Simulation of Rain in Videos

Sonia Starik Michael Werman

School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel
Jerusalem, Israel
E-mail: {starik,werman}@cs.huji.ac.il

Abstract

This paper treats the addition of simulated rain to a video sequence. In order to derive a fast and simple algorithm for rain simulation that produces realistic results, we investigated visual properties of rainfall in videos, in terms of time and space.

Assuming partial knowledge of the intrinsic camera parameters and user-defined parameters regarding rain intensity and velocity, we derive visual properties of the rain "strokes" in the video space and show how to use these strokes to modify the video to give a realistic impression of rain.

1. Introduction

Adding artificial rain to a video is an interesting task, which does not seem to have been addressed much. Several papers dealt with snow simulation, but the task of rain generation is different. A common approach to snow simulation is to build a particle system ([8],[10],[11],[5]), where the particles (snowflakes) are rendered in the 3D world, simulating the snowflakes motion. It is also possible to simulate rain using such a particle model with a motion that describes rain fall behavior (for example, [6] or demo at [1]). However, it turns out to be unnecessary to use such a complicated tool, since the falling trajectory of a raindrop is much simpler than that of a snowflake. In addition, a raindrop's motion is much faster, thus less information about motion between frames needs to be computed. In this work we build a simple model which does not model a 3D scene, but works directly on a video sequence.

We investigate visual properties of a rainfall projected onto a video sequence, in terms of time and space. An interesting observation is that one cannot usually recognize rainfall by looking at a single frame, but it can easily be recognized when looking at a video. Thus, rain can be thought of as a dynamic rather than as a spatial feature. Several research has been done on synthesis of dynamic textures ([2],[9],[4]). In these works models that capture the statistics of a sequence of images are learned and then used to predict future appearances of the textures, thus generating infinitely long sequence from a short input "example" sequence. Rather then learning the behavior of texture samples, we apply rainfall to an input video sequence given only a set of rough parameters defining the desired rainfall intensity and some parameters regarding the camera. We show that temporal independence between the rain frames can be assumed and reduce the problem to generating independent rain frames.

Intuitively, the temporal dependencies between raindrops appearances are essential for creating sequential rain simulation. However, we found out that the raindrops' motion cannot be tracked by human perception (probably due to the fast motion and high density of the drops, and to the discontinuity of the camera shots), a fact discovered by experiments that we carried out. Using videos of real rainfall on stable natural scene backgrounds, we mixed up the video frames in a random order and the results looked as natural as the original videos. Therefore, temporal independence of rain frames can be assumed, reducing the problem to generating a set of independent rain frames.

In order to generate a single rain frame, we assume some rough intrinsic camera parameters (focal length, camera exposure time, etc.) and some user-defined parameters relating to the 3D scene (rain density, rain velocity etc.). We also assume uniform distribution of the raindrops in the world. We use a 3D→2D projective camera model to derive visual properties of the *rain-strokes* (a raindrop looks like a bright stroke on an image, due to its fast falling speed relative to the camera exposure time), as they should appear in the image space (density, intensity, shape etc.). We create a rain mask for each frame in the sequence and then add it adaptively to the background image, based on the background color and the geometric properties of the generated rain-strokes.

To obtain more visual reality, we simulate changes in lighting conditions usually taking place when rain starts, by decreasing the amount of color information in the back-

ground images, creating a darkening effect.

This paper is organized as follows: In Sec. 2 we explain why temporal independence of the rain frames can be assumed. In Sec. 3 parameters regarding rainfall are discussed. Then in Sec. 4 we describe how to model a single rain frame mask, and in Sec. 5 how to apply the mask to a background image. We also describe how to separate rain from a video of static background in Sec. 6. Experimental results in Sec. 7 and discussion and future work in Sec. 8 summarize the work.

2. Temporal Modeling

Intuitively, dependencies among appearances of rain strokes in consecutive frames is essential for sequential rain simulation. However, although human perception of rain is more dynamic than spatial, we found that temporal independence of rain in consecutive frames can be assumed. This assumption is justified by human perceptual experiments we made on rain phenomena.

2.1. Tracking of Raindrops Motion

Finding the temporal dependencies between rain drops is equivalent to tracking their motion from frame to frame. However, the task seems impossible due to the very fast motion of the drops relative to their size and to their high density.

The most distinct drops (closest to the camera) are also the fastest in the video due to the perspective projection, so that their motion cannot be captured, because of the discontinuity of the camera shots. Between two consecutive frames the drop passes a substantial part of the image, and thus there is not much sense in looking at a small neighborhood to discover the drops' motion. The high density of the remote drops, on the other hand, and their similarity to each other, complicates their tracking as well. Hence, it seems that the tracking task is very complicated or even impossible.

As an illustration example, in Fig. 1 we show three consecutive video frames from a rain video. It seems that no explicit correlation between the rain strokes in each pair of consequent frames can be observed, justifying the assumption of strokes' temporal independence.

2.2. Human Perception

The correctness of the temporal independence assumption is also justified by human perception experiments on rain. We took videos of real rain on stable backgrounds of nature scenes, for different rain intensities. For each of these videos, we mixed the frames up in a random order and showed the result to independent spectators. Both the original and the "mixed" movies were considered

(a) Frame 1 (b) Frame 2 (c) Frame 3

Figure 1: Consecutive frames of a rain video. No raindrops tracking is possible.

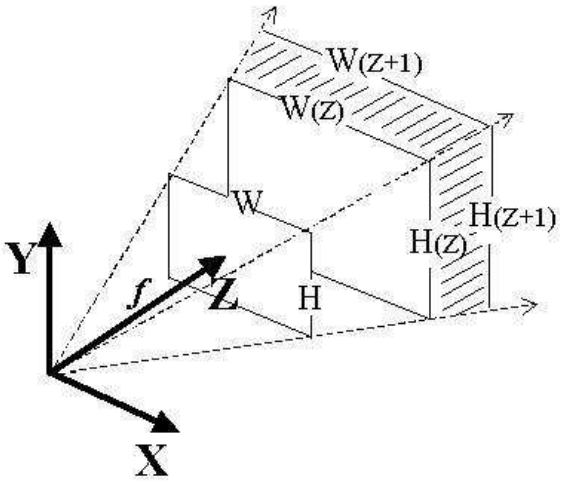

Figure 2: Projective Model.

equally natural-looking by the observers (see examples at *http://www.cs.huji.ac.il/∼werman/Papers/rain/rain.html*)

Another experiment was to take a video of rainfall and to flip it (raining upward). The spectators had difficulties determining the direction of the raindrop motion, although they could recognize the rainfall (see examples at *http://www.cs.huji.ac.il/∼werman/Papers/rain/rain.html*).

Therefore, our assumption is that tracking a raindrop motion is not important in order to create an impression of a rainfall, but rather the repeated appearance of some random spatial pattern. The modeling of such spatial pattern is explained in Sec. 4.

3. Real World Assumptions

We came up with the following set of parameters to determine the visual properties of a rainfall: density, size, speed of the raindrops and the direction of their motion. Of course, dependencies exist between these parameters.

There is meteorological research that examine these parameters [12], [7]. It was found that raindrop size follows an exponential two-parametric distribution [3]. The model shows that each drop (except for the largest ones) achieves its terminal velocity after the first 20 meters of its fall. The value of the terminal velocity is a function of the drop's size and wind velocity.

We built a user-friendly interactive system where the user can generate the desired rainfall by defining a small set of parameters (to do this we simplified the model). The user parameters are: a single raindrop size, a single raindrop speed, rainfall density, wind direction and the background depth. We add some small random perturbations to these parameters to obtain more visual realism.

Despite these simplifications, the raindrop motion that we obtain this way is complicated and chaotic enough to achieve good visual realism.

4. Single Frame Modeling

As we have shown in Sec. 2, the repeated appearance of some random "noise" of the same specific nature from frame to frame is what makes us "see" rain rather than tracking specific raindrop's motion. Consequently, the problem is reduced to modeling this rain "noise" for each frame independently. The model consists of computing the noise form, spatial + shape, and how to adjust this noise to the background image.

4.1. Projective 3D→2D Model

Let us assume that we have a 3D (nature) scene and a projective 3D→2D camera as shown in Fig. 2 where f is the focal length of the camera, z is the depth of the scene and $W \times H$ is the size of the 2D image. The projective transformation from a 3D space into the 2D image space is defined by

$$\begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} \approx \begin{pmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

From this simple projective model and some assumptions we make on the 3D world we derive the density of the rain strokes in the image space and their geometric properties.

It is important to note that in a projective model the quantity of rain-strokes and their shape differ for different depths. So, for a shallow depth less drops are captured by the camera, but these drops are larger.

We assume that the maximal scene depth and the density of the rain drops in the 3D world are user-defined parameters, we also assume drops follow a uniform distribution in 3D, and we assume known camera parameters f, W and H. From all these we derive the drops quantity and their size for each discretely sampled z.

4.1.1. Quantity of rain strokes in an image.

To estimate the quantity of raindrops located at $z < distance \leq (z + \Delta z)$ from the camera, we calculate the volume V of the truncated pyramid marked as dashed in Fig. 2:

$$V(z) = Pyramid(z + \Delta z) - Pyramid(z)$$

$$= \frac{(z + \Delta z) \cdot H(z + \Delta z) \cdot W(z + \Delta z)}{3} - \frac{z \cdot H(z) \cdot W(z)}{3}$$

by similarity of triangles, $W(z) = \frac{z \cdot W}{f}$, $H(z) = \frac{z \cdot H}{f}$ resulting with

$$V(z) = \frac{1}{3} \cdot \frac{W}{f} \cdot \frac{H}{f} \cdot [(z + \Delta z)^3 - (z)^3]$$

Since we assumed known f, W, H, the volume $V(z)$ can be calculated for any z, and given the drops average density we calculate $Quantity(z) = V \cdot Density(Rain)$.

4.1.2. Rain strokes size.

We assume a uniform terminal speed for all the raindrops in the world, V_{drop}. This is a user-defined parameter which determines the strength of the falling rain. We also assume that the camera speed (exposure time) is T_{camera}. Then $length(z) = \frac{f}{z} \cdot T_{camera} \cdot V_{drops}$. Regarding the width, assuming the average raindrop diameter in the world is w_0, we get $width(z) = \frac{f}{z} \cdot w_0$.

In summary, we assumed the following known parameters: focal length and camera speed (intrinsic camera parameters), as well as raindrops' density, their average size and speed and the scene's maximal depth (the world parameters). We assumed drops uniform distribution in the world and calculated the number of raindrops of each size that should appear in the image.

We add some small random perturbations to these parameters (to the amount of raindrops and to the size and speed of each individual drop) to obtain more realistic results.

5. Adjusting of a Rain Mask to a Background Image

The same raindrop looks different when it passes over a light or a dark background. For example, when we want to know if it is raining outside, we seek for a dark wall or trees and try to recognize white strokes over them. When all the background is very bright (like the sky or white walls), the recognition task becomes much more complicated. In the same manner, the visibility of the rain depends on the general brightness of the scene.

Therefore, changes of intensity in a new (with rain added) image depend on the original background image color, as well as on the lighting conditions. In addition, another important factor should be taken into consideration when modifying the image: due to the motion and the "lens effect" of a raindrop the background behind it is blurred. In principal, a physical model for the motion blur can be built, based on the raindrops physical properties, background image properties and lighting conditions. In our work we simplified this assumption and blurred the background behind the drops based only on the raindrops intensity. We implemented this idea by taking a weighted mixture of blurred and original images, weighted according to the raindrop intensity at a given location (as is explained further in Sec. 5.1).

Our *adaptive additive model* takes a "mask" of the generated rain noise, that is, a grey-scaled image of rain strokes of different sizes drawn on a black background (we build this mask according to the assumptions made about the rain fall properties in the 3D world and its projection onto the 2D image space) and adjust it to the background image (originally with no rain). The following assumptions were made in our model:

- Adding a rain stroke to the image increases the intensity of the background pixels.

- The intensity increases in inverse proportion to the background intensity (which means that we need to add more rain color to a dark background and less to a bright one).

- The drops are transparent, meaning that the original pixel color is mixed with the raindrop color.

- A rain-stroke blurs the background behind it.

All the above treats *local* background dependency, but *global* dependency on the general brightness of the scene should be taken into consideration as well. We assume that rain should be clearer on a darker image.

These assumptions are implemented as described below.

5.1. Additive Adaptive Model

Let $Mask$ be a randomly generated grey-scaled image of the rain, according to the single-frame rain generation model. Let BG be a background image to which we want to add the rain. In the resulting image $RainI$ at a pixel x is calculated as follows:

$RainI(x) = BG(x) + Mask \cdot RainColor \cdot (1 - BG(x)) \cdot (1 - mean(BG))$

where $RainColor$ is a constant determining the general rain color (depends on the desired rainfall intensity) and $mean(BG)$ is the average brightness of the background image. The term $(1 - BG(x))$ implements the idea that more rain color should be added to a darker background (local adaptiveness), and the term $(1 - mean(BG))$ expresses the idea of the inverse dependence of the rain color intensity of the general image brightness (global adaptiveness). (Note: the $Mask$ is normalized in order to avoid intensity overflow when adding the mask to the image).

We also blur the background behind the rain-strokes by taking a weighted mixture of a blurred version of rain mask applied image ($RainI$) and the original background (BG), according to the raindrop intensity at a given location: $ResultI = BlurredRainI \cdot Mask + BG \cdot (1 - Mask)$. We have tried a number of different blurring masks to obtain the $BlurredRainI$, all of them produced similar results. In most of our experiments we used $3X3$ convolution mask.

If the image is an RGB image, the same is done independently for each of the 3 channels.

5.2. Effect of Darkening

To obtain more visual realism, we simulate changes in lighting conditions which usually take place when a rainfall starts. However, in order to imitate light scattering during the rainfall, prior information on the 3D scene geometry and light sources are required. Lacking such information, we imitate the loss of color information caused by darkening (color difference decreases as light fades out), which usually occurs when it starts raining.

We implemented this by decreasing the amount of color information in the background image, by moving the intensity value of each pixel x in each of the R, G, B channels towards the mean value $mean(R(x), G(x), B(x))$.

See Fig. 3 for an illustration example.

6. Separation of Rain from Videos

Separating rain from a video sequence seems not less important than its generation. However, we found it is also much easier, at least for the videos of static scenes. Usually, it is enough to perform a median operation on a number of frames to obtain an image clear of rain (Fig. 4), due to the fact that a pixel x is rain-free in most of the frames. In more complicated cases this could be done after stabilizing the video.

Another nice experiment we have made was to compute differences between pairs of consecutive frames in videos of static background. Since the only dynamic thing in these videos is rain, the differences actually provide us with the rain masks, which can be planted in other videos. However, this method is less flexible, since we can't regulate the rain parameters we've described in Sec. 3.

7. Experimental Results

In Fig. 4 and Fig. 5 we show an example of the generated rain (however, it is better to see the videos at *http://www.cs.huji.ac.il/∼werman/Papers/rain/rain.html*). The example consists of two videos - the first one is a video

(a) No darkening (b) Darkening (c) More darkening

Figure 3: Effect of darkening: (a)-original, (b)-darkened and (c)-darkened even more

of a real rain and the second one is an artificial rain video which is adjusted (using our algorithm) to the background of the first one.

First, we separated out the rain strokes from the first video, using the median method described in Sec. 6. We then applied our algorithm to the resulting median image in order to produce simulated rainfall similar to the original rainfall.

The goal of such a comparison was to check our ability to produce rain videos looking as natural as the original ones. For each such example we have defined a set of rainfall parameters (as described in Sec. 3) which seemed to be close enough to the rainfall in the original rain video. By applying our simulation to the rain-free background with those parameters, we aimed to achieve the level of similarity between the original and the artificial movies such that the human perception system could not distinguish between them.

Another example we have made demonstrates the abilities of our algorithm to produce rainfall of various intensities (heavy, light and very weak), with and without darkening effect. Different parameters of raindrops' size, density and speed were used to calibrate the algorithm (see the examples of the generated videos on the web).

More examples of rain generation, rain separation and human perception in videos can be found at *http://www.cs.huji.ac.il/∼werman/Papers/rain/rain.html*.

8. Discussion and Future Work

In this paper we described a system to add artificial rain into video sequences, using simple 3D-world assumptions and a projective camera model. Rather than the complex task of modelling the 3D world, we worked with the video sequences directly. User-defined parameters were used to determine the rainfall intensity and the assumed 3D scene depth, contributing to the model flexibility.

A natural improvement of our model is to implement more realistic assumptions about the rainfall. For example, it was found recently [3] that raindrop size follows an exponential two-parametric distribution, and the velocity of the drops can be calculated as a function of their size and the wind velocity.

More research about human perception is of interest. It seems, for example, that remote drops have only weak influence on perception.

In addition to the rainfall itself there is much work that can be done to simulate other effects of rain, such as reflections, puddles, breaking of the raindrops when they reach solid objects etc. All those, though, require prior information about the 3D scene structure and the light sources.

Acknowledgments

We would like to thank Efim Belman for helpful discussions and technical help in this work and the anonymous referees for their comments.

References

[1] Rain particle system demo. *http://www.gamedev.net/features/diaries/khdiary*, 2002.

[2] Z. Bar-Joseph, R. El-Yaniv, D. Lischinski, and M. Werman. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):120–135, /2001.

[3] A.C. Best. *The Size distribution of Raindrops*, volume 76. Quart.J.Royal Meteor.Soc., 1950.

[4] Doretto, Gianfranco and Chiuso, Alessandro and Wu, Ying Nian and Soatto, Stefano. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003.

(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Median

Figure 4: Three consecutive frames from the original rain video and the median image

(a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4

Figure 5: Four consecutive frames from the generated rain video. Rainfall parameters: maximal scene depth=4, raindrops density=20, raindrops speed=5, tilt=0.

[5] Paul Fearing. Computer modelling of fallen snow. In *Computer Graphics (SIGRAPH 2000 Conference Proceedings)*, pages 37–46, 2000.

[6] Tabuchi Yoshihiko Kusanoto Kensuke, Tadamura Katsumi. A method for rendering realistic rain-fall animation with motion of view. volume 105-005, 2001.

[7] Pierre McComber. Sensitivity of selected freezing rain models to meteorological data. In *Proceedings of the 57th Annual Eastern snow Conference*, Syracuse, New York, USA, 2000.

[8] W. T. Reeves. Particle systems – a technique for modeling a class of fuzzy objects. 2(2):91–108, April 1983.

[9] Y. Wu S. Soatto, G. Doretto. Dynamic textures. volume (in press), 2001.

[10] Mikiya Shniya and Alain Fournier. Stochastic motion-motion under the influence of wind. In *EUROPGRAPHICS*, pages 119–128, Europian Association for Computer Graphics, 1992.

[11] Karl Sims. Particle animation and rendering using data parallel computation. In *Computer Graphics (SIGRAPH 90 Conference Proceedings)*, pages 405–413, 1990.

[12] J.F Straube and E.F.P.Burnett. Simplified prediction of driving rain deposition. In *Proceedings of International Physics Conference*, pages 375–382, Eindhoven, 2000.