Krysten Tachiyama
Greg Richardson
ANIM 332
Feb 27, 2021

The Particle Distribution Force Model for Water Simulations and its Applications in Modern Animations

Animation serves as a unique story-telling medium that interweaves art, math, and science in order to make an audience believe in a fictional world. Therein exists the ability to bring the most imaginative and supernatural ideas to life inside of a computer. While this unrestricted artistic freedom is exciting, it is important to ensure that the story being told on screen is still conceivable enough to immerse into; a world that is too unnatural can leave an audience disconnected and unconvinced. Natural phenomena such as light, smoke, fire, water, etc. are grounded in math and science to ensure that they are realistic and recognizable.

These phenomena can be replicated using multitudes of individual particles. Particle simulations are used as an effective tool for artists as they are a simplification of physics. Akin to the real world, animation uses forces to set particles into motion. Effects such as splashing water are traditionally derived from either noise or fluid dynamics, but both approaches come with caveats. Noise-based forces are efficient in terms of time but are not accurate enough in capturing innate details. While fluid dynamics are able to produce more realistic fluid properties, they are computationally expensive (Yuksel, Maxwell and Peterson, 2014). For feature films, it is important to utilize simulations that maximize quality while minimizing computing resources. This paper discusses a different type of force model that more efficiently simulates fluids by calculating forces based off of the distribution of particles in a given space. It then dives into *How to Train Your Dragon: The Hidden World* and *Frozen 2* to show how these films built upon this force model in order to create various water effects.

The Particle-to-Distribution Approach to Calculating Forces

The particle-to-distribution force model adopts a local distribution of neighboring particles, or a covariance body, in order to more efficiently make accurate simulations with greater optimization than previous traditional methods.

The ‘covariance body’ is a nonstandard term created by programmer and video game designer Jonathan Blow, and is used to represent a specific set of values that can be manipulated as a coherent body (2004). It is made up of 3 quantities: 1) A n × n covariance matrix C that determines the size of the particle distribution, 2) A n × 1 vector representing the center of mass x, and 3) The total scalar mass m.

The covariance matrix is especially important as it determines the shape of the covariance body and describes how mass is distributed in space around its center. In this context, variance characterizes the approximate width of the distribution of input values. These input values each have an associated mass such that the total mass of all the points is $m_{total} = \Sigma m_i$, where m_i is the mass of a the i^{th} point. In correspondence to the center of mass, the weighted mean of these

points is $v_{center} = m_{total}^{-1} \Sigma m_i v_i$ where v_i is the position of the i^{th} input particle. We can then let $\vec{v}_i = (x_i, y_i, \ldots, n_i)$ represent the position of a point relative to its center of mass in the n^{th} dimension. Therefore, the covariance matrix can then be calculated as $C = m_{total}^{-1} \Sigma m_i \vec{v}_i \vec{v}_i^{\,T}$, where $\vec{v}_i^{\,T}$ is the transpose of $\vec{v}_i$ (Blow, 2004). The product of $\vec{v}_i \vec{v}_i^{\,T}$ is important because it guarantees us a positive semidefinite matrix such that its eigenvalues are insured to be non-negative numbers. Because the sum of 2 positive semidefinite matrices is still positive semidefinite, the covariance matrix's eigenvalues will be non-zero as well. The eigen-stuff of covariance matrix determines the shape of the covariance body. By finding the square root of the eigenvalues, you get the length of each axis (for a n × n matrix, there can be at most n axis). The eigenvectors of the matrix determine the axis orientation and is used in describing the force per particle.

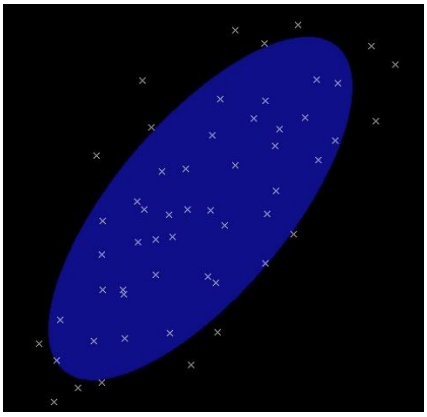

Figure 1: A 2D set of input particles and the elliptical shape determined by its covariance matrix (Blow, 2004).

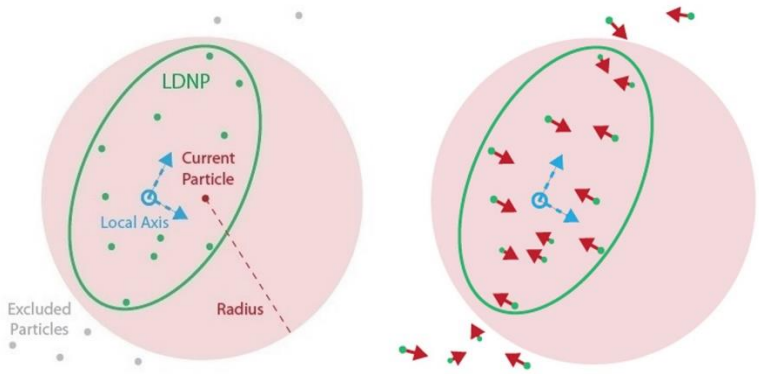

Figure 2: 2D examples of forces determined by the covariance body (Yuksel, Maxwell and Peterson, 2014)

In order to simulate forces, the user is able to define the borders of the covariance body as well as customize parameters that control the magnitude of forces in ten different directions. The first nine force directions either pushes/pulls, attracts, or rotates particles along the axes of the local coordinate space defined by the eigenvectors. The last direction attracts the particles to the center of the covariance body. The combinations of these forces are able to create intrinsic shapes. For example, "the force toward the axis of the largest eigenvector aligns all the particles into a long chain by reducing the variance in all but the major direction" (Yuksel, Maxwell and Peterson, 2014). This chain force is most often used for simulating splashes.

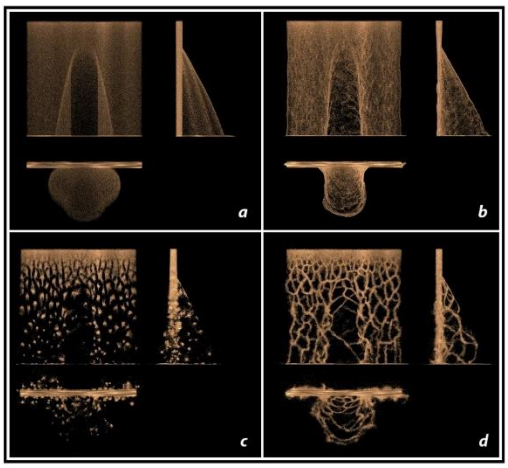

Figure 3: Comparison of (a) plain simulation to (b,c,d) different forces (Yuksel, Maxwell and Peterson, 2014)

Optimizing a large amount of particles is done using a k-d tree to identify neighboring particles and merging them together if they are closer than a fraction of covariance body's borders. The larger the search radius, the more computation time is saved due to the higher number of particles being removed. The k-d tree is a variation of the binary search tree where each of the tree's leaf nodes represents a k-dimensional point. Generally speaking, the time complexity of finding the nearest neighbor of a balanced k-d tree is an average of O(log n). K-means clustering is also used to utilize the merge points as initial seed. "This method ensures that the k-d tree complexity is a function of the search radius but is nearly independent of the number of particles. Given a fixed search radius, the computation time scales almost linearly with the number of simulated particles" (Yuksel, Maxwell and Peterson, 2014). Particle-to-particle interactions are traditionally expensive to calculate; optimizing this process enables the simulation of millions of interactions at a scalable level, allowing animators to utilize local distribution-based forces to enhance large simulations.

How to Train Your Dragon: The Hidden World Waterfall Instancing

The scenery of *How to Train Your Dragon: The Hidden World* is filled with beautiful Nordic landscapes that immerses its audience into a realistic and compelling universe. In order to achieve this, a substantial number of waterbodies needed to be simulated, including oceans, lakes, calderas, and about 4,000 waterfalls. Because the expenses required to create these complex sets were not scalable through a traditional approach, a more efficient method needed to be built. DreamWorks developed InstaFalls, a collection of tools that were used for "streamlining the creative process by minimizing the time spent on simulations, iterating faster thanks to real-time feedback, and managing the large quantity of generated data" (Opstal, Woo and Aubel, 2019). This system allowed artists to populate the environment sets with the necessary water elements without consuming too many resources.

Due to its ability to accurately produce liquid shapes, the force particle simulation previously discussed was built upon for InstaFalls. Because the original particle-to-distribution simulation required more artistic iterations than desired to construct a realistic waterfall effect, the calibrations of parameters were minimized to increase productivity.

Figure 4: A scene from How to Train Your Dragon 2 with pouring water simulated using forces based on the local distribution of neighboring particles (Yuksel, Maxwell and Peterson, 2014)

Waterfalls have an elemental characteristic in which droplets on the edge of the waterbody are caught by air friction, causing them to separate and slow down, and eventually evaporate. This peeling effect was achieved by using *OpenVDB*, an open-source software library that allows for the efficient handling of sparse volumetric data. By selectively removing divergence from the vector field, particles that deviate enough were removed from the waterfall and were used as emitters for a subsequent mist simulation.

InstaFalls is an instancing system that adapts to physical terrains in real time. It uses a library of cached simulations in a neutral state, meaning that the simulations are stored without collisions in temporary memory that allows for fast access. Strokes from the user emit a few guide curves that run a custom multi-threaded physics solver to produce a set of target curves. The physics solver is able to model intrinsic attributes of water by controlling properties such as collision, friction, bounce, and stickiness. It constantly runs in the background, allowing for dynamic calculations of new target curves when the surrounding environment is edited. Once a look was approved, it could be saved into presets and applied to all instances. The volumes of particles were additionally clustered for aesthetics or rendering convenience. In the sequence where Hiccup and Astrid discover the Hidden World, "over 3000 waterfalls were instanced from only 12 pre-simmed caches." This instancing method ensured that tools were based on painting and real-time feedback while providing a faster and more artistic-friendly way to populate water sets (Opstal, Woo and Aubel, 2019).

Frozen II: Simulating the Mane and Tail of the Nokk

The Nokk in Frozen 2 is presented as an elemental spirit of water that takes the shape of a horse and serves as the protector of the Dark Sea. Throughout the film, it oscillates between liquid and ice states while submerging below the ocean's surface and galloping above it. The challenge of designing the Nokk was balancing the instinctive behavior of water with while maintaining a believable look of a horse such that its appearance served the story.

Tracking a character made from water in an ocean environment meant that more noticeable parts, such as the mane and tail, were visually important for drawing in the audience's eyes. Simulating

these aspects of the horse proved to be challenging. According to visual effects supervisor, Steve Goldberg, “there were early tests where we had too much water coming off of the mane and the tail like a fire hydrant. So we slowly came down from that, with the art department and tech animation coming up with different ways to control the mane and tail like like a gentle waterfall” (Desowitz).

Starting with the Nokk’s body movement, the motion of the mane and tail was explored with a drawover pass that was used as a reference for technical animators to simulate overall curves. Real-time expressions allowed artists to append and modify fluid sheet and natural water-tearing simulations to emulate the fluid motion of its hair.

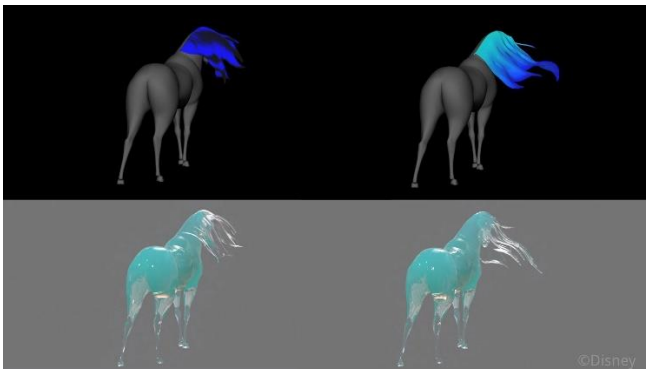

Figure 5: Fluid sheets and water tearing simulations applied to the mane and tail (Hutchins et al., 2020)

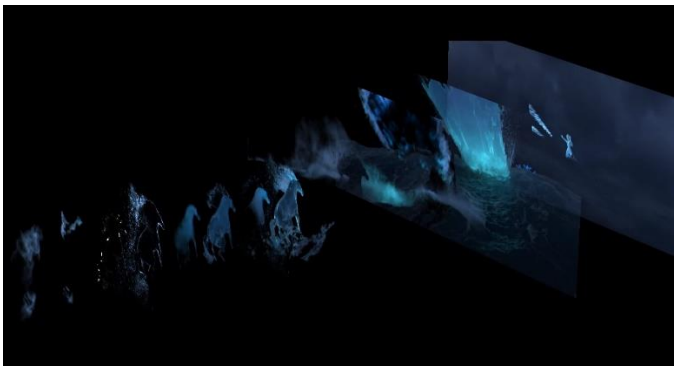

Figure 6: Layers showing the water and spray effects applied to the Nokk (Hutchine et al., 2020)

Water effects were inspired by the studies of laminar flow fountains, windy waterfalls, and crashing waves. Using the curves from technical animation as the source for particle simulations, the particle-to-distribution approach to applying forces was used as a foundation for effects animators to simulate the fluid-like behavior of the mane and tail. Additional simulations driven by these particles were ran as artists added multiple layers of water and spray to the Nokk’s hooves, mane, and tail. Individual shots were edited in order to elongate the main and tale with water droplets and spray simulations (Hutchins et al., 2019).

Conclusion

The particle-to-distribution force model relied heavily on linear algebra and software engineering in order to calculate the interaction between particles. While this approach was not an intuitive solution, it was a mathematically creative one that was able to produce more accurate and optimized simulations that previous models. The force model was able to be tweaked and built upon to enhance a large portion of the particle simulations for *How to Train Your Dragon: The Hidden World* and *Frozen 2*. Most importantly, while both films relied on the realistic effects given by the force model, the math and science was always applied in conjunction with artistic direction in order to create scenes that best served the story.

Works Cited

Blow, J. (2004). My Friend, the Covariance Body. Available at: http://number-none.com/product/My%20Friend,%20the%20Covariance%20Body/ [Accessed 27 Feb. 2021]

Desowitz, B. (2019). 'Frozen 2': How Disney Swooped 'Into the Unknown' With New Tech and Social Engineering. Available at: https://www.indiewire.com/2019/11/frozen-2-disney-into-the-unknown-tech-1202190687/ [Accessed 27 Feb. 2021].

Hutchins, D., Cameron, B., Bryant, M., Lehmann, R., and Tadivoeva, S. (2020). "Frozen 2": Creating the Water Horse. In: *ACM SIGRAPH 2020 Talks (SIGRAHO '20)*. [online]: ACM, Article 23, pp. 1-2. Available at: https://dl-acm-org.electra.lmu.edu/doi/10.1145/3388767.3407345 [Accessed 27 Feb. 2021].

Opstal, B.V., Woo, Y., and Aubel, A. (2019). Instafalls: How to Train Your Waterfalls. In: *ACM SIGGRAPH 2019 Talks (SIGGRAPH '19)*. [online]: Los Angeles: ACM, Article 73, pp. 1–2. Available at: https://doi-org.electra.lmu.edu/10.1145/3306307.3328156 [Accessed 27 Feb. 2021].

Yuksel, C., Maxwell, K., and Peterson, S. (2014). Shaping Particle Simulations with Interaction Forces. In: *ACM SIGGRAPH 2014 Talks (SIGGRAPH '14)*. [online] Los Angeles: ACM, Article 42, pp. 1. Available at: https://dl-acm-org.electra.lmu.edu/doi/10.1145/2614106.2614121 [Accessed 27 Feb. 2021].