weBIGeo: Interaktive Lawinensimulation im Web

Patrick Komon, Gerald Kimmersdorfer, Markus Rampp, Paula Spannring, Felix Oesterle, Jan-Thomas Fischer, Adam Celarek, Manuela Waldner

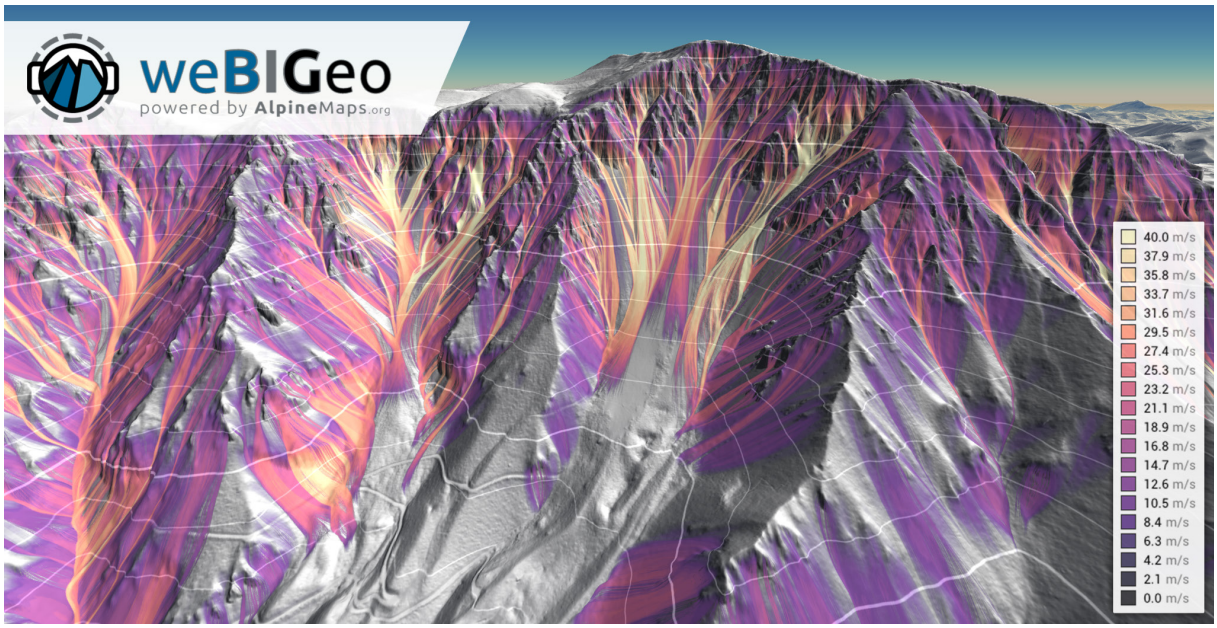

Abbildung 1: Wie suchen 3,3 Millionen Schneepartikel den Weg ins Tal? Mit weBIGeo machen wir solche Simulationen interaktiv möglich. Die Abbildung zeigt die Maximalgeschwindigkeit dieser Partikel entlang der gesamten Ostseite des Schneebergs. Die Simulation und Visualisierung dauern auf einem Mittelklasse-Rechner etwa 5 Sekunden.

Einleitung

Um potentielle Auslauflängen und Intensitäten von Lawinen zu berechnen, werden häufig tiefengemittelte Modelle, also solche, die Massebewegung entlang der Oberfläche des 3D Geländes beschreiben, verwendet. Solche physikalisch-basierten Modelle sind in operationellen Simulationswerkzeugen implementiert [Tonnel et al., 2023] und benötigen viele Eingabewerte, die in der Praxis meist nicht exakt erhoben werden können [Horton et al., 2013]. Lawinensimulationen für größere Regionen verlassen sich daher meist auf konzeptionelle, bzw. empirische Modelle, die für eine Annäherung der Gefahrenlage ausreichend sind [Hürlimann et al., 2008]. Empirische Lawinenmodelle basieren meist auf wenigen Eingabeparametern und liefern mit vereinfachten Annahmen rasch räumliche Abschätzungen zu Reichweite und Intensität, ohne den detaillierten Bewegungsverlauf in Raum und Zeit zu berechnen [D'Amboise et al., 2022]. Trotzdem sind auch empirische Modelle sensitiv gegenüber den Eingabewerten und benötigen sorgfältige Parameterwahl [Hürlimann et al., 2008]. Weder physikalisch-basierte noch empirische Modelle sind aktuell schnell genug für interaktive Systeme.

Wir stellen unser web-basiertes Tool weBIGeo vor, mit dem es erstmals möglich ist, großräumige Lawinensimulationen so effizient auszuführen, dass Eingabewerte interaktiv verändert werden können. weBIGeo basiert auf WebGPU, einer neuen Technologie, die es auch Webseiten ermöglicht, allgemeine Berechnungen auf der Grafikkarte des Endgerätes durchzuführen. Damit ist es möglich, innerhalb einer Region von mehreren hundert Quadratkilometern eine große Zahl von Lawinensimulationen parallel zu rechnen, um so eine Gefahrenkarte zu approximieren. Die Berechnung der Simulationen dauert – abhängig von der Grafikkarte des Endgerätes und der Eingabeparameter – wenige Millisekunden bis Sekunden. Das Ergebnis wird direkt auf einer 3D Geländekarte dargestellt und kann interaktiv untersucht und angepasst werden.

Technische Grundlagen

Heutzutage verfügt der überwiegende Großteil von Endgeräten über Grafikprozessoren (GPUs). Dabei handelt es sich um spezielle Hardware, die über hunderte bis tausende Berechnungseinheiten verfügt. Besonders für Programme, bei denen extrem viele, gleichartige Berechnungen durchgeführt werden, kann durch die Verwendung der GPU die Rechenzeit auf einen Bruchteil reduziert werden.

Bis vor Kurzem war für Webseiten die Nutzung der GPU nur über die Schnittstelle WebGL möglich. WebGL ist vor allem auf die Durchführung lokaler Berechnungen mit 3D-Geometrie ausgelegt, wie beispielsweise Schattierung bei bestimmten Lichteinfall. Auch das Berechnen der lokalen Steilheit an einem sichtbaren Punkt in einem 3D Geländemodell ist damit möglich. Berechnungen, die eine größere Region des Geländemodells mit einbeziehen, können nur über ineffiziente Umwege implementiert werden.

WebGPU ist eine neue Schnittstelle für GPU-Programmierung und ermöglicht die Implementierung beliebiger, paralleler Berechnungen in sogenannten “Compute Shader”. Das sind kleine Programme, die tausendfach bis millionenfach gleichzeitig auf der GPU ausgeführt werden. Dadurch ergeben sich viele neue Anwendungsmöglichkeiten, wie beispielsweise die Durchführung aufwendiger Simulationen direkt im Browser auf dem lokalen Endgerät.

Unser Projekt weBIGeo basiert auf AlpineMaps.org, einem Framework für das Rendern hochauflösender 3D-Karten, und ist eine Plattform welche es ermöglicht große geographische Daten mit Hilfe von WebGPU Compute Shaders auszuwerten und darzustellen. Der Source-Code inklusive technischer Dokumentation ist öffentlich auf GitHub https://github.com/weBIGeo/webigeo einsehbar. Eine Online-Demo ist unter https://webigeo.alpinemaps.org/ verfügbar. Da es sich bei WebGPU um eine neue Technologie handelt, ist die Verfügbarkeit derzeit noch eingeschränkt. Allgemein wird WebGPU derzeit vor allem von Chromium-basierten Browsern wie Google Chrome oder Microsoft Edge am Desktop unterstützt, während die Integration in andere Browser wie Safari und Firefox noch in Entwicklung ist.

Lawinensimulation auf der Grafikkarte

Grundlegend können nur jene Algorithmen durch die GPU schnell abgearbeitet werden, die parallelisierbar sind. Dafür muss es möglich sein, die Berechnung in einzelne Teile aufzuteilen, die voneinander unabhängig sind und daher gleichzeitig verarbeitet werden können. Schlussendlich werden die Einzelergebnisse wieder zu einem Gesamtergebnis zusammengefasst. Es gilt daher einen Weg zu finden, ein Modell zur Simulation von Lawinen in einzelne unabhängige Berechnungsschritte aufzuteilen.

Physikalische Modelle versuchen, die zugrunde liegenden physikalischen Gesetze möglichst genau abzubilden. Einige dieser Modelle, wie AvaFrame Com1DFA [Tonnel et al., 2023], modellieren Schneemassen als numerische Partikel. Diese müssen ausreichend klein und zahlreich sein, um eine Lawinenbahn mit zufriedenstellender Genauigkeit simulieren zu können. Die Partikel können miteinander interagieren und sind damit nicht unabhängig. Das macht die Parallelisierung zwar nicht unmöglich, aber langsam. Daher sind physikalische Modelle zu rechenintensiv für großräumige, interaktive Lawinensimulationen.

Datenbasiert-empirische Modelle, wie etwa FlowPy [D'Amboise et al., 2022], verwenden vergleichsweise einfache geometrische Bewegungsregeln. Diese beschreiben, mit welcher Wahrscheinlichkeit Schneemassen von einer Rasterzelle des digitalen Höhenmodells (Digital Elevation Model, DEM) in ihre Nachbarzellen fließen. Eine Zelle repräsentiert dabei typischerweise etwa 5x5 Meter [Horton et al., 2013]. Da sich, ausgehend von einer einzelnen Startzelle, der Fluss schrittweise auf immer mehr Zellen ausbreitet, lässt sich der Fluss pro Zelle nicht unabhängig bestimmen. Obwohl solche Modelle einfacher sind, sind sie trotzdem nicht optimal für die GPU.

Um eine effiziente Parallelisierbarkeit zu erreichen, wenden wir daher eine Monte Carlo Simulationsmethode an. Das Monte Carlo Prinzip besagt, dass das Verhalten eines Modells durch einen empirischen Prozess untersucht werden kann, in dem viele Zufallsstichproben gezogen werden, deren Verhalten beobachtet wird [Mooney, 1997]. In unserem Fall modellieren wir Lawinen durch eine große Zahl von numerischen Partikeln, welche sich unabhängig voneinander auf der Geländeoberfläche bewegen. Dazu wird ein einfaches Bewegungsmodell angenommen, welches komplexe physikalische Prozesse innerhalb einer Lawine, wie z.B. turbulente und chaotische Bewegungsformen, Interaktionen der Schneepartikel oder leichte Änderungen der Schneeoberfläche, nicht direkt abbildet. Um trotzdem eine realistische Ausbreitung der Lawine anzunähern - und unter der Annahme, dass diese nicht direkt abbildbaren Prozesse einem gewissen Zufallsprinzip unterliegen - werden die berechneten Flussrichtungen in geringem Ausmaß zufällig verändert.

Wir teilen die Auslösegebiete den entsprechenden Zellen des DEM zu und setzen in jeder Zelle eine konfigurierbare Anzahl von Partikeln frei, die zufällig innerhalb der Zelle verteilt werden. Die Größe der DEM-Zellen ist einstellbar. Für jeden Partikel wird in jedem Simulationsschritt i das Haltekriterium auswertet und die Bewegungsrichtung berechnet. Für das Haltekriterium orientieren wir uns an den geometrischen Größen aus [D'Amboise et al., 2022], welche sich unter Annahme der Energieerhaltung analog für einfache Bewegungsformen herleiten lassen. Wir verwenden den Auslaufwinkel α als Eingabeparameter, der die Reibung der Lawine

approximiert und direkten Einfluss auf deren Auslauflänge und Geschwindigkeit hat. Die Lawinenintensität lässt sich damit einfach als Geschwindigkeit v aus α und dem lokalen Lawinenwinkel abschätzen.

Die aktuelle Bewegungsrichtung $\boldsymbol{r_i}$ hängt vom Normalvektor $\boldsymbol{n}$ des Geländes an der aktuellen Position, der letzten Bewegungsrichtung $\boldsymbol{r_{i-1}}$ des Partikels und der aktuellen Geschwindigkeit v ab. In jedem Schritt wird ein leicht abweichender Normalvektor $\tilde{\boldsymbol{n}}$ zufällig in einem Kegel um den tatsächlichen Normalvektor gewählt und anschließend auf die xy-Ebene projiziert. Die maximale Abweichung θ ist konfigurierbar und bestimmt damit den Grad der Zufälligkeit der Monte Carlo Methode. Je größer die maximale Abweichung, desto breiter ist die typische kegelförmige Ausbreitung der Lawine. Der Einfluss der vorherigen Bewegungsrichtung $\boldsymbol{r_{i-1}}$ wird mit einem weiteren Eingabeparameter, Persistence p definiert (siehe Abbildung 2). Damit kann die nächste Bewegungsrichtung als

$$\boldsymbol{r_i} = \text{normalize}\left(p \cdot \boldsymbol{r_{i-1}} + \frac{1-p}{v} \cdot \tilde{\boldsymbol{n}}\right)$$

ausgedrückt werden. Somit ist die Bewegungsrichtung vergleichbar mit der Richtung des physikalischen Impulses. Um die neue Position zu bestimmen, wird die Bewegungsrichtung $\boldsymbol{r_i}$ mit der Schrittweite skaliert und zur letzten Position addiert. Abbildung 2 illustriert unseren Ansatz.

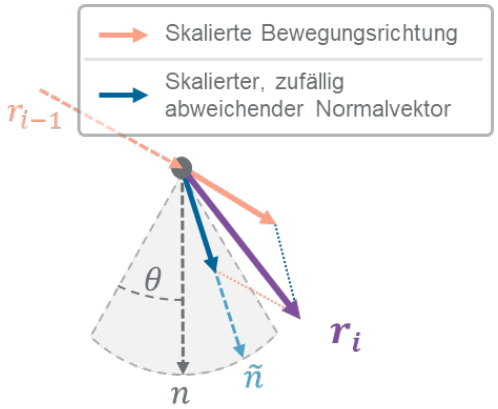

Abbildung 2: Die neue Bewegungsrichtung $\boldsymbol{r_i}$ wird in jedem Simulationsschritt durch einen zufällig abweichenden Normalvektor $\tilde{\boldsymbol{n}}$ und durch die alte Bewegungsrichtung $\boldsymbol{r_{i-1}}$ berechnet. In diesem Beispiel ist die Persistence $p = 0{,}5$, d.h. der abweichende Normalvektor $\tilde{\boldsymbol{n}}$ und die vorherige Bewegungsrichtung $\boldsymbol{r_{i-1}}$ werden gleich stark gewichtet.

In jedem Simulationsschritt werden Ausgabeparameter, wie z.B. Geschwindigkeit, Weglänge, Partikelanzahl etc., in einen Rasterlayer geschrieben. Diese Werte werden auf eine Farbpalette übertragen und direkt als Overlay in der 3D Visualisierung gerendert (siehe Abbildung 3). Ein Export für andere GIS-Anwendungen ist ebenso möglich, dauert jedoch bis zu wenigen Sekunden und benötigt daher mehr Zeit als die eigentliche Simulation.

Ergebnisse

Um die korrekte Implementierung, Funktionalität und Leistungsfähigkeit unseres Modells zu verifizieren, haben wir es auf den von AvaFrame bereitgestellten Testdatensätzen ausgeführt [Oesterle et al., 2025]. Dort enthalten sind sowohl synthetische als auch reale Szenarien. Wir vergleichen die Simulationsergebnisse mit jenen des physikalischen Modells com1DFA (v1.3) [Tonell et al., 2023] und des empirischen Modells FlowPy (v1.2) [D'Amboise et al., 2022]. Wir verwenden com1DFA mit Coulomb-Reibungsmodell mit Reibungskoeffizienten $\mu = \tan 25°$ (entspricht Auslaufwinkel $\alpha = 25°$) und ohne Krümmungeffekte. FlowPy verwenden wir mit den Standardparametern, einem Auslaufwinkel von $\alpha = 25°$ und einem Exponenten von $\exp = 8$, sowie auf einem reduzierten Höhenraster mit der für FlowPy empfohlenen Auflösung von 10x10m.

Für unser Modell haben wir die Parameter so gewählt, dass die Ergebnisse möglichst denen von com1DFA ähneln. Dabei liefern die Werte Persistenz $p = 0{,}6$, maximale Abweichung $\theta = 24°$ und Auslaufwinkel $\alpha = 25°$ die beste Übereinstimmung. Abbildung 3 zeigt Simulationsergebnisse für die Madleinlawine, die 1984 in Ischgl, Tirol abging, mit verschiedenen Simulationsparametern. Der Datensatz besteht aus einem 5x5m Höhenraster, wobei etwa 14 000 der Höhenzellen als Auslösegebiet markiert sind.

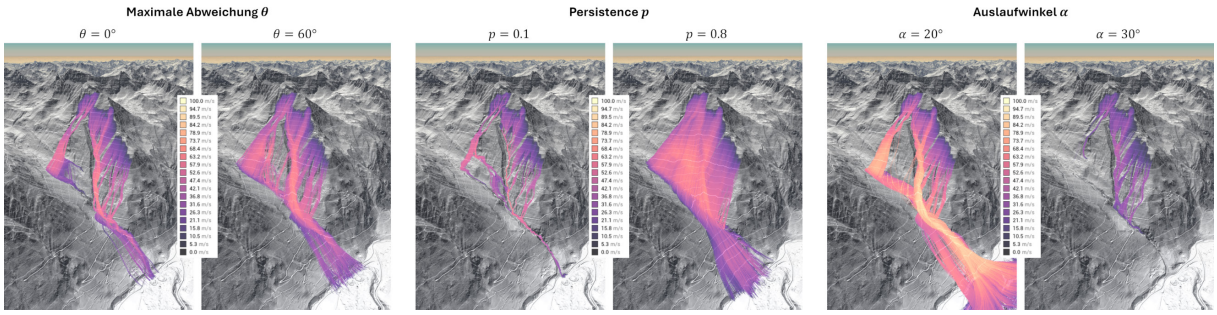

Abbildung 3: Einfluss der Eingabeparameter: Größere maximale Abweichung θ führt zu breiteren Lawinen. Hohe Persistence p führt zu geraden Lawinen. Ein kleinerer Auslaufwinkel α erhöht die Geschwindigkeit und Auslauflänge der Lawine.

In Abbildung 4 ist ein Vergleich unseres Ergebnisses mit den Umrissen der Ergebnisse von dem physikalischen Modell com1DFA und dem empirischen Modell FlowPy zu sehen. Hier wird ersichtlich, dass wir durch unsere in Echtzeit laufende Simulation beinahe deckungsgleiche Ergebnisse erzielen können. Das zeigt sich auch in weiteren Vergleichen, die unserer Dokumentation entnommen werden können: https://github.com/weBIGeo/webigeo.

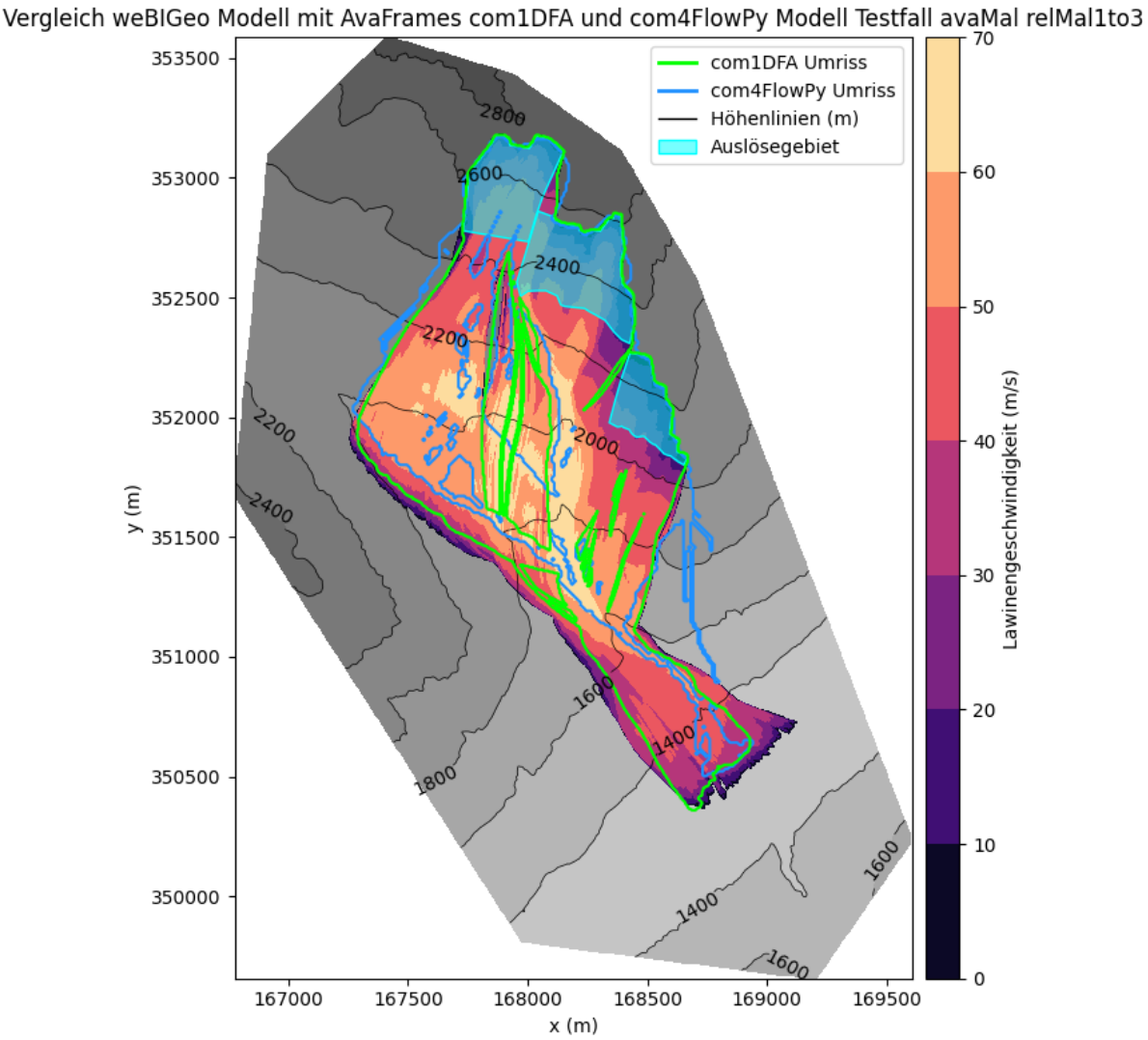

Abbildung 4: Vergleich des Ausmaßes der Lawinen: weBIGeo mit Parametern $p = 0{,}6$, $\theta = 24°$, $\alpha = 25°$, 8 Partikel pro Startzelle, Schritteweite von 2m, auf 5x5m DEM. com1DFA mit Coulomb-Reibungsmodell mit Reibungskoeffizienten $\mu = \tan 25°$ (entspricht $\alpha = 25°$), ohne Krümmungseffekte und ohne Entrainment auf 5x5m DEM. com4FlowPy mit Standardparametern auf 10x10m DEM.

Unsere Simulation mit etwa 117 000 numerischen Partikeln (8 pro Höhenzelle in Auslösegebiet) benötigt etwa 300ms auf einem Laptop mit integrierter Grafikkarte (Intel Iris Xe Graphics) und etwa 10ms auf einem Desktop-Rechner mit High-End-Grafikkarte (NVIDIA GeForce RTX 3070 Ti). Auf diesem Desktop-Rechner wurden die Ergebnisse für com1DFA in etwa 4min und für FlowPy in etwa 8min berechnet.

Diskussion

weBIGeo kann durch die Verwendung einer Monte Carlo Simulationsmethode auf der GPU großräumige Lawinensimulationen hundert- bis tausendfach schneller berechnen als aktuelle Implementierungen von State-of-the-Art Modellen. Die Ergebnisse können dabei direkt im Browser auf einer interaktiven 3D Karte inspiziert werden. Die Simulation kann selbst auf

handelsüblichen Laptops mit sehr kurzer Latenzzeit durchgeführt werden. Durch die Konfigurierbarkeit der räumlichen Auflösung des Geländemodells und der Anzahl der Partikel kann auch der Trade-Off zwischen Genauigkeit und Rechenaufwand kontrolliert werden.

Naturgemäß wird ein vereinfachtes Modell nicht deckungsgleich mit dem physikalischen sein, da wir mit unserem Ansatz keine physikalisch korrekte Interaktion zwischen den Partiklen simulieren können. Weil unser Reibungsmodell nur von der Lauflänge abhängig ist, sind unsere simulierten Lawinen tendenziell zu schnell und laufen zu weit. Dieses Problem könnte in Zukuft gelöst werden, indem im Reibungsmodell die maximale Geschwindigkeit beschränkt wird, ähnlich wie bei einer turbulenten Reibung. Im Gegensatz zu rasterzellenbasierten Modellen (z.B. [D'Amboise et al., 2022]) kann es passieren, dass wenig wahrscheinliche Bahnen nie genommen werden, und daher auch nicht angezeigt werden. Je mehr Partikel in der Simulation verwendet werden, desto genauer ist das Ergebnis.

Zusammenfassend bietet weBIGeo eine gute Möglichkeit, um erste schnelle Abschätzungen bezüglich Lawinengefahren unter verschiedenen Voraussetzungen durch systematische Variation der Parameter durchzuführen. Als web-basierte Lösung können Ergebnisse auch schnell und unkompliziert geteilt werden. In Zukunft könnten die gezeigten Simulationen einen Mehrwert in der Beurteilung von Lawinengefahr bieten, z.B. um neue Werkzeuge für professionelle und sportliche Aktivitäten im Lawinengelände zu entwickeln.

Finanzierung

Die Implementierung von weBIGeo wurde durch NetIdee (Projekt 6745) unterstützt.

Quellen

[D'Amboise et al., 2022] D'Amboise, C. J., Neuhauser, M., Teich, M., Huber, A., Kofler, A., Perzl, F., ... & Fischer, J. T. (2022). Flow-Py v1. 0: a customizable, open-source simulation tool to estimate runout and intensity of gravitational mass flows. *Geoscientific Model Development*, *15*(6), 2423-2439.

[Horton et al., 2013] Horton, P., Jaboyedoff, M., Rudaz, B. E. A., & Zimmermann, M. (2013). Flow-R, a model for susceptibility mapping of debris flows and other gravitational hazards at a regional scale. *Natural hazards and earth system sciences*, *13*(4), 869-885.

[Hürlimann et al., 2008] Hürlimann, M., Rickenmann, D., Medina, V., & Bateman, A. (2008). Evaluation of approaches to calculate debris-flow parameters for hazard assessment. *Engineering Geology*, *102*(3-4), 152-163.

[Mooney, 1997] Mooney, Christopher Z. Monte carlo simulation. No. 116. Sage, 1997.

[Oesterle et al., 2025] Felix Oesterle, Anna Wirbel, Jan-Thomas Fischer, Andreas Huber, & Paula Spannring. (2025). avaframe/AvaFrame: 1.12 (1.12). Zenodo. https://doi.org/10.5281/zenodo.15294608

[Tonnel et al., 2023] Tonnel, M., Wirbel, A., Oesterle, F., & Fischer, J.-T. (2023). AvaFrame com1DFA (v1.3): A thickness-integrated computational avalanche module – theory, numerics, and testing. Geoscientific Model Development, 16(23), 7013–7035. https://doi.org/10.5194/gmd-16-7013-2023