Enhanced Motion Blur Calculation with Optical Flow

Y .Zheng, H. Köstler, N. Thürey, U. Rüde

Insitute for System Simulation (LSS)
University Erlangen-Nuernberg
Cauerstrasse 6, 91058 Erlangen, Germany
Email: zheng@nt.e-technik.uni-erlangen.de,
{harald.koestler | nils.thuerey | ulrich.ruede}@cs.fau.de

Abstract

Computing motion blur is a complex task for modern raytracers, usually temporal supersampling is used to compute motion blur. However, this leads to increased rendering time and high computational complexity.

In the following we present an image based technique which is based on the optical flow method to achieve an approximated motion field in an image sequence. In contrast to vector fields generated by the raytracer itself, this enables us to also attain motion vectors for indirect movements, such as shadows or caustics. For the optical flow computation we use an extended Horn and Schunk model based on the assumption that changes in the illumination are ignored. Experiments to compare the results of motion blurred pictures, computed by a raytracer and the optical flow method, will be presented and evaluated with respect to visual effects and runtime.

1 Introduction

Motion blur is an important perceptual cue for computer generated images and animations. It can be synthetically generated in animated sequences, e.g. for stop motion animations, as shown e.g. in [1],[2]. Several commercial raytracing packages nowadays introduce image based motion blur techniques to overcome the high demands of accurately computed motion blur. This accurate motion blur would require correct supersampling of all moving objects, and thus usually significantly increases both the implementation complexity of the raytracer, and the required time to compute the image. A simpler method is to compute several full images at different points in time, and average these to model the exposure time of a traditional film camera. While this is easy to implement, it increases the necessary computations even more. Both approaches are also problematic if there are a few very fast moving objects in an otherwise static scene.

Figure 1: Example of a motion blurred picture computed with the combined optical flow method.

It is usually relatively simple to obtain motion vectors from direct ray hits in a raytracer. For lighting computations, a raytracer needs to compute the intersection point with an object. Hence, instead of only calculating the reflected color, motion information from this point on the object surface can be also computed, e.g. from the current rotation and translation of the object. As an approximation of the exposure time, the color of the object is drawn as a blurred line instead of a single pixel in the image. This also has the advantage, that the motion blur can be computed as a post-processing effect after the actual image computation by the raytracer.

On the other hand, this technique is unable to capture effect of indirect motion, such as the moving shadow of an object on the floor. While the raytracer can detect the movement of the object, detecting a moving shadow is significantly harder, especially if e.g. the light sources are moving as well.

A completely different approach to acquire the motion vector field for an image are optical flow algorithms. These compute the motion for each individual pixel from two frames in an animation by assuming that it does not change its intensity and moves similar to its neighbors. It can easily handle movements of objects or shadows, but has problems e.g. when intensity values change significantly from one frame to the other. Moreover, the optical flow produces smoothed out velocity fields, thus object boundaries are not accurately captured. Figure 1 is an example of a motion blurred picture computed with the combined optical flow method, another example of motion blur computed purely with optical flow can be found in Figure 9.

The approach that will be presented in the following thus not only computes the optical flow but enhances these computations by motion data from a raytracer where this information is reliantly available. In the following, the optical flow method, its combination with the raytracer information and the motion blur computation will be described in more detail.

2 Optical flow model

Optical flow is the apparent motion due to variations in the pattern of brightness. It provides us with a dense approximate motion field $\mathbf{u} = (u, v) : \Omega \rightarrow R^2$ out of two or more images in a video sequence $I(\mathbf{x}, t) : \Omega \times T \rightarrow R$, where $I(\mathbf{x}, t)$ describes the grey value intensity for a point $\mathbf{x} = (x, y)$ in the image domain $\Omega \subset R^2$ at time $t \in T = [0..t_{max}]$.

The first optical flow model was introduced by Horn and Schunk [3], has been studied intensively and was extended in several ways (e.g. [4, 5, 6, 7, 8, 9]. In the following we restrict ourselves to this model for simplicity.

The basic assumption for the optical flow model by Horn and Schunk is that a moving object in the image does not change its intensity values. This means that we neglect changes in the illumination. Mathematically this assumption can be written as $I(\mathbf{x}, t) = I(\mathbf{x} + d\mathbf{x}, t + dt)$. Using Taylor expansion we can linearize this equation and get the so called optical flow constraint (OFC)

$$I_x u + I_y v + I_t = \nabla I \cdot \mathbf{u} + I_t \ , \qquad (1)$$

with the image derivatives I_x, I_y, I_t and $\mathbf{u} = (u, v) = (dx/dt, dy/dt)$. The OFC cannot be fulfilled exactly for most real world examples. In addition to that we only have one equation for computing two unknowns at the moment, so we need another constraint. We use here the assumption that the variation of the velocity field is smooth, hence allowing the computation of the unique motion field via regularization theory ([10]). Altogether in a variational formulation these two assumptions lead to the following minimization problem

$$\min_{\mathbf{u}} E(\mathbf{u}) = \int_{\Omega} (\nabla I \cdot \mathbf{u} + I_t)^2 + \alpha |\nabla \mathbf{u}|^2 \, d\Omega \quad (2)$$

in the image domain Ω. The parameter $\alpha \in \mathbb{R}_+$ controls the influence of the regularizer $|\nabla \mathbf{u}|^2$. If α is constant we call this type of regularization homogeneous diffusion. As an extension of the simple Horn and Schunk model we either allow α to depend on the image derivatives and we call this kind of regularization isotropic diffusion ([11], [12]) or introduce a different data term to be more robust against varying illumination ([13]). This extends the simple model by adding a second data term in Equation (2) which assumes constancy of the spatial image gradient $\nabla I = (I_x, I_y)^T$. The energy function becomes

$$\begin{array}{r} E(\mathbf{u}) = \int_{\Omega} (\nabla I \cdot \mathbf{u} + I_t)^2 + \alpha |\nabla \mathbf{u}|^2 \\ + \alpha_2 |\nabla I(\mathbf{x} + \mathbf{u}) - \nabla I(\mathbf{x})|^2 d\Omega \end{array} \ , \quad (3)$$

with $\alpha_2 \in \mathbb{R}_+$. However, our main purpose is to blur the image according to the motion. Although the optical flow solution can be improved by the extended model, we find that there is almost no differences between the two models (see Figure 9). Using the variational calculus we find that a solution for problem (2) has to fulfill the Euler-Lagrange equations

$$\begin{array}{rcll} -\alpha \Delta u + I_x (I_x u + I_y v + I_t) & = & 0 & \\ -\alpha \Delta v + I_y (I_x u + I_y v + I_t) & = & 0 & \text{in } \Omega \\ \frac{\partial \mathbf{u}}{\partial \mathbf{n}} & = & 0 & \text{on } \partial\Omega \end{array} \quad (4)$$

with homogeneous Neumann boundary conditions.

In order to solve this system of PDEs we first discretize it by using finite differences on the rectangular regular grid Ω_h with mesh size h. For the discrete image derivatives we use higher order finite differences ([14]).

After the discretization one has to solve a system of linear equations with sparse system matrix (cf. [15]). This can be done efficiently by a multigrid method (cf. [16], [17], [15]). In our context

we use a pointwise block-Gauss Seidel smoother and standard intergrid transfer operators (cf. [18], [19]). In order to deal with the jumping coefficients I_x, I_y, I_t in (4) we apply Galerkin coarsening instead of direct coarsening to construct the coarse grid matrices.

3 Motion vector field computation

Figure 2 shows the comparison of two motion vector fields from the raytracer $\mathbf{u}_r = (u_r, v_r)$ and from optical flow $\mathbf{u}_f = (u_f, v_f)$. The image domain Ω can be partitioned into $\Omega = \Omega_r \cup \bar{\Omega}_r$, where Ω_r is the set of all points $\mathbf{x} \in \Omega$, where data is available from the raytracer and therefore $\mathbf{u}_r(\mathbf{x}) \neq 0$, and $\Omega \setminus \Omega_r = \bar{\Omega}_r$. We have evaluated two possibilities to combine these two motion vector fields to get $\mathbf{u} = (u, v)$.

Pixel wise comparison: A simple way is first to compute the optical flow vector field and then to separately check each pixel if there is a motion vector $\mathbf{u}(\mathbf{x}) \neq 0$ at $\mathbf{x} \in \Omega$ from the raytracer. If yes, we set $\mathbf{u}(\mathbf{x}) = \mathbf{u}_f(\mathbf{x})$. If there is no information from raytracer available, the optical flow data is used. In other words we use

$$\begin{array}{rcll} \mathbf{u}(\mathbf{x}) & = & \mathbf{u}_f(\mathbf{x}) & \text{if } \mathbf{x} \in \bar{\Omega}_r \\ \mathbf{u}(\mathbf{x}) & = & \mathbf{u}_r(\mathbf{x}) & \text{if } \mathbf{x} \in \Omega_r \end{array} . \quad (5)$$

Introduce landmarks: The second method to combine the two motion fields is to introduce the information from the raytracer as additional constraints (landmarks) in the process of the optical flow computation. That means we fix $\mathbf{u}(\mathbf{x}) = \mathbf{u}_r(\mathbf{x}) \, \forall \mathbf{x} \in \Omega_r$ when solving (4). One can think of this as setting these values as Dirichlet conditions in our PDE. Using a Gauss Seidel solver this can be done easily, if we use multigrid we have to take care of the proper treatment of the landmarks on coarser grids. We do this by fixing all points to 0 on coarser grids that interpolate to a landmark.

Figure 3 shows a comparison of the pixel wise vector field combination, and the landmark vector field computation. The visual differences are not large, but the landmark vector field does not introduce any discontinuities in the motion vector field on the boundary between Ω_r and $\bar{\Omega}_r$. These discontinuities can lead to visible artifacts, especially at object boundaries. Hence, we will use the landmark algorithm in the following.

4 Calculation of motion blur

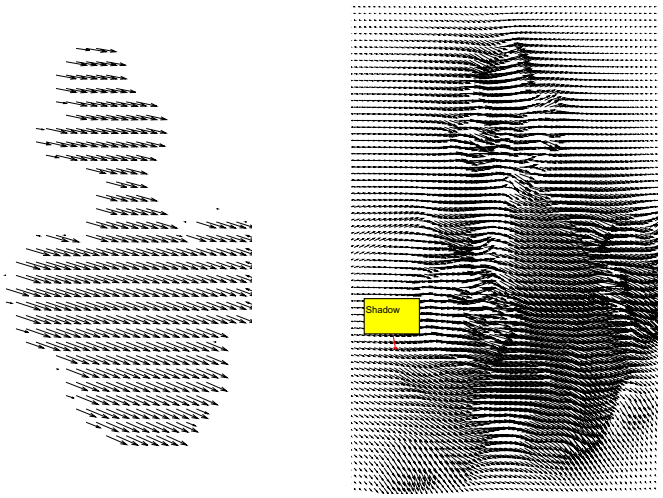
After the step of combination we have the complete motion vector field. Now the calculation of the image based motion blur will be applied. We approximate the convolution of the pixel intensity with a discrete number of steps along a line. Other methods would be to compute the duration each pixel remains within the area of a pixel, as done in [1][20], or the movement itself could be approximated by high order curve. However, we have found that the method explained below yields satisfactory results.

Let the motion vector of source pixel $P_s(i, j)$ with intensity $I(i, j)$ from one to the next frame be the movement of this pixel P from the position (i_0, j_0) in the first frame to the position (i_n, j_n) in the second frame after a exposure time interval δT. Dividing the time interval δT into N equally long subintervals $t_1 ... t_n$, we assume that the speed is constant. Now a fraction $I(i, j)/N$ of the intensity is added for all pixels in the image that are crossed by the movement through the image plane. Note that N can be chosen arbitrary according to the motion speed, but basic consideration is that N must larger than the longest motion vector. In the example of calculation shown by Figure 5 we have used $N = 15|\mathbf{u}|$.

Thus, the weight computations are similar to those for the motion blur. It is furthermore necessary to normalize the blurred intensities and retrieve the original colors of the source image after blurring. In order to normalize the intensity, we furthermore compute a weight for each image pixel $w(i, j)$, upon which the step fraction $1/N$ is added. Thus after the computation of all final pixel values $P_f(i, j)$, its value is normalized by $1/w(i, j)$, if $w(i, j)$ is larger than zero. If it is zero, this means that no moving pixel has crossed this destination pixel, and we set $P_f(i, j) = P_s(i, j)$.

5 Results

The following section will discuss results for three image sequences visually and by runtime. In the following, the open-source raytracer Blender is used to produce accurate motion blurred images as a reference. Blender computes this by raytracing 8 images for each single frame of animation. In the following, we will use the vector field computed by the Blender raytracer as $\mathbf{u}_r$.



(a) Motion vector field from raytracer (b) Motion vector field from optical flow

Figure 2: Comparison of two motion vector fields. The original image is taken from Figure 7 and shows a single moving object with a shadow on a floor plane.

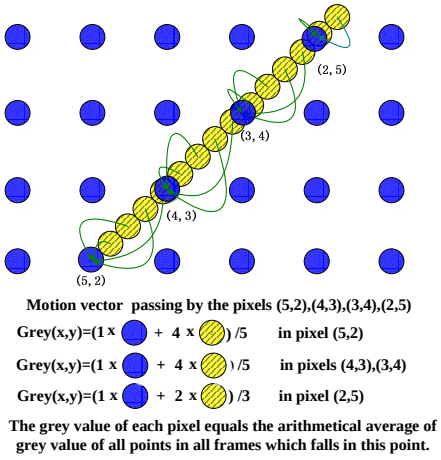


Figure 5: Calculation of the blurred pixel intensities.

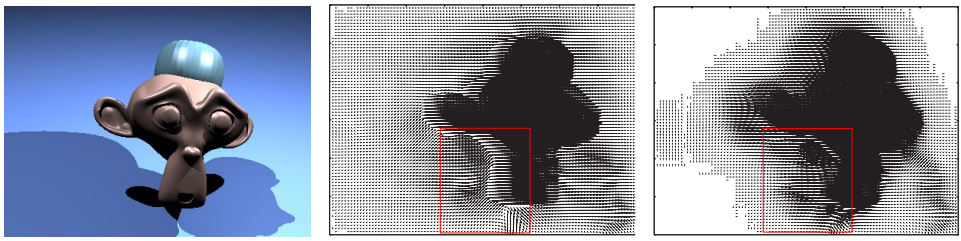
5.1 Visual effect comparison

To evaluate the visual appearance of the motion blur calculation, we use the three image sequences shown in Figure 7, 8 and 9. The second one uses a simple vase model, with a moving camera. Thus motion data is available throughout the image, and no data from the optical flow is used. As can be seen in Figure 8, the algorithm described in Section 4 yields results comparable to the reference motion blur.

The image sequences in Figure 7 and 9 both have a fast moving object, that casts a shadow onto the floor plane below. Here it can be seen that the combined motion blur computation captures both the blur of the moving object, as well as that of the moving shadow on the floor surface. In Figure 7 it can be seen that the static stars in the image can also be affected by the blur from optical flow motion field. This, however, does usually not lead to notable artifacts. Figure 9(c) is the blurred image using the extended model (3) which is more robust against varying illumination.

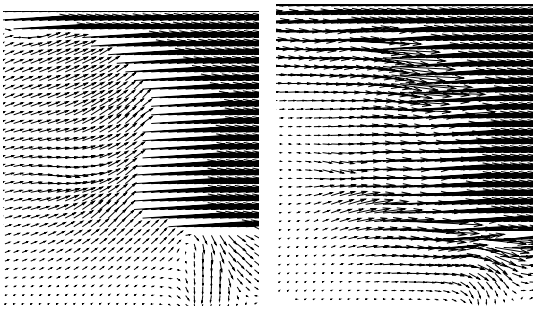
5.2 Runtime comparison

The runtime for the optical flow algorithm only depends on the image size, while for the raytracer the runtime strongly depends on the scene complexity. For a sophisticated animation, the rendering for a single frame can need huge amounts of time. Although all of our test sequences are relatively simple, the accurate motion blur of the sequence from Figure 7 still requires more than one minute to com-



(a) A single frame of the test sequence (b) Motion vector field for pixelwise comparison (c) Motion vector field using landmarks

Figure 3: Comparison of two motion vector fields.



(a) Pixelwise comparison (b) Using landmarks

Figure 4: Zoom of the motion vector field comparison from Figure 3 (marked red rectangles).

pute. The test runs were performed with a Pentium 4 2.4GHz CPU with 512KB of cache.

The overall time to compute the motion blur with the combined optical flow approach for Figure 7 with a resolution of 400×300 is 16,8 seconds, 12,5 seconds of which are spent for producing the source image and the motion vector field from the raytracer. 2,4 seconds are spent to compute the motion vector field from optical flow. Figure 6 shows the runtime comparison for all test cases. Our current implementation requires ca. 1,9 seconds to compute the motion blur itself. Thus, the total time to compute the motion blurred picture with our combined method is 4,3 seconds, independent of the scene complexity.

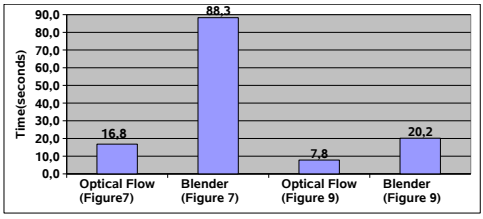


Figure 6: Runtime comparison for blurring single frame.

6 Conclusion and Outlook

We have successfully combined the two vector fields from raytracer and optical flow to compute the motion blur in an image sequence. Its runtime only depends on the image size, and is thus significantly lower than approximating the motion blur with the raytracer. Our combined optical flow approach on the other hand requires only slightly more time than a pure image based motion blur calculation, but enhances the visual appearance.

Future work will be to optimize the current implementation, and use more advanced optical flow

models. For fast movements *warping* [21] could be used to enhance the accuracy of the optical flow calculation. The results could moreover be enhanced by using render passes, a feature most current raytracers support, to give finer control of the motion blur for certain parts of an image.

References

[1] Brostow, G.J., Essa, I.: Image-based motion blur for stop motion animation. International Conference on Computer Graphics and Interactive Techniques,Proceedings of the 28th annual conference on Computer graphics and interactive techniques (2001)

[2] Potmesil, M., Chakravarty, I.: Modelling motion blur in computer-generated images. Computer Graphics, 17(3) (1983) 389–399

[3] Horn, B., Schunck, B.: Determining optical flow. Artificial Intelligence (1981) 185–203

[4] Battiti, R., Amaldi, E., Koch, C.: Computing optical flow across multiple scales: an adaptive coarse-to-fine strategy, 6(2). International Journal of Computer Vision (1991) 133–145

[5] Terzopoulos, D.: Image analysis using multigrid methods. IEEE Transactions on Pattern Analysis and Machine Intelligence (1986) 129–139

[6] Nagel, H.H., Enkelmann, W.: An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. IEEE Transactions on Pattern Analysis and Machine Intelligence , vol 8 (1986) 565–593

[7] Nagel, H.H., Enkelmann, W.: Variational approach to optical flow estimation managing discontinuities. Image and Vision Computing, 11(7) (1993) 419–439

[8] Schnörr, C.: Segmentation of visual motion by minimizing convex non-quadratic functionals. In Proc. Twelfth International Conference on Pattern Recognition, volume A (1993) 661– 663

[9] Weickert, J., Schnörr, C.: A theoretical framework for convex regularizers in PDE-based computation of image motion. The International Journal of Computer Vision,45(3) (2001) 245–264

[10] Poggio, T., Torre, V., Koch, C.: Computational vision and regularization theory. Nature, vol.37 (1985) 314–319

[11] Bruhn, A., Weickert, J., Kohlberger, T., Schnörr, C.: Variational optical flow computation in real time. IEEE Transactions on Image Processing, 6(5) (2005.) 1–6

[12] Alvarez, L., Esclarín, J., Lefebure, M., Sánchez, J.: A PDE model for computing

(a) Original image

(b) Reference motion blur computed by raytracer

(c) Motion blur computed with combined optical flow and raytracer data

Figure 7: An animation with a more complex object and a moving shadow.

the optical flow. Proceedings of CEDYA XVI (September 1999) 1349–1356

[13] Bruhn, A., Weickert, J.: Towards ultimate motion estimation: Combining highest accuracy with real-time performanc. ICCV, **1** (2005.) 749 – 755

[14] Köstler, H., Kalmoun, E., Rüde, U.: Parallel multigrid computation of the 3d optical flow. Technical Report 04-4, University of Erlangen-Nuremberg (2004)

[15] Kalmoun, E., Rüde, U.: A variational multigrid for computing the optical flow. Vision, Modeling and Visualization 2003 (Berlin 2003) 577–584 T. Ertl, B. Girod, G. Greiner, H. Niemann, H.-P. Seidel, E. Steinbach, R. Westermann (Eds.). Akademische Verlagsgesellschaft.

[16] Terzopoulos, D.: Image analysis using multigrid relaxation methods. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 8 (1986) 129–139

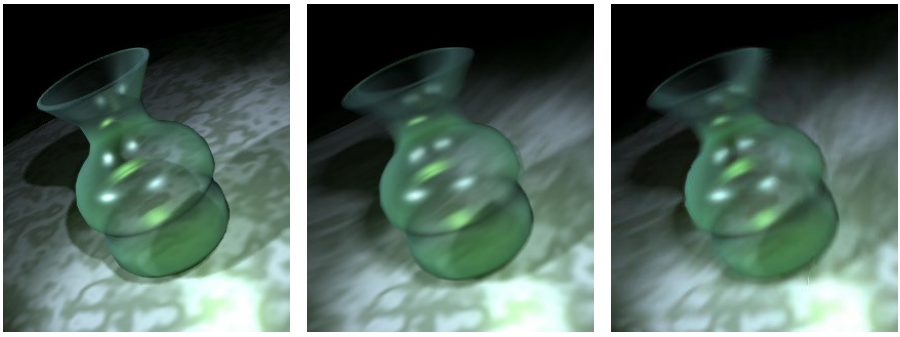
[17] Enkelmann, W.: Investigations of multigrid algorithms for the estimation of optical flow fieldsin image sequences. Computer Vision, Graphics, and Image Processing , vol 43 (1988) 150–177

[18] Trottenberg, U., Oosterlee, C., Schüller, A.: Multigrid. Academic Press (2001)

[19] Briggs, W., Henson, V., McCormick, S.: A Multigrid Tutorial. 2nd edn. Society for Industrial and Applied Mathematics (2000)

[20] Max, N., Lerner, D.: A two-and-a-half-d motion-blur algorithm. SIGGRAPH, 19(3) (1985) 85–93

[21] Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. (Computer Vision - ECCV 2004, Lecture Notes in Computer Science, Vol. 3024, Springer, Berlin, 25-36)

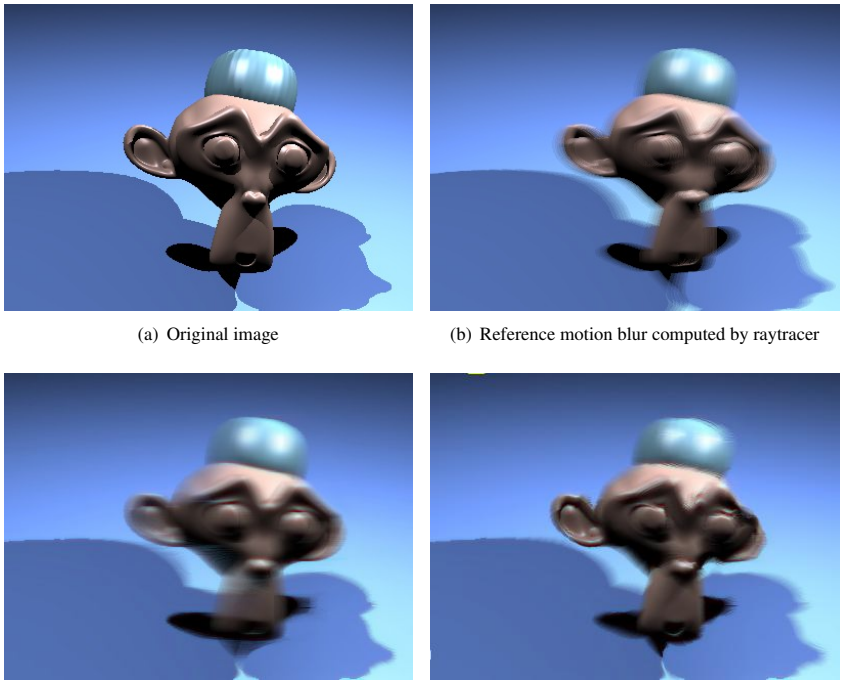


(a) Original image

(b) Reference motion blur computed by raytracer

(c) Motion blur computed with the raytracer vector field and the algorithm from Section 4

Figure 8: Motion blur for an animation with a moving camera.



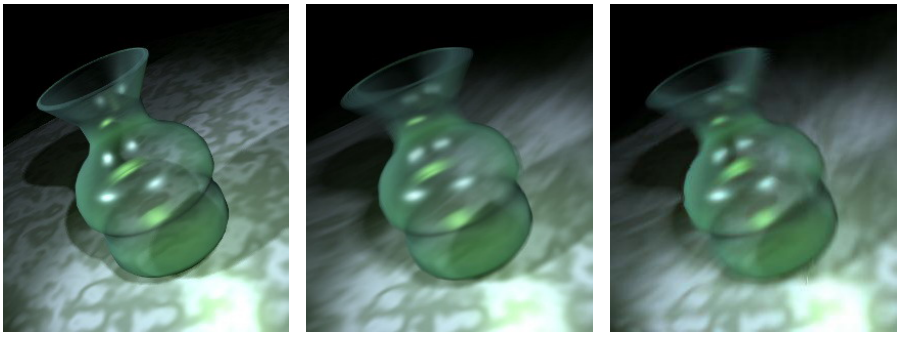
(a) Original image

(b) Reference motion blur computed by raytracer

(c) Motion blur computed with considering the varying illumination

(d) Negative example - wrong motion blur computed with pure optical flow data

Figure 9: Comparison of the motion blur for a single moving object with two shadows. The last picture shows an example of a wrong motion blur computation with only the optical flow vector field.

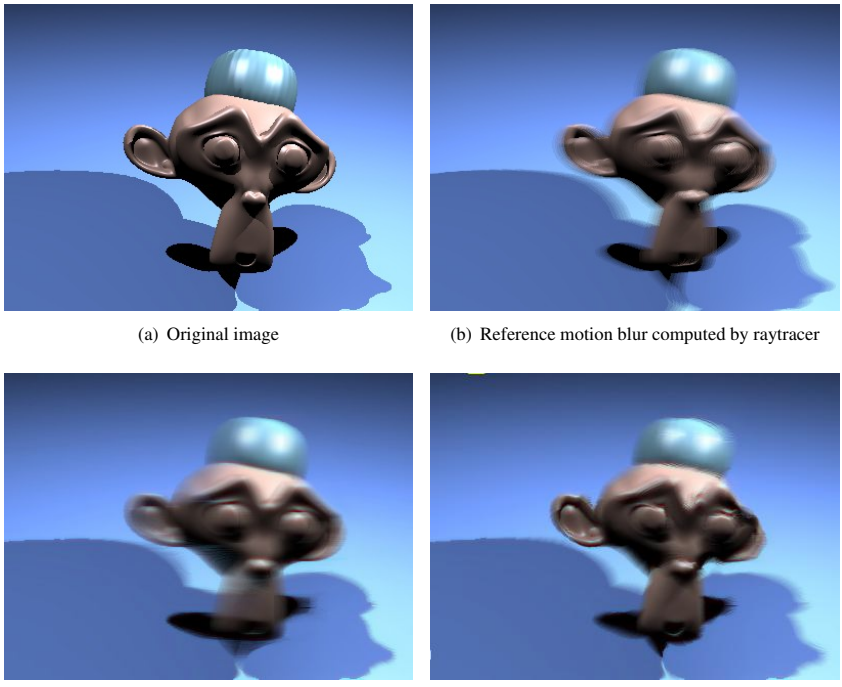


(a) Original image

(b) Reference motion blur computed by raytracer

(c) Motion blur computed with the raytracer vector field and the algorithm from Section 4

Figure 1: Motion blur for an animation with a moving camera.



(a) Original image

(b) Reference motion blur computed by raytracer

(c) Motion blur computed with considering the varying illumination

(d) Negative example - wrong motion blur computed with pure optical flow data

Figure 2: Comparison of the motion blur for a single moving object with two shadows. The last picture shows an example of a wrong motion blur computation with only the optical flow vector field.