Photorealistic Rendering of Rain Streaks

Kshitiz Garg Shree K. Nayar
Columbia University *

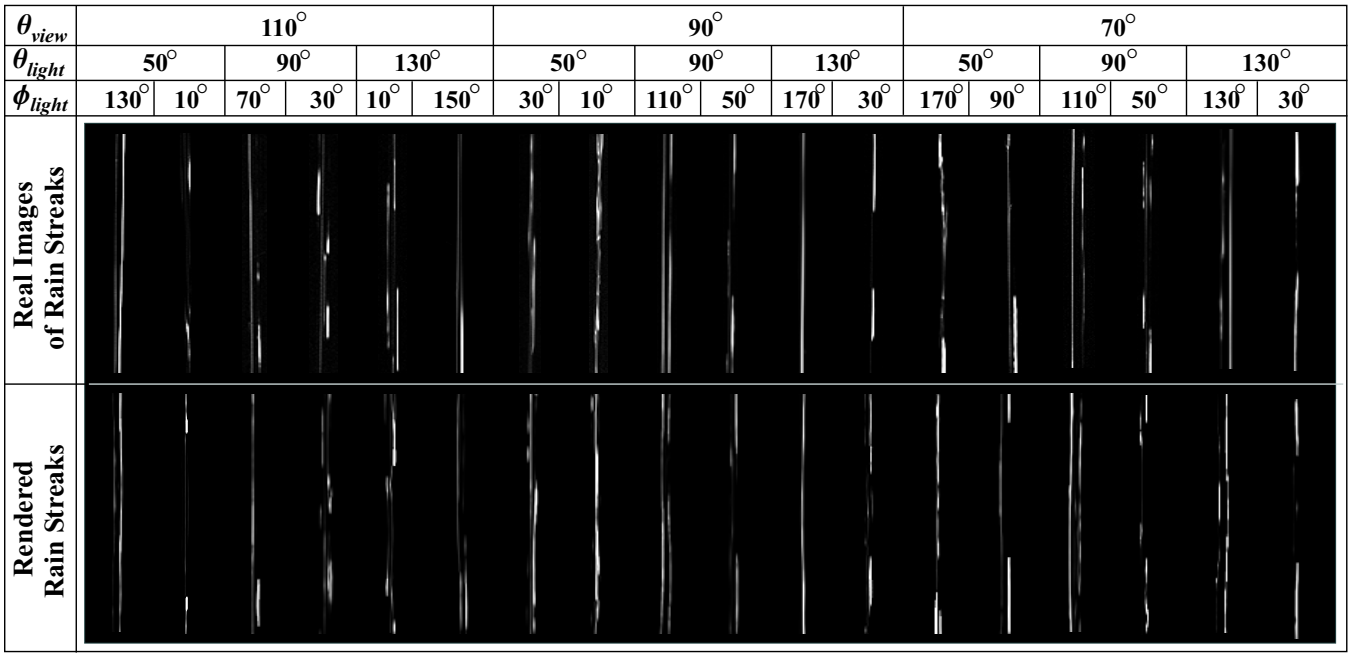


Figure 1: Appearance of actual rain streaks and rendered rain streaks. The top row shows actual images of streaks taken under many different lighting directions ($\theta_{light}, \phi_{light}$) and viewing directions θ_{view}. The complex intensity pattern within each rain streak is due to the interaction of light with the shape distortions (i.e. oscillations) of the drop as it falls. We have empirically determined the oscillation parameter values that are dominant in raindrops and used them to develop a rain streak appearance model. The bottom row shows rain streaks rendered using our appearance model. Each rendered streak has been cropped in order to align the phase of its intensity pattern with that of the actual image. Based on our rain streak model, we have developed an image-based rendering algorithm that can add photorealistic rain to images as well as videos with changing lighting and viewpoint.

Abstract

Photorealistic rendering of rain streaks with lighting and viewpoint effects is a challenging problem. Raindrops undergo rapid shape distortions as they fall, a phenomenon referred to as oscillations. Due to these oscillations, the reflection of light by, and the refraction of light through, a falling raindrop produce complex brightness patterns within a single motion-blurred rain streak captured by a camera or observed by a human. The brightness pattern of a rain streak typically includes speckles, multiple smeared highlights and curved brightness contours. In this work, we propose a new model for rain streak appearance that captures the complex interactions between the lighting direction, the viewing direction and the oscillating shape of the drop. Our model builds upon a raindrop oscillation model that has been developed in atmospheric sciences. We have measured rain streak appearances under a wide range of lighting and viewing conditions and empirically determined the oscillation parameters that are dominant in raindrops. Using these parameters, we have rendered thousands of rain streaks to create a database that captures the variations in streak appearance with respect to lighting and viewing directions. We have developed an efficient image-based rendering algorithm that uses our streak database to add rain to a single image or a captured video with moving objects and sources. The rendering algorithm is very simple to use as it only requires a coarse depth map of the scene and the locations and properties of the light sources. We have rendered rain in a wide range of scenarios and the results show that our physically-based rain streak model greatly enhances the visual realism of rendered rain.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture;

Keywords: raindrops, oscillations, rain streak appearance, rain streak database, particle system, rain rendering.

1 Introduction

Rain is often used in movies and animations to express the mood of a scene. For example, in the movies "Seven" and "The Matrix Revolutions" rain was used to highlight a sense of unrest. Filming rain scenes is, however, a laborious and expensive task that requires setting up sprinklers and light sources over a large physical area. The shooting of a single rain scene can take several days. Due to the high costs involved, it is often impractical to include rain scenes in small-budget movies. For these reasons, a simple algorithm for photorealistic rendering of rain is highly desirable. It would provide a convenient and inexpensive means to add rain effects in movies and animations. In addition, it would allow a film-maker to control the visual effects of rain during post-production. Photorealistic rain rendering can also be used to add visual realism in other graphics applications such as games.

*e-mail: {kshitiz,nayar}@cs.columbia.edu

Several methods for rendering rain have been developed in computer graphics, some of which are available in commercial softwares such as Maya, 3D Studio Max and Inferno. These methods use a particle system [Reeves 1983; Sims 1990] to simulate with a high degree of realism the motions and spatio-temporal distribution of the raindrops. Since each rendered image is assumed to have a finite integration time, the falling raindrops produce motion-blurred rain streaks in the image. However, the above rain rendering softwares are limited as they use very simple photometric models to render the appearances of individual rain streaks. Most often, the streaks are assumed to have simple shapes, such as rectangles or ellipses, and the brightness of each streak is assumed to be constant. Rain rendering methods that do not use particle systems have been developed [Starik and Werman 2003],[Langer et al. 2004],[Wang and Wade 2004] with the aim of reducing the computational cost associated with simulating the motions and spatio-temporal distribution of raindrops. However, like the methods based on particle systems, these too use very simple photometric models for rain streaks. Such simple photometric models can only be used when the rendered rain is at a great distance from the camera, in which case, all the streaks are thin enough to make the details of their brightness patterns irrelevant.

In close-up shots of rain, however, each raindrop projects to a large image streak, revealing the intensity pattern within it (see top row of Figure 1). This pattern is highly complex because of shape distortions that the raindrop undergoes as it falls. These shape distortions are due to oscillations induced by aerodynamic forces and surface tension. The interaction of the shape distortions with light result in speckles, multiple smeared highlights and curved brightness contours within the rain streak. Hence, for close-up shots, the previously used constant-brightness streak model produces unrealistic rain appearance. To address this problem, researchers have used hand-drawn textures of streaks for rendering rain close to the camera. A recent example of the use of hand-drawn textures is the movie "The Matrix Revolutions" [Lomas 2005]. This approach is clearly cumbersome, as the range of appearances of rain streaks is rather wide. In addition, streak appearance also varies significantly with lighting and viewpoint directions, making it harder to use hand-drawn textures for scenes that include lighting and viewpoint changes. We have learned from experts in the special effects and animation industry [Lomas 2005; Reed 2005] that automatic rendering of photorealistic rain remains an open and important problem.

In this work, we provide a comprehensive framework for rendering photorealistic rain streaks in images as well as videos with lighting and viewpoint changes. The following are our key contributions:

Rain Streak Appearance Model: We have conducted what we believe to be the first detailed study of the visual appearance of rain streaks. We have developed a model for rain streak appearance that captures the complex interactions between the lighting direction, the viewing direction and the raindrop oscillations. These interactions produce a wide range of striking visual effects that simply cannot be tabulated by hand. A few examples of rain streaks captured by a camera, corresponding to different lighting and viewing directions, are shown in the top row of Figure 1. Our work builds upon an oscillation model that was developed in atmospheric sciences [Tokay and Beard 1996; Andsager et al. 1999; Kubesh and Beard 1993]. This model does not specify the values of the oscillation parameters corresponding to real rain, which is essential for rendering rain streaks. To this end, we have collected a total of 810 real images of rain streaks under different lighting and viewing conditions. By visually comparing this set of measured streaks with streaks rendered with a wide range of oscillation parameters, we have determined the parameter values that are dominant in real rain. These parameter values enable us to render highly realistic appearances of rain streaks, a few examples of which are shown in the bottom row of Figure 1.

Database of Precomputed Rain Streak Textures: Since the appearance of a streak is complex, it needs to be rendered using a method such as ray-tracing. Therefore, the rendering of several thousands of streaks for each frame of a rain scene would be computationally prohibitive. Our approach is to render off-line and store a database of rain streaks. This database is then used by an efficient image-based algorithm to render rain scenes. The appearance of rain streaks depends on many factors - the lighting and viewing directions; the distances from the source and the camera; the oscillation parameters; the size of the drop and the camera's exposure time. A database that captures the effects of all of these parameters would be too large to store and use. Fortunately, the distances from the source and the camera, the size of the drop and the camera's exposure time produce simple transformations to the streak appearance, that can be efficiently rendered on-line. Hence, we only need to capture the effects of the lighting direction, the viewing direction and the oscillation parameters in the database. Our database includes about 6300 rendered streaks and is publicly available[1]. This database is similar in spirit to the hand-drawn textures used in the previous work. However, it captures a significantly wider range of streak appearance variability and is derived from a physics-based oscillation model.

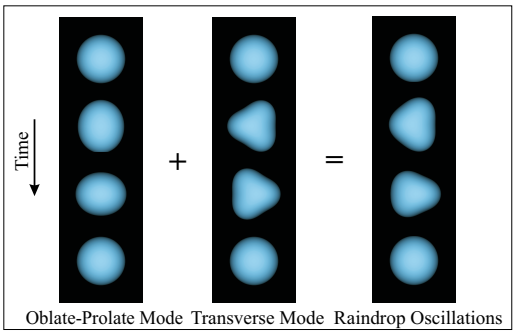


Figure 2: Oscillation model for rain. Raindrops undergo rapid shape distortions (i.e oscillations) as they fall. These oscillations are mainly due to combination of the two spherical harmonic modes – the oblate-prolate mode and the transverse mode.

Image-Based Rain Rendering Algorithm: We have developed an image-based rendering algorithm that applies simple transformations to the streaks in our database, to render streaks for novel lighting and camera parameters. Our algorithm can be used to add rain to a single image of a scene or to a captured video with moving objects and sources. The user specifies a rough depth map of the scene and the properties of the light sources. Once this is done, the algorithm can render rain with user-specified rain parameters, such as raindrop distribution, raindrop sizes and rain direction. We show results of adding rendered rain to several still images as well as videos with changing illumination and viewing directions. Our results show that our algorithm greatly enhances the realism of rendered rain.

2 Raindrop Oscillation Model

We begin by briefly describing the oscillation properties of falling raindrops. The shape of a drop at time t is denoted by $r[t,\theta,\phi]$, where r is the distance of a point on the drop's surface from its center, and θ and ϕ are the elevation and azimuthal angles of the point with respect to the y-axis and x-axis, respectively. The y-axis is opposite to the direction of the drop's fall and the x-axis is an arbitrary direction that is perpendicular to the y-axis. As a raindrop falls, it undergoes rapid shape distortions over time, as shown in Figure 2. These distortions are caused by the aerodynamic forces and the surface tension acting on the drop [Tokay and Beard 1996; Kubesh and Beard 1993]. The shape of oscillating raindrops can be expressed as a combination [Frohn and Roth 2000] of spherical harmonic modes[2]:

$$r[t,\theta,\phi] = r_0(1+\sum_{n,m} A_{n,m} sin(\omega_n t) P_{n,m}(\theta) cos(m\phi)) \,, \qquad (1)$$

where, r_0 is the undistorted radius (referred to as the drop size), $A_{n,m}$ is the amplitude of the spherical harmonic mode (n,m) and $P_{n,m}(\theta)$

[1] To request a copy, please send an e-mail to rainstreaks@cs.columbia.edu

[2] This is based on the assumption that the equilibrium shape (without oscillation) of raindrops is spherical.

is the Legendre function that describes the dependence of the shape on the angle θ for the mode (n,m). The frequencies of these modes depend on the order n and drop size r_0, and are given by

$$\omega_n = 2\pi([n(n-1)(n+2)\sigma]/(4\pi^2\rho r_0^3))^{1/2}. \quad (2)$$

Here, σ is the surface tension and ρ is the density of water, both of which are known[3].

Studies in atmospheric sciences [Kubesh and Beard 1993; Andsager et al. 1999] have empirically determined that the oscillations in a raindrop are predominantly confined to two modes: A rotationally-symmetric, oblate-prolate mode $(n=2, m=0)$ and a non rotationally-symmetric, transverse mode $(n=3, m=1)$. Figure 2 shows the shape distortions induced by each of these modes. Thus, the shape of a falling raindrop can be modeled as a combination of these two modes, as shown in Figure 2. This simplifies the raindrop shape model of equation (1) to

$$r[t,\theta,\phi] = r_0(1 + A_{2,0} sin(\omega_2 t) P_{2,0}(\theta) + A_{3,1} sin(\omega_3 t) cos(\phi) P_{3,1}(\theta)).$$

Note that the shape is not rotationally symmetric due to the $cos(\phi)$ factor in the second term of the above equation. It depends on the drop size r_0, the amplitudes $A_{2,0}$ and $A_{3,1}$ and the frequencies ω_2 and ω_3 of the two modes. The frequencies ω_2 and ω_3 depend on the drop size (see equation (2)) and are higher for smaller drops. Equation (2) also shows that the frequency of the transverse mode is approximately twice that of the oblate-prolate mode, i.e., $\omega_3 \approx 2\omega_2$. Since the oscillation frequencies are determined by the drop size, the shape of an oscillating drop depends only on three parameters, namely, $r_0, A_{2,0}$, and $A_{3,1}$. Previous studies[4] have not quantified the amplitudes $A_{2,0}$ and $A_{3,1}$ of the individual modes. In our context, we need to know the values of these amplitudes in order to render the appearances of rain streaks. In the next section, we describe how we have empirically determined these values.

3 Rain Streak Appearance Model

In this section, we analyze how the oscillations affect the visual appearance of rain streaks. As mentioned before, to model the appearance of the rain streaks we need to know the amplitudes $A_{2,0}$ and $A_{3,1}$ of the raindrop oscillation modes. We empirically determine these amplitudes, by capturing actual images of falling drops under many different lighting and viewing directions and comparing these images with a large number of rendered streak appearances.

Rendering Motion-Blurred Rain Streaks: Since we compare actual streak images with rendered ones, we first describe our method for rendering the motion-blurred streak. The appearance of a rain streak can be rendered by ray-tracing images of a transparent falling drop[5] at different time instants within the camera's exposure time and adding these images. In general, to avoid artifacts, the number of rendered images must equal the length M (in pixels) of the final rendered streak. Since the shape of a drop changes smoothly, we are able to render a smaller number N of images at a sparser set of time samples that are uniformly spaced over the exposure time of the camera. The appearance of a drop at any time t can then be computed by taking a time-weighted average of the neighboring rendered images. Adding the images for all the M discrete locations of the drop gives the appearance of the streak. In our implementation, we have used $M = 1042$ and found that artifact-free streaks can be obtained using $N = 46$.

[3]The following values are used: $\sigma = 0.0728 N/m^2$ and $\rho = 1000 kg/m^3$.

[4]Previous studies have focused on measuring the average axis-ratio (ratio of maximum to minimum radius) of raindrops that are caused by drop oscillations. These average axis-ratios are used to correct for biases in rain-rate measurements obtained from a dual-band polarization radar.

[5]We have used the PBRT package [Pharr and Humphreys 2004] for rendering the images of a falling raindrop. We use a matte disk (illuminated by a point light) as a finite area light source. The *direct lighting* surface shader (with maximum recursion depth of 5) is then used to simulate refraction and reflections of the source (disk) through the drop. The drop shape is represented using a triangle mesh with 16200 vertices. To compute the brightness at a pixel 64 samples (using *low discrepancy* sampling method) are taken.

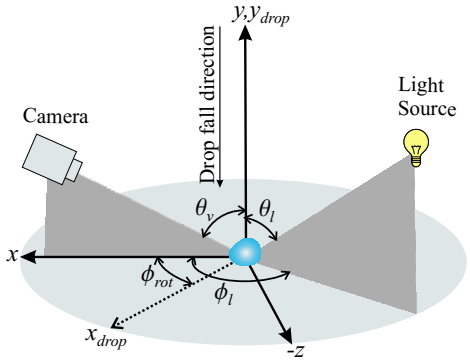


Figure 3: Coordinate system used for both the measurement and the rendering of rain streaks. The coordinate frame is located at the drop's center with its y-axis opposite to the direction of the drop's fall. The x-axis is defined as the projection of the camera's optical axis onto a plane perpendicular to the y-axis. θ and ϕ are the elevation and azimuthal angles, respectively. The subscripts l and v are used to denote lighting and viewing, respectively.

Measurement of Real Streak Appearances: Consider a falling drop viewed by a camera and illuminated by a source, as shown in Figure 3. The appearance of the streak produced by the drop depends on many factors - the source direction (θ_l, ϕ_l), the viewing direction (θ_v) of the camera, the drop's shape parameters $(r_0, A_{2,0}, A_{3,1})$, the drop's distances from the source and the camera, and the camera's integration time. In addition, the streak appearance depends on ϕ_{rot}, the angle that the drop's x-axis makes with the x-axis of the coordinate system, as shown in Figure 3. Since raindrops are asymmetric and have no preferred horizontal orientation, the drop's x-axis varies from one drop to the next. Unlike all the other parameters, the *oscillation parameters* $(Osc = A_{2,0}, A_{3,1}, \phi_{rot})$ are inherently random, resulting in many possible appearances of rain streaks even for a fixed source, camera, and drop size.

To determine the amplitudes $A_{2,0}$ and $A_{3,1}$, we capture images of real falling drops and compare their appearances with streaks rendered under a wide range of values for $A_{2,0}$ and $A_{3,1}$. This is done for several source-camera configurations. From the visual comparison between the actual and rendered streaks, we determine the most likely values for $A_{2,0}$ and $A_{3,1}$. To capture the real streak appearances, we have used an experimental setup similar to the one used by [Andsager et al. 1999] for studying the oscillations of drops. Water drops are released from a large height (15m) to ensure that the oscillations created in falling drops are similar to those in real rain[6]. A large height also ensures that the drops attain a constant (terminal) velocity before they are imaged.

The drops were illuminated by a point light source placed at a distance of 1m from the origin of the coordinate frame (see Figure 3). We captured high dynamic range images of falling drops from a distance of 3m with a Canon EOS-20D camera. The average size of the observed drops was determined to be $r_0 = 2.0$mm from the focal length measurements of the camera lens. Images were captured for many different source directions (θ_l, ϕ_l), some of which are shown in Figure 1. The source elevation angle θ_l was set at $(50°, 90°, 130°)$ and the azimuthal angle ϕ_l was varied from $10°$ to $170°$ in steps of $20°$. To measure the dependence on the viewing direction, we captured images for the viewing angle θ_v at $(70°, 90°, 110°)$. 10 images were taken for each lighting-viewing configuration in order to capture the variation in streak appearance due to the oscillation parameters. In total, we captured 810 streak images.

[6]It has been shown [Kubesh and Beard 1993] that the initial oscillations of a drop (from a drop generator) decay within a few meters of fall. As the drop continues to fall, new oscillations that are similar to those in real raindrops arise due to aerodynamic forces and surface tension.

Determining the Oscillation Parameters: We use the captured streak images to determine the oscillation amplitudes $A_{2,0}$ and $A_{3,1}$. For this we render streaks with many different oscillation amplitudes at $\phi_{rot} = (0^\circ, 90^\circ, 180^\circ, 270^\circ)$ for each lighting and viewing direction. The rendered streaks are then visually compared with the actual streak images to find the amplitudes that give the best match. Our results show that $A_{2,0}$ varies from 0.4 to 0.1 while $A_{3,1}$ varies from 0.2 to 0.05. In particular, the following set of oscillation amplitudes $(A_{2,0}, A_{3,1})$=(0.2, 0.1), and $(A_{2,0}, A_{3,1})$=(0.1, 0.1) produce rendered streaks that match the appearance of actual streaks very well over the different illumination and view directions. Figure 1 shows a comparison between the actual streaks and the streaks rendered using the above amplitude values for different lighting and viewing angles. Note that the rendered streaks look very similar to the actual images. Since, in practice, the amplitude values may vary slightly from the above two sets of values, an exact match between rendered and actual streaks cannot be expected.

Our analysis of the effects of oscillations on the appearance of streaks has led to a few important observations: (a) Raindrops do indeed undergo strong oscillations. Therefore, a spherical drop model is simply not adequate when rendering close-by rain streaks; (b) As mentioned in Section 2, the oscillation frequencies are related as $\omega_3 \approx 2\,\omega_2$. Thus, in any given drop, the same shape distortions repeat after a time period of $2\pi/\omega_2$, i.e., under distant lighting and viewing conditions, the streak appearance is periodic. Hence, streak appearance is essentially captured within a single time period.

4 Database of Rendered Streak Textures

Using the determined oscillation parameter values, we render a database of rain streak textures that captures the wide variation of streak appearance with respect to lighting and viewing directions. It would have been ideal to capture actual images of rain streaks to build the database. However, this is very difficult to do for the following reasons. First, although we have determined the range of oscillation amplitudes, the parameter ϕ_{rot} in *Osc* is a random variable and cannot be controlled. Moreover, capturing high quality streak images under a wide range of source and camera orientations is a challenging problem due to the limited depth of field and dynamic range of the camera. Hence, we use our streak appearance model to render the images in our database – it provides us full control over the sampling of our parameter space.

To capture the effects of lighting[7], we varied θ_l from 0° to 180° and ϕ_l from[8] 10° to 170° with steps of 20°. To capture the effects of the viewing direction, we varied θ_v also from 10° to 170° with steps[9] of 20°. To include the effects of oscillations, we used both sets of estimated amplitudes, $(A_{2,0}, A_{3,1})$=(0.2, 0.1) and $(A_{2,0}, A_{3,1})$=(0.1, 0.1). The orientation of the drop with respect to the camera was sampled as $\phi_{rot} = (0^\circ, 90^\circ, 180^\circ, 270^\circ)$ to account for the non-rotationally symmetric shape of the drop. For all of the above lighting and viewing directions, we also rendered streaks for the following two cases - spherical drops (no oscillations) and only oblate-prolate oscillation with amplitude $A_{2,0} = 0.2$ (no transverse oscillations).

The streaks are rendered using drop size $r_0 = 1.6$mm and exposure time $T_{db} = 1/60$s, which corresponds to the time period $2\pi/\omega_2$ of the drop oscillation. In short, the database samples the parameter space $(\theta_l, \phi_l, \theta_v, Osc)$. It also includes streaks rendered under ambient lighting for the (θ_v, Osc) values used above. The streaks are rendered as 16-bit monochrome images with a resolution of 32×1042 pixels[10]. In total, the database includes 6300 streak images at this resolution. To avoid artifacts due to severe down-sampling when rendering streaks far from the camera, the above streak images were also stored at 3 additional (lower) resolutions with streak widths of 16, 8 and 4. The complete database is about 400MB in size. The database can be used by any image-based rendering system and is publicly available.

5 Algorithm for Rain Rendering

We now present an efficient image-based algorithm that uses our streak texture database for rendering photorealistic rain in images and videos. The textures in the database were rendered by varying the oscillation parameters and the lighting and viewing directions. The variation of streak appearance due to other factors such as drop size, distance from light sources and the camera, and properties of the sources, can be computed from the textures in the database using simple image processing operations. In this section, we show how our rain rendering algorithm computes these textures. In addition, we describe how the algorithm renders the effects of camera parameters, such as defocus and exposure time. Figure 4(A) shows all the steps involved in the rendering process. We now describe each of these steps (from left to right in Figure 4(A)).

User Inputs: Our rendering algorithm requires an image/video of the scene, a coarse depth map of the scene, the locations and properties (size, isotropy/anisotropy, color, orientation) of the light sources and the camera parameters. The user also specifies the rain properties, including the density of rain, the drop size distribution[11], and direction of rainfall. For videos with moving light sources and objects, the user provides coarse depth layers and locations and orientations of the source for key frames in the video. Then, interpolation is used to compute the locations of the depth layers and the locations and orientations of the sources in the remaining frames of the video[12].

Rain Dynamics: We use the particle systems API [McAllister 2000] to create the spatio-temporal distribution of raindrops in 3D space and their trajectories. For efficiency, we only generate raindrops that pass through the field of view of the camera. Drops are created with uniform spatial distribution in a 3D volume that lies above the camera's field of view. At each time frame, we add new drops to this volume and update the 3D positions of existing drops. This gives the spatio-temporal distribution of drops. Each drop is also randomly assigned oscillation parameters *Osc* from the set of parameters used to create our streak database.

Rendering Novel Streak Textures: A scene can have multiple light sources. To render rain in such a scene, we compute the streak textures due to each light source separately. Computing these textures involves accounting for the effects of drop size, the properties of the light sources and distances from light sources and the camera. To render a rain streak we first find the appropriate texture resolution level – it is the resolution level with textures of widths just larger than the width of the projected rain streak.

Lighting and Viewing Dependence: As the drop falls during the exposure time, the source angle (θ_l, ϕ_l) and the camera angle θ_v change with respect to the drop. For a drop that is far from the light source and the camera, these angles do not vary much over the length of the streak and hence the streak texture for the angle $(\theta_l, \phi_l, \theta_v)$ can be used. If there is no streak texture in the database for these specific angles, then the texture is obtained by bilinear interpolation of the 8 neighboring (two along each of the three angular dimensions)

[7] The light source subtends a solid angle of approximately 14 steradian at the center of the drop and is at a distance of 10m from the drop.

[8] Due to symmetry, the streak texture for ϕ_l, $180^\circ < \phi_l < 360^\circ$, is obtained by horizontally flipping the texture for $(\phi_l - 180^\circ)$.

[9] The data suggests that a coarse sampling of 20° is sufficient for rendering rain streaks. This may be because drops are only a few pixels wide and severely motion-blurred. In applications where a finer sampling is needed, our streak appearance model can be used to populate the database with any desired sampling.

[10] The lengths of the streaks for $\theta_v \neq 90^\circ$ are smaller since the viewing direction of the camera is not orthogonal to the fall direction of the raindrops.

[11] Well known raindrop size distributions such as Marshal-Palmer or Gamma distribution can be used. The size distribution can also be customized to include larger drop sizes to create more dramatic rain effects. This is analogous to the technique used in movies to simulate a heavy downpour – using sprinklers that produce large drops.

[12] Alternatively, structure from motion algorithms [Faugeras 1993] may be used to automate this process.

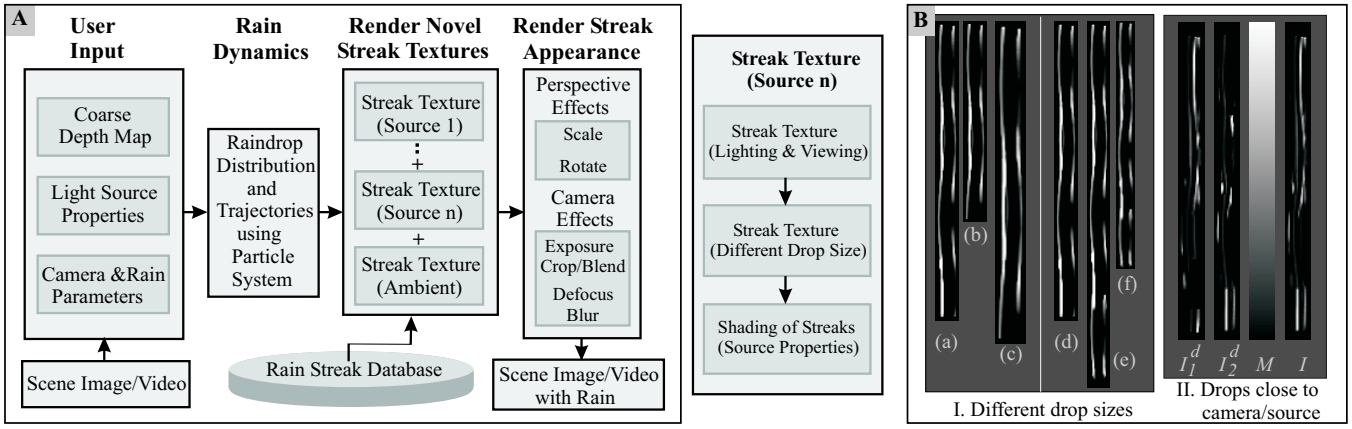


Figure 4: (A) Rain rendering pipeline. We have developed a fully automated algorithm for inserting rain in images and videos. The user provides an image or a video of the scene, a coarse depth map of the scene, the properties of the light sources, and the camera and rain parameters. The algorithm uses a particle system for creating the rain distribution and the drop trajectories. Then, it uses textures from our rain streak database to render novel rain streaks. Novel streaks are rendered for each light source. The box on the right shows details of the steps involved for each source. These steps account for the effects of drop size, distances from the source and the camera, and the properties of the source. The streak textures due to individual sources and the ambient light texture are added to obtain the final streak texture. These final textures are then scaled and rotated to account for perspective effects and blurred and cropped to account for camera defocus and exposure time. (B) Steps to compute the texture for (I) a drop whose size is different from drops in the database (II) a drop near a camera or source.

database textures. For a drop that is close to the light source or the camera, $(\theta_l, \phi_l, \theta_v)$ will change over the streak. Hence, the database textures cannot be directly used (even with interpolation) to obtain the desired streak. In such cases, we compute two streak textures, I_1 and I_2, using the angles corresponding to the top and the bottom of the streak. These two textures are later blended (after accounting for drop size) to obtain the texture for such a drop.

Different Drop Sizes: The appearance of a streak also depends on the drop size. Equation (2) shows that the size of a drop only affects the oscillation frequency, implying that drops of different sizes are subjected to the same oscillations as in the rendered database streaks. Therefore, a drop with a different size (from the database drop size) will have its intensity pattern represented in the database, but takes a different time period $T_{new} = 2\pi/\omega_2$ to complete its oscillation. Due to this difference in the time period, within the exposure time T_{exp} of the camera, only a T_{exp}/T_{new} portion of the database streak texture is visible. Hence, the textures for new drop sizes can be obtained from database textures using simple operations as described below. Consider a database texture as shown in Figure 4(B)I(a)(also shown in Figure 4(B)I(d)). If $T_{new} > T_{exp}$, the drop's streak is obtained by cropping a T_{exp}/T_{new} fraction of the database streak as shown in Figure 4(B)I(b). If $T_{new} < T_{exp}$, the drop's streak is obtained by concatenating copies of the database streak[13] and then cropping at the appropriate length as shown in Figure 4(B)I(e). These cropped/concatenated streaks are scaled depending on the drop size and velocity. The streaks ends are then blurred to smooth out the sharp edges produced by cropping. This gives us the streak texture for the given drop size (see Figure 4(B)I(c),(f)).

As discussed before, when a drop is close to the source two textures I_1 and I_2 are computed. In such cases, we account for drop size to obtain corresponding textures I_1^d and I_2^d. These are then blended using a mask M that varies linearly from 1 at the top of the streak to 0 at the bottom as shown in Figure 4(B)II. The resulting texture I thus incorporates the effect of changing camera or source angle within a streak and is given by: $I = I_1^d M + (1-M) I_2^d$.

Light Source Properties: Rain produces important visual effects, such as halos around isotropic light sources and light cones around anisotropic sources. To render these lighting effects, we modify the streak texture computed above by multiplying it with a mask. To create a halo effect, we use a mask whose intensity at a pixel i is equal to $1/d_i^2$, where d_i is the distance in 3D of the falling drop (with its center assumed to project to pixel i) from the light source[14]. Similarly for rendering the effects of anisotropic sources, such as spotlights, we multiply the original texture with an anisotropic mask that represents the anisotropic brightness distribution of the source.

We repeat the above steps for each light source in the scene. Then, we obtain the ambient streak texture by indexing the database using (θ_v, Osc). If a database texture for θ_v does not exist, we compute the texture by interpolating the neighboring database textures in θ_v. The ambient streak texture is then modified to account for the drop size, as explained before.

Once the monochromatic textures due to individual light sources and the ambient light are obtained, we scale each of these textures individually with the corresponding source intensity and color. These scaled textures are added to obtain the final colored streak texture.

Rendering Streak Appearance: Once the final streak is obtained, we apply simple transformations to it to account for the perspective and camera effects.

Perspective Effects: Based on the drop's distance from the camera and the angle that drop's velocity vector makes with the camera's optical axis, we scale the final streak texture to its projected size in the image. The projected streak is then rotated in the image plane to account for its direction of fall.

Camera Effects: Next, we render the effects of the camera's limited depth of field (defocus). If the scaled and rotated rain streak lies outside the depth of field of the camera, it is Gaussian-blurred with a depth-dependent sigma. Finally, we account for the effects of the camera's exposure time T_{exp}. The effect of the exposure time on the length of the streak was already accounted for when we computed the texture for the drop size. Here, we only consider the effect of exposure time on the transparency of the rain streak. It has been shown [Garg and Nayar 2005] that the intensity I_r at a rain-affected pixel is given by $I_r = (1-\alpha) I_b + \alpha I_{streak}$, where I_b and I_{streak} are the intensities of the background and the streak at a rain-effected pixel. The transparency factor α depends on drop size r_0 and velocity v and is given by $\alpha = 2 r_0 / v T_{exp}$. Also, the rain streaks become more

[13] For long exposure times, the streak texture repeats itself with the time period of oscillation.

[14] In order to minimize the database size we use this approximate method to render the variation in streak appearance with source distance. While this gives reasonable results, in applications where more accurate results are needed one could include the appearance variation with source distance in the database.

opaque with shorter exposure time. In the last step, we use the user-specified depth map of the scene to find the pixels for which the rain streak is not occluded by the scene. The streak is rendered only over those pixels.

The above steps are repeated for each raindrop that lies in the camera's field of view. The run-time of the algorithm is linear with respect to the total number of pixels affected by rain. That is, it depends not only on the number of streaks, but also on the size of the projected streaks. For a typical rain scene, a single frame takes about 10 seconds to render on a 2GHz Pentium-4 machine with 1 GB RAM.

6 Results

We now show the results of our rain rendering in a wide range of scenarios that include changing lighting and viewing conditions.

Scene 1 – Varying Illumination Direction: We begin by showing rain rendered against a black background to highlight the variation in rain appearance with illumination direction. Figure 5 I(a) shows the appearance of rain when it is illuminated by a distant source that makes an azimuthal angle of 60° with the camera. Figure 5 I(b) shows the rain appearance when illuminated at 10°. 0° corresponds to the camera's viewing direction. Note the strong dependence of streak appearance on the illumination direction. For comparison, we show rain rendered using the constant brightness model in Figures 5 I(c) and (d), with identical scene settings and rain parameters. As can be seen, the constant brightness model looks unrealistic since it fails to produce the complex streak patterns and the effects of changing illumination.

Scene 2 – Night Scene with an Isotropic Source: Next, we show rain added to a simple night scene with an isotropic light source that is immersed in rain. Our algorithm can realistically reproduce illumination effects such as the halos around sources, as seen in Figure 5 II(a). Figure 5 II(b) shows a zoomed-in image of the scene that reveals the complex patterns within the streaks. Our algorithm can also simulate camera effects such as defocusing. Figures 5 II(c) and (d) show the scene rendered with a large aperture size. In Figure 5 II(c) the camera is focused on the lamp and in II(d) it is focused in front of the lamp.

Scene 3 – Colored and Anisotropic Sources: Figure 5 III shows rain added to a scene with multiple light sources, including anisotropic traffic lamps. Note the realistic shading of the rain due to the traffic lights and how the appearance of the streaks changes with illumination direction. Figure 5 III(b) shows a zoomed-in shot of the same scene. Figure 5 III(c) shows rain rendered for the same scene using the constant brightness streak model. Figures 5 III(b) and III(c) clearly demonstrate the advantages of using our streak appearance model.

Scene 4 – Changing Sky Illumination: Figure 5 IV shows rain added to a scene where the illumination changes from an overcast sky to bright sunlight. The streaks have smooth intensity patterns when illuminated by ambient lighting, such as the overcast sky, as seen in Figure 5 IV(a). However, as the illumination changes to highly directional sunlight in Figure 5 IV(c), we see that high frequency intensity patterns appear within the streaks. Figure 5 IV(d) shows the selected image regions at the original resolution used for rendering. The differences in the streaks for the three lighting conditions are clearly visible.

Scene 5 – Moving Light Sources: Figure 5 V shows rain rendered for a scene with a moving car with headlights. Note how the appearance and brightness of the rain streaks vary as the headlights illuminate the rain from (a) the front of the camera and (b) the side of the camera. Figure 5 V(c) shows the selected image regions at the original resolution used for rendering.

These results demonstrate the ability of our physically-based rain rendering algorithm to add realistic looking rain to images and videos.

7 Conclusion and Future Work

We have proposed a model for rain streak appearance that captures the complex interactions between the lighting direction, the viewing direction and the oscillating shape of a falling drop. We have used this model to create a useful database of rain streaks that includes the wide variability in streak appearance. This database stands on its own as a contribution as it can be used by any image-based algorithm for rain rendering. Finally, we have developed a rain rendering algorithm that can be used to add realistic rain in images and videos. The algorithm requires minimal human input, provides a high degree of control, and is efficient. It provides an inexpensive way to add rain in movies and animations. This method is also applicable to rendering streaks produced by drops falling from reasonable heights (such as rooftops, terraces, trees, etc.) as they exhibit oscillations similar to those in raindrops. In future work, we would like to incorporate a more sophisticated dynamic model that can render the effects of turbulence and wind and also extend the algorithm to render curved streaks. Finally, we would like to achieve frame-rate performance by using commodity graphics hardware. We are also interested in the modeling and rendering of the splashing of raindrops against a wide variety of materials, a visual effect that is important for making scenes with rendered rain appear photorealistic.

Acknowledgements

This work was conducted at the Computer Vision Laboratory at Columbia University. It was supported in part by a grant from the Office of Naval Research (No. N00014-05-1-0188) and a grant from the National Science Foundation (No. IIS-04-12759). The authors are grateful to Andy Lomas of Dreamworks Animation and Michael Reed of Blue Sky Studios for the valuable information they provided on the methods used in the movie and animation industry for rendering realistic rain.

References

ANDSAGER, K., BEARD, K. V., AND LAIRD, N. F. 1999. Laboratory Measurements of Axis Ratios for Large Raindrops. *Journal of the Atmospheric Sciences 56*, 2673–2683.

FAUGERAS, O. 1993. *Three Dimensional Computer Vision: A Geometric Viewpoint.* MIT press.

FROHN, A., AND ROTH, N. 2000. *Dynamics of Droplets*. Springer.

GARG, K., AND NAYAR, S. K. 2005. When Does a Camera See Rain? *ICCV*, 1067–1074.

KUBESH, R. J., AND BEARD, K. V. 1993. Laboratory Measurements of Spontaneous Oscillations of Moderate-Size Raindrops. *J. Atmos. Sci. 50*, 1089–1098.

LANGER, M. S., ZHANG, L., KLEIN, A. W., BHATIA, A., PEREIRA, J., AND REKHI, D. 2004. A Spectral-particle Hybrid Method for Rendering Falling Snow. In *Rendering Techniques*, 217–226.

LOMAS, A. 2005. The Matrix Revolutions. *Personal Communication*.

MCALLISTER, D. K. 2000. The Design of an API for Particle System. *UNC-CH TR 00-007*.

PHARR, M., AND HUMPHREYS, G. 2004. *Physically Based Rendering*. Morgan Kaufmann.

REED, M. 2005. Blue Sky Studios. *Personal Communication*.

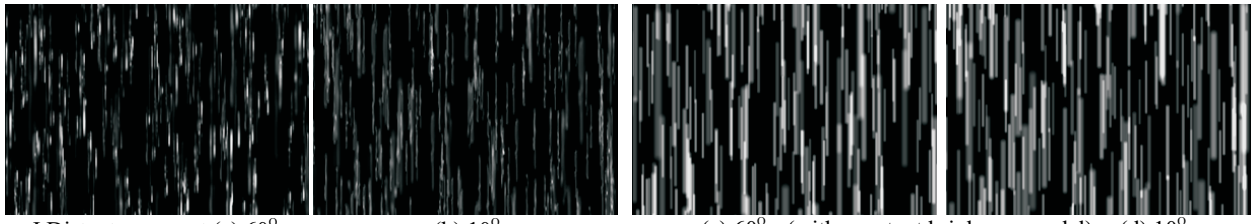
REEVES, W. T. 1983. Particle System- a Techinque for Modeling a Class of Fuzzy Objects. *ACM Trans. Graphics*, 91–108.

SIMS, K. 1990. Particle Animation and Rendering using Data Parallel Computations. *SIGGRAPH 90*, 405–413.

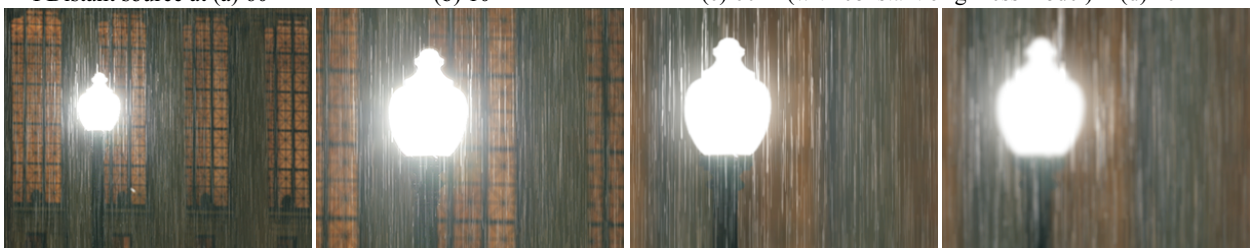
STARIK, K., AND WERMAN, M. 2003. Simulation of Rain in Videos. *Texture Workshop, ICCV*.

TOKAY, A., AND BEARD, K. 1996. A Field Study of Raindrop Oscillations. Part I. *Journal of Applied Meteorology 35*.

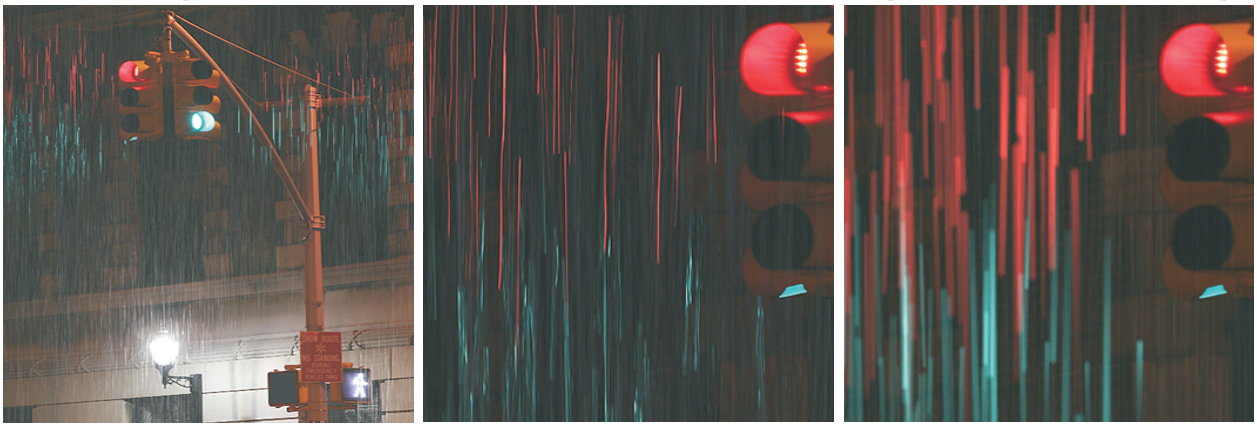
WANG, N., AND WADE, B. 2004. Rendering Falling Rain and Snow. *SIGGRAPH (sketches 0186)*.



I Distant source at (a) 60° (b) 10° (c) 60° (with constant brighness model) (d) 10°



II (a) An isotropic source (b) Zoomed-in shot (c) Focused on the lamp (d) Focused in front of lamp



III (a) Colored and anisotropic sources (b) Zoomed-in shot (c) Constant brightness model

IV (a) Overcast (b) Partial overcast (c) Sunny (d) Magnified Regions

V (a) Illuminated from front (b) Illuminated from side (c) Magnified Regions

Figure 5: Rain rendering results. I Rain rendered against a black background with a distant light source at a horizontal angle of 60° in I(a) and 10° in I(b). Note the change in streak appearance with the lighting. I(c)-(d) Frames with the same camera and rain settings but rendered with constant brightness streaks. II(a) A simple night scene with a single isotropic light source. Notice the halo generated by rain around the light source. II(b) Zoomed-in view of the scene that reveals the complex brightness patterns within the streaks. II(c)-(d). The same scene rendered with camera defocus effects. III(a) Rendered rain added to a scene with several colored and anisotropic light sources. Note the realistic shading effects around the sources. III(b) The complex patterns within the streaks are more visible in the zoomed-in image. Compare our rendering with that using the constant brightness streak model in III(c). IV Scene with changing outdoor illumination. IV(a) Rain rendered for overcast, IV(b) partially overcast and IV(c) sunny conditions. Note the variation in rain appearance as the illumination changes from ambient to directional. IV(d) Magnified images (original image resolution used for rendering) of the selected image regions (white boxes). V A scene with moving light sources. Note the differences in the appearance of rain streaks when illuminated from the direction of the camera viewpoint in V(a) from the front and from the side in V(b). V(c) Magnified images of the selected image regions (red boxes).