Capturing, rendering and simulation for large scale grassland

Bin Sheng, *Member, IEEE*

Abstract—Grass is a very import element of nature, however create and implement a large scale, realistic grassland is not that easy due to extremely high computation complexity and large amount of data needed for simulation and rendering. Common sense tells us grass blades are simple, however there are numerous kinds of grass blade in the world and it's not possible for any system to store all kinds of grass blades beforehand. Obtain grass blade with interactive camera can be an intuitive solution for this problem. We provide a method that can obtain grass blade shape with a depth camera, render large scale grass land efficiently and simulation every single grass blade with individual response on the fly.

Index Terms—Grass, Capture, Render, Simulation, GPU

Fig. 1: Test scene

1 Introduction

When an object move through a grass land, every grass blade on the path is pushed aside or run over, and recovers afterward. Individual responses from every single grass blade greatly improve the user experience and increase scene fidelity. With the development of virtual reality(VR) devices and applications, human-computer interaction is paid more attention than ever. Cameras are deployed extensively in recent years, providing more possibility for VR applications. Depth cameras are especially valued since they are able to provide depth information of any scene, which makes it much easier to rebuild any object or scene.

We intend to build a system which is able to obtain grass blade from camera, refine grass blade shape and use it in the large scale grassland scene with high rendering quality as well as realistic simulation response to collision. With a depth camera, we are able to capture grass blade shape instantly instead of using a small number of preset shapes.

In order to achieve real-time performance of rendering and simulation, every grass blade is modeled as a curve with no more than 64 knots. We expand this curve to a triangle strip in the course of rendering. In our system we usually choose 16 as the number of knots for a single curve to guarantee performance. Thus we have to refine the grass shape we capture from camera, and extract the curve which could best describe the grass blade, and calculate the expansion width for each knot in the curve.

Our system is designed to render and simulate a very large scale grassland in which every single grass blade has individual and vivid response to collision. Therefore we apply GPU-based instancing to lower the requirements for memory and bandwidth, and only pay simulation costs when needed. Meanwhile, this is a tile-based rendering and simulation system, no per-tile data is store unless simulation for the tile is required. We implement the method introduced by Han et al. [1] to simulate our grass blade and adopt the moethod introduced by Fan et al. [2] to do tile management.

Our main contribution is to capture grass blade shape with depth camera, shape refinement according to our simulation algorithm, and large scale grass land rendering as well as simulation with high fidelity.

2 Related Work

The most challenging part of grass modeling, rendering and simulation is caused by extremely large quantity. William Reeves and Ricki Blau [3] addressed those challenges in 1985.

Capturing grass blade with camera has not been explored much, however plenty of researches about hair capture have been conducted. There are some similarities between hair and grass blade for their shapes and movement features. Chai et al. [4] uses single image to do hair modeling, they capture hair style from image and are able to render origin hair style at different angle, with different hair materials and change hair style . Afterwards, they extends their work through user's high-level knowledge to get more accurate hair that matches image and is physically real at the same time. By doing it they are able to finish dynamic hair simulation and interactive hair style editing, which make it possible to apply this hair manipulation in video [5]. A structure-aware hair capture method [6] was introduced in 2013, they adopt a method to generate hair strand segments, set up a connection graph to guarantee hair growth to areas with missed geometry information and connect hair strands with consistent curvature. A method to capture hair using simulated examples was introduced by Hu et al. [7], they use simulated samples as reference to generate hair, and this method could be applied with unconstrained and constrained hair. They also came up with a method to capture braided hair style [8]. They adopt a data-driven reconstruction and use procedurally generated examples to fit captured hair strands. Xu et al. introduced a space-time optimization method to capture dynamic hair. This method could faithfully capture hair strands' shape as well as spatial details. Long et al. [9] introduced a method to model plant from a dozen of images. This method can recover plant's shape automatically while relying on user to provide some hints for segmentation. Tan et al. [10] proposed a method to generate 3D trees from images. They populate tree leaves from segmented source images and use shape patterns of visible branches to predict occluded branches. Tree modeling from single image was introduced afterwards [11]. Yan et al. [12] came up with a method for flower reconstruction from a single image, which uses a cone fitting scheme to maintains flower shape.

In previous works, grass is moved aside when interaction between grass and object happens [13]. Spring-mass system is also used to model grass blade and simulate grass-object interaction [14] [15]. We adopt the simulation algorithm introduced by Han et al. [1]. This method treats collision as hard constraint, meanwhile treat length, bending and twisting as soft constraints in the iterative solver for grass-object interaction. We employ the rendering and simulation framework introduced by [2]. With this frame work, we are able to perform collision computation on GPU and do grass blade instancing on the fly. We implement our capture module on the basis of this framework to obtain more accurate and diverse grass types.

3 Algorithm Overview

The overview of our algorithm is shown in Figure.2. We employ a depth camera in our system which is able to obtain a depth image of target plant. According to this depth image, we are able to calculate the contour of grass blade by distance masking method, which is difficult to achieve without depth information. The single grass blade could be partitioned from a cluster of blades by a particle-flow method [16].

According to contour of grass blade, we generate the skeleton, which is used to describe this blade and is also used in simulation. In vertex shader we expand each knot on the skeleton to form the blade. Expansion widths are calculated from blade's contour and smoothed before rendering.

For skeleton captured from camera, we design an refining algorithm to do simplification. This process is similar to level-of-detail(LOD). Since skeleton knots number is restricted by extremely large amount of grass blades, this refining algorithm could reduce skeleton knots number from over one hundred to 32, 16 or even less, according to knots' movement similarity.

We adopt the GPU instancing scheme used in [2]. We pre-generate a list of grass blade called grass patch. Grass scene is divided into a grid of tiles. Grass blades in each tile are specific number of continuous blades fetched from grass patch. Starting position in grass patch for each tile is calculated by tile's position in the scene. Blade's skeleton is expanded to triangles before rendering. Phong shading model is used in grass blade rendering for the sake of high performance. Subsurface scattering effects is also used in our system to increase fidelity [17]. Tile management guarantee per-blade memory is allocated only when simulation is needed.

4 Blade Capture

we aim to capture grass blade without manual operations and use capture result in the rendering and simulation system immediately after capturing. Skeleton knots number limits the detail level of our grass blade, however it also requires efficient refining algorithm to convert captured skeleton into appropriate blade model in our system. Motivated by other hair and plant capture methods, we adopt a depth camera to capture grass blade. Depth cameras are deployed in variety of applications and becoming prevailing more than ever, for its advantage to recover shapes and capture movements. With depth camera, we manage to capture grass blade and mask non-grass object on the fly.

Figure3 shows the depth image of a cluster of grass blades and its extracted contour. We just use depth information to mask objects that exceed specified depth to get a blob of blades and then extract edge of the blob. Also canny edge detection can be used to extract contour, however depth value, instead of gray is used to do calculation.

Fig. 2: Algorithm overview.

Fig. 3: (a)Depth image and (b)extracted contour

Fig. 4: Single blade partition: Given the (a)contour of a blade cluster,(b)calculate the center point and use particle-flow method to partition each single blade.

4.1 Single Blade Partition

Individual grass blades are partitioned by a particle-flow method [16]. After getting the contour of grass blades, the center of the contour can be easily calculated. Then the distance between any point on the contour and center point can be measured. Local minimum points of point-distance function imply the intersection points between adjacent blades. Boundary between two intersection points is contour of a single blade. Figure.4 shows the partition result projected in 2D plane.

Fig. 5: Skeleton calculation by match symmetrical points on contour.

4.2 Skeleton and Expansion Width Calculation

Given the contour of single blade, we could calculate the blade skeleton. According to our observation on plenty kinds of grass blade ,we suppose that captured blade is symmetrical. Then we could calculate skeleton point by matching symmetrical points on contour. Figure.5 present skeleton calculation according to blade contour. Expansion width for each point of skeleton could also be calculated by symmetrical points' distance.

5 Skeleton Refining

Having the blade skeleton from contour, it can't be used in our rendering and simulation system for its large number of vertices. Therefore we come up with a skeleton refining method to reduce vertices amount according to vertices' movement similarity.

In our system, we define any two vertices' movement similarity by the distance between them and their movement trends. For any two vertices, the smaller the distance between them is, the more similar they are. And for any two vertices, we could calculate the distance between them at any time. We could acquire the distance of two vertices every a few millisecond. Then we are able to calculate a series distance data for these two vertices. Variance is usually used to describe data stability. If distance variance is low, it means the distance between two vertices is stable. In another word, this indicate that these two vertices moves synchronously. We tend to merge vertices that moves similarly and keep vertices that have distinct movement features. Assume we acquire n frames every a few milliseconds, v_m^n is mth vertex at nth frame, therefore we introduce movement difference of any two vertices as:

$$D(v_1, v_2) = dis(v_1, v_2) \times \mu + var(v_1, v_2) \times (1 - \mu) \quad (1)$$

$$dis(v_1, v_2) = \frac{\sum_{i=0}^{n} d(v_1^i, v_2^i)}{n} \quad (2)$$

$$var(v_1, v_2) = (\frac{\sum_{i=0}^{n} d(v_1^i, v_2^i)^2}{n} - \frac{(\sum_{i=0}^{n} d(v_1^i, v_2^i))^2}{n}) \quad (3)$$

dis function represents distance factor and var function represents variance factor. μ is proportion factor for distance and variance, which can be set by user.

We use an iterative algorithm to complete this skeleton refining process. After skeleton calculation, we manage to obtain array of skeleton vertices. We design a greedy algorithm to iteratively merge two adjacent vertices, based on the assumption that if two vertices are the most similar then they should be adjacent vertices in the array. Because distance factor is an very important part in similarity definition, and closer vertices always get lower score for distance factor. This algorithm is illustrated as below:

Algorithm 1 Skeleton Refining

//SkeletonVertices = Calculated from captured result
while SkeletonVertices.size() $> m$ **do**
    DifferenceArr = $\emptyset$
    for v_i in SkeletonVertices **do**
        $d_i = D(v_i, v_{i+1})$
        DifferenceArr.add(d_i)
    end for

    //Search for minimum d in DifferenceArr
    ind = DifferenceArr.minimum()
    $newV = (v_i + v_{i+1})/2$

    //Delete old vertices
    DifferenceArr.delete(i)
    DifferenceArr.delete($i + 1$)

    //Insert new vertex at position i
    DifferenceArr.insert($newV$, i)
end while

Figure.6 illustrates a sequence of grass blade with different refining level. Refining level is set by user according to the performance requirement of application and hardware level.

6 Rendering and Simulation

7 Implementation and Results

8 Conclusion

The conclusion goes here.

Fig. 6: Demonstration of Skeleton refining process with a sequence of grass blade projected to 2D plane.

Acknowledgments

The authors would like to thank...

References

[1] D. Han and T. Harada, "Real-time hair simulation with efficient hair style preservation," pp. 45–51, 2012.

[2] Z. Fan, H. Li, K. Hillesland, and B. Sheng, "Simulation and rendering for millions of grass blades," in *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*. ACM, 2015, pp. 55–60.

[3] W. T. Reeves and R. Blau, "Approximate and probabilistic algorithms for shading and rendering structured particle systems," in *ACM Siggraph Computer Graphics*, vol. 19, no. 3. ACM, 1985, pp. 313–322.

[4] M. Chai, L. Wang, Y. Weng, Y. Yu, B. Guo, and K. Zhou, "Single-view hair modeling for portrait manipulation," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, p. 116, 2012.

[5] M. Chai, L. Wang, Y. Weng, X. Jin, and K. Zhou, "Dynamic hair manipulation in images and videos," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 75, 2013.

[6] L. Luo, H. Li, and S. Rusinkiewicz, "Structure-aware hair capture," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 76, 2013.

[7] L. Hu, C. Ma, L. Luo, and H. Li, "Robust hair capture using simulated examples," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 126, 2014.

[8] L. Hu, C. Ma, L. Luo, L.-Y. Wei, and H. Li, "Capturing braided hairstyles," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, p. 225, 2014.

[9] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang, "Image-based plant modeling," in *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 599–604.

[10] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image-based tree modeling," in *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 87.

[11] P. Tan, T. Fang, J. Xiao, P. Zhao, and L. Quan, "Single image tree modeling," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5, p. 108, 2008.

[12] F. Yan, M. Gong, D. Cohen-Or, O. Deussen, and B. Chen, "Flower reconstruction from a single photo," in *Computer Graphics Forum*, vol. 33, no. 2. Wiley Online Library, 2014, pp. 439–447.

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

[14] K. Chen and H. Johan, "Real-time continuum grass," in *Virtual Reality Conference (VR), 2010 IEEE*. IEEE, 2010, pp. 227–234.

[15] N. Hempe, J. Rossmann, and B. Sondermann, "Generation and rendering of interactive ground vegetation for real-time testing and validation of computer vision algorithms," 2013.

[16] B. Neubert, T. Franken, and O. Deussen, "Approximate image-based tree-modeling using particle flows," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 88, 2007.

[17] T. Sousa, "Vegetation procedural animation and shading in crysis," *GPU Gems*, vol. 3, pp. 373–385, 2007.

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.