A Simple Method to Animate Vegetation in Images Using Simulation-guided Grid-based Warping

Kan Chen[1,2], Henry Johan[2]
[1]*Nanyang Technological University, Singapore*
[2]*Fraunhofer IDM@NTU, Singapore*
Email: kchen1@ntu.edu.sg, hjohan@fraunhofer.sg

***Abstract*—In this paper, we propose a simple method to animate vegetation in images. The proposed method enables users to interactively animate vegetation in real-time. It can be used to enhance a vegetation photograph or painting with animated motions as well as to help animators to create cartoon animation of vegetation. Our method is suitable for simulating vegetation-wind interaction for a dense vegetation field which exhibit non-rigid property, especially in cartoon style images. In our method, wind field is modeled using fluid simulation. We adopt a harmonic oscillator based on wave simulation as the numerical model for the dynamics of vegetation. The velocity generated from the wind field is used to drive the wave simulation to determine the velocity of vegetation. Based on the velocity of vegetation, we employ a grid-based warping approach to animate different types of vegetation such as grass and trees. As a result, for vegetation in images, we can simulate grass and trees swaying in wind as well as under water plants swaying in water. Our method can produce natural-looking output motions comparable with real cartoons. Users can interactively control the location of the source, direction and strength of the wind or water flow and the property of vegetation on the fly.**

***Keywords*-2D animation; vegetation; fluid simulation; wave simulation; image wrapping**

I. Introduction

When we are looking at an image, such as a photograph or a painting, usually the existence of dynamic or moving objects can be perceived, although the image is indeed static. A painting looks vivid because we may feel the presence or hints of motion in it. For instance, if we look at a picture of a tree, we may think of it swaying in the wind. This motion effect helps us to better appreciate a photograph or a painting. On another note, nature scenes (especially vegetation scenes) exist in many cartoon animations. However, they still lack movement in some cases. For example, in the cartoon "Twins on the Pasture" (Figure 1), the flags are waving in wind, but the foreground grass remains still. Viewers may feel uncomfortable with the absence of motion. Naive approach by manually drawing animation frames and interpolating them can produce vegetation animation, but it requires tedious effort. Therefore, a method which can be used to animate vegetation in images is required. This method is helpful in enhancing the perception of photographs or paintings, creating wind visual effects in movies and assisting animators to create cartoon animations of vegetation.

Vegetation has very abundant species and has different properties. We observe that the movements of many types of vegetation (especially in cartoons) exhibit non-rigid property, such as tree crown, dense vegetation field, under water plant, etc. In their movements, the vegetation's shape is perceived as deforming and oscillating, effects such as wave effect may also exists. For example, in Figure 9, on the left, when the grass is swaying in wind, its shape is deforming, on the right, the tree crown moves like a continuum. However, existing methods of animating vegetation in images ([1], [2]) mainly handle only tree type vegetation with branches whose motions usually exhibit rigid property. Existing methods mainly focus on modeling single elements such as grass blades, leaves and branches, instead of addressing explicitly the overall appearance of the movements of a vegetation field, which we believe can be handled in a simpler and more efficient manner.

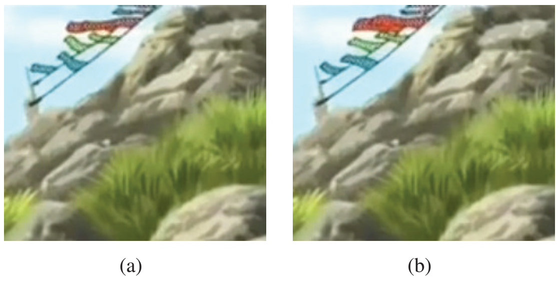
(a) (b)

Figure 1. Movie frames from animation "Twins on the Pasture" (©CCTV Animation Inc.). Wind is blowing, but the vegetation in foreground remains still.

Given a single image with vegetation, which can be a hand-drawn image, painting, cartoon or real photograph, our target is to animate the vegetation in the image. In general, for many applications as above, the realization of the effects of vegetation is more important than the physical accuracy. Thus, instead of physics simulation we proposed a novel method that combines a simple dynamics simulation and a procedural method to model the appearance of non-rigid movement of vegetation. As a result, using our method, users can interactively animate various types of vegetation under different wind condition, ranging from vibrates in breeze to swaying in strong wind. Our method consists of the following features:

1) We consider the vegetation field as one continuum. The wind field generated using fluid simulation, is used to drive the dynamics of the continuum.
2) Our method heuristically models the appearance of vegetation's motion. Many natural motions for vegetation such as oscillating in breeze or swaying in wind can be viewed as harmonic oscillation [2], [3]. We adopt a wave simulation as the numerical model for the dynamics of vegetation.
3) We employ a grid-based image warping method to animate vegetation in images. The amount of warping is determined by the velocity of vegetation. Our method also consider the properties of vegetation such as the rigidity of the vegetation.
4) Our method is simple and efficient, users can adjust the results to achieve their desired animation effects on the fly.

II. Related Work

A. Animation of vegetation in 3D scenes

Animation of grass: Perbet and Cani [4] proposed a procedural animation controlled by wind primitives. This work was extended to simulate the interaction between a vegetation and a moving object [5]. Wang et al. [6] combined free form deformation and explicit grass-grass collision detections. Banisch and Wuthric [7] used a mass-spring system where the masses and springs are attached to the shell layers. Habel et al. [8] achieved a texture based animation by distorting the texture lookups. Orthmann et al. [9] proposed an GPU-based approach to model responsive grass using a cloth model based on spring constraints. The conventional way to generate wind effects is by using the combination of sin and cos wave plus noise as in Pelzer [10] and Zioma [11]. Ramraj [12] simulated the wind effects using a simple water wave simulation. Chen and Johan [13] proposed to consider grass field as a two dimensional grid-based continuum and shift the complex interactions to the dynamics of continuum.

Animation of trees: Physics-based methods to animate trees can achieve convincing results. Stam [14] proposed to carry out modal analysis to animate a tree. The simulation is in the frequency domain and spectral method is used to reduce the computational cost. Chuang et al. [2] also used a spectral method which includes the motion spectrum of a damped harmonic oscillator to model the animation of plants. Habel et al. [3] proposed a physics guided animation using beam deformation and a similar spectral method as [2]. Diener et al. [15] proposed to compute a wind basis using modal analysis and project directional wind at run time. Moreover, Weber [16] presented a fast simulation method that allows users to interact with trees using a cloth dynamics model. In general, physics-based methods requires high computational cost and they lack intuitive direct control. Procedural method usually is based on noise functions to heuristically model the appearance of tree's motion instead of performing a simulation, for example, Ota et al. [17], Sousa [18] and Hu et al. [19], [20] used noise functions to drive the animation. Zhang et al. [21], [22] used pre-recorded motion graphs to create animation for trees.

Existing methods to animate vegetation in 3D scenes tend to consider every single elements such as grass blades and tree branches and they usually focus on one single type of vegetation either trees or grass. Most recently, Zhao and Barbič [23] proposed to pre-process plant geometry to a format suitable for physically based simulation, this method can be applied to animate many types of vegetation. Moreover, these methods usually require vegetation's 3D structure information, thus they cannot be directly applied to animate vegetation in 2D images.

B. Animation of vegetation in 2D images

Animating 2D still images: Shinya et al. [1] proposed to couple physically-based simulation with skeleton-based morphing techniques to animate plants in 2D images. [2] proposed to segment an image into layers and synthesize a motion texture to animate the still image layer by layer. However, these methods [1], [2] are based on defining skeletons or line segments as animation model, they are more suitable for tree type vegetation with branches, whose motions usually exhibit rigid property. Xu et .al [24] proposed a method to animate animals's motion in 2D images by finding the motion path and apply morphing technique. Schödl et al. introduced video texture that allows the creation of repetitive and endless video given a short video sequence [25]. Lin et al. [26] presented a system that generates high resolution animated videos from a small number of still images via the graph and Markov Chain model. These methods requires sufficient number of pairs of similar frames or images.

Example-based texture synthesis has been widely investigated, it can be applied to synthesize dynamic texture such as video texture and motion texture (refer to the survey in [27]). Wei et al. [28] proposed a pixel-based texture synthesis method to synthesize spatio-temporal textures such as fire, smoke and ocean waves. Kwatra et al. [29] proposed a patch-based texture synthesis method based on graph-cut to synthesize texture. Kwatra et al. [30] proposed an optimization-based texture synthesis technique guided by flow to synthesize animated texture. Ma et al. [31] combined example-based texture synthesis method and fluid simulation to synthesize motion texture to animate under water plant. Chen et al. [32] proposed a method to design a 2D time-varying vector fields which can be applied to animate still images.

Simulation in cartoons: There are very little works in simulating plants in cartoons. Fiore et al. proposed to build realistic 3D geometries of tree as underlying models [33] to animate cartoon style tree. Haevre et al. applied their method of animating 3D trees [34] to animate cartoon style trees. However, these methods depends on the explicit 3D models of trees.

Hair animation in cartoons has some similarities with vegetation animation in cartoons. Eiji et al. proposed simulation [35] and data-driven [36] approaches. Noble et al. [37] used NURBS surfaces to model and animate cartoon hair. For other effects in cartoons, Liao et al. proposed procedural methods to simulate water [38] and cracking [39] effects.

In general, creating wind effect in cartoons requires a certain amount of art skills and tedious manual work [35], thus in most existing cartoon animations, the vegetation either remains still or moves periodically. We perform a stable fluid simulation based on Navier-Stokes equations to generate natural wind field [40], so that no tedious art works are required to create visually plausible wind effect.

III. Our method

In this section, we propose our method to animate vegetation images. We begin with an overview that describes the basic idea of our method (Section III-A). We then address the important sub-problems, namely the segmentation and compositing (Section III-B), wind model (Section III-C), simulating dynamics of vegetation (Section III-D), and animating vegetation in images (Section III-E).

A. Basic idea

We want to generate the following effects efficiently: (1) shrub oscillating in breeze, tree crown swaying in wind, under-water plant swaying in water and wave effect of grass field, (2) when wind passing through a vegetation field, the plants closer to the wind source move first. The vegetation-vegetation interaction occurs. The movement is then propagated to other plants that are away from the wind source. To achieve the above two effects, we propose to combine fluid simulation, wave simulation and image warping methods. The wind field is modeled using a velocity field. We perform a fluid simulation to generate a natural wind field, the solver is based on stable fluid [40]. From this velocity field, we retrieve the wind's velocity vector ($\vec{V}$:(V_x, V_y), Section III-C) to compute the dynamics of the vegetation ($\vec{V_p}$:(V_{p_x}, V_{p_y}), Section III-D).

We adopt the wave simulation as the numerical model for the dynamics of the vegetation. The velocity $\vec{V}$ (Section III-C) from the wind field is used to initiate the amplitudes of the wave which represents the velocity of the vegetation $\vec{V_p}$ (Section III-D). Furthermore, those effects of vegetation field (swaying, oscillating) have similarities with the behaviors of wave. The initial amplitudes of wave are set at some locations. The amplitudes start oscillating and propagating to nearby locations. Thus, we adopt the wave model and mimic those essential effects of vegetation field using the wave simulation.

The animation of the vegetation is based on grid-based image warping and the amount of warping is computed based on the vegetation's velocity $\vec{V_p}$. Our proposed method is suitable for animating dense vegetation scene in images (especially for cartoon style images), such as grass field, crown of tree and group of bushes whose individual blades or leaves are dense and staggered together. Users can control the source, direction and strength of the wind.

B. Segmentation and compositing

We can use existing methods such as [41] to separate the vegetation to be animated (namely vegetation layers) from other parts of the input image, namely background. Another method is to segment the input image using magic wand tool of ©Adobe ©Photoshop [42]. Then we perform image inpainting technique [43] to fill the empty area in the background. For every animated vegetation layer, image matting method [42] is employed to composite the vegetation layer with the background. In cartoon animation pipeline, animation frames usually modeled in layers [44]. In this case, we can integrate our method into this pipeline by applying it on the vegetation layers.

C. Wind model

We implement a 2D fluid simulator to model wind. After one simulation time step, we perform vegetation animation and render one animation frame. The solver [40] solves the Navier-Stokes equations:

$$\nabla \cdot \vec{V} = 0, \quad (1)$$

$$\frac{\partial \vec{V}}{\partial t} + (\vec{V} \cdot \nabla)\vec{V} = -\frac{1}{\rho}\nabla p + \vec{f}, \quad (2)$$

where $\vec{V}$ is the velocity, p is the pressure, $\vec{f}$ is the external force, ρ is the density and t is the simulation time. Users can initiate wind by adding external forces to desired areas using a mouse. Users can control the frequency, direction and strength of the external forces. Wind frequency is controlled by specifying the time interval of adding external forces during simulation. Users also can add external forces at any simulation time. External forces add velocities into desired areas. The velocities' directions and magnitude are decided via setting parameters by the users. The velocities' magnitude is set between 0 to 1, which corresponds to the minimum and maximum strength of external forces. Then the velocities are updated by the external forces as determined by the 2D fluid simulator (Figure 2). The stable fluid solver ([40]) ensures the magnitude of velocities is bounded by 1. We impose an open boundary condition, which means the velocity field is non-reflective at boundaries. This generated wind field is used to determine the vegetation's dynamics and guide the vegetation animation.

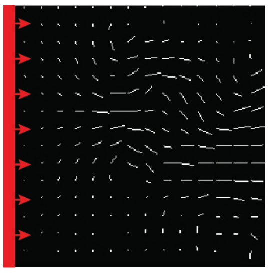

Figure 2. Wind field. The wind source is at the left boundary, and the wind direction is from left to right.

D. Simulating dynamics of vegetation

The velocity field generated using the fluid simulation, in general lacks oscillation effects which mimic the swaying motion of vegetation. Thus it is not suitable to directly applied the result of fluid simulation as the dynamics of vegetation. We consider a vegetation layer as one continuum represented using a 2D grid, and assume the vegetation parts (such as leaves, blades) are evenly distributed in the continuum. We adopt the wave simulation to model the dynamics of continuum, and these vegetation parts are the simulation elements. We consider each simulation element's velocity $\vec{V_p}$ as the wave amplitude and apply the wave equation and then verlet integration to compute the simulation element's new velocity $\vec{V_p}$. The wave amplitude ($\vec{V_p}$) is initiated by the wind velocity ($\vec{V}$) which is obtained using bilinear sampling from the velocity field generated using the fluid simulator in Section III-C. The wave functions we used are the same as in [45] and [13]. For each simulation time t, $a_{V_p}(t)$ is the accelerations of the wave propagation:

$$\begin{aligned} a_{V_p}(t) &= \frac{\partial^2 \vec{V_p}}{\partial t^2} = c^2 (\frac{\partial^2 \vec{V_p}}{\partial x^2} + \frac{\partial^2 \vec{V_p}}{\partial y^2}), \qquad (3) \\ \vec{V_p}(t+\Delta t) &= (1+\gamma)\vec{V_p}(t) - \gamma \vec{V_p}(t-\Delta t) \\ &+ a_{V_p}(t)(\Delta t^2), \qquad (4) \end{aligned}$$

By using the wave simulation, we provide a convenient way to control the vegetation animation. We can tune the wave damping factor, which reflects the softness of vegetation, by setting the γ (smaller γ: softer, greater γ: more rigid) and control how fast a vegetation propagates the wind effect by setting the wave speed factor c (smaller c: slow propagation, greater c: fast propagation). In our experiments, γ is in the range of [0.01, 1] and c is in the range of [1, 10].

E. Animating vegetation in images

To generate the motion of vegetation, we define uniform grids over a vegetation layer and apply inverse warping to deform it. The grid's size s_x, s_y is defined based on the size of parts of vegetation, for example for vegetation with smaller leaves/blades, the size of grid is smaller; as well as the rigidity of the vegetation, for example for less rigid vegetation, the size of grid is smaller. The position of the four corners (x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3) of each grid cell over original vegetation layer is displaced (Figure 3). At each frame, we compute the amount of displacement $\vec{n}$ by Equation 5 using the vegetation's velocity $\vec{V_p}$ which is obtained by bilinear sampling from the wave simulation results in Section III-D.

A vegetation layer may have a perspective viewing angle, for example the vegetation image may be captured from top or side, thus we want to control the amount of deformation at a certain direction, for example, if grass is viewed from the side, it is unlikely for it to move up and down. We also want to control the amount of deformation at a certain region, for example, the vegetation far away should have less deformation compared to the near one. Thus we introduce a scale factor $\vec{\lambda}$ to control the amount of displacement $\vec{n}$. All of $\vec{V_p}$, $\vec{\lambda}$ and $\vec{n}$ consist horizontal (x) and vertical (y) components:

$$\begin{aligned} n_x &= \lambda_x R V_{p_x}(t)\Delta t, \qquad (5) \\ n_y &= \lambda_y R V_{p_y}(t)\Delta t, \end{aligned}$$

$$R = \begin{cases} 1 & \text{if every parts can move,} \\ (\frac{l}{h})^r & \text{otherwise,} \end{cases} \qquad (6)$$

where Δt is the simulation time step, $\vec{\lambda}$ is set with respect to the grid cell's size, l is the distance to the horizon position of vegetation layer, h is the height of the vegetation layer, and $r \in [0, 10]$ is the rigidity parameter that controls the amount of displacement with respect to the distance to the vegetation's horizon, whose position is fixed and determined by the users.

If the vegetation's root can be seen, the horizon position is set as the root part of the vegetation, which is usually the bottom of the vegetation layer (Figure 3). Greater r means the vegetation is more constrained by the root position so it exhibits less movement and is more rigid. If the vegetation's root cannot be seen, for example, vegetation field is viewed from top with a perspective angle, the horizon position is set at the top of the vegetation layer (Figure 4). As such, the vegetation further to the viewer exhibit much less movement than the vegetation closer to the viewer. h and l are computed by finding the distance to the bottom (Figure 3) or top (Figure 4).

Since our method can animate vegetation in real-time, the user can change the values of parameters on the fly to adjust the results. In the case that the vegetation field is very large and not homogeneous, we can segment it into different layers and animate each layer separately. However, this requires extra work for segmentation, thus we propose a simpler control by defining a scale map (Figure 4). The scale map stores the magnitude of scale factor ($\vec{\lambda}$) for the vegetation layer, for example, the magnitude of $\vec{\lambda}$ of the vegetation region further to the viewer is smaller than the vegetation region closer to the viewer, which means the vegetation further to the viewer exhibit less movement. We directly set the magnitude of $\vec{\lambda}$ in Equations 5 and 6 based on the scale map in order to refine the velocity of the vegetation.

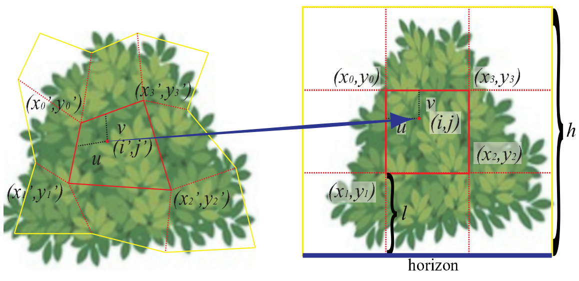

Figure 3. Grid-based warping. Left: Original vegetation layer with warped 2D grids. Right: Warped vegetation layer with 2D grids. Vegetation image is from www.colourbox.com (©Colourbox Educational).

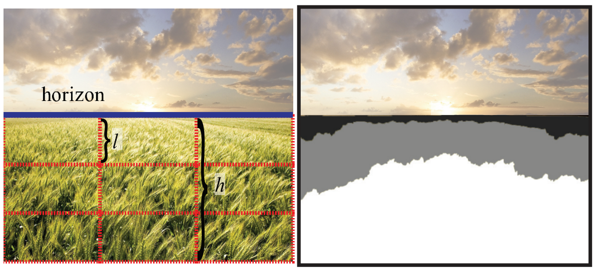

Figure 4. Example of scale map. (a) Original vegetation scene. (b) Scale map, darker region means smaller magnitude for scale factor $\vec{\lambda}$. Vegetation image is from www.iwallscreen.com.

Using displacement $\vec{n}$, the four corners of grid cells in original vegetation layer are displaced in the opposite direction of the velocity, since our warping method is based on inverse warping. The new positions of the four corners after displacement (x_0', y_0'), (x_1', y_1'), (x_2', y_2'), (x_3', y_3') (Figure 3) are computed as follows.

$$(x', y') = (x, y) + (-n_x s_x, -n_y s_y), \tag{7}$$

For point (i, j) in the warped vegetation layer, we compute its corresponding point (i', j') in the original vegetation layer using bilinear interpolation based on its coordinates in grid cell (u, v):

$$\begin{aligned}(i', j') &= (1-v)\left((1-u)(x_0', y_0') + u(x_0', y_0')\right) \\ &+ v\left((1-u)(x_1', y_1') + u(x_2', y_2')\right).\end{aligned} \tag{8}$$

We sample the value at (i', j') at the original vegetation layer as the value of point (i, j) (Figure 3).

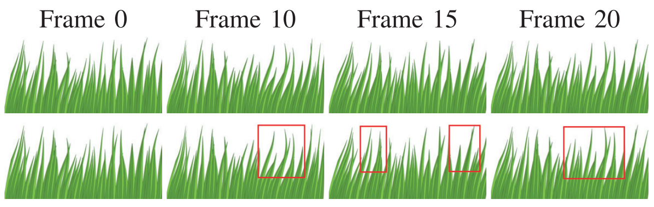

Figure 5. Result animation using the proposed method using a cartoon (©Teemu Korhonen) as an input, the wind direction is from left to right. The most effected regions are highlighted in the second row.

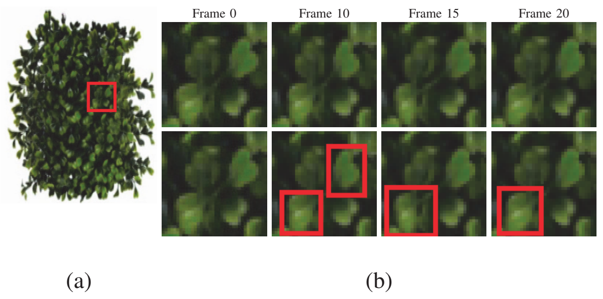

Figure 6. Result animation using the proposed method using a photograph (©Moonglowlily) as an input, the wind direction is from right to left. (a) is one frame of the animated photograph sequence, for clarity, one part of the animated photograph sequence is enlarged and shown in (b), in (b) the most affected regions are highlighted in the second row.

IV. Results and Discussions

The proposed method can be applied to animate vegetation in several types of input images, such as cartoon (Figures 5 and 8), photograph (Figure 6) and sketch (Figure 7). We applied our method to animate the types of vegetation that exhibit non-rigid property, such as dense vegetation: grass field (Figure 5), shrub field (Figures 6), tree crown (Figures 7 and 9, Right) and under water vegetation: sea wheat (Figure 8). We implemented our method on a laptop with Intel Core i7 M620 @2.67Ghz CPU, 4.0GB Memory and Nvidia NVS3100M GPU on the DirectX 9.0c platform. With a 2D fluid simulator (64 by 64 grids) [40], vegetation in images can be animated at 60FPS. We also compare our result with real cartoon and video (Figures 9). The input is one frame of the cartoon or video. Please refer to the supplementary video for the animation sequences.

In non-rigid motions for dense vegetation, the vegetation's shape is perceived as deforming and oscillating, effects such as wave effect may also exists. Our method can capture these non-rigid motions for vegetation in images such as the shape deformation for grass in a cartoon animation (Figure 9, Left) and the wave effect for tree crown in a video: the branches and leaves on the right bend first, then the bending propagates from right to left (Figure 9, Right). For the types of vegetation that exhibit non-rigid property, our method can achieve visually plausible result. Especially for cartoons, such as animation sequences from "Natsume Yuujin-Chou" ("The Natsume's Book of Friends") (©Yuki

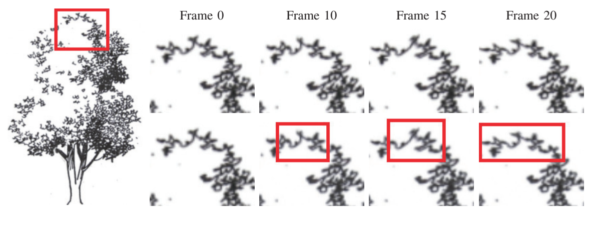

Figure 7. Result animation using the proposed method using a sketch (©HUISJ.COM) as an input, the wind direction is from right to left. (a) is one frame of the animated sketch sequence, for clarity, one part of the animated sketch sequence is enlarged and shown in (b), in (b) the most affected regions are highlighted in the second row.

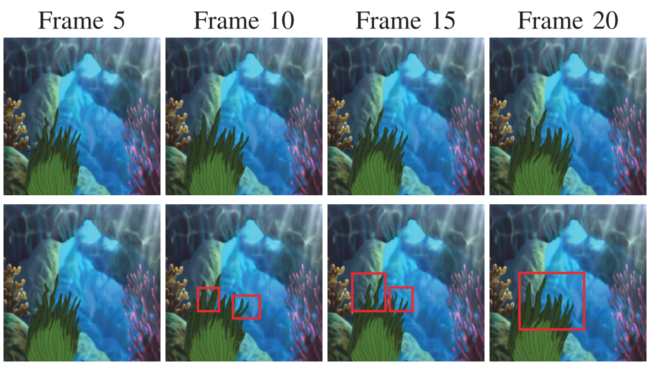

Figure 8. Result animation using the proposed method using a cartoon image from "The Adventures of Little Carp" (©CCTV Animation Inc.) as an input. The most effected regions are highlighted in the second row.

Midorikawa/Nihon Ad Systems Inc./Natsume Yuujin-Chou Project), our method achieves comparable results. However, due to copyright issues, we cannot present them in the paper.

Users can initiate wind by adding external forces to desired areas using a mouse and set parameters to control the frequency, direction and strength of the external forces. Users also can add external forces at any simulation time. The appearance of the vegetation animation can be controlled by specifying the grid's size (s_x, s_y), the scale factor (λ_x, λ_y) and the rigidity parameter (a). Our method can animate vegetation in real-time, as a result users can change the values of parameters on the fly to adjust the animation.

Our technique may perform less well for the type of vegetation that has obvious skeleton shapes such as branches. Because when a tree is swaying in wind, the shape of branches is hardly change, however, the shape of branches is deformed in our method. Therefore, in our experiments, we segment branches as the background. Grid-based warping is simple but not sufficient to simulate the occlusion effect for vegetation under the influence of wind (for example, the leaves or blades may occlude each other). Thus, our method is more suitable to capture the overall appearance of vegetation's motion rather than the detail motion of parts of vegetation. For example, our method performs better for cartoon style images whose structure is simple, or vegetation (such as shrub) oscillating in light breeze, in these cases less non-occlusion artifacts can be noticed. For some vegetation oscillating in light breeze, the vegetation's motion is perceived as each local part of it is slightly vibrating and the propagation and waving effects may not be obvious. Thus in this case, each local part may exhibit non-correlated motion and the wind direction may not be obvious.

V. Conclusion and Future Work

In this paper, we have presented a method for generating vegetation animation for an input image. The proposed method is based on fluid simulation, wave simulation and image warping. Our method does not require tedious work and art skills to create many key frames. This method addresses the non-rigid movements of vegetation in images. Our method is suitable for simulating the non-rigid movements of vegetation in images, especially for dense vegetation in cartoons interacting with wind. As a result, we can simulate grass, shrub, tree crown swaying in wind and aquarium plants swaying in water flow. Users can control the source, direction and strength of the wind or water flow and the property of vegetation on the fly. In the future, we plan to explore other types of deformation method besides the grid-based method to handle more types of vegetation and effects. We also plan to handle the collisions among vegetation layers. Another future work is to animate other natural elements in images, such as water, cloud and fire.

VI. Acknowledgment

We gratefully thank the reviewers for their constructive comments. This research was supported by Fraunhofer IDM@NTU, which is funded by the National Research Foundation and managed through the multi-agency Interactive & Digital Media Programme Office hosted by the Media Development Authority of Singapore.

References

[1] M. Shinya, M. Aoki, K. Tsutsuguchi, and N. Kotani, "Dynamic texture: physically based 2d animation," in *SIGGRAPH '99: ACM SIGGRAPH 99 Conference abstracts and applications*, 1999, p. 239.

[2] Y.-Y. Chuang, D. B. Goldman, K. C. Zheng, B. Curless, D. H. Salesin, and R. Szeliski, "Animating pictures with stochastic motion textures," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 853–860, 2005.

[3] R. Habel, A. Kusternig, and M. Wimmer, "Physically guided animation of trees," vol. 28, no. 2, pp. 523–532, 2009.

[4] F. Perbet and M.-P. Cani, "Animating prairies in real-time," in *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, 2001, pp. 103–110.

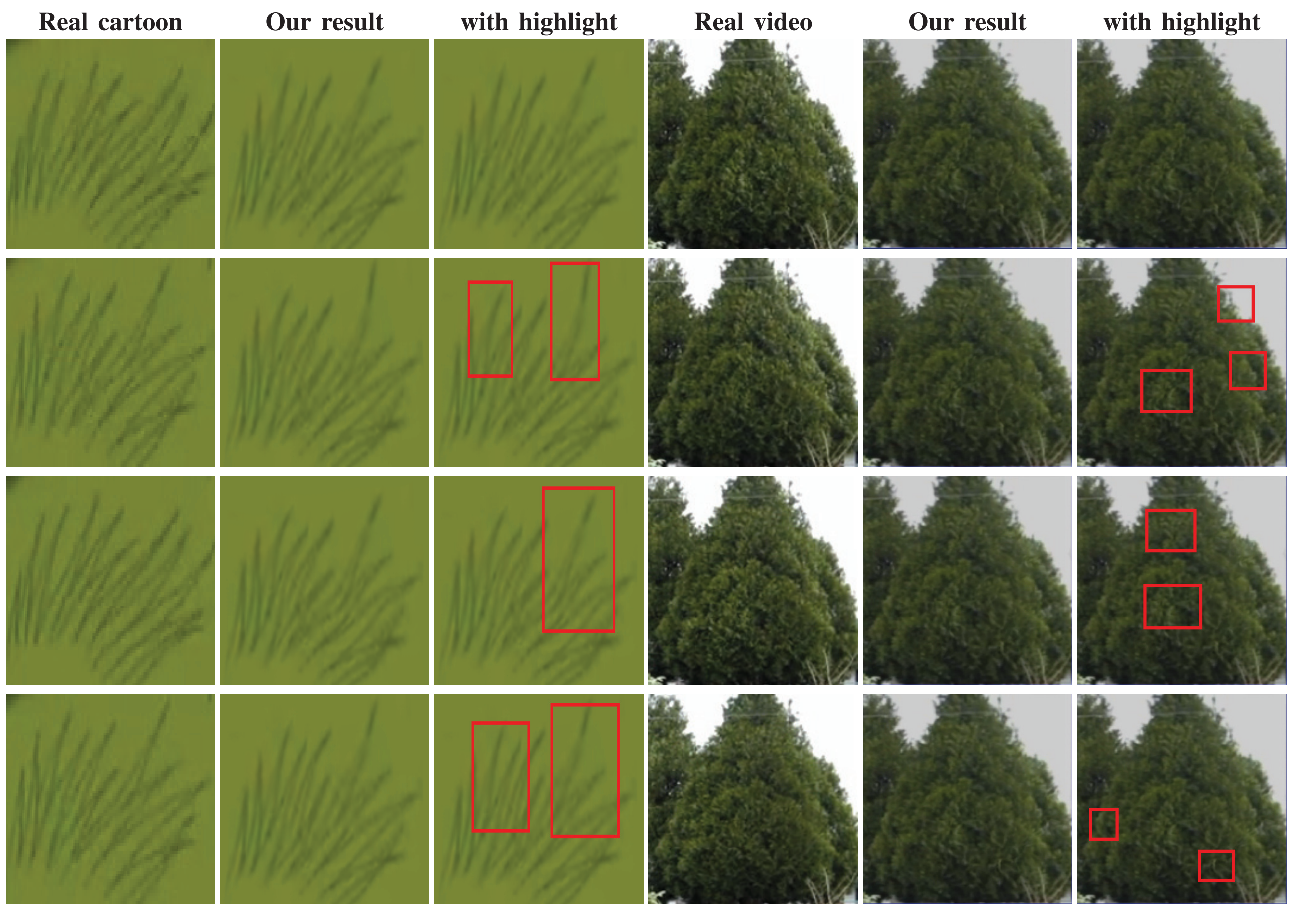

Figure 9. Left: Comparing our result with a real cartoon "Twins on the Pasture" (©CCTV Animation Inc.). Right: Comparing our result with a real video (from www.youtube.com). The wind direction is from bottom right to up left. In the last column of each example, the most affected regions of our result are highlighted.

[5] S. Guerraz, F. Perbet, D. Raulo, F. Faure, and M.-P. Cani, "A procedural approach to animate interactive natural sceneries," in *CASA '03: Proceedings of the 16th International Conference on Computer Animation and Social Agents*, 2003, pp. 73–78.

[6] C. Wang, Z. Wang, Q. Zhou, C. Song, Y. Guan, and Q. Peng, "Dynamic modeling and rendering of grass wagging in wind: Natural phenomena and special effects," *Computer Animation and Virtual Worlds*, vol. 16, no. 3-4, pp. 377–389, 2005.

[7] S. Banisch and C. A. Wuthric, "Making grass and fur move," *Journal of WSCG 2006*, vol. 14, pp. 25–32, 2006.

[8] R. Habel, M. Wimmer, and S. Jeschke, "Instant animated grass," *Journal of WSCG 2007*, vol. 15, pp. 123–128, 2007.

[9] O. Jensn, C. R. Salama, and A. Kolb, "GPU-based responsive grass," in *Journal of WSCG 2009*, 2009.

[10] K. Pelzer, "Rendering countless blades of waving grass," *GPU Gems*, pp. 107–121, Randima Femando, 2004.

[11] R. Zioma, "GPU-generated procedural wind animation for trees," *GPU Gems 3*, pp. 105–120, Hubert Nguyen, 2007.

[12] R. Ramraj, "Dynamic grass simulation and other natural effects," *Game Programming Gems 5*, pp. 411–419, Kim Pallister, 2005.

[13] K. Chen and H. Johan, "Real-time continuum grass," in *VR '10: Proceedings of the 2010 IEEE Virtual Reality Conference*, 2010, pp. 227–234.

[14] J. Stam, "Stochastic dynamics: Simulating the effects of turbulence on flexible structures," *Computer Graphics Forum*, vol. 16, no. 3, pp. 159–164, 1997.

[15] L. B. L. R. Julien Diener, Mathieu Rodriguez, "Wind projection basis for real-time animation of trees," *Computer Graphics Forum*, vol. 28, no. 2, pp. 20–28, 2009.

[16] J. Weber, "Fast simulation of realistic trees," *Computer Graphics and Applications, IEEE*, vol. 28, no. 3, pp. 67–75, 2008.

[17] S. Ota, T. Fujimoto, M. Tamura, K. Muraoka, K. Fujita, and N. Chiba, "1/f β noise-based real-time animation of trees swaying in wind fields," in *CGI '03: Proceedings of Computer Graphics International, 2003.*, 2003, pp. 52–59.

[18] T. Sousa, "Vegetation procedural animation and shading in Crysis," *GPU Gems 3*, pp. 373–385, Hubert Nguyen, 2007.

[19] S. Hu, T. Fujimoto, and N. Chiba, "Pseudo-dynamics model of a cantilever beam for animating flexible leaves and branches in wind field," *Comput. Animat. Virtual Worlds*, vol. 20, no. 2-3, pp. 279–287, 2009.

[20] S. Hu, N. Chiba, and D. He, "Realistic animation of interactive trees," *The Visual Computer*, vol. 28, no. 6-8, pp. 859–868, 2012.

[21] L. Zhang, Y. Zhang, Z. Jiang, L. Li, W. Chen, and Q. Peng, "Precomputing data-driven tree animation," *Computer Animation and Virtual Worlds*, vol. 18, no. 4-5, pp. 371–382, 2007.

[22] L. Zhang, Y. Zhang, W. Chen, and Q. Peng, "Real-time simulation of large-scale dynamic forest with gpu," in *APCCAS'08: Proceedings of the 2008 IEEE Asia Pacific Conference on Circuits and System.*, 2008, pp. 614–617.

[23] Y. Zhao and J. Barbič, "Interactive authoring of simulation-ready plants," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 84:1–84:12, 2013. [Online]. Available: http://doi.acm.org/10.1145/2461912.2461961

[24] X. Xu, L. Wan, X. Liu, T.-T. Wong, L. Wang, and C.-S. Leung, "Animating animal motion from still," *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 117:1–117:8, 2008.

[25] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa, "Video textures," in *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 489–498.

[26] Z. Lin, L. Wang, Y. Wang, S. Kang, and T. Fang, "High resolution animated scenes from stills," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 3, pp. 562–568, 2007.

[27] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the Art in Example-based Texture Synthesis," in *Eurographics 2009, State of the Art Report, EG-STAR*, 2009, pp. 93–117. [Online]. Available: http://hal.inria.fr/inria-00606853/en/

[28] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 479–488.

[29] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: image and video synthesis using graph cuts," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 277–286, 2003.

[30] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 795–802, 2005.

[31] C. Ma, L.-Y. Wei, B. Guo, and K. Zhou, "Motion field texture synthesis," *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 110:1–110:8, 2009.

[32] G. Chen, V. Kwatra, L.-Y. Wei, C. D. Hansen, and E. Zhang, "Design of 2d time-varying vector fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 10, pp. 1717–1730, 2012.

[33] F. Di Fiore, W. Van Haevre, and F. Van Reeth, "Rendering artistic and believable trees for cartoon animation," in *CGI '03: Proceedings of Computer Graphics International 2003*, 2003, pp. 144–151.

[34] W. V. Haevre, F. D. Fiore, and F. V. Reeth, "Physically-based driven tree animations," in *Eurographics Workshop on Natural Phenomena*, 2006.

[35] E. Sugisaki and Y. Yu, "Simulation-based cartoon hair animation," in *WSCG '05: Proceedings of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2005, pp. 117–122.

[36] E. Sugisaki, Y. Kazama, S. Morishima, N. Tanaka, and A. Sato, "Hair motion cloning from cartoon animation sequences," *Computer Animation and Virtual Worlds*, vol. 17, no. 3-4, pp. 491–499, 2006.

[37] P. Noble and W. Tang, "Modelling and animating cartoon hair with nurbs surfaces," in *CGI '04: Proceedings of the 2004 Computer Graphics International*, 2004, pp. 60–67.

[38] J. Liao, J. Yu, and J. Patterson, "Modeling ocean waves and interaction between objects and ocean water for cartoon animation," *Comput. Animat. Virtual Worlds*, vol. 22, no. 2-3, pp. 81–89, 2011.

[39] J. Liao and J. Yu, "Procedural models for cartoon cracks and fractures animations," *The Visual Computer*, vol. 28, no. 6-8, pp. 869–875, 2012.

[40] J. Stam, "Stable fluids," in *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, 1999, pp. 121–128.

[41] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[42] ©Adobe, ©*Photoshop CS5.1*. Adobe System Inc., 2010.

[43] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *ICCV '99: Proceedings of the International Conference on Computer Vision*, 1999, pp. 1033–1040.

[44] K. Laybourne, *The Algorithms and Principles of Non-photorealistic Graphics: Artistic Rendering and Cartoon Animation*. Three Rivers Press, 1998.

[45] J. Zelsnack, "Vertex texture fetch water," *NVIDIA SDK White Paper*, 2004.