Real-Time Sand Dune Simulation

BRENNEN TAYLOR, Texas A&M University, United States
JOHN KEYSER, Texas A&M University, United States

We present a novel real-time method for simulating aeolian sand transport and dune propagation. Our method is a GPU-based extension of the Desertscapes Simulation sand propagation model to additionally capture echo dunes and obstacle interaction. We validate our method by comparing it against an existing study of echo dune evolution in a wind tunnel environment. Additionally, we demonstrate the significantly improved performance of our method via comparison to the existing, CPU-based method. Lastly, we validate our method by comparing it to a published study exploring the evolution of dunes in a bidirectional wind environment driven by an offline, cellular autonoma based method. We conclude that the presented method is a simple and helpful tool for users in multiple domains who wish to capture physically plausible desertscape evolution in real time.

CCS Concepts: • **Computing methodologies** → **Physical simulation**; **Graphics processors**.

Additional Key Words and Phrases: sand-dunes, simulation, real-time, GPU

ACM Reference Format:
Brennen Taylor and John Keyser. 2023. Real-Time Sand Dune Simulation. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 1, Article 15 (May 2023), 18 pages. https://doi.org/10.1145/3585510

1 INTRODUCTION

Deserts are a type of biome characterized by low humidity and high temperatures, which evolve primarily due to aeolian (i.e. wind-driven) sand transport. These environments can vary widely in their presentation, often containing a variety of types of dune structures whose morphology is defined by local wind conditions and sand availability. For example, in environments with low sand availability and uniform wind, Barchan dunes are formed, presenting as crescent-shaped with a wind facing steep side and two horns extending downwind. For environments with an increased availability of sand and uniform wind, parallel transverse dunes are formed, extending perpendicular to the wind vector. By introducing bidirectional or complex wind regimes, linear or star dunes form. Similarly, a number of unique dunes form only with the introduction terrain gradients. For environments with more gentle terrain gradients, dunes slowly scale the slope face, forming climbing dunes. For those environments with steeper gradients like cliffs, echo dunes form upwind from and extend along the cliff face, kept at bay from climbing the terrain by reverse eddy currents.

Many disciplines, such as computer graphics, climatology, and geology, desire to accurately simulate and model these processes and capture the full range of dune morphology. In computer graphics, real-time video game studios and VFX studios often desire automated tools for simulating realistic terrain generation for virtual environments. In climatology and geology, scientists aim to accurately simulate and predict the effects of climate change on existing biomes on Earth.

Authors' addresses: Brennen Taylor, Texas A&M University, 400 Bizzell St, College Station, TX, United States, brtaylor1001@tamu.edu; John Keyser, Texas A&M University, 400 Bizzell St, College Station, TX, United States, keyser@tamu.edu.

2577-6193/2023/5-ART15 $15.00
https://doi.org/10.1145/3585510

Existing techniques for modeling desertscape evolution typically fall into two categories. The first category contains methods that focus on accurately modeling transport, often at the expense of limiting their use to the offline domain and/or restricting the dimensions or resolution of the simulated environment. The second category contains those methods targeting lower computational cost by approximating the physical processes at work, often resulting in less accurate predictions or a reduced ability to capture real-world features.

In this work, we present a natural evolution of the Desertscapes Simulation method [Paris et al. 2019] for dune simulation to the GPU, enabling a real-time simulation of dune evolution. Additionally, we present a novel extension of the sand transport model to correctly model echo and climbing dunes. Additionally, we augment the method to allow for interaction with physical objects in the scene.

Our main contributions are 1) a method of utilizing existing GPU hardware to transport sand material for dune formation in real-time; 2) an extension of the model to correctly simulate echo and climbing dunes; 3) an extension of the model to support obstacle interaction. Additionally, we verify that our model provides realistic results by comparing to an offline technique from the geoscience domain, an analysis we believe to be the first cross-domain verification of this behavior.

The rest of this article is organized as follows: In Section 2, we summarize existing work on simulating dune and sand ripple environments. In Section 3, we present our method for simulating aeolian sand transport on the GPU, as well as extensions to support echo and climbing dune simulation and obstacle interaction. In Section 4, we validate our method and compare it to the previous CPU-based model. Finally, in Section 5, we conclude the article with a summary of our presented method and a discussion of limitations and future directions of the work.

2 RELATED WORK

The existing approaches for simulating sand material transport vary widely in their target application. In addition to work that simulates transport, another line of work focuses on measuring the results of sand transport, such as from wind tunnel experiments capturing significantly scaled-down physical processes.

2.1 Real World Measurements

A number of works focus on simulating the behavior of dune creation and sand transport in wind tunnel environments. Attempting to reproduce coarse sand propagation on a smaller scale, these methods use fine sand and lower wind speeds to generate controlled, scaled-down versions of large-scale dune environments.

One notable study focuses specifically on climbing and echo dunes, simulating a multitude of environments in a wind tunnel environment [Tsoar 1983]. In this work, the authors find that a terrain height increase at an angle of 55 degrees differentiates climbing from echo dune creation, with climbing dunes occurring for less than 55 degrees and echo dunes forming in the alternative.

Another work focuses on investigating the evolution of barchan dunes via a wind tunnel experiment. In their investigation, the authors introduce an oscillating amount of sand to the environment and measure the change in the shape of barchan dunes, noting a monotonic, linear increase on the wind-facing side of the dune, while the horns of the dunes were non-monotonic with time.[Zhang et al. 2013].

A last study models sand dune formation on the western coast of Hainan island using a wind tunnel experiment [Li et al. 2007]. For this study, the authors construct a scale model of the island terrain, on which they introduce sand and a bidirectional wind force by placing the model on a slowly rotating disk in the tunnel. By doing so, the authors were able to produce scale versions of

the island sand deposition zones, as well as a variety of the dune types found on the island, such as elliptical dunes, barchan dunes, and transverse dune ridges.

2.2 Precision Driven Methods

Many approaches desire an accurate simulation and are willing to restrict the computation domain or greatly increase their computation time in order to obtain this accuracy.

Works in physics and geomorphology have targeted extremely accurate simulations of sand dunes and sand ripple generation. One such method focuses on generating physically accurate sand ripples [Huo et al. 2021]. In their work, the authors present a numerical model for directly simulating 3D, long term development of sand ripples via simulation of the 3D wind field and sand particles. Particle collisions, including mid-air collisions, as well as particle ejections from the sand surface due to collision with other particles striking the surface largely drive the evolution of this model. While accurate, this model is extremely expensive and in no way applicable to either offline or real-time graphics domains.

Another novel approach reintroduces a complex computation domain, at the expense of slower computation time [Rozier and Narteau 2013]. In their technique, these authors present a method for utilizing a cellular autonoma model, in which the computation domain is represented by voxels, each assigned one of a small number of states. During simulation, pairs of neighboring voxels are stochastically selected based on their current state, and converted to a new state, to complete the stochastically sampled event. Presenting a number of events, such as sand generation, sand transport, etc., this method is able to accurately capture a number of sand dune structures.

In two important follow-up studies, the behavior of the previously mentioned cellular-autonoma model is further explored under multiple levels of sand availability, and different variations of a bidirectional wind regime parameterized by the angle of a second wind direction from a base wind direction, as well as the ratio of the time the environment experiences the base wind vs the secondary wind. The first study explores the evolution of sand dunes in a bidirectional wind environment, focusing on low sand availability in the form of a localized sand source on a non-erodable bedrock, as well as a fully available sand source in the form of an erodable sand bed [Gao et al. 2015]. The second study then builds upon this first study, exploring the wide variety of dunes constructed with intermediate availability of sand between fully available, and extremely low availability [Tsoar 1983]. In this second work, the authors demonstrate that varying the environment parameters, notably sand availability and wind angle, has a large impact on the final sand dune evolution.

Focusing on a slightly different domain, one approach builds upon the cellular autonoma model, demonstrating its applicability to different materials (specifically, snow) which experience cohesive forces [Kochanski et al. 2019a]. The authors achieve this by introducing a number of additional stochastic event types. While not discussed in this prior work, the introduction of cohesion to this model would likely allow for the simulation of wet sand evolution.

2.3 Real-Time and Approximate Methods

Other methods target more interaction, allowing a user to change the simulation behavior in real time.

Early attempts to generate physically plausible terrain environments relied on the utilization of fractal noise and displacement to add important features to a height map for rendering [Mandelbrot and Wheeler 1983]. However, while these techniques allow for arbitrarily high-frequency details to be added to an environment, they struggle to generate realistic-looking terrain created naturally by physical processes.

Taking inspiration from nature, a number of techniques attempt to simulate erosion processes to generate more natural-looking terrain. One such technique improves on fractal noise for terrain

generation by adding physical erosion due to thermal shocks and hydraulic erosion [Musgrave et al. 1989]. It's important to note that, while noted for their early foray into erosion-based terrain simulation, these techniques do not target sand environments which are primarily due to aeolian wind processes.

An early work targeting desert environments presents a method for simulating sand ripples [Benes and Roa 2004]. Driven by a limited approximation of wind transport of sand, and saltation, this method generates plausible sand ripples and even accounts for obstacles. However, it's important to note that the algorithm driving this method, while inspired by physical processes, is not physically accurate.

In their early work, Wang and Hu focus on simulating aeolian sand transport and ripple formation in real-time [Wang and Hu 2009] [Wang and Hu 2012]. Their model captures the movement of sand particles through an environment, modeling the physical processes of sand particle interaction [Bagnold 1971], focusing on accurate sand interaction behavior while utilizing simplified wind and vegetation models. While their method successfully captures a wide variety of ripple formations, as demonstrated in the paper in multiple environment constructions, the method fails to capture large-scale dunes, focusing more on smaller-scale environmental features.

Another important approach describes a multi-layered, slab-based representation of terrain [Cordonnier et al. 2017] and terrain evolution. While focused on wind and water erosion of bedrock to form different terrain features, this work presents a framework that is built upon to simulate sand material transport.

A more recent method uses a high-level wind field to drive an interactive, but not yet real-time simulation of sand material transport [Paris et al. 2019]. This method simulates a multitude of sand dune types, including barchan, longitude, and anchored dunes utilizing a multi-layered, slab-based representation of terrain features. The authors show that their method is able to capture a wide variety of sand dune structures based on different sand availability and wind characterizations. However, their method is event-driven and thus restricted to the CPU, limiting overall performance, with the method falling short of real-time. Additionally, the model is limited to simply bedrock, sand, and vegetation layers, limiting the interaction of sand dunes with obstacles. However, as we show in this work, it is possible to extend this method to GPU hardware by relaxing the event-driven model, bringing the simulation to a real-time domain. Likewise, we demonstrate an extension of the method to correctly model echo dune formation when obstacles are placed in the environment.

2.4 Deep Learning Approaches

While our presented method does not make use of deep learning techniques, such as GANs, a summary of the latest related work would be incomplete without a discussion of recent works involving the application of neural networks to predict the evolution of material, such as snow or sand.

In their recent proposal [Kochanski et al. 2019b], one research group discusses an approach to training a GAN-type network to predict the evolution of sand dune morphology at much greater speeds than direct simulation could allow. Another work leverages neural networks to predict storm erosion of Dutch coastline dunes [Athanasiou et al. 2022]. While able to reproduce beach erosion behavior in the simulated dataset, this method is limited to a beach environment and thus has little application to sand dune propagation. Additionally, a lack of real-world measured data led the authors to generate and train on primarily simulated data, limiting the method's level of grounding in predicting real-world phenomena.

One last work utilizes neural networks to predict sand strength and cohesion parameters based on easy-to-measure properties of sand, such as roundness, size, etc [Sharma et al. 2022]. While not

focused on sand propagation, this method demonstrates the possibility of using a neural network to drive/tune the parameters of other simulation methods to achieve realistic results.

One commonality to all these approaches is their requirement of large data sets for training; data sets that are largely generated through expensive computation methods, such as a cellular automaton model of sand transport [Zhang et al. 2010], due to a lack of real-world, measured data. Given the wide variability in the types of interesting environments of focus for prediction in these neural network approaches, there is a strong desire for a fast, accurate simulation model for data generation which is able to accurately reproduce physically accurate dune creation and propagation.

3 METHOD

The previously published sand dune simulation model by Paris et al. [Paris et al. 2019] is a great candidate for utilizing the GPU due to its massively parallel nature, with many aspects of the simulation pipeline trivially parallelizable. However, the sand transport and cascade event resolution steps are not trivially parallelizable due to the possibility of a sand transport (including saltation and reptation) or a cascade event triggering an additional set of sand cascade events, which must then be resolved. The previous method solves this via an event-driven approach, where each sand transport or cascade event, during their time sequential resolution, could trigger additional events. While powerful and accurate, this event-driven model poorly maps to the GPU.

In our method, sand material is represented as small square slabs of height h_s meters and width l meters. The number of slabs at a particular location determines the amount of material present in that cell. A pair of N x N grids are used to store the (integer) number of sand G_s, and bedrock G_b slabs at each cell. We additionally introduce a new obstacle layer G_o that represents static obstacles placed in the scene. Each cell in the obstacle layer stores the height of the obstacle material located in that cell. These obstacles are treated as being fully embedded in the bedrock layer, such that sand is only able to travel over the obstacles, and can either be present for a given height extending from the bedrock or not be present at all. To compute the overall height of the simulated surface, the sand grid is layered on top of the obstacle grid, which is then layered on top of the bedrock grid, such that the overall surface height at the cell located at row r and column c is

$$H(r, c) = h_s(G_s[r, c]) + G_o[r, c] + h_s(G_b[r, c]) \tag{1}$$

Two additional grids are required, following the previous desertscapes model [Paris et al. 2019]. First, a vegetation grid G_v represents the amount of vegetation present in the scene, where the vegetation is placed on top of the sand. Each cell of the vegetation grid represents a value between 0 and 1, with 0 representing no vegetation, and 1 representing a full saturation of vegetation.

A second grid G_d can be provided that represents the bedrock density or hardness. Like vegetation, each cell in this grid stores a value between 0 and 1, with values of 0 corresponding to an easily erodible bedrock, and values of 1 representing difficult-to-erode bedrock.

Next, we present the overall desert simulation algorithm, focusing our discussion on those portions that are novel and/or required important changes to be correctly and efficiently implemented on GPU hardware.

3.1 Simulation

The result of a simulation step is updated bedrock and sand heightmaps, G_b and G_s, for rendering or export. Unlike the previous desertscapes model, whose event-driven approach allows for in-place updates of sand transport and sand cascade resolution, our parallel GPU approach requires synchronization to correctly resolve multi-threaded updates of the same grid cell. Thus for passes that require reading from and writing to the sand or bedrock buffers, we take the following approach

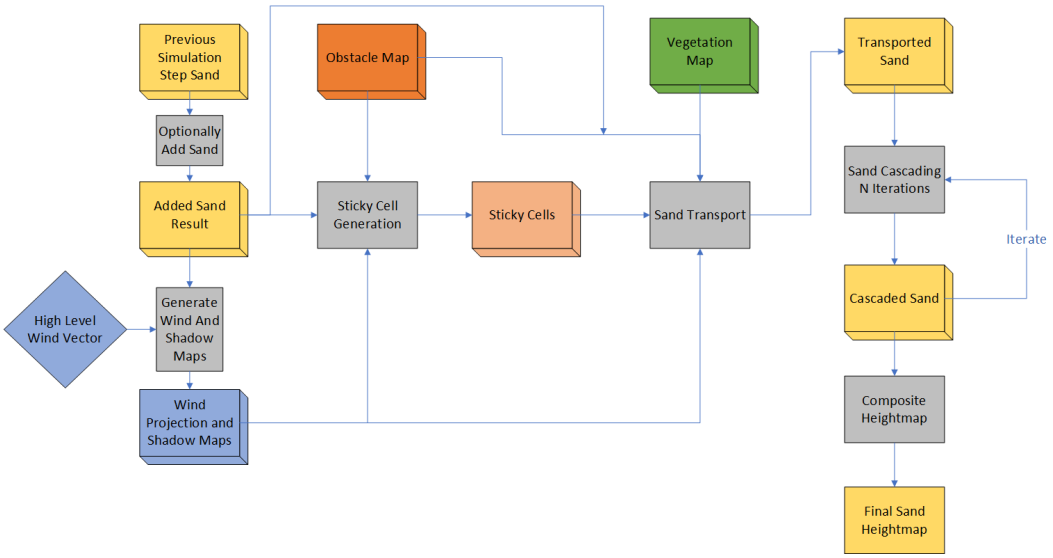


Fig. 1. An overview of the pipeline simulating a single step of the desert environment simulation.

- First, lock a copy of the bedrock and sand grid's current state for all readings in that pass.
- Second, write all changes to a separate copy of the bedrock and sand grid's current state, representing the updated grid.
- Lastly, continue forward using only the newly updated grid.

While this approach solves the issue of completing threads reading cell values of different logical timesteps within the same pass, there remains the issue of threads competing to update write values within the same pass.

To solve this, we utilize GPU compute shader atomic updates of cell values within the bedrock or sand write grids. The reader will note that rather than storing a single height value per bedrock or sand cell, we instead store an atomic integer value representing the number of discrete blocks, from which we reproduce the height of the sand or bedrock. This representation is required as compute shaders do not support atomic float operations.

With this, all synchronization issues are resolved, and thus we describe each step of the pipeline. An overview of the pipeline can be seen in Figure 1.

3.1.1 Sand Injection. Starting with the previous timestep's final sand heightmap $H(x, y)$, we optionally allow sand to be added via compute shader. All added sand is assumed to be directly added to the sand heightmap without experiencing cascade stabilization. We utilize this for the injection of sand in both our wind tunnel simulation [Tsoar 1983], and our comparison to the cellular autonoma model [Lü et al. 2018].

3.1.2 Wind Projection and Shadowing. Next, we compute the projection of wind onto the terrain surface, as well as the wind shadowing of the surface. These passes are kept consistent with those presented in the previous Desertscapes Simulation work [Paris et al. 2019] and are trivially parallelizable on the GPU as each cell writes only to itself. We summarize each below.

The projected wind field $W(p)$ at point p is computed by projecting a time-varying, high-altitude wind field $A(p)$ to the surface of the terrain, exactly following the previous work [Paris et al. 2019]. This projection first accounts for Venturi effects (i.e. the acceleration of wind at altitude)

by computing $V(p) = A(p)(1 + k_W H(p))$, where k_W is set to $5x10^{-3}$. Then, defining $H_{50}(p)$ and $H_{200}(p)$ as the sand heightfield convolved with a Gaussian kernel of radius 50 and 200 respectfully, we compute the projection of $V(p)$ onto the terrain with the following equation, presented here for completeness, and with $k_{H_{50}} = 5$ and $k_{H_{200}} = 30$. For a more general presentation and justification of these equations and parameters, please see the previous paper. [Paris et al. 2019].

$$W(p) = 0.2\ F_{50}(p) \circ V(p) + 0.8\ F_{200}(p) \circ V(p)$$
$$F_i(p) \circ V(p) = (1 - \alpha)V(p) + \alpha\ k_{H_i} \nabla H_i^{\perp}(p) \qquad \alpha = ||\nabla H_i(p)||$$

We note that the wind projection model, while physically based, is quite simplified compared to other methods. While not the direct focus of our work, we later discuss how this is an important direction for future work.

Next we calculate wind shadowing, representing the effect of terrain on slowing the projected wind field and increasing the likelihood of deposition or sand retention. Starting at the target cell $p = (x, y)$, we march upwind a max distance, $R_s = 10m$, tracking the cell $q = (m, n)$ with the maximum terrain height difference from the target cell. Finally, we calculate the angle between the target cell and this max cell as $tan\ \alpha = (H(q) - H(p))/(||p - q||)$. The wind shadow is simply computed as $S(x, y) = (\alpha - \theta_{min})/(\theta_{max} - \theta_{min})$ where $\theta_{min} = 10$ degrees and $\theta_{max} = 15$ degrees, and S is a grid storing the wind shadow.

3.1.3 Sticky Mask Generation. A new, novel pass presented in this work, comes next in the pipeline. This pass extends the previous method to correctly model echo dunes, dunes that form a short distance away from obstacles on their wind side. We design our sticky mask generation to closely match the conclusions of measured, wind tunnel simulations of echo dunes [Tsoar 1983].

As the wind blows across a terrain with an increasing slope, that wind begins to increase in speed. With a terrain slope of fewer than 55 degrees, sand dunes carried along by the wind will climb up this sloped terrain, forming climbing dunes. However, if the slope of the terrain increases past 55 degrees, the wind begins to behave differently. Wind streams at a higher altitude remain fairly normal, continuing up and over the obstacle. In contrast, those wind streams that are located closer to the terrain itself instead bend back due to their collision with the obstacle, forming a reverse eddy current. As sand blows toward the obstacle of height h_o, a sand dune begins to form within the distance range of $2h_o$ to $0.4h_o$ away from the obstacle where the reverse eddy and initial incoming wind collide [Tsoar 1983]. Over time, the magnitudes of the two wind currents equalize, and an echo sand dune forms and stabilizes within this region. Additionally, within a distance of $0.4h_o$ from the cliff face, no sand is found to be deposited, instead quickly being removed by strong erosion due to formed eddy currents.

To capture this behavior, we present an augmentation of the existing desertscapes work called sticky cells. In this extension, we iterate over all cells, stepping back against the wind direction to check the neighboring cell for a steep, i.e. greater than 55-degree, dropoff that signals an obstacle that is steep enough to trigger the reverse eddy currents. We refer to these identified cells as cliff cells.

For these detected cliff cells, we first compute the height of the cliff h_o as the differential between the cliff cell and its closest neighbor against the wind. Additionally, while not discussed in the wind tunnel experiment, we found that allowing cliffs of unbounded height led to incorrect-looking environments. Thus, we bound all cliff heights to a maximum cliff height h_{max} with $h_o = max(h_{max}, h_o)$, limiting the maximum distance an echo dune can form from a cliff and improving visual quality. We then continue to step back against the wind, marking cells located

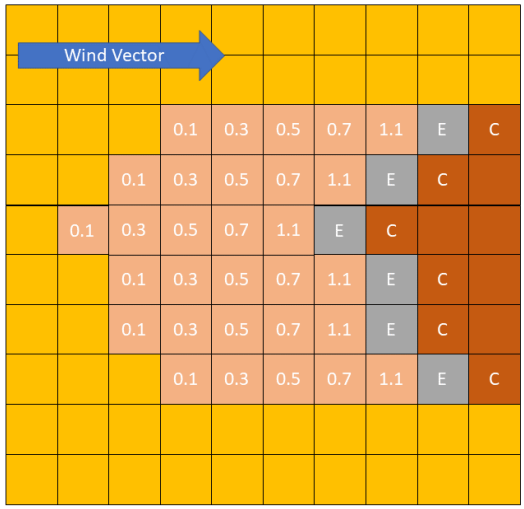


Fig. 2. An example demonstrating the generation of a sticky mask where cliff cells have a height differential of three cell widths. For the given wind vector, obstacle cells (in orange) determine cliff cells, 'c', and the cells closest to the cliff cells that are marked as erosion, 'e'. All other cells within the target distance range are marked as sticky, with their assigned value shown.

within a threshold minimum distance $0.4h_o$ and maximum distance $2h_o$ from the initial cliff cell as sticky cells; cells that should encourage deposition. These cells are marked as the inverse percentage through the range of sticky cells, with the first sticky cell being assigned $1 + k_b$ and the furthest being assigned $0 + k_b$, with k_b representing a small bias to avoid assignments of 0; we set $k_b = 0.1$. Additionally, we mark all cells located a maximum of $0.4h_o$ from the cliff cell as erosion cells; cells that experience rapid removal of sand. Figure 2 demonstrates an example sticky mask where the height differential of the cliff dunes is three cell widths high.

3.1.4 Sand Transport. Next, sand is transported based on the previously generated wind projection and shadow maps, as well as the new object and sticky cell maps. Primarily, the sand transport step models sand saltation, the lifting of sand by wind from the terrain surface, the transport of that lifted sand by wind forward through the environment, and the ultimate deposition of that sand. Additionally, the sand transport step correctly captures a number of potential rebound events, where the transported sand bounces off the terrain and continues further along, rather than being deposited. During these key bounce events, terrain sand located at the bounce position is struck and potentially moves downward along the local terrain slope in a process called reptation. While these sand transport phases remain largely unchanged from the desertscapes model, we modified the algorithm with a few key differences to support echo dune formation and object interaction [Paris et al. 2019]. Pseudocode for the sand transport algorithm is provided in Algorithm 1.

First, we modified the sand transport process to fully respect the obstacle mask, disallowing the deposition of sand in any obstacle cells. Our modified algorithm explicitly detects cases where sand that would normally be deposited would end up inside an obstacle and instead forcibly deposits the sand upwind of the obstacle. Likewise, we modify the reptation algorithm to disallow the transfer of sand into obstacle cells. Together, these modifications approximate the behavior of individual

sand grains colliding with the obstacle, and their resulting deposition after rebounding off the obstacle. Without these modifications, we find that visual artifacts consistently occur, with sand visibly accumulating inside obstacles.

Next, we modified the sand transport process to respect the sticky cell mask generated in the previous step. First, an additional check is added to each cell to prevent lifting in any cell marked as sticky. If a cell is sticky and thus has a sticky value greater than 0, sand transport from that cell is prevented with a probability of k_p (we use $k_p = 0.5$ in experiments). Additionally, for any propagating sand during saltation, if the cell at the current sand's position has a stickiness value v_k, sand is forcibly deposited in that cell with probability v_k.

Lastly, we add explicit handling of erosion cells, marked during the sticky mask generation. For any cells marked erosion, sand is forcibly removed at a rate of k_e blocks per simulation step (we use $k_e = 1000$ in experiments).

Algorithm 1 Modified Sand Transportation Algorithm

```
procedure ModifiedSandTransport
    Input : TransportParams tp, UniformRandom rand
    p ← GetCurrentCellIdx()
    Cell c_p ← AccessGrids(p)
    if c_p.obstacle > 0 || rand() < c_p.stickyMask || rand() < c_p.windShadow then return
    end if
    stepVec ← Normalize(c_p.wind) * tp.stepSize        ▷ Calculate sand step along wind vector
    Deposited ← false
    c_p.sandBlocks.Subtract(tp.transportBlockCount)    ▷ Lift sand from initial cell
    q ← p
    count ← 0
    while Deposited ≠ true || count < tp.maxSteps do
        q ← q + stepVec
        Cell c_q ← AccessGrids(q)
        if c_q.obstacle > 0 then                       ▷ Handle obstacle detection
            q ← q − stepVec                            ▷ Deposit upwind from obstacle
            Cell c_q ← AccessGrids(q)
            c_q.sandBlocks.Add(tp.transportBlockCount) return
        end if
        if rand() < c_q.stickyMask || rand() < c_q.windShadow then
            deposited ← true
            c_q.sandBlocks.Add(tp.transportBlockCount) return   ▷ Handle standard deposit
        end if
        count ← count + 1
    end while
    c_q.sandBlocks.Add(tp.transportBlockCount)         ▷ Deposit due to max distance
end procedure
```

3.1.5 Sand Cascade. The resolution of sand cascade events are not as straightforward to parallelize due to the strong interdependence between cells on a resolved sand cascade surface.

Our GPU sand cascade model must solve two problems introduced by moving to a fully parallel model on the GPU; 1) The new sand cascade model must be able to handle sand transported by all cells at once, potentially leading to a more significant amount of sand accumulating at a single point

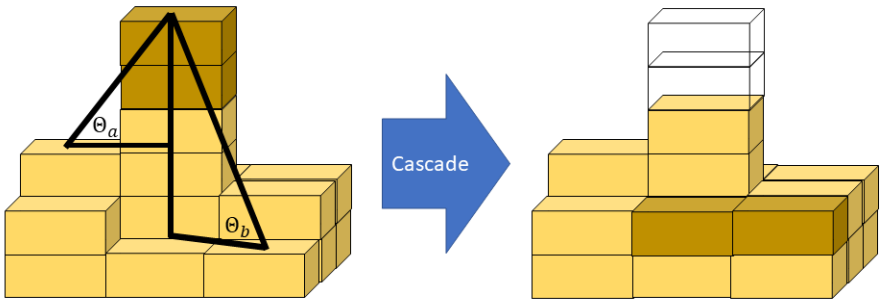


Fig. 3. A demonstration of the sand cascade computation. For a given center cell, the tangent of the angle relative to each neighboring cell is calculated. In this example, the two lowest neighboring columns are the only qualifying neighbors, thus each is assigned one of the two blocks cascading in this pass. The result can be seen on the right.

before the sand cascade step; 2) Sand cascade events are not independently resolved or able to trigger new, independent cascade events. Rather than identifying all required sand avalanching events, resolving those (potentially introducing new instabilities requiring resolution), and repeating until a stable state is reached before simulating the next transport event, we propose an iterative approach that gradually converges over the course of a fixed number of iterations; set to 50 in our case.

Our approach can be summarized as follows. For each cell, we compare that cell C_i to each of its eight neighbors C_j, identifying the angle between each the center of C_j and C_i, θ_{ji}. For all neighbors whose angle θ_{ji} is greater than the angle of repose of sand, 33°[Al-Hashemi and Al-Amoudi 2018], we calculate the tangent of their angle and sum them to form a total of the tangent angles, Θ. We then assign each of these qualifying neighbors a percentage of the total slope they are responsible for, $p_j = \frac{tan(\theta_{ji})}{\Theta}$. Additionally, we track neighboring cell C_j with the maximum tangent angle, allowing a direct computation on the maximum difference in sand blocks B_{max}. We then transport $k_c * B_{max} * p_j$ to each qualified neighboring cell C_j. In this model, k_c represents a tunable parameter to control the amount of sand allowed to avalanche during a single cascade pass. We found that setting $k_c = 0.25$ allowed for efficient simulation, converging in typically 50 iterations, while maintaining stability by preventing oscillation of sand due to high volumes of sand transfer per cascade step. Figure 3 shows a basic example of a single cascade event, visualizing two neighboring cells and their angles relative to the center cell. In this example, two sand blocks cascade to the two locations visualized on the right portion of the figure, and the blocks are removed from the center cell.

It is important to note that, due to the discrete nature of the sand block representation and constant rounding down when computing the number of blocks to transfer to a neighbor that it is possible for there to be a remaining number of blocks from the target transfer block count $k_c * B_{max}$. To account for this, we simply transfer these remaining, non-cascaded blocks to the neighbor of the greatest slope.

By design, each sand cascade iteration transfers only a subset of sand from each cell to its neighbors, improving overall stability by minimizing oscillating cascades of blocks between neighboring cells. However, when there is a significant difference in block height between neighboring cells,

a greater number of iterations must be performed to reach a converged state. In practice, the total number of required passes is highly dependent on the amount of sand transported in each simulation step, the behavior of the overall wind regime (with areas of high deposition, particularly from multiple wind directions such as star dunes requiring many passes), and the value of k_c, with more passes improving accuracy but increasing simulation time. We find that a target of 50 sand cascade passes for reasonable and common wind speeds leads to stable and fully converged sand cascading while still maintaining real-time simulation times. As wind speeds increase outside this range, an increase in the number of cascade iterations maintains stability.

3.1.6 Final Heightmap Generation. One final pass fully composites all grids in the current simulation timestep using equation 1 per cell into a final, renderable heightmap.

Lastly, a grid mesh placed in world space along the XZ expanse of the environment is rendered and the final, precomputed heightmap is sampled to obtain the height position of each vertex in world space.

4 RESULTS AND DISCUSSION

In this section, we discuss a number of experiments performed to demonstrate the validity of our GPU implementation and novel extensions to support echo dunes.

First, we show that our method is capable of generating a variety of sand dune formations at interactive rates. We compare directly to a CPU implementation of the previous Desertscapes model, provided by the authors [Paris et al. 2019], slightly modified to match their paper as their public implementation slightly differs from what their paper describes. Second, we validate the general dune simulation model by comparing the model-produced results to those of a cellular-autonoma model from the geoscience community that is treated as ground truth [Lü et al. 2018] [Rozier and Narteau 2013]. Lastly, we demonstrate the novel ability of our method to generate correct echo dunes. For this analysis, we compare the echo dunes generated by our model to those of a wind tunnel experiment [Tsoar 1983].

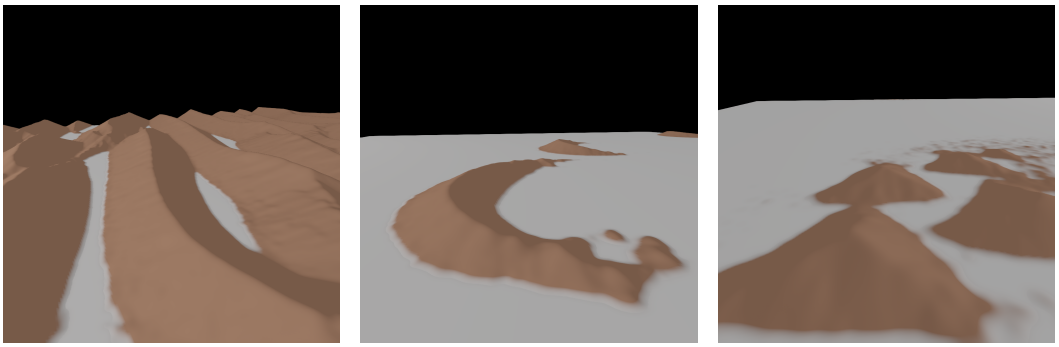
For all experiments, we utilize an implementation of our method in C++ that uses the Direct3D 11 API for GPU simulation and rendering. All experiments are performed on a machine with an i7-8700 CPU and Nvidia Titan V GPU. Unless otherwise specified, the simulated environment covers an area of 1km x 1km with a grid resolution of 1 meter.

4.1 Comparison with Desertscapes CPU Model

We compare our method with the previously published Desertscapes paper that our method builds upon [Paris et al. 2019]. We find that our presented method is significantly faster by fully leveraging GPU hardware while maintaining the same feature set and simulation capabilities. A demonstration of multiple supported types of dunes is shown in Figure 4, depicting transverse, barchan, and star dune formations generated using our GPU implementation.

We compare the performance of our efficient and parallel GPU implementation to that of the CPU, event-driven model. First, we show an environment with high wind speed and low sand availability, leading to barchan dune formation. We seed both models with the same distribution of sand, with each cell's initial sand height selected uniformly from a distribution ranging from 0.5m to 2.0m and a wind speed of $5ms^{-1}$. We perform this experiment with a number of simulation grid resolutions.

As shown in table 1, our method significantly outperforms the CPU-based method for barchan dune environments, with the GPU's performance ratio increasing with the increase in grid resolution. Additionally, our GPU implementation was trivially able to simulate a grid size of 4096, which for



(a) Transverse Dunes (b) Barchan Dunes (c) Star Dunes

Fig. 4. Demonstration of multiple dune types generated by our GPU implementation. Each is categorized by a unique sand availability and wind regime. Transverse Dune: High Availablity, Dominant Wind Direction. Barchan Dune: Low Availablity, Dominant Wind Direction. Star Dune: High Availablity, Multiple Wind Directions.

Table 1. Simulation of 1000 Barchan Environment Steps at Different Grid Resolutions

Grid Resolution	Detail Resolution	Paris 2019	Our Method	Speedup CPU/GPU
512	2m	28s	4.2s	6.7x
1024	1m	130s	8.3s	15.6x
2048	0.5m	958s	24.8s	38.6x
4096	0.25m	4656s	97.2s	47.9x

a 1km x 1km environment allows for details up to 0.25m in size, significantly outperforming the CPU which takes 1.6 hours.

Additionally, we demonstrate a similar comparison, but in an environment that generates transverse dunes. Notably, the sand generation parameters change to draw uniformly from sand heights of 3m to 5m, and the wind speed is changed to $3ms^{-1}$. With the increase in sand availability, it is easy to see that more time will be spent on sand transport and sand cascade resolution.

Table 2. Simulation of 1000 Transverse Environment Steps at Different Grid Resolutions

Grid Resolution	Detail Resolution	Paris 2019	Our Method	Speedup CPU/GPU
512	2m	91s	4.2s	21.7x
1024	1m	346s	8.3s	41.5x
2048	0.5m	2280s	24.8s	91.9x
4096	0.25m	13087s	121.4s	107.8x

As shown in table 2, our method again outperforms the CPU-based method in transverse dune environments, with the GPU's performance ratio again increasing with the increase in grid resolution. Additionally, it is important to note that our GPU implementation maintains consistent performance despite the change in environment, only noticeably differing from the barchan environment at a grid resolution of 4096. However, while our GPU implementation computation time increased only 26 seconds, the CPU performed significantly worse in the transverse dune environment, with a total time of 3.6 hours for a grid resolution of 4096.

It's clear that our GPU method, while maintaining the same features and generated behavior of the previously presented work, significantly outperforms the CPU event-based model.

4.2 Comparison with Cellular-Autonoma

Next, we validate the correctness of our model by comparing its behavior in a bidirectional wind regime against a model we consider ground truth. From the geoscience community, we select a work based on the offline, cellular autonoma approach to simulating sand transport [Lü et al. 2018; Rozier and Narteau 2013]. In particular, our selected work explores the behavior of the cellular autonoma model in a bidirectional wind regime under a number of different parameterizations; notably, the angle between wind vectors, the ratio of time spent between wind vectors, and the amount of sand available within the environment.

To conduct this experiment, we simulate a 1km x 1km environment driven by two dominant wind vectors. In the experiment, we simulate the evolution of generated sand dunes in the environment as we vary the relative angle and the availability of sand in the environment, comparing our generated results to that of the ground truth model. We match the wind vectors of the related work directly by aligning the dominant wind vector with the x-axis, and setting the secondary wind vector as a rotation in the terrain simulation plane by θ degrees. To model sand availability, we add a small, circular source of sand in each simulation step, randomly placed around the environment, until the amount of sand coverage in the environment closely matches the target sand availability. Lastly, we set the ratio of the time spent influenced by the dominant wind vector vs the secondary wind vector to 2, as we compare directly against the results found in related work [Lü et al. 2018].

As seen in Figure 5, our method clearly recreates equivalent environments to the results produced by the cellular autonoma analysis paper [Lü et al. 2018]; for this analysis, please refer to Figure 3 of the referenced work as we do not republish those results here.

For full sand availability, $\phi = 100\%$, our method generates resulting dunes that are nearly indistinguishable from those found in the referenced work. In particular, we find that both methods generate transverse dunes whose extents are largely perpendicular to the resulting wind vector direction when accounting for the relative time spent influenced by either dominant wind vector. As sand availability decreases, we find that our method continues to generate dunes very similar to those found in the referenced work. At a sand availability of $\phi = 64\%$, our method simulates transverse dunes with additional spacing between them. Once sand availability drops low enough, $\phi = 22\%$, generated dunes are barchan in shape. Additionally, as the relative wind angle increases to $\theta = 150^{\circ}$, the barchan dunes elongate along one horn, forming the same railroad spike shape found in the related work.

In addition, we find that for environments of extremely low sand availability, less than 7%, our method also produces equivalent results to those found in the related work. As seen in Figure 6, we reproduce the wind regime found in the lower left of figure 3 in the reference work [Lü et al. 2018], setting the wind angle to 45 degrees and a sand availability of 7%. Like the reference work, our method clearly generates a continuous stream of independent barchan dune formations extending from the source sand along the wind vector. While we do note additional noise in our image between the formed dunes, this has little impact on the large structure formation.

We believe this comparison further verifies the correctness of this model for use in the real-time graphics domain, with our model reproducing the large-scale, key structures in a wide variety of environments at fully interactive speeds.

4.3 Echo Dunes Correctness Test

For our last experiment, we demonstrate the ability of our novel method to generate echo dunes when interacting with obstacles.

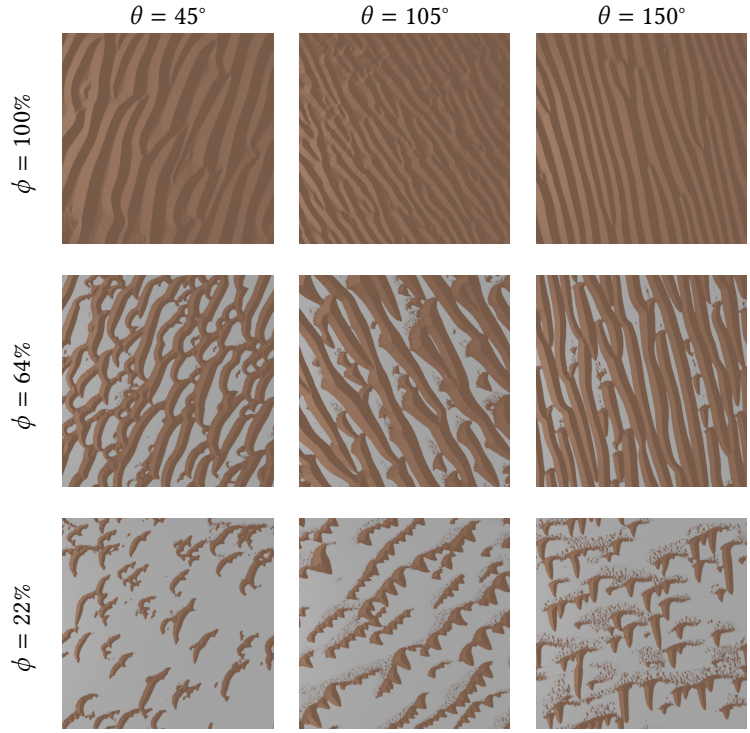


Fig. 5. Dune formation under different wind conditions. Results seen are very similar to those in figure 3 of related work [Lü et al. 2018]. θ represents sand availability from 0 to 100% of coverage in the environment. ϕ represents the angle between the two wind vectors.

.

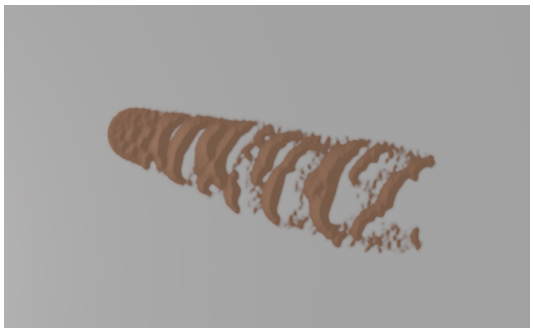
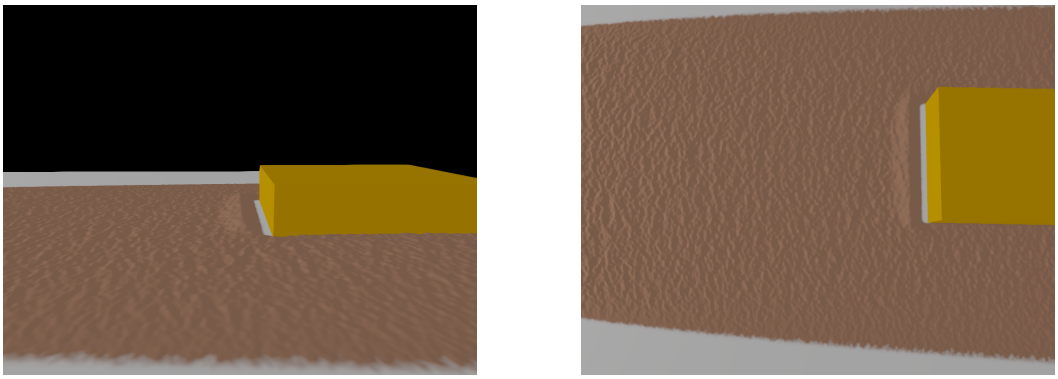


Fig. 6. A bidirectional wind regime, with an angle of 45° and a sand availability of 7%

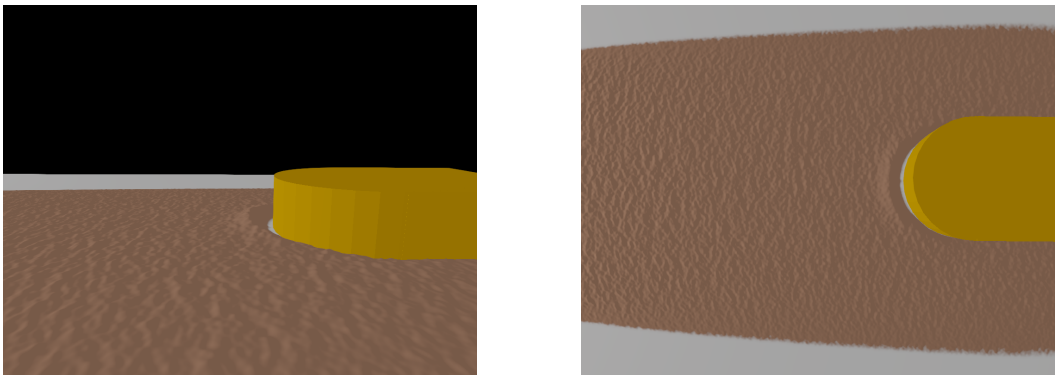
For this experiment, we simulate a wind tunnel environment consisting of a flat, non-sanded ground surface placed along the direction of the wind and an obstacle of significant height compared to the sand placed downwind. During the simulation, we allow sand to enter the scene at the other end of the ground, upwind from the obstacle, simulating virtually a previously published wind tunnel exploration of echo dune formation [Tsoar 1983].



(a) Side View (b) Top View

Fig. 7. Example of echo dune formation in wind tunnel environment with a flat obstacle placed downwind.

In the first environment, we place a flat obstacle downwind from the sand. As seen in Figure 7, as the transported sand reaches the obstacle, an echo dune forms on the wind-facing side of the obstacle. This echo dune is correctly shaped, leaning toward the obstacle with a longer, wind-side face and a shorter, obstacle-side face. Additionally, no sand is found between the echo dune and the obstacle, largely due to our erosion cells. It's important to note that sand transport is unaffected when the transporting wind does not interact with the obstacle, and thus sand to either side of the obstacle is transported as normal.

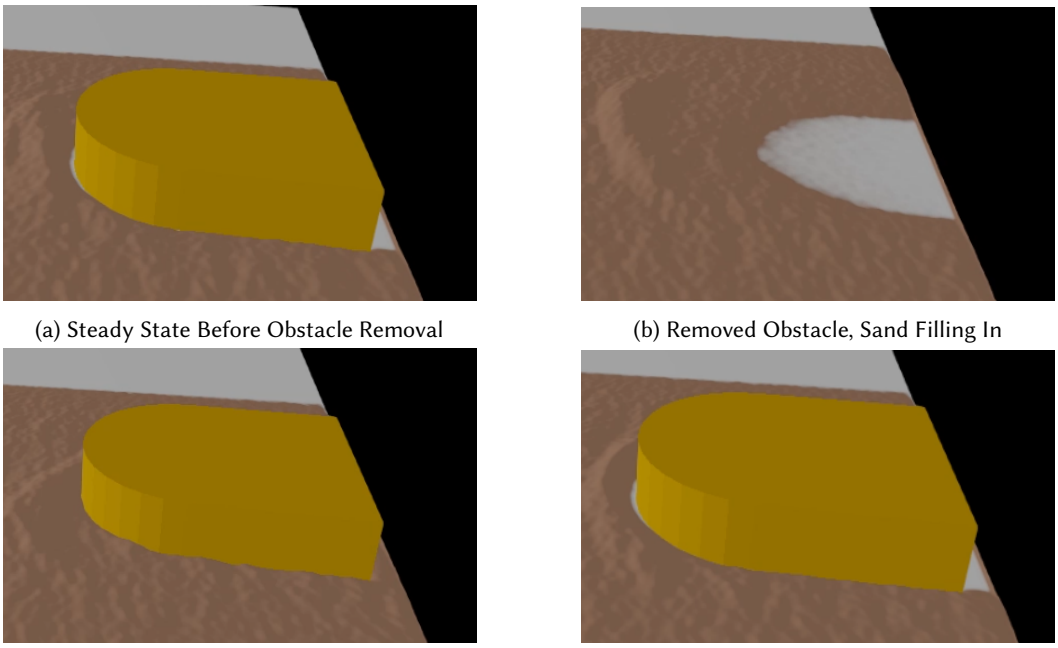


(a) Side View (b) Top View

Fig. 8. Example of echo dune formation in wind tunnel environment with a rounded obstacle placed downwind.

In a second environment, we demonstrate the same echo dune formation but with a rounded obstacle. As expected, with a rounded obstacle, the generated echo dune is most prominent when the normal of the obstacle's surface and the wind direction align, with the echo dune reducing in strength until falling off altogether when reaching the side of the obstacle. This behavior can be clearly seen in Figure 8.

Lastly, Figure 9 shows a timelapse of echo dunes in an environment where an obstacle is removed and then replaced. First, the steady state of a placed obstacle is shown. Next, a frame after the removal of the object is shown, where sand begins propagating in the region once covered by the



(a) Steady State Before Obstacle Removal

(b) Removed Obstacle, Sand Filling In

(c) Replaced Obstacle, Echo Dune Forming

(d) Steady State Reached

Fig. 9. Timeline of Echo Dune Formation during Removal and Replacement of Obstacle

object. We then show a frame just after the obstacle is placed, but before the steady state is reached again. Note how the echo dune is forming, but not quite steady. Lastly, we show the reached steady state after a few more steps of the simulation.

5 CONCLUSION AND FUTURE WORK

In conclusion, we present a novel extension of the desertscapes sand transport simulation model to the GPU, bringing sand dune simulation fully to the real-time domain for artist use. Additionally, we demonstrate the ability of our method to generate equivalent sand dune steady states to those found in offline, more extensive computation domains [Lü et al. 2018]. Lastly, we introduce a novel extension of the method to generate echo dunes on the wind-facing side of cliff faces that matches wind tunnel experiments [Tsoar 1983].

5.1 Limitations

While useful, our presented method does have its limitations. Most importantly, the method is limited in the total area that can be simulated. Currently, the model allows for only a single grid, which must be scaled proportionally to the size of the environment desired for simulation, resulting in the unfortunate tradeoff between computation complexity, and overall resolution achievable due to the selected grid resolution. Additionally, we assume a square grid, so environments such as a wind tunnel waste a significant amount of simulation time and area on grid cells that never contain sand.

Secondly, we find that our method struggles to correctly model, low sand availability environments $\phi = 7\%$ once the bidirectional wind angle increases above 65 degrees. We hypothesize that this is largely due to a failure of the wind projection model to correctly capture complex behavior

that leads to the merging of barchan dunes into a single, elongated linear dune in these low sand availability environments.

Lastly, while in this work we present an extension that correctly models echo dunes wind side of obstacles, correct modeling of the large-scale lee behavior of obstacles remains an open problem.

5.2 Future Work

One direction of future work would be to extend our presented method to tile across multiple simulation grids. By allowing sand to transfer across simulation grid boundaries, the ability to simulate across multiple grids would allow for the simulation of a significantly larger overall terrain without limiting the resolution of each simulation grid. In a similar vein, relaxing the square grid constraint would also help allocate cells to the most important parts of the grid.

A second direction of future work would be to explore the application of the presented method to other biomes whose evolution is similarly driven by aeolian processes. For example, our approach could be applied to simulate the transport of snow in arctic or antarctic environments. Additionally, the method could be used to model and study the formation and evolution of desertscape structures on other planets with dominant aeolian processes, such as Mars.

One last direction of future work would be to use machine learning to enhance the capabilities and/or performance of our model. In one approach, machine learning could be used to learn and fine-tune the parameters of our simulation for a specific environment, reducing the amount of artist interaction and automating more of the process. In a different approach, machine learning could be used to generate inputs to the physical simulation based on a rough, low-detail artist specification of a desired final output. A last, promising direction of work could utilize neural networks to bypass parts of, or all of, the simulation directly. For a specified amount of time, this neural network could take the place of the simulation, directly predicting what the evolution of the environment would look like due to aeolian processes; possibly even more complex than those presented in this work. In total, these techniques have the potential to improve how artists or other users interact with the presented simulation technique.

REFERENCES

Hamzah M. Beakawi Al-Hashemi and Omar S. Baghabra Al-Amoudi. 2018. A review on the angle of repose of granular materials. *Powder Technology* 330 (may 2018), 397–417. https://doi.org/10.1016/j.powtec.2018.02.003

Panagiotis Athanasiou, Ap van Dongeren, Alessio Giardino, Michalis Vousdoukas, Jose A. A. Antolinez, and Roshanka Ranasinghe. 2022. Estimating dune erosion at the regional scale using a meta-model based on Neural Networks. (mar 2022). https://doi.org/10.5194/nhess-2022-106

R. A. Bagnold. 1971. *The Physics of Blown Sand and Desert Dunes.* Springer Netherlands. https://doi.org/10.1007/978-94-009-5682-7

Bedrich Benes and Toney Roa. 2004. Simulating Desert Scenery. *International Conference in Central Europe on Computer Graphics and Visualization* (2004).

Guillaume Cordonnier, Eric Galin, James Gain, Bedrich Benes, Eric Guérin, Adrien Peytavie, and Marie-Paule Cani. 2017. Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Transactions on Graphics* 36, 4 (jul 2017), 1–12. https://doi.org/10.1145/3072959.3073667

Xin Gao, Clément Narteau, Olivier Rozier, and Sylvain Courrech du Pont. 2015. Phase diagrams of dune shape and orientation depending on sand availability. *Scientific Reports* 5, 1 (sep 2015). https://doi.org/10.1038/srep14677

Xinghui Huo, Hongchao Dun, Ning Huang, and Jie Zhang. 2021. 3D Direct Numerical Simulation on the Emergence and Development of Aeolian Sand Ripples. *Frontiers in Physics* 9 (jun 2021). https://doi.org/10.3389/fphy.2021.662389

Kelly Kochanski, Gian-Carlo Defazio, Eric Green, Richard Barnes, Carlos Downie, Adam Rubin, and Barry Rountree. 2019a. Rescal-snow: Simulating snow dunes with cellular automata. *Journal of Open Source Software* 4, 42 (oct 2019), 1699. https://doi.org/10.21105/joss.01699

Kelly Kochanski, Divya Mohan, Jenna Horrall, Barry Rountree, and Ghaleb Abdulla. 2019b. Deep learning predictions of sand dune migration. (Dec. 2019). arXiv:1912.10798 [cs.LG]

Sen Li, Xianwan Liu, Huichuan Li, Yinghua Zheng, and Xinghu Wei. 2007. A wind tunnel simulation of the dynamic processes involved in sand dune formation on the western coast of Hainan Island. *Journal of Geographical Sciences* 17, 4 (oct 2007), 453–468. https://doi.org/10.1007/s11442-007-0453-7

Ping Lü, Zhibao Dong, and Olivier Rozier. 2018. The Combined Effect of Sediment Availability and Wind Regime on the Morphology of Aeolian Sand Dunes. *Journal of Geophysical Research: Earth Surface* 123, 11 (nov 2018), 2878–2886. https://doi.org/10.1029/2017jf004361

Benoit B. Mandelbrot and John A. Wheeler. 1983. The Fractal Geometry of Nature. *American Journal of Physics* 51, 3 (mar 1983), 286–287. https://doi.org/10.1119/1.13295

F. K. Musgrave, C. E. Kolb, and R. S. Mace. 1989. The synthesis and rendering of eroded fractal terrains. *ACM SIGGRAPH Computer Graphics* 23, 3 (jul 1989), 41–50. https://doi.org/10.1145/74334.74337

A. Paris, A. Peytavie, E. Guérin, O. Argudo, and E. Galin. 2019. Desertscape Simulation. *Computer Graphics Forum* 38, 7 (oct 2019), 47–55. https://doi.org/10.1111/cgf.13815

Olivier Rozier and Clément Narteau. 2013. A real-space cellular automaton laboratory. *Earth Surface Processes and Landforms* 39, 1 (oct 2013), 98–109. https://doi.org/10.1002/esp.3479

Yagya Sharma, Mayank Dave, Tarun Gehlot, and Deepanshu Solanki. 2022. Neural network model to predict strength parameters of dune sand at Jodhpur City. *Materials Today: Proceedings* 62 (2022), 4498–4503. https://doi.org/10.1016/j.matpr.2022.04.945

Haim Tsoar. 1983. Wind Tunnel Modeling of Echo and Climbing Dunes. In *Eolian Sediments and Processes*. Elsevier, 247–259. https://doi.org/10.1016/s0070-4571(08)70798-2

Ning Wang and Bao-Gang Hu. 2009. Aeolian Sand Movement and Interacting with Vegetation: A GPU Based Simulation and Visualization Method. In *2009 Third International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*. IEEE. https://doi.org/10.1109/pma.2009.14

Ning Wang and Bao-Gang Hu. 2012. Real-Time Simulation of Aeolian Sand Movement and Sand Ripple Evolution: A Method Based on the Physics of Blown Sand. *Journal of Computer Science and Technology* 27, 1 (jan 2012), 135–146. https://doi.org/10.1007/s11390-012-1212-5

D. Zhang, C. Narteau, and O. Rozier. 2010. Morphodynamics of barchan and transverse dunes using a cellular automaton model. *Journal of Geophysical Research* 115, F3 (sep 2010). https://doi.org/10.1029/2009jf001620

Yang Zhang, Yuan Wang, and Pan Jia. 2013. Evolution of downsized crescent-shaped dune in wind tunnel experiment. *Science China Physics, Mechanics and Astronomy* 57, 1 (sep 2013), 143–151. https://doi.org/10.1007/s11433-013-5261-8