Master Thesis

im Rahmen des
Universitätslehrganges „Geographical Information Science & Systems“
(UNIGIS MSc) am Interfakultären Fachbereich für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

„GIS basierte Geländeanalyse zur automatisierten Eruierung geeigneter Standorte für Pumpspeicherkraftwerke“

am Beispiel des Bundeslandes Vorarlberg

vorgelegt von

Ing. Govinda Gross
104481, UNIGIS MSc Jahrgang 2016

Betreuer:

Dr. Christian Neuwirth

Zur Erlangung des Grades
„Master of Science (Geographical Information Science & Systems) – MSc (GIS)”

Feldkirch, 31.12.2019

Danksagung

Für die spannenden Inhalte und die gute Betreuung während der Studienzeit möchte ich mich herzlichst bei dem gesamten UNIGIS Team bedanken.

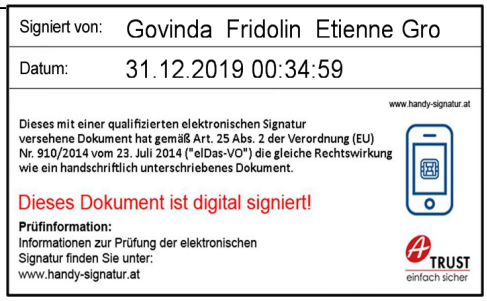
Mein besonderer Dank gilt meiner Familie, die mich in den letzten Jahren tatkräftig unterstützt hat und in der intensivsten Zeit auch vermehrt auf gemeinsame Aktivitäten verzichten musste.

Außerdem möchte ich mich bei Daniil Gefen für das ausführliche Feedback zum ersten Entwurf und für das Korrekturlesen bedanken – das war eine große Hilfe.

Erklärung der eigenständigen Abfassung der Arbeit

Ich versichere, diese Master Thesis ohne fremde Hilfe und ohne Verwendung anderer als der angeführten Quellen angefertigt zu haben und bestätige, dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen ist. Alle Ausführungen der Arbeit, die wörtlich oder sinngemäß übernommen wurden, sind entsprechend gekennzeichnet.

Feldkirch, 31.12.2019



Zusammenfassung

Ein Problem bei vermehrtem Einsatz erneuerbarer Energieträger wie Wind- und Solarenergie ist die Diskrepanz zwischen Energiegewinnungs- und Energiebedarfszeitpunkt – Einspeiseprofile decken sich nicht mit Lastprofilen. Mögliche Abhilfen sind Speicherlösungen, welche es erlauben große Mengen an Energie für einige Stunden/Tage zwischenzuspeichern. Pumpspeicherkraftwerke (PSKW) sind in diesem Bereich etabliert – es ist jedoch eine Herausforderung automatisiert neue geeignete Standorte zu finden.

Es gibt verschiedene Möglichkeiten neue PSKW zu errichten - diese Masterthesis fokussiert sich jedoch auf die Suche nach geeigneten Standorten für Talsperren, um neue Reservoire zu erhalten. In dieser Arbeit wurde ein Prozess entwickelt, welcher ausgehend von einem bestehenden Höhenmodell automatisiert als geeignet erachtete Standorte eruiert und eine Datenbank aus wichtigen Kennzahlen zu den gefundenen Standorten erstellt. Dies soll eine Unterstützung bei der Auswahl geeigneter Gebiete für neue Pumpspeicherkraftwerke in der Energiewirtschaft darstellen und in einer ersten Planungsphase schnell und kosteneffizient mögliche Dammstandorte aufzeigen.

Als primäre Datenquelle wurden SRTM-1 Daten verwendet und der Prozess anhand des Bundeslandes Vorarlberg untersucht sowie mittels einer alternativen Datenquelle plausibilisiert.

Schlagwörter: Pumpspeicherkraftwerk, PSKW, Talsperre, Engstelle, Geländeanalyse, SRTM-1, C#, ArcGIS Pro Add-In

Abstract

A main problem with the broader utilization of renewable energy sources like wind and solar is the time discrepancy between energy generation and demand. A possible remedy is the application of large-scale energy storage solutions that are able to store the energy for several hours/days until it is needed. Pumped hydro storage (PHS) is well established, but an automated identification of new suitable locations is a key challenge.

Although there are several possible ways to construct new PHS plants, this master's thesis focuses mainly on the identification of suitable locations for the creation of new reservoirs through placement of a dam at narrow spots in valleys. A methodology and subsequently an Add-In for ArcGIS Pro have been developed that takes an existing elevation model and automatically identifies potentially suitable locations, computes a set of key figures for each detected site and stores all results in a database. This tool is intended for the energy industry to support the initial selection of suitable regions for the creation of new PHS plants and could visualize locations of possible dam candidates in a cost-effective way.

The primary elevation data was sourced from SRTM-1 files and the methodology examined with the study area of Vorarlberg (most western province of Austria). Finally, a comparison with an alternative elevation model was made.

Keywords: Pumped Hydro Storage, PHS, Dam, Narrows, Terrain Analysis, SRTM-1, C#, ArcGIS Pro Add-In

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
DEM	Digital Elevation Model (= DHM)
DGM	Digitales Geländemodell (= DTM)
DHM	Digitales Höhenmodell (= DEM)
DTM	Digital Terrain Model (= DGM)
GDEM	Global Digital Elevation Model
GL	Gigaliter (1.000.000.000 Liter)
KL	Konturlinie (Höhenlinie)
LWKW	Laufwasserkraftwerk
MW	Megawatt (1.000.000 Watt)
MWh	Megawattstunde
PI	Punktintervall
PSKW	Pumpspeicherkraftwerk
SRTM	Shuttle Radar Topography Mission
SKW	Speicherkraftwerk
T1 – T7	Topologie 1 – 7 (Tabelle 1)
TIN	Triangulated Irregular Network
UG	Untersuchungsgebiet
UVP	Umweltverträglichkeitsprüfung

1 Einleitung

1.1 Motivation

Nicht konstante erneuerbare Energieträger wie Wind- und Solarenergie konnten in den vergangenen Jahren ein starkes Wachstum aufweisen – dies wird auch für die kommenden Jahre erwartet (IEA 2018). Diese Energieformen unterliegen jedoch systembedingt natürlichen Schwankungen, welche sich nicht mit der Nachfrage im Netz decken (Abbildung 1).

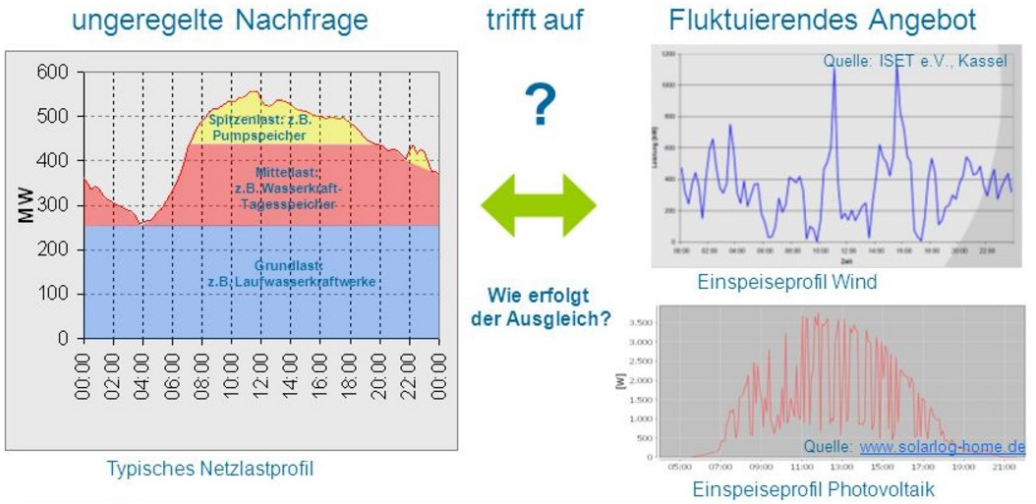


Abbildung 1: Netzlastprofil vs. Einspeiseprofil (Strebl 2012)

Um kurzzeitige Lücken überbrücken zu können und eine Unterversorgung in Spitzenzeiten zu verhindern werden Energiespeicherlösungen mit großen Kapazitäten immer wichtiger (Blarke und Lund 2008). Dies ist zudem eine integrale Voraussetzung, um zukünftig den Anteil an erneuerbarer Energie weiter steigern zu können (Bueno und Carta 2006) bzw. die Anzahl der konventionellen Grundlastkraftwerke (meist Kern- und Kohlekraftwerke) zu reduzieren.

Für diese Aufgabe bieten sich Pumpspeicherkraftwerke (PSKW) speziell dadurch an, da recht große Kapazitäten im GWh-Bereich, lange Lebensdauern/Entladezeiten sowie hohe Wirkungsgrade erreicht werden können und es sich hierbei um eine seit Jahrzenten bewährte Technologie handelt (Steffen 2012; Zakeri und Syri 2015).

Die Limitierung liegt hauptsächlich in der Anzahl der ökonomisch sowie ökologisch geeigneten Standorte (Chen et al. 2009). Die konkrete Problemstellung ist daher, das Potential von Standorten automatisiert über weite Regionen zu analysieren um vielleicht auch auf den ersten Blick nicht offensichtliche gut geeignete Standorte ausfindig machen zu können. Die Ergebnisse dieser Arbeit sind hauptsächlich für die Energiewirtschaft von potentiellem Interesse.

1.2 Literaturüberblick

1.2.1 Topologie

Im Rahmen einer Studie für das Joint Research Centre der Europäischen Kommission identifizierte der SETIS Expertenworkshop (Lacal-Arántegui und Tzimas 2012) bereits 7 mögliche Topologie-Konstellationen T1 – T7 (Tabelle 1), welche die Voraussetzungen für die Errichtung eines PSKW erfüllen können.

Tabelle 1: Kurze Beschreibung der möglichen PSKW-Topologien. Quelle: Lacal-Arántegui und Tzimas (2012)

Topology	Assessment based on:
T1	Linking two existing reservoirs with one or several penstock(s), and adding a powerhouse to transform them to a PHS scheme
T2	Transformation of one existing lake or reservoir to PHS by detecting a suitable site for a second reservoir. The second reservoir could be on a flat or non-sloping area, by digging or building shallow dams, on a depression or in a valley
T3	A greenfield PHS based on a suitable topographical context: either valleys which can be closed with a dam, depressions, hill tops which could be slashed, etc. This topology is broader i.e. neither based on existing lakes or reservoirs nor assuming a flat area for building the second reservoir
T4	Sea-based PHS: a greenfield PHS that uses the sea as the lower reservoir and a new nearby reservoir, or the sea as upper basin and a cavern as lower reservoir
T5	Multi-reservoir systems including both PHS and conventional hydropower
T6	The lower reservoir is basically a large river providing sufficient inflow into the PHS system.
T7	Use of an abandoned mine pit as the basis for the PHS. The methodology to be used would be similar to the topology 2 one.

Für Konstellationen der Kategorie T1 werden zwei bestehende Reservoire mittels Druckleitungen miteinander verbunden und durch die Ergänzung eines Kraftwerkes als PSKW genutzt. Hierbei liegen die Kriterien hauptsächlich bei einem minimal notwendigen Höhenunterschied und einer maximal praktikablen Leitungslänge bzw. Distanz zwischen den Reservoiren. Dies lässt sich sehr gut mittels GIS Methoden automatisieren und wurde beispielsweise von Gimeno-Gutiérrez und Lacal-Arántegui (2013), Lu und Wang (2017), Rogeau et al. (2017) und Soha et al. (2017) in unterschiedlichen Untersuchungsgebieten umgesetzt. Der große Vorteil dieser Variante liegt darin, dass die Kosten für die Errichtung der Reservoire entfallen und aufgrund geringerer Baumaßnahmen auch weniger Auswirkungen auf die Umwelt zu erwarten sind. T5 behandelt die Option, durch die Verbindung von bestehenden konventionellen Wasserkraftwerken mit eigenem Speichersee in unterschiedlichen Höhenlagen eine Nutzung als PSKW zu ermöglichen.

T2, T3, T4, T6 und T7 hingegen basieren darauf, dass zumindest eines der zwei benötigten Reservoire neu errichtet werden muss – entweder in Kombination mit einem bestehenden Reservoir (T2), einem zweiten neuen Reservoir (T3), dem Meer als unteren oder auch oberen Reservoir (T4), einem Fluss als unteren Reservoir (T6) oder einer verlassenen Mine als unteren Reservoir (T7).

Durch die Berücksichtigung der Möglichkeit neue Reservoire erstellen zu können, wird die Anzahl der potentiell realisierbaren neuen PSKW deutlich erhöht. So wurden beispielsweise von Gimeno-Gutiérrez und Lacal-Arántegui (2013) bei der Untersuchung des Potentials der EU-Mitgliedsstaaten sowie der Türkei für die Konstellationen T1 „nur“ 99 potentiell realisierbare neue Verbindungen (mit den in der Studie verwendeten Einschränkungen) gefunden – jedoch über 2.000 potentiell realisierbare neue Standorte für die Variante nach T2 selektiert.

1.2.2 Dammtyp

Ein weiteres Unterscheidungsmerkmal, das es uns erlaubt eine Gruppierung der vorhandenen Literatur durchzuführen, ist der Dammtyp, welcher für die Errichtung eines neuen Reservoirs in Betracht gezogen wird.

1.2.2.1 Ringdamm

Abbildung 2: Goldisthal Oberbecken (Vattenfall GmbH)

Ein Ringdamm ist eine in sich geschlossene Dammkonstruktion, welche den kompletten Umfang des Reservoirs umschließt. Für eine neue Errichtung muss der Untergrund nicht absolut flach sein, da Material eines evtl. notwendigen Aushubs bzw. Abtrags sogleich vor Ort im Bau des Dammes wiederverwendet werden kann. Studien, welche diesen Dammtyp für Ihre Untersuchungen wählten sind Connolly et al. (2010), Fitzgerald et al. (2012), Jiménez Capilla et al. (2016), Soha et al. (2017) und Lu et al. (2018).

Vorteile dieses Typs sind unter anderem die bauartbedingte hohe Versiegelung des Beckens, wodurch ungewollte Abflüsse sehr gering ausfallen, die Unabhängigkeit von engen Talformen sowie die größere Flexibilität in der exakten Standortwahl, da das gesamte Reservoir neu erstellt wird und hierdurch auch kleinräumiger kritischen Bereichen wie etwa Siedlungsgebieten, Natura 2000 Zonen etc. ausgewichen werden kann. Unklare Umweltverträglichkeit der alternativen Talsperren führten beispielsweise Gimeno-Gutiérrez und Lacal-Arántegui (2013) als Grund an, sich in ihrer Potential-Analyse über 31 Länder ausschließlich auf die Standortsuche für Reservoire mit Ringdamm zu konzentrieren.

1.2.2.2 Talsperre

Abbildung 3: Leibis Talsperre (PGSP GmbH)

Bei der Errichtung von Talsperren wird die natürliche Geländeform eines meist engen Tals als Teil des „Dammes“ integriert und lediglich ein verhältnismäßig kleines Stück über einen künstlichen Damm abgesperrt. Die Relation von potentiellem Speichervolumen zu notwendigem Dammvolumen ist bei diesem Typ deutlich höher und somit die Konstruktionskosten je Gigaliter (GL) geringer.

Ein weiterer Vorteil dieses Typs ist der natürliche Wasserzufluss, welcher durch das gesamte Wassereinzugsgebiet gespeist wird – im Gegensatz hierzu kann bei einem Ringdamm nur die tatsächlich durch den Damm umschlossene Fläche als „Einzugsgebiet“ beitragen.

Megaprojekte wie die Drei-Schluchten-Talsperre am Jangtsekiang in China haben in den letzten Jahrzehnten berechtigterweise viele Bedenken aufgeworfen. Die Zwangsumsiedlungen von Millionen von Menschen, die Gefährdung von Tier- und Pflanzenarten deren natürlicher Lebensraum durch die großflächige Flutung zerstört wird und die Sicherheitsrisiken im Falle eines Dammbruchs können in diesem Zusammenhang genannt werden. Das Ausmaß dieser Probleme ist jedoch dem enormen Umfang dieser Projekte geschuldet. Die Verwendung kleinerer Talsperren zur Flutung zuvor unbewohnter und karger Täler ist auch

heute speziell in bergigen Gebieten eine attraktive Option, wie Lu und Wang (2017) im Untersuchungsgebiet Tibet zeigten. Lu et al. (2018) berücksichtigten für ihre Studie in Australien beide Dammtypen, Petheram et al. (2017) ebenfalls in Australien exklusiv die Talsperren.

Da für die Errichtung eines Laufwasserkraftwerkes (LWKW) mit Speicherkapazität ebenfalls eine Talsperre benötigt wird, wurden in dieser Arbeit auch die Studien von Larentis et al. (2010) sowie Yi et al. (2010) in die Liste der primär relevanten Literatur aufgenommen.

1.2.3 Methodik

Da ein PSKW aus zwei Reservoiren besteht, ist die allgemein angewendete Methodik zur Eruierung geeigneter PSKW-Standorte immer, Reservoir-Paare zu finden, welche eine minimale horizontale und maximale vertikale Separation aufweisen. Im einfachsten Topologie Szenario T1 (siehe Tabelle 1) wird eine Datenbank bereits bestehender Reservoirstandorte erstellt, welche jeweils die Höhe des Wasserspiegels bei maximaler Füllung sowie die Koordinaten des Damm- oder des Reservoirmittelpunktes (Gimeno-Gutiérrez und Lacal-Arántegui 2013) enthält. Weiters ist das Fassungsvermögen von Bedeutung, da ein Reservoir mit zu geringem Volumen wirtschaftlich für den Betrieb eines PSKW nicht rentabel ist. Informationen zum Fassungsvermögen werden bei bestehenden Reservoiren grundsätzlich vom Erbauer zur Verfügung gestellt. Für neu zu errichtende künstliche Reservoire kann das maximale Volumen aus dem digitalen Höhenmodell (DHM) errechnet werden. Schwieriger gestaltet sich jedoch die Volumensschätzung bei natürlichen Seen, da hier kaum genaue Höheninformationen des Geländes unterhalb des Wasserspiegels zur Verfügung stehen. Lu und Wang (2017) arbeiteten deshalb vereinfachend mit der Annahme, dass betroffene Seen im Durchschnitt eine Tiefe von 20 m aufweisen würden. Rogeau et al. (2017) hingegen wendeten die von Hollister et al. (2011) erarbeitete Methode zur Schätzung der durchschnittlichen Seetiefe unter Berücksichtigung der umliegenden Topographie an.

Gewählte Grenzwerte:

- **Maximale horizontale Separation: 3.5 km** (Rogeau et al. 2017), **5 km** (Soha et al. 2017), **10 km** (Lu und Wang 2017), **bis zu 20 km** (Gimeno-Gutiérrez und Lacal-Arántegui 2013)
- **Minimale vertikale Separation: 10 m** (Rogeau et al. 2017), **50 m** (Soha et al. 2017; Gimeno-Gutiérrez und Lacal-Arántegui 2013), **500 m** (Lu und Wang 2017)
- **Mindestvolumen: 8.000 m³** (Rogeau et al. 2017), **100.000 km³** (Gimeno-Gutiérrez und Lacal-Arántegui 2013), **1.200.000 m³** (Lu und Wang 2017)
- **Mindestkapazität: 35 MWh** (Soha et al. 2017)

Die Grenzwerte unterscheiden sich teilweise um ein Vielfaches, was auf die unterschiedlichen Größenordnungen der einzelnen Arbeiten zurückzuführen ist. Zudem kann die Anzahl der Treffer deutlich erhöht werden, wenn die Grenzwerte nicht zu restriktiv gewählt sind. Durch eine Bewertung der einzelnen Treffer – etwa über das Verhältnis von vertikaler zu horizontaler Separation – kann abschließend eine entsprechende Reihung vorgenommen werden.

Für T2 – T7 ist die Errichtung zumindest eines neuen Reservoirs erforderlich. Die hierzu entwickelten Methoden zur automatisierten Standortbestimmung sind weiters wieder nach Dammtyp unterteilt.

1.2.3.1 Methodik Ringdamm

Gewöhnlich wird in einem ersten Schritt das zu untersuchende Gebiet durch eine Reihe von Kriterien – wie Klima, Risiko, Nähe zu Erzeugern bzw. Verbrauchern etc. – eingeschränkt, da dies im Vergleich zu den Geländeanalysen mit deutlich geringerem Rechenaufwand verbunden ist. Falls Topologien der Variante T3 nicht in Betracht gezogen werden und lediglich ein zweites Reservoir zu einem bestehenden gesucht wird, kann das Suchgebiet über ein weiteres Kriterium – Lage innerhalb eines Puffers entsprechend der maximalen horizontalen Separation zu bestehenden Reservoiren – eingeschränkt werden.

Da für die Errichtung eines Ringdammes ein ausreichend flaches Gebiet benötigt wird - - Gimeno-Gutiérrez und Lacal-Arántegui (2013) wählten hierzu eine maximale Neigung von 5%, Soha et al. (2017) hingegen maximal 7,5° (entspricht 13,2%) – wird anschließend die Geländeneigung jeweils aus den verwendeten DHM Datensätzen ermittelt.

Für die Erreichung der erforderlichen Mindestkapazität können die Parameter Dammhöhe und Reservoirfläche variiert werden. Da jedoch eine Erhöhung der Dammhöhe den erforderlichen Materialbedarf im Quadrat erhöht, eine Veränderung der Reservoirfläche sich hingegen im Idealfall nur mit Faktor $\sqrt{x}$ auswirkt (siehe 2.3 Dimensionierung eines Dammes), wird in der Literatur in diesem Fall grundsätzlich die Dammhöhe fixiert und ein Gebiet ausreichender Größe gesucht, um das notwendige Mindestvolumen fassen zu können. Annahmen für die fixierten Dammhöhen lagen bei 20 m (Fitzgerald et al. 2012; Gimeno-Gutiérrez und Lacal-Arántegui 2013) sowie 30 m (Connolly et al. 2010).

1.2.3.2 Methodik Talsperre

Zur Vorselektion der Gebiete kann hierbei die selbe Methodik wie bei den Ringdämmen angewandt werden. Bei der Geländeanalyse ist jedoch das Ziel, einen Standort für einen neuen Damm (Talsperre) ausfindig zu machen, welcher möglichst klein dimensioniert ist und dabei ein möglichst großvolumiges Reservoir erzeugt. Hierfür wird meist entlang von

Drainagelinien in gewissen Intervallen (Abbildung 4) ein Damm unterschiedlicher Höhe simuliert und die daraus entstehenden Reservoirkennzahlen errechnet. Werden Mindestgrenzwerte wie 1 GL (Lu et al. 2018) oder 5 GL (Petheram et al. 2017) Reservoirvolumen nicht erreicht, wird das Ergebnis verworfen, ansonsten mit den Kennwerten in der Datenbank abgelegt. Sobald alle potentiellen Standorte ermittelt sind, kann wiederum eine Reihung nach Eignung, Volumen, Höhenunterschieden etc. vorgenommen werden.

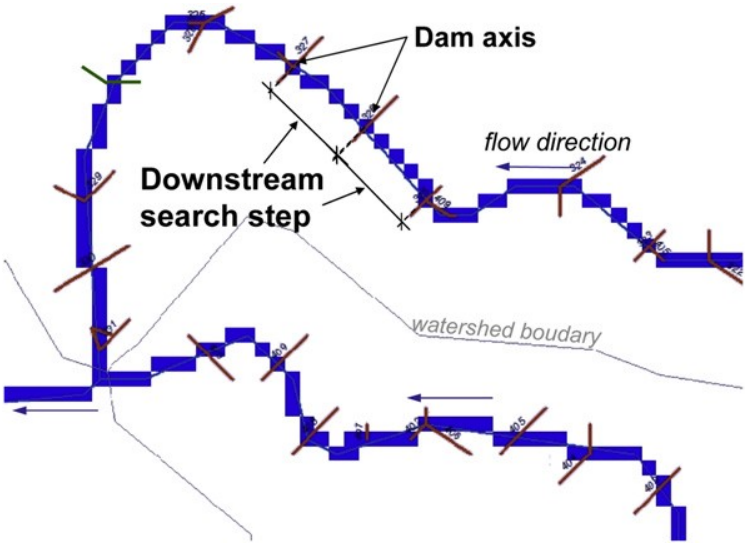


Abbildung 4: Damm-Simulationspunkte entlang eines Drainagenetzwerkes (Larentis et al. 2010)

Da der exakte Dammstandort für eine genaue Berechnung der notwendigen Dammausmaße ausschlaggebend ist, wäre eine Simulation in möglichst kleinen Intervallen – beispielsweise entsprechend der Auflösung der zugrundeliegenden Höhendaten wünschenswert. Dies würde jedoch eine sehr hohe Anzahl an Simulationspunkten ergeben und die benötigte Rechenzeit eine Analyse über größere Gebiete verhindern.

Lu und Wang (2017) definierten in ihrer Studie in Tibet die Schnittpunkte zwischen Drainagelinien und einem Mehrfachringpuffer in 500 m Intervallen um bestehende Reservoire als potentielle Dammstandorte. Lu et al. (2018) wählten hierzu in Australien die Schnittpunkte zwischen Drainagelinien und 10m Höhenlinien.

Ausgehend von den Startpunkten entlang der Drainagelinien wird anschließend entlang einer Normalen zur Drainagelinie der virtuelle Damm in jeweiliger für die Berechnung gewählten Höhe erweitert bis eine Zelle des DHM erreicht wird, welche dieselbe Höhe (Höhe des Startpunktes + Dammhöhe) aufweist. Sind auf beiden Talseiten diese Schnittpunkte mit dem DHM gefunden, entspricht die potentielle Dammlänge der Distanz zwischen diesen

beiden Punkten. Das Dammvolumen kann aus dem Höhenprofil entlang dieser Linie abgeleitet werden. Das Reservoirvolumen ist ebenfalls über das DHM berechenbar, indem die Höhendifferenz zwischen potentieller Wasserspiegel- und Rasterhöhe aller hinter dem Damm liegenden Rasterzellen des DHM mit der Fläche einer Rasterzelle multipliziert wird.

Petheram et al. (2017) gaben an eigene performantere Simulationsmethoden entwickelt zu haben, welche es erlauben würden selbst Milliarden von potentiellen Standorten mit konventionellen Computern und in angebrachter Zeit zu überprüfen. Leider war es hier jedoch nicht möglich weitere Details zu diesen Methoden in Erfahrung zu bringen.

1.2.4 Daten

Der Großteil der angeführten Studien verwendete die frei verfügbaren SRTM Datensätze in der 30 m (1 Bogensekunde) oder 90 m (3 Bogensekunden) Variante. Connolly et al. (2010) und Soha et al. (2017) griffen für ihre vergleichsweise überschaubaren Untersuchungsgebiete auf regional vorhandene 10 m DTM zurück. Gimeno-Gutiérrez und Lacal-Arántegui (2013) hingegen legten ihrer Arbeit aufgrund des großen zu untersuchenden Gebiets ein 250 m DTM zugrunde.

Da die SRTM Datensätze durch C-Band Radar (5,6 cm Wellenlänge) erstellt wurden und dieses nur schlecht in Vegetation eindringt, spiegeln die Höhenwerte bei dichter Vegetation teilweise die Höhe der Baumkronen und nicht die Höhe des Geländes wider. Die angeführten Studien, welche die SRTM Datensätze nutzten, haben diesen Umstand soweit bekannt nicht weiter berücksichtigt. Der Einfluss auf das Ergebnis wird auch für den Zweck dieser Arbeit als zu vernachlässigen eingestuft, da sich die Auswirkung – falls überhaupt zutreffend - hauptsächlich auf eine leichte Unterschätzung der Reservoirvolumen beschränkt und eine Eruierung der Engstellen nicht maßgeblich beeinträchtigt erscheint. Wie auch in der Literatur werden DHM sowie DGM (eng.: DEM und DTM) synonym verwendet.

Tabelle 2: Übersicht relevanter Literatur (modifiziert nach Lu et al. (2018))

Literatur	Typ und Topologie (Tabelle 1)	Damm-Typ	Untersuchungsgebiet	DTM/DEM Daten	Tools/Software	Methodik	Berechnungsdauer	Ergebnisse
Connolly et al. 2010	PSWK / T3	Ringdamm	800 km² in Irland	10 m DTM aus dem Ordinance Survey Irland	Atlas Computers Ltd's Survey Control Centre sowie eigener Algorithmus	DTM -> TIN; Suche nach zwei Gebieten mit akzeptabler Ebenheit, maximaler vertikaler und minimaler horizontaler Separation	6-10 Tage auf 8 PCs	5 potentiell geeignete Standorte im Untersuchungsgebiet. Limitierende Faktoren: Kosten für die Daten und Berechnungszeit
Larentis et al. 2010	LWKW	Talsperre mit max. 50 m Dammhöhe	26.500 km² in Brasilien	90 m SRTM	Hydrospot (Fortran)	Entlang von Drainagelinien wurde alle 450 m ein Testpunkt erstellt und auf mögliche Dammhöhen bis maximal 50 m und Überbrückungen von Flussschlingen geprüft.	unbekannt	274 potentielle Kraftwerksstandorte (199 reine Fließkraftwerke, 75 mit Speicherkapazität)
Yi et al. 2010	LWKW	Talsperre mit max. 20 m Dammhöhe	554 km² in Korea	30 m DEM (Datenquelle unbekannt)	ArcView mit Avenue Script	An Punkten entlang von Drainagelinien wurden Dämme mit Höhen zwischen 5 und 20 m simuliert (100 m Ringpuffer) und anhand des Verhältnisses von berechneter Speicherkapazität zu notwendiger Dammbreite bewertet.	unbekannt	4 potentiell geeignete Dammstandorte, 2 reine Fließkraftwerke
Fitzgerald et al. 2012	PSWK / T2	Ringdamm	785.000 km² in der Türkei	90 m SRTM, CORINE Land Cover	ArcGIS 9 mit ModelBuilder	Suche nach einem Gebiet mit akzeptabler Ebenheit und Größe innerhalb eines 5km Puffers um bestehende Reservoire mit mindestens 150m Höhenunterschied.	unbekannt	448 potentiell geeignete Standorte
Gimeno-Gutiérrez und Lacal-Arántegui 2013	PSKW / T1, T2	Ringdamm	30 Länder in Europa sowie die Türkei	250 m GMTED2010 (Aufgrund von Rechenperformance kein höher auflösendes DEM)	ArcGIS	Suche nach flachen Gebieten mit maximaler Neigung von 5% in einem Umkreis von bis zu 20km von bestehenden Reservoiren, wo der Bau eines zumindest 100.000 m³ fassenden zweiten Reservoirs mittels Ringdamm möglich ist und ein Höhenunterschied von zumindest 50 m besteht.	unbekannt	99 potentiell realisierbare neue Verbindungen (T1 innerhalb 5 km) sowie 2.025 potentiell realisierbare neue Standorte durch Bau eines Reservoirs mit Ringdamm (T2) innerhalb von 5 km eines bestehenden Reservoirs

Literatur	Typ und Topologie (Tabelle 1)	Damm-Typ	Untersuchungsgebiet	DTM/DEM Daten	Tools/Software	Methodik	Berechnungsdauer	Ergebnisse
Jiménez Capilla et al. 2016	PSKW / T2	Ringdamm	550 km² in Spanien	Datenbanken der Regionalregierung von Andalusien	ArcGIS 10	Durch Anwendung einer Multkriterienanalyse mit den Kriterien Gelände, Standort, Klima und Risiko wurde das Untersuchungsgebiet bewertet und letztlich drei Gebiete mit über 80% Eignung eruiert.	unbekannt	3 mögliche Standorte für ein oberes Reservoir
Pethera m et al. 2017	SKW	Talsperre	16.950 km² in Australien	30 m SRTM (corrected)	DamSite Spatial mit Python Skripten	Suche nach Dammstandorten mit beschränkten Dimensionen und einem gleichzeitig erzeugenden Reservoir mit Mindestvolumen von 5 Gigalitern, indem inkrementell entlang von Drainagelinien Dämme verschiedener Höhe „errichtet“ wurden.	unbekannt	Über 100 potentiell geeignete Standorte mit mindestens 5 Gigalitern an Volumen und 10km² Wassereinzugsgebiet
Lu und Wang 2017	PSKW / T1, T2	Talsperre	1.220.000 km² in Tibet	30 m ASTER GDEM	ArcGIS	Schnittpunkte zwischen Drainagelinien und einem Mehrfachring-Puffer (5.000 m in 500 m Intervallen) ausgehend von einem bestehenden Reservoir, wurden als potentielle Dammstandorte selektiert und nach weiteren Parametern (Distanz zu bestehendem Reservoir, Höhenunterschied, Volumen,…) gefiltert.	unbekannt	69 potentielle Verbindungen (T1) und 489 potentielle Standorte für Reservoirs mit Talsperren (T2)
Rogeau et al. 2017	PSKW / T1	- (nur bestehende Reservoire)	675.000 km² in Frankreich	25 m DEM von IGN Alti, IGN Topo,	RStudio	Es wurden bestehende Seen mit mindestens 8.000 m² Oberfläche und natürliche Senken mit mindestens 8.000 m³ Volumen berücksichtigt. Für jedes dieser möglichen Reservoire wurden alle Verbindungen zu Reservoiren innerhalb eines 3.5 km Puffers und mit mindestens 10 m Höhenunterschied selektiert.	13-20 Tage für Datenaufbereitung, ca. 1 Stunde für Selektion geeigneter Reservoir-Paare	850.000 potentielle Verbindungen
Soha et al. 2017	PSKW / T1,T2,T7	Ringdamm	1324 km² in Ungarn	10 m DTM	ArcMap 10 und Surfer 13	Suche nach Gebieten mit maximaler Neigung von 7.5° innerhalb eines 5km Puffers um bestehende Reservoire oder Minen	unbekannt	15 potentiell geeignete Standorte
Lu et al. 2018	PSKW / T2, T3	Ringdamm und Talsperre	31 km² in Australien (aus ca. 1.000.000 km² durch diverse Ausschlusskriterien gefiltert)	30 m SRTM (corrected)	STORES (Python) mit NumPy, SciPy und ArcPy	Für Talsperren: Entlang von Drainagelinien wurde jeweils beim Schnittpunkt mit einer 10m Höhenlinie ein 40m hoher virtueller Damm erzeugt und geprüft, ob die Oberfläche des dadurch entstehenden Reservoirs mindestens 10ha und das Volumen zumindest 1 GL umfasst.	unbekannt	423 Talsperren und 22 Ringdämme

1.3 Forschungsziel

Für die tatsächliche Errichtung von neuen PSKW sind eine Reihe von Kriterien zu erfüllen und Parameter zu berücksichtigen (2.4). Für die Suche nach potentiellen neuen Standorten kann hier beispielsweise mit bestehenden GIS Werkzeugen eine Multikriterienanalyse durchgeführt werden, um grundsätzlich geeignete Gebiete ausfindig zu machen (kein Schutzgebiet, Distanz zu bestehender Infrastruktur, unbewohnt,…). Diese Arbeit beschränkt sich jedoch lediglich auf die Analyse der Eignung des Geländes zur Errichtung eines neuen Reservoirs auf Basis von frei zur Verfügung stehenden Höhendaten. Andere Faktoren werden bewusst nicht berücksichtigt, da diese für die eigentliche Funktionalität eines Geländeanalysealgorithmus nicht relevant sind.

Weiters muss das genaue Ziel der Geländeanalyse definiert werden, denn für die Errichtung eines Ringdammes wird eine andere Geländeform benötigt als für die Errichtung einer Talsperre. Die Verwendung von Ringdämmen für die Errichtung neuer Reservoirs ist zwar speziell für kleinere Kapazitäten grundsätzlich aufgrund der vorhandenen Vorteile (siehe 1.2.2.1) eine gute Option, mögliche Standorte mittlerer Kapazität unter Einsatz einer Talsperre sind jedoch kosteneffizienter, da das Gelände den größten Teil der Dammfunktion übernimmt. Anhand der gefundenen Literatur ist wieder ein Anstieg des Interesses an dieser Variante zu erkennen (1.2).

Das Ziel dieser Arbeit ist es, anhand eines Untersuchungsgebietes mit der Ausdehnung des Bundeslandes Vorarlberg (2.601 km²) einen performanten Geländeanalysealgorithmus zu entwickeln um geeignete bestehende Geländeformen in Kombination mit einer Talsperre als mögliche Becken bzw. Reservoire eines PSKW identifizieren zu können. Dieser soll die möglichen Standort-Kandidaten in kleinen Intervallen (Auflösung der zugrunde liegenden Höhendaten) und mit unterschiedlichen Dammhöhen simulieren um die optimalsten Standorte – ein möglichst hohes Verhältnis von Reservoirvolumen zu Dammvolumen - ausfindig zu machen. Um eine leichte Wiederverwendbarkeit des Algorithmus zu fördern, soll die Datengrundlage hierfür durch die frei verfügbaren „Shuttle Radar Topography Mission“-Datensätze (SRTM-1) mit der Auflösung von 1 Bogensekunde (30 m am Äquator) gebildet werden. Abschließend wird ein Vergleich mit einem zusätzlich für das Testgebiet verfügbaren Datensatz mit 5 m horizontaler Auflösung angestellt und die Eignung der SRTM-Daten für die Verwendung mit dem entwickelten Algorithmus untersucht.

2 Grundlagen

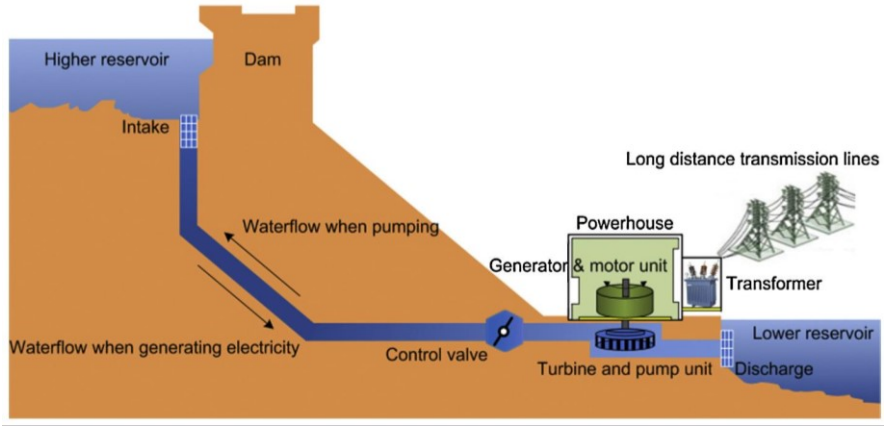


Abbildung 5: Schema eines PSKW (Luo et al. 2015)

Ein PSKW ist ein Wasserkraftwerk, welches primär zur Speicherung von Energie verwendet wird. Hierzu werden zwei Reservoire unterschiedlicher Höhenlage benötigt - das obere Speicherbecken sowie das untere Tiefbecken (Abbildung 5). Diese werden mit einer Druckleitung verbunden, an welcher das eigentliche Kraftwerk mit der Generator- sowie Motoreinheit platziert ist. Wenn überschüssige Energie aus dem Netz zur Verfügung steht, kann Wasser aus dem unteren Reservoir über den Betrieb des Motors in das obere Reservoir gepumpt werden. Steht keine überschüssige Energie mehr zur Verfügung bzw. ist die Kapazität des Kraftwerkes bereits ausgeschöpft und das obere Reservoir maximal befüllt, kann die Druckleitung gesperrt und die Energie nahezu verlustfrei (siehe 2.4) in Form von Lageenergie des Wassers gespeichert werden. Wird Energie benötigt, kann das Wasser aus dem oberen Reservoir über die Druckleitung die Turbine antreiben, wodurch der Generator die Bewegungsenergie wieder in elektrische Energie umwandelt. PSKW können meist innerhalb von 1 - 2 Minuten zwischen Lade- und Entladebetrieb wechseln und somit rasch auf die aktuelle Netzlast reagieren. Dasselbe Wasser kann grundsätzlich in vielen Lade- und Entladezyklen wiederverwendet werden und muss lediglich um die durch Verdunstung und Versickerung verlorene Menge ergänzt werden. Die Wirtschaftlichkeit eines PSKW ist dadurch gegeben, dass die Energiepreise sich an Angebot und Nachfrage anpassen, die Preise also geringer sind, wenn mehr Energie zur Verfügung steht, als gerade benötigt wird. Im umgekehrten Fall wird bei hoher Nachfrage die Energie wieder zu einem höheren Preis verkauft. Für die erreichbare Energiespeicherkapazität ist das Wasserspeichervolumen sowie der Höhenunterschied zwischen den Reservoiren ausschlaggebend (2.1). Als unteres Reservoir kann grundsätzlich jedes ausreichend große Wasservorkommen genutzt werden, d.h. künstliche Becken, Seen, Flüsse oder sogar auch direkt das Meer. Als oberes Reservoir wird üblicherweise ein künstliches Becken oder ein bestehender See ohne natürlichen Ablauf herangezogen.

2.1 Energiespeicherkapazität

Das theoretisch erreichbare Energiespeicherpotential eines PSKW kann über folgende Formel berechnet werden:

$$E = \frac{\rho \; g \; h \; V \; \mu}{3.600}$$

E = Energiespeicherkapazität (Wh)

ρ = Dichte (≈ 1.000 kg/m³ für Wasser)

g = Erdbeschleunigung (≈ 9,8 m/s²)

h = Höhendifferenz zwischen den Reservoiren (m)

V = verwendbares Wasservolumen (m³)

μ = Wirkungsgrad des Generators (%)

Die Division durch 3.600 wird durchgeführt, um die Einheit Watt-Stunde zu erhalten.

Rechenbeispiel:

Gegeben ist ein PSKW mit einem nutzbaren Volumen von 100.000 m³, einer Höhendifferenz von 100 m und einem Wirkungsgrad des Generators von 90%.

$$E = \frac{1.000 * 9{,}8 * 100 * 100.000 * 0.9}{3.600} = 24.500.000 \; Wh = 24{,}5 \; MWh$$

Das Kraftwerk im Beispiel hätte entsprechend eine Kapazität von 24,5 MWh – könnte also potentiell 7 Stunden durchgehend 3,5 MW in das Netz einspeisen. Für die Berechnung der benötigten „Ladeenergie“ zur Füllung eines komplett geleerten oberen Reservoirs muss der Wirkungsgrad der Pumpe (beispielsweise 85%) anstatt des Generator-Wirkungsgrades berücksichtigt werden.

$$E_{Ladung} = \frac{1.000 * 9{,}8 * 100 * 100.000}{3.600 * 0{,}85} \cong 32 \; MWh$$

2.2 Wirkungsgrad

Unter Vernachlässigung der natürlichen Zuflüsse durch das Einzugsgebiet des Reservoirs sowie der Wasserverluste durch Verdunstung und Versickerung wird grundsätzlich für den Betrieb eines PSKW mehr Energie benötigt, als durch das Kraftwerk wieder gewonnen werden kann. Die Faktoren, welche sich auf den Wirkungsgrad des PSKW primär auswirken sind der Wirkungsgrad der Pumpe, welche das Wasser in das obere Reservoir pumpt, der Wirkungsgrad der Turbine und des Generators, welche die kinetische Energie des herabfließenden Wassers wieder in elektrische Energie umwandeln sowie die hydraulischen Verluste durch den Wasserwiderstand. Zudem wird ein kleiner Energieanteil auch für den Eigenbedarf des PSKW benötigt. Der tatsächlich realisierte Gesamtwirkungsgrad liegt im Bereich zwischen 75% und 85% (Ardizzon et al. 2014; Chen et al. 2009; Luo et al. 2015).

Zusätzlich treten Übertragungsverluste durch die Übertragung der Energie von und zu dem öffentlichen Netz auf, weshalb kürzere Distanzen zu bestehenden Netzen nicht nur aufgrund der niedrigeren Anschlusskosten, sondern auch wegen der geringeren Übertragungsverluste zu bevorzugen sind. Dies wird zwar nicht in den Wirkungsgrad des PSKW einbezogen, spielt jedoch für die Planung eines neuen Standortes sehr wohl eine Rolle.

2.3 Dimensionierung eines Dammes

Die Dimensionen sind bei der Errichtung eines Dammes von großer Bedeutung, da das Dammvolumen die erforderliche zu beschaffende und zu verarbeitende Materialmenge festsetzt und somit entscheidend die Kosten und benötigte Dauer der Umsetzung beeinflusst.

Die für die Errichtung eines Schüttdammes erforderliche Materialmenge (Dammvolumen) entspricht dem Produkt aus Dammquerschnitt und Dammlänge. Der Querschnitt wiederum entspricht im vereinfachten Fall (Abbildung 6) dem Produkt aus Dammhöhe und Dammbreite / 2 – also Dammhöhe^2. Die Böschungswinkel sind entsprechend dem verwendeten Schüttmaterial zu wählen und unabhängig der Dammhöhe in ihrem Wert fixiert.

Um die Mindestkapazität eines Reservoirs zu erhöhen, kann entweder die Tiefe (entspricht bei einem Ringdamm der Dammhöhe) oder die Reservoirfläche erhöht werden.

Unter der vereinfachten Annahme eines kreisförmigen Reservoirs (Dammlänge entspricht dem Kreisumfang) und eines Dammquerschnitts in Form eines gleichschenkligen Dreiecks (entspr. Abbildung 6) stellt sich heraus, dass eine Erhöhung des Reservoirvolumens um den Faktor x durch Vergrößerung der Reservoirfläche einen Anstieg des Dammvolumens

um den Faktor $\sqrt{x}$ zur Folge hat. Wird die Erhöhung des Reservoirvolumens hingegen allein durch Veränderung der Dammhöhe erzielt, steigt das Dammvolumen um den Faktor x^2.

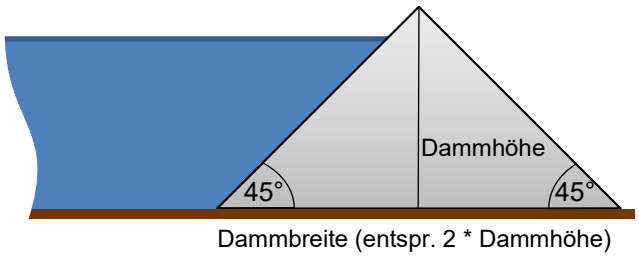


Abbildung 6: Querschnitt vereinfachter Schüttdamm

Beispielrechnung unter Vernachlässigung des Kapazitätsverlustes durch den Dammanteil innerhalb des Reservoirs:

Dammhöhe h = 10 m

Reservoirfläche A = 1.000 m²

Reservoirvolumen = Dammhöhe * Reservoirfläche = 10.000 m³

Umfang $U = 2\,\pi\,\sqrt{\frac{A}{\pi}} = 112{,}1\,m$

Dammvolumen (vereinfacht) = h² * U = **11.210 m³**

Bei einer theoretisch benötigten Erhöhung des Reservoirvolumens um den Faktor x=4, also auf 40.000 m³, würde sich bei gleichbleibender Dammhöhe der Umfang auf 224,2 m, und somit das Dammvolumen auf 22.420 m³ verdoppeln (Faktor $\sqrt{x}$). Bei gleichbleibendem Umfang müsste die Dammhöhe jedoch auf 40 m erhöht werden, was eine Versechzehnfachung (Faktor x^2) des Dammvolumens auf 179.360 m³ zur Folge hätte.

Dies trifft auch auf eine Talsperre zu, wenn diese als Schüttdamm ausgeführt wird. Bei Talsperren macht zwar eine Längenänderung des Dammes an einem bestimmten Ort keinen Sinn, bei der Suche neuer Dammstandorte muss diese Tatsache jedoch berücksichtigt werden und mit Hinblick auf eine Minimierung des resultierenden Dammvolumens für ein Reservoir derselben Größe ein längerer Damm gegenüber einem höheren bevorzugt werden.

2.4 Planungsfaktoren

Bei der konkreten Planung eines neuen PSKW müssen verschiedene Faktoren berücksichtigt werden, darunter unter anderen:

a. **Wasserverfügbarkeit**
   Das für den Betrieb des PSKW notwendige Wasser muss in ausreichender Menge für die erste Füllung sowie zum Ausgleich der Wasserverluste vorhanden sein. Idealerweise besteht ein natürlicher Zufluss durch Niederschläge innerhalb des Einzugsgebietes des oberen Reservoirs. Dies verbessert zudem den Wirkungsgrad, da auch Wasser zur Turbine geleitet werden kann, welches zuvor nicht hochgepumpt werden musste. Um den Betrieb zu gewährleisten würde jedoch auch ein Zufluss in das untere Reservoir genügen.
b. **Technische und wirtschaftliche Umsetzbarkeit**
   Um im Rahmen der technischen und wirtschaftlichen Umsetzbarkeit zu bleiben, werden Grenzwerte für einzelne Parameter festgelegt und lediglich Standorte analysiert, welche diese Bedingungen erfüllen. Dies betrifft beispielsweise die maximale Distanz zwischen zwei Reservoiren, ein Mindestvolumen eines Reservoirs, Maximaldimensionen eines potentiellen Dammes oder auch die Erreichbarkeit des Standortes – um mit entsprechend notwendigem Gerät die Errichtung durchführen zu können.
c. **Verluste durch Verdunstung**
   Die Wasserverluste durch Verdunstung sind von einigen Faktoren wie Lufttemperatur, Luftfeuchtigkeit, Windstärke und Sonneneinstrahlung abhängig und deshalb nur sehr grob berechenbar. Lacal-Arántegui und Tzimas (2012) erwähnen einen geschätzten Verdunstungsverlust von 10-15 % innerhalb eines Jahres für ein Reservoir in Zypern – was Klimabedingt eher eines der verdunstungsstärkeren Gebiete ist. Eine mögliche Option zur Milderung von Verdunstungsverlusten ist der Einsatz von schwimmenden Abdeckungen (Lu et al. 2018).
d. **Verluste durch Versickerung**
   Durch eine geologische Analyse des potentiellen Reservoir-Beckens kann die Eignung des Untergrundes – speziell die Wasserdurchlässigkeit – untersucht werden. Ist die Versickerungsrate zu hoch, müsste die Oberfläche versiegelt werden oder genügend ausgleichende Zuflüsse in das obere Reservoir bestehen, um den potentiellen Standort umsetzen zu können.
e. **Zufluss innerhalb des Einzugsgebietes**
   Mit bestehenden GIS-Methoden und Werkzeugen kann das Wassereinzugsgebiet (Watershed) einfach eruiert werden. Sobald das gesamte Einzugsgebiet ermittelt ist, kann mit Niederschlagsdaten der Vergangenheit ein erwarteter natürlicher Zufluss errechnet

werden. Für eine positive Auswirkung auf die Energie- und Kosteneffizienz muss dieser Zufluss dem oberen Reservoir gelten.

f. Potentielle Gefahren durch Überlauf, Dammbruch oder Erdrutsch

Für eine genauere Standortanalyse ist auch die Gefahrabschätzung durch potentielle plötzliche Wasseraustritte durch Überlauf (Oroville Damm, 2017 in den USA), Dammbruch (Leguaseca Staumauer, 1988 in Spanien) oder Erdrutsch (Vajont Stausee, 1963 in Italien) notwendig. Hier wird der Wasserabfluss mit Maximalmenge des Reservoirvolumens simuliert und das Risiko einer möglichen Überschwemmung bewohnter Gebiete abgeschätzt.

g. Auswirkungen auf Tier- und Pflanzenwelt

Für die konkrete Abschätzung der Auswirkungen auf Flora und Fauna muss eine UVP (Umweltverträglichkeitsprüfung) durchgeführt werden.

h. Geographische Nähe zu Energieerzeuger und -verbraucher

Da durch den Energietransport distanzabhängige Übertragungsverluste auftreten, ist es für den erreichbaren Gesamtwirkungsgrad des PSKW vorteilhaft, wenn die Energieerzeuger, deren überschüssige Energie gespeichert werden soll und die Verbraucher, welchen die Energie zu einem späteren Zeitpunkt zur Verfügung gestellt wird, nahe dem Kraftwerk sind. Der Gesamtwirkungsgrad des PSKW steigt an, wenn die Distanz zwischen Erzeuger und Verbraucher abnimmt.

i. Ausschluss durch aktuelle Landnutzung

Aktuell bewohnte oder besonders geschützte Gebiete (UNESCO Welterbe und Natura 2000) werden von der Analyse meist von Beginn an ausgeschlossen. Das Ziel ist es neue Standorte zu finden, welche möglichst geringe negative Auswirkungen auf die Umwelt und Bevölkerung zur Folge haben.

3 Methode

Das Ziel ist eine performante Methodik zur Eruierung geeigneter Standorte für die Errichtung von Talsperren zu entwickeln, wodurch neue Standorte für die Errichtung von PSKW gefunden werden können. Verwendete Methoden in der Literatur (Lu und Wang 2017; Lu et al. 2018; Larentis et al. 2010; Petheram et al. 2017; Yi et al. 2010) bestehen grundsätzlich darin, aus einem DEM ein Drainagenetzwerk abzuleiten und entlang der Drainagelinien in definierten Intervallen Prüfpunkte für die Simulation bzw. Analyse eines möglichen Dammes zu erstellen. Derselbe Prüfpunkt wird für die Simulation verschiedener Dammhöhen herangezogen und schließlich anhand der ermittelten Werte wie Reservoir-Volumen und Dammdimensionen die geeignetsten Kandidaten selektiert. Eine Schwierigkeit dieses Ansatzes ist es, dass die beste Ausrichtung eines jeden Damm-Kandidaten meist nicht exakt im rechten Winkel zur Drainagelinie liegt und somit auch für die Eruierung der Ausrichtung an jeder Position das DEM analysiert werden muss. Larentis et al. (2010) setzten dies Pixelweise um, indem sie von der Drainagelinie aus in beiden Seiten nach der nächst-höhergelegenen DHM-Zelle suchten. Wenn nur gleich hohe DHM-Zellen angrenzten, wurde im rechten Winkel zur Drainagelinie selektiert. Dies ist sehr rechenintensiv und liefert trotzdem nicht immer das ideale Ergebnis, da jeweils nur die Nachbarzellen berücksichtigt werden können und das zu sperrende Tal nicht als Ganzes betrachtet wird. Weiters muss ein geringes Suchintervall entlang der Drainagelinie gewählt werden um keine potentiell besser geeigneten Positionen zu übersehen. Bei großen Untersuchungsgebieten ist dies jedoch aufgrund des sehr hohen Rechenaufwandes oft nicht möglich und es muss ein Kompromiss auf Kosten des Suchintervalls gefunden werden.

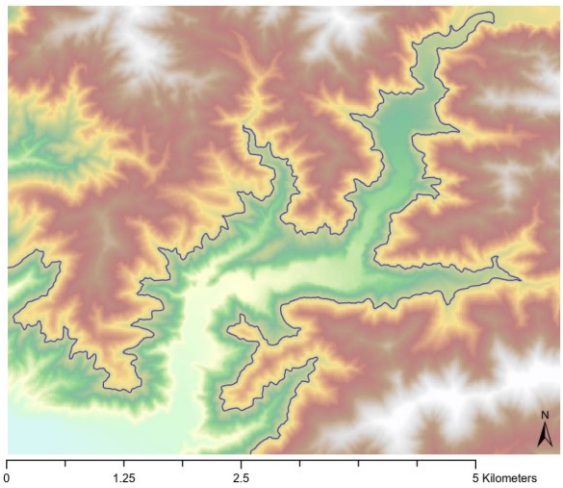


Abbildung 7: Beispiel Höhenlinie

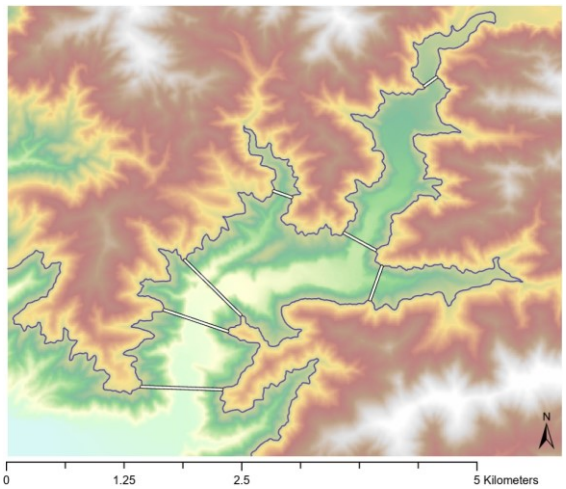


Abbildung 8: Engstellen

Im Gegensatz dazu basiert der in dieser Arbeit gewählte Ablauf auf der Analyse von Höhenlinien, um die geeignetsten Standorte für die Errichtung eines möglichst großvolumigen künstlichen Beckens durch setzen einer möglichst kleindimensionierten Talsperre ableiten

zu können. Dies soll erreicht werden, indem in einem Intervall in der Größenordnung der Auflösung der SRTM-1 Daten die Dimensionen aller möglichen Dammkonstellationen (Richtungen und Höhen) auf performante Weise geprüft und nur für die aussichtsreichsten Kandidaten alle Kennwerte berechnet werden und die weitere theoretische Eignung analysiert wird.

In Abbildung 7 ist eine 800 m Höhenlinie des San Bernardino County (Seven Oaks Reservoir in Kalifornien) zu sehen. Mögliche geeignete Positionen sind anhand dieser sogleich ersichtlich, wenn manuell nach Engstellen entlang der Höhenlinie gesucht wird. In Abbildung 8 sind einige offensichtliche Engstellen mit Linien verbunden. Um unterschiedliche mögliche Positionen miteinander vergleichen zu können, müssen diese zuerst bewertet werden. In dieser Arbeit wurde das Verhältnis von Reservoirvolumen zu Dammvolumen als Reihungs-Kriterium herangezogen. Da die Berechnung des Reservoirvolumens vergleichsweise rechenintensiv ist, wird dieser Schritt jedoch erst zu einem Zeitpunkt durchgeführt, zu dem die Anzahl der Dammkandidaten bereits deutlich eingeschränkt ist. Es wird nicht versucht den optimalsten Standort eines Gebietes ausfindig zu machen, sondern eine Datenbank vieler geeigneter Dammkandidaten an verschiedenen Positionen und mit unterschiedlichen Höhen zu erstellen, wodurch später ausreichend Kombinationsmöglichkeiten für die Auswahl der Reservoir-Paare zur Errichtung eines PSKW zur Verfügung stehen sollen.

Um die Engstellen automatisiert zu eruieren, sollen entlang der Höhenlinien in kleinen Intervallen Prüfpunkte erstellt werden. Anschließend werden zwischen diesen Prüfpunkten neue zweidimensionale Dammkandidaten simuliert und die Dammlänge sowie der resultierende Reservoirumfang berechnet. Dies kann mit einfachen arithmetischen Mitteln und ohne Einsatz von vergleichsweise rechenintensiven GIS-Werkzeugen erfolgen. Diese große Anzahl an Kandidaten lässt sich bereits durch das Verhältnis von Reservoirumfang und Dammlänge reihen – wobei ein hohes Verhältnis, d.h. großer Umfang und kleine Dammlänge, einen geeigneteren Kandidaten ausweist. Als Mindestwert für ein noch akzeptables Längenverhältnis wurde 10 gewählt und alle Kandidaten mit niedrigeren Werten entfernt. Als Ergebnis dieses Schrittes soll bereits eine ausreichend reduzierte Auswahl an grundsätzlich geeigneten Dammkandidaten vorliegen um diese mittels bestehender Werkzeuge weiter analysieren und die optimalsten Kandidaten selektieren zu können.

3.1 Ablauf

Die Entwicklung des genauen Ablaufes (Abbildung 9) wurde anhand eines reduzierten Testgebietes (Abbildung 7) mit ca. 28 km² Ausdehnung und eines DEM mit Auflösung von 10 m (Datenquelle: USGS National Elevation Dataset) durchgeführt und ergab den nachfolgenden Prozess. Das eigentliche Untersuchungsgebiet für diese Arbeit ist ca. 100-mal größer und wurde erst nachträglich mit dem finalen Prozess analysiert.

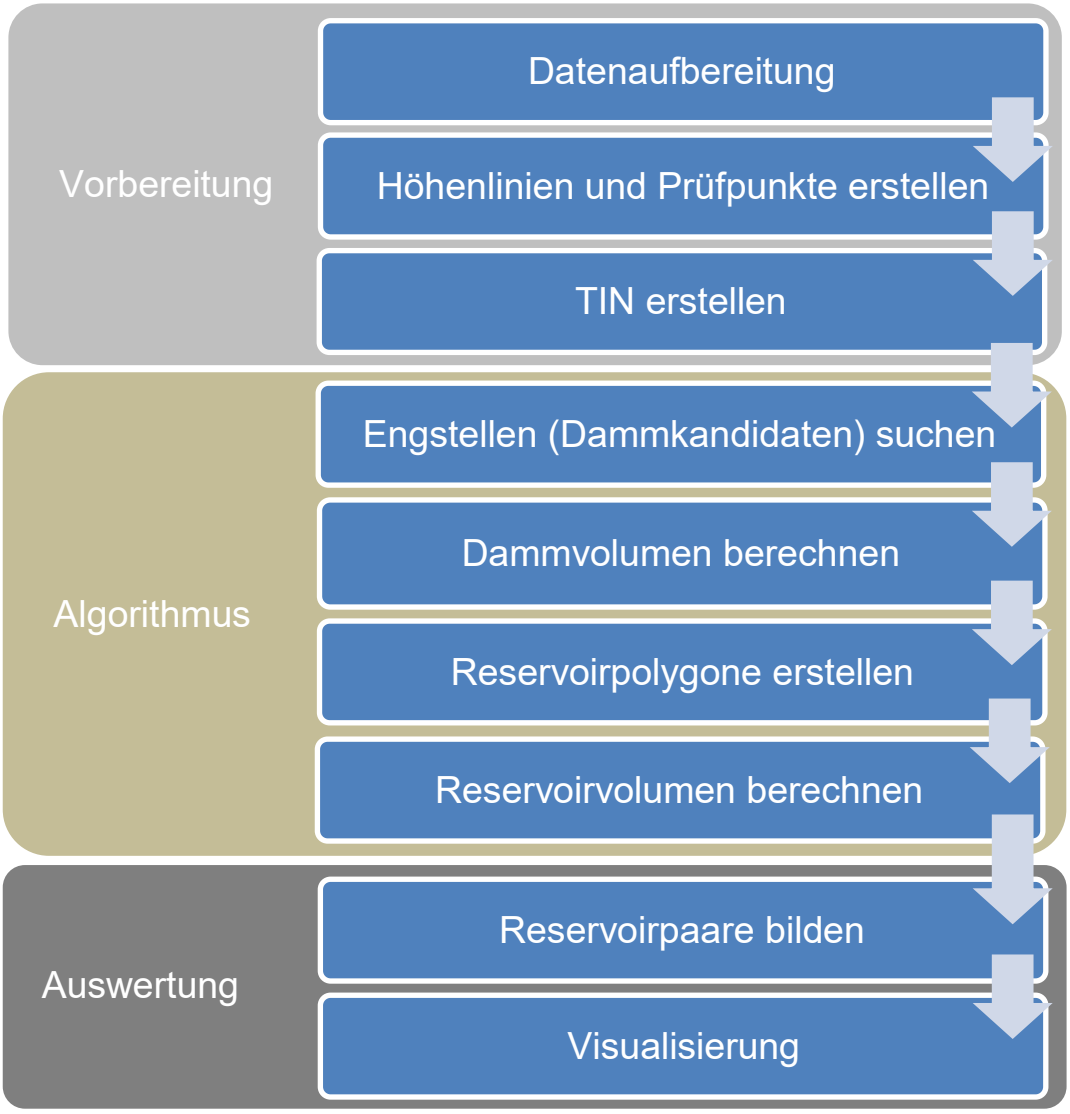


Abbildung 9: Prozessablauf

Besonderes Augenmerk wurde darauf gelegt, dass die Dammkandidaten in möglichst frühen Teilschritten radikal reduziert werden, da speziell die Erstellung der Reservoir-Polygone für die Reservoir-Volumensberechnung ziemlich rechenintensiv ist.

Die Bedienung erfolgt anhand des Ribbon-Menüs welches durch das entwickelte ArcGIS Pro Add-In im eigenen Reiter „Reservoir“ zugänglich ist (Abbildung 10).

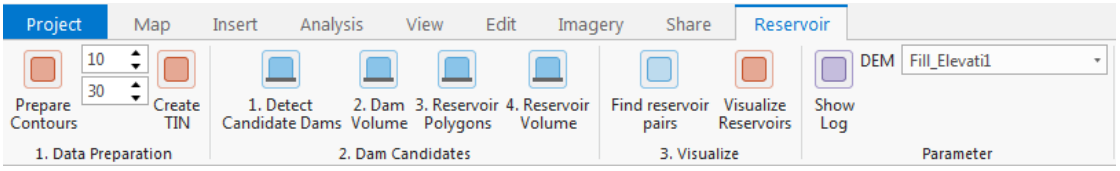


Abbildung 10: Add-In Menü in ArcGIS Pro

Die durch den ModelBuilder erstellten Modelle wurden in einer eigenen Toolbox verpackt und liegen dem Add-In bei. Zudem sind alle Modelle bildlich in dieser Arbeit enthalten. Um alle Funktionen des Add-In nutzen zu können, muss die Toolbox mit dem Namen Reservoir.tbx dem eigenen Projekt hinzugefügt werden. Der Quellcode ist wie in Anhang A: Quellcode beschrieben frei zugänglich und auch eine bereits kompilierte Version des Add-In wird zum Download angeboten.

3.1.1 Datenaufbereitung

Für die nachfolgenden Schritte ist ein DHM ohne Lücken notwendig. Da die Ausgangsdaten und das Untersuchungsgebiet sehr individuell sein können, wurde hier kein Automatismus angestrebt, sondern das DHM manuell aufbereitet. Die Aufbereitung der Daten für das Untersuchungsgebiet Vorarlberg ist unter 3.2.1 SRTM beschrieben. Die Daten des kleineren Seven Oaks Testgebietes (Abbildung 11) waren bereits lückenfrei.

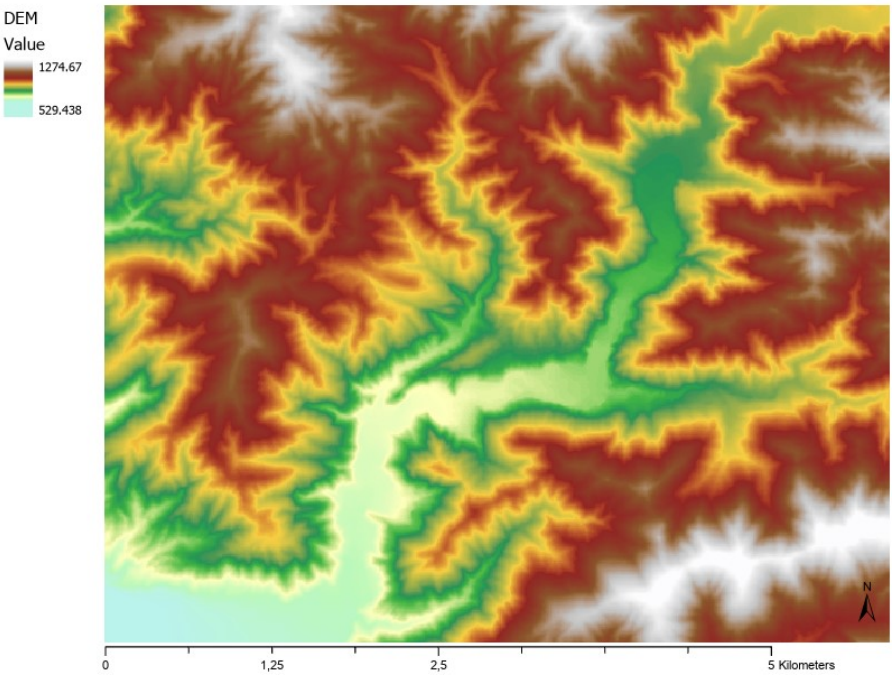


Abbildung 11: DHM des Seven Oaks Testgebietes (Kalifornien)

3.1.2 Höhenlinien und Prüfpunkte erstellen

Die Erstellung der Höhenlinien und Prüfpunkte wurde in einem Modell mit bestehenden Tools abgebildet und aus dem Add-In aufgerufen (A.1 PrepareContoursButton):

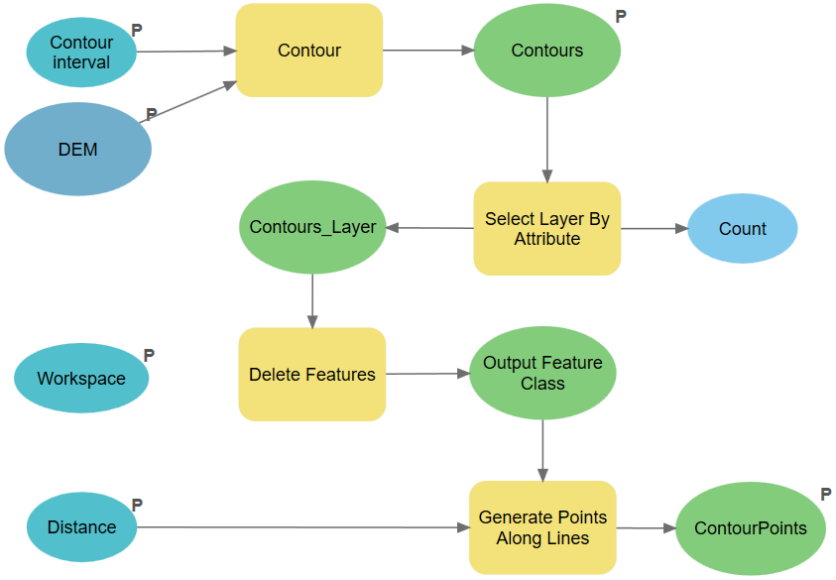


Abbildung 12: Modell "Höhenlinien und Prüfpunkte erstellen"

Mit einem „P“ markierte Elemente sind Parameter des Modells. Parameter können beim Aufruf des Tools übergeben und somit einzeln konfiguriert werden. Der Workspace wurde zusätzlich als Parameter übergeben und für die Pfade der Ausgabe Feature Classes als %Workspace% Inline-Variablen herangezogen, da das Modell mit Standardpfaden ansonsten nicht ohne Anpassungen an einem zweiten Rechner ausgeführt werden konnte. Die Ausgabe Feature Classes „Contours“ sowie „ContourPoints“ wurden ebenfalls als Parameter markiert, da diese ansonsten nicht zur Anzeige hinzugefügt werden, wenn das Modell über das Add-In aufgerufen wird – unabhängig des gesetzten Markers „Zur Anzeige hinzufügen“.

Zuerst werden in definierten Höhenintervallen (Default: 10 m) Höhenlinien erzeugt (Tool: „Contour“) und der Anzeige als Ebene „Contours“ hinzugefügt. Hierdurch werden auch viele grundsätzlich für unsere Zwecke ungeeignete Höhenlinien erstellt (siehe markierte Konturlinien in Abbildung 13). Diese durch Bergkuppen bzw. Hügel resultierenden oder aus einem Becken außerhalb des Untersuchungsgebietes hereinragenden Linien repräsentieren keine geeigneten Geländeformen für die Errichtung von Reservoiren in entsprechender Größe. Deshalb werden alle Linien mit einer Länge < 10.000 m selektiert und gelöscht (Tools: „Select Layer By Attribute“, „Delete Features“). Zuletzt werden mit dem Tool „Generate Points Along Lines“ in definierten Intervallen (Default: 30 m) Punkte entlang der Höhenlinien generiert und diese in einem eigenen Layer der Anzeige hinzugefügt (Abbildung 14). Für eine bessere Erkennbarkeit wird in Abbildung 15 und in den folgenden Schritten falls erforderlich

auch eine Darstellung der jeweiligen Ergebnisse der einzelnen 800 m Höhenlinie abgebildet.

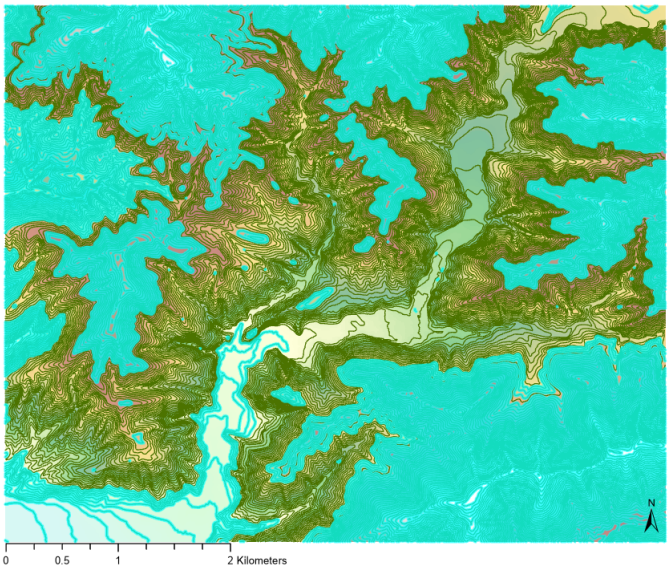


Abbildung 13: Ungeeignete Konturlinien (Türkis markiert)

Tabelle 3: Parameter für Erstellung der Prüfpunkte

Beschreibung	Standard - Wert
Höhenintervall Konturlinien	10 m
Punktintervall entlang Konturlinie	30 m
Minimale Konturlänge	10.000 m

Als Ergebnis dieses Schrittes wurden für das Seven Oaks Testgebiet mit den beschriebenen Einschränkungen 50 Konturlinien mit insgesamt 31.098 Prüfpunkten erzeugt (Abbildung 14).

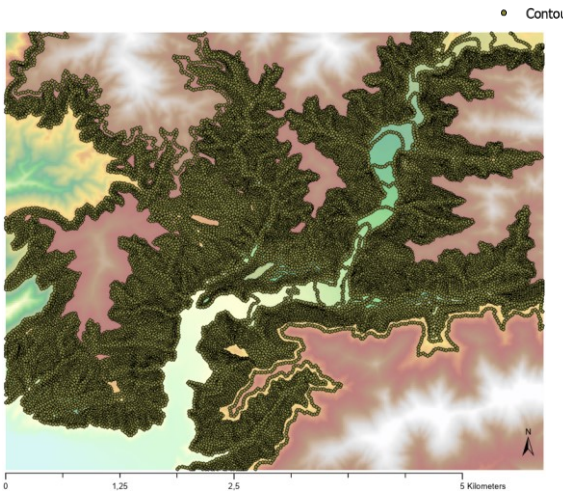


Abbildung 14: Erstellte Prüfpunkte

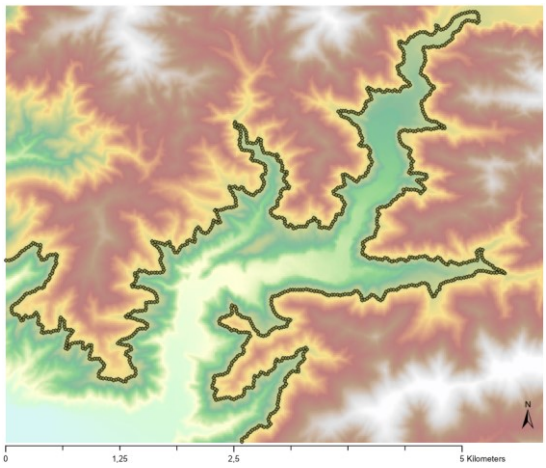


Abbildung 15: Punkte an 800 m Kontur

3.1.3 TIN erstellen

Für die gewählte Variante der Volumensberechnung der Reservoire wird ein „Triangulated Irregular Network“ (TIN) des Geländes benötigt. Um für den Algorithmus einen entsprechenden Layer vorzubereiten wurde ein sehr einfaches Modell (Abbildung 16) erstellt, welches aus dem Add-In Menü aufgerufen wird (A.2 CreateTINButton) und anschließend aus dem angegebenen DHM mit einer Z Toleranz von 10 m ein TIN mit der Bezeichnung „TIN“ erzeugt. Die Z Toleranz von 10 m wurde gewählt um die Anzahl an Punkten möglichst gering zu halten und trotzdem keine zu ungenauen Ergebnisse in der Volumensberechnung zu erhalten. Das TIN für das Testgebiet konnte somit mit 11.947 Punkten dargestellt werden (Abbildung 17).

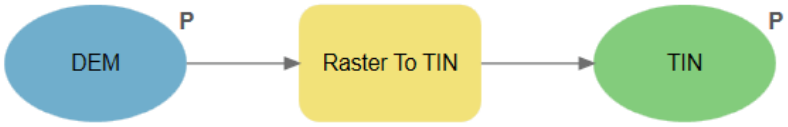


Abbildung 16: Modell "TIN erstellen"

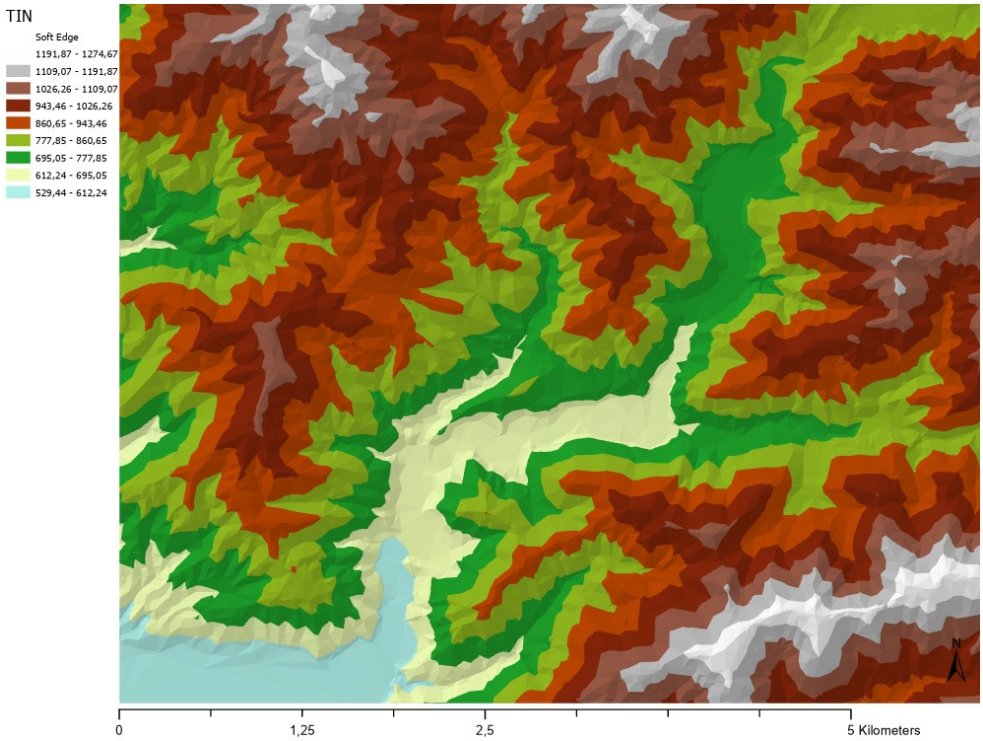


Abbildung 17: Erstelltes TIN des Testgebietes

3.1.4 Engstellen (Dammkandidaten) suchen

Da für die Ermittlung der Engstellen entlang der Höhenlinien keine bestehenden Tools gefunden werden konnten, wurde dieser Teil in Form eines C# Algorithmus implementiert (A.3 DetectCandidateDamsButton).

Tabelle 4: Parameter für Dammanalyse

Beschreibung	Wert
Maximale $Distanz_{Euklid}$ (= Dammlänge)	1.000 m
Maximale Dammhöhe	300 m
Minimale Dammhöhe	10 m
Maximale $Distanz_{Isohypse}$ (≈ Reservoirumfang)	30.000 m
Minimale $Distanz_{Isohypse}$	1.000 m
Minimales Verhältnis $Distanz_{Isohypse}$ / $Distanz_{Euklid}$	10

Die zugrundeliegende Funktionsweise des Algorithmus ist, jeden Punkt mit jedem anderen Punkt derselben Höhenlinie zu verbinden und die euklidische Distanz über den Satz des Pythagoras zu berechnen. Dies allein ist jedoch nicht ausreichend, da nebeneinanderliegende Punkte automatisch eine geringe euklidische Distanz zueinander aufweisen, jedoch für die Errichtung einer Talsperre absolut ungeeignet sind. Deshalb muss zusätzlich die Distanz entlang der Höhenlinie berücksichtigt werden. Die euklidische Distanz zwischen zwei Punkten entspricht der Länge der virtuellen Dammkrone, die Distanz entlang der Höhenlinie entspricht dem virtuellen Reservoirumfang. Je höher der Quotient von Reservoirumfang zu Dammlänge, desto besser ist die Eignung des Kandidaten.

Ein Problem dieser Methode ist, dass die Anzahl der möglichen Verbindungen quadratisch mit der Anzahl der Punkte ansteigt. Es stehen grundsätzlich n * (n – 1) Verbindungs-Kombinationen zur Verfügung, wenn jeder Punkt mit jedem anderen Punkt kombiniert werden würde. Da die Richtung des Dammes jedoch irrelevant ist, kann diese Zahl nochmals halbiert werden:

$$Möglichkeiten\ (M) = \frac{n * (n-1)}{2}$$

Beispiel: Bei einem Punktintervall von 100 m entlang einer 100 km Höhenlinie stehen 1.000 Prüfpunkte mit 499.500 eindeutigen Verbindungsmöglichkeiten zur Verfügung. Wird jedoch ein größeres Untersuchungsgebiet gewählt und die Länge der Höhenlinie verzehnfacht,

entstehen bei gleichbleibender Punktdichte nun 10.000 Prüfpunkte und somit 49.995.000 Möglichkeiten (**≈ Faktor 100**).

Dieses Problem wurde durch einige Eingriffe im Algorithmus entschärft:

Mit der Definition, dass ein mögliches Reservoir in der hier gesuchten Größenordnung einen Mindestumfang von 1 km und einen Maximalumfang von 30 km aufweisen soll, wird beginnend mit dem ersten Punkt entlang einer Isohypse jeweils nur mit Punkten verknüpft, welche mindestens 1 km und maximal 30 km entlang dieser Linie vom Ausgangspunkt entfernt sind. Dies ergibt für obiges Beispiel:

$$Möglichkeiten\ (M_2) = (n - 300) * 290 + \sum_{k=1}^{290} k$$

Für 1.000 Prüfpunkte ergibt nun M_2 = 245.195

Für 10.000 Prüfpunkte und zehnfacher Länge ergibt M_2 = 2.855.195 (**= Faktor 11.6**)

100.000 Prüfpunkte und hundertfache Länge ergeben M_2 = 28.955.195 (**= Faktor 118**)

Durch das nun lineare Verhalten ist eine Skalierbarkeit auch für größere Untersuchungsgebiete gegeben.

Konkret werden jeweils alle unter 3.1.2 erzeugten Punkte einer Höhenlinie in den Speicher geladen und in zwei sortierten Listen - Startpunkte und Endpunkte – vorbereitet.

Die Liste „Startpunkte“ enthält alle Punkte der Höhenlinie. Die Liste „Endpunkte“ ist eine Kopie, welcher bereits zu Beginn alle innerhalb der ersten 1.000 m entlang der Höhenlinie liegenden Punkte entfernt wurden.

Um die Distanz entlang der Höhenlinie ($Distanz_{Isohypse}$) zu berechnen, wird das ID Feld der Punkte herangezogen. Da dieses durch das Punkterstellungstool bereits in aufsteigender Reihenfolge entlang der Höhenlinie mit ganzzahligen Werten belegt wurde gilt:

$$Distanz_{Isohypse} = |SP.ID - EP.ID| * Punktintervall$$

Falls die $Distanz_{Isohypse}$ 30.000 m überschreitet, wird die Prüfroutine für diesen Startpunkt abgebrochen, der erste Punkt der Endpunktliste entfernt und die Schleife mit dem folgenden Punkt der Startpunktliste erneut ausgeführt.

Für die Berechnung der $Distanz_{Euklid}$ (Dammlänge) wird der Satz des Pythagoras auf die Punktkoordinaten SP und EP angewendet (Abbildung 18).

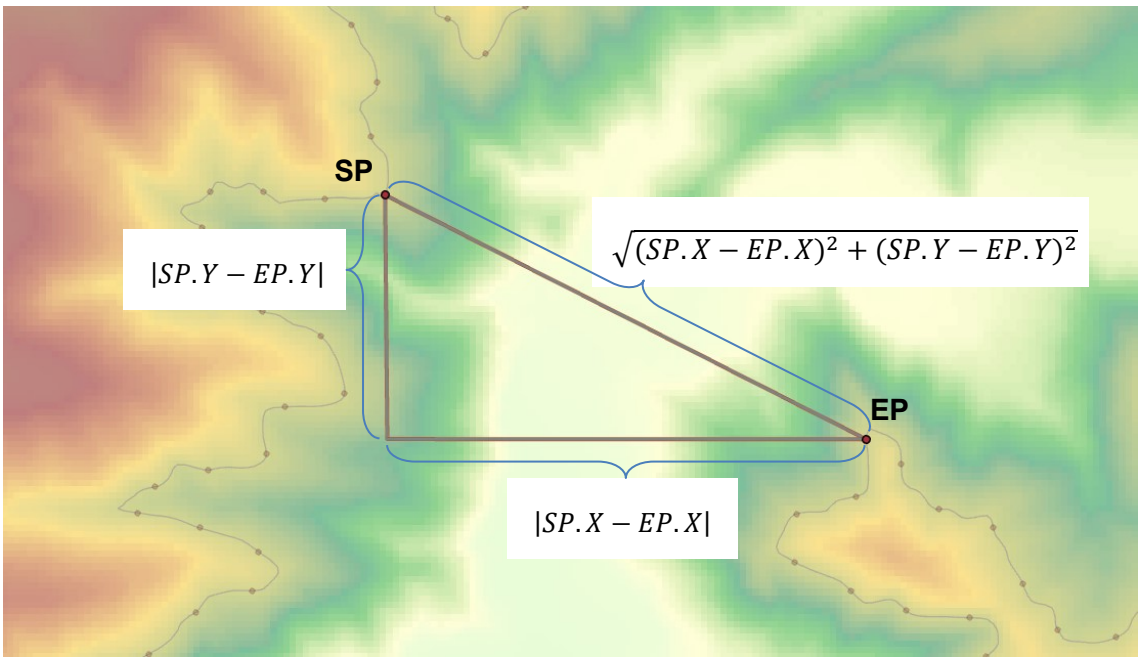


Abbildung 18: Berechnung der Dammlänge

$$Distanz_{Euklid} = \sqrt{(SP.X - EP.X)^2 + (SP.Y - EP.Y)^2}$$

Überschreitet die ermittelte Dammlänge 1.000 m oder unterschreitet der Quotient von $Distanz_{Isohypse}$ / $Distanz_{Euklid}$ (Längenbewertung) den Wert 10, wird diese Kombination verworfen und der nächste Endpunkt geprüft. Anderenfalls handelt es sich um einen tatsächlichen Kandidaten und es wird für dieses Start- und Endpunkt Paar ein „CandidateDam“ – Objekt erstellt, welches alle bisher ermittelten Eckdaten des Kandidaten speichert. Die genaue Implementierung ist in der Methode „AnalyseCountourPoints“ (A.3 DetectCandidateDamsButton) ersichtlich.

Je Höhenlinie werden die ermittelten Kandidaten nach der Längenbewertung absteigend sortiert und einzeln 3D PolyLine Features erstellt. An dieser Stelle wird nochmals eine Einschränkung der Kandidatenmenge vorgenommen, indem nur der bestgereihte Kandidat in einer Umgebung von 500 m entlang der Höhenlinie (jeweils Start- und Endpunkt) behalten wird. So wird vermieden, dass viele sehr ähnliche Dammkandidaten für dieselbe Höhe erstellt werden und den Prozess in weiterer Folge verlangsamen. Die Höhe (Z-Value) der Polylinie wird auf die Höhe der Höhenlinie gesetzt und um 5 m erhöht. Dies soll einerseits die zusätzliche Höhe der Dammkrone über dem Wasserspiel abbilden und zudem sicherstellen, dass unter 3.1.5 nur Kandidaten entfernt werden, welche das Gelände tatsächlich schneiden.

Für das Seven Oaks Testgebiet mit den 50 erstellten Höhenlinien und 31.098 Prüfpunkten wurden in 8 Sekunden aus 10.043.108 möglichen Kombinationen in diesem Schritt 170 Kandidaten selektiert (Abbildung 19) und in der GeoDatenbank gespeichert. An der 800 m Beispielkontur wurden 11 Kandidaten gefunden (Abbildung 21). Die Berechnungszeit konnte durch den Einsatz von MultiThreading gering gehalten werden.

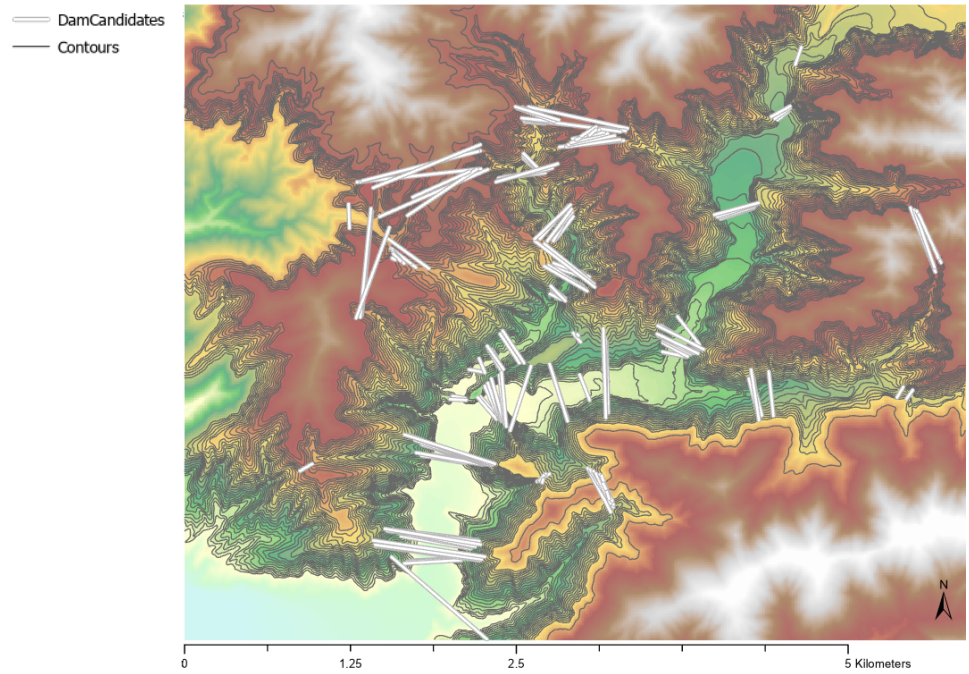


Abbildung 19: Erstellte Dammkandidaten

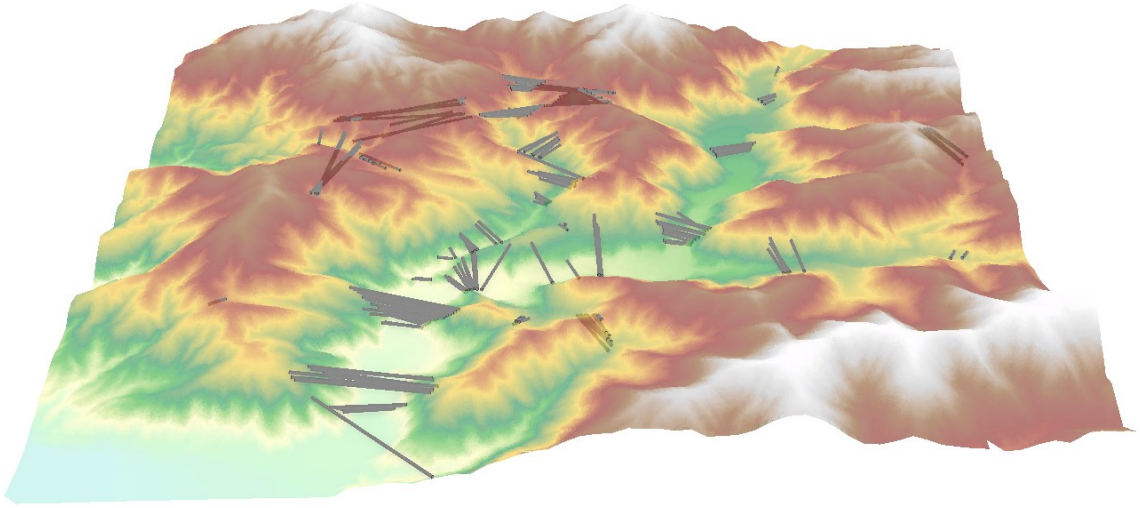


Abbildung 20: Erstellte Dammkandidaten 3D

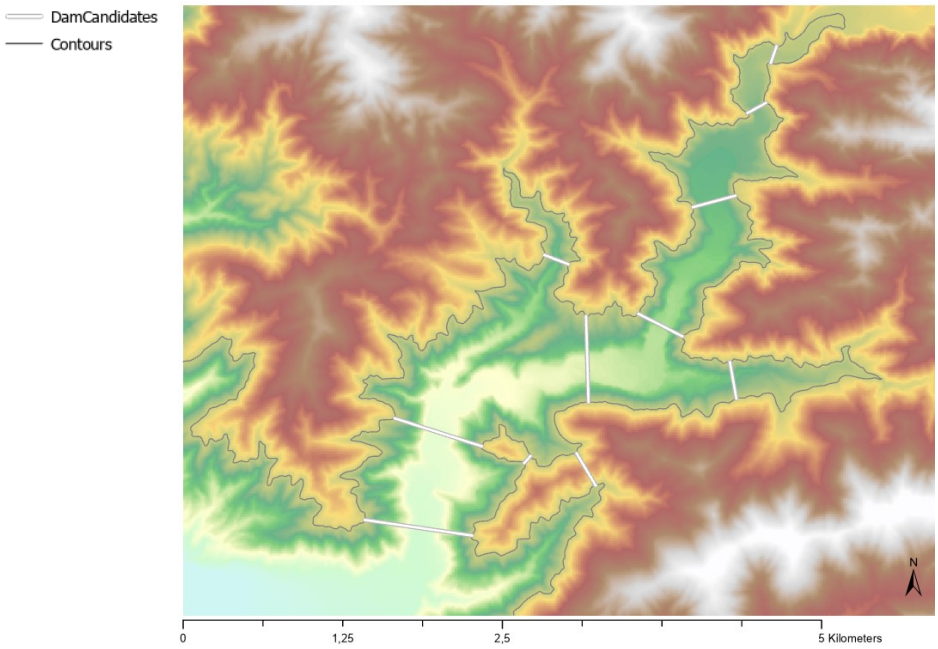


Abbildung 21: Kandidaten an 800 m Kontur

Wie in Abbildung 19, Abbildung 21 und Abbildung 20 zu erkennen, verlaufen einige Dammkandidaten direkt durch das Gelände. Dies tritt hauptsächlich deshalb auf, da dem Algorithmus die Richtung der Talseite nicht bekannt ist und hierdurch auch Engstellen an Höhenlinien quer durch das Gelände hindurch (also Bergseitig) mit einem Dammkandidaten verbunden werden. Zudem gibt es Konstellationen, in denen der Damm zwar Talseitig startet, ein Teil des Kandidaten jedoch trotzdem das Gelände schneidet. Beide Varianten sind unerwünscht und nicht als geeignet zu betrachten, werden aber hier erstellt und gespeichert, da diese im nachfolgenden Schritt 3.1.5 effizienter erkannt und bereinigt werden können.

Zu diesem Schritt ist noch anzumerken, dass getrennte Konturlinien derselben Höhe nicht gemeinsam berücksichtigt werden, sondern die Teile jeweils nur für sich. Dies ist der Tatsache geschuldet, dass ein Reservoir in solch einer Konstellation aus dem Untersuchungsgebiet hinausragen würde und somit innerhalb des Gebietes keine genaue Berechnung von Umfang und Volumen durchgeführt werden kann (Abbildung 22). Wenn in diesem Beispiel beide Konturlinien gemeinsam berücksichtigt bzw. diese zu einer zusammengefügt werden sollten, müsste das Untersuchungsgebiet entsprechend vergrößert werden.

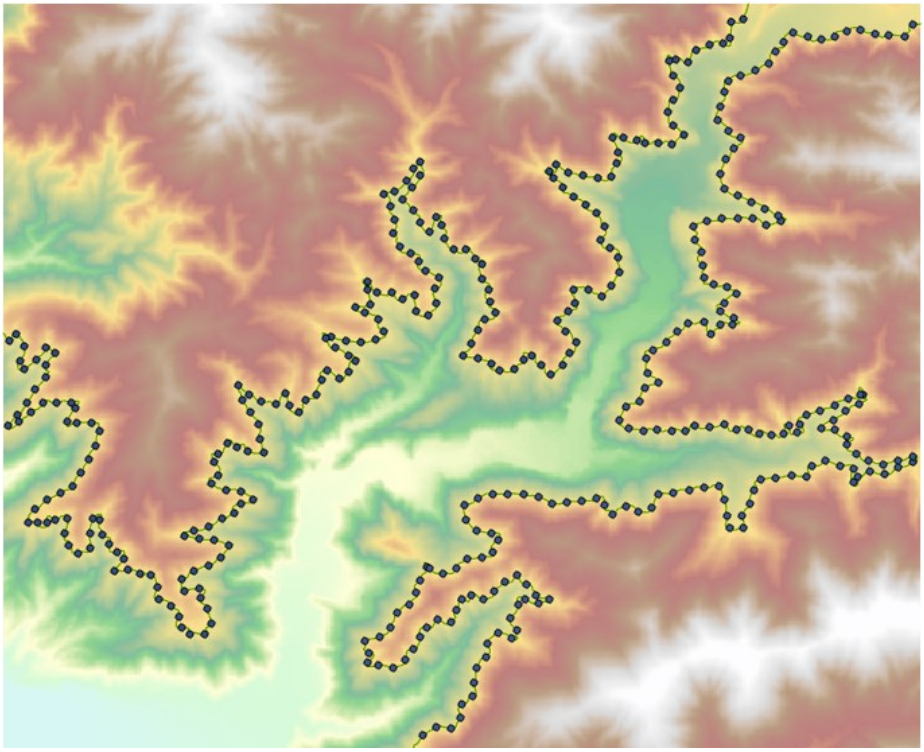


Abbildung 22: Zwei getrennte Konturlinien derselben Höhe

3.1.5 Dammvolumen berechnen

Als nächsten Schritt wird für alle Dammkandidaten ein Geländeschnitt (Profil) erzeugt und anhand dessen die maximale Dammhöhe sowie ein ungefähres Dammvolumen (2.3) berechnet. Zudem können ungeeignete Kandidaten anhand des Profils leicht erkannt und entfernt werden. Ungeeignet sind alle Kandidaten, deren maximale Dammhöhe außerhalb der Grenzwerte liegt (Tabelle 4) oder falls ein Höhenwert des Profils die Höhe der Höhenlinie übersteigt. Letzteres betrifft alle Konstellationen, in denen der Dammkandidat das Gelände schneidet. Die Eliminierung der geländeschneidenden Kandidaten wurde zuerst über das Tool „Intersect 3D Line with Surface“ getestet. Dieser Ansatz benötigte jedoch ca. die 4-fache Berechnungszeit, als über die hier beschriebene Profil-Variante.

Um alle Anforderungen dieses Arbeitsschrittes zu erfüllen wurde wieder eine C# Implementierung gewählt (A.4 DamVolumeButton). In dieser wird das bestehende Tool „Stack Profile“ der 3D Analyst Toolbox für alle Dammkandidaten gesammelt ausgeführt und das Profil zum verwendeten DHM erstellt. Diese Profilwerte werden durch das Tool direkt in eine Ergebnistabelle geschrieben und können anschließend ausgewertet werden.

OBJECTID	LINE_ID	FIRST_DIST	FIRST_Z
1	10305	0	800,0246
2	10305	9,96211	795,0665
3	10305	19,924122	788,9109
4	10305	29,886206	782,0198
5	10305	39,848316	774,5569
...	...	...	...
71	10305	697,345039	787,0398
72	10305	707,307123	790,9168
73	10305	717,269136	794,9861
74	10305	727,231246	799,9896
75	10306	0	800,0026

Abbildung 23: Ergebnistabelle des Tools StackProfile

Wie in Abbildung 23 zu sehen, stehen nun für alle Kandidaten-Linien (Line_ID) die Profilwerte (FIRST_Z) im Intervall der zugrundeliegenden DHM-Zellen (FIRST_DIST) zur Verfügung.

Hiermit lassen sich die Profile leicht visualisieren (Abbildung 24) und für die Eruierung der Kennwerte heranziehen. Für die maximale Dammhöhe wird der minimale Profilwert (FIRST_Z) von der Konturhöhe des entsprechenden Dammkandidaten abgezogen. Damit

ergibt sich am Beispiel des Kandidaten 2 aus Abbildung 24 eine maximale Dammhöhe von 800 m – 597 m = 203 m.

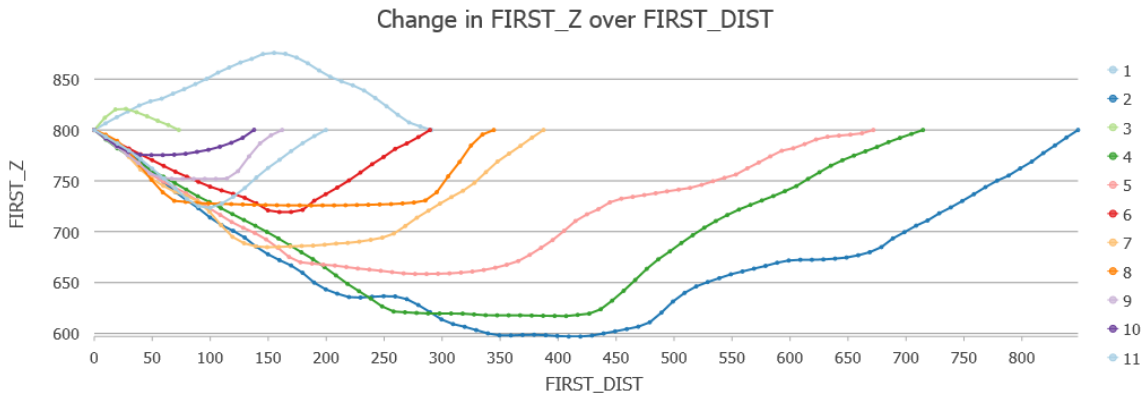


Abbildung 24: Profile der Dammkandidaten der 800 m Kontur

An dieser Stelle wird ebenfalls die Reduzierung der Kandidaten durchgeführt. Wird ein Höhenwert ausgegeben, der über der Höhe der Höhenlinie liegt für welche der Damm erstellt wurde, so handelt es sich um einen das Gelände schneidenden Kandidaten, der gelöscht werden kann (2 Kandidaten in Abbildung 24). Wird eine maximale Dammhöhe von 300 m über- bzw. eine minimale Dammhöhe von 10 m unterschritten, so werden die betroffenen Kandidaten ebenfalls aus der Auswahl entfernt. Dies entfernt nicht nur „echte“ Dammkandidaten außerhalb unserer gesuchten Dimensionen, sondern auch eine weitere Variante von „ungeeigneten Kandidaten“ – fälschlicherweise erkannte Engstellen an Sattelformen im Gelände, welche keine echten Dammstandorte repräsentieren können. Die Profile betroffener Kandidaten zeigen, dass weder Stützpunkte unterhalb noch oberhalb der Konturlinie existieren. Diese werden somit bereits durch die Anwendung der 10 m Mindest-Dammhöhen-Einschränkung entfernt.

Für die Berechnung des Dammvolumens wird je Dammkandidat in einer Schleife durch alle Stützpunkte des Profils iteriert und das Volumen aller Segmente aufsummiert. Das Volumen eines Segmentes wird für die hier angenommene Dammvariante (Abbildung 6) durch $Dammhöhe^2$ * *Segmentlänge* berechnet (2.3). Der aktuelle Schritt wurde in 5 Sekunden abgeschlossen und konnte zusätzlich zur Berechnung der Dammvolumina 46 ungeeignete Kandidaten erkennen und eliminieren.

124 Kandidaten bleiben somit erhalten (Abbildung 25). Für die 800 m Beispielkontur bleiben 9 Kandidaten erhalten (Abbildung 26).

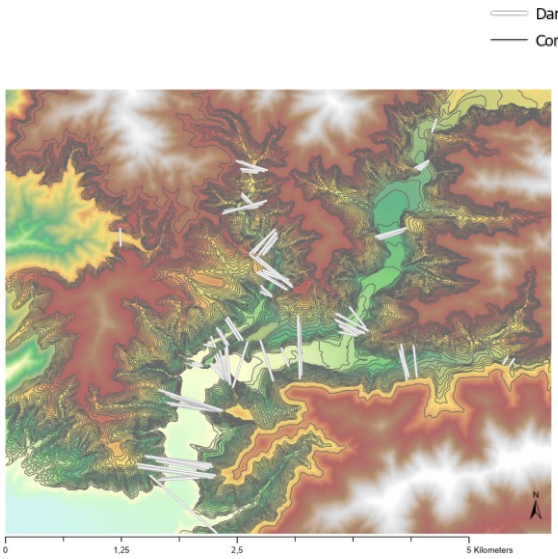


Abbildung 25: „Echte“ Dammkandidaten (keine Schnittpunkte mit Gelände)

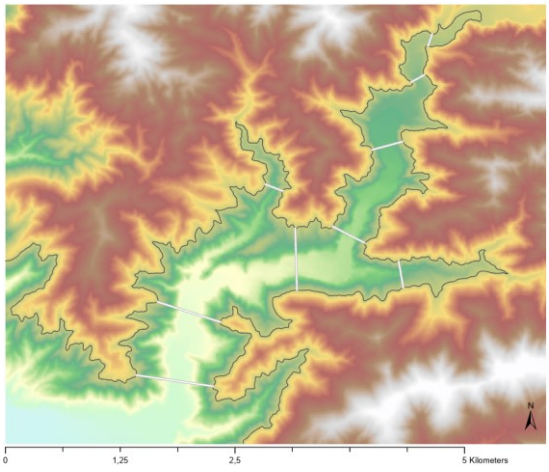


Abbildung 26: „Echte“ Dammkandidaten an 800 m Kontur

Die exakte Volumensberechnung einer Talsperre hängt maßgeblich von der gewählten Architektur des Dammes ab. Ein Erd- bzw. Steinschüttdamm benötigt durch die Trapez-Form deutlich mehr Material als eine Gewichtsstaumauer oder eine Gewölbestaumauer. In dieser Arbeit werden die potentiellen Dammstandorte jedoch nicht auf Basis des Dammvolumens eruiert, sondern dieses lediglich zur Reihung der Eignung der Kandidaten herangezogen. Um genauere Reihungen bzw. Volumensschätzungen für bestimmte Dammarchitekturen zu erhalten, müsste in diesem Schritt lediglich die entsprechende Berechnungsformel angewendet werden.

3.1.6 Reservoirpolygone erstellen

Das Volumen des durch die Errichtung einer Talsperre entstehenden Reservoirs stellt einen sehr wichtigen Kennwert dar und wird zudem in weiterer Folge bei der Paarfindung zur Berechnung der potentiellen Energiespeicherkapazität benötigt.

Als effizienteste gefundene Variante der Volumensberechnung mit den bestehenden Tools wurde das Werkzeug „Polygon Volume“ aus der 3D Analyst Toolbox identifiziert. Dieses kann alle Polygone eines Featurelayers durch Angabe eines Höhenfeldes mit einem TIN des Geländes verarbeiten und das Volumen unterhalb bzw. oberhalb des Polygons ermitteln. Für unsere Zwecke muss die Höhe (Z-Wert) des Polygons der Höhe der entsprechenden Höhenlinie, aus welcher ein Dammkandidat ermittelt wurde, entsprechen und das Volumen zwischen dieser Referenzfläche und dem Gelände berechnet werden. Ein TIN des gesamten Testgebietes wurde bereits in Schritt 3.1.3 erstellt. So ist nun lediglich ein Polygon je Dammkandidat notwendig, welches der Wasseroberfläche des jeweiligen Reservoirs bei maximaler Füllung entspricht.

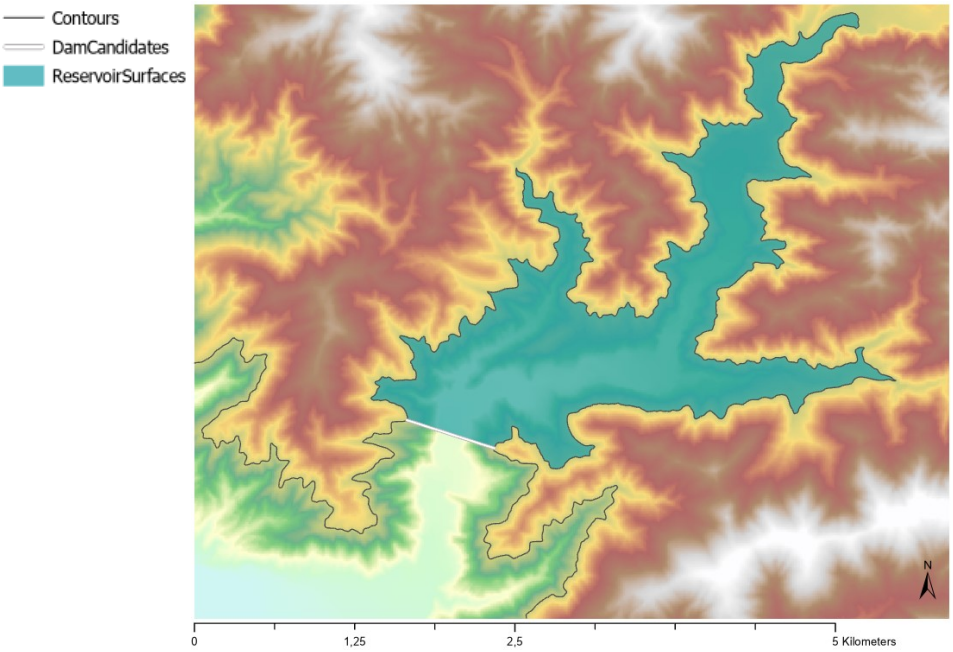


Abbildung 27: Reservoir Polygon

Ein entsprechendes Polygon ist in Abbildung 27 abgebildet. Das Polygon ist exakt durch die Höhenlinie des Reservoirs sowie durch den Damm selbst begrenzt. Da die Erstellung dieser Polygone für alle bereits eruierten Dammkandidaten nach einigen Versuchen nicht effizient mittels der bestehenden Tool-Palette umgesetzt werden konnte, wurde wiederum ein C# Add-In hierfür erstellt (A.5 CreateReservoirPolygonsButton).

Dieses erstellt aus der jeweiligen Höhenlinie eine neue Polylinie, bei der die beiden Enden durch den Start- bzw. den Endpunkt des Dammkandidaten beschnitten sind. Anschließend

kann diese Polylinie direkt in ein Polygon gewandelt werden – eine explizite Berücksichtigung des Dammes ist hierzu nicht notwendig, da die beiden offenen Enden automatisch mit einer Linie geschlossen werden. Dem Polygon wurden nun in der Featureklasse noch die ID des zugehörigen Dammkandidaten sowie die Konturhöhe angefügt. Diese einzelne Polygonerstellung stellt derzeit den zeitintensivsten Teil des Ablaufes dar – die Erstellung von 122 Reservoirpolygonen benötigte am Testgerät ca. 50 Sekunden.

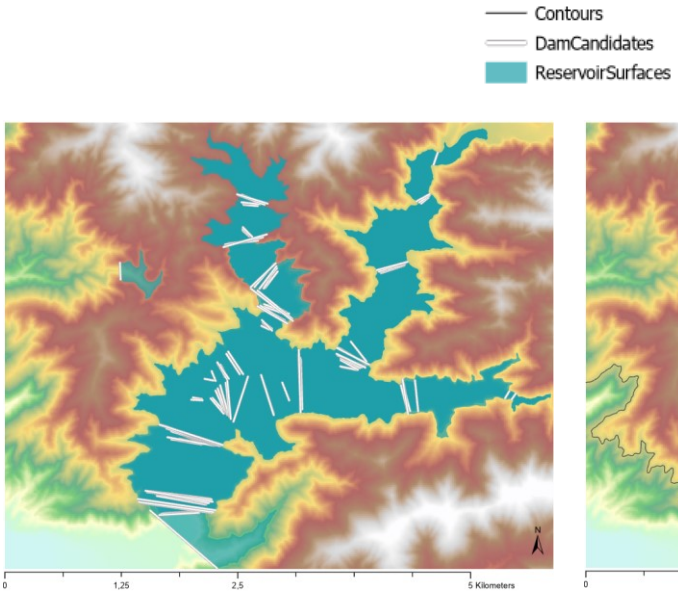


Abbildung 28: Reservoir Polygone aller Konturen

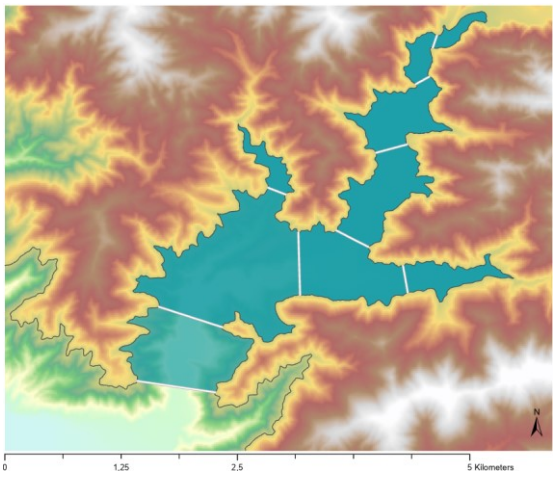


Abbildung 29: Reservoir Polygone der 800 m Kontur

3.1.7 Reservoirvolumen berechnen

Zur Ermittlung des Reservoirvolumens summierten Yi et al. (2010) alle Höhenwerte der Zellen des DHM, welche sich innerhalb einer Reservoirfläche befanden auf (Abbildung 30). Der Differenzwert zwischen $Höhe_{Wasserstand}$ * Zellenanzahl – $Summe_{DHM}$ = Reservoirvolumen.

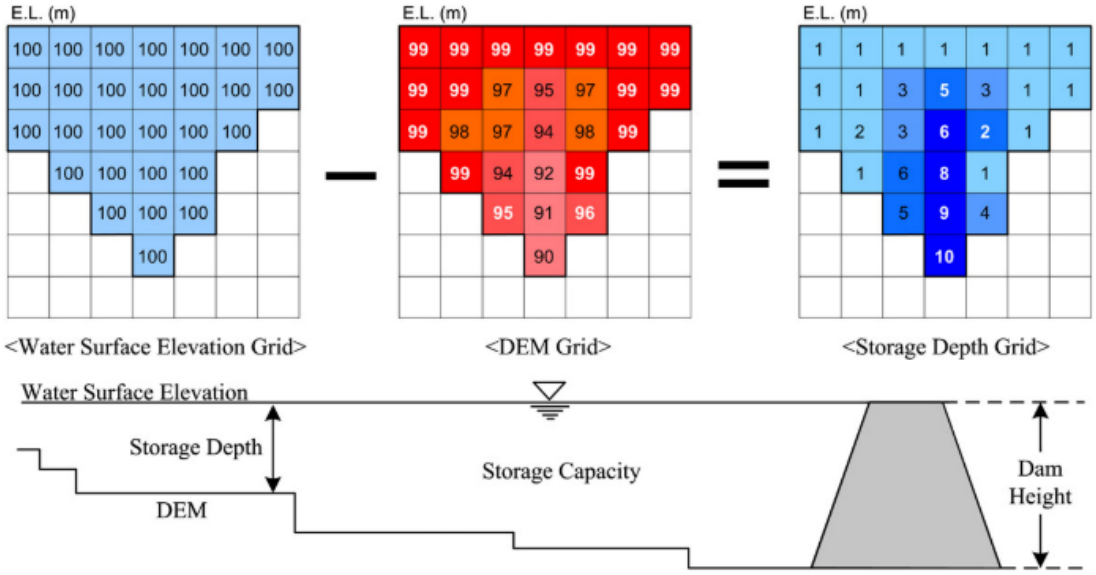


Abbildung 30: Reservoirvolumensberechnung nach Yi et al. (2010)

Dies ist ein logischer und einfacher Ansatz, benötigt jedoch ebenso eine Verschneidung des DHM mit einem Reservoir-Oberflächenpolygon, um die betroffenen Zellen zu selektieren. Die Berechnung selbst hätte schließlich in einer Schleife für jedes Polygon einzeln durchgeführt werden müssen – zumindest war kein anderer Ansatz bekannt. Diese iterativen Prozesse mit jeweiligen Aufrufen bestehender Tools haben durch den Overhead der vielfachen Toolinitialisierungen naturgemäß eine schlechtere Performance, als bei einem einzelnen Aufruf eines Werkzeuges, das die Verarbeitung mehrerer Features erlaubt. Das Tool „Polygon Volume“ unterstützt dies bereits und wurde somit für diese Aufgabe verwendet.

Da durch den vorherigen Schritt nun beide notwendigen Parameter (das TIN sowie die Reservoir Polygone) für das Tool „Polygon Volume“ zur Verfügung stehen, kann dieses in einem Modell ausgeführt (A.6 ReservoirVolumeAndRankingButton) und die Ergebnisse der Berechnung in die Dammkandidaten-Features übertragen werden (Abbildung 31).

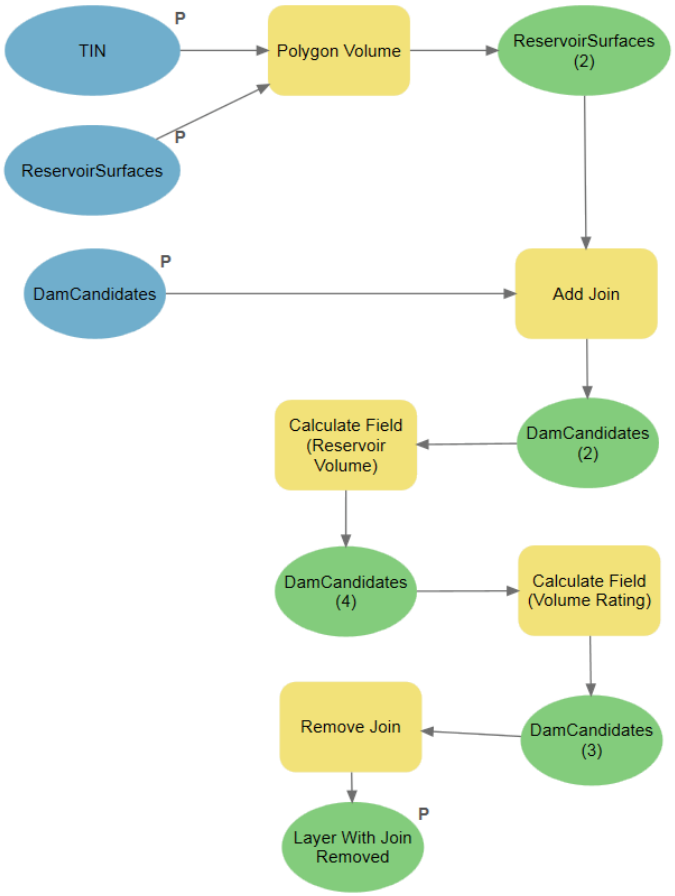


Abbildung 31: Modell „Reservoir Volumen“

Das Tool „Polygon Volume“ fügt der Featureklasse ReservoirSurfaces ein neues Feld „Volume“ mit dem berechneten Volumenswert in m³ hinzu. Durch einen Join dieser Ergebnistabelle mit den Dammkandidaten kann dieser Wert in die Dammkandidaten-Ebene übernommen werden. Schließlich wird eine einfache Bewertung anhand des Verhältnisses von Reservoirvolumen zu Dammvolumen durchgeführt und dieses ebenfalls bei den Dammkandidaten hinterlegt (Tabelle 5). Anhand dieser Kennzahl kann nun eine Reihung der Kandidaten nach theoretischer Eignung (möglichst geringes Dammvolumen mit möglichst hohem Reservoirvolumen) durchgeführt werden. Grundsätzlich könnten diese Ergebnisse auch bereits für die Errichtung eines einfachen Speicherkraftwerkes herangezogen werden. Da diese jedoch lediglich von natürlichem Wasserzufluss gespeist werden, wäre hier zumindest eine weitere Analyse des Einzugsgebietes und der erwarteten Niederschlagsmengen durchzuführen um eine diesbezügliche Eignung bewerten zu können.

Das Modell konnte innerhalb von 35 Sekunden das Volumen aller Reservoire ermitteln und die Volumensbewertung durchführen.

Tabelle 5: Ergebnistabelle der Dammkandidaten für Kontur 800

Contour Height	Circum-ference	Dam Length	Length Rating	Dam Height	Dam Volume	Reservoir Volume	Volume Rating
800	24'000	727	33.00	181	11'408'728	254'569'835	22.31
800	26'900	867	31.00	203	16'190'149	347'103'846	21.44
800	9'800	390	25.08	115	2'628'276	55'433'822	21.09
800	15'400	669	23.00	142	5'870'424	114'334'758	19.48
800	2'900	159	18.23	49	216'718	3'582'716	16.53
800	6'300	351	17.91	75	1'355'485	22'855'566	16.86
800	2'400	202	11.83	77	418'932	3'195'784	7.63
800	1'500	135	11.05	24	43'908	565'085	12.87
800	2'800	271	10.30	73	548'863	5'231'792	9.53

3.1.8 Reservoirpaare bilden

Sobald eine Auswahl an möglichen Reservoiren in einem Gebiet zur Verfügung steht, kann unter Anwendung von einigen Einschränkungen nach geeigneten Reservoir-Paaren für die Errichtung eines PSKW gesucht werden.

Konkret werden räumlich naheliegende Reservoir-Paare mit möglichst großer Höhendifferenz und ähnlichem Volumen gebildet, deren Reservoir Polygone sich jedoch nicht überschneiden dürfen.

Dies lag nicht direkt im Fokus dieser Arbeit, war jedoch für eine Validierung der Ergebnisse der vorangegangenen Schritte sinnvoll.

Tabelle 6: Parameter für Suche nach Reservoir - Paaren

Beschreibung	Wert
Maximale Distanz zwischen Dammmittelpunkten	5.000 m
Minimal nutzbarer Höhenunterschied	50 m
Maximaler Volumensunterschied zwischen Reservoiren	25%

Hierzu wurde wiederum ein kurzer C# Algorithmus umgesetzt (A.7 PairReservoirsButton), welcher die Parameter aus Tabelle 6 auf die zuvor eruierten Dammkandidaten anwendet und entsprechend 2.1 einen Richtwert für die durch die jeweilige Kombination zweier Reservoire theoretisch mögliche Energiespeicherkapazität errechnet.

Die horizontale Distanz zwischen den Reservoiren wurde wie durch Gimeno-Gutiérrez und Lacal-Arántegui (2013) über die Dammmittelpunkte berechnet und ist ein wichtiger Faktor

zur Eingrenzung der möglichen Kombinationen sowie zur Reihung aller zutreffenden Kombinationen. Da in unserem Test- und Untersuchungsgebiet eine relativ hohe Dichte an möglichen Reservoiren zur Verfügung steht, wurde der Parameter für die maximale Distanz zwischen zwei Dammmittelpunkten wie auch durch Fitzgerald et al. (2012) und Soha et al. (2017) auf 5 km gesetzt. Falls in diesem Schritt keine zufriedenstellende Auswahl an PSKW-Kandidaten ermittelt werden kann, ist die Erhöhung dieses Limits auf bis zu 20 km üblich (Rogeau et al. 2017; Lu et al. 2018; Lacal-Arántegui und Tzimas 2012). Der Dammmittelpunkt wurde gewählt, da dieser in der Regel dem tiefsten Punkt eines durch eine Talsperre erstellten Reservoirs entspricht und somit den Entnahmeort im oberen Reservoir bei Generator- und im unteren Reservoir bei Pumpbetrieb darstellt.

Da sich der Höhenunterschied maßgeblich auf die erreichbare Energiespeicherkapazität auswirkt, wurde hier ein Limit von mindestens 50 m Höhendifferenz gewählt. Die hier berücksichtigte Höhendifferenz wurde durch die Differenz zwischen dem tiefsten Punkt des oberen Reservoirs (Konturhöhe – Dammhöhe) und dem höchsten Punkt des unteren Reservoirs gebildet. Dies ist ein konservativer Ansatz, welcher die geringste mögliche Höhe im Betrieb des PSKW-Kandidaten beschreibt, wenn das obere Reservoir beinahe geleert und das untere beinahe voll gefüllt ist. Die tatsächlich erreichbare Höhendifferenz liegt im Mittel wohl etwas darüber, hängt jedoch von der genauen Form der Reservoire und auch vom tatsächlichen Wasserstand während des Betriebes ab.

Für die Berechnung der Energiespeicherkapazität wurde das Volumen des kleinvolumigeren Reservoirs herangezogen und entsprechend Lu et al. (2018) ein Abschlag von 15% angewendet um Mindestwasserstände zu berücksichtigen.

Da viele Dammkandidaten unterschiedlicher Konturhöhen räumlich sehr eng beisammen liegen (da sich eine Engstelle im Tal meist auf mehrere Höhenlinien auswirkt), gibt es viele räumlich ähnliche Kombinationen, die sich jedoch durch Dammhöhe und Volumen der einzelnen Reservoirs unterscheiden. Hier ist der logische Ansatz eher zwei Reservoirs zu verbinden, welche über ein ähnliches Volumen verfügen, da ansonsten der überschüssige Teil des größeren Reservoirs nicht genutzt werden kann. Hierdurch werden effizienter Weise speziell für die unteren Reservoire meist kleinere Dammhöhen ausgewählt als an diesem Standort möglich wäre. Falls andere Überlegungen wie beispielsweise eine Kombination mit konventioneller Wasserkraft angestellt würden, müsste der Algorithmus entsprechend angepasst werden.

Die gewählten PSKW-Kandidaten werden inklusive aller eruierten Eckdaten wie Energiespeicherpotential in MWh und Distanz in Metern als Linien-Features in der Datenbank abgelegt (Tabelle 7). Innerhalb des Testgebietes wurden mittels obiger Methode innerhalb

von 4 Sekunden 51 PSKW-Kandidaten gefunden (Abbildung 32). Die Linien visualisieren die Verbindung zwischen den Mittelpunkten der zwei Dämme der PSKW-Kandidaten.

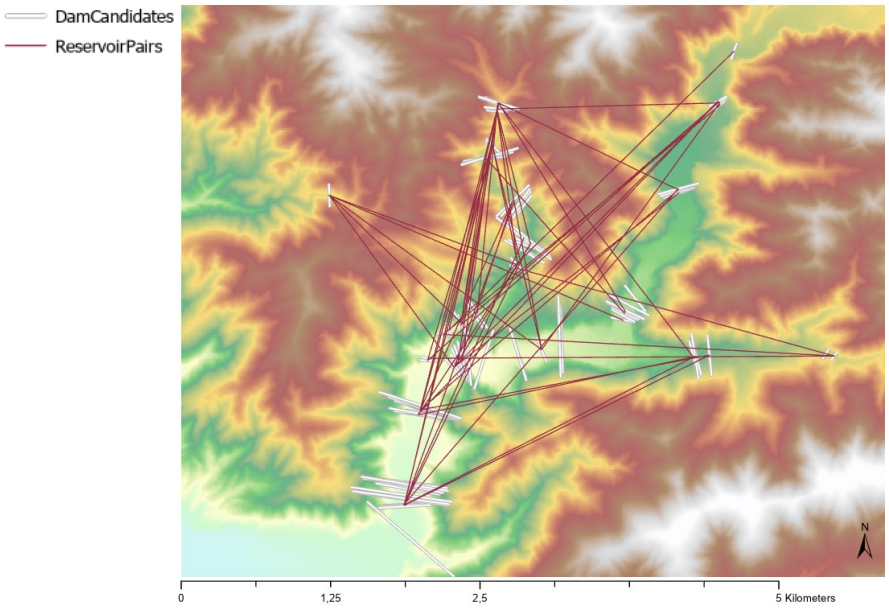


Abbildung 32: Reservoir Paare

Tabelle 7: Eckdaten der 10 PSKW-Kandidaten mit der höchsten Kapazität

Lower DamID	Upper DamID	Capacity In MWh	Dis-tance	Lower Height	Upper Height	Capacity Distance Ratio	Usable Height Difference	Capacity Utilization
354	516	5.525,42	3.000	660	940	184,13	140	90%
368	516	4.825,90	2.185	690	940	220,86	110	89%
352	521	4.706,80	3.456	650	960	136,18	198	90%
377	516	3.948,46	1.728	710	940	228,42	90	89%
359	521	3.932,85	2.650	670	960	148,36	178	93%
369	521	3.755,93	2.200	690	960	170,67	158	97%
358	481	2.961,38	2.441	670	890	121,28	53	82%
352	477	2.927,25	2.707	650	880	108,13	111	99%
363	482	2.684,82	1.909	680	890	140,62	81	98%
358	445	2.585,57	3.499	670	800	73,87	55	84%

3.1.9 Visualisierung

Um die Ausmaße eines einzelnen Reservoirs (Abbildung 33) oder eines PSKW-Kandidaten (Abbildung 34) schnell erfassbar zu machen, wurde eine letzte C# Implementierung (A.8 VisualizeDamButton) zur Visualisierung einzelner Dämme/Reservoirs umgesetzt. Hier werden alle selektierten Dämme der Tabelle „DamCandidates“ sowie alle selektierten PSKW-Kandidaten der Tabelle „ReservoirPairs“ durch Erstellung neuer 3D Features visualisiert. Für einen Damm wird anhand der gespeicherten Dimensionsdaten ein Multipatch Feature an der Position des Dammes erstellt. Für die Wasseroberfläche werden die entsprechenden Reservoirpolygone aus dem bereits bestehenden Layer kopiert.

Bei jeder Anwendung der Visualisierung werden zuvor erstellte Objekte innerhalb der Visualisierungsebene entfernt, sodass immer nur die aktuelle Selektion sichtbar gemacht wird.

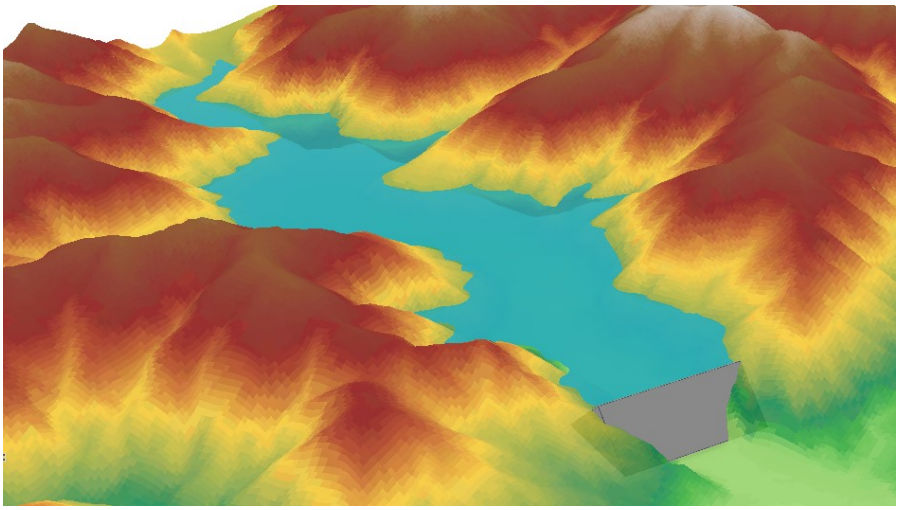


Abbildung 33: 3D Visualisierung eines Reservoirs

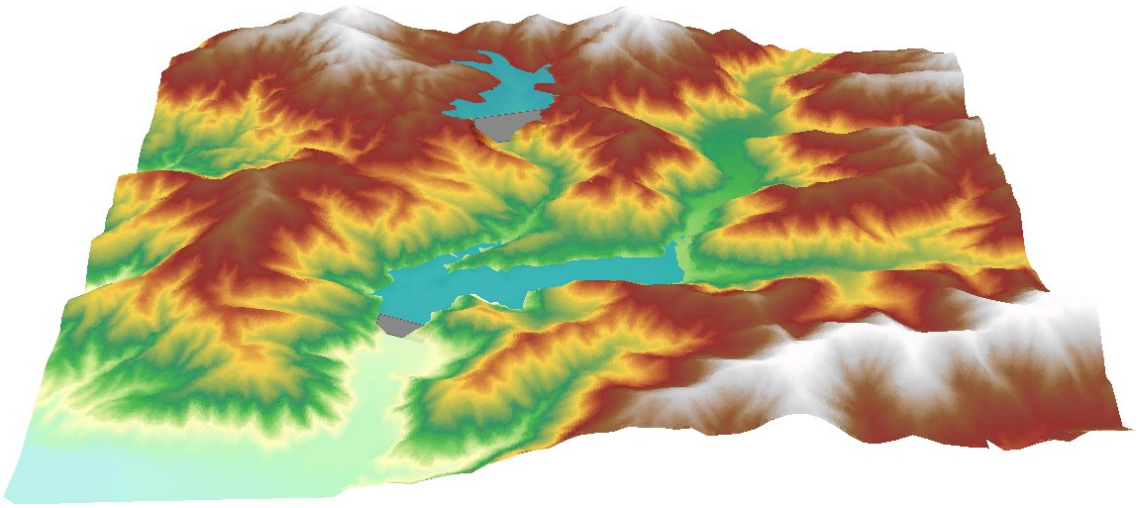


Abbildung 34: 3D Visualisierung eines PSKW – Kandidaten

3.2 Daten

Wie Eingangs unter 1.2.4 erwähnt werden in der bestehenden Literatur oftmals SRTM Datensätze für die Geländeanalyse in Dammstandortfindungs-Projekten herangezogen. Dies hat vor allem den Hintergrund, dass diese Daten für einen Großteil der Landmasse weltweit zur Verfügung stehen (Abbildung 39) und zudem kostenfrei bezogen werden können. Eine eigene Erstellung eines DHM durch Laserscanbefliegung eines Gebietes kann sehr kostspielig sein und sollte daher speziell für große Landstriche, wenn möglich vermieden werden. Deshalb ist ein Kernpunkt dieser Arbeit, zu klären ob die SRTM Daten (3.2.1) mit Auflösung von einer Bogensekunde (ca. 30 m am Äquator) für den vorgestellten Algorithmus geeignet sind. Für eine Vergleichsanalyse wurden öffentlich verfügbare Höhendaten des Landes Vorarlberg (3.2.2) mit einer Auflösung von 5 m herangezogen. Die effektive Auflösung der SRTM Datensätze in Metern hängt vom jeweiligen Breitengrad des Standortes ab, da die Distanz zwischen den Längengraden abnimmt, je weiter man sich vom Äquator entfernt. In der Literatur (siehe Tabelle 2) wird jedoch gemeinhin die Auflösung für die SRTM Daten mit 30 m bzw. 90 m bezeichnet (1 bzw. 3 Bogensekunden Variante). Dies wurde auch in dieser Arbeit so beibehalten.

3.2.1 SRTM Datensätze (30 m)

Die **S**huttle **R**adar **T**opography **M**ission war eine im Jahr 2000 durchgeführte Fernerkundungsmission aus dem Weltall in welcher innerhalb von 11 Tagen vom Shuttle Endeavour aus über Radar-Antennen die Daten für ein weltweit einheitliches Höhenmodell gesammelt wurden. Die Daten stehen selbst für entlegene Regionen zur Verfügung (Abbildung 35) und sind kostenlos von der Webseite des USGS beziehbar.

Abbildung 35: Weltweite SRTM Datenverfügbarkeit markiert
(USGS - U.S. Geological Survey)

Das DHM-Raster wurde mittels „Project Raster“ Tool aus dem geographischen Koordinatensystem „GCS WGS 1984“ in das projizierte Koordinatensystem „MGI Austria GK West“ gewandelt. Dies ist für den Algorithmus notwendig, welcher die Einheit Meter für die Anwendung der Grenzwerte und Berechnung der Distanzen und Volumen mittels Koordinaten voraussetzt.

Raster Information

Columns	2227
Rows	3389
Number of Bands	1
Cell Size X	25,6498354192609
Cell Size Y	25,6498354192614
Uncompressed Size	14,40 MB
Format	FGDBR
Source Type	Generic
Pixel Type	unsigned short
Pixel Depth	16 Bit
NoData Value	
Colormap	absent
Pyramids	level: 5, resampling: Nearest Neighbor
Compression	LZ77
Mensuration Capabilities	Basic

Abbildung 36: Raster Informationen des SRTM Datensatzes

Zu beachten ist hier, dass auch die aktuellsten „void filled“ Versionen in einigen Kacheln noch Lücken aufweisen können. Diese müssen vor der Analyse bereinigt werden.

3.2.2 VoGIS Datensätze (5 m)

VoGIS ist das offizielle GIS-Portal des Landes Vorarlberg, welches nebst eines online Atlas auch eine Vielzahl an Rohdaten zum freien Download anbietet (Land Vorarlberg 2019). Hier wurde das DHM mit 5 m Auflösung bezogen, welches für die Vergleichsanalyse mit den SRTM-Datensätzen herangezogen wurde.

Raster Information

Columns	11500
Rows	17500
Number of Bands	1
Cell Size X	5
Cell Size Y	5
Uncompressed Size	767,71 MB
Format	IMAGINE Image
Source Type	Generic
Pixel Type	floating point
Pixel Depth	32 Bit
NoData Value	-9999
Colormap	absent
Pyramids	level: 8, resampling: Nearest Neighbor
Compression	RLE
Mensuration Capabilities	Basic

Abbildung 37: Raster Informationen des VoGIS Datensatzes

Das originale DHM-Raster gt2011_05m.img wurde mittels „Define Projection“ auf das in der den Daten beigefügten .prj Datei definierten geographische Koordinatensystem „Transverse_Mercator“ definiert und anschließend wieder über das „Project Raster“ Tool in das projizierte Koordinatensystem „MGI Austria GK West“ gewandelt. Das Höhenmodell war ansonsten bereits einsatzfähig und wies keinerlei Lücken oder andere erkennbare Fehler auf.

3.3 Untersuchungsgebiet

Als Untersuchungsgebiet wurde für diese Arbeit das gesamte Bundesland Vorarlberg herangezogen. Der Grund für die Wahl dieses Gebietes lag einerseits bei dem hier vorkommenden bergigen Gelände, für welches die Eignung zur Errichtung von Talsperren grundsätzlich zu erwarten ist und andererseits bei der geographischen und emotionalen Nähe als Herkunftsland des Autors.

Das Gebiet umfasst eine Fläche von ca. 2.601 km² sowie Höhenwerte zwischen 372 m und 3208 m (SRTM Datensätze aus 3.2.1) bzw. zwischen 393 m und 3312 m (VoGIS Datensätze aus 3.2.2). Es bestehen bereits mehrere Wasser- und auch Pumpspeicherkraftwerke in Vorarlberg – mit dem Obervermuntwerk II wurde erst 2019 das bisher letzte offiziell eröffnet.

Für die erste Analyse auf Basis der SRTM Datensätze wurden im USGS Earth Explorer die 4 Mosaikteile bezogen, welche das Untersuchungsgebiet abdecken (Abbildung 38). Diese wurden durch das Werkzeug „Mosaic to New Raster“ zu einem Raster zusammengeführt und auf das Koordinatensystem „MGI Austria GK West (EPSG:31254)“ projiziert.

Zur Eingrenzung des Gebietes wurde ein Polygon der Verwaltungsgrenze von Vorarlberg aus dem Downloadbereich des VoGIS (Land Vorarlberg 2019) verwendet – in Abbildung 38 in Rot dargestellt.

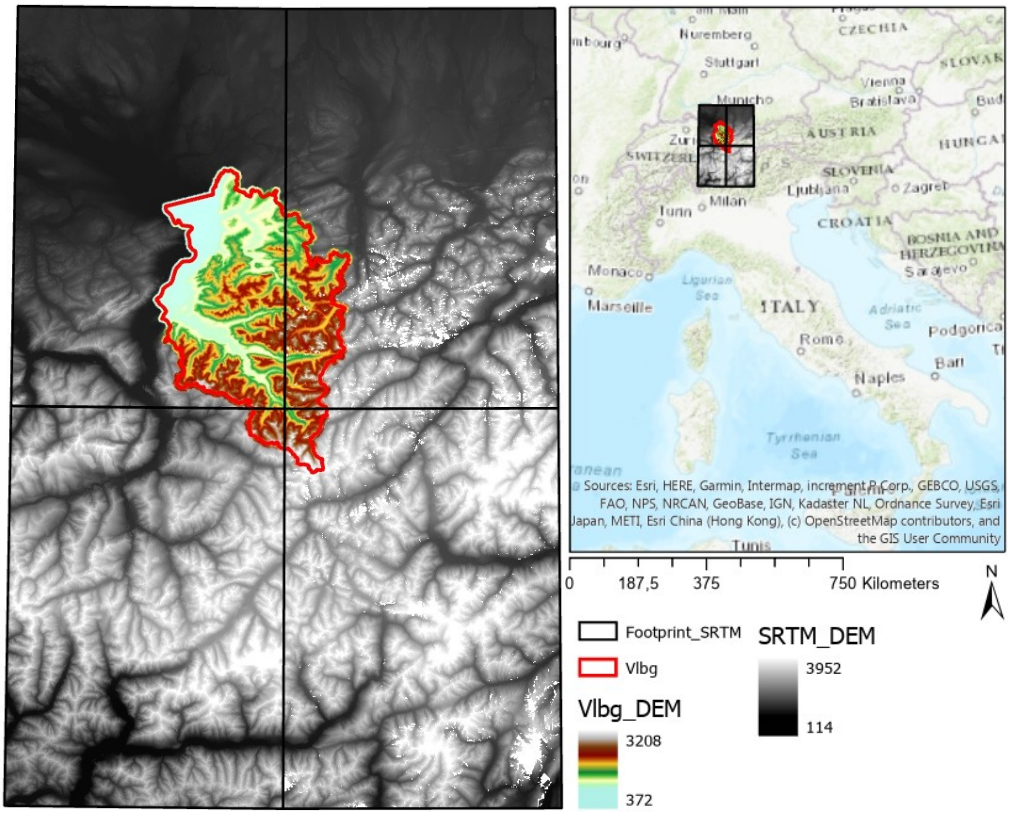


Abbildung 38: 4 SRTM Kacheln decken das Untersuchungsgebiet ab

Da die zwei östlich gelegenen Kacheln über einige Lücken verfügten (siehe Abbildung 38 in Weiß), wurde in Vorbereitung auf die Interpolation der Daten das Polygon mit der Verwaltungsgrenze mittels 1 km Buffer vergrößert und das Raster darauf beschnitten. Durch die bestehende Raster Surface Function „Elevation Void Fill“ wurden darauffolgend die Lücken im Raster durch Ebenenanpassung/IDW „aufgefüllt“ und abschließend auf das originale Verwaltungsgrenzen-Polygon beschnitten (Abbildung 39, links).

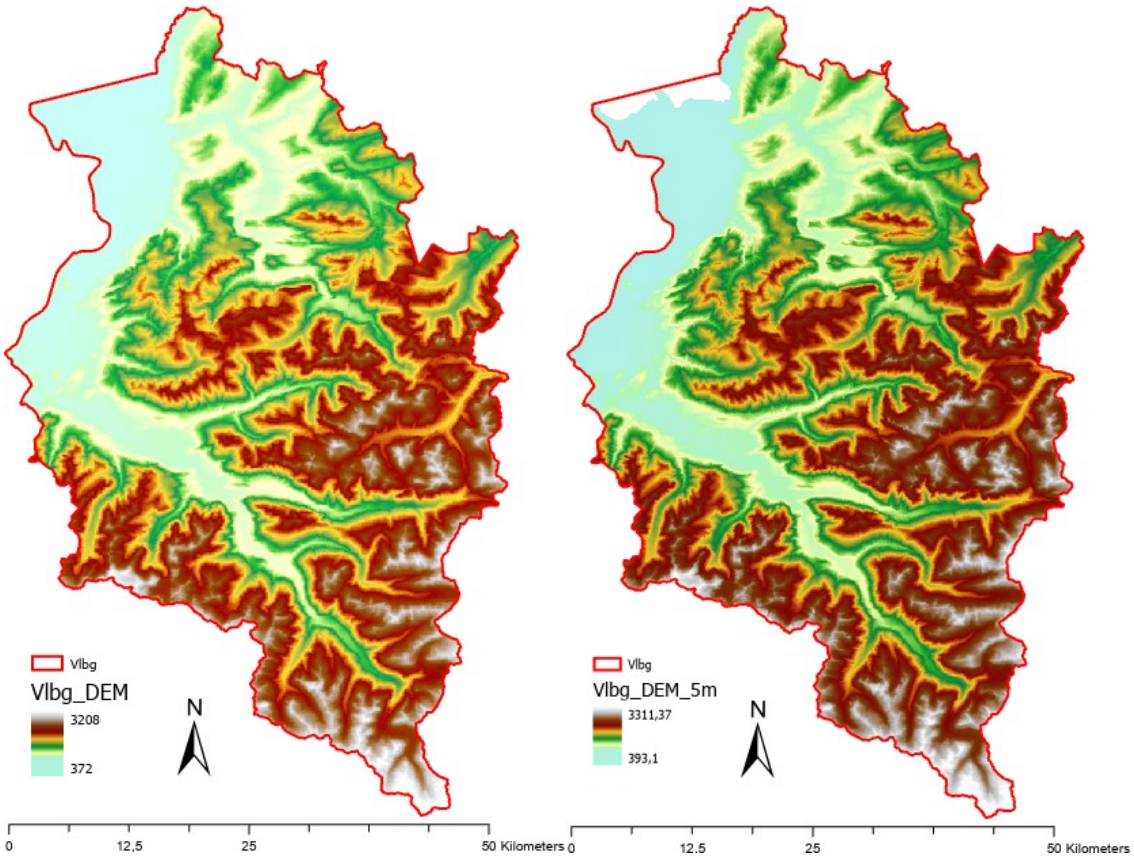


Abbildung 39: SRTM-DHM (links) und VoGIS-DHM (rechts)

Für die zweite Analyse auf Basis der VoGIS Datensätze war lediglich der Schritt der korrekten Projektion in das Koordinatensystem „MGI Austria GK West (EPSG:31254)“ notwendig, da der Datensatz bereits in exakter Abmessung vorlag und keine Lücken vorhanden waren. Lediglich der Anteil des Bodensees war in diesen Daten nicht enthalten (Abbildung 39, rechts), was jedoch für diese Arbeit nicht von Relevanz war.

Es ist zu beachten, dass gemäß Zielsetzung dieser Arbeit keinerlei Einschränkung auf das zu analysierende Gebiet angewendet wurde und somit auch Reservoire in eigentlich bewohnten oder naturgeschützten Bereichen eruiert wurden. Bei konkreter Suche nach tatsächlich realisierbaren Reservoiren müsste dies selbstverständlich berücksichtigt und betroffene Kandidaten aussortiert bzw. die Analyse für betroffene Bereiche gar nicht durchgeführt werden.

4 Ergebnisse

In diesem Kapitel werden die Ergebnisse der Analysen des Untersuchungsgebietes dargelegt. Die gewählten Parameter entsprechen – soweit nicht abweichend erwähnt – den Parametern lt. Tabelle 3, Tabelle 4 und Tabelle 6.

Analyse A (4.1) wurde auf Basis der SRTM-Datensätze durchgeführt. Für einen Vergleich der Ergebnisse mit einem hochauflösenderen DHM wurde in Analyse B (4.2) die VoGIS Datenbasis verwendet. Die Gegenüberstellung der Ergebnisse findet sich unter 4.3. Aufgrund der hohen gefundenen Anzahl an Dammkandidaten innerhalb des gesamten Untersuchungsgebietes wurde für einen detaillierten Vergleich weiters eine Analyse auf eine einzelne Höhenlinie durchgeführt und zusätzlich zu den beiden unterschiedlichen DHM Grundlagen auch eine Variante mit 5 m Prüfpunktintervall in den Vergleich miteinbezogen.

Hardware-Setup:

Da die Arbeit auch die Performanz des Algorithmus untersucht und die entsprechenden Berechnungszeiten ausgewertet wurden, ist das Hardware-Setup zu erwähnen:

Personal Computer:	3.6GHz Intel i7 8 Core, 16GB RAM, SSD HDD

Software-Setup:

Betriebssystem:	Win10 64Bit
GIS Software:	ArcGIS Pro 2.4.1
Entwicklungsumgebung:	Visual Studio Community 2019 mit ArcGIS Pro SDK Extension

4.1 Analyse A (30 m DHM)

In Analyse A wurde der Algorithmus auf die SRTM-Daten (3.2.1) angewendet.

Innerhalb der 2.601 km² des Untersuchungsgebietes wurden 1.456 Konturlinien zwischen 400 m und 2.790 m Höhe und insgesamt 3.400.414 Prüfpunkte erstellt (gemäß 3.1.2). Die Erstellung der Konturlinien benötigte dabei ca. 30 Sekunden, die der Prüfpunkte hingegen 7 Stunden. Durch den Algorithmus zur Findung der Engstellen wurden aus 15.920.915.763 möglichen Punktkombinationen innerhalb von 1 Stunde und 40 Minuten 5.011 Dammkandidaten eruiert (gemäß 3.1.4). Die Berechnung der Dammvolumen gem. 3.1.5 wurde in 40 Sekunden durchgeführt. Zudem wurden hier 2.841 ungeeignete (durch das Gelände verlaufende) Kandidaten entfernt. Die Erstellung der Reservoir Polygone für alle 2.170 verbleibenden Dammkandidaten durch den in 3.1.6 erörterten Ablauf nahm 42 Minuten in Anspruch. Nach der abschließenden Berechnung der Reservoirvolumen innerhalb von 5 Minuten standen neben den Eckdaten wie Ort, Höhe, Umfang, Dammlänge und Fassungsvermögen ebenso die Längen- bzw. Volumensreihungen für alle 2.170 Dammkandidaten zur Verfügung.

Die Suche nach geeigneten Reservoirpaaren entsprechend 3.1.8 für die potentielle Errichtung von neuen PSKW ergab 745 mögliche Kombinationen mit Energiespeicherkapazitäten von 32 MWh bis 331 GWh und konnte in 20 Sekunden abgeschlossen werden.

Eine Darstellung aller gefundenen Dammkandidaten, den korrespondierenden Reservoiroberflächen und Reservoirpaaren ist in Abbildung 40 ersichtlich. In Abbildung 41 ist der eruierte Dammkandidat und im Vergleich dazu in Abbildung 42 das aus Google Earth in selber Perspektive erstellte Bild des tatsächlich an dieser Position existierenden Lünersee Speichersees abgebildet. In Abbildung 43 ist die Visualisierung eines Reservoirpaares im Montafon dargestellt.

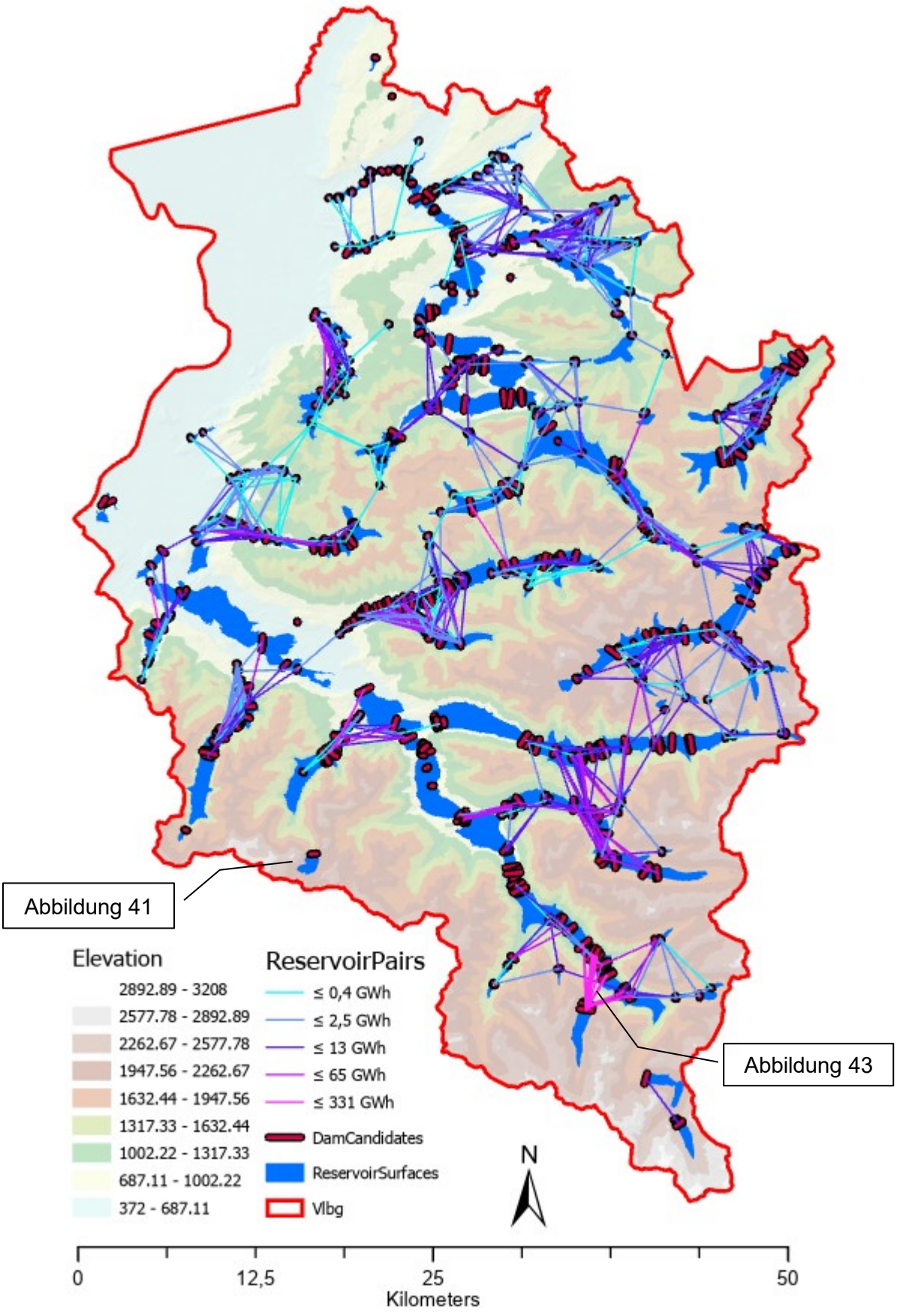


Abbildung 40: Dammkandidaten, Reservoirflächen und -Paare aus Analyse A

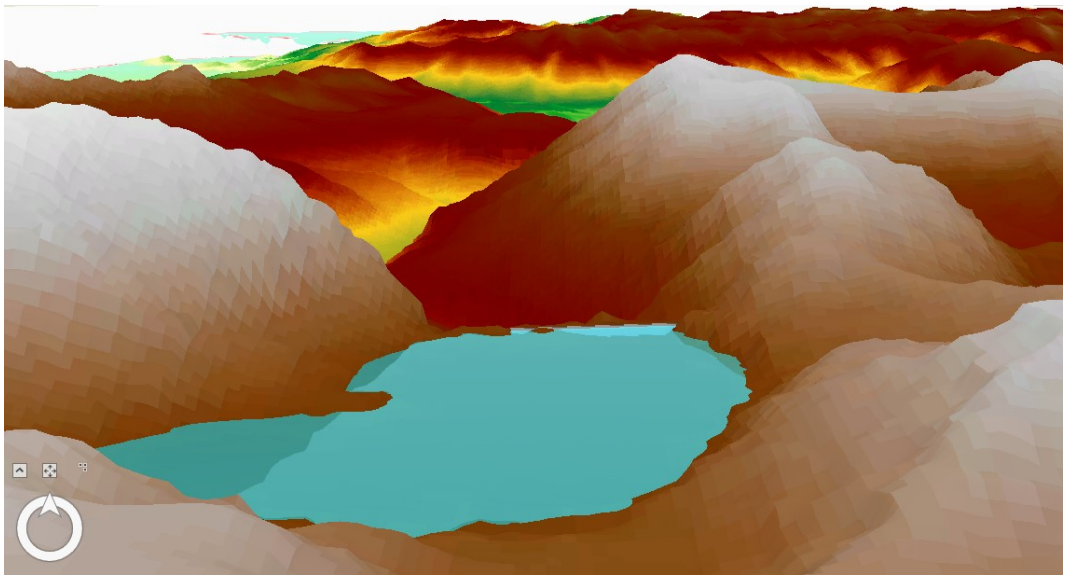


Abbildung 41: Dammkandidat an Position des Lünersee Staudammes aus 4.1

Abbildung 42: Lünersee Stausee aus Google Earth

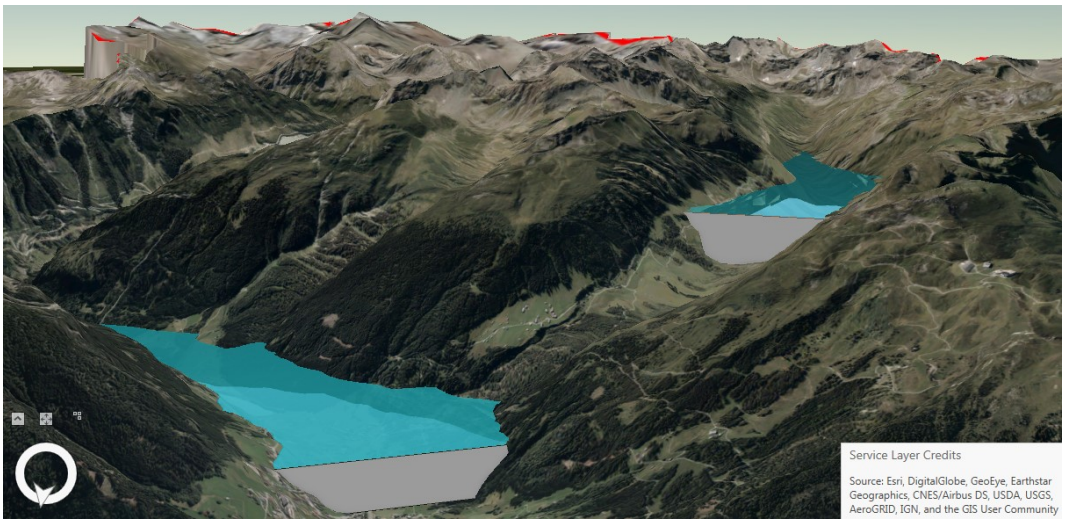


Abbildung 43: Visualisierung eines Reservoirpaares bzw. potentiellen PSKW

4.2 Analyse B (5 m DHM)

Als Datengrundlage für Analyse B wurde der VoGIS Datensatz (3.2.2) herangezogen. Das Vorgehen war ansonsten ident zur vorhergehenden Analyse. Die Anzahlen und Berechnungszeiten im Vergleich zu Analyse A sind der Tabelle 8 und Tabelle 9 zu entnehmen.

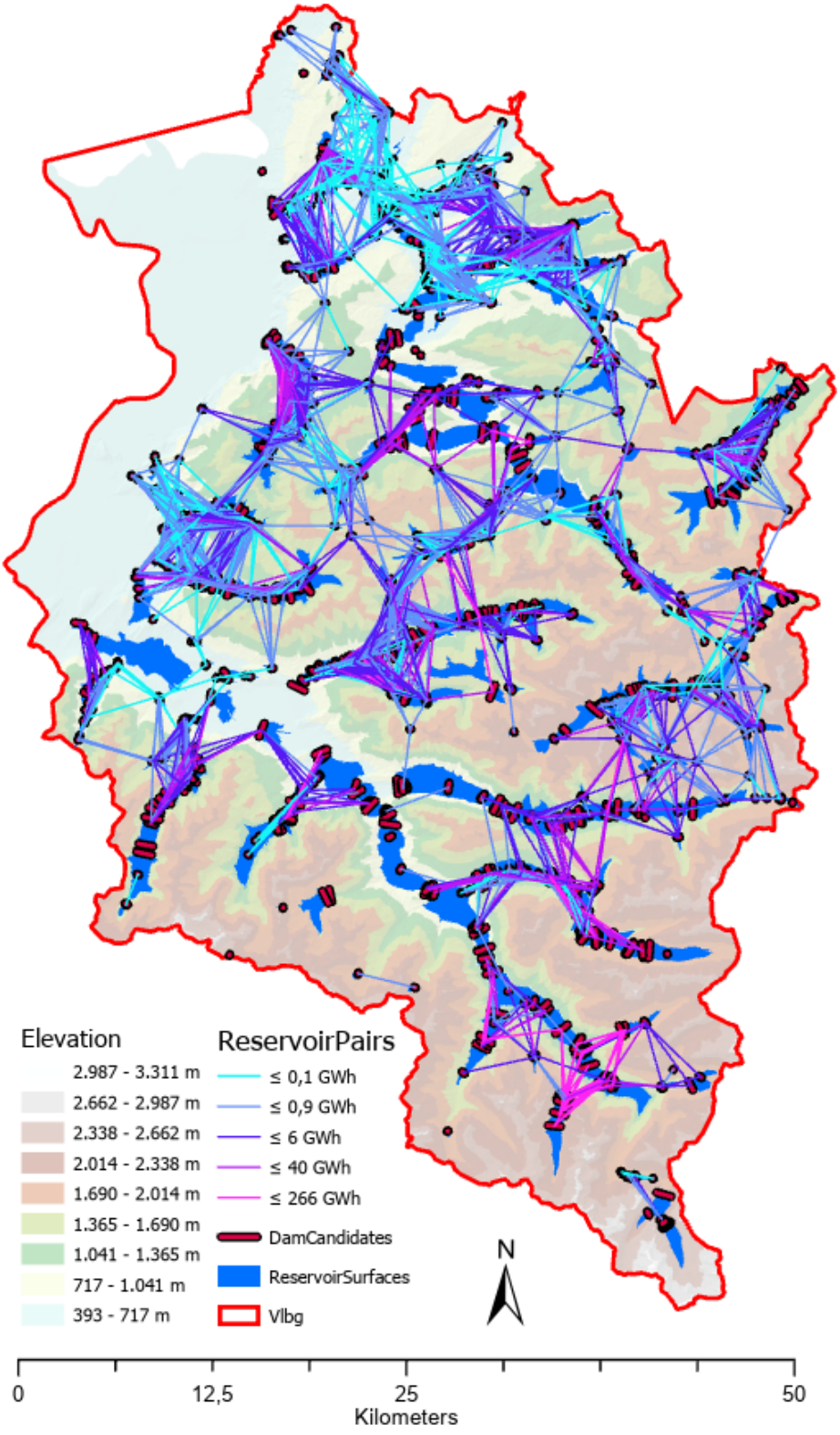


Abbildung 44: Dammkandidaten, Reservoirflächen und -Paare aus Analyse B

4.3 Gegenüberstellung A - B

Tabelle 8: Ergebnisvergleich Analyse A - B

	Analyse A	Analyse B
Dammkandidaten	2.170	3.534
Reservoirhöhen	440 m – 2.270 m	420 m – 2.460
Reservoirvolumen	92 ML - 1.637 GL	32 ML – 1.040 GL
Median der Reservoirvolumen	45,2 GL	14,6 GL
Dammvolumen im Bereich	3 ML - 45,3 GL	0,3 ML – 55,8 GL
Median der Dammvolumen	1,8 GL	713,5 ML
Anzahl eruierter Reservoirpaare	746	2.782
Energiespeicherkapazitäten	32 MWh – 331 GWh	6 MWh – 266 GWh
Median der Energiespeicherkapazitäten	1,7 GWh	448,6 MWh

Tabelle 9: Berechnungszeiten und Anzahlen für 30 m und 5 m Daten

Prozess-Schritt	**Analyse A**		**Analyse B**	
	Dauer	Anzahlen	Dauer	Anzahlen
Konturlinien & Prüfpunkte	420 Min.	1.456 KL 3.400.414 PP	3.700 Min.	1.702 KL 4.014.006 PP
TIN erstellen	< 1 Min.	602.878 SP	2 Min.	1.347.333 SP
Damm–Kandidaten	51 Min.	4.933 DK 15.920.915.763 PK	67 Min.	7.680 DK 20.722.006.014 PK
Dammvolumen	< 1 Min.	**2.170 DK** verbleibend	2 Min.	**3.534 DK** verbleibend
Res. Polygone	42 Min.	2.170 P	1.166 Min.	3.534 P
Res. Volumen	5 Min.		28 Min.	
Res. Paare	< 1 Min.	**745 RP** 2.353.365 RK	< 1 Min.	**2.782 RP** 6.242.811 RK
Summe	**≈ 520 Min. (≈ 9 Std.)**		**≈ 4.966 Min. (≈ 83 Std.)**	

Abkürzungen: Konturlinie (KL), Prüfpunkt (PP), Stützpunkt (SP), Dammkandidat (DK), Punktkombination (PK), Polygon (P), Reservoir-Paar (RP), Reservoirkombination (RK)

Wie in Tabelle 9 zu erkennen ist, benötigte die Analyse B die 9-fache Berechnungszeit der Analyse A. Die Anzahlen der einzelnen Elemente waren allesamt in Analyse B höher – mit dem größten relativen Unterschied bei den eruierten Reservoirpaaren, welche beinahe in 4-facher Anzahl (im Vergleich zur Analyse A) gefunden wurden. Aus den Median-Werten (Tabelle 8) lässt sich ableiten, dass mittels der VoGIS-Daten (Analyse B) deutlich mehr kleindimensionierte Reservoire ermittelt wurden als mit den SRTM-Daten (Analyse A). Die Ursache dieser Unterschiede wird in der nachfolgenden Diskussion genauer betrachtet.

In Abbildung 45 ist die Verteilung der Reservoire nach Konturhöhen für beide Analysen zu sehen. Abgesehen von den quantitativen Unterschieden ist der Verlauf sehr ähnlich.

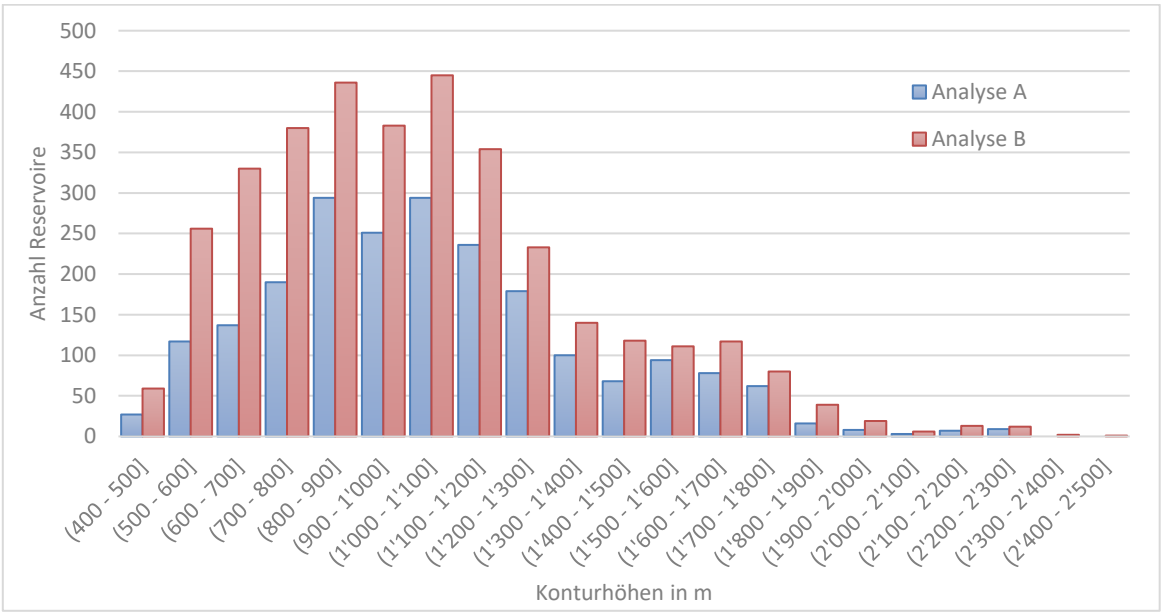


Abbildung 45: Anzahl Reservoire nach Konturhöhe

Weiters ist in Abbildung 46 eine Verteilung nach Reservoirvolumen dargestellt und der Volumensbereich, in dem sich die Ergebnisse der beiden Analysen am stärksten unterscheiden in feineren Schritten aufgelöst in Abbildung 47 abgebildet. Dies unterstreicht die Erkenntnis aus der Tabelle 8, in welcher bereits über die Mediane erkennbar war, dass Analyse B im unteren Volumensbereich mehr Reservoire ermittelte.

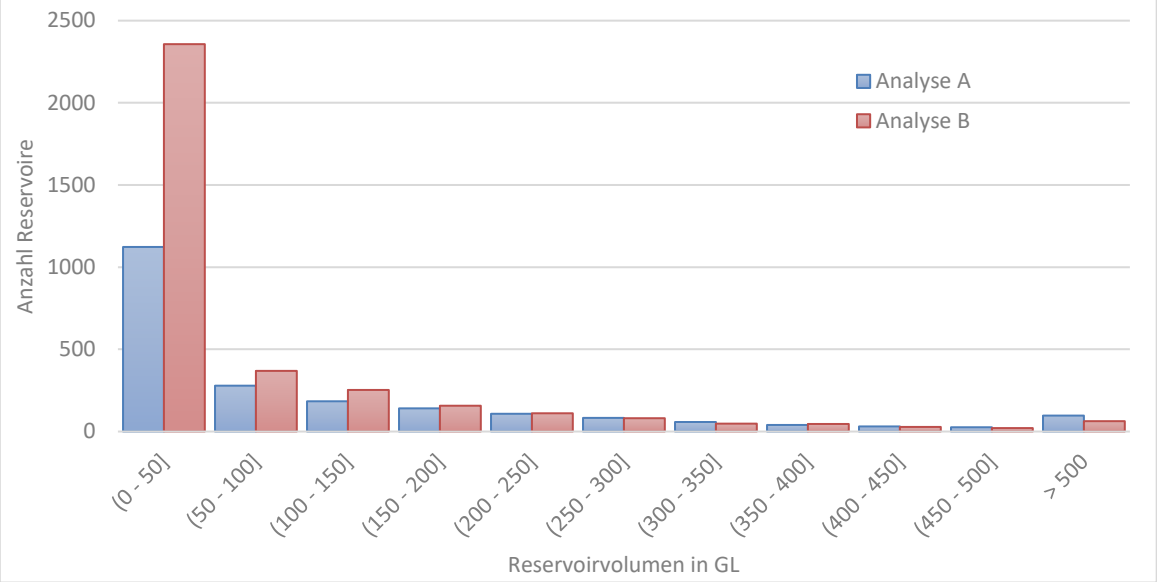


Abbildung 46: Anzahl Reservoire nach Reservoirvolumen

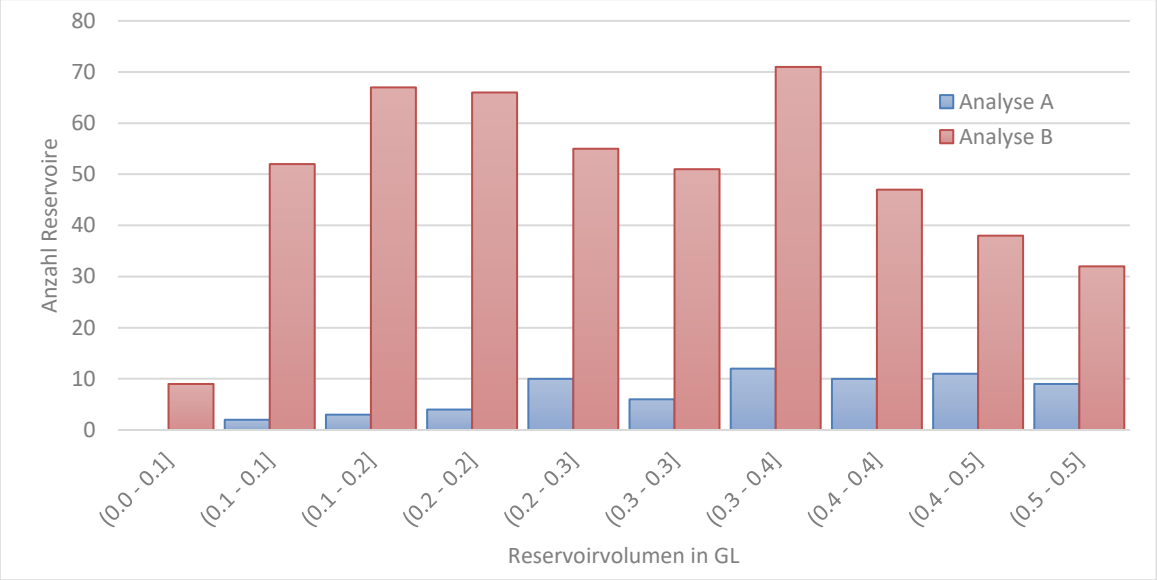


Abbildung 47: Anzahl Reservoire nach Reservoirvolumen < 0,5 GL

In Abbildung 48 und Abbildung 49 ist das Reservoirvolumen jedes einzelnen gefundenen Kandidaten im Verhältnis zu Dammvolumen bzw. Konturhöhe dargestellt.

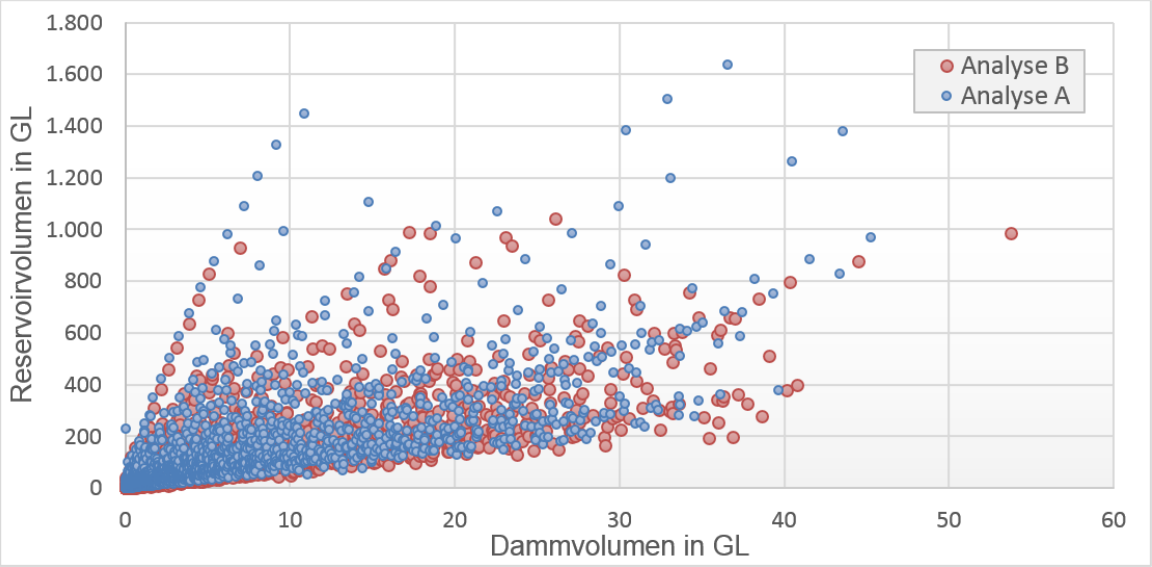


Abbildung 48: Reservoirvolumen zu Dammvolumen

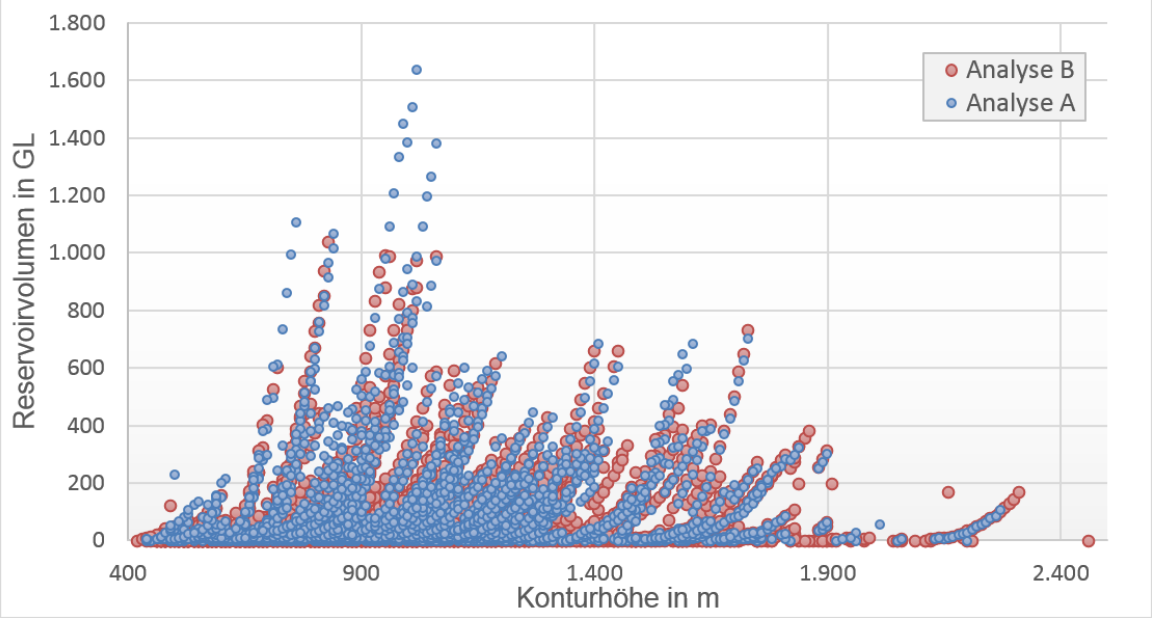


Abbildung 49: Reservoirvolumen nach Konturhöhe

Ein Diagramm zur Visualisierung des nichtlinearen Verhältnisses zwischen Dammvolumen und Dammhöhe (siehe auch 2.3) ist in Abbildung 50 ersichtlich.

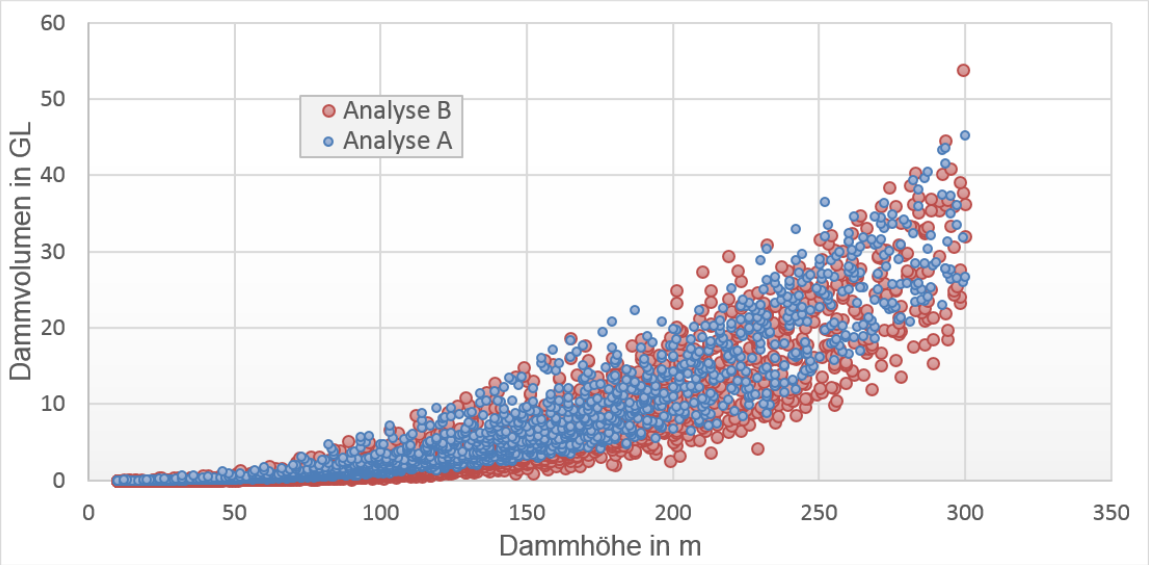


Abbildung 50: Dammvolumen nach Dammhöhe

Abbildung 51 zeigt die Verteilung der gefundenen Energiespeicherkapazitäten – aufgrund des großen Wertebereiches mit logarithmischer Y-Achse.

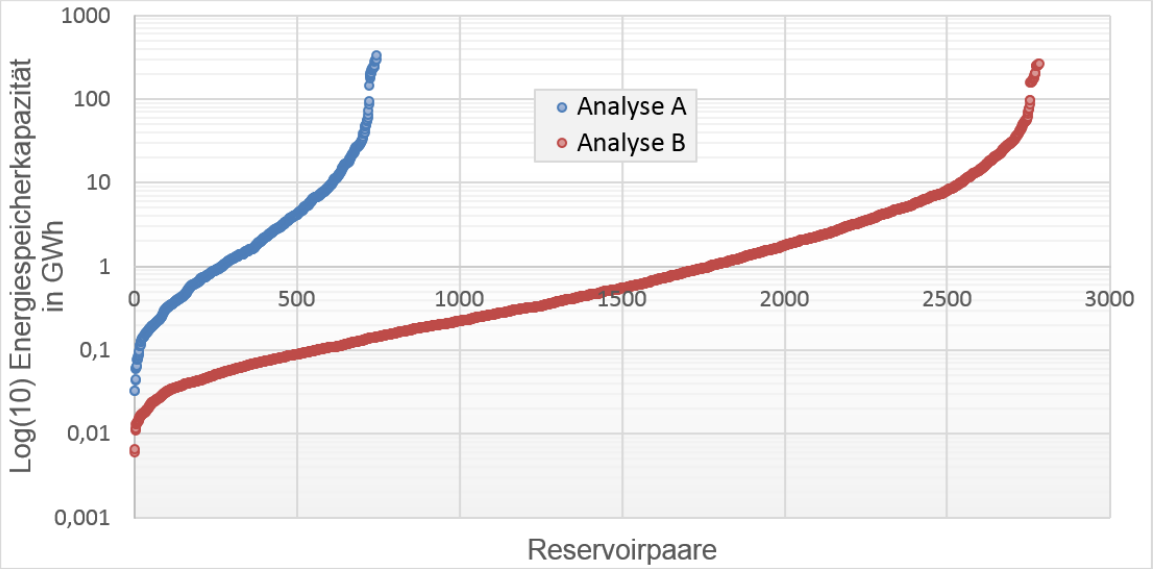


Abbildung 51: Verteilung der Energiespeicherkapazitäten Log(10)

Abbildung 52 stellt die gefundenen Energiespeicherkapazitäten in Abhängigkeit der Reservoirhöhenunterschiede dar. Histogramme der Reservoirpaare nach Distanz zwischen Dammmittelpunkten sowie Höhenunterschied der Reservoire sind in Abbildung 53 und Abbildung 54 abgebildet.

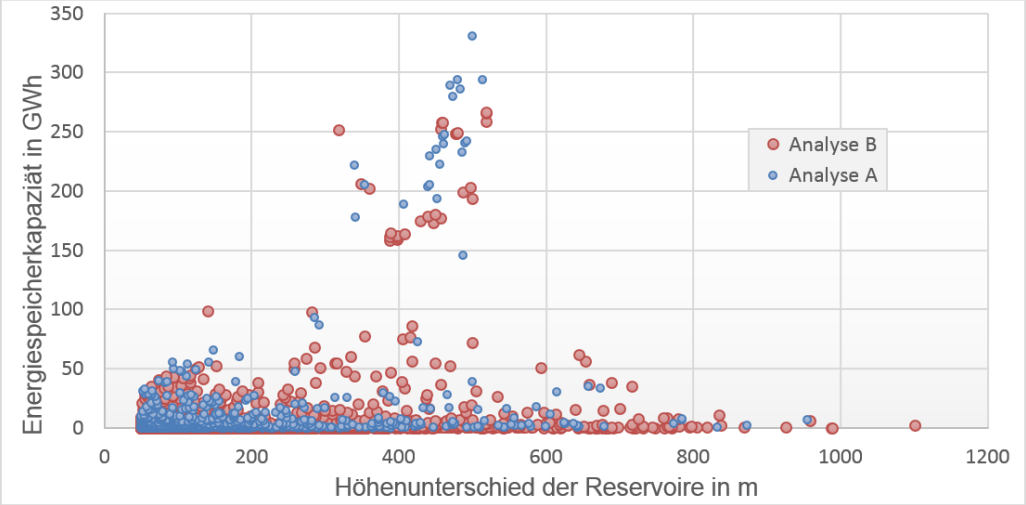


Abbildung 52: Energiespeicherkapazität nach Höhenunterschied

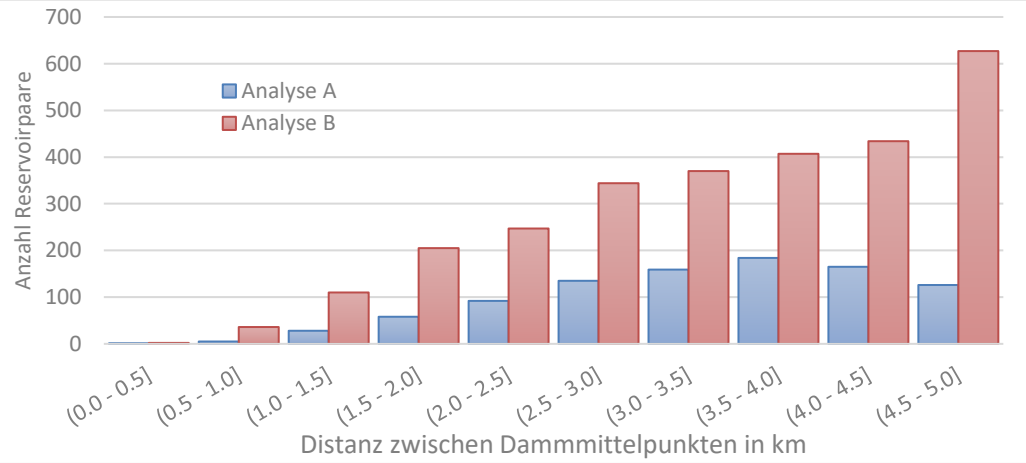


Abbildung 53: Anzahl Reservoirpaare nach Distanz zwischen Dammmittelpunkten

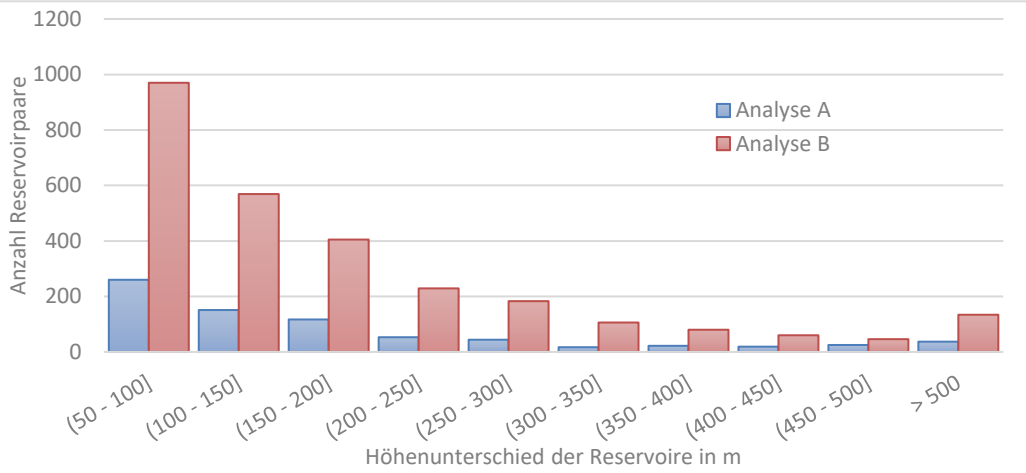


Abbildung 54: Anzahl Reservoirpaare nach Höhenunterschied der Reservoire

4.3.1 Berechnungszeiten

Die Berechnungszeiten von Analyse A und B sowie eines zusätzlichen Referenzlaufes mit dem DHM aus Analyse A, jedoch mit 100 m Punktintervall sind in Gesamtdauer (Abbildung 55) und Berechnungsanteile je Teilschritt (Abbildung 56) dargestellt.

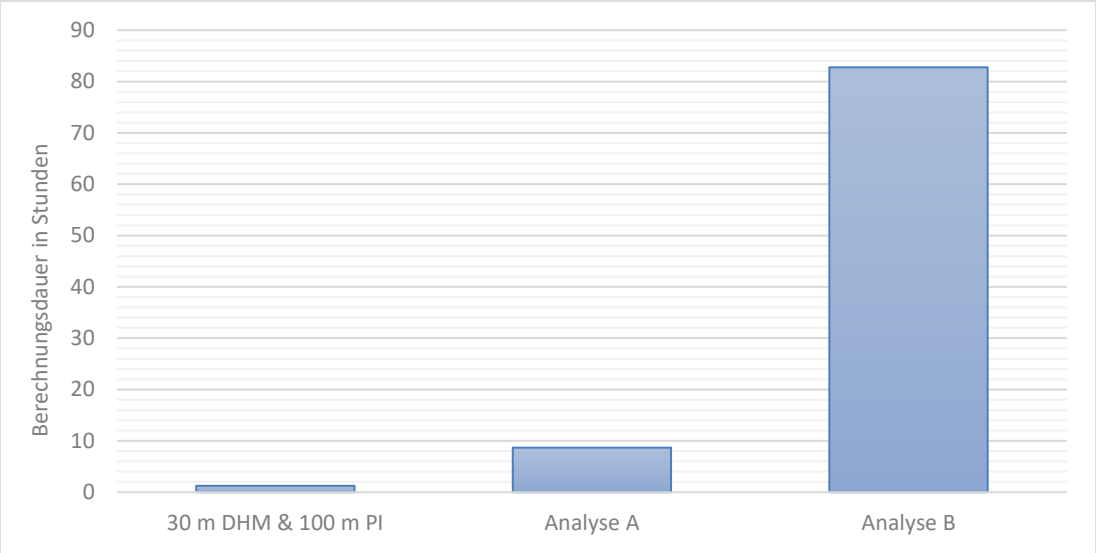


Abbildung 55: Vergleich der Berechnungsdauern

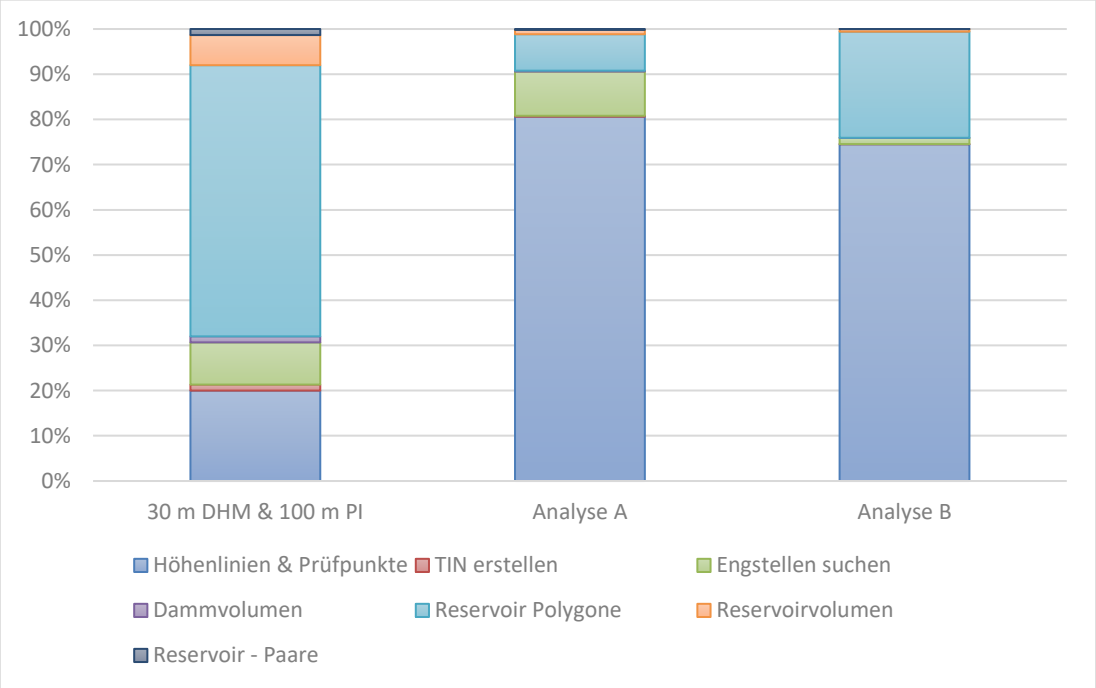


Abbildung 56: Vergleich Berechnungsanteile je Teilschritt

4.4 Analyse C - E (Einzelkonturen und verringertes Punktintervall)

Da sich das Intervall der Prüfpunkte entlang der Konturlinien maßgeblich auf die Berechnungszeit auswirkt, wurde anhand einer einzelnen Konturlinie die Veränderung der Ergebnisse für ein auf 5 m (entsprechend der Auflösung des VoGIS DHM) reduziertes Punktintervall geprüft. Hier handelte es sich um die 1.030 m Linie der in 4.2 erstellten Konturlinien mit 542 km Länge, für welche unter 4.2 mit einem 30 m Punktintervall 45 Dammkandidaten eruiert wurden. Für einen kompletten Kontext wurde zusätzlich auch die 1.040 m Konturlinie aus dem SRTM Datensatz einzeln analysiert – welche bei Überlagerung der Linien die beste Übereinstimmung zur 1.030 m Konturlinie aus Analyse B besaß.

Die Ergebnisse sind in Tabelle 10 sowie Abbildung 57 dargestellt sowie für eine bessere Erkennbarkeit in drei Ausschnitten exemplarisch vergrößert abgebildet (Abbildung 58, Abbildung 59 und Abbildung 60).

Tabelle 10: Parameter und Ergebnisvergleich Analyse C - E

	Analyse C		**Analyse D**		**Analyse E**	
Konturlinie	1.040 m aus 4.1		1.030 m aus 4.2		1.030 m aus 4.2	
DHM Aufl.	30 m		5 m		5 m	
PI	30 m		30 m		5 m	
Schritt	Dauer	Anzahlen	Dauer	Anzahlen	Dauer	Anzahlen
Konturlinien & Prüfpunkte	3 Min.	1 KL 15.788 PP	27 Min.	1 KL 18.071 PP	161 Min.	1 KL 108.418 PP
Damm–Kandidaten	< 1 Min.	30 DK 124.022.741 PK	< 1 Min.	50 DK 161.873.786 PK	36 Min.	49 DK 5.826.948.341 PK
Dammvolumen	< 1 Min.	**27 DK** verbleibend	< 1 Min.	**45 DK** verbleibend	< 1 Min.	**44 DK** verbleibend
Reservoir Polygone	< 1 Min.	27 P	5 Min.	45 P	5 Min.	44 P
Reservoir-Volumen	< 1 Min.		< 1 Min.		< 1 Min.	
Summe	**≈ 7 Min. (≈ 0,1 Std.)**		**≈ 35 Min. (≈ 0,6 Std.)**		**≈ 204 Min. (≈ 3,4 Std.)**	

Abkürzungen: Punktintervall (PI), Konturlinie (KL), Prüfpunkt (PP), Stützpunkt (SP), Dammkandidat (DK), Punktkombination (PK), Polygon (P)

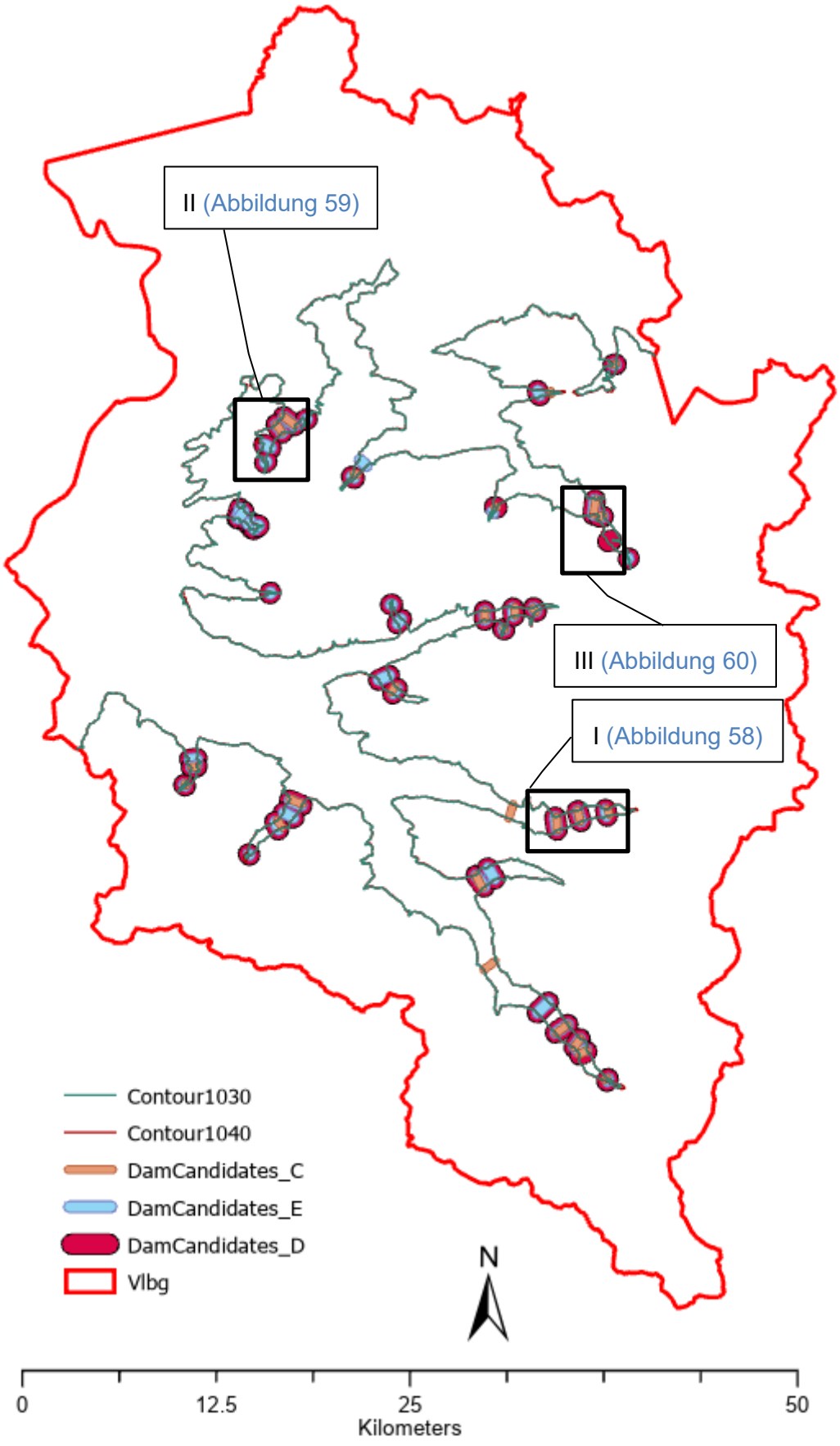


Abbildung 57: Vergleich Dammkandidaten aus Analysen C - E

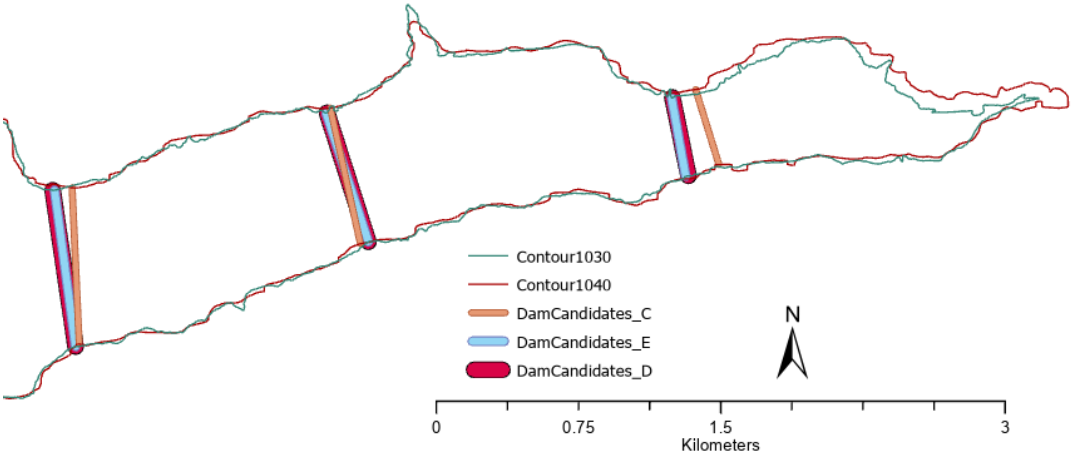


Abbildung 58: Dammkandidaten Ausschnitt I

Wie in Abbildung 58 ersichtlich, sind die Dammkandidaten aus Analyse D (30 m PI) und E (5 m PI) nahezu ident und nicht weit entfernt der auf Basis der SRTM-Daten ermittelten Kandidaten.

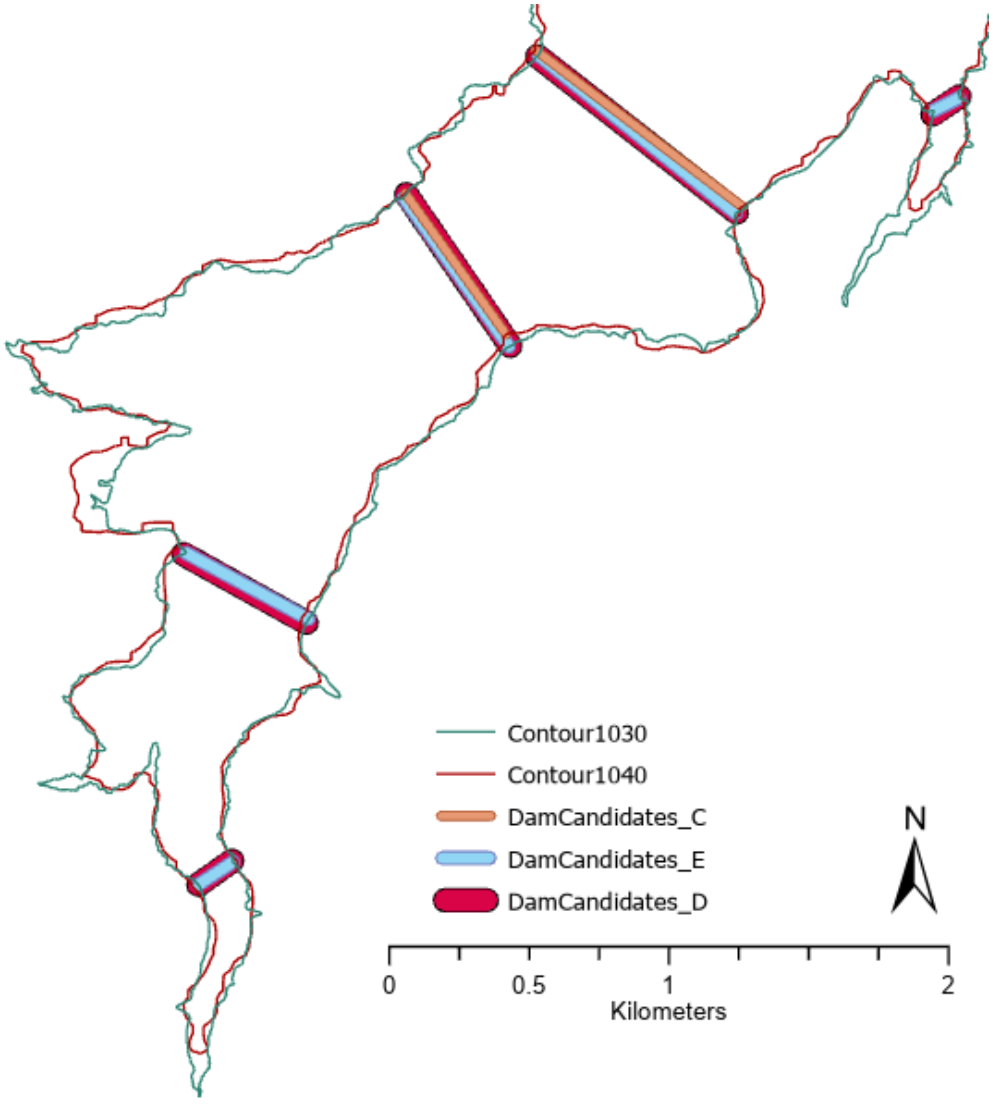


Abbildung 59: Dammkandidaten Ausschnitt II

Auch die in Abbildung 59 und Abbildung 60 abgebildeten Kandidaten aus Analyse D und E liegen direkt übereinander, mit Ausnahme eines einzelnen Kandidaten aus Analyse D (an der Engstelle in der Mitte von Abbildung 60 in Rot). Dieser klassische Grenzfall kommt zustande, da aufgrund von wenigen Metern Längendifferenz ein Ausschlusskriterium aus 3.1.4 erfüllt wurde: Existenz eines weiter talabwärts gelegenen Kandidaten derselben Konturlinie (größerer Reservoirumfang) welcher eine kürzere Dammlänge aufweist.

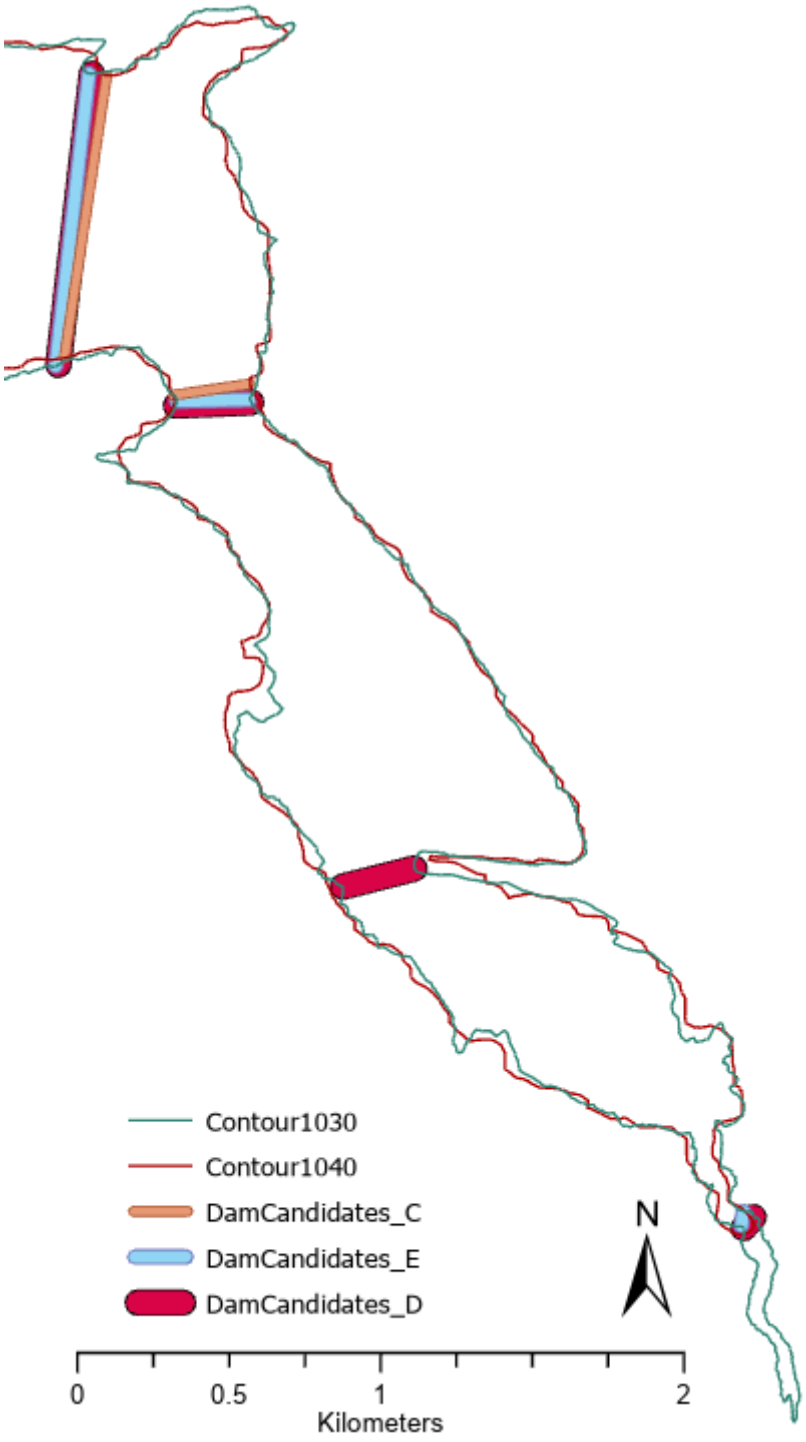


Abbildung 60: Dammkandidaten Ausschnitt III

Im Punktdiagramm in Abbildung 61 wurde die räumliche Verteilung der ermittelten Dammkandidaten anhand der Position des Dammstartpunktes in % der Gesamtkonturlänge (X – Achse) und der jeweiligen Dammlänge (Y – Achse) überlagert visualisiert. Hier sind nur wenige Unterschiede zwischen der Analyse D und E erkennbar – Ergebnisse aus Analyse C sind in der Zahl geringer, aber wo vorhanden sowohl in Position als auch in Dammlänge sehr nah an den Ergebnissen der beiden anderen Analysen.

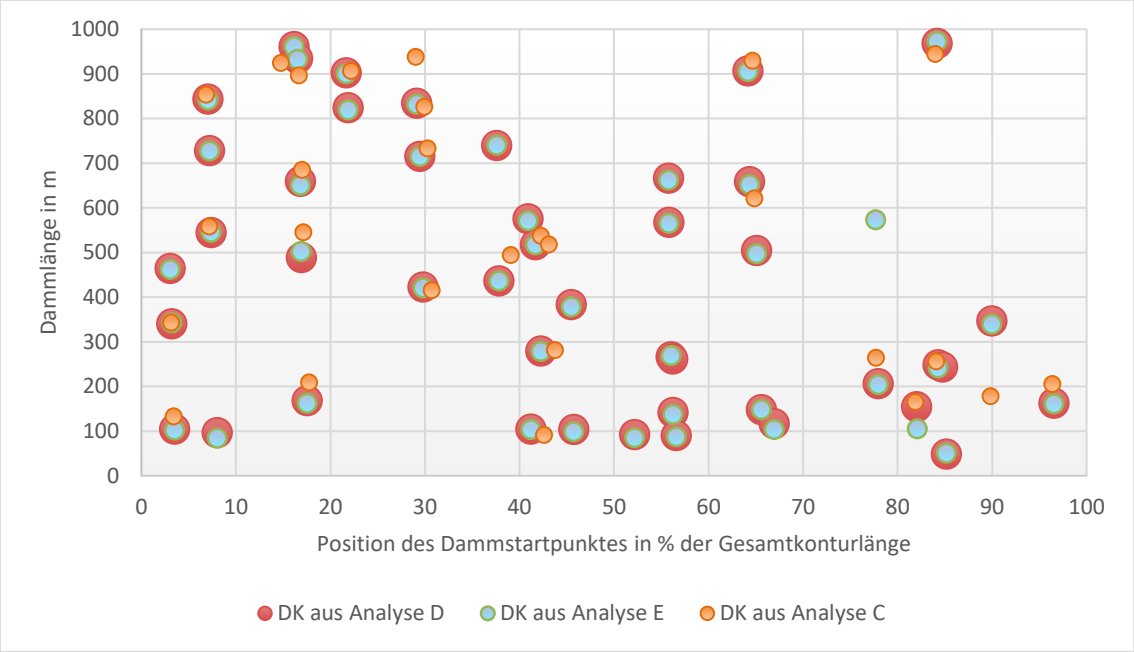


Abbildung 61: Dammpositionen und -längen der Analysen C - E im Vergleich

5 Diskussion

Anhand von Abbildung 40 und Abbildung 44 ist zu erkennen, dass der Algorithmus eine Vielzahl an Dammkandidaten und potentiellen Reservoirpaaren unterschiedlichster Kapazitäten und Lagen anhand der Höhenmodelle identifizieren konnte. Ungeeignete bzw. durch das Gelände verlaufende Dammkandidaten wurden automatisiert entfernt, sodass im Ergebnis kein einzelner betroffener Kandidat mehr gefunden werden konnte. In Abbildung 41 und Abbildung 42 ist ein Vergleich zwischen dem durch den Algorithmus gefundenen Dammkandidaten bzw. dem daraus resultierenden Reservoir an Position des real existierenden Lünersee Staudammes und einer orthofotografischen Darstellung aus Google Earth gegenübergestellt. Das eruierte Volumen an diesem Standort entspricht jedoch nicht dem tatsächlichen Volumen, da zum Zeitpunkt der Shuttle-Mission im Jahr 2000 der Lünersee bereits existierte und somit der damalige Wasserstand als Geländehöhe im Algorithmus berücksichtigt wurde. In Analyse B wurde der Dammkandidat an dieser Position sogar gänzlich aussortiert, da der Wasserstand am Tag der VoGIS Datenaufnahme wohl nahe am Maximalstand war und dadurch die Mindesthöhe des Dammes von 10 m (3.1.5) nicht erreicht wurde.

Der Vergleich zwischen Analyse A (SRTM Daten) und Analyse B (VoGIS Daten) lieferte einerseits eine große Anzahl an übereinstimmenden Kandidaten, andererseits jedoch auch eine deutliche Abweichung von 1.364 Kandidaten, welche in Analyse B mehr gefunden wurden. Aufgrund der größeren Anzahl an Dammkandidaten wurde auch in 3.1.8 eine deutlich höhere Anzahl (Faktor 4) an Reservoirpaaren ermittelt. Anhand der Abbildung 46 und Abbildung 47 ist schnell erkennbar, dass mit dem VoGIS Datensatz sehr viel mehr Reservoire mit geringem Volumen identifiziert wurden, als mit dem SRTM Datensatz. Nach einer Analyse der Ursache dieses erklärungsbedürftigen Unterschiedes wurde klar, dass es sich hier um eine Auswirkung des „Coastline Paradox“ (Mandelbrot 1967) handelt, da sich die Konturlinien analog zu den von Mandelbrot untersuchten Küstenlinien verhalten – die Küstenlinie entspricht konkret der Kontur der Höhe 0 m ü. NN. Anhand der Abbildung 62 ist schnell erkennbar, dass sich die „gemessene“ Küstenlänge (Konturlänge) erhöht, je kleiner die Messintervalle sind. Die Messintervalle bei der Erstellung von Konturlinien sind durch die Auflösung der DHM festgesetzt.

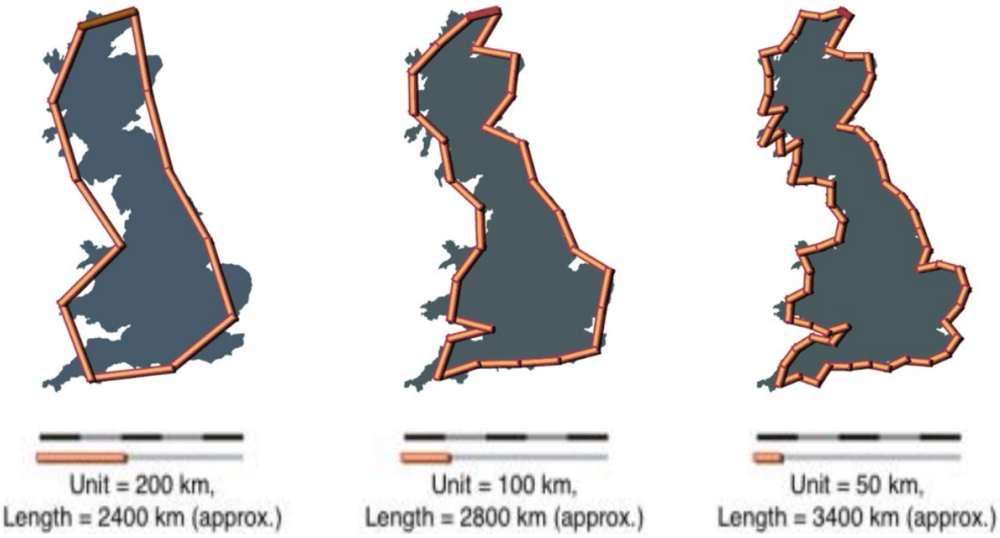


Abbildung 62: Coastline Paradox (Fatima et al. 2015)

Dieser Umstand ist in dieser Arbeit beispielsweise in der Tabelle 10 erkennbar, da für die Höhenlinie aus den SRTM-Daten 15.788 Prüfpunkte erstellt wurden, für dieselbe überlagerte aus den VoGIS Daten ermittelte Höhenlinie jedoch zusätzliche 2.283 Prüfpunkte notwendig waren um die gesamte Linie abzudecken. Dies entspricht bei dem verwendeten 30 m Punktintervall einer zusätzlichen Länge der Linie von ca. 68 km bzw. 14 %. Die zuvor erwähnte deutliche Abweichung der Ergebnisanzahlen ist nun konkret darin begründet, dass im Algorithmus in Schritt 3.1.2 und 3.1.4 einige längenabhängige Parameter zur Einschränkung der Anzahl der Konturlinien und Dammkandidaten verwendet werden und die Konturlänge sich abhängig von der Auflösung des für die Erstellung der Konturlinien herangezogenen Höhenmodells verändert.

Hiervon betroffene Parameter sind:

- In Schritt 3.1.2 „Höhenlinien und Prüfpunkte erstellen“
    - **Minimale Konturlänge**

      Durch die erhöhten Konturlängen in Analyse B erfüllen 246 zusätzliche Konturlinien das Kriterium der minimalen Konturlänge von 10.000 m und werden in den Folgeschritten weiter analysiert.
- Schritt 3.1.4 „Engstellen (Dammkandidaten) suchen“
    - **Maximale Distanz$_{Isohypse}$ (≈ Reservoirumfang)**

      Die Abbruchbedingung, welche die Suche ausgehend von einem Startpunkt entlang einer Konturlinie limitiert, sobald die Länge entlang der Linie 30.000 m übersteigt, wird somit in Analyse B schneller erfüllt. Hierdurch wurden in Analyse A großvolumigere Reservoire eruiert als in Analyse B (Abbildung 49, Tabelle 8) und auch die größten Energiespeicherkapazitäten aus Analyse A konnten in Analyse B dadurch nicht eruiert werden (Abbildung 52).

- **Minimale Distanz$_{Isohypse}$**
  Dieser Parameter verursacht den großen quantitativen Unterschied der ermittelten Dammkandidaten, da die minimale Distanz von 1.000 m entlang einer Höhenlinie in Analyse B zwischen zwei Punkten bereits erreicht sein kann, obwohl dies in Analyse A noch nicht der Fall ist. Dies erhöht somit die Anzahl an gefundenen kleinvolumigen Reservoiren.
- **Minimales Verhältnis Distanz$_{Isohypse}$ / Distanz$_{Euklid}$**
  Das Mindestverhältnis von 10 zwischen Reservoirumfang und Dammlänge wird ebenfalls in Analyse B eher erreicht.

In Abbildung 49 sind deutliche Muster erkennbar. Diese entstehen, da an einer Engstelle oft mehrere Dammkandidaten unterschiedlicher Höhe eruiert wurden und sich die Punkte im Diagramm durch die 10 m Konturhöhenintervalle und das jeweils durch die vergrößerte Dammhöhe erhöhte Reservoirvolumen erkennbar aneinanderreihen. Die einzelnen Punktreihen repräsentieren also eigentlich meist dieselbe Dammposition, aber mit unterschiedlichen Dammhöhen.

Für die manuelle Prüfung ausgewählter Kandidaten ist eine 3D Darstellung eine einfache Unterstützung – eine entsprechende Visualisierung eines potentiellen PSKW ist in Abbildung 43 ersichtlich.

Die Darstellung der möglichen Energiespeicherkapazitäten nach Höhenunterschied zwischen den Reservoiren (Abbildung 52) scheint auf den ersten Blick einen Sweetspot für sehr hohe Energiespeicherkapazitäten vermuten zu lassen. Ein genauer Blick in die Daten zeigte jedoch, dass alle Kapazitäten > 100 GWh aus der Kombination einer sehr geringen Anzahl an Reservoiren miteinander entstehen, welche allesamt in Abbildung 40 im Süden des Untersuchungsgebietes positioniert sind.

Im Vergleich der Berechnungszeiten (4.3.1) ist ersichtlich das beinahe 10-mal so viel Berechnungsaufwand durch den Einsatz des hochauflösenderen Höhenmodelles für dasselbe Gebiet notwendig war und der Flaschenhals in der Performance durch die Erstellung der Prüfpunkte sowie der Reservoir-Polygone gegeben ist. Dies ist der Fall, da die Höhenlinien und somit die Reservoir-Polygone in Analyse B aufgrund der detaillierteren Auflösung aus deutlich mehr Vertices bestehen, welche die Berechnung verlangsamen. Für diese beiden Schritte war es leider nicht mehr möglich bis zur Finalisierung der Arbeit eine Anwendung der notwendigen ArcGIS Werkzeuge unter Ausnützung aller verfügbaren Prozessorkerne (MultiThreading) umzusetzen. Da die eruierten Standorte jedoch grundsätzlich (abgesehen von den durch die zuvor beschriebene „Coastline Paradox“ – Problematik zusätzlich in Analyse B eruierten Dammkandidaten) sehr gut übereinstimmten, ist die Wahl der SRTM Daten (Analyse A) nicht nur in Hinblick auf die Datenverfügbarkeit sondern auch auf die hierdurch

deutlich geringere Berechnungszeit mit grundsätzlich vergleichbaren Ergebnissen zu bevorzugen. Eine Reduzierung des Punktintervalls auf 5 m, wie in Analyse E für eine einzelne Konturlinie durchgeführt, erreichte keine nennenswert verbesserten Ergebnisse, weshalb die um den Faktor 6 erhöhte Berechnungsdauer keineswegs gerechtfertigt werden kann.

Der Vergleich aus Analyse C – E (4.4) auf jeweils eine einzelne Konturlinie zeigt, dass der Algorithmus grundsätzlich aus unterschiedlichen Datengrundlagen dieselben geeigneten potentiellen Dammstandorte eruieren konnte und die SRTM Daten qualitativ vergleichbare Ergebnisse liefern. Außer den durch die gewählten Parameter in den SRTM Daten nicht berücksichtigten sehr kleinen Reservoiren (Abbildung 59) und einigen wenigen Grenzfällen (Abbildung 60) stimmen die gefundenen Standorte sehr genau mit den Standorten überein, welche auf Basis der deutlich höher auflösenden VoGIS Datensätze ermittelt werden konnten (Abbildung 61).

Der entwickelte Ablauf ist grundsätzlich für verschiedene Dammarchitekturen (Schüttdämme, Gewichts- oder Bogenstaumauern) geeignet, da diese alle idealerweise an einer Engstelle platziert werden, wodurch sich das benötigte Materialvolumen und die Kosten reduzieren. Es müsste lediglich die Dammvolumensberechnung (3.1.5) entsprechend angepasst werden, um genauere Ergebnisse für eine spezifische Variante zu erhalten.

Vergleichbare Studien (Tabelle 2) wählten sehr grobe Prüfintervalle (450 m durch Larentis et al. (2010) bzw. 500 m durch Lu und Wang (2017)) oder fixierte Dammhöhen (40 m durch Lu et al. (2018)) entlang Drainagelinien oder selektierten gar nur bestehende Reservoire (Rogeau et al. 2017). Lediglich Petheram et al. (2017) erzielte nach Ansicht des Autors ein ähnliches Ergebnis mit der selbst entwickelten Software „DamSite“. Diese stand jedoch leider nicht für direkte Vergleiche zur Verfügung und auch weitere Details zur konkreten Umsetzung des dort entwickelten Algorithmus zur Suche nach geeigneten Dammstandorten war trotz Nachfrage beim Studienautor aufgrund von bestehenden Urheberrechten nicht zu erhalten. Diese Arbeit unterscheidet sich jedoch von allen anderen gefundenen Studien in dem Punkt, dass diese in ihrer Methodik jeweils auf zuvor erstellten Drainage- bzw. Flusslinien basierten und die Dammkonstruktion von dort beginnend im rechten Winkel oder Schrittweise nach außen durchgeführt wurde. Diese Arbeit verzichtet hingegen komplett auf hydrologische Analysen wie „Abflussakkumulation“, „Fließrichtung“ oder „Wassereinzugsgebiet“ und stützt sich ausschließlich auf die über die Höhenlinien ermittelten Engstellen in einem Tal.

Aus Sicht des Autors konnte das Forschungsziel dahingehend erreicht werden, dass ein performanter Geländeanalysealgorithmus erstellt werden konnte, welcher in der Lage ist selbst mit einem einzelnen herkömmlichen Desktop Computer in vertretbarer Zeit auf Basis von frei verfügbaren Höhendaten automatisiert eine Vielzahl an potentiell geeigneten

Dammstandorten zu eruieren. Die SRTM-1 Datensätze sind nach den Vergleichen von Analysen A – E von Qualität und Auflösung für diese Anwendung als geeignet zu betrachten. Wie sich aus den Ergebnissen zeigte, sind einige im Algorithmus verwendeten Parameter (wie oben beschrieben) jedoch spezifisch auf die 30 m Auflösung der SRTM – Datenbasis ausgerichtet und erwirken bei Verwendung einer Datenquelle mit abweichender Auflösung Unterschiede in den Ergebnissen – speziell in den Randbereichen des Reservoirvolumensspektrums.

6 Zusammenfassung und Ausblick

Der erarbeitete Algorithmus konnte automatisiert und performant eine Vielzahl an potentiellen Dammkandidaten bzw. Reservoirpaaren für das Untersuchungsgebiet identifizieren. Der Vergleich der auf Basis der verschiedenen Höhenmodellen entstandenen Ergebnisse zeigte einerseits eine grundsätzliche Eignung der SRTM Daten, andererseits jedoch auch, dass der Algorithmus in der aktuellen Form eine unerwünschte Abhängigkeit von der Auflösung des verwendeten Höhenmodelles aufweist. Für eine Anwendung mit Höhenmodellen einer horizontalen Auflösung, welche deutlich von den SRTM-1 Datensätzen abweicht, müssten die Längenparameter im Algorithmus neu kalibriert werden.

Sofern eine allgemeine mit verschiedensten Höhenmodellen kompatible Lösung angestrebt wird, sollten die Längenparameter im Algorithmus von den Konturlängen entkoppelt oder diese automatisiert an die Auflösung der verwendeten Höhendaten anpasst werden, um für ein identisches Untersuchungsgebiet auch ein im Ausmaß identisches Ergebnis zu erhalten.

In einer zusätzlichen Weiterentwicklung des Werkzeuges wäre die automatisierte Ausführung auf einzelne SRTM - Kacheln mit Berücksichtigung einer Überlappung der Randbereiche sinnvoll. Die Ergebnisse sollten hierbei in einer gemeinsamen Datenbank zusammengeführt werden. Hierdurch wäre auch eine Aufteilung auf mehrere Rechner und somit eine einfache Skalierbarkeit der Analyse sogar auf alle weltweit verfügbaren SRTM Kacheln denkbar.

7 Literatur- und Quellenverzeichnis

ArcGIS Pro SDK Community Samples. Online verfügbar unter https://github.com/Esri/arcgis-pro-sdk-community-samples, zuletzt geprüft am 28.11.2019.

ArcGIS Pro Tool Reference. Online verfügbar unter https://pro.arcgis.com/en/pro-app/tool-reference/main/arcgis-pro-tool-reference.htm, zuletzt geprüft am 28.11.2019.

Ardizzon, G.; Cavazzini, G.; Pavesi, G. (2014): A new generation of small hydro and pumped-hydro power plants. Advances and future challenges. In: *Renewable and Sustainable Energy Reviews* 31, S. 746–761. DOI: 10.1016/j.rser.2013.12.043.

Blarke, M. B.; Lund, H. (2008): The effectiveness of storage and relocation options in renewable energy systems. In: *Renewable Energy* 33 (7), S. 1499–1507. DOI: 10.1016/j.renene.2007.09.001.

Bueno, C.; Carta, J. A. (2006): Wind powered pumped hydro storage systems, a means of increasing the penetration of renewable energy in the Canary Islands. In: *Renewable and Sustainable Energy Reviews* 10, S. 312–340. DOI: 10.1016/j.rser.2004.09.005.

Chen, H.; Cong, T. N.; Yang, W.; Tan, C.; Li, Y.; Ding, Y. (2009): Progress in electrical energy storage system. A critical review. In: *Progress in Natural Science* 19 (3), S. 291–312. DOI: 10.1016/j.pnsc.2008.07.014.

Connolly, D.; MacLaughlin, S.; Leahy, M. (2010): Development of a computer program to locate potential sites for pumped hydroelectric energy storage. In: *Energy* 35 (1), S. 375–381. DOI: 10.1016/j.energy.2009.10.004.

Fatima, H.; Arsalan, M.; Khalid, A.; Marjan, K.; Kumar, M. (2015): Spatio -Temporal Analysis of Shoreline Changes along Makran Coast Using Remote Sensing and Geographical Information System.

Fitzgerald, N.; Lacal Arántegui, R.; McKeogh, E.; Leahy, P. (2012): A GIS-based model to calculate the potential for transforming conventional hydropower schemes and non-hydro reservoirs to pumped hydropower schemes. In: *Energy* 41 (1), S. 483–490. DOI: 10.1016/j.energy.2012.02.044.

Gimeno-Gutiérrez, M.; Lacal-Arántegui, R. (2013): Assessment of the European potential for pumped hydropower energy storage. A GIS-based assessment of pumped hydropower storage potential. Luxembourg: Publications Office of the European Union (JRC scientific and policy reports, 25940).

Google Earth. Online verfügbar unter https://earth.google.com/web/, zuletzt geprüft am 29.11.2019.

Hollister, J. W.; Milstead, W. B.; Urrutia, M. A. (2011): Predicting maximum lake depth from surrounding topography. In: *PloS one* 6 (9), e25764. DOI: 10.1371/journal.pone.0025764.

IEA (2018): Renewables 2018. Analysis and forecasts to 2023. Paris: OECD Publishing (Market report series).

Jiménez Capilla, J. A.; Carrión, J. A.; Alameda-Hernandez, E. (2016): Optimal site selection for upper reservoirs in pump-back systems, using geographical information systems and multicriteria analysis. In: *Renewable Energy* 86, S. 429–440. DOI: 10.1016/j.renene.2015.08.035.

Lacal-Arántegui, R.; Tzimas, E. (2012): SETIS expert workshop on the assessment of the potential of pumped hydropower storage. Online verfügbar unter https://setis.ec.europa.eu/sites/default/files/reports/SETIS-expert-workshop-assessment-potential-pumped-hydropower-storage.pdf, zuletzt geprüft am 16.08.2019.

Land Vorarlberg (2019): VoGIS. Online verfügbar unter https://vogis.cnv.at/geodaten/, zuletzt geprüft am 16.08.2019.

Larentis, D. G.; Collischonn, W.; Olivera, F.; Tucci, C. E. M. (2010): Gis-based procedures for hydropower potential spotting. In: *Energy* 35 (10), S. 4237–4243. DOI: 10.1016/j.energy.2010.07.014.

Lu, B.; Stocks, M.; Blakers, A.; Anderson, K. (2018): Geographic information system algorithms to locate prospective sites for pumped hydro energy storage. In: *Applied Energy* 222, S. 300–312. DOI: 10.1016/j.apenergy.2018.03.177.

Lu, X.; Wang, S. (2017): A GIS-based assessment of Tibet's potential for pumped hydropower energy storage. In: *Renewable and Sustainable Energy Reviews* 69, S. 1045–1054. DOI: 10.1016/j.rser.2016.09.089.

Luo, X.; Wang, J.; Dooner, M.; Clarke, J. (2015): Overview of current development in electrical energy storage technologies and the application potential in power system operation. In: *Applied Energy* 137, S. 511–536. DOI: 10.1016/j.apenergy.2014.09.081.

Mandelbrot, B. (1967): How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension. In: *Science* 156 (3775), S. 636–638. Online verfügbar unter http://www.jstor.org/stable/1721427.

Petheram, C.; Gallant, J.; Read, A. (2017): An automated and rapid method for identifying dam wall locations and estimating reservoir yield over large areas. In: *Environmental Modelling & Software* 92, S. 189–201. DOI: 10.1016/j.envsoft.2017.02.016.

PGSP GmbH: Talsperre Leibis. Online verfügbar unter http://www.pgsp.de/architektur_p1.html#1, zuletzt geprüft am 06.08.2019.

Rogeau, A.; Girard, R.; Kariniotakis, G. (2017): A generic GIS-based method for small Pumped Hydro Energy Storage (PHES) potential evaluation at large scale. In: *Applied Energy* 197, S. 241–253. DOI: 10.1016/j.apenergy.2017.03.103.

Soha, T.; Munkácsy, B.; Harmat, Á.; Csontos, C.; Horváth, G.; Tamás, L. et al. (2017): GIS-based assessment of the opportunities for small-scale pumped hydro energy storage in middle-mountain areas focusing on artificial landscape features. In: *Energy* 141, S. 1363–1373. DOI: 10.1016/j.energy.2017.11.051.

Steffen, B. (2012): Prospects for pumped-hydro storage in Germany. In: *Energy Policy, June 2012, Vol.45, pp.263-267* 45, S. 420–429. DOI: 10.1016/j.enpol.2012.02.052.

Strebl, M. (2012): Smart Grids Modellregion Salzburg. Online verfügbar unter https://www.stadt-salzburg.at/pdf/smart_grids_modellregion_salzburg.pdf, zuletzt geprüft am 20.10.2019.

USGS - U.S. Geological Survey: EarthExplorer - Home. USGS - U.S. Geological Survey. Online verfügbar unter https://earthexplorer.usgs.gov/, zuletzt geprüft am 18.10.2019.

Vattenfall GmbH: Goldisthal Oberbecken. Online verfügbar unter https://group.vattenfall.com/de/newsroom/blog-news-presse/blog/2018/september/wildbienen-genießen-artenschutz-in-goldisthal, zuletzt geprüft am 06.08.2019.

Yi, C. S.; Lee, J. H.; Shim, M. P. (2010): Site location analysis for small hydropower using geo-spatial information system. In: *Renewable Energy* 35 (4), S. 852–861. DOI: 10.1016/j.renene.2009.08.003.

Zakeri, B.; Syri, S. (2015): Electrical energy storage systems. A comparative life cycle cost analysis. In: *Renewable and Sustainable Energy Reviews* 42, S. 569–596. DOI: 10.1016/j.rser.2014.10.011.

Anhang A: Quellcode

Zur Erstellung des Add-In wurde die ArcGIS Pro Tool Reference sowie die umfangreichen ArcGIS Pro SDK Community Samples verwendet.

Der komplette Quellcode in Form eines ArcGIS Pro Module Add-In sowie die erstellte ArcGIS Toolbox ist verfügbar unter dem folgenden GIT-Repository:
https://github.com/Codewolke/Reservoir

Das bereits kompilierte und verwendbare Add-In ist verfügbar unter:
https://github.com/Codewolke/Reservoir/blob/master/AddIn/Reservoir.esriAddinX

In diesem Anhang ist der Quellcode der 8 im ArcGIS Pro Module Add-In vorhandenen Buttons aufgeführt, welcher den genauen umgesetzten Ablauf und die Details der Umsetzung zeigt. Ein komplettes ArcGIS Pro Module Add-In benötigt jedoch noch weitere Ressourcen (Bilder, Oberflächen- und Modulkonfigurationen, …), welche nicht spezifisch für das hier umgesetzte Add-In sind und deshalb lediglich im kompletten Repository zur Verfügung gestellt werden.

A.1 PrepareContoursButton

```
internal class PrepareContoursButton : Button
{
    protected override async void OnClick()
    {
        string inputDEM = Parameter.DEMCombo.SelectedItem.ToString();
        string contourInterval = Parameter.ContourIntervalBox.Text;
        string PointInterval = Parameter.PointIntervalBox.Text + " meters";
        string Workspace = Project.Current.DefaultGeodatabasePath;
        var args = Geoprocessing.MakeValueArray(contourInterval, PointInterval, Work-
space, inputDEM);
        await SharedFunctions.RunModel(args, "Prepare Contours");
    }
}
```

A.2 CreateTINButton

```
internal class CreateTINButton : Button
{
    protected override async void OnClick()
    {
        string inputDEM = Parameter.DEMCombo.SelectedItem.ToString();
        string tinLayer = "TIN";
        var args = Geoprocessing.MakeValueArray(inputDEM, tinLayer);
        await SharedFunctions.RunModel(args, "CreateTIN");
    }
}
```

A.3 DetectCandidateDamsButton

```
internal class DetectCandiateDamsButton : Button
{
     private static int pointsIntervalOnContour;
     private static SpatialReference SpatialReference;
     private static long PotentialCandidates = 0;
     private static long TotalPointsCount = 0;
     private static long PointsAnalyzed = 0;
     private static List<int> HeightsToProcess = new List<int>();
     private static Dictionary<int, double> ContourLengths = new Dictionary<int, dou-
ble>();
     private static List<CandidateDam> chosenCandidates;
     private static CancelableProgressorSource cps;
     private static object lockingObject = new object();

     protected override async void OnClick()
     {
          SharedFunctions.Log("Search for candidate dams started");
          pointsIntervalOnContour = Convert.ToInt32(Parameter.PointIntervalBox.Text);
          DateTime startTime = DateTime.Now;
          chosenCandidates = new List<CandidateDam>();
          PotentialCandidates = 0;

          var pd = new ProgressDialog("Search for candidate dams", "Can-
celed", 100, false);
          cps = new CancelableProgressorSource(pd);
          cps.Progressor.Max = 100;
          PointsAnalyzed = 0;
          TotalPointsCount = 0;

          try
          {
               await Project.Current.SaveEditsAsync();
               BasicFeatureLayer layer = null;

               await QueuedTask.Run(async () =>
               {
                    if (!SharedFunctions.LayerExists("ContourPoints"))
                         return;

                    CancellationToken ctoken = new CancellationToken();

                    //create line feature layer if it does not exist
                    BasicFeatureLayer damCandidatesLayer = await CreateDamFea-
tureClass("DamCandidates");

                    var contourPointsLayer = MapView.Active.Map.FindLayers("Contour-
Points").FirstOrDefault();
                    layer = contourPointsLayer as BasicFeatureLayer;

                    // store the spatial reference of the current layer
                    SpatialReference = layer.GetSpatialReference();

                    //Cursor for selected points
                    RowCursor cursor = layer.GetSelection().Search();

                    //If no selection was set, use full points layer
                    if (layer.GetSelection().GetCount() == 0)
                         cursor = layer.Search();

                    Dictionary<int, SortedDictionary<int, SortedDictionary<long, Map-
Point>>> contourHeights = new Dictionary<int, SortedDictionary<int, SortedDiction-
ary<long, MapPoint>>>();

                    cps.Progressor.Status = "Loading ContourPoints into memory";
                    SharedFunctions.Log("Loading all ContourPoints into memory");
```

```
                while (cursor.MoveNext())
                {
                    if (ctoken.IsCancellationRequested)
                    {
                        SharedFunctions.Log("Canceled");
                        return;
                    }
                    using (Row row = cursor.Current)
                    {
                        var point = row[1] as MapPoint;
                        var pointID = (int)row[0];
                        var contourHeight = (int)(double)row[4];
                        var contourID = (int)row[2];

                        if (!ContourLengths.ContainsKey(contourID))
                            ContourLengths.Add(contourID, (double)row["Shape_Length"]);
                        if (!contourHeights.ContainsKey((int)contourHeight))
                            contourHeights.Add((int)contourHeight, new SortedDiction-
ary<int, SortedDictionary<long, MapPoint>>());
                        if (!contourHeights[contourHeight].ContainsKey((int)contourID))
                            contourHeights[contourHeight].Add((int)contourID, new Sort-
edDictionary<long, MapPoint>());
                        contourHeights[contourHeight][(int)contourID].Add(poin-
tID, point);
                        TotalPointsCount++;
                    }
                }
                cps.Progressor.Status = "Analyze Contours";
                SharedFunctions.Log("Analyze Contours");
                bool multiThreading = (Parameter.MultiThreadingBox == null || !Parame-
ter.MultiThreadingBox.IsChecked.HasValue || Parameter.MultiThreadingBox.Is-
Checked.Value);
                if (multiThreading)
                {
                    HeightsToProcess = contourHeights.Keys.ToList();
                    int ThreadCount = Math.Min(HeightsToProcess.Count, Environment.Pro-
cessorCount);
                    SharedFunctions.Log("Divided work into " + Thread-
Count + " threads...");
                    await Task.WhenAll(Enumerable.Range(1, ThreadCount).Se-
lect(c => Task.Run(
                        () =>
                        {
                            while (HeightsToProcess.Count > 0)// && !ctoken.IsCancella-
tionRequested)
                            {
                                int height = -1;
                                lock (HeightsToProcess)
                                {
                                    height = HeightsToProcess.FirstOrDefault();
                                    if (height != 0)
                                        HeightsToProcess.Remove(height);
                                }
                                if (height != -1)
                                {
                                    var calc = new Dictionary<int, SortedDiction-
ary<int, SortedDictionary<long, MapPoint>>>();
                                    calc.Add(height, contourHeights[height]);
                                    AnalyseContourHeights(calc, ctoken);
                                }
                            }
                        }
                        , ctoken))
                        );
                }
                else
                {
```

```
                    //Single Thread:
                    AnalyseContourHeights(contourHeights, ctoken);
                }
                cps.Progressor.Status = "Saving all " + chosenCandidates.Count + " can-
didates";
                SharedFunctions.Log("Saving all " + chosenCandidates.Count + " candi-
dates");
                foreach (var candidate in chosenCandidates)
                {
                    if (ctoken.IsCancellationRequested)
                    {
                        SharedFunctions.Log("Canceled");
                        return;
                    }
                    //Create coordinates for Polyline Feature with height Contour-
Height + 5 Meters!
                    List<Coordinate3D> coordinates = new List<Coordinate3D>() {
                        new Coordinate3D(candidate.StartPoint.X, candi-
date.StartPoint.Y, candidate.ContourHeight + 5),
                        new Coordinate3D(candidate.EndPoint.X, candidate.End-
Point.Y, candidate.ContourHeight + 5)};

                    //save all selected candidates into the db
                    var attributes = new Dictionary<string, object>
                            {
                                { "Shape", PolylineBuilder.CreatePolyline(coordi-
nates) },
                                { "ContourID", (long)candidate.ContourID },
                                { "StartPointID", (long)candidate.StartPointID },
                                { "EndPointID", (long)candidate.EndPointID },
                                { "ContourHeight", (short)candidate.ContourHeight },
                                { "LengthRating", (float)candidate.Rating },
                                { "DistanceOnLine", (long)candidate.DistanceOnLine },
                                { "Length", (short)candidate.Length },
                                { "StartPointDistance", (long)candidate.StartPointDis-
tance },
                                { "EndPointDistance", (long)candidate.EndPointDis-
tance },
                                { "DamSpansContourStart", (short)(candidate.DamSpans-
ContourStart ? 1 : 0) }
                            };
                    var editOp = new EditOperation() { Name = "Create dam candi-
date", SelectNewFeatures = false };
                    editOp.Create(damCandidatesLayer, attributes);
                    ////Execute the operations
                    editOp.Execute();
                }
            }, cps.Progressor);

            //save all edits
            await Project.Current.SaveEditsAsync();

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
        finally
        {
            DateTime endTime = DateTime.Now;
            SharedFunctions.Log("Analysed " + PotentialCandi-
dates.ToString("N0") + " candidates ( " + chosenCandidates.Count.ToString("N0") + " se-
lected) in " + (endTime - startTime).TotalSeconds.ToString("N") + " seconds");
        }
    }
```

```
    private static void AnalyseContourHeights(Dictionary<int, SortedDiction-
ary<int, SortedDictionary<long, MapPoint>>> contourHeights, CancellationToken ctoken)
    {
        int selectedCandidates = 0;
        foreach (var contourHeight in contourHeights)
        {
            //Analyse lines of one ContourLine
            foreach (var contourID in contourHeight.Value)
            {
                //skip contours with less than 2.000 m length
                if (contourID.Value.Count < (int)2000 / pointsIntervalOnContour)
                    continue;
                List<CandidateDam> candidates = AnalyseCountourPoints(con-
tourID.Value, contourHeight.Key, contourID.Key, ctoken);
                lock (lockingObject)
                {
                    chosenCandidates.AddRange(candidates);
                    selectedCandidates += candidates.Count;
                }
            }
            SharedFunctions.Log(selectedCandidates.ToString("N0") + " candidates se-
lected for " + contourHeight.Key + "m contours (Total: " + chosenCandi-
dates.Count.ToString("N0") + " of " + PotentialCandidates.ToString("N0") + " poten-
tials)");
        }
    }

    public async static Task<BasicFeatureLayer> CreateDamFeatureClass(string name)
    {
        var existingLayer = MapView.Active.Map.FindLayers(name).FirstOrDefault();
        if (existingLayer != null)
            return existingLayer as BasicFeatureLayer;

        SharedFunctions.Log("Creating DamCandidates layer");
        List<object> arguments = new List<object> { CoreModule.CurrentProject.De-
faultGeodatabasePath, name, "POLYLINE", "", "DISABLED", "ENABLED" };
        arguments.Add(SpatialReference);
        IGPResult result = await Geoprocessing.ExecuteToolAsync("Create-
Featureclass_management", Geoprocessing.MakeValueArray(arguments.ToArray()));
        var layer = MapView.Active.Map.FindLayers(name).FirstOrDefault() as BasicFea-
tureLayer;
        await SharedFunctions.ExecuteAddFieldTool(layer, "ContourID", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "StartPointID", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "EndPoin-
tID", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "ContourHeight", "SHORT");
        await SharedFunctions.ExecuteAddFieldTool(layer, "LengthRating", "FLOAT");
        await SharedFunctions.ExecuteAddFieldTool(layer, "DistanceOnLine", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "Length", "SHORT");
        await SharedFunctions.ExecuteAddFieldTool(layer, "StartPointDistance", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "EndPointDistance", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "DamHeight", "SHORT");
        await SharedFunctions.ExecuteAddFieldTool(layer, "DamVolume", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "ReservoirVolume", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "VolumeRating", "FLOAT");
        await SharedFunctions.ExecuteAddFieldTool(layer, "DamSpansCon-
tourStart", "SHORT");

        return layer;
    }

    private static List<CandidateDam> AnalyseCountourPoints(SortedDictionary<long, Map-
Point> points, int contourHeight, int contourID, CancellationToken ctoken)
    {
        int minPointID = (int)points.Min(c => c.Key);
        var first = points.First();
        var last = points.Last();
```

```
        //find out if the contour is a closed loop
        bool closedLoop = (first.Value.X == last.Value.X && first.Value.Y == last.Value
.Y);
        //remove the last point
        points.Remove((long)last.Key);
        List <CandidateDam> candidates = new List<CandidateDam>();
        SortedDictionary<long, MapPoint> endpoints = new SortedDictionary<long, Map-
Point>(points);
        //remove the first x points from the endpoints, as there is no valid candi-
date for a dam within the first 1000m
        for (int i = 0; i < 1000 / pointsIntervalOnContour; i++)
        {
            lock(lockingObject)
                PotentialCandidates++;
            if (endpoints.Keys.Count == 0)
                break;
            endpoints.Remove(endpoints.Keys.First());
        }
        foreach (var start in points)
        {
            lock(lockingObject)
            {
                PointsAnalyzed++;
                if(PointsAnalyzed % 1000 == 0)
                {
                    cps.Progressor.Value = Convert.ToUInt32(PointsAnalyzed * 100 / To-
talPointsCount);
                    cps.Progressor.Status = string.Format("{0} Points of {1} ana-
lyzed", PointsAnalyzed.ToString("n0"), TotalPointsCount.ToString("n0"));
                }
            }
            List<CandidateDam> pointCandidates = new List<CandidateDam>();
            List<CandidateDam> chosenPointCandidates = new List<CandidateDam>();
            foreach (var end in endpoints)
            {
                var DamSpansContourStart = false;
                lock (lockingObject)
                    PotentialCandidates++;
                //Manual Length calculation with pythagoras
                var Length = (deci-
mal)Math.Sqrt(Math.Pow(start.Value.X - end.Value.X, 2) + Math.Pow(start.Value.Y - end.Va
lue.Y, 2));
                if (Length > 1000)
                    continue;
                var distanceOnLine = Math.Abs((start.Key - end.Key) * pointsIntervalOn-
Contour);
                //if the current contour is a closed loop, always take the shorter dis-
tance on either side
                if(closedLoop && distanceOnLine > (decimal)ContourLengths[con-
tourID] / 2.0m)
                {
                    distanceOnLine = Convert.ToInt64(ContourLengths[contourID]) - dis-
tanceOnLine;
                    DamSpansContourStart = true;
                }
                //if a candidate is detected that has less than 1.000 m reservoir cir-
cumference, skip it (only happens when DamSpansContourStart)
                if (distanceOnLine < 1000)
                    continue;
                //if the distanceOnLine grows bigger than 30.000 m we stop test-
ing this startpoint, as the resulting reservoir would exceed our target magnitude
                if (distanceOnLine > 30000 && !closedLoop)
                    break;
                //only save candidates that have a length rating > 10
                if (distanceOnLine / Length < 10)
                    continue;
```

```
                CandidateDam cand = CreateCandidateDam(contourHeight, minPoin-
tID, start, end, Length, distanceOnLine, DamSpansContourStart, contourID);
                pointCandidates.Add(cand);
            }
            //if closed loop and result was over 30.000 m then delete now
            if(closedLoop)
            {
                foreach (var item in pointCandidates.Where(c => c.Distance-
OnLine > 30000).ToList())
                {
                    candidates.Remove(item);
                }
            }
            if (pointCandidates.Count > 0)
            {
                int pointCandidateLimit = 3;
                decimal lastRating = 0;
                //choose max 3 candidates per starting point and try to pre-
serve real alternative dam variants (crossing of multiple branches)
                foreach (var pointCandidate in pointCandidates.OrderByDescend-
ing(c => c.Rating))
                {
                    if (pointCandidateLimit == 0)
                        break;
                    //if the new rating is more than 20% worse than the last rat-
ing, break the loop
                    if (pointCandidate.Rating < lastRating * 0.8m)
                        break;
                    //keep only the best candidate within a dis-
tance of 2.000 m on each side
                    if (chosenPointCandidates.Any(c => Math.Abs(c.Distance-
OnLine - pointCandidate.DistanceOnLine) < 1000))
                        continue;
                    //if there is already an obviously better candi-
date (lower length AND more distanceOnLine, then skip as well
                    //if (chosenPointCandidates.Any(c => c.Length < pointCandi-
date.Length && c.DistanceOnLine > pointCandidate.DistanceOnLine))
                    //    continue;

                    chosenPointCandidates.Add(pointCandidate);
                    pointCandidateLimit--;
                    lastRating = pointCandidate.Rating;
                }
                candidates.AddRange(chosenPointCandidates);
            }
            //remove one more point from the end-
points, so the first one to check against will be at least 1000m ahead again
            if (endpoints.Keys.Count > 0)
                endpoints.Remove(endpoints.Keys.First());
        }
        //now make sure the same applies for the endpoints:
        foreach (var endPointID in candidates.Select(c => c.EndPointID).Dis-
tinct().ToList())
        {
            var endpointCandidates = candidates.Where(c => c.EndPointID == endPoin-
tID || c.StartPointID == endPointID).ToList();
            foreach (var endPointCandidate in endpointCandidates)
            {
                //keep only the best candidate of this end-
point within a range of 2.000 m on each side
                if (endpointCandidates.Any(c => c.Rating > endPointCandidate.Rat-
ing && Math.Abs(c.DistanceOnLine - endPointCandidate.DistanceOnLine) < 2000))
                    candidates.Remove(endPointCandidate);
                //remove, if there are already obviously better candi-
dates (lower length AND more distanceOnLine)
                //This also removes potentially good candidates fur-
ther away... this could be restricted by a distance limit...
```

```
                else if (candidates.Any(c => c.Length < endPointCandi-
date.Length && c.DistanceOnLine > endPointCandidate.DistanceOnLine && c.StartPoin-
tID <= endPointCandidate.StartPointID && c.EndPointID >= endPointCandidate.EndPointID))
                    candidates.Remove(endPointCandidate);
            }
        }
        int vicinity = 1000 / pointsIntervalOnContour;
        //select all candidates that are in the vicinity of another candi-
date on the same contour but are lower ranked
        //this can only be selected for now that we are sure to have true dam candi-
dates in the list
        var candidatesToDelete = candidates.Where(c => candidates.Any(d => d.Con-
tourID == c.ContourID && d != c && c.Rating < d.Rating &&
                                                                    ( (Math.Abs(c.S
tartPointID - d.StartPointID) < vicinity && Math.Abs(c.EndPointID - d.EndPointID) < vi-
cinity)
                                                                     || (Math.Abs(c
.StartPointID - d.EndPointID) < vicinity && Math.Abs(c.EndPointID - d.StartPoin-
tID) < vicinity)
                                                                    )
                                                                )
                                                        ).ToList();
        foreach (var candidateToDelete in candidatesToDelete)
        {
            candidates.Remove(candidateToDelete);
        }
        //if the contour is a closed loop, we have to have another look at neigh-
bors across the start/end of the contour
        if(closedLoop)
        {
            var candidatesToDelete2 = candidates.Where(c => candidates.Any(d => d.Con-
tourID == c.ContourID && d != c && c.Rating < d.Rating &&
                                                                (((points.Count - Math.Abs(
c.StartPointID - d.StartPointID)) < vicinity && Math.Abs(c.EndPointID - d.EndPoin-
tID) < vicinity)
                                                                 ||
                                                                 (Math.Abs(c.StartPoin-
tID - d.StartPointID) < vicinity && (points.Count - Math.Abs(c.EndPointID - d.EndPoin-
tID)) < vicinity)
                                                                 ||
                                                                 ((points.Count - Math.Abs(
c.StartPointID - d.EndPointID)) < vicinity && Math.Abs(c.EndPointID - d.StartPoin-
tID) < vicinity)
                                                                 ||
                                                                 (Math.Abs(c.StartPoin-
tID - d.EndPointID) < vicinity && (points.Count - Math.Abs(c.EndPointID - d.StartPoin-
tID)) < vicinity)
                                                                 )
                                                             )
                                                    ).ToList();
            foreach (var candidateToDelete in candidatesToDelete2)
            {
                candidates.Remove(candidateToDelete);
            }
        }
        return candidates;
    }

    private static CandidateDam CreateCandidateDam(int contourHeight, int minPoin-
tID,  KeyValuePair<long, MapPoint> start, KeyValuePair<long, MapPoint> end, deci-
mal Length, long distanceOnLine, bool damSpansContourStart, int contourID)
    {
        CandidateDam cand = new CandidateDam();
        cand.ContourID = contourID;
        cand.ContourHeight = contourHeight;
        cand.StartPoint = start.Value;
        cand.StartPointID = start.Key;
```

```
            cand.StartPointDistance = (start.Key - minPointID) * pointsIntervalOnContour;
            cand.EndPointID = end.Key;
            cand.EndPoint = end.Value;
            cand.EndPointDistance = (end.Key - minPointID) * pointsIntervalOnContour;
            cand.Length = Length;
            cand.DistanceOnLine = distanceOnLine;
            cand.DamSpansContourStart = damSpansContourStart;
            return cand;
        }
    }
```

A.4 DamVolumeButton

```
internal class DamVolumeButton : Button
{
    private int IncomingCandidates = 0;
    private int OutgoingCandidates = 0;
    protected override async void OnClick()
    {
        SharedFunctions.Log("DamVolume Calculation started");
        DateTime startTime = DateTime.Now;
        await Project.Current.SaveEditsAsync();
        IncomingCandidates = 0;
        OutgoingCandidates = 0;
        List<CandidateDam> candidates = new List<CandidateDam>();
        List<CandidateDam> candidates2Delete = new List<CandidateDam>();
        try
        {
            await QueuedTask.Run(async () =>
            {
                if (!SharedFunctions.LayerExists("DamCandidates") || !SharedFunc-
tions.LayerExists("Contours"))
                    return;
                var _damCandidatesLayer = MapView.Active.Map.FindLayers("DamCandi-
dates").FirstOrDefault();

                BasicFeatureLayer damCandidatesLayer = _damCandidatesLayer as BasicFea-
tureLayer;
                SharedFunctions.LoadDamCandidatesFromLayer(candidates, damCandidates-
Layer);
                IncomingCandidates = candidates.Count();
                OutgoingCandidates = IncomingCandidates;

                //create stacked profile
                string inputDEM = Parameter.DEMCombo.SelectedItem.ToString();
                var valueArray = Geoprocessing.MakeValueArray(damCandidates-
Layer.Name, inputDEM, "DamProfile", null);
                var environments = Geoprocessing.MakeEnvironmentArray(overwriteout-
put: true);
                var cts = new CancellationTokenSource();
                await Project.Current.SaveEditsAsync();
                var res = await Geoprocessing.ExecuteToolAsync("StackProfile", valueAr-
ray, environments, cts.Token, null, GPExecuteToolFlags.Default);

                //analyse profile and add data to line feature or delete line feature
                //profile can be used to calculate frontal dam area and dam volume?!
                var damProfileTable = MapView.Active.Map.FindStandaloneTables("DamPro-
file").FirstOrDefault();
                if (damProfileTable == null)
                {
                    SharedFunctions.Log("No DamProfile Table found!");
                }

                CandidateDam cand = null;
                double prev_dist = 0;
                int prev_lineID = -1;
```

```
                using (var profileCursor = damProfileTable.Search())
                {
                    while (profileCursor.MoveNext())
                    {
                        using (Row profileRow = profileCursor.Current)
                        {
                            var first_dist = (double)profileRow[1];
                            var first_z = (double)profileRow[2];
                            var line_ID = (int)profileRow[5];

                            if (prev_lineID != line_ID)
                            {
                                prev_dist = 0;
                                cand = candidates.SingleOrDefault(c => c.Ob-
jectID == line_ID);
                                // set Volume and ZMin to the default values
                                cand.DamVolume = 0;
                                cand.ZMin = cand.ContourHeight;
                            }
                            else if (candidates2Delete.Contains(cand))
                                continue;
                            //set lowest point of dam to calculate max dam height later
                            if (cand.ZMin > (int)first_z)
                                cand.ZMin = (int)first_z;

                            if ((int)first_z > (cand.ContourHeight + 5))
                            {
                                candidates2Delete.Add(cand);
                                continue;
                            }

                            //add volume of current block of dam... the assump-
tion is a triangle dam shape (cross section) with alpha = 45°
                            //thus the area of the cross section is calcu-
lated by <height²> and the volume by <height² * length>
                            cand.DamVolume += (long)(Math.Pow((cand.Contour-
Height - first_z), 2) * (first_dist - prev_dist));

                            prev_lineID = line_ID;
                            prev_dist = first_dist;
                        }
                    }
                }

                await DeleteCandidates(candidates, damCandidatesLayer, candidates2De-
lete);
                //remove candidates with dam heights outside of the limits
                await DeleteCandidates(candidates, damCandidatesLayer, candi-
dates.Where(c => (c.ContourHeight - c.ZMin) < 10 || (c.Contour-
Height - c.ZMin) > 300).ToList());

                //update the new attributes to the feature
                foreach (var candidate in candidates)
                {
                    var editOp2 = new EditOperation();
                    Dictionary<string, object> attributes = new Dictionary<string, ob-
ject>();
                    attributes.Add("DamHeight", (short)(candidate.ContourHeight - can-
didate.ZMin));
                    attributes.Add("DamVolume", (long)(candidate.DamVolume));
                    editOp2.Modify(damCandidatesLayer, candidate.ObjectID, attributes);
                    var resu = await editOp2.ExecuteAsync();
                }

                await Project.Current.SaveEditsAsync();
            });
        }
```

```
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
            finally
            {
                DateTime endTime = DateTime.Now;
                SharedFunctions.Log("Analysed in " + (endTime - startTime).TotalSec-
onds.ToString("N") + " seconds completed (" + OutgoingCandi-
dates.ToString("N0") + " of " + IncomingCandidates.ToString("N0") + " candi-
dates left)");
            }
        }

        private async Task DeleteCandidates(List<CandidateDam> candidates, BasicFeature-
Layer damCandidatesLayer, List<CandidateDam> candidatesToDelete)
        {
            foreach (var candidateToDelete in candidatesToDelete)
            {
                candidates.Remove(candidateToDelete);
                OutgoingCandidates--;
            }
            if (candidatesToDelete.Count > 0)
            {
                var deleteOp = new EditOperation();
                deleteOp.Delete(damCandidatesLayer, candidatesToDelete.Select(c => c.Ob-
jectID).ToList());
                await deleteOp.ExecuteAsync();
            }
        }
}
```

A.5 CreateReservoirPolygonsButton

```
internal class CreateReservoirPolygonsButton : Button
{
    private static SpatialReference SpatialReference;
    private static List<ReservoirSurface> surfaces;
    private static BasicFeatureLayer reservoirSurfacesLayer;
    private static BasicFeatureLayer contourLayer;
    private static BasicFeatureLayer damLayer;
    private static List<long> ContoursToProcess = new List<long>();
    protected override async void OnClick()
    {
        SharedFunctions.Log("Starting To Create Reservoir Polygons");
        DateTime startTime = DateTime.Now;
        surfaces = new List<ReservoirSurface>();

        try
        {
            await Project.Current.SaveEditsAsync();
            await QueuedTask.Run(async () =>
            {
                if (!SharedFunctions.LayerExists("DamCandidates") || !SharedFunc-
tions.LayerExists("Contours"))
                    return;
                damLayer = MapView.Active.Map.FindLayers("DamCandidates").FirstOrDe-
fault() as BasicFeatureLayer;
                contourLayer = MapView.Active.Map.FindLayers("Contours").FirstOrDe-
fault() as BasicFeatureLayer;

                SpatialReference = damLayer.GetSpatialReference();
                reservoirSurfacesLayer = await CreatePolygonFeatureClass("ReservoirSur-
faces");

                ConstructPolygons();
```

```
            });
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
        finally
        {
            DateTime endTime = DateTime.Now;
            SharedFunctions.Log("Analysed in " + (endTime - startTime).TotalSec-
onds.ToString() + " seconds");
        }
    }
    public async static Task<BasicFeatureLayer> CreatePolygonFeatureClass(string name)
    {
        var existingLayer = MapView.Active.Map.FindLayers(name).FirstOrDefault();
        if (existingLayer != null)
            return existingLayer as BasicFeatureLayer;
        List<object> arguments = new List<object> { CoreModule.CurrentProject.DefaultGe-
odatabasePath, name, "POLYGON", "", "DISABLED", "ENABLED" };
        arguments.Add(SpatialReference);
        IGPResult result = await Geoprocessing.ExecuteToolAsync("CreateFeatureclass_man-
agement", Geoprocessing.MakeValueArray(arguments.ToArray()));
        var layer = MapView.Active.Map.FindLayers(name).FirstOrDefault() as BasicFea-
tureLayer;
        await SharedFunctions.ExecuteAddFieldTool(layer, "DamID", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "ContourHeight", "SHORT");

        return layer;
    }

    private async static void ConstructPolygons()
    {
        List<CandidateDam> candidates = new List<CandidateDam>();
        SharedFunctions.LoadDamCandidatesFromLayer(candidates, damLayer);
        List<Contour> contours = new List<Contour>();
        SharedFunctions.LoadContoursFromLayer(contours, contourLayer);
        //select only contours, that actually have candidate dams on it
        contours = contours.Where(c => candidates.Any(d => d.ContourID == c.Ob-
jectID)).ToList();
        bool multiThreading = (Parameter.MultiThreadingBox == null || !Parameter.Multi-
ThreadingBox.IsChecked.HasValue || Parameter.MultiThreadingBox.IsChecked.Value);
        //ArcGIS.Core.Geometry Tools currently don't seem to support multithreading.
        //Question https://community.esri.com/thread/243147-multithreading-parallel-pro-
cessing-in-arcgis-pro-addin
        //received no answer so far. Until a solution is found, multithread-
ing logic has to be deactivated
        multiThreading = false;

        if (multiThreading)
        {
            List<PolylineBuilder> polylineBuilders = new List<PolylineBuilder>();
            for (int i = 0; i < Environment.ProcessorCount; i++)
            {
                polylineBuilders.Add(new PolylineBuilder(SpatialReference));
            }

            ContoursToProcess = contours.Select(c => c.ObjectID).ToList();
            SharedFunctions.Log("Divided work into " + Environment.Processor-
Count + " threads for all logical Processors...");
            await Task.WhenAll(polylineBuilders.Select(c => Task.Run(//Enumera-
ble.Range(1, Environment.ProcessorCount).Select(c => Task.Run(
                async () =>
                {
                    while (ContoursToProcess.Count > 0)
                    {
                        long contourID = -1;
```

```
                        lock (ContoursToProcess)
                        {
                            contourID = ContoursToProcess.FirstOrDefault();
                            if (contourID != 0)
                                ContoursToProcess.Remove(contourID);
                        }
                        if (contourID != -1)
                        {
                            var calc = new List<Contour>() { contours.Single(d => d.Ob-
jectID == contourID) };
                            await PolygonsForContours(candidates, calc, c);
                        }
                    }
                }
                ))
                );
        }
        else await PolygonsForContours(candidates, contours, new PolylineBuilder(Spa-
tialReference));

        SharedFunctions.Log("Save all " + surfaces.Count + " surfaces");
        foreach (var surface in surfaces)
        {
            var attributes = new Dictionary<string, object>
                            {
                                { "Shape", surface.Polygon },
                                { "DamID", (long)surface.DamID },
                                { "ContourHeight", (short)surface.ContourHeight }
                            };
            var createOperation = new EditOperation() { Name = "Create reservoir poly-
gon", SelectNewFeatures = false };
            createOperation.Create(reservoirSurfacesLayer, attributes);
            await createOperation.ExecuteAsync();
        }
        await Project.Current.SaveEditsAsync();
    }

    private static async Task PolygonsForContours(List<CandidateDam> candi-
dates, List<Contour> contours, PolylineBuilder polylineBuilder)
    {
        foreach (var contour in contours)
        {
            var contourGeometry = contour.Polyline;
            int counter = 0;
            int contourHeight = 0;
            foreach (var candidate in candidates.Where(c => c.ContourID == contour.Ob-
jectID).ToList())
            {
                try
                {
                    while (polylineBuilder.CountParts > 0)
                    {
                        polylineBuilder.RemovePart(0);
                    }

                    //add the full contour
                    polylineBuilder.AddParts(contourGeometry.Parts);
                    //split at the endpoint
                    polylineBuilder.SplitAtDistance(candidate.EndPointDis-
tance, false, true);

                    if (candidate.DamSpansContourStart)
                    {
                        //remove the part of the contour after the endpoint
                        //split at the startpoint
                        if (candidate.StartPointDistance != 0)
                        {
```

```
                            polylineBuilder.SplitAtDistance(candidate.StartPointDis-
tance, false, true);
                            //remove the part of the polyline before the startpoint
                            polylineBuilder.RemovePart(1);
                        }
                        //Handle the situation, when the startpoint is on the very be-
ginning of the contour line
                        else
                        {
                            polylineBuilder.RemovePart(0);
                        }
                    }
                    else
                    {
                        //remove the part of the contour after the endpoint
                        polylineBuilder.RemovePart(1);
                        //split at the startpoint
                        if (candidate.StartPointDistance != 0)
                        {
                            polylineBuilder.SplitAtDistance(candidate.StartPointDis-
tance, false, true);
                            //remove the part of the polyline before the startpoint
                            polylineBuilder.RemovePart(0);
                        }
                    }
                    var newPolygon3D = PolygonBuilder.CreatePolygon(polylineBuilder.To-
Geometry().Copy3DCoordinatesToList(), SpatialReference);

                    ReservoirSurface surface = new ReservoirSurface();
                    surface.Polygon = GeometryEngine.Instance.Move(newPoly-
gon3D, 0, 0, candidate.ContourHeight) as Polygon;
                    surface.DamID = (long)candidate.ObjectID;
                    surface.ContourHeight = (short)candidate.ContourHeight;
                    surfaces.Add(surface);

                    counter++;
                    contourHeight = candidate.ContourHeight;
                }
                catch (Exception ex)
                {
                    SharedFunctions.Log("Error for DamID " + candidate.Ob-
jectID + " for ContourHeight " + candidate.ContourHeight + ": (" + ex.Message + ")");
                }
            }

            SharedFunctions.Log("Created " + surfaces.Count.ToString("N0") + " Poly-
gons ... " + counter.ToString("N0") + " for Contour " + contour.ObjectID + " (" + con-
tour.Height + " m)");
        }
    }
}
```

A.6 ReservoirVolumeAndRankingButton

```
internal class ReservoirVolumeAndRankingButton : Button
{
    protected override async void OnClick()
    {
        if(!SharedFunctions.LayerExists("TIN") || !SharedFunctions.LayerExists("DamCan-
didates") || !SharedFunctions.LayerExists("ReservoirSurfaces"))
            return;

        await Project.Current.SaveEditsAsync();

        string TINLayer = "TIN";
```

```
        string damCandidatesLayer = "DamCandidates";
        string reservoirSurfacesLayer = "ReservoirSurfaces";
        var args = Geoprocessing.MakeValueArray(reservoirSurfacesLayer, damCandidates-
Layer, TINLayer, damCandidatesLayer);
        await SharedFunctions.RunModel(args, "Reservoir Volume");
    }
}
```

A.7 PairReservoirsButton

```
internal class PairReservoirsButton : Button
{
    private static SpatialReference SpatialReference;
    protected override async void OnClick()
    {
        SharedFunctions.Log("Starting To Pair Reservoirs");
        DateTime startTime = DateTime.Now;

        await Project.Current.SaveEditsAsync();

        try
        {
            await QueuedTask.Run(async () =>
            {
                if (!SharedFunctions.LayerExists("DamCandidates") || !SharedFunc-
tions.LayerExists("ReservoirSurfaces"))
                    return;
                var damCandidatesLayer = MapView.Active.Map.FindLayers("DamCandi-
dates").FirstOrDefault();
                var reservoirSurfacesLayer = MapView.Active.Map.FindLayers("Reser-
voirSurfaces").FirstOrDefault();

                SpatialReference = damCandidatesLayer.GetSpatialReference();
                var reservoirPairsLayer = await CreateFeatureClass("ReservoirPairs");

                await FindPairs(reservoirPairsLayer, reservoirSurfacesLayer as BasicFea-
tureLayer, damCandidatesLayer as BasicFeatureLayer);
            });
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
        finally
        {
            DateTime endTime = DateTime.Now;
            SharedFunctions.Log("Analysed in " + (endTime - startTime).TotalSec-
onds.ToString() + " seconds");
        }
    }
    public async static Task<BasicFeatureLayer> CreateFeatureClass(string name)
    {
        var existingLayer = MapView.Active.Map.FindLayers(name).FirstOrDefault();
        if (existingLayer != null)
            return existingLayer as BasicFeatureLayer;
        List<object> arguments = new List<object> { CoreModule.CurrentProject.DefaultGe-
odatabasePath, name, "POLYLINE", "", "DISABLED", "ENABLED" };
        arguments.Add(SpatialReference);
        IGPResult result = await Geoprocessing.ExecuteToolAsync("CreateFeatureclass_man-
agement", Geoprocessing.MakeValueArray(arguments.ToArray()));
        var layer = MapView.Active.Map.FindLayers(name).FirstOrDefault() as BasicFea-
tureLayer;
        await SharedFunctions.ExecuteAddFieldTool(layer, "LowerDamID", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "UpperDamID", "LONG");
        await SharedFunctions.ExecuteAddFieldTool(layer, "CapacityInMWh", "FLOAT");
        await SharedFunctions.ExecuteAddFieldTool(layer, "Distance", "LONG");
```

```
        await SharedFunctions.ExecuteAddFieldTool(layer, "LowerHeight", "SHORT");
        await SharedFunctions.ExecuteAddFieldTool(layer, "UpperHeight", "SHORT");
        await SharedFunctions.ExecuteAddFieldTool(layer, "CapacityDistanceRa-
tio", "FLOAT");
        await SharedFunctions.ExecuteAddFieldTool(layer, "UsableHeightDiffer-
ence", "SHORT");
        await SharedFunctions.ExecuteAddFieldTool(layer, "CapacityUtiliza-
tion", "FLOAT");

        return layer;
    }

    private async static Task<bool> FindPairs(BasicFeatureLayer reservoir-
PairsLayer, BasicFeatureLayer reservoirSurfacesLayer, BasicFeatureLayer damLayer)
    {
        List<ReservoirPair> reservoirPairs = new List<ReservoirPair>();
        List<CandidateDam> res1List = new List<CandidateDam>();
        SharedFunctions.LoadDamCandidatesFromLayer(res1List, damLayer);
        List<ReservoirSurface> reservoirSurfaceList = new List<ReservoirSurface>();
        SharedFunctions.LoadReservoirSurfacesFromLayer(reservoirSurfaceList, reser-
voirSurfacesLayer);
        List<CandidateDam> res2List = new List<CandidateDam>(res1List);
        int analyzedCounter = 0;
        int createdCounter = 0;
        foreach (var dam1 in res1List)
        {
            res2List.Remove(dam1);
            foreach (var dam2 in res2List)
            {
                try
                {
                    analyzedCounter++;
                    CandidateDam lowerDam = null;
                    CandidateDam upperDam = null;
                    if (dam1.ContourHeight > dam2.ContourHeight)
                    {
                        lowerDam = dam2;
                        upperDam = dam1;
                    }
                    else
                    {
                        lowerDam = dam1;
                        upperDam = dam2;
                    }

                    //check for height-difference:
                    int usableHeightDifference = upperDam.ContourHeight - upperDam.Dam-
Height - lowerDam.ContourHeight;
                    if (usableHeightDifference < 50)
                        continue;

                    //check for horizontal distance
                    Coordinate3D lowerDamCenter = new Coordinate3D((low-
erDam.StartPoint.X + lowerDam.EndPoint.X) / 2, (lowerDam.StartPoint.Y + lowerDam.End-
Point.Y) / 2, (lowerDam.StartPoint.Z + lowerDam.EndPoint.Z) / 2);
                    Coordinate3D upperDamCenter = new Coordinate3D((up-
perDam.StartPoint.X + upperDam.EndPoint.X) / 2, (upperDam.StartPoint.Y + upperDam.End-
Point.Y) / 2, (upperDam.StartPoint.Z + upperDam.EndPoint.Z) / 2);
                    decimal distanceBetweenDamCenters = SharedFunctions.DistanceBe-
tween(lowerDamCenter, upperDamCenter);
                    if (distanceBetweenDamCenters > 5000)
                        continue;

                    //check for reservoir overlap //NOT NECCESSARY due to height-differ-
ence check, where all corresponding cases are already filtered
```

```
                    //if (GeometryEngine.Instance.Overlaps(reservoirSurfaceList.Sin-
gle(c => c.DamID == dam1.ObjectID).Polygon, reservoirSurfaceList.Sin-
gle(c => c.DamID == dam1.ObjectID).Polygon))
                    //    continue;

                    //calculate CapacityInMWh:
                    long usableVolume = upperDam.ReservoirVolume;
                    if (lowerDam.ReservoirVolume < usableVolume)
                        usableVolume = lowerDam.ReservoirVolume;

                    //only assume 85% of the water as usable in reality
                    usableVolume = (long)(0.85 * usableVolume);
                    float capacityInMWh = (float)(1000 * 9.8 * usableHeightDiffer-
ence * usableVolume * 0.9) / (float)((long)3600 * 1000000);

                    //calculate utilizationPercentage:
                    float capacityUtilization = 0;
                    if (upperDam.ReservoirVolume != 0)
                        capacityUtilization = (float)lowerDam.ReservoirVolume / up-
perDam.ReservoirVolume;
                    if (capacityUtilization > 1)
                        capacityUtilization = 1 / (float)capacityUtilization;

                    //check for Utilization of at least 75%
                    if (capacityUtilization < 0.75)
                        continue;

                    decimal capacityDistanceRatio = (decimal)capacityInMWh * 100 / dis-
tanceBetweenDamCenters;

                    List<Coordinate3D> coordinates = new List<Coordinate3D>() { low-
erDamCenter, upperDamCenter };
                    Polyline polyline = PolylineBuilder.CreatePolyline(coordinates);

                    ReservoirPair reservoirPair = new ReservoirPair();
                    reservoirPair.LowerDam = lowerDam;
                    reservoirPair.UpperDam = upperDam;
                    reservoirPair.LowerDamCenter = lowerDamCenter;
                    reservoirPair.UpperDamCenter = upperDamCenter;
                    reservoirPair.Polyline = polyline;
                    reservoirPair.LowerDamID = lowerDam.ObjectID;
                    reservoirPair.UpperDamID = upperDam.ObjectID;
                    reservoirPair.CapacityInMWh = capacityInMWh;
                    reservoirPair.Distance = distanceBetweenDamCenters;
                    reservoirPair.LowerHeight = lowerDam.ContourHeight;
                    reservoirPair.UpperHeight = upperDam.ContourHeight;
                    reservoirPair.CapacityDistanceRatio = capacityDistanceRatio;
                    reservoirPair.UsableHeightDifference = usableHeightDifference;
                    reservoirPair.CapacityUtilization = capacityUtilization;

                    reservoirPairs.Add(reservoirPair);
                }
                catch (Exception ex)
                {
                    SharedFunctions.Log("Error for at-
tempted Pair with dam1: " + dam1.ObjectID + " and dam2: " + dam2.Ob-
jectID + " (" + ex.Message + ")");
                }
            }
        }
        //try to further minimize the reservoirPairs selection by only keeping
        //the best connection within a buffer of 100 m (around both dam center points)
        List<ReservoirPair> pairsToDelete = new List<ReservoirPair>();
        foreach (var reservoirPair in reservoirPairs.ToList())
        {
            if (pairsToDelete.Contains(reservoirPair))
                continue;
```

```
            var similarPairs = reservoirPairs.Where(c => SharedFunctions.DistanceBe-
tween(reservoirPair.LowerDamCenter, c.LowerDamCenter) <= 100
                                        && SharedFunctions.DistanceBetween(reservoir-
Pair.UpperDamCenter, c.UpperDamCenter) <= 100).ToList();
            if (similarPairs.Count > 1)
            {
                var bestPair = similarPairs.OrderByDescending(c => c.CapacityDistanceRa-
tio * (decimal)c.CapacityUtilization).First();
                similarPairs.Remove(bestPair);
                pairsToDelete = pairsToDelete.Union(similarPairs).ToList();
            }
        }
        foreach (var pairToDelete in pairsToDelete)
        {
            reservoirPairs.Remove(pairToDelete);
        }

        //insert the remaining objects into the DB
        foreach (var reservoirPair in reservoirPairs)
        {
            var attributes = new Dictionary<string, object>
                            {
                                { "Shape", reservoirPair.Polyline },
                                { "LowerDamID", (long)reservoirPair.LowerDamID },
                                { "UpperDamID", (long)reservoirPair.UpperDamID },
                                { "CapacityInMWh", (float)reservoirPair.CapacityInMWh },
                                { "Distance", (long)reservoirPair.Distance },
                                { "LowerHeight", (short)reservoirPair.LowerHeight },
                                { "UpperHeight", (short)reservoirPair.UpperHeight },
                                { "CapacityDistanceRatio", (float)reservoirPair.Capaci-
tyDistanceRatio },
                                { "UsableHeightDifference", (short)reservoirPair.Usable-
HeightDifference },
                                { "CapacityUtilization", (float)reservoirPair.Capaci-
tyUtilization }
                            };
            var createOperation = new EditOperation() { Name = "Create reser-
voir pair", SelectNewFeatures = false };
            createOperation.Create(reservoirPairsLayer, attributes);
            await createOperation.ExecuteAsync();
            createdCounter++;
        }

        SharedFunctions.Log(analyzedCounter + " combinations analysed and " + created-
Counter + " viable pairs found");

        await Project.Current.SaveEditsAsync();

        return true;
    }
}
```

A.8 VisualizeDamButton

```
internal class VisualizeDamButton : Button
{
    private static List<IDisposable> _overlayObjectList = new List<IDisposable>();
    private static SpatialReference SpatialReference;
    protected override async void OnClick()
    {
        SharedFunctions.Log("Starting To Create 3D Visualization");
        DateTime startTime = DateTime.Now;

        CIMLineSymbol symbol = SymbolFactory.Instance.ConstructLineSymbol(ColorFac-
tory.Instance.GreyRGB, 5.0, SimpleLineStyle.Solid);
        CIMSymbolReference symbolReference = symbol.MakeSymbolReference();
```

```
        try
        {
            await QueuedTask.Run(async () =>
            {
                if (!SharedFunctions.LayerExists("DamCandidates") || !SharedFunc-
tions.LayerExists("ReservoirSurfaces"))
                    return;
                var damCandidatesLayer = MapView.Active.Map.FindLayers("DamCandi-
dates").FirstOrDefault();
                var reservoirSurfacesLayer = MapView.Active.Map.FindLayers("Reser-
voirSurfaces").FirstOrDefault();

                SpatialReference = damCandidatesLayer.GetSpatialReference();
                var damVisLayer = await CreateMultiPatchFeatureClass("DamVisualiza-
tion");
                var reservoirVisLayer = await CreatePolygonFeatureClass("ReservoirVisu-
alization");

                Visualize(damVisLayer, reservoirVisLayer, reservoirSurfac-
esLayer as BasicFeatureLayer, damCandidatesLayer as BasicFeatureLayer);
            });
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
        finally
        {
            DateTime endTime = DateTime.Now;
            SharedFunctions.Log("Visualized in " + (endTime - startTime).TotalSec-
onds.ToString() + " seconds");
        }
    }
    public async static Task<BasicFeatureLayer> CreateMultiPatchFea-
tureClass(string name)
    {
        var existingLayer = MapView.Active.Map.FindLayers(name).FirstOrDefault();
        if (existingLayer != null)
            return existingLayer as BasicFeatureLayer;
        List<object> arguments = new List<object> { CoreModule.CurrentProject.DefaultGe-
odatabasePath, name, "MULTIPATCH", "", "DISABLED", "ENABLED" };
        arguments.Add(SpatialReference);
        IGPResult result = await Geoprocessing.ExecuteToolAsync("CreateFeatureclass_man-
agement", Geoprocessing.MakeValueArray(arguments.ToArray()));
        var layer = MapView.Active.Map.FindLayers(name).FirstOrDefault() as BasicFea-
tureLayer;
        await SharedFunctions.ExecuteAddFieldTool(layer, "DamID", "LONG");

        return layer;
    }
    public async static Task<BasicFeatureLayer> CreatePolygonFeatureClass(string name)
    {
        var existingLayer = MapView.Active.Map.FindLayers(name).FirstOrDefault();
        if (existingLayer != null)
            return existingLayer as BasicFeatureLayer;
        List<object> arguments = new List<object> { CoreModule.CurrentProject.DefaultGe-
odatabasePath, name, "POLYGON", "", "DISABLED", "ENABLED" };
        arguments.Add(SpatialReference);
        IGPResult result = await Geoprocessing.ExecuteToolAsync("CreateFeatureclass_man-
agement", Geoprocessing.MakeValueArray(arguments.ToArray()));
        var layer = MapView.Active.Map.FindLayers(name).FirstOrDefault() as BasicFea-
tureLayer;
        await SharedFunctions.ExecuteAddFieldTool(layer, "DamID", "LONG");

        return layer;
    }
```

```
    private async static void Visualize(BasicFeatureLayer dam3dLayer, BasicFeature-
Layer reservoirVisLayer, BasicFeatureLayer reservoirSurfacesLayer, BasicFeature-
Layer damLayer)
    {
        SharedFunctions.DeleteAllFeatures(reservoirVisLayer);
        SharedFunctions.DeleteAllFeatures(dam3dLayer);

        List<long> damIDs = new List<long>();
        var reservoirPairsLayer = MapView.Active.Map.FindLayers("Reservoir-
Pairs").FirstOrDefault();
        if (reservoirPairsLayer != null && ((BasicFeatureLayer)reservoirPairsLayer).Se-
lectionCount > 0)
        {
            var reservoirPairsCursor = ((BasicFeatureLayer)reservoirPairsLayer).Get-
Selection().Search();
            while (reservoirPairsCursor.MoveNext())
            {
                using (Row row = reservoirPairsCursor.Current)
                {
                    int damId = (int)row["LowerDamId"];
                    if (!damIDs.Contains(damId))
                        damIDs.Add(damId);
                    damId = (int)row["UpperDamId"];
                    if (!damIDs.Contains(damId))
                        damIDs.Add(damId);
                }
            }
        }
        if (damLayer.SelectionCount > 0)
        {
            var damCursor = damLayer.GetSelection().Search();
            while (damCursor.MoveNext())
            {
                using (Row row = damCursor.Current)
                {
                    int damId = (int)row["ObjectID"];
                    if (!damIDs.Contains(damId))
                        damIDs.Add(damId);
                }
            }
        }
        List<CandidateDam> candidates = new List<CandidateDam>();
        SharedFunctions.LoadDamCandidatesFromLayer(candidates, damLayer);
        foreach (var dam in candidates.Where(c => damIDs.Contains(c.ObjectID)).ToList())
        {
            double contourHeight = dam.ContourHeight;

            try
            {
                MapPoint startPoint = dam.StartPoint;
                MapPoint endPoint = dam.EndPoint;
                double damHeight = (double)dam.DamHeight;
                double damLength = (double)dam.Length;

                Coordinate3D coord1 = startPoint.Coordinate3D;
                int factor = ((startPoint.Y < endPoint.Y && startPoint.X > endPoint.X)
                              || (startPoint.Y > endPoint.Y && startPoint.X < end-
Point.X)
                              ) ? 1 : -1;
                Coordinate3D coord2 = new Coordinate3D(coord1.X + factor * dam-
Height / damLength * Math.Abs(startPoint.Y - endPoint.Y) //X
                                                       , coord1.Y + damHeight / dam-
Length * Math.Abs(startPoint.X - endPoint.X) //Y
                                                       , coord1.Z - damHeight); //Z
                Coordinate3D coord3 = new Coordinate3D(coord1.X - factor * dam-
Height / damLength * Math.Abs(startPoint.Y - endPoint.Y) //X
```

```
                                                        , coord1.Y - damHeight / dam-
Length * Math.Abs(startPoint.X - endPoint.X) //Y
                                                        , coord1.Z - damHeight); //Z

                //Workaround for Bug in ArcGIS Pro 2.4.1: if values are equal, extru-
sion will fail
                coord2.X += 0.1;
                coord2.Y += 0.1;
                coord3.X += 0.1;
                coord3.Y += 0.1;
                List<Coordinate3D> coords = new List<Coordinate3D>();
                coords.Add(coord1);
                coords.Add(coord2);
                coords.Add(coord3);

                var newPolygon3D = PolygonBuilder.CreatePolygon(coords, SpatialRefer-
ence);
                Coordinate3D coord = new Coordinate3D(endPoint.X - startPoint.X, end-
Point.Y - startPoint.Y, 0.1);
                var multipatch = GeometryEngine.Instance.ConstructMultipatchExtru-
deAlongVector3D(newPolygon3D, coord);
                var attributes2 = new Dictionary<string, object>
                        {
                            { "Shape", multipatch },
                            { "DamID", (long)dam.ObjectID }
                        };
                var createOperation2 = new EditOperation() { Name = "Create mul-
tipatch", SelectNewFeatures = false };
                createOperation2.Create(dam3dLayer, attributes2);
                await createOperation2.ExecuteAsync();

                //add SurfacePolygon to Visualization:
                var queryFilter = new QueryFilter { WhereClause = string.For-
mat("DamID = {0}", dam.ObjectID) };
                var surfaceCursor = reservoirSurfacesLayer.Select(queryFilter).Search();
                if (surfaceCursor.MoveNext())
                {
                    using (Row row = surfaceCursor.Current)
                    {
                        var polygon = (row as Feature).GetShape() as Polygon;
                        attributes2 = new Dictionary<string, object>
                        {
                            { "Shape", polygon },
                            { "DamID", (long)dam.ObjectID }
                        };
                        var createOperationSurface = new EditOperation() { Name = "Cre-
ate surface", SelectNewFeatures = false };
                        createOperationSurface.Create(reservoirVisLayer, attributes2);
                        await createOperationSurface.ExecuteAsync();
                    }
                }
            }
            catch (Exception ex)
            {
                SharedFunctions.Log("Error for 3D Dam with DamID " + dam.Ob-
jectID + ": " + ex.Message);
            }

            SharedFunctions.Log("3D Dam created for Dam " + dam.ObjectID);
        }
        await Project.Current.SaveEditsAsync();
    }
}
```