vSLRcam – Taking Pictures in Virtual Environments

Angela Brennecke
University of Magdeburg, Germany
abrennec@isg.cs.uni-magdeburg.de

Christian Panzer
University of Magdeburg, Germany
christianpanzer@googlemail.com

Stefan Schlechtweg
Anhalt University of Applied Science
Köthen, Germany
stefan.schlechtweg@inf.hs-anhalt.de

ABSTRACT

Our work presents a virtual single lens reflection camera (vSLRcam) application which is employed in a virtual training environment for crime scene investigation. vSLRcam's back-end is a GPU based simulation of a realistic camera model taking into account SLR camera properties like apperture, shutter speed, lens, etc., as well as their interdependencies. Thus, we can obtain realistic lens effects like motion blur or depth of field in real-time. The application user interface allows for parameterizing the inidividual camera attributes to achieve those effects and, as a result, to take realistic pictures of the scene. The resulting images come very close to real world photographs with equal parameter values. Our main contributions are a common framework for the SLR camera attributes and the simulation of their interdependecies in a single application which is capable of rendering photographic lens effects in real-time.

Keywords: realistic camera model, virtual environments, real-time rendering, motion blur, depth of field

1 INTRODUCTION

The simulation of lens effects produced by a realistic camera model is a recurrent field of interest in computer graphics research. While incipient works usually applied raytracing techniques to achieve effects like motion blur or depth of field, by now GPU-based shader programming allows for real-time rendering of the effects even within virtual environments (VEs) and 3D games. This not only strengthens the environment's immersion depth and realism but also makes real-time cinematography applicable to it [7].

Usually, in games only certain lens effects are simulated and they become an inherent part of the game environment. A manual parameterization of the effects is therefore not intended and incorporated. In contrast, we wanted to completely approximate a single-lens-reflection camera and integrate it into our virtual training environment. Moreover, we wanted a realistic camera model to be the basis for the virtual SLR camera. We put the main emphasis on the simulation of individual camera components and their contributions to the final image on the one hand and their interdependencies on the other hand in order to realistically generate photographic lens effects. Even though we did not focus on film negative types in the first place, we also integrated granularity effects and added a parameterization for the film speed.

In order to allow for real-time rendering, the virtual SLR camera was implemented using OpenGL's Shading Language on modern graphics hardware. The application further was realized for usage in scene graph based VEs and was integrated into the virtual training environment OpenCrimeScene for testing [4]. OpenCrimeScene is designed as a serious game for crime scene investigation. It will be used by police students for training purposes, e. g. crime scene photography. As the students have to understand the interdependencies of camera components in order to take useful pictures of the crime scene, the vSLRcam has to meet realistic standards.

2 BACKGROUND

Photography is the process of projecting 3D objects onto a 2D image plane through a center of projection including geometric distortions in the final image. The image plane has to be made out of a light sensitive material in order to capture the picture constantly. This is usually a photo sensitive negative film for analog or a CCD sensor for digital cameras. A picture then is a reproduction of light intensities reflected from the object surfaces which lie in the angle of view.

We cannot go into detail on the underlying principles here and, thus, we clearify only the relationships between the terms *optical principles*, *realistic camera model*, and *SLR camera*. We assume that you are familiar with the first two points and focus on camera components hereafter.

1. Optical principles of reflection and refraction explain how light intensities from one place can be reproduced at another.
2. A realistic camera model is a geometrical explanation for the process of projecting 3D objects onto a 2D image plane.
3. An SLR camera approximates a realistic camera model.

2.1 Camera Components

Basically, an SLR camera is a photographic tool which consists of an opaque body containing a photo sensitive image plane and a camera lens. The amount of light that reaches the image plane during the exposure can be controlled by the aperture and the shutter which are components of the camera lens and camera respectively. The duration of the exposure is crucial for a balanced image illumination. Moreover, different visual effects within the image can be achieved by coordinating the aperture and shutter.

Camera lens

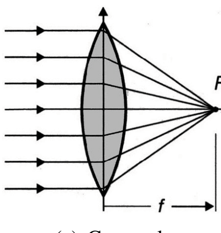
Lenses are made of translucent material and, hence, have refractive power. There are convex and concave lenses, with the former being responsible for converging and the latter being responsible for diverging incoming light rays (cf. Fig. 1). The decisive lens parameter is the *focal length f* which is the distance between the center of the lens and the *focal point F*. That is, all rays travelling in parallel to the optical axis get refracted through the lens and intersect in *F*. In case of concave lenses, the focal point lies on the rear side of the lens' refraction border [3, 8].

Thus, the refractive power of lenses offers to steer the incoming light rays into a particular direction. When grouping convex and concave lenses together, even more control can be gained over the light distribution. Consequently, the camera lens usually consists of a whole group of convex and concave lenses. The decisive property of it certainly is the focal length, as it determines the camera lens's angle of view as well as its enlargement factor. A small focal length allows for a wide pane whereas a small pane is caused by a large focal length.

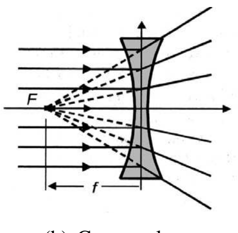
There are four standardized camera lens types which range from small focal length to large focal length. These are wide angle, normal, and tele as well as zoom lens for varying focal lengths.

Aperture

The aperture regulates the amount of incoming light by increasing or decreasing its size. It is specified in so-called *f-numbers* and is set in fixed steps, so-called *f-stops*, as can be seen in Figure 2. An f-number is calculated as the ratio of the focal length and the opening's diameter. Given a focal length of 28 mm and a current opening diameter of 10 mm, the f-number then is $2,8\,f = \frac{28\,mm}{10\,mm}$. In order to regulate the amount of incident light falling onto the image plain, each camera furthermore is endued with a so-called *shutter*.



(a) Convex lens



(b) Concave lens

Figure 1: Convex and concave lenses which converge and diverge light rays to the (virtual) focal point F [11].

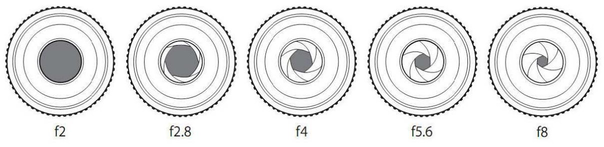


Figure 2: Different aperture f-stops [1].

Shutter and Shutter Speed

If the aperture takes care of *how much* light enters the camera lens, the shutter takes care of *how long* this amount of light reaches the image plane. It is part of the camera's body and is positioned between the camera lens and the image plane. Generally, the shutter is made out of small leafs which are opened for a certain amount of time, the *exposure time* (also referred to as *shutter speed*). It is specified in seconds *s*, whereas each time step is doubling or halving the previous one, e. g. like $\cdots, \frac{1}{15}\,s, \frac{1}{30}\,s, \frac{1}{60}\,s, \frac{1}{125}\,s, \frac{1}{250}\,s$, etc.

Negative Film or CCD Sensor — The Image Plain

The image plane of a camera either is equipped with a negative film for analog models or a CCD sensor for digital cameras. Both are photo sensitive and are thus able to capture the incoming light. A CCD sensor simply converts the incoming light into an electrical signal and stores it to memory. A negative film, in contrast, brings its own visual effect. This can be simulated by a digital camera, however it is unique for each film type.

Negative films are coated with a light-sensitive emulsion of silver halide salts that contains crystals in variable size. The exposure of this emulsion results in a permanent image capture, the negative, which has to be chemically processed to become the final image, the positive. The emulsion is responsible for the film's light sensitivity, the so-called *film speed*. The smaller the crystal size, the less light-sensitive (slow) the film is but the finer the final image details become. In turn, film types which are very light sensitive (fast) can cope well with dark surrounding light conditions but often result in granulous images (cf. Fig. 3). The film speed is specified in ISO[1] values which typically range between 100 to 1600.

[1] ISO stands for *International Standardization Organization*. Digital cameras usually approximate the film speed.

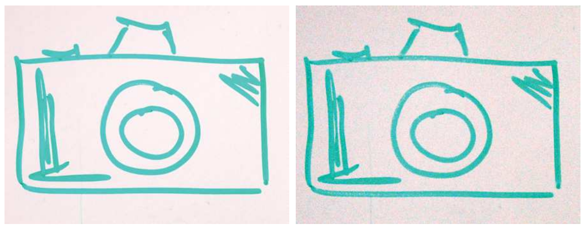


Figure 3: The film speed determines the image granularity. On the left an ISO 200 film speed leads to fine results whereas on the right an ISO 1600 film speed results in granulous images.

2.2 Interdependencies

Given the camera components several photographic effects can be derived. This is due to the interdependencies of the single components and their parameter settings.

Exposure

The procedure of light falling onto and reacting with the photo sensitive image plane is called *light exposure*. It is specified as exposure value *EV* and depends on the illumination level (regulated by the aperture and shutter) and the image plane's sensitivity. The illumination level now is regulated by a balanced combination of aperture size and shutter speed. For example, the same illumination level can be gained by a widely opened aperture and a short shutter speed or a small opening and a longer shutter speed. Both ways, the same amount of light enters the image plane.

As mentioned in Section 2.1, the film speed specifies the film's light sensitivity. Fast films can be useful under dark lighting conditions because still a small amount of light suffices to obtain a correct illumination but might lead to granulous image effects. Slow films, on the opposite, need more incoming light. Hence, either the aperture has to be widely opened or the shutter speed has to be slow. This, however, could cause blurring effects.

Motion Blur

Motion blur describes a blurring effect of the whole or parts of the image. The effect is caused either by the camera's or the motif's motion and a slow shutter speed. Figure 4 illustrates the effect. On the left hand side you see a swinging person whose motion is frozen in the image due to a short shutter speed ($\frac{1}{30}s$) whereas the picture on the right hand side strongly shows motion blurring caused by a slower shutter speed ($\frac{1}{4}s$). Besides, the pictures also demonstrate the relation of aperture and shutter speed for a correct illumination. In order to achieve equal illumination levels, the left picture had a wide opened aperture of value 8 whereas the right one was taken with a small opening of value 22.

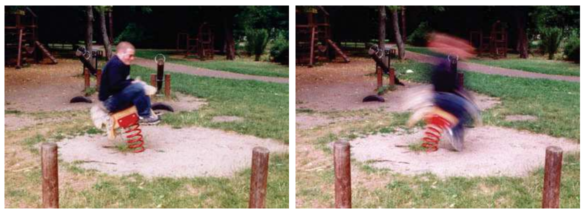


Figure 4: The exposure time can be used to produce motion blur effects. In the right picture a longer exposure time causes a blurred image of the swinging person. (Pictures courtesy of Konrad Mühler.)

Depth of Field

A second effect that is caused by aperture size and shutter speed setting is the depth of field. When taking a picture certain objects are focussed and consequently get displayed sharply on the photograph. The depth of field describes the area around the object in focus which is also projected without blurring onto the image plane. Geometrically, this can be explained by the *focus plane* to which the image plane has to be related (cf. Fig. 5). All objects placed on the focus plane will be projected alike on the image plane. Objects which are placed far beyond the focus plane, however, will be blurred, they form so-called *circles of confusion* on the image. The size of the circle of confusion determines whether the object's points lie within or without the depth of field and grows with increasing distance to the focus plane.

The aperture is the main indicator for regulating the depth of field. The thinner the incoming light cone, the smaller will the circle of confusion be (cf. Fig. 5). Furthermore, a small focal length as well as the object's distance to the camera increase the depth of field (cf. Fig. 6).

3 RELATED WORK

There have been a few approaches to simulating camera lens effects in computer graphics. Recent works deal with image correction techniques due to weak results of digital photography, e. g. [2]. Generally, the approaches concentrate on single camera aspects or lens effects without taking the ensemble into account. This is what we wanted to address. Beside, the existing approaches either are based on raytracing techniques and thus cannot achieve interactive frame rates or they do

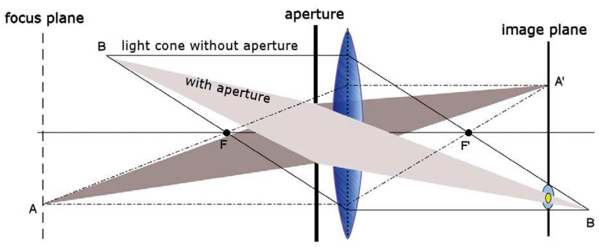


Figure 5: The illustration shows the influence of aperture size on depth of field. All points (A) on the focues plane project points onto the image plane (A'). In contrast, distant points (B) project circles of confusion on the image plane (B'). Reducing the aperture size also leads to smaller circles of confusion which then become part of the depth of field.

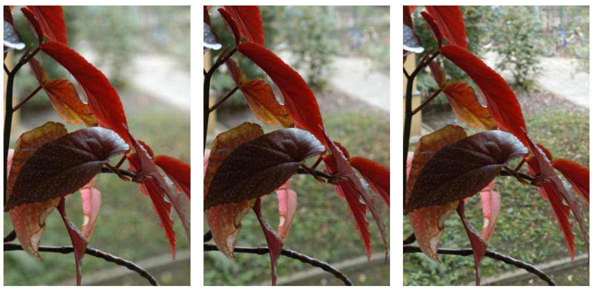


Figure 6: The depth of field spreads out differently due to the aperture settings. By minimizing the f-number, the depth of field area increases (from left to right f4.5, f8, and f20).

allow for real-time rendering but not for integration into interactive virtual environments, e. g., they render single objects only. Both is mandatory for us, though.

The pioneers in the area of rendering depth of field effects were Potmesil and Chakravarty who developed a post-processing technique to render depth of field based on raytracing [13]. The approach generates depth of field in a (pre-)rendered image from a standard pinhole camera by blurring each pixel with a pre-computed circle of confusion. Even though the technique is far to slow for real-time graphics it has inspired several further works, e. g. [15, 9, 17], which apply hardware shader programming. Other approaches were based on distributed raytraycing or made use of an accumulation buffer to simulate depth of field, e. g. [5, 6]. The latest depth of field simulation by [10] is based on GPU programming and leads to beautiful results. However, each of these approaches is not applicable at interactive frame rates. For simulating depth of field as part of our virtual SLR-camera we present a technique which is also based on Potmesil's works [13]. Yet, we make use of the GPU to achieve real-time rendering.

The simulation of a motion blur effect is a desirable feature especially in computer games as it increases the game's realism. The first attempt also was undertaken by Potmesil et al. [14]. They generated the motion blur effect by applying a time convolution filter to the originally rendered image together with a moving Fourier transformation function. However, this technique is not capable of real-time rendering. Moreover, it is far from being physically correct since it only uses a single input image. A similar approach has been made by Shimizu et al. using hardware shader programming [16]. The authors integrated a pre-computed vector field to determine the optical flow of the individual 3D objects first. Then, by warping and filtering the input image several times according to the optical flow, motion blur is generated. The technique works in real-time but is only suitable for single objects as well as for pre-computed vector fields. This is too restricted for a virtual training environment.

Our approach for simulating motion blur, however, has to depend on the camera settings in the first place

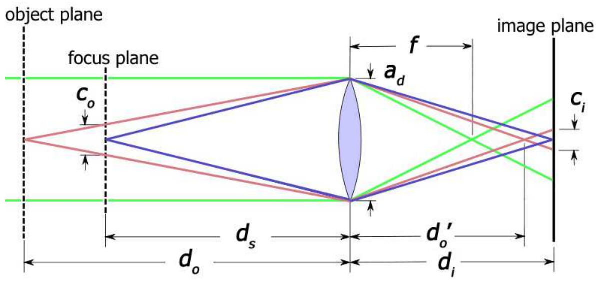


Figure 7: The illustration shows the optical paths within an SLR camera. Parameters included are the focal length f, the aperture's diameter a_d as well as the distances to the focus plane d_s, image plane d_i, and object plane d_o as well as its' image d_o'. The latter are necessary for generating the circles of confusion diameters c_i derived from non-focussed object points c_o.

and thus is based on the works of Haeberli and Akeley [6]. They introduced the accumulation buffer which allows for accumulating several images into one output image. In contrast, we do not use an accumulation buffer but rather implement the image accumulation using hardware shader programming.

4 A VIRTUAL SLR CAMERA

As shown in the previous sections, the SLR camera components are related to one another and can produce certain photographic effects. As a virtual counterpart, the vSLRcam has to offer the same functionality. Each component and according parameters as well as the component's internal relationships need to be identified first.

Parameters

The decisive parameters to generate a certain visual effect are given by the camera components lens, aperture, shutter, and film type. The according parameters which can be specified by the user are:

- focal length f and distance to the focus plane d_s
- f-number a
- exposure time t
- ISO value i_s and film format i_f

To realistically render photographic effects like, e. g. depth of field, further parameters have to be derived. Figure 7 illustrates the main geometrical parameters that are necessary. Our approach is based on a thin lens approximation which simplifies the individual parameter calculations [3].

4.1 Lens Effects

The lens effects we would like to realize are an adjustable angle of view, depth of field, and motion blur. Furthermore, we have to determine the correct image illumination regarding the current lighting conditions as we also want to allow for over- and under-exposure. The above specified parameters will be used to approximate these effects.

Angle of View

The angle of view is the result of a specific camera lens type, see 2.1. It, thus, is responsible for the viewing pane of the resulting image. However, it is not the focal length f which determines the angle of view but rather the distance d_i from the lens center to the image plane on the one hand and the image plane's diagonal i_d (which is specified by the film format i_f) on the other hand. As a result, the angle of view also changes during focussing as the distance to the image plane d_i is changing. The angle of view is given by:

$$\alpha = 2 \cdot \arctan\left(\frac{i_d}{2 \cdot d_i}\right) \tag{1}$$

The parameters that have to be specified by the user are focal length f, film format i_f, and focus distance d_s. Given these parameters, i_d as well as d_i have to be derived. To calculate the former, $\sqrt{width^2 + height^2}$ has to be employed whereas the latter can be obtained by applying the thin lens formula:

$$\frac{1}{f} = \frac{1}{d_i} + \frac{1}{d_s} \quad \Longrightarrow \quad d_i = \frac{f \cdot d_s}{d_s - f} \tag{2}$$

Finally the angle of view is given by

$$\alpha = 2 \cdot \arctan\left(\frac{i_d \cdot (d_s - f)}{2 \cdot f \cdot d_s}\right) \tag{3}$$

Realization The adjustment of the angle of view is simply implemented by changing OpenGL's view frustum accordingly. Figure 8 shows different lens type simulations.

Depth of Field

Rendering the depth of field effect is a bit more complex. For each image point a circle of confusion has to be computed. To start with, we calculate the circle of confusion c_o in object space which projects on the focus plane at distance d_s. This is illustrated in Figure 7. By applying the intercept theorems we receive

$$\frac{c_o}{a_d} = \frac{d_o - d_s}{d_o} \quad \Longrightarrow \quad c_o = a_d \cdot \frac{d_o - d_s}{d_o} \tag{4}$$

With a_d being calculated as the ratio of focal length f and f-number a, the equation becomes

$$c_o = \frac{f}{a} \cdot \frac{d_o - d_s}{d_o} \tag{5}$$

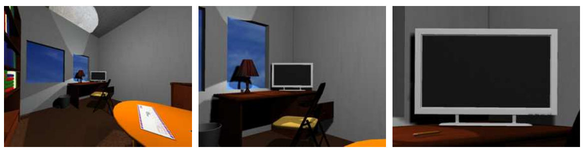

(a) Wide angle lens (b) Normal lens (c) Tele lens

Figure 8: Simulation of different camera lens types resulting in different angle of views. The camera position is the same for each picture.

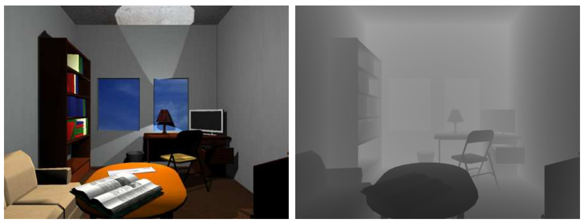

(a) T_1 (b) T_2

Figure 9: The texture on the left is a normal scene rendering whereas the texture on the right holds the depth information to calculate the depth of field.

Then, again by applying the intercept theorem for the circle of confusion c_i we get

$$\frac{c_i}{c_o} = \frac{d_i}{d_s} \quad \Longrightarrow \quad c_i = c_o \cdot \frac{d_i}{d_s} \tag{6}$$

The image plane distance d_i already has been calculated in equation 2, thus,

$$c_i = c_o \cdot \frac{f \cdot d_s}{(d_s - f) \cdot d_s} = c_o \cdot \frac{f}{d_s - f} \tag{7}$$

and finally by substituting c_o we have

$$c_i = \frac{f}{a} \cdot \frac{d_o - d_s}{d_o} \cdot \frac{f}{d_s - f} \tag{8}$$

as the formula to calculate the circle of confusion of each object point in the image.

Realization Equation 8 now has to be applied to each image point in order to receive a realistic depth of field distribution. Our approach is based on the works of [13]. However, in order to render the effect in real-time we implement it using OpenGL Shader Language.

The contributing parameters are the focal length f, the f-number a and the distance to the focus plane d_s which can be specified by the user. The distances to the object points, described by d_o, are given by the scene as z-values.

The implementation now consists of four rendering passes, each rendering the scene to an individual texture, T_1, T_2, T_3, and T_4, respectively. We do not use multiple rendering textures here, because some of the textures serve only as static input data which is also needed for other lens effects later on.

The first rendering pass is a pure OpenGL pass which simply obtains the correct color distribution. The second pass then renders the depth values to T_2 and will be used as a lookup table for the camera–object distances d_o (cf. Fig.9).

The third rendering pass receives both textures T_1 and T_2 as well as the user specified parameters as input variables. The calculation of the depth of field then is done in the fragment shader as follows: For each pixel from T_1 the circle of confusion's diameter is calculated according to Equation 8 as well as an equally sized poisson disc filter being centered on the corresponding pixel. The poisson filter's sample points are associated to the proximity pixels that lie within the radius of the circle of confusion. When the sampling is performed each pixel color as well as the surrounding pixel colors contribute to the new pixel color being stored in the texture T_3. Although this technique leads to very realistic results it does not prevent the leaking of color from sharp objects to the blurred background. We do not circumvent this drawback yet. Beside, the poisson sampling leads to other regular artefacts (cf. Fig. 10 (above)). We simply overcome these by re-filtering T_3 with another poisson disc sampling and store the result in texture T_4 (cf. Fig. 10 (below)).

Exposure

During the exposure the image plane is exposed to the incoming light. If the amount of light is too high, the final image becomes over-exposed. If it is too low it becomes under-exposed. To achieve a correct illumination two different exposure values have to be calculated: EV_s indicating the scene illumination and EV_c as a result of the current aperture/shutter speed setting. The applied equations are based on standardized 2-base logarithmic scales. A correct illumination then means that the difference $EV_{delta} = |EV_s - EV_c|$ is minimal. This can be achieved by adjusting either aperture size or shutter speed.

The scene illumination EV_s is approximated by three parameters: The scene illuminance E specified in lux lx, the film speed i_s specified in ISO, and a photometer dependent constant C by default given as 250 lx.[2] The equation then is given as:

$$EV_s = log_2\left(\frac{E \cdot i_s}{C}\right) \tag{9}$$

The exposure value caused by the current aperture/shutter speed combination EV_c is given by:

$$EV_c = log_2\left(\frac{a^2}{t}\right) \tag{10}$$

Every increase or decrease of an exposure value by one can either be caused by increasing or decreasing the f-number or the exposure time by one step each. That means, each increase or decrease of an exposure value by one leads to either twice or half of the amount of incoming light. A common SLR camera can switch between about 10 f-stops which corresponds to 10 different exposure values. Higher or lower values will be displayed black or white.

Realization To realize a virtual exposure the values of EV_s, EV_c, and EV_{delta} have to be calculated. Therefore, the parameters f-number a, exposure time t, film speed i_s, and photometer constant C need to be specified by the user or are given. A correct specification of the scene illuminance E would require a global illumination model. As the underlying OpenGL API is based on a local illumination model E has to be approximated. We specify E manually and set it to 500 lx which is realistic for indoor scenes.

Given these parameters we then apply two rendering passes. The first pass simply renders the scene to texture T_1. In the second pass we calculate EV_c and EV_s as well as the difference E_{delta}. Due to the result, the vSLRcam then can either propose a new aperture/shutter speed setting to accomplish a correct illumination or the over-/under-exposed image is rendered.

To achieve the latter, each pixel color from T_1 has to be modified in the fragment shader. The tricky part here is to correctly map the exposure values to the color space in order to realistically simulate an over- or under-exposure. As the pixel colors range between 0.0 and 1.0 the exposure value has to map to the interval -1.0 to +1.0. Otherwise, no black pixel could completely turn white and vice versa. Thus, each change of an exposure value by one leads to a color change of 0.2. That is, the exposure value range of 2.0 devided by the 10 different exposure value steps a common SLR camera allows for. Hence, the new color can be calculated by:

$$newColor = oldColor + 0.2 \cdot EV_{delta} \tag{11}$$

Consult Figure 11 for examples on virtual exposure.

Motion Blur

The motion blur effect occurs during exposure when either parts of the scene or the camera move. Each change in position of the scene or scene objects is captured on the image plane and results in blurred areas, because the exposure H is the sum of the individual illuminance values E over the exposure time t. This is given by:

$$H = \int E\ dt \tag{12}$$

Realization To realistically simulate motion blur the scene's individual illuminances E have to be summed up over time t. Again, t is given but E has to be approximated. We assume each frame being rendered during the exposure as the current scene illuminance E and sum up the frames by a weighting factor $\alpha : [0,1] \rightarrow \mathbb{R}$.

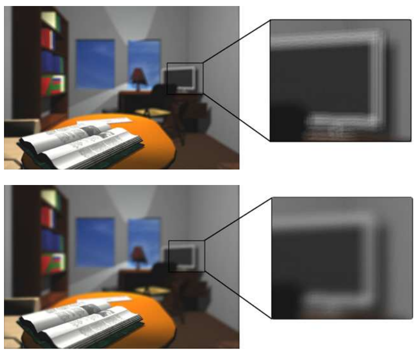
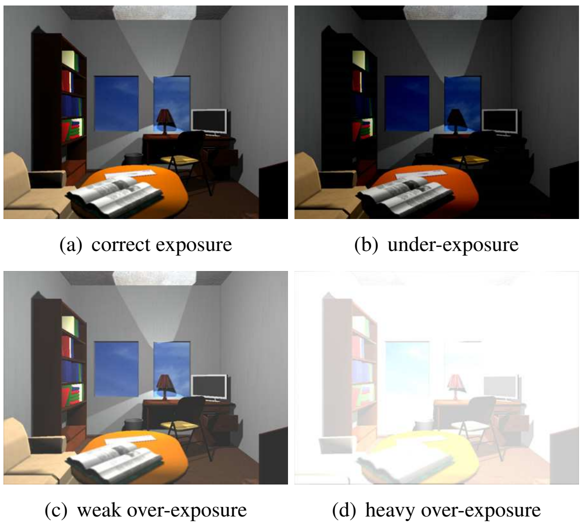


Figure 10: Both pictures contain depth of field. Due to the poisson disc filter the upper picture consists of regular artifacts. By re-sampling the image, the artifacts can be suppressed, though (lower image).

[2] C is the incident light meter calibration constant.


(a) correct exposure (b) under-exposure

(c) weak over-exposure (d) heavy over-exposure

Figure 11: The illustration shows four types of exposure. (a) shows the first rendering pass image. (b) This then becomes under-exposed with settings $a = 4{,}0f$ and $t = \frac{1}{60}$. Moreover, (c) and (d) are over-exposed by values $a = 2{,}0f$ and $\frac{1}{60}$ (c) and $a = 1{,}4f$ and $t = \frac{1}{15}$.

This factor α takes care of how strong each frame contributes to the final image together with the following equation:

$$step_n = (1-\alpha)F_n + \alpha^1(1-\alpha)F_{n-1} + \alpha^2(1-\alpha)F_{n-2} + \ldots\ldots + \alpha^n F_0 \quad (13)$$

Equation 13 assures two things: First, the contribution of each frame to the final image is decreased by the number of rendered frames n. Second, when we start the exposure we immediately receive an output image. This way we can render motion blurred images realistically and in real-time (cf. Fig. 12).

The problem here is that for long exposure times (large n) α can become so small that the corresponding frame gets transparent and therefore does not contribute to the final image anymore. This contradicts our request for physical correctness and, thus, the question is how to define α? First, we state the following two conditions:

1. The frames per second (FPS) determine the number of frames which need to be accumulated to receive a correct illumination in the motion blurred image.
2. The color intensity of an image is nearly transparent if reduced to $\frac{1}{50}$ of the original value and it becomes invisible if reduced further.

Following Equation 13, the first frame F_0 contribution is specified by α^n in each step. That means, to

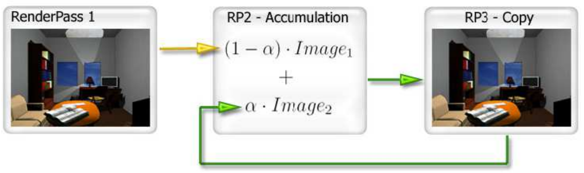
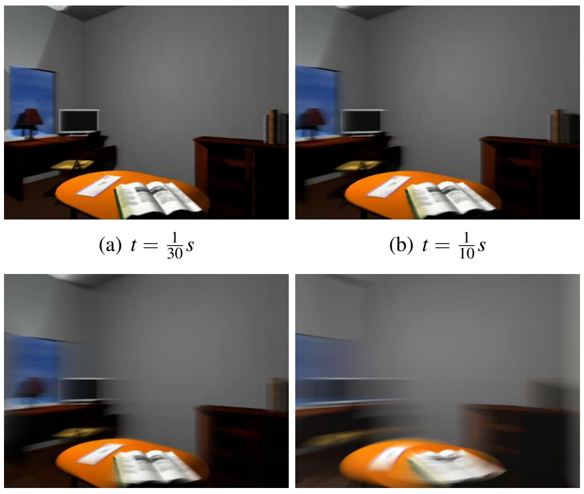


Figure 12: Three rendering passes are responsible for accumulating the current frame with the previous frames by weighting factor α.


(a) $t = \frac{1}{30}s$ (b) $t = \frac{1}{10}s$

(c) $t = \frac{1}{5}s$ (d) $t = 1s$

Figure 13: Simulation of motion blur by varying exposure times t.

assure a contribution of at least $\frac{1}{50}$ of the frame to the final image, we have to define a threshold ε which has to be at most equal to α^n: $\varepsilon \leq \alpha^n$. Hence, α can be calculated by:

$$\alpha = \sqrt[n]{\varepsilon} \quad (14)$$

Given an exposure time t of $\frac{1}{8}s$ and $80\,fps$ we need to accumulate $n = 10$ frames. This results in $\alpha = 0.676$. See Figure 13 for some example output images.

5 RESULTS

The virtual SLR camera approximates the main components of a real SLR camera and allows for realistically simulating the resulting lens effects. These even can be rendered in real-time which offers us the possibility to integrate the vSLRcam into our virtual training environment OpenCrimeScene. Even though our user interface does not conform to a realistic SLR camera display yet, the necessary settings can easily be adjusted. Figure 14 shows an example of virtual pictures that are used to document the crime scene.

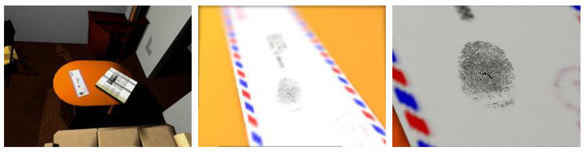
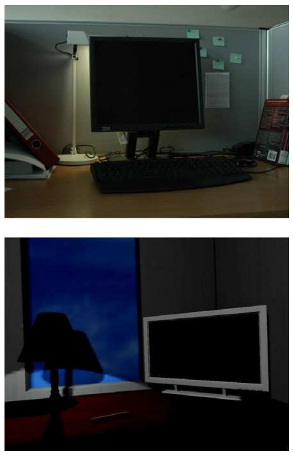


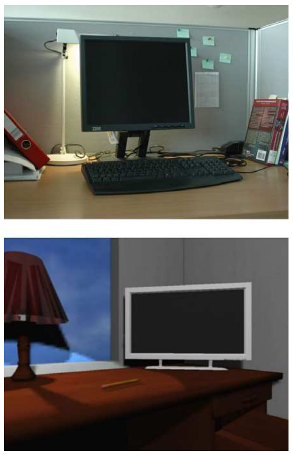
Figure 14: The vSLRcam can be used to document the crime scene. Here, the virtual photographs depict a letter with a fingerprint on it. An overview picture illustrates the context first. Then the letter is focussed more closely (from left to right).

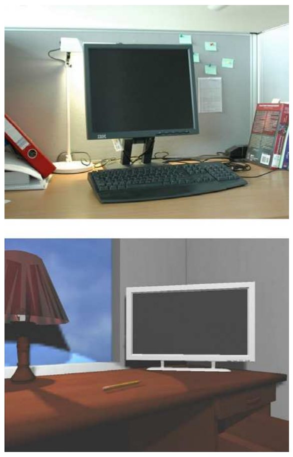
We evaluated the virtual photographs with real pictures taken by a digital SLR camera, the Nikon D70 (cf. Fig. 15). The camera parameters have been equally set and lead to very similar images. Further results can be found in [12].

6 CONCLUSION

In this paper we presented the vSLRcam. The application simulates a real SLR camera by realistically ap-


(a) $a = 3{,}5\,f, t = \frac{1}{200}$


(b) $a = 3{,}5\,f, t = \frac{1}{60}$


(c) $a = 3{,}5\,f, t = \frac{1}{30}$

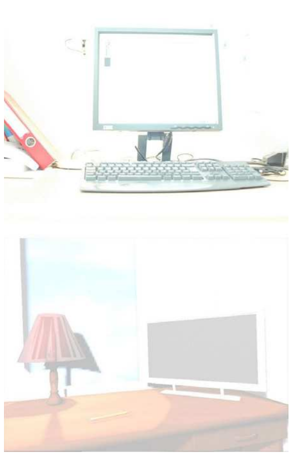

(d) $a = 3{,}5\,f, t = \frac{1}{4}$

Figure 15: Comparison of the vSLRcam with a real Nikon D70. The upper images were taken by the Nikon D70 whereas the lower images are virtual shots. You can see that the scene illuminances correspond very well. The images (a) and (b) show under-exposure and correct illumination whereas images (c) and (d) show slight and strong over-exposure. The camera settings have been the same for real and virtual picture taking leading to very similar results with our technique.

proximating its individual components and allows for the real-time rendering of photographic effects due to the camera's parameter settings. The integration of the vSLRcam application into our virtual training environment allows for the users to associate with a common SLR camera's functionality. Moreover, this is especially supported by the realistic looking photographs the virtual camera generates. Besides, the camera effects could also be made a part of the virtual environment as they can be rendered in real-time. Furthermore, the camera application could be used for cinematographic purposes, e. g. to train camera views or to simulate tracking shots. Firstly, however, we want to improve the user interface to become more intuitive.

ACKNOWLEDGEMENTS

We would like to thank our colleagues from the Police University of Applied Science Saxony-Anhalt, especially Peter Eichardt for his support on this topic.

REFERENCES

[1] Apple Computer, Inc. *Aperture - Digital Photography Fundamentals*, 2006.

[2] Soonmin Bae and Frédo Durand. Defocus Magnification. *Computer Graphics Forum*, 26(3), 2007.

[3] Brian A. Barsky, Daniel R. Horn, Stanley A. Klein, Jeffrey A. Pang, and Meng Yu. Camera Models and Optical Systems Used in Computer Graphics: Part I, Object-Based Techniques. In *ICCSA (3)*, pages 246–255, Berlin, 2003. Springer.

[4] Angela Brennecke, Stefan Schlechtweg, and Thomas Strothotte. Opencrimescene Review Log: Interaction Log in a Virtual Crime Scene Investigation Learning Environment. In *GRAPP 2007*, pages 185–190. INSTICC Press, 2007.

[5] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Procs. of ACM SIGGRAPH '84*, pages 137–145, New York, NY, USA, 1984. ACM.

[6] Paul Haeberli and Kurt Akeley. The accumulation buffer: Hardware support for high-quality rendering. In *Procs. of ACM SIGGRAPH '90*, volume 24, pages 309–318, New York, NY, USA, 1990. ACM.

[7] Brian Hawkins. *Real-Time Cinematography for Games*. Charles River Media, Boston, MA, USA, 2005.

[8] Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. An image-based model for realistic lens systems in interactive computer graphics. In *Procs. of Graphics Interface '97*, pages 68–75, Toronto, Ontario, Canada, 1997. Canadian Information Processing Society.

[9] Michael Kass, Aaron Lefohn, and John D. Owens. Interactive depth of field using simulated diffusion. Technical Report 06-01, Pixar Animation Studios, January 2006.

[10] Martin Kraus and Magnus Strengert. Depth-of-Field Rendering by Pyramidal Image Processing. *Computer Graphics Forum*, 26(3), 2007.

[11] Jost J. Marchesi. *Handbuch der Fotografie*, volume 1. Verlag Photographie AG, Schaffhausen, 1993.

[12] Christian Panzer. Die virtuelle Spiegelreflexkamera für die Tatortfotografie. Diploma Thesis, Otto-von-Guericke University of Magdeburg, Germany, July 2007.

[13] Michael Potmesil and Indranil Chakravarty. A lens and aperture camera model for synthetic image generation. In *Procs. of ACM SIGGRAPH '81)*, volume 15, pages 297–305, New York, NY, USA, 1981. ACM.

[14] Michael Potmesil and Indranil Chakravarty. Modeling motion blur in computer-generated images. In *Procs. of ACM SIGGRAPH '83)*, volume 17, pages 389–399, New York, NY, USA, 1983. ACM.

[15] Thorsten Scheuermann and Natalya Tatarchuk. Advanced Depth of Field Rendering. In Wolfgang Engel, editor, *ShaderX3: Advanced Rendering Techniques in DirectX and OpenGL*. Charles River Media, Cambridge, MA, 2004.

[16] C. Shimizu, A. Shesh, and B. Chen. Hardware Accelerated Motion Blur Generation. Technical report, Department of Computer Science, University of Minnesota at Twin Cities, 2003.

[17] Mark Pullen Tianshu Zhou, Jim X. Chen. Accurate Depth of Field Simulation in Real Time. *Computer Graphics Forum*, 26(1):15–23, 2007.