lines of the background plane with the ground plane and the water surface plane, respectively. By projecting X^{w} and X^{g} onto the image plane, we obtain their corresponding lines, S^{w} and S^{g}. If X^{w} is coincident with X^{g}, then $S^{\mathrm{w}} = S^{\mathrm{g}}$.

As shown in Figure 3(b), we use S^{w} to divide the input picture into two parts: the top and the bottom. The bottom part is the texture for the water surface.

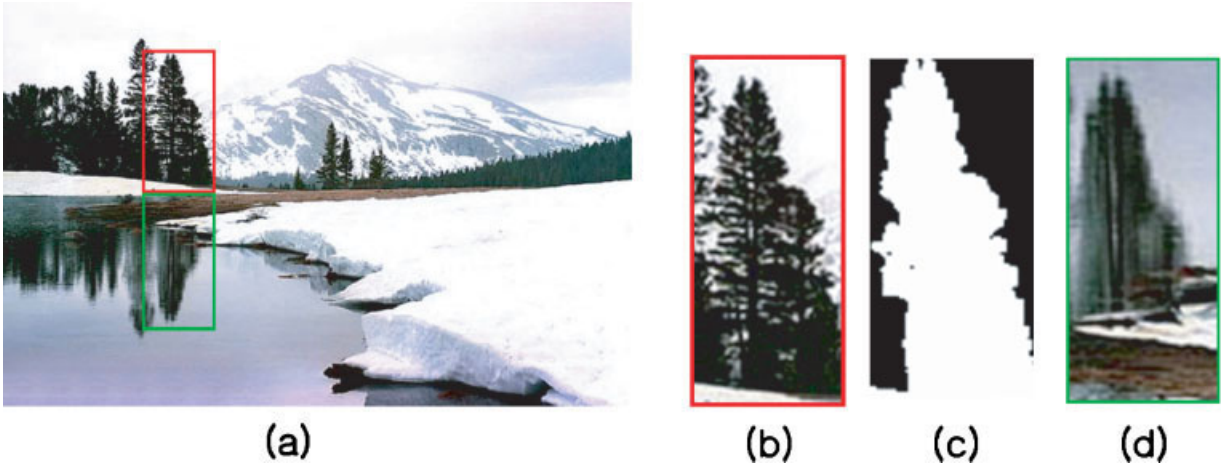


Figure 2. Extraction of foreground objects: (a) input picture, (b) texture map, (c) mask map, and (d) reflection map. Image courtesy of Shad Sluiter.

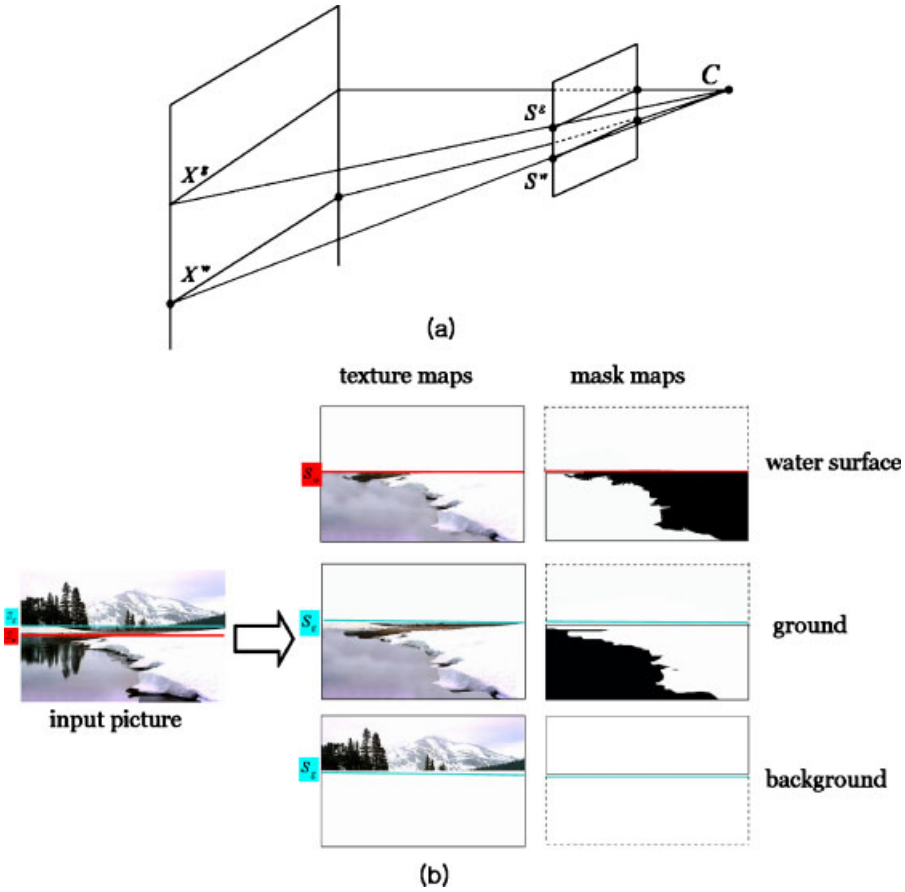


Figure 3. (a) Input picture partitioning and (b) texture and mask maps for the background, the ground, and the water surface.

We interactively trace the boundary between the ground and the water surface, and turn on the mask bit corresponding to each pixel on the water surface. Next, we use S^g to divide the resulting picture into two parts after masking out the pixels that belong to the water surface. The top part is the texture for the background, and the bottom part is that for the ground. We also turn on the mask bit corresponding to each pixel that belongs to the ground. The mask map of the background is obtained trivially by turning the all mask bits on.

We adopt the billboard model[1] to represent a foreground object, in which a billboard is the axis-parallel rectangle in the 3D scene model: The projection of the billboard onto the image plane bounds the texture of the foreground object in the input picture. Since the extracted textures from the input picture are distorted due to foreshortening, we must nullify the distortion to use the textures later for rendering. For the foreground objects and the background, the texture correction can be done by adopting the inverse texture mapping scheme in References [1,3]. As illustrated in Figure 4, this scheme maps a texture back onto its billboard. However, in order to apply the scheme to a texture, the object's geometry should be captured beforehand.

Since the position of background plane is known, we can correct the foreshortening effect trivially for the background texture. For the texture of a foreground object (billboard), however, we need to compute its geometry for this correction. We first project the two corner points on the bottom edge of the texture onto the ground plane (or the water surface plane) to determine the corner point positions. Assuming that the plane containing the object is perpendicular to the ground plane (or the water surface plane), this also gives the depth of the object. The positions of the remaining two corner points are determined by projecting them onto the plane containing the object. Given the geometry, we map the texture back onto the object to obtain an undistorted texture, which is stored in the object's texture map.

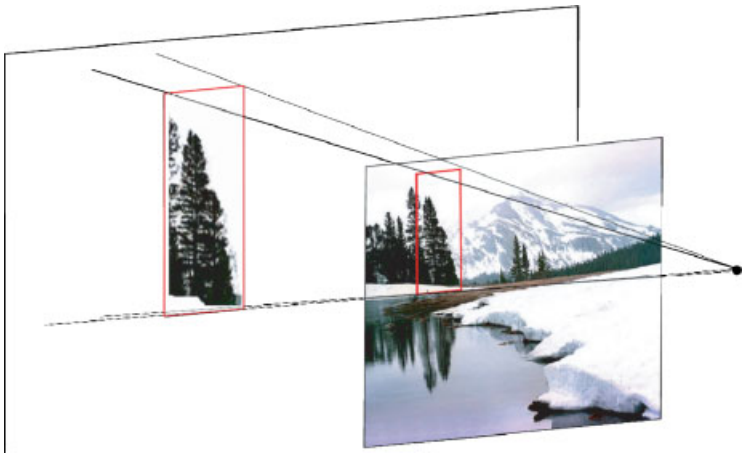


Figure 4. Inverse texture mapping.

The texture of a foreground object or the background is invariant with respect to the camera position. However, the image of the same object on the water surface is determined by light reflection toward the camera, and thus varies from frame to frame depending on the camera position. We build a reflection map that is invariant with respect to the camera position unlike the reflected image on the water surface.

The size of a reflection map is set to the same as that of the corresponding texture map, so that we can establish a natural pixel-by-pixel correspondence between the reflection map and the texture map. Let P'' be a reflected point of P about the water surface plane. As illustrated in Figure 5, the color value of the pixel of the reflection map corresponding to point P on the scene component is sampled from point P' on the water surface plane, at which the ray from the camera to the reflection point P'' makes a mirror reflection toward point P.

The reflection maps for the foreground objects are first created, followed by those for the ground and background planes. Among the foreground objects, the reflection maps are created in depth order from the camera. Whenever a reflection map is created for a foreground object, this object is immediately removed, and the resulting hole on the input picture is filled manually to process the next object.

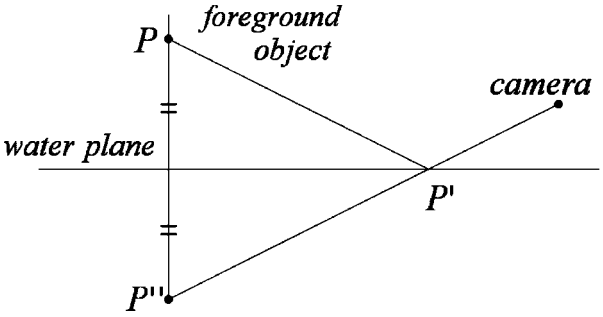


Figure 5. Reflection map construction.

By the way in which a reflection map is created, it contains the color information for reflected images on the water surface for a scene component such as a foreground object and the background. This information is exploited later to synthesize a reflected image on the water surface in accordance with camera movement, while avoiding the complicated process of light interaction with the water surface.

Object Motion

In this section, we describe how we incorporate object movements into TIP, inspired by the work in References [2,14]. We deal mainly with passive objects such as trees, water, boats, and clouds, the movements of which are driven by natural forces such as wind. The one exception is that a boat may also have its own driving force. To make these objects move, stochastic motion textures in Reference [2] are employed. A commercial water simulation tool, RealFlow®[17] is also used to make the water surface move and interact more dynamically with other objects such as the ground, rocks, and boats. For stochastic motion textures, we refer the readers to the work in Reference [2]. Here, we briefly discuss how we exploit the tool, RealFlow® for our purpose.

The local coordinate frame of the water surface is first constructed so that the surface lies on the x-z ($y = 0$) plane

with the x and z coordinate axes intact. The boundary of the water surface (or an interacting object) is traced using its mask map and fitted to a NURBS curve. This curve is reflected about the vertical plane containing the curve end points, to obtain a closed curve. Extruding the latter curve, every foreground object is approximated as a (generalized) cylinder. A module of RealFlow® called RealWave® is employed for water surface simulation. Based on a fractal wave, RealWave® alone gives the movement of the water surface when the surface does not interact with other scene components. This module is equipped with features to control the global water movement with parameters such as water speed, the maximum wave height, and the number of octaves, and so on.

A dynamics solver in RealFlow® is employed to simulate interaction between the water surface and rigid objects. We deal with three types of objects: static, floating, and moving. A static object (e.g., a rock and a bridge) is not affected by water movement although the object itself interacts with the water surface. A floating object (e.g., a boat at anchor) moves passively due to the movement of the water surface. Driven by the user-given external forces and trajectory together with the results from RealWave®, a moving object (e.g., a sailing boat) interacts with the water surface to create dynamic waves propagated from the object.

The output of RealFlow® includes the geometry of the water surface in the local coordinate frame and geometric information on the interacting objects such as positions and orientations. We use the simulation data to animate the water surface and the objects interacting with it.

Rendering

The camera (pinhole) in the 3D scene is moved to synthesize a novel image stream in an on-line manner. In particular, the 3D scene at each frame is rendered, exploiting the three maps that we have built initially while also fixing the focal length f to the initially computed value.

Our choice for rendering is ray tracing. A ray is shot from the camera center to each pixel of the output image. An intersection check is performed between the ray and the scene components in order of the foreground objects, the ground plane, the background plane, and the water surface plane. Among the foreground objects, the intersection check is performed in depth order from the camera.

Suppose that the ray hits a scene component such as a foreground object, the ground, or the background that has a higher priority than the water surface. Let P and P' be the intersection point on this component and its corresponding point on its texture map, respectively. P' can be considered as the texture coordinates of point P. If the component is a moving object, then P' is obtained by canceling the displacement of P made by motion. Therefore, the color value at a pixel of the output image is sampled from the texture map unless the pixel's mask bit nearest to point P' is turned off. In particular, the color values of the four nearest pixels to point P' is bilinearly interpolated to acquire the color value of the output pixel. If the mask bit is turned off, the ray proceeds further into the 3D scene to repeat the intersection check.

Now, suppose that the ray hits the water surface. Let P and P' be defined in the same manner as given above. If the mask bit of the nearest texture pixel to P' is turned on, then the ray from the camera to the output pixel is mirror-reflected about the normal vector at P. The normal vector at P does not necessarily point in the global-up direction, since this vector could have been perturbed by the movement of the water surface. Now, the mirror-reflected ray takes over the role of the ray from the camera. Specifically, the mirror-reflected ray is traced to determine the color of the output pixel in the same way as the original ray is traced, except that the reflection map of each scene component substitutes for its texture map. For a moving object hit by the mirror-reflected ray, the displacement of each point by motion is nullified to access the reflection map for color sampling.

The original ray as wells as the reflected ray may not hit any scene component at a point with its corresponding mask bit turned on, since the size of the input picture is finite. In this case, we randomly sample a color from the texture map of the background plane. When the reflected ray does not hit a scene component, the average color value of the neighborhood points of the point P on the water surface or the color value of the nearest point to P may also be sampled for the missing ray.

Results

For experiments, we used four pictures with reflected images on water surfaces. The second and third pictures are paintings and the others are photographs, as shown in Figure 6. The size of every picture was reduced to one tenth of its original size. The experiments were performed on an Intel Pentium® PC (P4 3.0 GHz processor,

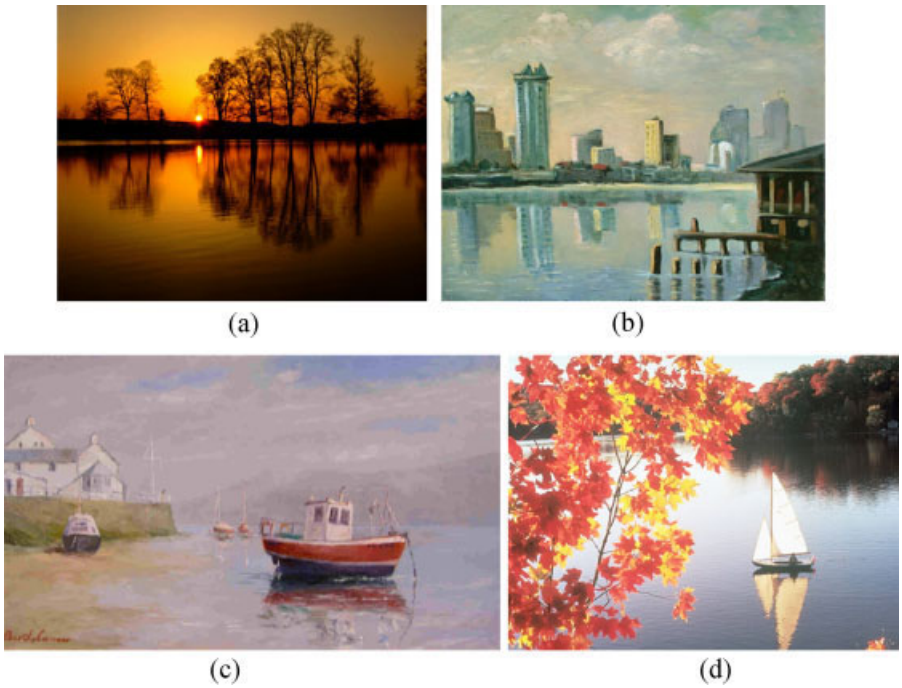


Figure 6. Input pictures. Images courtesy of edenpics.com(a), Thavibu Gallery(b), Peter Bartholomew(c), and Korea Computer Magazine(d), respectively.

2.0 GB RAM, and GeForce® FX 5950 graphics card). It took from several minutes to several hours to generate an animation for each picture, depending on time for manual interaction. The most time-consuming part was the foreground object extraction and hole filling even with interactive tools. After preprocessing, our method generated more than 10 frames per second for all 4 pictures. The performance of the method depends on the scene complexity, in particular, the number of foreground objects. The test scenes have at most five foreground objects (Figure 6(b)).

The animation results are given in the accompanying movie file. We can observe that reflected images on water surfaces vary convincingly from frame to frame in accordance with the camera position changes as well as object movements. In each experiment, we assumed that an image initially has a flat water surface. The first two experiments (Figure 6(a) and (b)) were for scenes without any moving foreground objects, and the others were with moving foreground objects (Figure 6(c) and (d)). Stochastic motion textures were employed to make foreground objects on the ground move. For the water surface and the objects on it, RealFlow® was used to make them move.

The conjugate gradient method in MATLAB® was employed to solve the non-linear optimization problem of estimating the 3D scene parameters with respect to the camera position. Five pairs of points for each picture were taken as the input data for the optimization. The solver immediately converged to a good local minimum with a reasonable initial guess for the parameters. Empirically, we found that the initial guess $(f, h^{\mathrm{w}}, d)^{\mathrm{T}} = (5200, 5000)^{\mathrm{T}}$ works well. We set $C = (0, 0, 0)^{\mathrm{T}}$ and $h^{\mathrm{c}} = 180$. Results for parameter estimation are available in Reference [18].

Discussion

In this section, we discuss limitations and weaknesses of the proposed method. As a generalization of TIP, the proposed method shares the inherent limitations of its predecessors.[1,3] The foreground objects are represented as billboards, which requires a special care during navigation. This can partially be solved by rotating a billboard in accordance with the time-varying camera position so that the billboard always faces the camera.[21,22]

The color information in the texture and reflection maps gives the sum of the color contributions by view-dependent and -independent light interactions with the environment captured at the initial camera position. In principle, the view-dependent contribution changes as the camera (or an object) moves. However, we use the

initially captured maps regardless of the camera (object) movement. Although our method gives pleasing walk-through (fly-through) animations, the illumination model is not quite correct in a physical sense, which is an inherent limitation of TIP.

With a single picture as an input, manual interactions by the user for scene modeling are also inherent in TIP. To alleviate the user's burden, an optimization formulation was proposed to find the 3D scene parameters with respect to the camera position. Since the objective function is non-linear with local minima, a reasonable initial guess is needed for a good solution. The water surface is assumed to be flat initially in the proposed formulation although it can be moved after extracting the reflection maps. With this assumption, the initial normal vector of the water surface is trivially obtained. Otherwise, it would be necessary to estimate the normal vector field that matches the 2D image of the water surface in the input picture, which is hard, if not impossible, with a single input picture.

Conclusions

We generalized TIP by incorporating water reflection and object movements. Technically, our contributions are twofold: First, a non-linear formulation was presented to find the three 3D scene parameters with respect to the camera position: the focal length of the camera, its height above the water surface, and the distance of the background plane from the camera. Next, a novel image-based scheme was proposed to generate a view-dependent image stream with the reflected images of the 3D scene on the water surface. As shown in experiments, the proposed approach has worked well even with both objects and the water surface in motion. We leave the issues raised in Discussion Section as future research topics.

ACKNOWLEDGMENTS

We appreciate the anonymous reviewers for their constructive comments and suggestions. This research was supported by the Korea Science and Engineering Foundation, and the Brain Korea 21 Program of Ministry of Education & Human Resources Development.

References

1. Horry Y, Anjyo K, Arai K. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proceedings of SIGGRAPH 1997*, 1997, pp. 225–232.
2. Chuang YY, Goldman DB, Zheng KC, Curless B, Salesin DH, Szeliski R. Animating pictures with stochastic motion textures. In *Proceedings of SIGGRAPH 2005*, 2005, pp. 853–860.
3. Kang H, Pyo SH, Anjyo K, Shin SY. Tour into the picture using a vanishing line and its extension to panoramic images. In *Proceedings of EuroGraphics 2001*, 2001, pp. 132–141.
4. Chen SE. QuickTime VR—an image-based approach to virtual environment navigation. In *Proceedings of SIGGRAPH 1995*, 1995, pp. 29–38.
5. Chen SE, Williams L. View interpolation for image synthesis. In *Proceedings of SIGGRAPH 1993*, 1993, pp. 279–288.
6. Darsa L, Silva BC, Varshney A. Navigating static environments using image-space simplification and morphing. In *Proceedings of Symposium on Interactive 3D Graphics 1997*, 1997, pp. 25–34.
7. Gortler SJ, Grzeczuk R, Szeliski R, Cohen MF. The lumigraph. In *Proceedings of SIGGRAPH 1996*, 1996, pp. 43–54.
8. Liebowitz D, Criminisi A, Zisserman A. Creating architectural models from images. In *Proceedings of EuroGraphics 1999*, 1999, pp. 39–50.
9. Levoy M, Hanrahan P. Light field rendering. In *Proceedings of SIGGRAPH 1996*, 1996, pp. 31–42.
10. Lippman A. Movie maps: an application of the optical videodisc to computer graphics. In *Proceedings of SIGGRAPH 1980*, 1980, pp. 32–43.
11. McMillan L, Bishop G. Plenoptic modeling: an image-based rendering system. In *Proceedings of SIGGRAPH 1995*, 1995, pp. 39–46.
12. Miller G, Hoffert E, Chen SE, *et al*. The virtual musuem: interactive 3D navigation of a multimedia database. *The Journal of Visualization and Computer Animation*, 1992, pp. 183–197.
13. Sloan PP, Cohen MF, Gortler SJ. Time critical lumigraph rendering. In *Proceedings of Symposium on Interactive 3D Graphics 1997*, 1997, pp. 17–23.
14. Kang H, Shin SY. Creating walk-through images from a video sequence of a dynamic scene. *Presence* 2004; **13**(6): 638–655.
15. Hoiem D, Efros AA, Hebert M. Automatic photo pop-up. In *Proceedings of SIGGRAPH 2005*, 2005, pp. 577–584.
16. Zhang L, Dugas-Phocion G, Samson J-S, Seitz SM. Single view modeling of free-form scenes. *IEEE Conference on CVPR*, 2001, pp. 990–998.
17. RealFlow. Nextlimit Techonologies. www.nextlimit.com, 2005.
18. Park J, Heo N, Choi S, Shin SY. Tour into the picture with water surface reflection and object movements. http://cg.kaist.ac.kr/~c2alpha/, 2006.
19. Adobe Photoshop. Adobe Systems Inc. http://www.adobe.com, 2005.
20. Criminisi A, Perez P, Toyama K. Object removal by exemplar-based inpainting. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition(CVPR) 2003*, 2003, pp. 721–728.
21. Qin X, Nakamae E, Tadamura K, Nagai Y. Fast photo-realistic rendering of trees in daylight. *Computer Graphics Forum*, 2003, pp. 243–252.
22. Renderman. *Pixar. renderman.pixar.com*, 2005.

Authors' biographies:

Jinho Park received his B.S. and M.S. degrees in applied mathematics from Korea Advanced Institute of Science and Technology in 1999 and 2001, respectively. He is a Ph.D. student in the Department of Electrical Engineering and Computer Science at the Korea Advanced Institute of Science and Technology, Daejeon, Korea. His research interests include image based rendering and fluid simulation.

Nambin Heo received his B.S. degree from Computer Science department from Korea Advanced Institute of Science and Technology in 2004. He is a M.S. student in the Department of Electrical Engineering and Computer Science at the Korea Advanced Institute of Science and Technology, Daejeon, Korea. His research interest include fluid simulation, images-based rendering.

Sunghee Choi is an assistant professor of computer science at Korea Advanced Institute of Science and Technology (KAIST). Her research interests include computational geometry, computer graphics and visualization. Choi has a Ph.D. in computer science from the University of Texas at Austin.

Sung Yong Shin received his B.S. degree in industrial engineering in 1970 from Hanyang University, Seoul, Korea, and his M.S. degree and Ph.D. in Industrial and Operations Engineering from the University of Michigan in 1983 and 1986, respectively. He is a professor in the Department of Electrical Engineering and Computer Science at the Korea Advanced Institute of Science and Technology, Daejeon, Korea. He has been working at KAIST since 1987. At KAIST, he teaches computer graphics and computational geometry. He also leads a computer graphics research group that has been nominated as a national research laboratory by the Government of Korea. His recent research interests include motion capture and retargeting, performance-based computer animation, real-time rendering, and collision detection. He is currently editors of *IEEE TVCG*, *Graphical Models*, *The Visual Computer*, and *Computer Animation & Virtual Worlds*.