Motion Consistency for Image-Based Rendering

Philippe Simard, Marcel Mitran and Frank P. Ferrie

Centre for Intelligent Machines
McGill University
3480 University Street Room 410, Montreal, Canada, H3A 2A7
Email: {psimard,mmitran,ferrie}@cim.mcgill.ca

Abstract

This paper presents an enhancement to current image-based rendering methods. It suggests that motion consistency can be used in a real-time dynamic rendering system to increase computational efficiency up to one order of magnitude. Parameterization of the image warping function with respect to the viewer's motion can be used to provide an efficient method for extrapolating new viewpoints using a single rendered image. The system is successfully tested in the context of a flight simulator using a first order parameterization.

1 Introduction

Traditional real-time, geometric-based rendering techniques often require a pipeline of dedicated hardware to quickly perform tasks such as scan conversion and shading [6]. Such conventional rendering systems are limited by their ability to generate realistic scenes for two reasons: computer models still fail to offer photo-realistic representations, and even if sufficiently detailed models were available, rendering would be inherently slowed due to the direct relationship between scene-complexity and rendering rates.

Image-based rendering (IBR) offers an alternative to geometric based rendering. IBR algorithms achieve photo-realistic images while reducing the complexity observed in traditional geometric rendering.

Texture mapping was the earliest form of IBR. This method allows detailed planar images to be mapped to three-dimensional surfaces in the model space, thus producing detailed scenes while rendering a reduced set of polygons. Texture mapped scenes can be described as geometric models augmented with image information. As such, this rendering process has the same complexity as a raster scan pipeline.

Later forms of IBR perform image interpolation [3,9,17]. These methods assume that pixel correspondence and the associated motion are known for two or more views. These rendering processes are bound by image resolution instead of the model complexity. The main disadvantage of image interpolation is that it is not causal. For an interactive system, many views need to be rendered *a priori*. Thus a predictive strategy must be adopted to render possible paths taken by the viewer.

More recent approaches to IBR extrapolate new views by augmenting images with depth information (using machine vision methods [4], or geometrical models) [12,13,17]. As such, novel points of view can be generated without re-rendering the model. This is achieved by projecting the 3D motion vectors of each depth pixel onto the image plane to compute the pixel translations commonly referred to as *optical flow* [8,11]. The optical flow is used to warp the original image into a new view.

There have been several additions to depth-augmented image rendering algorithms that take advantage of spatial consistency [10,17]. This paper is novel in suggesting that consistency in the viewer's motion can also be used to improve the depth image representation. As such, images are augmented with a viewer centered parametric description of the flow-field rather than just depth.

An increase in efficiency of up to one order of magnitude is demonstrated. The suggested primitive provides a versatile and causal representation of both the image and the desired motion. It requires a single rendered

view of any desired geometric complexity to render any number of novel views with image resolution complexity.

The remainder of this paper is structured as follows: Section 2 reviews key elements of current literature in the areas of IBR. Section 3 derives and discusses the flow parameterization model. Section 4 introduces a flight simulator test environment and qualifies the model. Finally, the paper concludes in Section 5.

2 Previous Work

Early work by Laveau and Faugeras [9] interpolate images given pixel correspondence between two known views. Using projective geometry, a novel view can be constructed by correctly computing the flow vectors along the epipolar lines.

Chen and Williams [3] show that Laveau and Faugeras' work can be linearized. They use linear scaling of the flow vectors between two images to produce an intermediate view. This work demonstrates that a linear approximation of the perspective projection model works well if the change in viewpoint is small. Similarly, Seitz and Dyer [15,16] use principles of projective geometry to constrain image morphing.

McMillan [12, 13] describes the framework for image extrapolation from depth-augmented images. Much of this work considers traditional graphics issues, such as occlusion and interpolation, in the context of IBR.

Legyel and Snyder [10] introduce *sprites* in the context of the Talisman warp engine. This primitive takes advantage of spatial consistency for parameterizing the depth images. This consists of grouping smooth surfaces together into planes. The planes can then be augmented with depth values or simply texture mapped with the original image.

Shade *et al.* [17] show that spatial consistency can be used to provide a computationally efficient framework. As such, an incremental approach to rendering along scan-lines is introduced.

3 Parametric Motion Flow

This section presents a framework for improved computational efficiency in IBR by building on the work of Chen and Williams, McMillan and Shade *et al*. It introduces a representation that avoids the motion prediction necessary in image interpolation techniques while providing improved efficiency for current extrapolating IBR algorithms.

It is widely accepted that for real-time model exploration there is much temporal consistency as well as spatial consistency between frames. Video coding techniques [1,7,18] take advantage of this. As described earlier, Shade *et al*. have shown how spatial consistency can be used to speed up the rendering. This paper provides a similar approach that capitalizes on viewer motion consistency instead of just scene consistency. Thus, a parametric model for computing the flow field of the image with respect to the viewer's motion will be derived.

A linear parameterization is used in this paper for two reasons: Chen and Williams have already shown that a linear approximation works well for interpolating images, and, as this model is of the first order, it demonstrates rigidity for the coarsest approximation.

3.1 Optical Flow

The notation used in this paper refers to points in 3D space using capital letters. Points in the camera's image plane are denoted using lower case characters. Bold characters indicate homogeneous motion operators. A subscript *i* is used to denote which iteration of the camera motion is being considered.

A pinhole camera of focal length *f* is assumed and a viewer-based coordinate system is adopted (Figure 1). The origin is at the focal point of the camera. The image plane is at Z= *f*. The Z-axis runs along the optical axis, and the X- and Y-axes are parallel to the x- and y-axis of the image plane respectively.

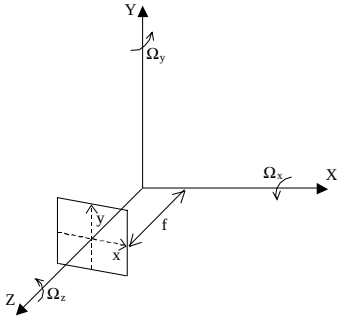

Figure 1: Viewer centered coordinate system

The ego-motion of the camera is decomposed into a rotation about an axis passing through the origin, and a translation. Any three-dimensional motion can be represented as such [5]. This is denoted as ($\mathbf{T}_i \circ \mathbf{R}_i$), where $\circ$ indicates a composite function.

Given a point in three-dimensional space, (X_i,Y_i,Z_i), and a camera motion, $(T_x,T_y,T_z)_i \circ (R_x,R_y,R_z)_i$, the new location of the point, $(X_{i+1},Y_{i+1},Z_{i+1})$, is given by (1). This homogenous operator assumes the small rotation approximation, $\sin\theta = \theta$,

$$\begin{bmatrix} X_{i+1} \\ Y_{i+1} \\ Z_{i+1} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & -\Omega_z & \Omega_y & T_x \\ \Omega_z & 1 & -\Omega_x & T_y \\ -\Omega_y & \Omega_x & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}. \quad (1)$$

A second system of equations (2) provides a perspective projection model for the pinhole camera with focal length f,

$$\begin{bmatrix} x_i \\ y_i \\ f \\ f/Z_i \end{bmatrix} = \begin{bmatrix} f/Z_i & 0 & 0 & 0 \\ 0 & f/Z_i & 0 & 0 \\ 0 & 0 & f/Z_i & 0 \\ 0 & 0 & 0 & f/Z_i \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}. \quad (2)$$

From (1) and (2), a system of equations for the new location, (x_{i+1}, y_{i+1}, f), of a point in the image, (x_i, y_i, f), after the viewpoint has moved can be derived [2,12]:

$$x_{i+1} = \frac{f A_i}{C_i},$$

$$y_{i+1} = \frac{f B_i}{C_i}, \quad (3)$$

where,

$$A_i = x_i - \Omega_z y_i + f\,\Omega_y + \frac{T_x f}{Z_i},$$

$$B_i = y_i + \Omega_z x_i - f\,\Omega_x + \frac{T_y f}{Z_i},$$

$$C_i = \Omega_x y_i - \Omega_y x_i + f + \frac{T_z f}{Z_i}. \quad (4)$$

3.2 First Order Parameterization

A first order parameterization of the flow field is used to highlight how temporal consistency in the warping function can be used. The first two terms in the Taylor expansions of (3) are,

$$\begin{aligned} \tilde{x}_{i+1} = {} & x_{i+1}(T_{x0}, T_{y0}, T_{z0}, \Omega_{x0}, \Omega_{y0}, \Omega_{z0}) \\ & + \frac{\partial x_i}{\partial T_x}(T_x - T_{x0}) \\ & + \frac{\partial x_i}{\partial T_z}(T_z - T_{z0}) + \frac{\partial x_i}{\partial \Omega_x}(\Omega_x - \Omega_{x0}) \\ & + \frac{\partial x_i}{\partial \Omega_y}(\Omega_y - \Omega_{y0}) + \frac{\partial x_i}{\partial \Omega_z}(\Omega_z - \Omega_{z0}). \end{aligned} \quad (5)$$

This can be re-expressed as,

$$\begin{aligned} \tilde{x}_{i+1} = {} & \frac{A_0}{C_0} + \frac{f^2 T_x}{Z_i C_0} - \frac{f\,y_i \Omega_z}{C_0} - \frac{f^2 A_0}{Z_i C_0^2}(T_z - T_{z0}) \\ & - \frac{f A_0 y_i}{C_0^2}(\Omega_x - \Omega_{x0}) - \left(\frac{f^2}{C_0} + \frac{f A_0 x_i}{C_0^2}\right)(\Omega_y - \Omega_{y0}), \end{aligned} \quad (6)$$

which can be written in the following form,

$$\begin{aligned} \tilde{x}_{i+1} = {} & OP_x + \Delta_{Tx} T_x + \Delta_{xTz} T_z \\ & + \Delta_{x\Omega x}\Omega_x + \Delta_{x\Omega y}\Omega_y + \Delta_{x\Omega z}\Omega_z, \end{aligned} \quad (7)$$

where:

$$\begin{aligned} & OP_x = \frac{A_0}{C_0} + \frac{f^2 A_0 T_{z0}}{Z_i C_0^2} + \frac{f A_0 y_i \Omega_{x0}}{C_0^2} - \left(\frac{f^2}{C_0} + \frac{f A_0 x_i}{C_0^2}\right)\Omega_{y0}, \\ & \Delta_{Tx} = \frac{f^2}{Z_i C_0}, \quad \Delta_{xTz} = \frac{-f^2 A_0}{Z_i C_0^2}, \quad \Delta_{x\Omega x} = \frac{-f A_0 y_i}{C_0^2}, \\ & \Delta_{x\Omega y} = \left(\frac{f^2}{C_0} + \frac{f A_0 x_i}{C_0^2}\right) \quad \text{and} \quad \Delta_{x\Omega z} = \frac{-f\,y_i}{C_0}. \end{aligned} \quad (8)$$

Similarly,

$$\tilde{y}_{i+1} = OP_y + \Delta_{Ty} T_y + \Delta_{yTz} T_z + \Delta_{y\Omega x} \Omega_x + \Delta_{y\Omega y} \Omega_y + \Delta_{y\Omega z} \Omega_z, \quad (9)$$

where,

$$OP_y = \frac{B_0}{C_0} + \frac{f^2 B_0 T_{z0}}{Z_i C_0^2} - \frac{f B_0 x_i \Omega_{y0}}{C_0^2} + \left(\frac{f^2}{C_0} + \frac{f B_0 y_i}{C_0^2} \right) \Omega_{x0},$$

$$\Delta_{Ty} = \frac{f^2}{Z_i C_0}, \quad \Delta_{yTz} = \frac{-f^2 B_0}{Z_i C_0^2}, \quad \Delta_{y\Omega x} = \frac{f B_0 x_i}{C_0^2}, \quad (10)$$

$$\Delta_{x\Omega y} = -\left(\frac{f^2}{C_0} + \frac{f B_0 y_i}{C_0^2} \right) \quad \text{and} \quad \Delta_{y\Omega z} = \frac{f x_i}{C_0}.$$

Equations (7) and (9) represent a linear parametric description of the image-flow field, (x_{i+1}, y_{i+1}, f). Twelve parameters are used: two are constants and the ten others are scale factors. The constants *OPx* and *OPy* represent the motion velocities about which the flow equations were linearized. The delta terms represent scale factors for the contribution of each motion parameter. Thus, by scaling the contribution of each motion parameter, novel views can be generated.

3.3 Sources of Errors

The parametric flow primitive introduces two forms of error. The first comes from the small rotation approximation made in the three-dimensional motion operator (1). The second is the result of linearizing the different hyperbolic components of (3).

The rotational approximation is negligible under most conditions as it is reasonable to assume small rotations between frames for an observer given a realistic frame rate. Also, such an error is easily bounded.

The error introduced by linearizing the flow-field is small. Only three of the six degrees of freedom contribute to the error when linearized. This is because translations parallel to the image plane (T_x and T_y) and rotations about the camera's viewing axis (R_z) scale the flow field linearly.

Section 4 will qualify the error in the context of a flight simulator. The six degrees of motion will be tested individually. A realistic full flight path will also be produced.

3.4 Increase in Performance

This section compares the computational performance of the suggested linear flow method with that of standard depth image rendering for an N image sequence. Multiplication and division are considered equivalent, as are addition and subtraction.

The standard depth image method [12], when ignoring spatial consistencies*, requires computing equations (3) for each image in the sequence. The total number of operations, when optimized, is Nx(13 multiplications + 9 additions) per pixel.

The linearized flow method requires full computation of A_i, B_i and C_i (4), as well as the scale parameters (8) and (10) for the first image in the sequence. This results in an additional 34 multiplications + 9 additions with respect to the depth image method.

Assuming that the motion of the observer is held constant (no acceleration), the scale parameters need not be multiplied by the motion parameters after the first image. The different contributions can be added or discarded at no computational cost when generating novel views. Thus the flow equation can then be reduced to a maximum of 8 additions per pixel for the N-1 subsequent frames. Under such conditions, the linear flow method proposed becomes advantageous over the depth image method for N > 3.

Given that some acceleration is probably desired, a maximum of two extra multiplications per degree of freedom accelerated are required (only one for T_x and T_y). For the worst-case, ten extra multiplications per image per pixel are required. This is an equivalent complexity to traditional depth image extrapolation.

It is shown in the next section that, for the implementation of a flight simulator, the

* Note that the incremental method of [17] can be implemented on top of the motion consistency method presented, as temporal and spatial consistencies are independent. Spatial consistency was ignored to allow for better gauging of motion consistency.

linearized flow model allows for image sequences for which N is between 10 and 30. Thus, this method increases the frame rate three to ten times.

4 Flight Simulator

A dynamic exploration system was constructed under the premise that an IBR primitive should be used in conjunction with a geometrical rendering engine. The IBR should offload the graphics engine as much as possible. Refresh events are triggered when the predicted linearized flow error exceeds a given threshold. This results in a new base image being generated and new flow parameters being computed.

4.1 The Simulation Environment

A flight simulator was used as a test environment to measure the error introduced by the linearized flow primitive. The simulator was implemented in C++ using an OpenGL rendering package. A hilly terrain with forested zones was used as an environment model (Figure 2). Such a terrain provided a wide range of depth values for testing the linear flow primitive.

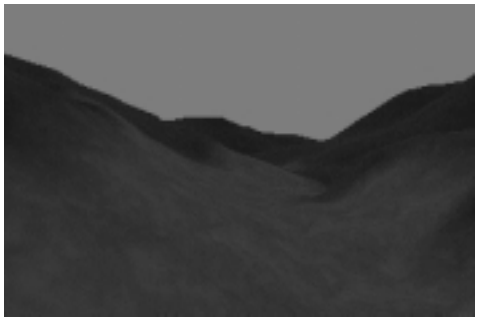

Figure 2: Sample image of flight simulator

The ground truth optical flow was computed using the depth image primitives and true homogenous transformations. Thus the small rotation approximation was removed in this case.

Both the depth image and parametric flow methods were implemented to display up to pixel accuracy only. Linear interpolation was used in the case of forward motion to reduce the effects of pixel stretching. Popescu *et al.* [14] suggest a method for pixel warping that is more appropriate for a full implementation of such a system.

The disocclusion problem, typical to all IBR methods, was not dealt with in this system. Layered depth images [17] could have been used. This would have simply increased processing time for both IBR methods equivalently while providing no additional insight into how well the parametric linear flow model works.

The optical flow was computed to floating point precision for the purpose of error measurement. The mean-square-error (*MSE)* was computed between ground truth pixel translations, *(DIx, DIy)* and the parameterized linear model *(LFx, LFy)* for each rendered image *i*,

$$MSE_i = \sum_{\text{Image}} \sqrt{(LFx - DIx)^2 + (LFy - DIy)^2}\,. \quad (11)$$

The six degrees of freedom of the viewer were separately tested with six different motion sequences (Table 1). This allowed gauging of the error introduced by the linearized flow primitive.

Table 1: Velocities parameters for motion of observer

Motion	Km/h	Motion	Deg/sec
T_x	100	R_x	3
T_y	100	R_y	3
T_z	200	R_z	9

A realistic flyby sequence was also generated. Simultaneously motions along multiple degrees of freedom and accelerations were applied (Table 2). The velocity of the viewer in each motion sequence was set to reflect realistic values for a helicopter flying at low altitude over a terrain. A resolution of 320x240 was assumed.

Table 2: Velocities for full motion

Motion	Km/h	Motion	Deg/sec
T_x	0	R_x	0.6
T_y	12.5	R_y	3.0
T_z	100-150	R_z	3.0

The system was run with an open loop control for the base-image refresh rates (fixed refresh rate). The refresh rate, maximum flow error and computational gain are provided in Table 3. Computational gain is measured with respect to standard depth image methods. A maximum *MSE* of half a pixel was chosen to provide smooth frame transitions when refreshing the base image.

Flow parameterization values were computed for half the expected displacement for each degree of freedom, given the previous refresh rate. This minimized the error resulting from linearizing of the hyperbolic components. Thus parabolic errors were expected for these motions.

A closed loop control system, based on a predicted *MSE*, was used to automatically update the refresh rate and parameterization values for the full motion sequence.

4.2 Error Measurements

The results for four of the seven sequences are displayed graphically in Figure 3, Figure 4, Figure 5 and Figure 6. The *MSE* was computed for each rendered frame. Stars indicate when the model parameters were refreshed and a new base frame was rendered geometrically.

The results of each motion sequence are summarized in Table 3. Note that the only motion that introduced serious levels of error was the translation along the Z-axis. This motion provided an increase in performance of four. For all other forms of motion (excluding the full motion), an increase in performance of at least six was achieved. The performance was most enhanced for rotations about the x-axis, where an increase of a full order of magnitude was observed.

Table 3: Compilation of observed error results

Motion	Refresh (avg)	Max MSE	Comp. Gain
T_x	20	0	6.7
T_y	20	0	6.7
T_z	12	0.63	4.0
R_x	30	0.16	10.0
R_y	20	0.16	6.7
R_z	20	0.41	6.7
Full	14.3	0.60	3.4

As expected, translations parallel to the image plane (T_x and T_y) resulted in negligible error. Thus the error graphs are omitted from the paper. Translation perpendicular to the image plane (T_z) produced the greatest linearization error (Figure 3). The flight simulator system required refreshing the base image and scale parameters every twelve frames to ensure that the *MSE* remained reasonably close to the desired half-pixel threshold.

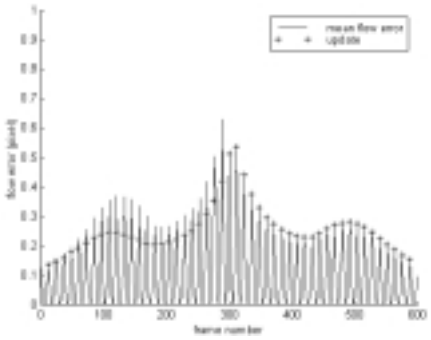

Figure 3: MSE for T_z

Rotations about the x- and y-axis were identical. Thus, they produced the same error function. Little linearization error was observed for these rotations (Figure 4). A maximum *MSE* of 0.17 pixels resulted when refreshing the image every thirty frames. The rotation about the y-axis was refreshed every twenty frames due to greater disocclusions along the horizontal edges.

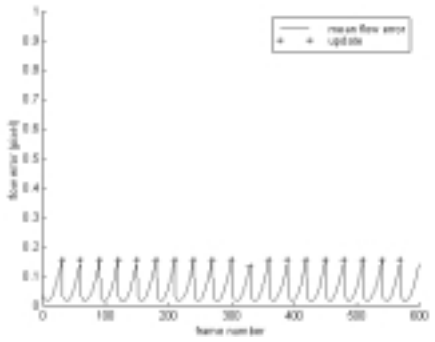

Figure 4: MSE for R_x and R_y

Rotations about the z-axis produced some error due to the small rotation assumption. For a roll of 9 deg/sec the base image and parameterization values were refreshed every twenty frames (Figure 5).

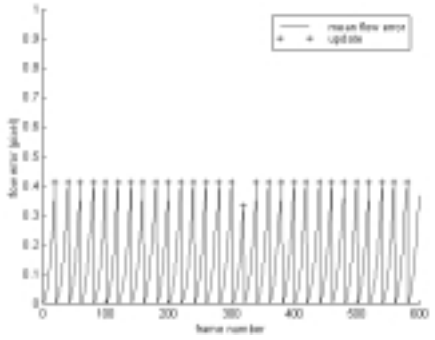

Figure 5: MSE for R_z

The full motion sequence provided a realistic flight path near the terrain. Coarse closed loop control was applied for refreshing the base image and flow parameters. The average refresh rate was every 14.3 frames (Figure 6). A computational gain of 3.4 was observed. This demonstrates that for a realistic sequence, in which the parameterization error is compounded, a significant increase in computational efficiency can be obtained from motion consistency.

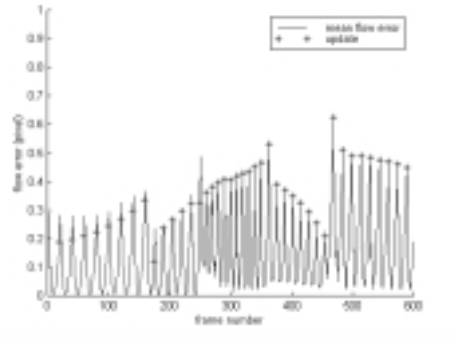

Figure 6: MSE for full motion

5 Conclusion

This paper described an enhancement to current extrapolating image-based rendering algorithms. The framework is developed in the context of a real-time dynamic environment. The algorithm is based on ideas presented in previous work on image-based rendering. Prior work shows that interpolation of flow can be used to generate new viewpoints. Interpolation methods are, however, not causal. Current depth-image rendering techniques provide a causal representation but do not take advantage of consistencies in the motion of the viewer.

Thus a generalized representation is introduced. This approach augments pixels with the individual flow contributions of the six degrees of freedom of the viewer. This provides a parameterization of the flow field. A linear approximation of the flow equations is chosen to demonstrate that consistency of motion can be used to accelerate the IBR process. Three of the six degrees of freedom turn out to be linear. The other three motion contributions are linearized to provide a more efficient computational structure while introducing little error.

This new method was shown to provide increased efficiency. An improvement of up to a full order of magnitude was demonstrated, while causality was maintained. The algorithm was tested in the context of a helicopter flight simulator. Realistic values for the motion of the helicopter were used to test error contribution of each motion. A real low

altitude flight path was also generated and tested. Results showed that the linearized flow method increased the frame rate up to ten times for a maximum pixel flow error of half a pixel.

References

[1] Agrawala, M., Beers, A.C., and Chaddha, N., "Model-Based Motion Estimation for Synthetic Animations", *SIGGRAPH '95*, pp. 477-488, 1995.

[2] Carlsson, S. and Eklundh J.O., "Object Detection using Model Based Prediction and Motion Parallax", *1st European Conference on Computer Vision*, pp. 297-306, 1990.

[3] Chen, S.E., and Williams, L., "View Interpolation for Image Synthesis", *SIGGRAPH '93*, pp. 279-288, 1993.

[4] Debevec, P.E., Taylor, C.J., and Malik, J., "Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach", *SIGGRAPH '96*, pp. 11-20, 1996.

[5] Fang, J.-Q. and Huang T.S., "Solving Three Dimensional Small-Rotation Motion Equations", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 253-258, 1983.

[6] Foley, J.D., van Damn, A., Feiner, S.K., and Huges, J.F., "Computer Graphics: Practices and Principles 2nd Edition", *Addison Wesley*, 1990.

[7] Guenter, B. K., Yun, H.C., and Mersereau, R.M., "Motion Compensated Compression of Computer Animation Frames", *SIGGRAPH '93*, pp. 297-304, 1993.

[8] Horn, B.K., "Robot Vision", *MIT Press*, 1986.

[9] Laveau, S., and Faugeras, O., "3-D Scene Representation as a Collection of Images", *Proceedings of the International Conference on Pattern Recognition*, pp. 689-691, 1994.

[10] Lengyel, J. and Snyder, J., "Rendering With Coherent Layers", *SIGGRAPH '97*, pp. 233-242, 1997.

[11] Longuet-Higgins, H.C. and Prazdny K., "The Interpolation of a Moving Retinal Image", *Proc. Royal Society London B*, no.208, pp. 385-397, 1980.

[12] McMillan, L., and Bishop, G., "Plenoptic Modeling: An Image-Based Rendering System", *SIGGRAPH '95*, pp. 39-46, 1995.

[13] McMillan, L., "An Image-Based Approach to Three-Dimensional Computer Graphics", *PhD thesis*, University of North Carolina at Chapel Hill, 1997.

[14] Popescu, V., Eyles, J., Lastra, A., Steinhurst, J., England, N., and Nyland, L., "The WarpEngine: An Architecture for the Post-Polygonal Age", *SIGGRAPH '2000*, pp. 433-442, 2000.

[15] Seitz, S.M., and Dyer, C.R., "Physically-Valid View Synthesis by Image Interpolation", *Proceedings of the IEEE Workshop on Representations of Visual Scenes*, pp. 18-25, 1995.

[16] Seitz, S.M., and Dyer, C.R., "View Morphing", *SIGGRAPH '96*, pp. 21-30, 1996.

[17] Shade, J., Gortler, S., He, L., and Szeliski, R., "Layered Depth Images", *SIGGRAPH '98*, pp. 231-242, 1998.

[18] Wallach, D.S., Kunapalli S., and Cohen M.F., "Accelerated MPEG Compression of Dynamic Polygonal Scenes", *SIGGRAPH '94*, pp. 193-196, 1994.