Paper:

Behavior Learning and Animation Synthesis of Falling Flat Objects

Kohta Aoki*, Osamu Hasegawa,***, and Hiroshi Nagahashi****

*Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology
**Imaging Science and Engineering Laboratory, Tokyo Institute of Technology
4259 Nagatsuta-cho, Midori-ku, Yokohama 226-8503, Japan
***PRESTO, Japan Science and Technology Corp. (JST)
E-mail: {aoki, hasegawa, longb}@isl.titech.ac.jp
[Received August 11, 2003; accepted December 1, 2003]

In this paper, we describe an approach to learning patterns from sample data sequences and generating new data sequences through learned models. The target application of this work is the animation of natural phenomena, especially falling behavior of flat objects. The natural object or phenomenon to be animated is recorded using one camera, and its characteristic behavior is captured. Feature vectors are defined as the representation of behavior and are automatically extracted from captured videos. By learning the structure of a set of sample vector sequences, the learned model can generate a novel pattern through the underlying structure. These generated patterns could differ from every original vector sequence but preserve characteristics of subject behavior. We can use such patterns to synthesize natural-looking animation.

Keywords: pattern learning, computer animation, natural phenomena, self-organizing map

1. Introduction

The animation of natural phenomena is a challenging task in computer graphics, but a very useful one in many commercial and academic applications. Every natural phenomenon has some characteristic behavior. If such behavior is captured, it can be efficiently used in the animation of natural phenomena.

We propose an approach to the creation of image-based animation of natural objects or phenomena by learning patterns from various sample data that characterizes their behavior or motions. This paper seeks to animate a falling leaf or petal. Several researchers have reported physics- or dynamics-based simulations for the falling behavior of a flat object [**?**, **?**]. Parameter optimization for these simulations and models is generally complicated. We focus on the synthesis of natural-looking animation rather than on the acquisition of such models.

One approach to acquiring motion data is a motion capture system, which tracks markers attached to an object and produces three-dimensional representation of its movements. Many techniques have been proposed to synthesize novel motions and animation by editing obtained motion data [**?**, **?**, **?**]. However, motion capture is seldom applied to natural phenomena, because attached markers disturb natural motion, such as the fluttering of a butterfly's wings.

Our approach uses only behavior data of subject that can be obtained from video recordings. Some leaf- or petal-shaped image objects, or *sprites*, are composed onto appropriate background images through such sample data. Other sample-based techniques for animation synthesis have been developed, for example, speech animation by blending separated facial parts in sample images [**?**], and an approach to synthesizing various streams of images by rearranging original frames from source videos [**?**]. Ezzat *et al.* [**?**] presented a technique for synthesizing *videorealistic* speech animation, however there are some constraints on composing an image sequence of mouth movements into background images. Character animation by controlling sprites [**?**] can create effects similar to our approach. This approach is an extension to frame arrangement [**?**] and therefore should prepare many sprite frames for various animation, while we can synthesize much animation using behavior data, which is more easily obtained.

Machine learning methods that create animation and motions are described in [**?**, **?**], where physics-based models are driven by learned controllers. In our approach, learning involves finding the structure of sample data. The learned model can generate a novel pattern that preserves the underlying structure. These processes of our approach correspond to two types of inference: *induction* and *deduction*. That is, induction is to learn the underlying structure of a behavior data set, and deduction is to generate various patterns through such structure. We exploit the self-organizing map (SOM) learning algorithm [**?**], which is a neural network model. A one-dimensional map may be regarded as a vector sequence or a pattern. In other words, we associate pattern generation with learning by the SOM algorithm. Natural animation can be created by moving sprites according to generated patterns that represent subject behavior.

Our system consists of three phases, as shown in **Fig.1**.

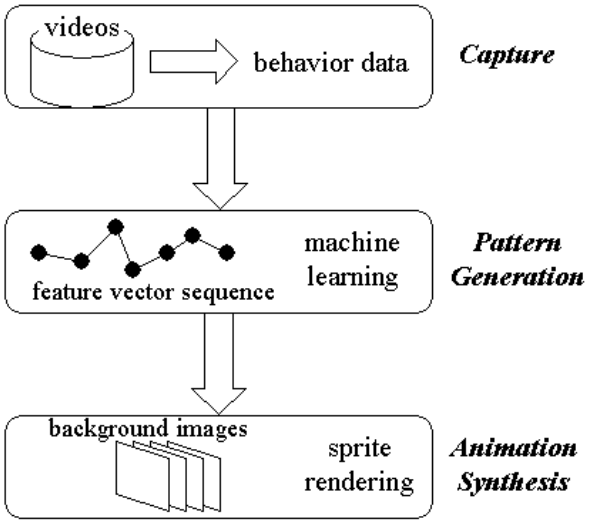


Fig. 1. System overview.

We first capture with one camera various scenes where a leaf is falling, and then track it in these videos. The characteristic behavior of a falling leaf is defined as a temporal sequence of feature vectors (Section 2). An SOM network partly approximates a set of sample vectors in the learning process. The learned network may be equivalent to a novel pattern (Section 3). We use these generated patterns to animate sprites of leaves (Section 4), and photorealistic videos that depict the subject are composed (Section 5).

2. Feature Extraction

First, we must record the subject as accurately as possible, so we used a high-speed camera capable of capturing 125 frames per second, and eventually captured 70 video streams. Each stream is composed of 50-90 frames with a resolution of 480 $\times$ 420 pixels. A target object, i.e., a leaf, is tracked in these videos. Then, the object region is simply segmented from each frame by background subtraction, because the recording was done in a monotone environment. These video recordings are only used to extract some features of subject behavior, so monochrome videos may be more convenient than color ones.

For a segmented object region in each frame, features are calculated to represent characteristic behavior of the subject. Through preliminary experiments, we determine the following appearance-based features:

- angle θ of the central principal axis,
- height L_1 and width L_2 of the minimum rectangle that includes the object region.

Motion-based features are defined as follows:

- dx and dy of movement per pixel between two consecutive frames.

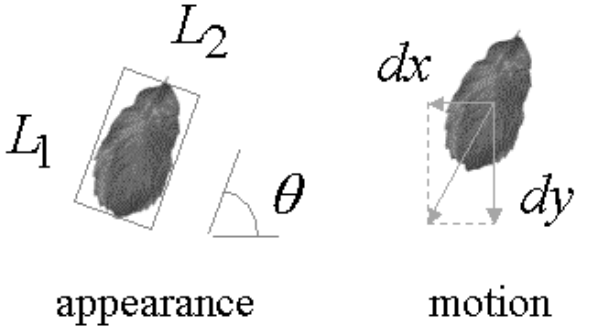


Fig. 2. Features expressing characteristic behavior of a falling leaf.

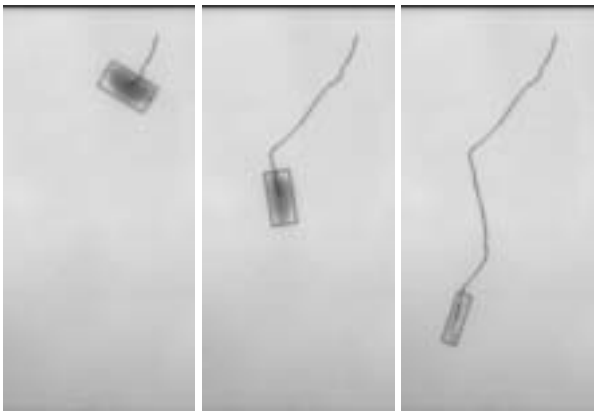


Fig. 3. Automatic tracking and extraction of a falling leaf from a given video stream captured by a high-speed camera.

These five features are simple but fundamental and effective at representing subject behavior, as shown in **Fig.2**. A sequence of feature vectors is automatically calculated from one video stream as follows:

$$[\,\theta(t)\;L_1(t)\;L_2(t)\;dx(t)\;dy(t)\,],\quad (t=1,\ldots,T)\cdot\cdot \quad (1)$$

where T is the length of a sequence.

Thus, our approach characterizes the falling behavior of a leaf using only the data obtained from two-dimensional image sequences. The obtained data, or feature vector sequences, is a primitive representation, but we can create sprite animation of a falling leaf. **Figure 3** shows an example of automatic tracking and extraction of a falling leaf from the given video stream captured by the high-speed camera.

3. Learning and Pattern Generation

One common technique for learning sequential patterns is a simple recurrent network or *Elman Net* [**?**]. The network can develop internal representations for some input patterns, and produce the correct output for a given input.

Hidden Markov models (HMMs) are often used to statistically model a variety of sequential data [**?**]. Novel patterns generated using these techniques could possess almost the same attributes as existing ones, but conversely, the pattern diversity may be low.

We therefore exploit the self-organizing map (SOM) learning algorithm [**?**], which is a neural network model. In our approach, pattern generation is more closely associated with machine learning. This enables us to obtain various patterns from given samples, and according to the learning rule, each generated pattern is reasonable in that it preserves sample characteristics. In this section, we describe the SOM algorithm, and then propose a new technique for pattern generation.

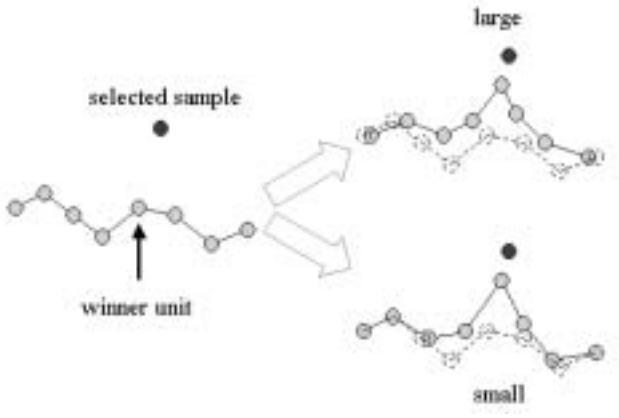


Fig. 4. Different learning processes due to the value of neighborhood radius σ_t: if distance d_{ci} is equal, the updating influence is determined by the value of σ_t.

3.1. Learning

The SOM is an unsupervised neural network, which means that there is no external teacher who controls the learning process. An SOM network is a set of neurons organized usually on a regular low-dimensional grid. Each neuron in a network is represented by a weight vector, or a reference vector, $m = [m_1, \ldots, m_d]$, where d is equal to the dimension of training sample vectors. These neurons are connected to neighboring neurons by a certain relation, which defines the topology, or structure, of a map. In our approach, these neurons are positioned on a one-dimensional map such as a line shape, and hence a network can be regarded as a sequence of weight vectors or a pattern.

The SOM learning algorithm is similar to vector quantization (VQ), such as a *k*-means. The SOM algorithm differs from VQ in that, along with the closest weight vector to a presented sample, its topological neighbors on the map are updated. This leads to the maintenance of continuity of a vector sequence or a network.

Each weight vector of a neuron is randomly initialized before training. In each training step, one sample vector x is randomly selected from an input data set. The neuron whose weight vector is the shortest distance from a selected sample is determined by the equation

$$c = \arg\min_i ||x - m_i|| \qquad \ldots\ldots\ldots\ldots \quad (2)$$

where $||\cdot||$ denotes the Euclidean distance. This neuron is called *winner neuron c*.

The weight vector of a winner neuron is updated so that it moves toward a presented vector in the vector space, and neighboring neurons of the winner are treated similarly. The weight vector of neuron i is updated according to

$$m_i(t+1) = m_i(t) + \alpha(t) h_{ci}(t) \{x(t) - m_i(t)\} \quad . \quad (3)$$

where t denotes a time step, and $x(t)$ is the selected vector at each time t. There are two learning parameters: learning rate $\alpha(t)$ and neighborhood function $h_{ci}(t)$ around the winner neuron c.

The learning rate is an exponential function that monotonically decreases with time, and controls the gradual decay of learning. The neighborhood function is defined as

$$h_{ci}(t) = \exp\left(-\frac{d_{ci}^2}{2\sigma_t^2}\right) \qquad \ldots\ldots\ldots \quad (4)$$

where σ_t is the neighborhood radius at step t, and d_{ci} is the topological distance between two neurons c and i on the map grid. If the distance d_{ci} is large (i.e., if neuron i is far from the winner c), the updating influence is small, because the value of $h_{ci}(t)$ in Eq. **??** becomes small. The function $h_{ci}(t)$ depends on the value of σ_t if the value of d_{ci} is constant. In this sense, the radius σ_t defines the restrictive scope of updating.

The training described above is repeated until a predetermined number of steps is reached. The results of learning could be more greatly influenced by two parameters than by the state of an initial network and the randomness of selected sample vectors.

3.2. Pattern Generation

We associate pattern generation with learning by the SOM algorithm. In a feature space that consists of more than one vector sequence, homogeneous subpatterns are distributed, while mutually different distributions may also be observed. Our approach can generate various patterns from such feature space even if the same set of vector sequences is given. Learning involves finding the structure of sample vectors, and the structure is varied due to two learning parameters. Pattern generation, or learning, is performed specifying these parameters. In other words, this is the extraction of patterns from the parameterized structure.

Changing learning parameters, e.g., neighborhood radius σ_t, results in different learning processes, as shown in **Fig.4**. The weight vector of each neuron moves closer to a selected sample vector in a feature space if a larger neighborhood radius is given, because the value of $h_{ci}(t)$ (Eq. **??**) becomes relatively large and as a consequence, the updating influence is large. When the neighborhood

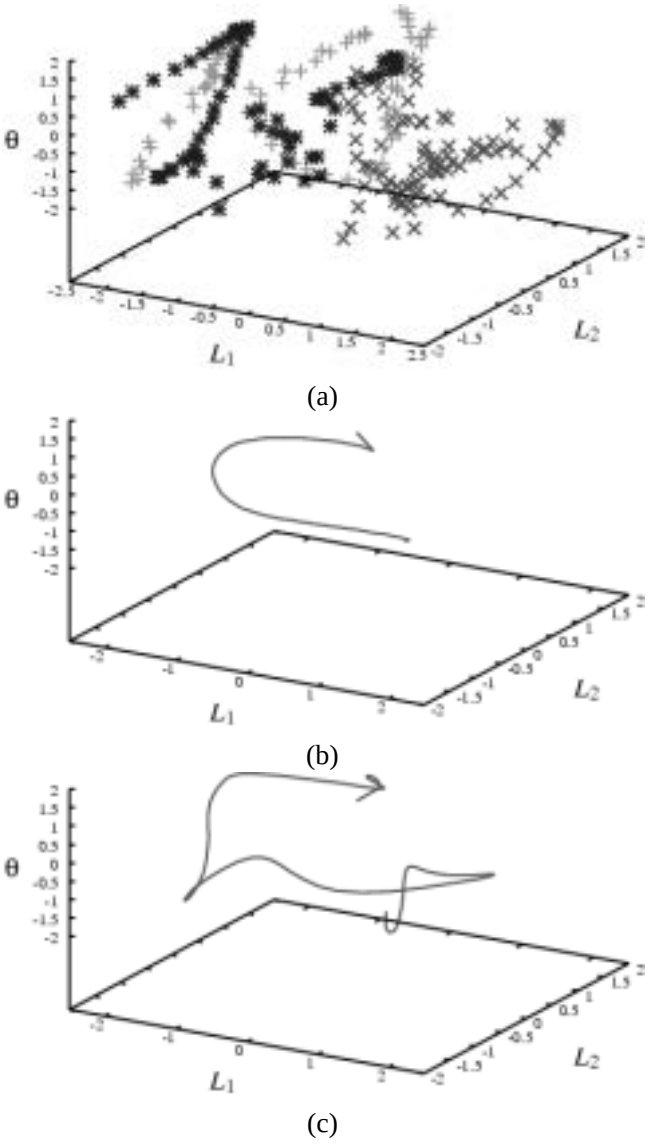


Fig. 5. Two networks trained by different neighborhood radii: (a) three sequences of feature vectors used as an input data set, (b) a network given a relatively large neighborhood radius, (c) a network given a small radius.

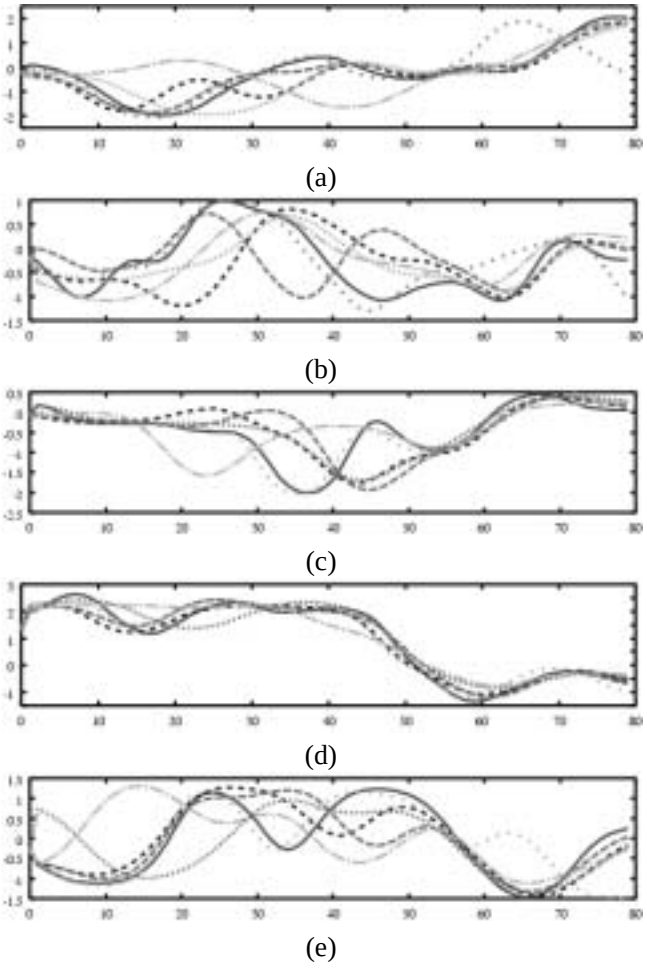


Fig. 6. Patterns generated from samples in **Fig.5** (a) changing the neighborhood radius. The feature shown in each figure is as follows: (a) dx, (b) dy, (c) θ, (d) L_1, (e) L_2. The vertical axis denotes normalized values.

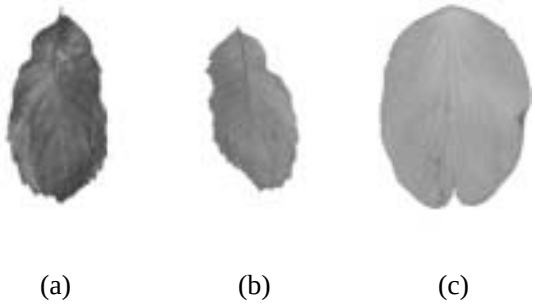
radius is small, updated weight vectors may move away from vectors that are not updated. In short, a network represented by such weight vectors spreads across feature space.

Figure 5 shows two results of learning given three types of feature vector sequences (a): two networks trained specifying the large and small neighborhood radius are shown in **Fig.5** (b) and (c). Note that a feature space is depicted with only three of five parameters in these figures: angle θ, height L_1, and width L_2. The former learned network can be seen as an average pattern found among sample vectors, and the latter network expresses a pattern that partly approximates each vector sequence. Both generated patterns are reasonable because they are thought to be derived from the underlying structure of samples.

Shown in **Fig.6** are examples of patterns generated from samples in **Fig.5** (a). These patterns vary widely, while they seem to have the same form. This indicates that generated patterns preserve the structure of sample data.

4. Animation Synthesis

In this work, creating animation requires sprites, which are image objects moving independently of a background. **Figure 7** shows examples of sprites that depict both sides of a leaf and a cherry blossom petal. In order to animate sprites, we use patterns generated by learning sample data sequences. One sprite is resized and rotated according to the appearance-based features of a pattern, and is moved on natural background images by the motion-based features. Because generated patterns represent the characteristics of the subject behavior, they can be used to create natural animation (**Fig.8**).



(a) (b) (c)

Fig. 7. Sprites: (a) Upper side of leaf, (b) Lower side of leaf, (c) Cherry blossom petal.

Fig. 8. Animation of a falling leaf duplicated by a pattern.

4.1. Superimposition

Sprite animation is superimposed on background images so a resulting image sequence is photorealistic. The basic idea of our approach is that an object could look smaller and slower if we see it from a greater distance. We adopt layered depth representation to reflect our sense of vision. That is, layers with relative coefficients are manually specified against background images. Each sprite is arbitrarily located on one of the layers and is animated according to a sequence of feature vectors that are multiplied by the layer's coefficients. In other words, using small coefficients, a sprite on a deeper layer is made smaller and moved more slowly. This leads to the synthesis of natural-looking animation.

5. Results

We conducted animation synthesis and the creation of videos: one scene where many autumnal leaves flutter down on a tree-lined street, and the other where numerous cherry blossom petals fall.

Falling leaves: A falling leaf was recorded 70 times, or in other words, 70 different data sequences of its behavior were obtained. The falling behavior of a leaf was not affected by wind because the video recording was done indoors. We randomly select several sequences from the set of obtained data and use them to train an SOM network. Similarly, other sequences selected from the data set are presented to another network. Thus, we train many networks by the SOM learning algorithm and generate more patterns than samples.

Figure 9 shows an example of created videos, which depicts many falling leaves. We prepare two sprites that represent the front and reverse side of a leaf. If the transverse motion of an object (i.e., the sign of dx) changes, these sprites are replaced. We assume five layers against the background video footage, and scatter many leaf-shaped sprites over each layer. Additionally, we delay the timing of the appearance of each sprite. These lead to varied, more natural-looking animation.

Falling cherry blossom petals: A leaf and a petal have the same shape, i.e., flat. A petal, however, may be smaller and lighter than a leaf and by this difference, should flutter down more slowly or fall more quickly if it is blown by wind. The animation of sprites, including such wind effects, is controlled by adjusting generated patterns, which is similar to the approach mentioned in Section 4.1. That is, the speed of sprites is adjusted by multiplying patterns by appropriate coefficients. Therefore, the same data set as obtained above can be reused to generate novel patterns and to animate a petal-shaped sprite. Photorealistic videos that depict falling petals are created using the same procedure as above (**Fig.10**).

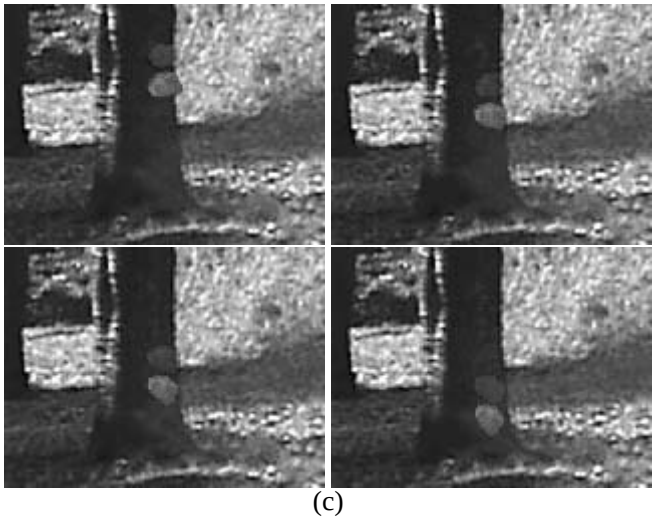
6. Conclusion and Future Work

In this paper, we proposed an approach to learning and generating patterns from data sequences that represent the subject behavior. Generated patterns are used to synthesize animation. A natural phenomenon or object is recorded using a video camera, and its behavior is represented as a sequence of feature vectors. Our method can generate various patterns from such vector sequences by exploiting the self-organizing map algorithm. The generated patterns preserve the underlying structure of a sample data set, so natural-looking animation is synthesized through these patterns.

As described above, the subject behavior, i.e., a falling leaf in this work, was not affected by wind. However, we can add wind effects to patterns generated by our method and synthesize the animation of leaves fluttering in the

(a)

(b)

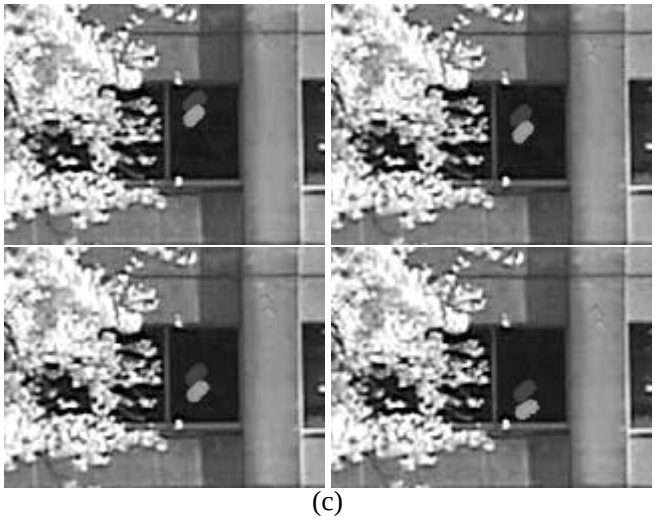


(c)

Fig. 9. Scene where many leaves are falling to the street: (a) Frame in created video. (b) Enlarged part. (c) Consecutive frames showing a falling leaf.

(a)

(b)



(c)

Fig. 10. Scene where numerous Japanese cherry blossom petals flutter: (a) Frame in created video. (b) Enlarged part. (c) Consecutive frames showing a falling petal.

wind. These patterns are equivalent to sequences of feature vectors, and therefore the vectors of wind effects can be linearly added to them: for example, we use vectors affecting transverse motion as wind in the x direction. Because of vector representation, by applying geometric transformation such as an affine, sprites can be superimposed on an image sequence taken from a different viewpoint. We hope to implement these additional operations in future work.

The effectiveness of our approach depends on whether adequate feature vectors of subject behavior can be extracted from video recordings. We successfully animated numerous falling leaves and petals using this approach. Our approach has potential applications for the animation of other subjects that have more complex behavior or bodies, such as facial expressions, human actions, and floating of a butterfly.

The behavior of a subject may be difficult to express using only two-dimensional data. It is possible to combine three-dimensional information with vectorized features of behavior, and more advanced spatial expressions must be applied to the synthesis of natural-looking animation. Moreover, learning algorithms for vector quantization, such as the SOM algorithm, could be improved for such high-dimensional vector space.

References:

[1] Takayuki Aoki, "3d simulation for falling papers", Computer Physics Communication, 142:326–329, 2001.
[2] Yashuo Bengio, "Markovian models for sequential data", Neural Computing Surveys, 2:129–162, 1999.
[3] Eric Cosatto and Hans Peter Graf, "Sample-based synthesis of photo-realistic talking heads", In Proceedings of Computer Animation '98, pages 103–110, 1998.
[4] Jeffrey L Elman, "Finding structure in time", Cognitive Science, 14:179–211, 1990.
[5] Tony Ezzat, Gadi Geiger, and Tomaso Poggio, "Trainable videorealistic speech animation", In Proceedings of ACM SIGGRAPH 2002, pages 388–398, 2002.
[6] Petros Faloutsos, Michiel van de Panne, and Demetri Terzopoulos, "Composable controllers for physics-based character animation", In Proceedings of ACM SIGGRAPH 2001, pages 251–260, 2001.
[7] Radek Grzeszczuk, Demetri Terzopoulos, and Geoffrey Hinton, "Neuroanimator: Fast neural network emulation and control of physics-based models", In Proceedings of ACM SIGGRAPH 1998, pages 9–20, 1998.
[8] Teuvo Kohonen, "Self-Organizing Maps", Springer, Berlin, Heidelberg, 1995.
[9] Lucas Kovar, Michael Gleicher, and Frédéric Pighin, "Motion graphs", In Proceedings of ACM SIGGRAPH 2002, pages 473–482, 2002.
[10] Yan Li, Tianshu Wang, and Heung-Yeung Shum, "Motion texture: A two-level statistical model for character motion synthesis", In Proceedings of ACM SIGGRAPH 2002, pages 465–472, 2002.
[11] Arno Schödl and Irfan Essa, "Controlled animation of video sprites", In Proceedings of the First ACM SIGGRAPH Symposium on Computer Animation, pages 121–128, 2002.
[12] Arno Schödl, Richard Szeliski, David H Salesin, and Irfan Essa, "Video textures", In Proceedings of ACM SIGGRAPH 2000, pages 489–498, 2000.
[13] Yoshihiro Tanabe and Kunihiko Kaneko, "Behavior of a falling paper", Physical Review Letter, 73 (10):1372–1375, 1994.
[14] Andrew Witkin and Zoran Popović, "Motion warping", In Proceedings of ACM SIGGRAPH 1995, pages 105–108, 1995.

Name:
Kohta Aoki

Affiliation:
Doctoral Student, Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology

Address:
4259 Nagatsuta-cho, Midori-ku, Yokohama 226-8503, JAPAN

Brief Biographical History:
2003- M.E. degree from Tokyo Institute of Technology
2003- Doctoral Student, T.I.Tech.

Main Works:
- "Animation Synthesis by Observation and Learning," IEEE International Symposium on Computational Intelligence in Robotics and Automation (IEEE CIRA2003), pp. 1258–1263, Jan. (2003).

Membership in Learned Societies:
- The Institute of Electronics, Information and Communication Engineers

Name:
Osamu Hasegawa

Affiliation:
Associate Professor, Imaging Science and Engineering Lab, Tokyo Institute of Technology

Address:
4259 Nagatsuta-cho, Midori-ku, Yokohama 226-8503, Japan

Brief Biographical History:
1993- Dr. Eng. from Univ. of Tokyo. Joined Electrotechnical Lab.
1999- Visiting Scientist, Carnegie Mellon Univ.
2002- Associate Professor, TITech.

Main Works:
- "A Vision System that Recognizes Object on General Streets," Proc. of The Eighth IAPR Workshop on Machine Vision Applications (MVA02), pp. 618-623, (2002)
- "A Kernel Logit Approach for Face and Non-face Classification," Journal of Japanese Academy of Facial Studies, vol. 3, no. 3, pp. 119-126, (2003)
- "A method for monitoring activities of multiple objects by using stochastic model," IEICE Trans. on Information and Systems, Vol. 84-D No. 12, pp. 1705-1712, (2001)

Membership in Learned Societies:
- The Institute of Electrical and Electronics Engineers
- Information Processing Society of Japan
- The Institute of Electronics, Information and Communication Engineers

Name:
Hiroshi Nagahashi

Affiliation:
Professor, Imaging Science and Engineering Lab, Tokyo Institute of Technology

Address:
4259 Nagatsuta-cho, Midori-ku, Yokohama 226-8503, Japan

Brief Biographical History:
1981- Faculty of Engineering, Yamagata University
1990- Imaging Science and Engineering Lab., Tokyo Institute of Technology

Main Works:
• "Parallel Computation of Parametric Piecewise Modeling Method," IEICE Trans. on Information & Systems, Vol. E85-D, 2, pp. 411-417, (2002).

Membership in Learned Societies:
• The Institute of Electronics, Information and Communication Engineers (IEICE)
• The Institute of Electrical and Electronics Engineers (IEEE)
• The Institute of Image Information and Television Engineers